DELTA MODULATION
TECHNIQUES FOR LOW BIT-RATE
DIGITAL SPEECH ENCODING

by

J.M. IRVINE, B.Sc (Electrical) Engineering
(Cape Town)

Submitted to the University of Cape Town in fulfilment
of the requirements for the degree of Master of Science
in Engineering.

April 1985.
The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.
"Nature, as we often say, makes nothing in vain, and man is the only animal whom she has endowed with the gift of speech. And whereas mere voice is but an indication of pleasure or pain, and is therefore found in other animals, the power of speech is intended to set forth the expedient and inexpedient, and therefore likewise the just and the unjust. And it is a characteristic of man that he alone has any sense of good and evil, of just and unjust, and the like, and the association of living beings who have this sense makes a family and a state."

- Aristotle, Politics.
## CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>i</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACKNOWLEDGEMENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ii</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOMENCLATURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>iii</td>
</tr>
</tbody>
</table>

## 1

### INTRODUCTION

## 2

### DIGITAL SPEECH CODING

<table>
<thead>
<tr>
<th>2.1</th>
<th>Properties of Speech</th>
<th>2-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1</td>
<td>A Model of the Speech Source</td>
<td>2-4</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Characteristics of Speech Signals</td>
<td>2-6</td>
</tr>
<tr>
<td>2.1.2.1</td>
<td>Signal Pass-Band</td>
<td>2-6</td>
</tr>
<tr>
<td>2.1.2.2</td>
<td>Signal Stationarity</td>
<td>2-6</td>
</tr>
<tr>
<td>2.1.2.3</td>
<td>Amplitude Probability Density Function</td>
<td>2-8</td>
</tr>
<tr>
<td>2.1.2.4</td>
<td>Power Spectral Density</td>
<td>2-9</td>
</tr>
<tr>
<td>2.1.2.5</td>
<td>The Autocorrelation Function</td>
<td>2-10</td>
</tr>
<tr>
<td>2.2</td>
<td>Transmission Issues</td>
<td>2-11</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Channel Errors</td>
<td>2-11</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Voice-band Data</td>
<td>2-12</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Variable-rate Systems</td>
<td>2-14</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Tandem Coding</td>
<td>2-14</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Encryption</td>
<td>2-15</td>
</tr>
<tr>
<td>2.3</td>
<td>Digital Speech Coding Techniques</td>
<td>2-15</td>
</tr>
</tbody>
</table>
2.3.1 Waveform Coders
  2.3.1.1 Time Domain Coding
  2.3.1.2 Frequency Domain Coding
2.3.2 Vocoder
  2.3.2.1 Frequency Domain Vocoder
  2.3.2.2 Time Domain Vocoder

3 Speech Codec Design

3.1 Choice of Encoding Technique
3.2 Delta Modulation Algorithms
  3.2.1 Linear Delta Modulation
  3.2.2 Continuously Variable Slope Delta Modulation
  3.2.3 Constant Factor Delta Modulation
  3.2.4 Hybrid Companding Delta Modulation
  3.2.5 Song Voice Adaptive Delta Modulation
  3.2.6 Hybridisation of the Song Voice Adaptive system
  3.2.6.1 Song Hybrid Companding Delta Modulation
  3.2.6.2 Modified Song Hybrid Companding Delta Modulation

4 Hardware Implementation of a Song Voice Adaptive Delta Modulator

4.1 Error Generation and Quantization
4.2 Step-size Generation
4.3 Approximation-signal Generation
4.4 Timing Considerations 4-8
4.5 Network Interfacing 4-10
4.5.1 Serial/Parallel Interface 4-11

5 COMPUTER SIMULATION 5-1
5.1 Interfacing 5-3
5.1.1 Data Input 5-8
5.1.2 Data Output 5-9
5.2 Codec Simulation 5-9
5.3 Measurement 5-10
5.4 Peripheral 5-11
5.4.1 Filtering 5-11
5.4.2 Interpolation 5-12
5.5 Special Considerations 5-14
5.5.1 Signal Range 5-14
5.5.2 Step Size 5-15

6 CODEC PERFORMANCE ANALYSIS 6-1
6.1 Objective Performance Evaluation 6-1
6.1.1 Dynamic Range Test 6-5
6.1.2 Speech Material 6-6
6.2 Subjective Quality Evaluation 6-6
6.2.1 Isopreference Test 6-11
6.2.2 IEEE Recommendations 6-13
6.2.3 Subjective Test Procedure 6-15
6.2.3.1 Intelligibility and Speaker Recognition Test 6-17
6.2.3.2 Isopreference Test

7

**SYSTEM PERFORMANCE**

7.1 Autocorrelation Function of a Speech Signal 7-1

7.2 Objective Results 7-1

7.2.1 Circuit Optimization 7-3

7.2.1.1 Syllabic Gain Factor (\(\omega\)) 7-4

7.2.1.2 Prediction Time Constant (1/\(\beta\)) 7-8

7.2.1.3 Syllabic Companding Period 7-12

7.2.1.4 Minimum Step Size (\(S_e\)) 7-13

7.2.2 Dynamic Ranges of HCDM, SHCDM and MSHCDM Encoders 7-13

7.2.2.1 Variation of the Step Size Range 7-19

7.2.2.2 Variation of Granular SNR 7-19

7.2.3 Cross-correlation Functions 7-22

7.2.4 Using the Feedforward Mode for Syllabic Companding 7-26

7.2.5 Extending the Constant Factor Logic for HCDM 7-27

7.2.6 High Quality Microphone vs Telephone Handset 7-30

7.2.7 Step Size Histograms 7-34

7.3 Subjective Results 7-36

7.3.1 Significance of Results 7-36

7.3.2 Intelligibility Test 7-37

7.3.3 Speaker Recognition 7-39

7.3.4 Isopreference Test 7-41
7.4 Summary

7.4.1 Implementation Considerations

8 PROSPECTIVE FUTURE DEVELOPMENT

8.1 Extending to a Variable-Rate System

8.2 Optimization of the Algorithm Implementation

8.3 Correlating the Results of Objective and Subjective Tests

8.3.1 Inclusion of Network Parameters

REFERENCES

APPENDICES

A Derivation of the equations for Song Voice Adaptive Delta Modulation

B Circuit Diagrams for the SVADM codec and the communications link

C Software listings for the full duplex communications link

D User's guide to the Codec Simulation Package - CODSIM

E Software listings for CODSIM

F Derivation of design equations for FIR filter

G Table for Significance tests (Subjective tests)
ABSTRACT

Two new hybrid companding delta modulators for speech encoding are presented here. These modulators differ from the Hybrid Companding Delta Modulator (HCDM) proposed by Un et al. in that the two new encoders employ Song Voice Adaptation as the basis of the instantaneous compandor, rather than Constant Factor adaptation. A detailed analysis of the performance, both objective and subjective, of these hybrid codecs has been carried out. Results show that overall the two codecs developed as part of this project are better than the HCDM codec. In addition the new codecs offer simpler implementation in digital hardware than the HCDM. A Computer Aided Test (CAT) system has been developed to simplify the design and test processes for speech codecs.
ACKNOWLEDGEMENTS

The author of this work is deeply indebted to Professor Hugh Bradlow for supervising this research project. My thanks also to co-researcher Stephen Hall for his valuable insight and assistance.

A special word of thanks to the Power Systems section of the Department, in particular Keith Hoffman and Professor N. Enslin, for making available the computer system on which the bulk of the work detailed here was carried out.

This research project was funded in part by the Council for Scientific and Industrial Research (CSIR).
NOMENCLATURE

ACF : Autocorrelation Function.
A/D : Analogue-to-Digital.
ADC : Analogue-to-Digital Converter.
ADM : Adaptive Delta Modulation.
APC : Adaptive Predictive Coding.
ATC : Adaptive Transform Coding.
BER (or ber) : Bit Error Rate.

\[ b \] : Error signal - the difference between the input and decoded signals (usually quantized to two levels).

CFDM : Constant Factor Delta Modulation.
CODEC : A word derived from the combination of CODER and DECODER.
COMPAND : A combination of the words COMPRESS and EXPAND.
CVSD : Continuously Variable Slope Delta Modulation.
C(n) : The autocorrelation function.
D/A : Digital-to-Analogue.
DAC : Digital-to-Analogue Converter.
DCT : Discrete Cosine Transform.
DFT : Discrete Fourier Transform.
DM : Delta Modulation.
\[ E( Y_x - X_x ) \] : Mean square of the difference between \( Y_x \) and \( X_x \).
NOMENCLATURE

(cf. \( \langle e^2 \rangle_{av} \)).

\[ E(Y_k; X_{k-1}, X_{k-2}, b_{k-1}, b_{k-2}) : \text{Expected, or estimated, value of } Y_k, \text{ based on } X_{k-1}, \ldots, b_{k-2}. \]

\[ \langle e^2 \rangle_{av} : \text{The average of } e^2 \text{ over all sampling instances.} \]

\( f_c \) : Filter cut-off frequency.

\( \text{FFT} \) : Fast Fourier Transform.

\( \text{FIR} \) : Finite Impulse Response.

\( F(n) \) : This is the value of the continuous function \( F \).

\( f_s \) : Sampling frequency.

\( g(i)T \) : This is equivalent to \( g_i \) (cf. \( V_n \)) - used to represent the discretely sampled variable \( g(i) \) at the \( i^{th} \) sampling instant. The sampling frequency is \( 1/T \).

\( \text{HCDM} \) : Hybrid Companding Delta Modulation.

\( I/O \) : Input/Output.

\( \text{kbps} \) : Kilobits per second.

\( \text{LAN} \) : Local Area Network.

\( \text{LDM} \) : Linear Delta Modulation.

\( \text{LPC} \) : Linear Predictive Coding.

\( \text{LSB} \) : Least Significant Bit.

\( \text{MSB} \) : Most Significant Bit.

\( \text{MSHCDM} \) : Modified Song Hybrid Companding Delta Modulation.

\( \text{PAD} \) : Packet Assembler/Dissassembler.
NOMENCLATURE

PCM : Pulse Code Modulation.
pdf : Probability density function.
RMS : Root Mean Square.
SBC : Sub-Band Coding.
SHCDM : Song Hybrid Companding Delta Modulation.
SNO : Signal-to-Overload Noise Ratio.
SNR : Signal-to-Noise Ratio.
SNRSEG : Segmental Signal-to-Noise Ratio.
SQNR : Signal-to-Quantization Noise Ratio.
SVADM : Song Voice Adaptive Delta Modulation.
TASI : Time Assignment Speech Interpolation.

V_n : This is used to indicate the value of the discretely sampled variable V at the n^th sampling instant.

VOCODER : A combination of the words VOICE and CODER.
X_n : Decoded signal.
Y_n : Input signal.
ρ : Correlation factor (cf. C(n)).
σ : Standard deviation.
CHAPTER 1

INTRODUCTION

The trend in the communications field is to integrate the voice and data services. This has prompted research into developing techniques for achieving low bit rate digital voice communication of an acceptable quality. Low cost digital communication networks have become practically feasible due to advances in the technology for manufacturing integrated circuits. The continued improvement in throughput rates (i.e. processing speeds), via improved VLSI (Very Large Scale Integration) techniques and architectures, have enabled complex signal processing algorithms to be realised as high-speed single-chip modules.

Analogue circuits, which are subject to noise and temperature drift problems, can be replaced by digital signal processing methods which are inherently less sensitive to these factors. When considered against the background of increasing research into ways and means of reducing the channel bandwidth (for signal transmission) the application of these digital processing techniques becomes even more significant.

Whereas previously the methods for achieving satisfactory quality using low bit-rate digital voice communications would have been extremely cumbersome to implement, the new generation of digital integrated circuits offer an attractive means to solving this problem.
CHAPTER 1: INTRODUCTION

One of the most important parameters affecting the efficiency of the processing algorithm is the signal type to be transmitted. For the purposes of this research the message ensemble was assumed to contain only human speech sounds. Although in terms of overall network performance, non-speech signals (such as voice-band data signals) may be significant in their effect on the network, only speech signals were considered when designing for efficient communication. The difficulty, in terms of achieving maximum transmission efficiency, is to design a system that transmits only the perceptually significant information (speech signals contain redundant information which is not readily perceived by the average listener). Two means of achieving this would appear to be, firstly, the incorporation into the transmission system of as many as possible of the characteristics of the speech signal, or secondly, the discarding of that information which is not perceptually relevant. The first method is based largely on the characterization of the sound source while the second stems from the properties of the sound perception mechanism.

The nature of the speech production source (i.e., the human speaker) makes speech unique in terms of its characteristics. The singular nature of speech has allowed various other factors besides the production and perception characteristics to be used as design parameters. Successful digital speech communication could be
CHAPTER 1: INTRODUCTION

accomplished by means of the analysis of both the physical aspects of the production/perception processes and the semantic/psychological factors.

Initial work involved in digitizing speech communication channels centred around implementing PCM (Pulse Code Modulation) systems. Despite the ever-improving performance levels of alternative systems (such as delta modulation) the large capital investment in PCM has precluded the use of these alternatives. However, the local area networks (LAN's), or local subscriber networks offer significant scope for implementing alternative schemes which can be designed to be more signal-specific and more economical in terms of bandwidth utilisation than PCM.

The use of digital speech encoding techniques means that a single LAN channel can serve both the data and speech requirements of the user. However, in offering the user the speech communication facility it is envisaged that a certain reduction in the subjective speech quality is to be expected. The main object of this research has been to develop a digital speech encoder capable of operating at relatively low information transmission rates (baud rates) while at the same time providing acceptable levels of speech quality.
CHAPTER 1: INTRODUCTION

In this work a digital encoder is presented which has been found to offer an improvement in speech quality when compared with previously developed techniques\(^{42}\). The significance of the encoder design lies not only in the improved performance, but also in the fact that its design is simpler than alternative systems. The latter point is particularly important in that the eventual aim of the research project is to produce a low-cost digital speech codec that can be used in a commercial communications network.

A simulation system has been developed to enable the performance evaluation and optimization of the various speech codecs under consideration to be performed by means of computer modelling techniques. In other words, the hardware designs have been converted to software programs for the purposes of evaluation and optimization.
CHAPTER 2

DIGITAL SPEECH CODING

There are two major areas affecting the design of a digital speech encoder. These are firstly, the statistical properties of the speech signal and, secondly, the characteristics of the transmission system (ie. the communications network). The former is going to have the bulk of the effect on the codec design, but the latter cannot be totally ignored and classified as a network design issue.

2.1 Properties of Speech.

One of our main objectives is, quite simply, to reduce the information transfer rate required for encoded speech data. The characteristics of the speech production and perception processes enable this to be done.

- The first, and perhaps the most obvious, characteristic is the spasmodic nature of the production process - we tend to speak in bursts rather than continuously. Not only are there significant pauses between sentences and phrases, but also between words and even syllables. Characteristically interactive conversational speech contains silences that accommodate over 50% of the elapsed time of the conversation[11]. Hence, a significant band-width saving could be realised if these silences were not transmitted.

- Speech signals contain significant redundancies[46]. This is
partly because of the physical mechanism of the vocal tract and partly due to the structure of language. Removal of the redundant information from the signal would appear to be one of the methods for reducing the information rate for transmitted speech. However, reduction of the redundancies also leads to a reduction of the speaker recognition capabilities and a general reduction of the audio quality as it is these apparently useless redundancies which provide the listener with information regarding the speaker's identity, manner and personality. In essence a voice message can be split into two distinct elements - firstly, the content of the message (ie. what was said) and secondly, the peripheral information (ie. who said it, how they said it, etc.). The content of the message, or the intelligence, appears largely in the spectral envelope of the signal. Hence, although the redundant information can be removed without destroying the message content, its removal at the encoder only becomes satisfactory if, at the decoder, it can be recovered from the other sections of the signal.

The perception of the speech signal is initiated by the human ear in effect performing a crude Fourier transform on the incoming acoustic signal. The presence of high energy signals in the lower regions of the spectrum tend to reduce the sensitivity of the ear to the frequency components in the upper
CHAPTER 2: DIGITAL SPEECH CODING

reaches of the spectrum and hence, noise or distortion added to the signal will not be detected if it is below some threshold. This phenomenon is known as "auditory masking" and it is this that makes noise more perceptable in the presence of high frequency signals than low frequency signals.

The addition of noise or distortion to a speech signal (by an encoder) is not necessarily disastrous. This is due to the ability of the human brain to extract the pertinent information from the received signal. Owing to the stored vocabulary and its semantic (i.e., meaning-related) faculty the brain is able to interpolate the information missing from the speech signal.

This last characteristic is however not sufficiently well understood to be exploited in reducing the bandwidth of speech transmissions. It is rather the first three properties which are used in increasing the speech transmission channel efficiency.

The nature of the problem was stated rather succinctly by Flanagan:

"Despite the equivocal aspects surrounding estimates of human channel capacity and speech information rates, it is clear that a mismatch exists between the capacity of the conventional voice channel and the information rate of the
source feeding it. One approach toward improving the match is to incorporate into the transmission system as many as possible of the constraints characterising the production and perception of speech. This information, built into the communication link, is information that need not be transmitted... The nature of the incorporated constraints therefore influences the form of the coding for the speech information.

2.1.1 A Model of the Speech Source.

There are basically three types of sounds generated by the vocal system, namely "Voiced" sounds, "Fricatives" (or "Unvoiced" sounds) and "Stops".

- Voiced sounds: these are produced by a nearly periodic series of pulses generated by the vocal chords which control the flow of air from the lungs to the pharynx.

- Fricatives: more commonly referred to as "unvoiced" sounds, these are the result of air turbulence at constrictions in the vocal tract and give rise to noise-like sounds.

- Stops: these are essentially an acoustical transient resulting from an abrupt release of pressure from behind an occlusion. An example of this would be the 't' in 'tea' where the tongue is used to block the flow
of air resulting in a pressure build-up which is subsequently abruptly released.

Because stops can be categorized as either voiced ('b' as in 'boat') or unvoiced ('p' as in 'purse') it is common to differentiate between two sound sources (i.e. voiced or unvoiced) rather than three (cf. fig. 2.1).

From the model shown in fig. 2.1 the sound source excitation is either noise-like (for unvoiced segments) or pulse-like (for voiced segments). This source signal is then passed through a time-varying acoustic filter (the vocal tract) to produce the speech output. This model is a first-order approximation that assumes that the two sound sources are linearly separable, and ignores the possible interaction of the sound source with the

Fig. 2.1: Model of the Speech Source.
vocal tract and the modulating effect of the nasal cavity, which is a second-order effect \cite{9}.

This linear first-order model is going to limit the performance of the coding schemes based on it. However, its main advantage is its relative simplicity which lends itself to widespread use (refer to the sections on time-domain encoders).

2.1.2 Characteristics of Speech Signals.

2.1.2.1 Signal Pass-Band.

In order to digitize the speech signal it is necessary to bandlimit it to enable time-sampling at a practical rate. For communications systems the signals are typically bandlimited between 300 and 3300 Hz. The Nyquist condition states that the minimum sampling rate must be twice the highest frequency in the signal pass-band. If the sampling frequency is below this rate then aliasing errors are introduced.

2.1.2.2 Signal Stationarity.

A rather obvious feature of speech is the difference between two speakers' voices - the spectral content of speech signals varies from speaker to speaker (fig. 2.2). The vocal tract has a number of resonant frequencies which tend to accentuate bands of frequencies centred about these resonances, known as
"formants". Typically, there are four formants in the speech spectrum. However, not only do these formants vary from speaker to speaker, but also with time. This is in fact true for the entire spectral image of speech and not only the formants. This gives rise to the non-stationary characteristics of speech - both the amplitude and the harmonic content of the signal are changing with time.

Speech signals demonstrate both long-time (0.5 sec) and short-time (20 - 40 msec.) characteristics and it is when viewed over the 'long'
periods that the non-stationarity is most evident whereas over the 'short' period the signal is locally stationary.

2.1.2.3 Amplitude Probability Density Function.

Speech signals typically have a large dynamic range, in the region of 40 dB. As mentioned previously, a large percentage of interactive speech communication time is occupied by silences and hence it is to be expected that the probability of near-zero signal amplitudes would be high. In fact, the probability density function (pdf) of speech is characterized by a high probability of zero or near zero amplitudes which then monotonically decrease towards the very high amplitudes. Speech signals are not random and hence the amplitude pdf is not Gaussian but rather (for the long-time case) the sum of Gaussian and Laplacian pdf's.

Because voiced speech segments have a higher probability of occurrence (than unvoiced segments) the overall quality of coded speech is going to be determined in part by how well the voiced segments are coded. Added to this is the fact that voiced sounds are more susceptible (from a listener's point of view) to the addition of granular noise (defined later in the section on delta modulation) than unvoiced sounds.
2.1.2.4 Power Spectral Density (PSD).

Referring back to fig. 2.2 it can be seen that the high frequency components do not contribute very much to the total energy of the speech signal. In fact, the major portion of the signal energy is confined to the first formant. That is not to say that the high frequency components are insignificant to the information content of the signal. On the contrary, they contain important information and must therefore be adequately coded.

The characterization of the signal according to its harmonic content is given by the PSD and is usually evaluated over a long speech segment. One of the reasons for doing this is that the short-time spectrum could be misleading in that a particular sound may exhibit spectral properties which would result in the poor performance of a particular coder. On the other hand, the long-time spectrum (i.e., the average of a series of short-time spectra) could exhibit properties which might be deemed to be favourable for a particular coder. This average measure is likely to be more useful because it is the average performance of a coder that is important to the general user. In other words, the long-time spectrum is a measure of the spectral content of speech signals as opposed to the short-time spectra of specific sounds.
2.1.2.5 The Autocorrelation Function (ACF).

The ACF, denoted by $C(n)$, is an indication of the correlation between two speech samples separated by 'n' sampling periods. $C(0)$ is the correlation of one sample with itself and is normalised to unity (ie. $C(0) = 1$). All other values of $C(n)$ are then normalised relative to $C(0)$. The ACF, when taken over a speech segment, will give a measure of the predictability of the speech samples (eg. a value for $C(n)$ of 0.9 would suggest that the subsequent samples could be predicted from the previous ones, while a value of 0.1 would indicate a low degree of predictability, which one would expect for random signals). The correlation between adjacent samples, $C(1)$, for speech has a typical average value (for Nyquist sampled speech) of $0.9^{43}$. The significance of this is that the variance of the difference between adjacent samples of speech is less than the variance of the samples themselves. The relevance of this to low bit rate encoding will be discussed later.

At this stage it is worth mentioning that the ACF and the PSD form a Fourier Transform pair and hence, a coding scheme exploiting the one is essentially equivalent to a scheme exploiting the other, the only difference being that the one works in the time domain and the other in the frequency domain.
2.2 Transmission Issues.

The digital encoder is just one part of a communications system and its design cannot be considered in isolation from the transmission network. Two of the main considerations here are the possible need to transmit non-speech signals (i.e., signals other than speech occupying the same spectral band) and the probability of channel errors.

2.2.1 Channel Errors.

The presence of high energy noise fields in the vicinity of the transmission network is likely to induce errors in the digital signal which would result in an erroneous decoder output. Hence, for a practical system it is important that the decoder is capable of recovering from transmission errors. Most speech coding methods can tolerate bit error rates (BER) of the order of $10^{-3}$ (i.e., one error in 1000 bits) and an order of magnitude gain (i.e., BER of $10^{-2}$) can be achieved by implementing 'robust' coders that utilise built-in error protection techniques\(^4\). This error protection can take the form of an explicit transmission of code responsible for protecting the signal information (i.e., extra data is added to the coded signal to enable the detection of transmission errors by the decoder). Alternatively, using 'leaky' adaption logic, the effects of transmission errors will be 'leaked' out and after a predetermined length of time the effect
of the error will be negligible. The advantage of the second technique is that no extra information need be transmitted, thereby conserving channel bandwidth.

2.2.2 Voice-band Data.

Voice-band data signals are analogue signals that have been created from digital data so as to enable their transmission over an analogue telephone channel. Although they have been modulated into the speech band their statistical properties are considerably different to speech signals (as evidenced by fig. 2.4), and hence it is unlikely that an encoder designed for efficient digitization of speech will be anything but sub-optimal for voice-band data.

Obviously if the transmission channels were all digital voice-band data signals would not exist as it is pointless modulating the digital data into analogue form and then reconverting it to a digital form. However, during the current transition from analogue to digital communications equipment it is common for the communications network to include both analogue and digital channels.

Most voice-band data signals have data rates up to 4.8 kbps compared to the operating bit rates of speech waveform encoders which are in the 16 to 64 kbps range\(^{25,26}\). Hence, it would
appear inefficient to use a waveform coder for digitizing the voice-band data as opposed to simply demodulating and multiplexing it with other data. For example, demodulating and multiplexing 4.8 kbps data is nearly 7 times more efficient than using a waveform encoder operating at 32 kbps. However, the actual modulation/demodulation process is dependent on the data rate used and the demodulator must be designed for a particular type of data signal. Waveform coders on the other hand are considerably more versatile, being able to handle a range of voice-band data signals as well as facsimile and speech.

![Autocorrelation functions of speech and voice-band data.](image)

If it is desirable to have optimum performance for both speech and voice-band data it might be necessary to use separate coders for each. On the other hand, a 'compromise design' might yield satisfactory performance for both speech and data.
2.2.3 Variable-rate Systems

There are communications systems that exploit the non-stationary intermittent nature of speech while others adapt to the varying demand of the source (not everybody wants to speak at the same time). Examples of these two are TASI (Time Assignment Speech Interpolation) and packet switching networks respectively. The channels for these systems do not operate in a fixed bit-rate mode and hence the encoder/decoder design would have to incorporate some form of interfacing between the codec and the channel.

2.2.4 Tandem Coding

Depending on the communications link it may be necessary to serially utilise different coder stages\(^9\). For example, fig. 2.5 shows a link involving one type of encoder operating at 16 kbps interfacing with a second type operating at 2.4 kbps. It is

![Tandem connection of codecs operating at different sampling rates.](image)

Fig. 2.5: Tandem connection of codecs operating at different sampling rates.
likely that the use of such tandem connections will result in performance degradation with most of the losses occurring in the first stage. The subsequent losses limit the total number of stages that can be satisfactorily used.

2.2.5 Encryption

Two methods of accomplishing digital encryption include masking the data bit stream with a pseudorandom binary noise sequence (known by the intended receiver), and permutation of the bit positions within a block of data. Permutation retains a higher residual intelligibility than the masking technique and the effectiveness of the scheme used depends on the type of encoder used. Hence, if a certain type of encryption is already in use it may dictate the types of voice encoders that could be satisfactorily integrated into the communications network.

2.3 Digital Speech Coding Techniques

Broadly speaking, speech coders can be divided into one of two categories - waveform coders, or source coders. Source coders, generically referred to as 'vocoders' (from the words 'voice coders'), rely for their operation on a priori information on how the signal was generated at the source. This can be achieved by means of a parametric model such as the one shown in fig. 2.1. The second category, waveform coders, attempt to copy the signal
waveform and in general they are signal- (and hence source-) independent, but can be optimized for a specific signal according to the statistical characterization of the signal waveform.

There are coders which tend to make the differentiation between waveform coders and vocoders less distinct. These coders (e.g. Linear Predictive Coders) exhibit definite vocoder characteristics in the form of a central parametric model of the speech source and yet are classified as waveform coders because they work by trying to produce a facsimile copy of the input signal.

In the subsequent sections the coders have been grouped according to their overall operating characteristics - do they parameterize the signal (Vocoders), or do they attempt to produce a copy of the signal waveform (Waveform coders) ?

2.3.1 Waveform Coders

2.3.1.1 Time Domain Coding

(a) Pulse Code Modulation (PCM):

Probably the simplest form of analogue-to-digital conversion, PCM entails sampling the signal in time (at the Nyquist rate or faster) and quantizing it to one of a set of discrete amplitudes (i.e. the sampled value is rounded off by the quantizer to one of the values in the set). The number of discrete amplitude levels
CHAPTER 2: DIGITAL SPEECH CODING

is determined by the number of bits used in the quantizer (e.g. a 12-bit quantizer yields $2^{12} = 4096$ levels). The error introduced by the rounding process (the quantization error) depends on the difference between amplitude levels. In linear PCM the quantizer step sizes are all equal and no data compression is possible because none of the characteristics of speech are exploited.

By making the quantization step size vary logarithmically over the range of the signal amplitudes (as in non-linear PCM) it is possible to exploit the redundancy inherent in the amplitude pdf. Linear PCM assumes that all quantizer levels are utilised in an equitable fashion. However, the amplitude pdf of speech suggests that by allocating more levels to the highly probable low amplitude signals and less to the more infrequent high amplitudes would yield better performance. This is because subjectively the quality of coding is likely to be dependent on the quality of the coding of the high-probability signals. For a given encoding quality non-linear PCM can operate at reduced bit rates compared to linear PCM (e.g. at a SNR of 35 dB a linear PCM coder requires a 12-bit quantizer whereas a non-linear PCM coder requires a 7-bit log-quantizer).

The long-time amplitude pdf as used in non-linear PCM is a static characteristic. By taking the dynamic speech amplitude variations
into account an increase in the dynamic range of the input signal can be achieved. In adaptive PCM the quantizer step size is varied according to the RMS level of the signal. If the step size is determined from the quantizer output, rather than from the input, the step size information need not be explicitly transmitted to the receiver as it can be recovered from the transmitted signal.

PCM systems are however susceptible to significant performance degradation in the presence of channel errors. The bits in a sample have different weightings and as a result the heavier the weighting of the bit that is corrupted by the channel, the greater is the effect (subjectively) on the receiver output. Non-linear PCM is, at present, the most widely used voice coding technique for telephone networks. The advantage of PCM is that although it may not be optimal for any one particular signal (most notably in the case of linear PCM) it is capable of coding any type of signal (voice-band data, etc.).

(b) Differential Pulse Code Modulation (DPCM):
The difference between adjacent samples, \( D = x_n - x_{n-1} \), has a variance which is less than the variance of the signal itself as long as the correlation between samples is greater than 0.5\(^{0.5} \). The implication of this is that it is possible to quantize this
difference with fewer quantization levels (and hence, bits) than is possible when quantizing the signal itself, and yet maintain the same signal-to-noise level. Essentially the input signal is being discretely differentiated with respect to time, the derivatives are being quantized, coded and transmitted, and at the receiver the input difference samples are integrated to yield the original signal. By coding only the sample-to-sample change in signal level the redundancies have been reduced because, by coding that part of the signal that hasn't changed is a waste of transmission bandwidth as that information is already at the receiver.

As for PCM, an adaptive scheme can be utilised to enable the differential pulse code modulator to respond to the dynamic

![Block diagram of a DPCM system](image)

Fig. 2.6: Block diagram of a DPCM system.
variations of the speech signal amplitudes. Because the transmitted code does not represent the signal itself a different scheme has to be employed in determining the RMS signal value.

(c) Predictive Coding:

A DPCM system is, in effect, a predictive coder. Referring to fig. 2.7 the predictor \( P \) would be the integrators used in the DPCM system - they predict, based on the previous input sample and their state during the previous sampling time, the next sample value.

If the quantizer input is represented as:

\[
D_r = x_r - a \cdot x_{r-1}
\]

then the variance of \( D_r \) is a minimum for \( a = C(1) \) (the correlation between adjacent samples). In the DPCM case the value \( a \cdot x_{r-1} \) is the output of the prediction filter and is the first order prediction of \( x_r \). The general predictive coder would essentially be a DPCM of order \( p \). In other words, the \( p^{th} \) order prediction of \( x_r \) would be derived from the \( p \) previous samples, and not simply \( x_{r-1} \):

\[
x'_r = \sum_{n=1}^{p} a_n \cdot x_{r-n}
\]

In order to control the accumulation of quantizing errors at the receiver, the value \( x'_{r-n} \) (cf. fig. 2.7) is used instead of \( x_{r-n} \).
This technique is known as Linear Predictive Coding (LPC). The prediction coefficients, \( a_n \), are chosen to maximize the long-time SNR. The error (or noise) is given by:

\[ e = x_r - x'. \]

\( \langle e^2 \rangle_v \) is the average of \( e^2 \) over all sampling instances. The prediction coefficients for maximum SNR are then found by setting the partial derivative of \( \langle e^2 \rangle_v \) with respect to each \( a_n \), to zero. The prediction coefficients are then found by solving:

\[ A \cdot T = V \]

where:

- \( A \) - the vector of optimal prediction coefficients,
- \( T \) - a symmetrical \( p \times p \) matrix of the autocorrelation function values,
- \( V \) - a vector of autocorrelation function values\(^{14,10}\).

The degree of improvement of this system over linear PCM tends to saturate at a value for \( p \) of about 2 or 3\(^{14,3}\).

---

**Fig. 2.7** : Block diagram of a predictive coder.
The problem involved with this technique is that, being static (i.e. the values for a(n) are optimized once only) the coder's performance is subject to the vagaries of speech non-stationarity and, as a result, may work well for one speech segment or speaker, but not for another.

(d) Adaptive Predictive Coding (APC):

APC basically involves periodically updating the LPC prediction coefficients to improve the coding efficiency for the general speech input. A common technique utilizes two predictors - one is based on the short-time spectral envelope (i.e. formant structure) while the other is based on the quasi-periodic nature of voiced speech which gives rise to the spectral fine structure as shown in fig. 2.3a.

The periodic updating of the vector A can be achieved by storing a certain length of speech, calculating the auto-correlation matrix T and vector V and then adjusting A accordingly. The net result is that instead of the long-time correlation, the short-time correlation is used to reduce the data rate. This reduction is aided by the fact that the coding of the prediction coefficients can be quite coarse and they can be updated at a relatively slow rate.
However, if it is desirable to reduce the data rate even further the receiver can use the incoming data to update the prediction coefficients, thereby eliminating the need to transmit these coefficients, but at the expense of extra processing.

The auditory masking properties of speech would suggest that much of the predictive noise originates from the frequency areas where the signal level is low (e.g. outside the formant regions). In other words, subjectively the SNR in the formant regions would be significantly better than other areas of the speech spectrum. By adding an extra feedback network around the quantizer it is possible to shape the quantizing noise spectrum to enhance the subjective quality of the codec[4].

As mentioned previously, predictive coding techniques can straddle the boundary between waveform coders and vocoders. If, instead of prediction filters, signal source models are used and the information regarding the voiced/unvoiced characteristics and the pitch information are transmitted (instead of the prediction coefficients), then it would be possible to generate the prediction filter excitation locally at the receiver based on the above data. This type of scheme is neither strictly waveform coding nor purely vocoding (to be discussed later) but a hybrid of the two.
Although the processing overhead is quite high (the solution of a set of simultaneous linear equations involves a large amount of processing in order to determine the prediction coefficients) predictive coding is an important encoding technique at very low transmission data rates (2400 bps or lower) and at least one real-time implementation has been achieved using commercially available components\cite{12}.

(e) Delta Modulation (DM):
In DPCM the signal correlations were used to reduce the number of levels required to quantize the prediction error. If this is taken to the limit case of using a two-level quantizer (ie. the prediction error is either positive or negative) the input signal needs to be oversampled (sampled at more than the Nyquist rate) in order to increase the adjacent sample correlation. This then is Delta Modulation - the 1-bit version of DPCM.

In its simplest form (Linear DM - LDM) the coder approximates the input signal by a ramp staircase with equally sized steps (cf. fig. 2.8). At sample time 'n' the polarity of the error (the difference between the input sample $x(n)$ and the approximation sample $y(n-1)$) is quantized and transmitted. The staircase function is then updated in the direction of the error.
However, although DM is suitable for speech coding, the simple scheme described above is totally inadequate. Firstly, when the coder is 'hunting' about a constant input signal level so-called granular noise is produced. This is most likely to occur during silences in speech segments and could prove to be irritating to the listener. Secondly, because the slope of the ramp staircase is fixed, if the slope of the input signal is greater than that of the staircase (e.g. during the transients produced by 'stops') then the coder produces 'slope overload' distortion - i.e. once the frequency-amplitude product (of the input) exceeds a certain threshold the coder goes into slope overload. Obviously, as the

![Diagram showing slope overload and granular noise](image)

- The shaded area represents 'slope overload' noise.
- The remaining section represents the source of the so-called 'granular' noise.

**Fig. 2.8**: Waveforms produced by a Linear Delta Modulator.

signal frequency increases so the maximum possible signal amplitude, before the onset of slope overload, decreases. What this means is that the dynamic range of LDM tends to decrease with
frequency in the same manner as the PSD curve. Granular noise can be reduced by reducing the step size whereas slope overload noise is reduced by increasing the step size and sampling rate.

Again, as with the schemes discussed previously, the static non-adaptive technique tends to fail due to the characteristics of speech.

In order to improve the dynamic range of the DM and to minimise the total noise power adaptive techniques are used - hence 'Adaptive Delta Modulation' (ADM). A number of companding schemes have been developed\(^{13-17}\), all with one thing in common - the modification of the step size is determined from the encoder output bit stream. The general adaptation rule is given by:

\[
S_r = f( S_{r-1}, b_r, b_{r-1}, \ldots, b_{r-n} )
\]

where

\(S_r\) is the new step size, and

\(b_r\) is the sign of the error.

Different schemes use different memory lengths and basically there are two categories of companding algorithms. Step size adaptation can be at either the syllabic rate or the instantaneous rate.

- Syllabic companding uses a short segment (typically about 5 msec in length) of the speech to adjust the step size, thereby utilising the short-time quasi-stationarity charac-
CHAPTER 2: DIGITAL SPEECH CODING

teristics.

- Instantaneous companding on the other hand updates the step size at each sampling instant.

The main problem with syllabic companding is its inability to handle the transient-like changes present in speech. While instantaneous companding can respond to the transients its fixed basic step size means that, due to the large variation in dynamic range, it may work well for one segment of speech but not another\(^\text{17}\). The performance of an instantaneous compander is also likely to be degraded during silent or steady-state portions of speech. It would seem then that some form of combination of instantaneous and syllabic companding schemes could also be used as a compromise on the shortcomings of the individual schemes.

The development of a hybrid delta modulator (HCDM) is a fairly recent development\(^\text{17}\) and is claimed to be better (in terms of SNR and dynamic range) than the previous methods\(^\text{18}\).

Earlier we saw how DPCM was extended from a first order system to a higher order by extending the prediction parameters to include previous values. For DM systems this is equivalent to incorporating the higher order differences and more stages of integration. Double integration includes two integrating stages and it has been found that combining single and double integration, the second
Fig. 2.9: DM waveforms obtained using different types of integrator combinations.

Integrator being progressively engaged above 2 kHz, is an improvement on pure single or double integration (the latter tends to suffer from stability problems). This combination is referred to as 'mixed' integration and Fig. 2.9 shows the types of
waveforms obtained from the different delta modulators.

Whichever form of prediction is used (single, double, etc.) it is still possible to include some form of companding as the relative effects are complementary.\(^3\,7\)

Although speech is often limited to the 300 - 3300 Hz band, the actual bandwidth of the signal varies with time (fig. 2.10) and does not always fully occupy the allotted band - generally only the lower frequencies are occupied during voiced sounds while the signal is 'high pass' during unvoiced sounds. Commonly a non-adaptive low-pass filter is used on the output to remove coder noise outside the speech pass-band. If instead an adaptive filter is used (i.e. one that tracks the actual bandwidth of the speech),
2.3.1.2 Frequency Domain Coding.

Instead of treating speech as a single-band signal, as in time-domain coding, the frequency band could be divided into a number of sections with each being coded separately. The advantage of this is that the encoding accuracy can be determined according to the activity of the separate spectral bands - e.g. bands with very little or no energy need not be coded whereas bands with a high energy signal can be coded accurately. This is a different approach to the problem mentioned at the end of the last section (i.e. adaptively filtering the DM output). The main distinguishing feature between the different encoding algorithms is usually, as in the time domain case, the degree of prediction used.

(a) Sub-Band Coding (SBC):

This technique involves dividing the speech band into a number of sub-bands (by a bank of band-pass filters), translating each band down to zero frequency, and then coding each band using adaptive PCM. At the receiver the process is reversed with each band being modulated back up to its original position and then added to the rest to yield the original signal. The advantage of this method is that each band can be quantized and coded according to the
perceptual properties of that band - bands sensitive to quantization noise (e.g. components with low signal energy) can be allocated lower quantizer step sizes to reduce the quantization noise while bands relatively insensitive to noise can be quantized coarsely.

The pitch and formant information in the lower frequency bands can be accurately encoded by allocating a large number of bits to these bands while the unvoiced noise-like sounds in the higher frequency bands can tolerate fewer quantizer bits\(^4\). In this way the masking of one frequency band by the noise in another is prevented.

(b) Adaptive Transform Coding (ATC):

This technique is considerably more complex than SBC as it involves dividing the input signal into time segments and then transforming these windowed segments into the frequency domain. Each segment is then represented by a set of transform coefficients which are separately quantized and transmitted. At the receiver these coefficients are then inverse transformed to yield the original signal segment. The transform can be achieved by either a Fast Fourier Transform (FFT) or a Discrete Cosine Transform (DCT) which has been found to be particularly well suited to speech\(^4\).

The performance of this encoder can be enhanced by employing a
dynamically adaptive scheme, based on the spectral properties of speech, to the transform process. The use of a bit allocation algorithm (e.g., as described in [4]) can help in controlling the noise spectrum. Because the spectral levels of a speech segment are not known a priori they must be estimated. The information regarding the dynamic properties of the speech spectra is referred to as 'side information' (Fig. 2.11). Depending on the added complexity of the system, the ATC method can be expected to operate down to about 8 kbps.

![Block diagram of an Adaptive Transform Coder](image)

Fig. 2.11: Block diagram of an Adaptive Transform Coder.

2.3.2 Vocoders.

Source coders are, in general, used for special purpose commun-
cations (e.g. where very low bit-rates are required) and are not suitable for use in commercial telephone networks. If one looks at fig. 2.12 this can be clearly seen - telephone networks fall into the 'Toll Quality' category. Vocoder depend for their operation on the source model (shown earlier in fig. 2.1) where the sound generating system is divided into a sound source and a modulator. In the vocoder itself the information to be coded includes the switching information (between the pulse generator and the random noise generator), amplitude signal (for intensity of excitation of the two sources), and the pitch information (specified by the pitch period of the pulse generator). There are a number of possible parametric models that can be used by the vocoder. These vary from the use of linear prediction coefficients describing the spectral envelope (LPC vocoder), to the specification of major spectral resonances (formant vocoder) and a description of the short-time amplitude spectrum for specific frequencies (channel vocoder). Other methods include the specification of samples of the short-time
auto-correlation function (the autocorrelation vocoder), or the use of coefficients of a set of orthonormal functions that approximate the speech waveform (orthogonal function vocoder).

2.3.2.1 Frequency Domain Vocoding.

As was mentioned previously, the human ear performs Fourier analysis of the speech signal. In other words, the perceptual analysis of speech sounds is based on their spectra. It would thus make sense to digitally characterize the signal according to its spectrum. In this way it is possible to drastically reduce the total bandwidth required for transmitting speech when compared to transmitting the signal directly. It has been found that the spectrum can be adequately described by 16 frequency values which are updated every 20 milliseconds.\footnote{From the Nyquist sampling theorem each value would then require a bandwidth of \( \frac{1}{2 \times 20 \text{ msec}} = 25 \text{ Hz.} \) The 16 channels would then need \( 16 \times 25 = 400 \text{ Hz.} \) On the other hand, direct transmission of the signal would need about 3500 Hz, or nearly 9 times the bandwidth. If the pitch information, and the voiced/unvoiced data is included this would typically require an extra 300 Hz bandwidth, bringing the total for the vocoder to 700 Hz, which is still considerably less than the direct method.}

The channel vocoder consists of a vocoder analyzer at the trans-
miter (which measures the 16 frequency values, the pitch information and determines the voiced/unvoiced state), and a vocoder synthesizer at the receiver. At the receiver the 16 received channel signals control the frequency response of a time varying filter, in that way reproducing the spectral envelope of the original signal. The filter is excited with a signal which is either white noise or a pulse-train whose period is obtained from the pitch information signal.

A formant vocoder is essentially the same except that instead of 16 frequency values, only the formant frequencies are analyzed. However, the techniques for evaluating the formant data are considerably more complex than those used in the channel vocoder.

2.3.2.2 Time Domain Vocoder

(a) Orthogonal Expansion Vocoders:
It is possible to expand the signal waveform, or the power (or amplitude) spectrum, or the autocorrelation function into a series of orthogonal functions. Waveform expansions can be done via Laguerre polynomials, while power spectrum expansions are best done by the eigenfunctions of the autocovariance matrix - the so-called Karhunen-Loeve expansions. Another possibility involves expanding the logarithm of the power spectrum into a cosine series to yield the 'cepstrum' (and hence the cepstral vocoder). The
advantage of the cepstrum is that no additional analysis is needed to measure the voice pitch\(^{43}\).

(b) Autocorrelation Vocoders:

For this type of vocoder the telephone-bandwidth speech is sampled at about 8 kHz - i.e. samples are 125 \(\mu\)sec apart. To ensure sufficient spectral resolution a large number of samples would have to be used. Compared to the spectral samples the autocorrelation samples need about twice the quantization resolution (7-8 bits/sample compared to 3-4 bits/sample\(^{43}\)). By synthesizing the speech signal based on the autocorrelation function yields a signal whose spectrum is the squared version of the original signal\(^{43}\) thus necessitating spectral square-rooting.

(c) LPC Vocoder:

As with the orthogonal expansion vocoder the LPC vocoder is not purely a time-domain vocoder as it exploits characteristics of the spectral properties of speech as well, and could equally have been classified with the frequency-domain vocoders. The LPC vocoder is basically an APC system except that the prediction residual has been replaced by the pulse and noise sources.

An essential part of all vocoders is the pitch measurement sub-system. This is also one of the more difficult sections of the
CHAPTER 2 : DIGITAL SPEECH CODING

analysis of speech, particularly telephone speech which has had its fundamental frequency removed. As yet no completely satisfactory method has been developed owing mainly to the shortcoming of simplistically separating the voiced and unvoiced sounds. A product of this problem has been the emergence of hybrid techniques where part of the spectrum is waveform coded while the rest is vocoded. A major drawback of vocoders is their sensitivity to different speakers, and the synthetic (automaton-like) nature of the output speech, resulting in a substantial degradation of the speaker recognizability. This must however be weighed against the substantial economies in the transmission bandwidth possible with vocoders.
CHAPTER 3

SPEECH CODEC DESIGN

3.1 Choice of Encoding Technique.

Essentially there are three stages to selecting the codec:

(i) the choice between waveform coding or vocoding techniques,
(ii) time-domain or frequency-domain coding,
(iii) the final choice involves the specific technique to be used.

Based on the present state of the art waveform coders are more attractive for commercial communications links than vocoders. This is because of, as was mentioned previously, firstly, the complex techniques involved in overcoming the pitch measurement problems, and secondly, the sensitivity of vocoders to different speech segments and speakers. The synthetic quality of the output speech is also not considered to be attractive for a general purpose link. Hence, the use of vocoding techniques has not been considered in this work.

The frequency domain techniques vary from, on the one hand, a relatively simple concept requiring a considerable amount of hardware, to a much more complex scheme with high overheads in terms of both hardware and processing. At the lower end of the complexity scale SBC requires a bank of band-pass filters with relatively high selectivity. The use of either analogue or digital filters would yield circuits of considerable complexity with a large amount of hardware required. The number of filters
required for ATC is less than SBC but the processing required is considerably more complex. The requirements that the codec used be implementable as a low cost, low complexity integrated circuit tends to gravitate against these techniques.

Primarily we are attempting to improve the transmission channel efficiency by reducing the bit rate while maintaining an acceptable level of speech quality. This immediately eliminates PCM and non-linear PCM as the band-width savings possible with these techniques are relatively minimal compared to alternative techniques. The choice then is essentially between the predictive coders (LPC, APC, etc.) and delta modulation. The complexity and costs of the predictive coders are too restrictive. The implementation of an LPC system requires, for example, three 16-bit microcomputers, another 8-bit microcomputer plus other peripheral elements (memory, etc.) and an approximate estimate of the cost of such a device (in production quantities) is in the region of $1000.\textsuperscript{12}

It was decided then to investigate the use of delta modulation as a means of providing a low cost, low complexity digital speech encoder capable of operating at subjectively acceptable performance levels at low bit rates.
3.2 **Delta Modulation Algorithms.**

3.2.1 **Linear Delta Modulation.**

LDM was first described in a French patent in 1946 and although the simplest form of delta modulation, which has been shown to be totally inadequate for digital speech encoding, it has been included in this chapter as it is the foundation from which all other delta modulators have been developed.

The operation of the LDM encoder is very simple. A bandlimited input analogue signal is encoded into a binary signal which is transmitted to the decoder where the analogue signal is recovered. The binary signal at the transmitter is also fed back to the encoder where it is locally decoded and compared to the input analogue signal. The error (i.e. the difference between the input and decoded signals) is quantized by a two-level quantizer. It is the sampled quantized error signal that forms the binary output signal. This binary signal is in fact a digital representation of the sign of the error signal.

The local decoder in the feedback loop of the coder is an integrator, which is a linear network element. The integrator can be implemented using analogue techniques but, because it is common to quantize the input signal in both time and amplitude (using a sampler, zero-order hold circuit and an A/D converter) the rest of
the circuit can be implemented digitally. In this way it is possible to reduce the sensitivity of the circuit to component value fluctuations, and it also offers the facility of using integrated circuit techniques more easily than in the case of analogue implementation.

![Block diagram of the LDM encoder.](image)

The algorithm for the LDM encoder shown in fig. 3.1 is given by:

\[ b_n = \text{sgn}(Y_n - X_{n-1}) \]

\[ X_n = X_{n-1} + b_n S_n \]

where

- \( b_n \) is the quantized error signal,
- \( Y_n \) is the quantized input signal,
- \( X_n \) is the LDM approximation signal, and

PAGE 3-4
S, is the increment (i.e. step size) of the approximation signal.

In other words, the coder updates the approximation signal in the direction of the error between the present input signal sample, $Y_n$, and the value of the approximation signal, $X_{n-1}$, at the previous sampling time to produce the updated decoder output $X_n$. From fig. 3.1 it should be clear that the decoder in the receiver is equivalent to the local decoder in the feedback loop of the encoder.

The choice of an integrator for a voice encoding scheme is significant when one considers the ACF for speech. The adjacent-sample correlation, $C(1)$, gives an indication of what percentage of the present sample is contained in the next one. Clearly, only a D.C. signal will have a long-time averaged value of 100% (i.e. $C(1) = 1$) and hence it is unlikely that the perfect integrator, shown in fig. 3.2(a), will be optimum for speech coding. The performance of the encoder is improved if a leaky integrator, shown in fig. 3.2(b), is used instead. Not only does this exploit the characteristics indicated by the ACF, but it also has the added advantage of improving the performance of the codec in the presence of errors induced in the transmission channel. Any error at the input to the decoder will slowly be 'leaked' away by the
leakage factor 'L' and after a predetermined length of time the decoder state will be the same as the encoder to within an acceptable degree of error.

\[ y_n = y_{n-1} + x_n \quad \text{(a)} \]
\[ y_n = L \cdot y_{n-1} + x_n \quad \text{(b)} \]

\( L < 1 \)

Fig. 3.2: Discrete integrators - with and without leakage.

The shortcomings of the LDM encoder are well known and have been summarised in the previous chapter.

3.2.2 Continuously Variable Slope Delta Modulation (CVSD).

The problem with LDM is that there is only one level of the input signal at which the signal-to-noise ratio is maximized (cf. chapter 7). One way of improving the performance is to ensure that the input signal level is always close to the optimum level. In order to achieve this for speech the dynamic range of the signal would have to be reduced. This can be done using a
compressor which reduces the large amplitude signal levels relative to the lower levels. The distortion introduced by this technique is compensated for by using a complementary expander at the output of the decoder. Alternatively, this companding effect could be achieved by the delta modulator itself, and more simply than by complementary analogue circuits. 

Of the adaptive delta modulators that exploit the syllabic characteristics of speech the most commonly used is the CVSD codec. The block diagram of the CVSD coder is shown in Fig. 3.3. The main sections are the comparator and quantizer which produces the two-level version of the error signal, a compandor (containing a 'syllabic filter'), and a prediction filter (i.e. the leaky integrator described previously). The compandor is made up

Fig. 3.3: Block diagram of the CVSD coder.
of a slope detector and a low-pass filter with a 3-dB point ranging between 30 Hz and 160 Hz depending on the particular implementation\textsuperscript{18,21,22}. The step-size integrator (a low-pass filter) generates the envelope of the speech signal which is typically between 60 and 100 Hz\textsuperscript{21}. The operation of the CVSD is quite simple: the slope detector compares a windowed sequence of the binary output signal (typically three or four bits in length) and generates a pulse if three (or four) consecutive 1's or 0's is detected. This pulse excites the syllabic filter, whose output polarity is controlled by the binary encoder output signal, and is used as the input to the prediction filter, which has a typical value for its time constant of 1 millisecond. The output of the prediction filter is compared to the actual input signal with the difference between the two being quantized to produce the binary output signal.

The algorithm for the CVSD coder is given by:

\[ b_n = \text{sgn}(Y_n - X_{n-1}) \]
\[ S_n = B \cdot S_{n-1} + (1 - B) \cdot (V + S_n) \]
\[ X_n = L \cdot X_{n-1} + H \cdot (1 - L) \cdot S_n \cdot b_n \]

where

- \( b_n \) is the quantized error signal at the \( n^{th} \) sampling instant,
- \( Y_n \) is the actual \( n^{th} \) input sample,
CHAPTER 3: SPEECH CODEC DESIGN

\( x_n \) is the estimate of the input signal,

\( s_n \) is the step-size,

\( s_o \) is the minimum step size,

\( V \) is the output of the slope detector - \( V \) is a constant positive voltage if three consecutive outputs of the encoder are identical, otherwise \( V \) is zero,

\( L \) is the prediction constant,

\( B \) is the step-size leakage constant, and

\( H \) is the gain of the prediction integrator.

**Fig. 3.4** Modified version of the discrete integrator.

Determining the prediction and step-size leakage constants:

If the integrator shown in fig. 3.2(b) is modified to that shown in fig. 3.4, the time domain equations describing its operation become:

\[
y_n = L \cdot y_{n-1} + (1-L) \cdot x_n.
\]
The Z-domain equivalent:

\[ Y(Z) = L \cdot Y(Z) \cdot Z^{-1} + (1 - L) \cdot X(Z) \]

\[ \therefore Y(Z) \cdot (1 - L \cdot Z^{-1}) = (1 - L) \cdot X(Z) \]

\[ \therefore \frac{Y(Z)}{X(Z)} = \frac{1 - L}{(1 - L) \cdot Z^{-1}}. \]

The poles of this transfer function are given by:

\[ 1 - L \cdot Z^{-1} = 0 \]

\[ \Rightarrow Z = L \]

and,

\[ Z = e^{j\omega T}. \]

Hence,

\[ e^{-j\omega T} = \frac{1}{L} \]

\[ \therefore f_s = (-1/j2\pi T) \cdot \text{Ln}(1/L). \]

Correspondingly, the RC-equivalent filter has poles at:

\[ 1 + j\omega RC = 0 \]

\[ \therefore f_s = -1/(j2\pi RC). \]

Equating the two:

\[ (-1/j2\pi T) \cdot \text{Ln}(1/L) = -1/(j2\pi RC) \]

\[ \therefore L = e^{-T/RC}. \]

So, for an RC-time constant of 1 msec, and a sampling frequency of 16 kHz,

\[ L = e^{-1/16} = 0.94. \]

Similarly, for \( B \), the step-size leakage factor; for a leakage time constant of 6.4 msec,

\[ B = e^{-1/(16 \cdot 6.4)} = 0.99. \]
CHAPTER 3 : SPEECH CODEC DESIGN

The values for H and V vary considerably in the literature and they can take on values from 3 and 150 respectively down to 1 and 16 respectively. In practice it would be necessary to determine the optimum values for these parameters prior to evaluating the performance of the codec.

3.2.3 Constant Factor Delta Modulation (CFDM)

Whereas the CVSD codec exploited the syllabic characteristics of speech the CFDM codec uses instantaneous exponential adaptation to improve on the performance of the LDM codec.

At every sampling instant the step size of the codec is modified by a specific factor. To do this a one-bit memory is included as the adaptation depends on the immediately past binary output, $b_{n-1}$, and the present output, $b_n$. These two outputs are compared to each other and if they are the same then the step size is increased by a constant factor $P$, else it is decreased by factor $Q$.

\[
\begin{align*}
\text{if } b_n \cdot b_{n-1} = 1, & \quad k_n = P \\
\text{if } b_n \cdot b_{n-1} = 0, & \quad k_n = Q \\
S_n = S_{n-1} \cdot k_n \\
X_n = L \cdot X_{n-1} + S_n \cdot b_n
\end{align*}
\]

where
b_n is the encoder binary output,

k_n is the multiplication factor for the step-size adaptation,

S_n is the step size, and

X_n is the approximation to the input signal, Y_n, with L being the prediction constant.

Fig. 3.5: Block diagram of the CFDM codec.

Obviously the operation of the codec is highly dependent on the values of the time invariant factors P and Q. There are two conditions which must be taken into account when determining the values for P and Q. During the slope overload portion of operation (b_n = b_{n-1}) P must be large enough to respond to the input signal, while during the hunting phase (b_n ≠ b_{n-1}) Q must be
small enough to enable the approximation signal to converge to the input signal. Linear DM, which can neither match the high slopes of some signals nor converge to a steady-state value, is equivalent to a CFDM codec with both P and Q set to unity. In order to reduce the amount of slope overload noise it is necessary that $P > 1$, while in order to enable the signal to converge to a steady-state value it is necessary that $Q < 1$. If the product $P \cdot Q$ exceeded $(1 + \varepsilon)$, where $\varepsilon$ is positive, then the adaptation would tend to become unstable. This is because the net effect of the multiplications by P and Q would be an increasing step size (the effect of $P$ is greater than $Q$). The step size would eventually become so large that the output of the decoder would oscillate between the maximum and minimum values independently of the input signal. Hence, to ensure stability, the product $P \cdot Q$ must be limited to a maximum value of 1. It has been determined\textsuperscript{23} that the optimum condition is in fact $P \cdot Q = 1$. It was also found, by means of computer simulation, that the optimum value for $P$ was 1.5 and this was independent of the sampling frequency used\textsuperscript{23}.

In terms of a practical implementation the reciprocity of $P$ and $Q$ means that the possible step sizes within a certain range can only take on a limited number of discrete values. This means that a relatively compact step size 'dictionary' can be used. If, for example, the values for $P$ and $Q$ were 1.5 and 0.5 respectively,
some of the possible step sizes between 1 and 2.25 (inclusive) are:

1.000 1.125 1.265 1.424 1.500 1.688 1.898 2.136 2.250

whereas the optimum solution yields only three values in this range:

1.000 1.500 2.250.

Fig. 3.6 shows the difference between the two adaptive schemes discussed so far. It can be seen how, under severe slope overload conditions, the step size of the CFDM coder varies at a much faster rate than for CVSD. The CVSD coder represented here has values for H and V of 3 and 150 respectively. It should be remembered that the exact variation of step size for the CVSD coder depends on these values.

![Graph showing the variation of step size for CVSD and CFDM encoders.](image-url)
3.2.4 Hybrid Companding Delta Modulation (HCDM).

Although the previous two methods provide a significant improvement over LDM they both have shortcomings related to the type of companding used (cf. chapter 2). The development then of a hybrid companding scheme that exploits both the syllabic and instantaneous characteristics of speech would appear to provide a potential improvement over both of the methods discussed previously.

The slow response of the syllabic compandors to the sharp changes in the speech waveform could be remedied by increasing the sampling rate, but that would be contrary to what we're trying to achieve. The problem with instantaneous companding is that the step size range is fixed, regardless of the actual speech signal characteristics. As a result of the variation of the input signal power and slope the step size range may be optimum for one phoneme of speech but not another. If the instantaneous compandor is given access to the syllabic information it would be possible for the step size range to adjust to the different speech phonemes. This is essentially what the HCDM encoder does. Instead of using a fixed basic step size the instantaneous compandor has a basic step size that is modified, according to the signal level over the syllabic period, by the syllabic compandor. The basic step size thus produced is then modified by the instantaneous compandor to
produce the actual step size used by the prediction filter. The block diagram of the HCDM encoder is shown in fig. 3.7.

The input sample, \( Y_n \), is compared to the last output of the prediction filter (i.e., the estimate, \( X_{n-1} \), of the input). The step size is then modified according to the sign of the difference between the input signal and the estimate which is then increased or decreased using the new step size. The general algorithm for

\[ b_n = \text{sgn}( Y_n - X_{n-1} ) \]

Fig. 3.7: Block diagram of the HCDM encoder.
CHAPTER 3: SPEECH CODEC DESIGN

- The long-term basic step size is obtained from the RMS slope energy of the low pass filtered decoded signal. It is updated at the syllabic rate (i.e., approximately every 5 msec).

\[ E = \left[ \sum_{n=1}^{M-1} \left( X_n - X_{n-1} \right)^2 / (M-1) \right]^{1/2} \]

\[ \Lambda = \alpha E \]

where

\( X_n \) is the \( n^{th} \) predicted speech sample,
\( M \) is the number of samples in the syllabic period,
\( E \) is the RMS slope energy,
\( \alpha \) is the slope energy scale factor, and
\( \Lambda \) is the long-term basic step size.

- The instantaneous step size adaptation is determined from the present output (i.e., sign of the error) and the previous two outputs.

\[ k_n = f( b_n, b_{n-1}, b_{n-2} ) \]

\[ Y_n = k_n Y_{n-1} \]

where

\( k_n \) is the instantaneous multiplication factor,
\( Y_n \) is the instantaneous step size modification factor, and
\( f( b_n, b_{n-1}, b_{n-2} ) \) is specified by the logic rule shown in Table 3.1. It should be pointed out that the constant factor DM used here is a second order version of Jayant's
CFDM described in [23]. There are alternative versions of the logic rule shown here that, for instance, use the five most recent outputs (i.e. $b_n$, ..., $b_{n-4}$), but the difference in performance between these methods is not significant enough to warrant the extra hardware (refer to chapter 7).

Table 3.1: The HCDM companding logic (instantaneous).

<table>
<thead>
<tr>
<th>$b_n$</th>
<th>$b_{n-1}$</th>
<th>$b_{n-2}$</th>
<th>$k_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+1$</td>
<td>$+1$</td>
<td>$+1$</td>
<td>1.50</td>
</tr>
<tr>
<td>$-1$</td>
<td>$-1$</td>
<td>$-1$</td>
<td>1.50</td>
</tr>
<tr>
<td>$+1$</td>
<td>$+1$</td>
<td>$-1$</td>
<td>1.00</td>
</tr>
<tr>
<td>$-1$</td>
<td>$-1$</td>
<td>$+1$</td>
<td>1.00</td>
</tr>
<tr>
<td>$+1$</td>
<td>$-1$</td>
<td>$+1$</td>
<td>0.66</td>
</tr>
<tr>
<td>$-1$</td>
<td>$+1$</td>
<td>$+1$</td>
<td>0.66</td>
</tr>
<tr>
<td>$+1$</td>
<td>$-1$</td>
<td>$-1$</td>
<td>0.66</td>
</tr>
<tr>
<td>$+1$</td>
<td>$+1$</td>
<td>$-1$</td>
<td>0.66</td>
</tr>
</tbody>
</table>

The step size, $S_n$, is updated by combining the syllabic and instantaneous factors.

$$S_n = \Lambda \times \gamma_n$$

The estimate of the input sample is then given by:

$$X_n = L \cdot X_{n-1} + b_n \cdot S_n$$

where

$L$, the prediction constant is given by:

$$L = \exp(-\beta T)$$

with $\beta$ being the inverse of the prediction time constant, and $T$ being the sampling period.

NOTE: The slope energy, $E$, is determined from the decoded speech signal. If instead the actual input signal was used.
the slope information would have to be transmitted to the decoder. However, it has been found\(^ {173}\) that the advantage offered by the feedforward mode is minimal compared to the performance of the feedback scheme. Secondly, the syllabic companding could be obtained from the variation of the signal amplitude rather than the slope, but because the delta modulator essentially tracks the input slope, as opposed to the amplitude, the use of the slope information was preferred\(^ {173}\).

3.2.5 Song Voice Adaptive Delta Modulation (SVADM).

The SVADM scheme derives from an analytic, as opposed to empiric, approach to optimizing the transmitter and receiver. The design uses the fact that, in general, the optimum predictor (in the transmitter) is different to the optimum estimator (in the receiver)\(^ {173}\). A set of non-linear equations was derived for a theoretically optimum system and these equations were then reduced to piecewise linear equations to enable a digital implementation of the system. For a detailed derivation of these equations refer to Appendix A. Initially the system was developed for a Markov-Gaussian source and then optimised for voice signals.
The general equation describing the song system is given as:

\[ r_k = X_{k+1} \]
\[ r_k = X_k + g(b_k, X_k - X_{k-1}) + h(b_{k-1}, X_k - X_{k-1}) \]

From the derivation given in Appendix A,

For \( X_k - X_{k-1} > 0 \):

\[ g(b_k, X_k - X_{k-1}) = +0.08 + U(X_k - X_{k-1} - 0.08) \quad b_k = +1 \]
\[ -0.08 - U(X_k - X_{k-1} - 0.08) \quad b_k = -1 \]

For \( X_k - X_{k-1} < 0 \):

\[ g(b_k, X_k - X_{k-1}) = +0.08 - U(X_k - X_{k-1} + 0.08) \quad b_k = +1 \]
\[ -0.08 + U(X_k - X_{k-1} + 0.08) \quad b_k = -1, \]

and

\[ h(b_{k-1}, X_k - X_{k-1}) = 0.04 \times b_{k-1} \]

The graphs of the functions \( g \) and \( h \) are plotted in fig. 3.9.

From fig. 3.9 it can be seen that for speech the step size
function generator is linear except for the region near the origin where the minimum step size is limited to prevent a 'dead' zone.

The above set of equations was originally developed for a system using an 8-bit A/D converter at the input with an input signal range of -5V to +5V. This yields a minimum quantization step of 40 mV, and so, these equations can be combined and rewritten for the general system:

\[ S_{k+1} = |S_k|b_k + S_0b_{k-1} \]

where

- \( S_{k+1} \) is the new step size,
- \( S_0 \) is the minimum step size (corresponding to the 40 mV step in the 8-bit system).

The updated approximation to the input signal can then be written as:

\[ X_{k+1} = X_k + S_{k+1} \]

These two equations then characterize the SVADM system for speech signals. The form of these equations makes it very simple to implement the circuit using digital integrated circuits (cf. chapter 4).

The system specified by the equations includes the equivalent of a perfect integrator in the prediction filter. However, as has been mentioned before, the use of a leaky integrator serves the dual purpose of enhancing the performance for voice signals as well as...
enabling the decoder to recover from channel errors. The value of \( L \), the prediction constant, is set according to the sampling frequency. However, for a practical hardwired implementation it
is sometimes desirable to choose one value for all sampling rates, as was done by Dhadesugoor et al\textsuperscript{[21]}. Alternatively, the prediction constant can be determined from the RC-equivalent filter as was done for the CVSD, CFDM and HCDM systems.

3.2.6 Hybridisation of the Song Voice Adaptive system.

Essentially the Song Voice ADM is a delta modulator initially designed for a first-order Markov-Gaussian source and then empirically optimized for speech signals. However, the non-stationarity and quasi-periodicity of speech prevent the speech source from being classified as first-order Markov-Gaussian. Just as a combination of syllabic and instantaneous companding was found to improve the performance of the CVSD and CFDM codecs, so it would seem that by incorporating particular characteristics of speech signals into the design of the codec the performance of the Song Voice system could be enhanced. We have investigated means of modifying the design of the SVADM system to include syllabic companding, thereby incorporating the non-stationary characteristics of speech in the codec. In the course of our research we have developed two techniques for achieving this and they are referred to as 'Song Hybrid Companding Delta Modulation' (SHCDM) and "Modified Song Hybrid Companding Delta Modulation" (MSHCDM).
Both of these techniques utilise the same types of companding with the difference being in the method by which the instantaneous and syllabic companding factors are combined. The types of companding used are CVSD for the syllabic, and SVADM for the instantaneous.

3.2.6.1 Song Hybrid Companding Delta Modulation

The block diagram of the SHCDM encoder is shown below in fig. 3.11.

![Block diagram of the SHCDM encoder](image)

Fig. 3.11: Block diagram of the SHCDM encoder.

The associated algorithm is given by:

- The binary output is determined from the sign of the difference between the input and approximation signals.

\[ b_n = \text{sgn}(Y_{n+1} - X_n) \]
The instantaneous step size adaptation is determined in the same manner as for SVADM.

\[ Y_{n+1} = \left| Y_n \right| \cdot b_n + S_0 \cdot b_{n-1} \]

where

\[ S_0 \] is the basic step size.

The syllabic step size is obtained from the RMS slope energy of the decoded speech signal.

\[ E = \left( \sum_{n=1}^{M-1} (X_n - X_{n-1})^2 / (M-1) \right)^{\frac{1}{2p}} \]

\[ \Lambda = \alpha \cdot E \]

(For an explanation of the symbols refer to the section on HCDM - 3.2.4.)

The step size is then obtained by combining the syllabic and instantaneous factors.

\[ S_{n+1} = \Lambda \cdot Y_{n+1} \]

The updated approximation to the input signal is then given by:

\[ X_{n+1} = L \cdot X_n + S_{n+1} \]

where

\[ L \] is the prediction constant.

In this version of the hybridised Song Voice encoder the instantaneous step factor is generated using a fixed basic step (instantaneous) step size. This step factor is then used to modify the
basic step size of the syllabic compandor, thereby generating the hybrid step size.

3.2.6.2 Modified Song Hybrid Companding Delta Modulation.

Whereas the SHCDM encoder uses instantaneous companding to modulate the basic syllabic step size, the MSHCDM encoder uses syllabic companding to modify the basic step size of the Song Voice encoder. From fig. 3.12 it can be seen that the fixed basic step size \( S_0 \) of the instantaneous compandor has been replaced by the syllabically generated step size \( \Lambda \).

![Block diagram of the MSHCDM encoder](image)

The algorithm for the MSHCDM encoder is given by:

The algorithm for the MSHCDM encoder is given by:
The syllabic factor:
\[ \Lambda = \alpha \cdot E, \quad \text{as before.} \]

The step size is then generated using Song Voice companding with the basic step size provided by the syllabic compandor.

\[ S_{n+1} = |S_n| \cdot b_n + b_{n-1} \cdot (\Lambda \cdot S_0) \]

where

\( S_0 \) is generally set to unity.

The approximation signal is then given by:

\[ X_{n+1} = L \cdot X_n + S_{n+1}. \]
CHAPTER 4

HARDWARE IMPLEMENTATION OF A
SONG VOICE ADAPTIVE DELTA MODULATOR

During the early stages of this research it was envisaged that the major portion of the work would be concerned with network issues rather than the design of an efficient speech codec (the initial title for the project was "Voice Packet Switching" and involved the development of a suitable protocol). Owing to this our choice of a speech codec was based primarily on simplicity and ease of implementation. As far as was reasonably possible we wished to avoid any design and development work on a speech codec, it being preferable to utilise an already developed and tested algorithm.

Delta modulation was chosen as the means for providing a low cost digital speech codec (the motivation for the choice follows similar lines to those outlined in chapter 3) and in particular, the Song Voice Adaptive Delta Modulator (SVADM) was implemented. This codec was chosen in preference to the more conventional Continuously Variable Slope Delta Modulator (CVSD) owing to its reputed better performance and the fact that in a packet switched network it was found to be better than CVSD²¹³.

Although the SVADM algorithm has been presented previously (cf. chapter 3, section 2.5) it is repeated here for ease of reference.

\[ b_n = \text{sgn}(Y_{n+1} - X_n) \]
\[ S_{n+1} = |S_n| \cdot b_n + S_0 \cdot b_{n-1} \]
CHAPTER 4 : HARDWARE IMPLEMENTATION OF THE SVADM CODEC

\[ X_{n+1} = X_n + S_{n+1} \]

where

\( S_{n+1} \) is the new step size, and

\( X_{n+1} \) is the updated approximation signal.

The difference between the encoder and decoder is that the decoder does not include the circuit to generate \( b_n \), the error signal. Referring to fig. 3.10 the decoder is equivalent to the local decoder contained in the feedback loop of the encoder. In terms of hardware the encoder and decoder are identical except for the comparison section of the error generation circuit (described below).

This algorithm is simple to implement using commercially available digital integrated circuits. As with the codec implemented by Dhadesugoor et al\(^{[21]}\), 10-bit arithmetic is used with an input signal range of \(-10 \text{ V} \ldots +10 \text{ V}\). This gives a minimum step size, \( S_0 \), of approximately 10 mV. The design of the circuit has been divided into three discrete sections based on fig. 3.10 - these are the error generation and quantization section, the step size generator, and the approximation signal generator.

The main components of the circuit are the 10-bit D/A converter, the adders and latches. Simple logic components, such as
exclusive-OR gates, are also used. In order to implement the basic algorithm a total of 25 integrated circuits (including the ADC, comparator and amplifier chips) were used. Although digital speech coding techniques have been used some portions of the circuit were implemented using analogue techniques (the reasons are discussed in the relevant section of the text). For a detailed circuit design with component values refer to Appendix B.

4.1 Error Generation and Quantization.

The simplest means of achieving this function is to compare the analogue versions of the input signal, $Y_{n+1}$, and the approximation signal, $X_n$, by means of an analogue comparator. The analogue comparator is easily implemented using an operational amplifier.

The output of the comparator is sampled and quantized using a D-latch circuit.

An alternative technique for determining the error signal entails using an A/D converter (with its associated Sample-and-Hold module) and a digital comparator. Bearing in mind that 10-bit arithmetic is used and that the A/D converter would have to work at a sampling rate of 8 kHz (ie. the maximum conversion time would have to be in the region of 70 μsec) it was decided, for the initial design, to use the simpler and cheaper analogue comparison
Because the remainder of the circuit is implemented using digital techniques it is necessary to D/A convert the approximation signal, $X_n$, to facilitate the analogue comparison.

![Diagram](image)

**Note:** $X_0$ is the LSB of $X_n$.

$X_9$ is the MSB of $X_n$.

**Fig. 4.1: Error generation circuit block diagram.**

The diode on the comparator output is used to convert the output from a bi-polar one to a uni-polar one switching between the positive supply rail and ground. The voltage divider is used to scale the comparator output to a TTL-compatible level.

This circuit is responsible for the generation of the error signal:

$$b_n = \text{sgn}(Y_{n+1} - X_n).$$
4.2 Step-size Generation.

The increment to the step size, $S_o$, is 10 mV, which corresponds to a digital value of '1' (the full range for 10-bit arithmetic is $-512 \ldots +511$). This is multiplied by the sign of the previous error, $b_{n-1}$ (i.e. the digital step increment is $\pm 1$). Two's complement arithmetic is used throughout. If the comparator output switches to the positive supply rail (i.e. the previous approximation signal was too small) then the step increment is +1, else it is -1.

The most involved function of the step-size generation circuit is the circuit to achieve $|S_n| \cdot b_n$. The possible options for this operation are:

- $b_n = +1$
  
  This implies $|S_n| \cdot b_n = S_n$.

  Hence, for $S_n < 0$, $|S_n| = -S_n$
  
  while, for $S_n > 0$, $|S_n| = S_n$.

- $b_n = -1$
  
  In this case, for $S_n < 0$, $|S_n| \cdot b_n = S_n$,

  while, for $S_n > 0$, $|S_n| \cdot b_n = -S_n$.

Hence, using the sign-bit of the step-size word, $S_v$, and the bit representing $b_n$, the value for $S_n$ is multiplied by -1 if

$S_v \oplus b_n = 1 \equiv C$  (cf. fig. 4.2).
NOTE: Because two's complement arithmetic is used a positive sign is represented by '0' and a negative sign by '1'.

ie. \( b_n = +1 \) is represented by 0,
\( b_n = -1 \) is represented by 1.

In two's complement arithmetic negating a number is achieved by inverting all the bits of the word and adding 1. This is realised by using the control bit, \( C \), and an exclusive-OR gate and an adder.

ie. \(-S_n = (C \oplus S_n) + 1\).

This can be generalised to:

\[ |S_n| \cdot b_n = (C \oplus S_n) + C. \]

Fig. 4.2: Circuit block diagram for achieving \( |S_n| \cdot b_n \).

The value obtained in this way is then simply added to the increment to produce the new step-size.
It should be pointed out at this stage that although 10-bit arithmetic is used it is possible that certain functions (e.g., the step-size generation) could be implemented using fewer bits. The minimum word length would be determined from the maximum value of the particular variable in question. For example, in the case of the step-size, if the maximum value of this variable under normal operating conditions did not exceed 128 then the step-size generation circuit could be implemented using only 8-bit arithmetic. This in turn could have an effect on the chip count (e.g., two 4-bit adders are needed for an 8-bit adder whereas 3 chips would be necessary for a 10-bit adder). Hence, it would be necessary to study the characteristics of the variables in order
CHAPTER 4 : HARDWARE IMPLEMENTATION OF THE SVADM CODEC

to optimize the hardware design.

4.3 Approximation-signal Generation.
The updating of the approximation signal is achieved by simply adding the new step-size to the most recent value of the approximation signal. Although not included in this implementation the performance can be improved by using a leaky integrator to generate the approximation signal (the effect of this on the performance is shown in chapter 7). The approximation signal must then be D/A converted to facilitate the comparison with the next input sample.

4.4 Timing Considerations.
Referring to the detailed circuit diagram shown in Appendix B, it can be seen that three clock signals are used. The reason for this is to prevent race conditions being caused by two interactive circuit units changing state at the same time. For example, if the comparator output is latched at the same time as the D/A converter output is being updated then b_n could take on spurious values unrelated to the actual comparator output.

To prevent these race conditions the comparator output must only be latched once the DAC (D/A Converter) output has been updated. Similarly, the DAC can only be clocked once the approximation
signal has been updated. The previous error signal (and previous output), $b_{n-1}$, must be latched before the new comparator output is latched. If the step-size value and the approximation signal value, $S_{n+1}$ and $X_{n+1}$, are clocked simultaneously the propagation delay, particularly in the step-size generation circuit, will prevent any race conditions occurring at the $X_{n+1}/X_n$ latch. Hence, there are three sets of functions which must be clocked separately:

- **Clock 1**: triggers the latches for $b_{n-1}$, $S_n$, and $X_n$.
- **Clock 2**: triggers the DAC.
- **Clock 3**: triggers the latch for $b_n$.

The complete timing diagram for the codec is shown in fig. 4.4.
CHAPTER 4 : HARDWARE IMPLEMENTATION OF THE SVADM CODEC

4.5 Network Interfacing.

The next development stage involved interfacing the codec to a microprocessor as this would facilitate integrating voice communication into a computer communications network. For Local Area Networks (LAN's) in particular the advantages in terms of network flexibility are significant (e.g. a single communications link could serve both the voice and data needs of a number of users). The microprocessor would be responsible for the interfacing between the speech codec and the digital communications network (which would involve the packetization and depacketization processes). A block diagram, fig. 4.5, outlines the major components of the interface.

The system shown in fig. 4.5 is a very simple one in that it has the capability of linking only two codec terminals (Tel. A & B) whereas the real system would have to handle a considerable number of telephones. However, the network shown below was sufficient to implement a first-stage design - i.e. a simple full duplex two-way link between two speech terminals with a transparent network interface - which would enable the testing of the codec. It was initially intended to extend this system so that the microprocessor be utilised as a PAD (Packet Assembler Dissassembler) and exploratory work was done in this direction (under the author's supervision).
4.5.1 Serial/Parallel Interface.

The delta modulator encoder outputs a serial bit stream at a fixed rate. Similarly, the input to the decoder is also a serial bit stream. On the other hand, the microprocessor outputs and inputs parallel bytes of data. The primary function of this interface is to convert between the serial and parallel data modes.

Basically the interface consists of a standard SABus Parallel I/O unit with a few modifications and extensions. The serial/parallel conversion is achieved using shift registers with a byte synchronised clock controlling the transfer to and from the microprocessor's memory. Two input and two output channels are provided and hence, two codecs can be interfaced. All the timing signals, including the three clock signals required for the codecs.
(cf. section 4), are generated by this unit.

As with the codecs themselves, the timing specifications are crucial for the interface. The serial/parallel shift registers cannot be updated at the same time as \( b_n \), and neither can they be updated when data is being transferred to/from the microprocessor. The timing diagram for the I/O card is shown below with the block diagram shown in fig. 4.7 (for a detailed circuit diagram refer to Appendix B).

The software required to implement the straight-through full duplex link on the microprocessor was written on the PDP 11/23 computer in 8085 Assembler code and downloaded to the microprocessor. The algorithm flow charts and the control software are
CHAPTER 4: HARDWARE IMPLEMENTATION OF THE SVADM CODEC

Fig. 4.7: Block diagram of the interface unit.

listed in Appendix C.

Once the system described above had been implemented it was necessary to test it and to verify the performance of the SVADM codec (as described by Dhadesugoor et al\12\). Because of the dynamic nature of speech signals the determination of the various objective measures (as described in chapter 6) becomes problematic without suitable data storage facilities. At this point the emphasis of this research had swung towards investigating possible
codecs for speech that could be implemented as a one, or two chip low cost delta modulator set. It was therefore necessary to optimize the design with respect to the various circuit parameters, word lengths (e.g. the word lengths necessary for the variables, such as the step size, of the design algorithm), etc.

Rather than build hardwired versions of the modified codecs it was decided to implement a computer simulation facility which would enable the design to be optimized via the software, and only when the optimum design had been finalised would it be committed to a hardware implementation, and later to a chip.

One of the reasons for implementing an integrated circuit version of the speech codec is that the codec implementation using discrete chips required two SABus cards to contain the circuitry. Given that in a commercial communication network these cards would have to be duplicated at each terminal it is clear that an integrated circuit version would be both more economical and physically more acceptable.

The simulation process has the added advantage of enabling the performance evaluation to be readily achieved using software. The simulation system is described in the next chapter.
CHAPTER 5
COMPUTER SIMULATION

The process of testing and optimising the design of the codecs has been made considerably easier by the implementation of a simulation package on a minicomputer. This simulation package does not incorporate any Computer Aided Design (CAD) facilities requiring the basic design of the codecs to have been established prior to using this system. In essence this is a Computer Aided Test (CAT) station which enables the user to optimise and evaluate the codec design before implementing it in hardware.

The simulation package that was designed for this research project is divided into four main sections.

- **Interfacing**: In order to simulate a speech processing system it is necessary to be able to input and output speech signals. This section is responsible for the interfacing between the analogue speech source (the talker/listener) and the digital processing software.

- **Codec simulation**: Essentially the central feature of the system, this section simulates the speech codec algorithms which were discussed previously in chapter 3.

- **Measurement**: The various objective quality measures, discussed in chapter 6, are determined arithmetically by this section.

- **Peripheral**: Because of the speed limitations of the hardware of the simulation system (i.e. the computer and
CHAPTER 5: COMPUTER SIMULATION

interfacing hardware) a unit to perform digital interpolation of the sampled data was included. Another feature of this section is a digital filtering package.

Fig. 5.1 shows the structure of the simulation package. The four sections listed previously are subsequently individually discussed in detail.

The computer used for the simulation was the Hewlett Packard 200 Series HP9836C which is designed around a Motorola 68000 microprocessor. The interface module shown in fig 5.1 was implemented using the HP6942A Multiprogrammer which has the facility for various plug-in sub-modules (e.g. Analogue-to-digital converters, etc.). The hard disc storage unit was used as a mass storage
device (i.e. for storing raw and processed speech data). The simulation software was written in BASIC and PASCAL - the control software for the interface unit was written in BASIC because, in order to access the sub-modules in this unit extended addressing is needed and this facility is not supported by the PASCAL available on the computer used for this project. The software responsible for processing the speech data (i.e. the software simulating the actual codecs and the software performing the measurement and peripheral functions) was written in PASCAL and compiled into assembler code in order to afford the maximum processing speed. A slight increase in processing speed (at the expense of considerable extra effort) could be achieved if the code was originally written in assembler. This is due to the fact that the PASCAL compiler is not 100% efficient, but because the processing speed is not critical the minimal improvement achieved is not sufficient to warrant this course of action.

The procedure for using the simulation package is described in detail in Appendix D, suffice it to say that it was designed to be user-interactive. The listings of the software programs and their associated algorithm flow-charts are given in Appendix E.

5.1 Interfacing.

Quite simply the interfacing involves converting the speech
information from analogue form to digital form and vice versa. In the first case, the talker speaks into the microphone. The signal from the microphone is then filtered by analogue means, sampled, quantized, digitized and transferred to the computer. Speech data stored in the computer can be sent to a listener via a digital-to-analogue converter, an analogue filter and a loud-speaker.

Two types of microphone were used. For the subjective analysis of quality (cf chapter 6) the mouthpiece of a standard telephone handset was used whereas, for the objective analysis a high quality microphone was used. The frequency responses of the two microphones used are shown in fig. 5.2. For the subjective

![Diagram](image_url)

**Fig. 5.2**: Frequency responses of the microphones used in the simulation system.
CHAPTER 5: COMPUTER SIMULATION

analysis the earpiece of the telephone handset was used to provide a measure of the quality of the different codecs in a realistic system (it is envisaged that the codecs under consideration would be incorporated into a LAN in which the terminals use standard telephone handsets. For the objective tests a high quality microphone was used in order to allow direct comparisons with previously published results.

The filter used on the input was an 8-pole Butterworth filter with a cut-off frequency of 2.5 kHz \( f_c = 2500 \text{ Hz} \). On the output a 6th-order Butterworth low-pass filter was used, also with a cut-off frequency of 2.5 kHz. For the subjective tests the input filter was replaced by a 6th order band-pass filter (300 - 2500 Hz) because in public communications systems it is common to multiplex a number of band-pass filtered signals into a single co-axial cable channel and this is typically the pass band used.

The purpose of the input filter is to prevent aliasing effects. The purpose of a band-pass filter on the input is to conform to general communications standards while a low-pass filter on the input enables the objective results to be compared to previously published results.

The output filters serve the dual purpose of removing the
out-of-band noise, produced by the codecs and the A/D converter, and converting the discretely sampled-data to a continuous wave.

The sub-modules used in the Multiprogrammer include A/D, D/A converters, buffer memory and a timer module.

- Timer module: this was programmed to operate at the required sampling rate of 8 kHz (actually, a sampling rate of 16 kHz was required but the interfacing hardware could not match the requirements, hence the sampling rate specified). All other sub-modules are controlled by the timing signals generated by the timer. The transfer of data from the multiprogrammer to the controller (i.e. the computer) is, essentially, controlled by the timer as well.

Fig. 5.3: Block diagram of the A/D system.
Buffer memory: during the input phase the data from the A/D converter is clocked into the buffer memory by the timer. When a block of data, of a size specified by the software, has been clocked into the buffer an interrupt is generated which then initiates the transfer of data from the buffer memory to the computer. While the transfer is in progress the sampled and quantized speech data continues to be clocked into the remaining memory space available in the buffer. During the output phase the speech data is downloaded from the computer to the buffer memory and clocked out to the D/A converter by the timer.

Fig. 5.4: Block diagram of the D/A system.

Analogue-to-digital conversion: this is performed by a 12-
bit A/D converter with an input signal range set to -10V +10V. This gives a dynamic range for the input signal of over 60 dB. The sampling of data is controlled by the timer.

Digital-to-analogue conversion: a 12-bit D/A converter with an output range of -10V +10V is used for this purpose.

5.1.1 Data input.
As mentioned before, the input of data is interrupt controlled. The optimum buffer size for the chosen sampling rate is first determined and then the memory card in the Multiprogrammer is initialised accordingly. The timer card is set to operate at the required sampling rate. The commencement of data acquisition is then initiated by the user, who has previously specified the length of the speech segment to be recorded (the maximum allowable length of the segment is 10 seconds). The rest of the operation is done under the control of the computer. Once the segment has been stored in the computer's memory the RMS level and the maximum and minimum levels are displayed. The user then determines whether or not to store the data on the mass storage device and indicates the choice, in response to a prompt from the controller (the computer).
5.1.2 Data output.
The selected speech segment (stored on the mass storage device) is transferred to the computer's memory. The user then selects which portions of the segment he/she would like to listen to. The segment (or part thereof) is then transferred to the buffer memory and clocked out to the D/A card at the operating rate of the codecs (i.e. 16 kHz). It is possible for the specified segment of speech data to be output once only or repeatedly. In other words, if the user so desires, a segment of speech data can be continuously repeated until interrupted, via the keyboard of the controller, by the user.

5.2 Codec simulation.
A total of seven different codecs have been simulated. From the discussion in chapters 2 and 3 it should be clear that only the encoder need be simulated because an ideal channel has been assumed and the local decoder in the encoder is equivalent to the decoder at the receiver. Four of these have been included purely for reference while the remaining three are the hybrid codecs that are central to this work. The seven codecs are those discussed in chapter 3 - i.e. LDM, CVSD, CFDM, SVADM, HCDM, SHCDM, MSHCDM.

The simulation of the codecs cannot be done in real time because firstly, the time overheads involved in transferring the data
between the computer and the Multiprogrammer are excessive, and secondly, the actual time taken to process each sample is longer than the sample period. Hence, the data, once having been processed by the codecs, is stored on disc and then, at a later stage, it can be downloaded to the Multiprogrammer.

The user selects which codec is to be used to process the speech data. The data is then passed to the routine simulating the codec. In all cases where maximum processing speed is required the software has been implemented as a compiled PASCAL module. Realising the simulation software entailed translating the algorithms for the codecs (as listed in chapter 3) into an equivalent PASCAL form.

The codec simulation serves two purposes. Firstly, the speech data processed by the codecs can be stored on disc and subsequently used for subjective listening tests. Secondly, the various objective measures can be evaluated with or without the processed data being stored on disc.

5.3 Measurement.

There are a number of objective quality measures (discussed in chapter 6) which have been included into the simulation package. Besides being able to process speech data and listen to it the
The user is provided with the facility for determining these objective quantities.

Basically the method for assessing the quality (objectively) is as follows:

The recorded speech data is processed by the desired codec and the difference between the corresponding samples of the input and processed signals is stored. This difference signal is then filtered, using a digital filter simulated on the computer (cf. section 4.1 of this chapter), to remove the out-of-band components of the signal. The respective powers of this signal and the input signal are then compared yielding the signal-to-noise ratio.

5.4 Peripheral

The term 'peripheral' has been used here to describe those functions, essential to the simulation, that cannot be included in any of the other sections. The two functions included in this section are filtering and interpolation.

5.4.1 Filtering

A non-recursive digital filter operating at 16, 24, 32 or 64 kHz has been implemented. A detailed description of the filter design procedure and its resultant characteristics is given in Appendix
F. The purpose of the filter is to remove the out-of-band components of the noise signal so as to enable the calculation of the various signal-to-noise ratios. This filter does not replace the analogue filter at the codec output but simply simulates it for measurement purposes.

5.4.2 Interpolation

The sampling frequency of the A/D card is limited to a maximum value of 11.4 kHz (88 µsec sampling period). Above this value the buffer size becomes insufficient for a sentence-length speech segment (the buffer size is determined by the required sampling frequency and the data transfer rate using the HP-IB). This problem can be overcome by digitally interpolating the sampled data to a higher effective sampling rate.

Essentially, interpolation involves fitting a curve to the discrete sampled data values. In order to interpolate to any multiple of the base sampling rate a continuous curve could be fitted to the available data. This could be done using Lagrange polynomials:

$$p_{n-1}(x) = \sum_{i=1}^{n} \left( \prod_{r \neq i}^{n} \frac{P(x-x_r)}{P(x_i-x)} \right) y_i$$

where $n$ is the number of samples in the set, and $x_i$ and $x_r$ are the known sample values.
With a known set of samples the polynomial \( p_{n-1}(x) \) could be calculated and then all values on the fitted curve could be determined. When the number of samples in the set becomes very large the amount of processing becomes excessive and hence this method would be very costly in terms of processing time.

In the case of digital communications some of the common sampling rates are 8, 16, 32 and 64 kHz and hence for the purposes of the simulation it would only be necessary to interpolate to the integer multiples of the base sampling rate. This can be achieved using a digital filter.

In fig. 5.5 a waveform is sampled at 8 kHz. If null samples (ie. samples with value zero) are inserted between each actual sample then the information contained in the sampled waveform has not been changed. The time between samples though has been halved, thereby effectively doubling the sampling rate. If this signal is then passed through a digital filter operating at the new sampling rate the effect will be to average out the signal. In order to restore the RMS level of the signal to the pre-insertion value (ie. before the null samples)
were inserted) the samples are multiplied by a constant factor whose value is determined by the ratio of the new-to-original sampling rates (i.e. if the effective sampling rate is doubled then this factor has a value of 2).

A basic value of 8 kHz has been used as the practical sampling frequency. If a higher effective sampling frequency is required then the 8 kHz sampled data is digitally interpolated to the desired rate. It should be noted that in this package only integer multiples of the base rate are attainable (i.e. sampling rates of 8, 16, 24, 32, 64, etc. are attainable).

The data to be interpolated is read off the mass storage device and passed to the interpolation routine (this data was previously stored on the mass storage device during the input phase).

5.5 Special Considerations

There are a number of differences between the simulated version and a hardwired implementation of the codec circuits. The advantage of the simulated version is that the circuit performance can be evaluated with or without these variations included.

5.5.1 Signal Range

The first of these differences involves the maximum range of the
decoded speech signal. At the input the signal range is limited by the range of the A/D converter and the power supply voltages for the pre-sampling filter. Numerically this range, for a 12-bit ADC (Analogue-to-Digital Converter), is \([-2048 \ldots +2047\]). In the simulation software the arithmetic signal range is limited by the word size of the computer. In this case a 16-bit word is used for integers (ie. a range of \([-32768 \ldots +32767]\)) while the 32-bit floating point arithmetic limits the range for real numbers to \([-10^{308} \ldots +10^{308} - 1\]). Hence, it is possible to let the decoded speech signal, as represented in the computer, have a range which is larger than the range of the input signal.

In order to evaluate the performance of the codec algorithm without the constraints introduced by a practical implementation the decoded signal range was only limited by the numeric range of the computer. On the other hand, to simulate the practical codec the decoded signal range is limited to match the range of the input signal (ie. numerically the decoded signal would be limited to values between \([-2048\) and \(+2047\]).

5.5.2 Step Size.

Another parameter which could be limited in practice is the step size or step size dictionary. For example, in the analogue implementation of the HCDM coder given by Un et al\(^{17}\) the
Constant Factor step size dictionary is limited to eight values 
(1.5^0, 1.5^1, 1.5^2, \ldots, 1.5^7). The reason for doing this is 
obvious: it is not possible to use an infinite number of gain-
setting resistors in the circuit. The significance of this is 
that the step size range produced by the constant factor compandor 
is limited to a maximum and a minimum value.

Again, in order to evaluate the performance of the algorithms no 
explicit limits were put on the step size ranges.

However, in the case of the codecs using constant factor 
adaptation (ie. CFDM and HCDM) a limit was set for the minimum 
value of the step size multiplication factor. During the 
'hunting' phase (when the input signal is in a steady state, or 
near-steady state condition) the locally decoded signal oscillates 
about the input signal which in turn causes the constant factor 
modulator step size to decrease (ie. k_n = 2/3 and Y_n = Y_{n-1} \times 
0.66). If the hunting phase continues for long enough then Y_n 
becomes small enough to degrade the performance of the codec. 
This is caused by the response of the encoder being impeded by the 
very small step factor. In other words, when there is a 
change in the input signal level after a period of inactivity the 
indefinately small step factor prevents the encoder from tracking 
the signal and a severe case of slope overload is experienced. If
the minimum step factor is limited then it does not take as long for this value to recover from the steady state condition, thereby preventing dead-bands from occurring in the decoded speech signal output.

Similarly, the value of \( Y_n \) can be limited to a maximum value in the software, but for the objective performance evaluation this was not done.

Thus for the simulations comparing the relative performance of the HCDM, SHCDM and MSHCDM, the signal value and step size ranges were allowed to follow the algorithm exactly without any of the constraints introduced by a practical hardwired implementation of the codec circuit. In chapter 7 the effects of these constraints on the various codecs' performance have been studied.
CHAPTER 6

CODEC PERFORMANCE ANALYSIS

The assessment of speech coder performance can be divided into two discrete sections - objective and subjective testing. The objective measures, on the one hand, are not sufficiently well established to provide an unambiguous indication of a codec's performance while, on the other hand, there does not exist a generally applicable method for subjectively evaluating the quality of the codec. Compounding these problems is the fact that the correlation between the results of the objective and subjective tests is not well understood and as a result there does not exist one well-defined empirical method for determining the quality of a speech processing system. Due to this anomaly our codec performance evaluation has been based on both objective and subjective test procedures, with the emphasis being placed on the subjective techniques.

6.1 Objective Performance Evaluation

It has been established that acceptable voice communication is reliant on the preservation of the short-time amplitude spectrum of the signal\(^4\). In the case of waveform coders the quality could be characterized by a signal-to-noise ratio (the noise being the difference between the original input signal and the eventual output signal). The use of this measure can be justified on the basis of the operation of the waveform coders - i.e. the waveform coder essentially attempts to preserves the amplitude spectrum of
the signal by producing a replica of the time domain waveform.

In the case of sampled data signals the signal-to-noise ratio (SNR) can be defined as:

\[
\text{SNR} = 10 \log \left( \frac{\sum_{i=1}^{N} Y^2(i)}{\sum_{i=1}^{N} e^2(i)} \right)
\]

where

- \( N \) is the number of samples in the speech segment used for the test (typically the summations are over the duration of a sentence-length segment),
- \( Y(i) \) is the \( i^{th} \) input signal sample, and
- \( e(i) \) is the corresponding noise sample.

\[ e(i) = Y(i) - X(i) \]

where

- \( X(i) \) is the output of the codec.

However, although this is probably the most commonly used method for measuring coder distortion, it does have one significant drawback. The effect of high energy components of the speech segment is to swamp the contribution to the SNR measure of the lower energy segments (e.g., some unvoiced sounds). An alternative measure, which has been found to exhibit better correlation with subjective measures\(^{22,28}\), is the segmental signal-to-noise ratio which assigns a more equitable weighting to the high and low
energy components of the signal. The segmental SNR computes the
SNR for short portions of the speech segment (typically 15 - 20
msec in duration) and then averages these short-time measures over
the duration of the full segment.

The segmental signal-to-noise ratio is then defined as:

$$\text{SNRSEG} = \left( \sum_{m=0}^{M-1} 10 \log \left( \frac{\sum_{i=1}^{N} Y^2(i + Nm)}{\sum_{i=1}^{N} e^2(i + Nm)} \right) \right) / M$$

where

- $M$ is the total number of short-time segments,
- $N$ is the number of samples in the short-time segment, and
- $Y$ and $e$ are as defined previously.

As has been mentioned before, there are basically two types of
noise produced by a delta modulator, namely slope-overload noise
and granular noise. The relative effects of the two types of
noise on a listener's subjective response have been found to be
different\(^{(293)}\) (i.e. the one type of noise is less objectionable
than the other). There are two measures which describe the
amounts of distortion due to one or the other of these noise types
- Signal-to-Granular Noise Ratio (SNG) and Signal-to-Overload
Noise Ratio (SNO). If $e(i)$ is the $i^{th}$ noise sample and $e_{g}(i)$ and
$e_{o}(i)$ the $i^{th}$ granular and overload noise samples respectively,
then the SNG and SNO are defined as:

PAGE 6-3
\[ e_s(i) = \begin{cases} \frac{e(i)}{e(i) \cdot e(i+1)} & \text{if } e(i) \cdot e(i+1) < 0 \\ 0 & \text{otherwise} \end{cases} \]

and

\[ e_s(i) = \begin{cases} \frac{e(i)}{e(i) \cdot e(i+1)} & \text{if } e(i) \cdot e(i+1) > 0 \\ 0 & \text{otherwise} \end{cases} \]

With this classification of the noise, SNG and SNO are then defined by:

\[ SNG = 10 \log \left( \sum_{i=1}^{N} y^2(i) / \sum_{i=1}^{N} e_s^2(i) \right) \]

and

\[ SNO = 10 \log \left( \sum_{i=1}^{N} y^2(i) / \sum_{i=1}^{N} e_s^2(i) \right) \]

As in the case of the total signal-to-noise ratio there are corresponding segmental values for SNG and SNO.

The segmental signal-to-granular noise ratio:

\[ SNGSEG = \left( \sum_{m=0}^{M-1} 10 \log \left( \sum_{i=1}^{N} y^2(i+Nm) / \sum_{i=1}^{N} e_s^2(i+Nm) \right) \right) / M \]

The segmental signal-to-overload noise ratio:

\[ SNOSEG = \left( \sum_{m=0}^{M-1} 10 \log \left( \sum_{i=1}^{N} y^2(i+Nm) / \sum_{i=1}^{N} e_s^2(i+Nm) \right) \right) / M \]

(The symbols have the same interpretation as before.)
The test set-up for measuring these quantities is shown above.

6.1.1 **Dynamic Range Test.**

This test involves computing the signal-to-noise ratios (SNR, SNRSEG, etc) for different levels of the source signal. Having established the peak value of the signal-to-noise ratio over the range of input signal levels, the dynamic range is then defined as the range of signal levels for which the signal-to-noise ratio is greater than, or equal to a value 3 dB below the peak value.
6.1.2 Speech Material.

Real speech, spoken by male and female voices, was used as the source excitation. Short phonetically balanced sentences were spoken by the different speakers with the sentences being taken from the Harvard set of sentences.

6.2 Subjective Quality Evaluation.

Because the objective measures described previously fail to give an absolute measure of a coder's quality, and because the system users' assessment of performance is subjective by nature (based on the total auditory impression on the listener) formal judgements on coded speech quality are based on subjective testing.

The use of speech intelligibility alone as a measure of the quality of a speech communication system is not sufficient. Two codecs, for example, may exhibit similar word/sentence intelligibility scores while listeners find one more acceptable than the other. The main problem lies in the fact that there are invariably a large number of parameters (e.g., received signal volume, intelligibility, noise level, etc.) affecting the listener's evaluation, and it is difficult to estimate the relative importance of the different parameters, either singly or in combination with each other. Essentially what is needed is a one-dimensional scale which would enable a speech transmission.
system to be rated on the basis of listener preferences, thereby enabling the communication system to be compared to other systems regardless of the physical parameters of each.

Hecker and Guttman\(^{32}\) have categorized the subjective testing procedures as either 'analytic' or 'utilitarian'.

- **Analytic techniques**: these techniques attempt to establish a circuit's performance criteria on the basis of the various psychological components affecting a listener's judgement (e.g. the effects of low-pass, band-pass or high-pass filtering of the signal on the assessment of the coded speech's quality; or the difference between listener responses due to signal distortion as opposed to the presence of background distortion).

- **Utilitarian techniques**: these are concerned with obtaining an over-all measure of the speech quality without investigating the psychological factors. Despite the problems associated with using a unidimensional scale for assessing speech quality the utilitarian techniques are preferred (from an engineering point of view) for quality measurements\(^{27,35}\) as this allows direct comparisons to be made between codecs. In other words it would allow a choice to be made between different types of codecs based simply on how they rate on a single quality scale.
Basically the utilitarian methods can be divided into two main categories (after Munson and Karlin [33]) - direct, and indirect comparisons (cf Table 6.1). In the case of indirect comparison the listener is only presented with the system in question and must rely on previous experience with other systems in order to make a judgement. Direct comparisons involve presenting the listener with two circuits in close succession and the listener then evaluating the one relative to the other.

Table 6.1: Classification of speech quality assessment methods [33].

<table>
<thead>
<tr>
<th>Indirect comparisons</th>
<th>Category tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>o Loudness</td>
</tr>
<tr>
<td></td>
<td>o Quality</td>
</tr>
<tr>
<td></td>
<td>o Effort</td>
</tr>
<tr>
<td></td>
<td>Intelligibility tests</td>
</tr>
<tr>
<td></td>
<td>o Words</td>
</tr>
<tr>
<td></td>
<td>o Sentences</td>
</tr>
<tr>
<td></td>
<td>o Immediate appreciation</td>
</tr>
<tr>
<td></td>
<td>Articulation tests</td>
</tr>
<tr>
<td></td>
<td>o Sounds</td>
</tr>
<tr>
<td></td>
<td>o Syllables</td>
</tr>
<tr>
<td></td>
<td>Repetition count</td>
</tr>
<tr>
<td></td>
<td>Message rate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Direct comparisons</th>
<th>Fixed reference, variable unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Variable reference, fixed unknown</td>
</tr>
<tr>
<td></td>
<td>Variable reference, variable unknown</td>
</tr>
</tbody>
</table>

(a) Indirect comparisons.

o Category judgement: these tests are usually concerned with a particular aspect of the system (i.e. loudness, quality or effort required to use the system). The listener is required to evaluate the circuit according to the categories listed.
Typical terms used are:

excellent, good, fair, poor, bad - for 'quality';
much too loud, too loud, satisfactory, too soft, much too soft - for 'loudness';
no effort, no appreciable effort, moderate effort, considerable effort, impossible to understand - for ease of use of the system.

- Intelligibility tests: this could involve evaluating the words or sentences correctly interpreted and identified by the listener. Alternatively, a simple YES-NO decision as to whether the sentence was understood (without the listener reviewing what was heard) could be used (i.e. immediate appreciation).

- Articulation tests: similar to intelligibility tests, these involve identifying the sounds received by the listener.

- Repetition count: this test determines the number of times a speech message must be repeated to effect satisfactory transmission.

- Message rate: this test involves a two-way link over which two communicants attempt to solve a problem as efficiently as possible.

These various tests are heavily dependent on factors outside the
control of the tester. For instance, different listeners are likely to have widely differing interpretations of the meanings of terms such as 'good', 'poor', 'too loud', etc (cf category tests). Learning factors on a new circuit could influence the outcome of the tests (i.e. the listeners' responses could change as they become more familiar with the characteristics of the circuit). A particular problem with the 'immediate appreciation' test is the lack of control or confirmation - a listener may think he/she has interpreted the sentence correctly whereas, in fact, a mistake has been made.

(b) Direct comparisons.
The general procedure for this type of testing involves pair comparisons between two circuits. The one circuit would be a reference, the value of whose parameters are well established, while the second would be a test circuit whose parameters are to be assessed. Basically the test involves varying the operating conditions of either the test or reference circuits (or both) and establishing the point at which the test subject (i.e. the listener) judges the two circuits to be equal. The test could be used to evaluate a particular aspect of a system's performance (such as loudness), or simply to assess the over-all preferences of the listeners. A significant point about this test procedure is that the test subject is not required to specify the reasons
CHAPTER 6: CODEC PERFORMANCE ANALYSIS

for their judgement.

As shown in Table 6.1 there are three basic modes of comparison.

- **Fixed reference**: the test circuit is varied until it is judged equal to the fixed reference. The only subjective attribute commonly tested using this method is the loudness balance. This is because the effects of the physical circuit parameters on the various attributes, such as quality, are not always easily ascertained.

- **Variable reference**: the complement to the above method, this test involves varying the parameters of the reference circuit and comparing the effect with the fixed test circuit.

- **Isopreference method** (discussed in more detail shortly): this test method involves varying both the test and reference circuit conditions.

6.2.1 *Isopreference Test.*

Whereas in the past the value of comparisons between circuits based on a single parameter (e.g. intelligibility) were limited the aim of the isopreference method is to implicitly take the effect of all parameters into account. As implied by the above discussion, the isopreference test (designed by Munson and Karlin, and modified by Tedford and Frazier) involves varying both the test and reference conditions - a basic reference
signal has varying controlled amounts of noise added to it with the resultant signal being compared to a test signal whose energy level is varied over some range. Isopreference contours are constructed by finding speech level/noise level combinations which are equally preferred.

For example, in fig. 6.3, the point 'A' represents a certain signal condition (i.e., a certain long-time averaged signal and noise level). From preference tests involving direct comparisons, points 'B', 'C' and 'D' (representing three other signal/circuit conditions) were found to be equally preferred to the condition specified by 'A', enabling those points to be linked by a contour. Similarly, using condition 'E' as a starting point, 'F' and 'G' were found to be equally preferred to 'E', enabling a second contour to be drawn. The distances between the contours can be used to provide the so-called transmission preference unit scale (TPU) which was intended to be used as a unidimensional speech quality measure.

A simplified measure, derived from the isopreference procedure,
involves calculating a subjective signal-to-noise ratio.

Essentially this test involves comparing the test system with a reference system which has been corrupted by a controlled amount of noise, with a known spectral shape (the signal level and noise level of the reference are both known). The subjective signal-to-noise ratio of the test is then defined as the SNR of the reference system which was equally preferred to the test system. This signal-to-noise ratio was recommended as a speech quality measure by the IEEE subcommittee on subjective measurements\(^{27}\).

In an attempt to consolidate the issue of subjective quality measurements the IEEE subcommittee recommended three measures as opposed to one single test procedure.

6.2.2 IEEE Recommendations.

Some of the more important attributes of the testing scheme are outlined below.

- Quality attributes other than Preference must be neutralized - eg. intelligibility should not be an issue as it should be well established.

- Results should be quantifiable on a unidimensional scale.

- Testing methods should allow for the evaluation of one-way communications systems only as two-way systems introduce
behavioral indices (the speakers modify their speech characteristics in order to accommodate the system's properties - eg. very noisy communications channels may induce the speakers to talk louder).

- The language used, and the semantic content thereof should be familiar to the listener.
- The speech material used for the test and reference signals should be of the same type. Narrative material (eg. extracts from novels, etc) or short homogeneous sentences (eg. the 'Harvard Sentences' \(^{273}\)) can be used.
- The listener group should be a subset of the intended users of the system being analyzed.
- If the system specifications require trained operating personnel then the test group (of listeners) must be similarly trained.

The three tests recommended by the IEEE subcommittee are the category judgement test, isopreference test and the relative preference test. Whereas the isopreference test involves using speech signals corrupted by different levels of noise the relative preference method uses different types of speech distortion. In this work the relative preference test has been discarded because the different delta modulators under consideration essentially produce the same type of noise. Because of the problems
associated with defining terms such as 'good', 'excellent', and 'poor' (in terms of speech quality) the category judgement tests were not performed. This problem becomes particularly relevant when attempting to compare the results of the category tests for different speech encoders because of the inevitable variation in interpretation by different listeners of the descriptive terms.

6.2.3 Subjective Test Procedure.

The test procedure that was used for subjectively assessing the quality of the processed speech involved two separate stages. Firstly, a combined speaker recognition/intelligibility test was performed. This was then followed by a preference test. Based on section 6.2.2 the following test conditions applied.

o Speech material used: a set of sentences was selected at random from the 1969 set of 'Harvard Sentences'. In this way it was possible to ensure that phonetically balanced sentences of approximately the same length were used throughout. The tests were conducted in English and all listeners were screened to ensure that their mother-tongue was English. During one full test sequence (e.g. the isopreference test) no speech material was repeated.

o Talkers: to prevent the tests being biased by a talker-system
interaction (i.e. a particular talker's voice may highlight certain characteristics of one system) a variety of talkers were used. Two males and two females were used to read the sentences mentioned above. For both sexes two age groups were used (20 - 35 years, and 35+ years). More specifically, the ages of the talkers were as follows:

Male 1 : 23 years old,
Male 2 : 46 years old,
Female 1 : 24 years old, and
Female 2 : 48 years old.

o Listeners: because it is envisaged that our systems would be in general usage (e.g. in a commercial communications network) the listener group was chosen at random from the staff and students of the university where this research took place. No-tests for hearing deficiencies were carried out as we were primarily interested in the response of the average listener. A group of 50 listeners was used with only one member of this group having had any prior experience with any form of subjective listening test.

o Training: an untrained group of listeners was used. Prior to performing the three tests the testing procedure was explained
to the listener.

Test equipment: all speech material was recorded via a standard telephone handset. The speech signal was then band-pass filtered (300 - 2500 Hz), sampled at 8 kHz and digitally stored on magnetic disc. It was then processed by either the simulated hybrid delta modulators or the reference signal-generator (the interfacing and simulation is described in detail elsewhere in this work). Playback of the speech material was via a low-pass filter (cut-off at 2500 Hz) and a standard telephone handset. Because it has been found that the background noise level of the room is not critical to the tests\(^3\) a sound proof room was not considered necessary and instead a suitably quiet office was used as the venue for the tests, with listeners being tested one at a time.

6.2.3.1 Intelligibility and Speaker Recognition Test

The purpose of this test is to establish that the intelligibility and the speaker-identity preservation for the codecs is at an acceptable level, thereby providing a basis upon which to conduct the isopreference tests.

Although the two characteristics (speaker recognition and intelligibility) are commonly considered separately, the nature of these
two tests enabled them to be combined into one.

Sentence intelligibility, as opposed to word intelligibility, was used as it matches the practical communications characteristics more closely (a telephone conversation is more likely to consist of sentences and phrases than single words). The identification of speakers by human listeners is regarded as an important aspect of codec performance and is, in fact, an additional subjective measure of the quality of the coded speech\(^4\). However, it is generally a more important issue for vocoders than waveform coders\(^4\). Basically, the listener was presented with a sentence processed by one of the three hybrid delta modulators and then asked to repeat (vocally) the sentence and to identify the speaker. Because the four speakers used were not familiar to the listeners, before each processed sentence the listener was presented with two recorded sentences spoken, one each, by either the two male speakers or the two females (it is unlikely that a

Table 6.2: Example of the test sequence for the Intelligibility & Speaker Recognition test.

<table>
<thead>
<tr>
<th>Male 1</th>
<th>Reference sentence spoken by either Male 1 or Male 2, processed by one of the three hybrid delta modulators.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male 2</td>
<td>Reference sentence</td>
</tr>
<tr>
<td>Test sentence</td>
<td>Reference sentence</td>
</tr>
</tbody>
</table>
male voice would be mistaken for a female voice). These two sentences are used as references so that the listener is acquainted with what each speaker's voice sounds like. An example of the test sequence heard by a listener is given in Table 6.2. The listener then indicates which speaker said the test sentence, and repeats the sentence. An example of the test form that was to be filled in by the listener is shown in fig. 6.4. A 'PARTLY CORRECT' sentence is one for which the listener got the basic semantic content correct, but some of the words incorrect.

<table>
<thead>
<tr>
<th>Sentence Number</th>
<th>Did you understand the sentence?</th>
<th>YES</th>
<th>NO</th>
<th>PARTLY</th>
<th>Identify the speaker.</th>
<th>1</th>
<th>2</th>
<th>Not Sure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6.4: Intelligibility & Speaker Recognition test answer form.

A total of 15 processed sentences were presented to the listener (5 each for the three hybrid DM's). The sequence of speakers was randomised as is the sequence of the codecs. This randomization
process is to prevent the listener from detecting a pattern or sequence in the test procedure, or from being able to predict the speaker for the next sentence.

6.2.3.2 Isopreference Test.

The procedure used here was based on the recommendations of the IEEE subcommittee\textsuperscript{[27]}, and not the full test specified by Munson and Karlin. In essence the test involves deriving the psychometric curves\textsuperscript{[33]} for the preference tests (for each of the three hybrid delta modulators), and then, from these curves, determining the subjective signal-to-noise ratios\textsuperscript{[33]}.

Following the intelligibility/speaker recognition test the listener was presented with a preference test. Basically the listener hears two sentences (a reference and a test) and is required to indicate which of the two they would prefer as a source of information. In total the listener was presented with 24 such reference-test pairs (8 each per hybrid codec).

The reference signal was generated by adding a controlled amount of noise to the original speech signal. It has been found\textsuperscript{[27]} that in the case of digitally processed signals multiplicative, as opposed to additive, random noise is most suitable. From the definition given by Schroeder\textsuperscript{[36]} the reference signal $r(i)$ is
given by:
\[ r(i) = \frac{s(i) + n(i)}{k} \]

where

\[ k = \left( 1 + a^2 \right)^{1/2} \]

where

'a' is the coefficient specifying the SNR,

\[ n(i) = a \cdot s(i) \cdot e(i), \]

e(i) being a random variable taking on values of +1 or -1 with equal probability,

\[ s(i) \] is the sampled input speech signal, and

\[ n(i) \] is the pink noise signal.

In other words, the reference signal is generated by taking a controlled proportion of the randomly inverted input signal and adding it to the original input signal.

Once again, the sentences (24 reference and 24 test) were spoken by the four speakers. The 24 test sentences were processed (8 each) by the three codecs. The sequence in which the test sentences were presented was randomized to prevent the listeners identifying the different codecs, and the allocation of speakers to the different codecs was also randomized (each codec had at least one sentence spoken by each of the four speakers). The
eight reference sentences, for each codec, were given different SNR values (-6, -3, 0, 2, 4, 6, 8, 10 dB) and the sequence in which they were presented to the listener was randomized. This was done to prevent the listener detecting a progression in the amount of degradation added to the references.

A test consisted of a repeated reference-test signal pair. In this way the listener can compare the reference (B) to the test signal (A) as well as the test to the reference. The full isopreference test is illustrated in fig. 6.5.

After each A-B-A-B sequence the listener is required to indicate their preference. An example of the form that was to be filled in by the listener is shown in fig. 6.6.
<table>
<thead>
<tr>
<th>Test Sequence Number</th>
<th>Preference</th>
<th>Test Sequence Number</th>
<th>Preference</th>
<th>Test Sequence Number</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>A</td>
<td>(2)</td>
<td>B</td>
<td>(3)</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>UNSURE</td>
<td></td>
<td></td>
<td></td>
<td>UNSURE</td>
</tr>
<tr>
<td>(4)</td>
<td>A</td>
<td>(5)</td>
<td>A</td>
<td>(6)</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>UNSURE</td>
<td></td>
<td></td>
<td></td>
<td>UNSURE</td>
</tr>
<tr>
<td>(7)</td>
<td>A</td>
<td>(8)</td>
<td>A</td>
<td>(9)</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>UNSURE</td>
<td></td>
<td></td>
<td></td>
<td>UNSURE</td>
</tr>
<tr>
<td>(10)</td>
<td>A</td>
<td>(11)</td>
<td>A</td>
<td>(12)</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>UNSURE</td>
<td></td>
<td></td>
<td></td>
<td>UNSURE</td>
</tr>
</tbody>
</table>

Fig. 6.6: Isopreference test answer sheet.
CHAPTER 7

SYSTEM PERFORMANCE

This chapter has been divided into two major sections - objective and subjective quality measures. All the results presented here have been obtained using the techniques described in chapter 6. For the subjective results the tests, exactly as described in the previous chapter, have been used. For the objective results the major tests are the same as those described in chapter 6. However, for some of the tests (e.g. section 7.2.4 which shows the effect of using the feedforward mode of calculating the syllabic factor for the hybrid codecs) slight modifications were made to either the algorithms or the measurement techniques. In cases where the techniques differ from those described previously the changes have been detailed prior to presenting the results.

7.1 Autocorrelation Function of a Speech Signal.

Fig. 7.1 shows the autocorrelation function for the speech segment used in the tests detailed here. The purpose of this is simply to confirm that the correlation between adjacent samples is in fact within the limits given in chapter 2 (i.e. \( C(7) > 0.5 \)).

7.2 Objective Results.

Before the codecs could be evaluated it was necessary to optimize the parameters affecting the performance of the circuits so as to enable direct comparisons between the three to be made.
Sampling frequency: 16 kHz

Fig. 7.1: Autocorrelation Function of sampled speech.
CHAPTER 7 : SYSTEM PERFORMANCE

7.2.1 Circuit Optimization.

The basic procedure for evaluating the optimum value of a circuit parameter involves setting all others to a fixed value and then varying the parameter in question over some range. The parameters are optimized with respect to the SQNR. Initially it would appear that, based on the fact that slope overload noise is subjectively preferable to granular noise\textsuperscript{293}, the SNG would be a better measure of codec quality. However, the SQNR is used as the quality measure for optimization purposes because, when a delta modulator is tracking the input signal optimally, all the noise is granular (this stems from the nature of the operation of the delta modulator). By optimizing with respect to SNG it would be possible to achieve an encoder which would be virtually permanently slope overloaded. Hence the use of SQNR.

The parameters which have the main influence on the system performance are the quantizer minimum step size, the time constants of the prediction and syllabic filters, and the scale factor of the syllabic compandor.

In all cases (unless otherwise specified) the simulation runs were made using a real 10 second segment of male speech and a noiseless channel.
7.2.1.1 Syllabic Gain Factor ($\alpha$).

The scale factor $\alpha$ for the long-term basic step size controls, to a large extent, the quantization noise produced by the encoder. If $\alpha$ were too small, the step size would be too small and the encoder would operate in the slope overload mode for most of the time. On the other hand, if $\alpha$ were too large, the output of the local decoder would oscillate about the input signal as a result of the large basic step size. In the extreme case this could lead to the encoder becoming unstable.

Fig. 7.2 shows the variation of the signal-to-noise ratio as a function of $\alpha$ for the three hybrid codecs.

For the HCDM encoder an optimum value for $\alpha$ of 0.8 correlates with the findings of Un et al.$^{[17,18]}$, while the peak SQNR's for the SHCDM and MSHCDM encoders occur at values of 0.2 and 0.25 respectively.

An interesting feature of the curves in fig. 7.2 is the relative insensitivity of the SQNR's of the SHCDM and MSHCDM encoders to the variation of $\alpha$. The advantage of this is that a value of $\alpha$ other than the optimum value is likely to be less drastic, in terms of performance degradation, for these two encoders than for the HCDM encoder. This would enable the values for $\alpha$ to be chosen
Fig. 7.2: SQNR as a function of Alpha
Based on other considerations as well (e.g. ease of implementation).

Based on the preceding discussion, it would appear that a value for $\alpha$ of 0.25, instead of 0.2, for the SHCDM encoder would be satisfactory. For both the SHCDM and MSHCDM encoders the digital implementation of the multiplication by $\alpha$ is easily realised as a two-place arithmetic shift. However, a value of 0.8 (for the HCDM encoder) is not very easily realised using digital methods.

Referring to fig. 7.3, the variation of the segmental signal-to-noise ratio as function of $\alpha$, the peak of SNRSEG for the HCDM encoder occurs, not at 0.8 but, at 0.7. As the segmental measures have been found to correlate better with the subjective results\textsuperscript{43} it would appear that this would be a better value to use. Alternatively, a compromise value of 0.75 would have the advantage of simplifying the digital implementation of the multiplication operation (0.75 could be realised by dividing by four, a process requiring a two-place arithmetic shift, and subtracting the result of this from the original value). However, the SNRSEG measure was not used by Un et al\textsuperscript{17,18} and so no comparison with published findings can be made in this regard. In the case of the SHCDM and MSHCDM encoders the results of the SNRSEG tests serve to confirm the findings of the SQNR tests.
Fig. 7.3: SNRSEG as a function of Alpha
7.2.1.2 Prediction Time Constant (1/\beta).

The high degree of correlation between adjacent samples of speech sampled at 16 kHz (cf. fig. 7.1) suggests that the optimum value for the leakage factor of the prediction filter would lie somewhere between 0.8 and 1.0. The leakage factor \( L \) is related to the prediction time constant \( 1/\beta \) by:

\[
L = \exp(-\beta T).
\]

Fig. 7.4 shows the variation of SQNR as a function of \( L \). These curves were determined using the values of \( \alpha \) established in the previous section. The syllabic time constant was set to 5 msec and the minimum step size to 5 mV. From these curves it is evident that a value for \( L \) of 0.96 would suffice for all three encoders. This is not entirely surprising as it should be remembered that this leakage factor is derived more from the characteristics of the speech signal than from those of the encoding algorithms (although there is some inter-relation). From this value it is possible to calculate the value of the prediction time constant.

\[
\beta = -(1/T) \ln(L)
\]

\[
\therefore \beta = -16.10^3 \ln(0.96)
\]

\[
\therefore \beta = 1/(1.53 \text{ msec}).
\]

NOTE: In fig. 7.4 the SQNR curve for the SHCDM encoder was only
$L = \exp\left(-\frac{1}{\text{Beta} \times \text{T}}\right)$

Fig. 7.4: SQNR as a function of the Leakage Factor ($L$)
Fig. 7.5: SQNR as a function of the Syllabic Period.

Sample periods = 12.5 msec.
Fig. 7.6: SNG as a function of the Minimum Step Size
plotted for values of $L$ between 0.92 and 1.00 as the curve for the MSHCDM encoder was used to estimate the likely range within which the peak value of SQNR would occur.

7.2.1.3 Syllabic Companding Period. Based on the findings of Un et al.\textsuperscript{17, 18} it was not expected that the value for the syllabic time constant would be critical. Fig. 7.5 demonstrates that a value of between 2 msec and 20 msec would suffice.

The tests to obtain the curves shown in Fig. 7.5 were conducted with $\alpha$ and $\beta$ set to the values determined previously. The minimum step size was limited to the numerical value of 1 (i.e., 5 mV), and a 3 second speech segment was used.

Owing to the fact that, between the limits mentioned, the value for the syllabic time constant is not critical a value of 5 msec was chosen, thereby enabling a direct comparison with the findings of Un et al to be made. Because the syllabic compandor is common to all three of the hybrid codecs, the factors influencing their relative performances are not likely to be affected by this section.
CHAPTER 7: SYSTEM PERFORMANCE

7.2.1.4 Minimum Step Size ($S_0$).

The measure used to investigate the effect of using a minimum step size other than 1 was the SNG. If a large minimum step size is used then, during steady state portions of the input signal especially, a large granular noise component is to be expected. Fig. 7.6 shows the variation of the SNG as a function of the minimum step size. No limit was placed on the maximum step size and $\alpha$, $\beta$, etc. were set to the values determined previously.

Based on the results plotted in fig. 7.6 a minimum step size of 1 (ie. 5 mV) was used for all subsequent simulation runs and tests.

7.2.2 Dynamic Ranges of the HCDM, SHCDM and MSHCDM Encoders.

As was discussed in earlier chapters, the problem with instantaneous companding is that, for different speech signals the optimum quantizer step-size ranges may be different (cf. fig. 7.7). The hybrid companding schemes serve to adjust the step-size adaptation range according to the RMS input signal level. Fig. 7.8 shows the variation of the SQNR as a function of the RMS signal level for a 10 second segment of speech. The input signal level was varied over a range of 80 dB and a step size range of 60 dB was used.
Fig. 7.8: Dynamic Range of the hybrid encoders.
Fig. 7.9a: SNRSEG as a function of Signal Amplitude.
CHAPTER 7: SYSTEM PERFORMANCE

From the curves shown in fig. 7.8 it can be seen that there is no appreciable difference in the SQNR's of the three codecs over the dynamic range. The results presented for the HCDM encoder correlate very closely with those published by Un et al\(^{17,18}\). Although the HCDM encoder has a marginally higher SQNR it does not necessarily imply that it will be subjectively better than the other two hybrid codecs. Because the segmental values of SQNR show a better correlation with subjective results\(^{14,33}\) it was decided to plot these curves to obtain a better indication of what might be expected of the subjective tests. The variation of the segmental signal-to-noise ratio (SNRSEG) as a function of the RMS signal level is shown in fig. 7.9a.

From the segmental results it would appear that the two new codecs (SHCDM and MSHCDM) might offer an improvement in performance over that of the HCDM codec.
Encoder type: MSHCDM

Max Step Size = 10000 100000 400000

Fig. 7.9b: Variation of Dynamic Range with maximum step size.
Encoder Type: MSHCDM

Maximum Step Size = 1000, 10000, 40000

SNRSEG (dB)

Relative Signal Amplitude

Fig. 7.9c: Variation of SNRSEG with maximum step size.
7.2.2.1 Variation of the Step Size Range.

The roll-off of the SQNR at higher values of the RMS signal level is caused by the limit placed on the step size range. If no limit were placed on this parameter the SQNR curves would exhibit a roll-off only at the lower end of the RMS signal level. By increasing the step size range the level at which the "upper" roll-off point occurs can be increased. This effect is illustrated in figs. 7.9b and 7.9c (the effect is shown only for the MSHCDM codec).

7.2.2.2 Variation of Granular SNR.

Based on the previous discussion relating to SNG as a performance measure, it should be noted that, having optimized the codecs with respect to SQNR, this measure (i.e. SNG) could be used to give some indication of the relative performances of the codecs. Using the fact that slope overload noise is preferable to granular noise\textsuperscript{293} the variation of SNG as a function of the input signal level is plotted in fig. 7.10. From these results, as well as the segmental signal-to-granular noise results (SNGSEG) shown in fig. 7.11, it can be seen that both the SHCDM and MSHCDM codecs offer better performance than the HCDM codec over the bulk of the dynamic range. This assessment is particularly relevant in the light of the fact that, particularly for adaptive systems, the SNG and SNGSEG measures are the best predictors of overall subjective
Fig. 7.10: SNG as a function of Signal Amplitude.
Fig. 7.11: SNGSEG as a function of Signal Amplitude.
NOTE: The Signal-to-Slope Overload Noise ratio can be determined from the SNG using the relationship:

\[ SNO = -10 \log \left( 10^{-5 S N R / 10} - 10^{-5 S N G / 10} \right) \]

7.2.3 Cross-correlation Functions

The use of the auto- and cross-correlation functions as a measure of performance is related to the use of the PSD function for determining the quality of a waveform coder via the Fourier Transform (cf. chapter 2 section 1.2.5).

As the waveform coder attempts to preserve the spectral characteristics of the input signal by producing a facsimile copy thereof, so the cross-correlation function gives a measure of the degree to which this has been achieved. The auto-correlation function and the PSD (of the input signal) form a Fourier Transform pair (ie. the PSD function of the input signal can be determined from the auto-correlation function via Fourier Transform techniques). Similarly, the auto-correlation function of the output signal contains information pertaining to the PSD of the output:

The cross-correlation function on the other hand is the correlation between the input and output signals. If the auto-
Correlation between original and processed speech signals

Encoder type: HCDM

Sampling frequency: 16 kHz

Fig. 7.12a: Cross-correlation Function of sampled speech.
Fig. 7.12b: Cross-correlation Function of sampled speech.

Correlation between original and processed speech signals.

Encoder type: SHCDM

Sampling frequency: 16 kHz
Correlation between original and processed speech signals

Encoder type: MSHCDM

Sampling frequency: 16 kHz

Fig. 7.12c: Cross-correlation Function of sampled speech.
and cross-correlation functions are identical then the output signal represents an exact replica of the input signal. So, by comparing the two correlation functions it is possible to determine the degree to which the PSD of the input has been preserved.

The cross-correlation functions are plotted in fig. 7.12a, b, and c. The use of the cross-correlation functions is not intended to provide a rigorous method of comparing the relative performances of the different codecs, but rather an approximate indication of the degree to which the PSD of the input has been preserved (an alternative method would be to actually calculate the PSD function for the input and output speech signals).

7.2.4 Using the Feedforward Mode for Syllabic Companding.

In previous discussion it was mentioned that in order to preserve channel bandwidth the syllabic companding factor was calculated using the decoded signal instead of the actual input signal. Fig. 7.13 shows the effect of using the feedforward mode for calculating the syllabic factor (ie. using the actual input speech signal). This has been done for the MSHCDM codec alone because the effect is likely to be similar for the other two hybrid codecs.
From fig. 7.13 it is clear that there is no particular advantage to be gained from using the feedforward mode, especially when weighed against the additional bandwidth required to transmit the syllabic information.

7.2.5 Extending the Constant Factor Logic for HCDM.

In the hybrid compandor of Magill et al\textsuperscript{24} the constant factor logic utilised five quantizer error signals (i.e. $b_n, \ldots, b_{n-4}$) as opposed to the three ($b_n, \ldots, b_{n-2}$) used in the HCDM encoder.

The effect of this extended companding logic has been investigated with the dynamic range curves (SQNR vs RMS signal level) given in fig. 7.14. The extended logic is given in Table 7.1.

Table 7.1: Instantaneous Companding Logic Rule.

<table>
<thead>
<tr>
<th>$b^n$</th>
<th>$b^{n-1}$</th>
<th>$b^{n-2}$</th>
<th>$b^{n-3}$</th>
<th>$b^{n-4}$</th>
<th>Multiplication factor ($k_n$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>1.50</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td>1.50</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>-</td>
<td></td>
<td></td>
<td>1.00</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>+</td>
<td></td>
<td></td>
<td>1.00</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td>0.66</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td>1.00</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>0.66</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td>1.00</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>-</td>
<td></td>
<td></td>
<td>0.66</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td></td>
<td></td>
<td>0.66</td>
</tr>
</tbody>
</table>
Fig. 7.13: SQNR as a function of RMS signal amplitude.

Encoder type: MSHCDM
Feedforward mode for Syllabic companding
Fig. 7.14: SQNR as a function of RMS signal level.

Encoder type: HCDM
Extended Constant Factor Logic

- : Extended CF Logic
- : Normal 2nd-order CF Logic

Relative Signal Amplitude
CHAPTER 7: SYSTEM PERFORMANCE

From the results plotted in Fig. 7.14 it is clear that there is very little advantage to be gained from extending the logic required to generate the instantaneous companding factor. Hence, in order to minimize the circuitry that would be required to implement this module the second-order constant factor compandor (using the present binary output and the previous two) is used.

7.2.6 High Quality Microphone vs Telephone Handset.

For all of the objective tests a high quality microphone has been used whereas for the subjective tests the speech was recorded using the microphone contained in a standard telephone handset. The frequency characteristics of the two are significantly different (cf. Fig. 5.2).

The performance of the three hybrid codecs has been measured using the telephone handset microphone, and the results are shown in Fig. 7.15. A reduction in the peak value of the SQNR is the result of the removal, by the telephone microphone (which exhibits band-pass characteristics), of the lower frequency components of the speech signal. It is these low frequency components which the delta modulators code the easiest and hence, when they are removed the 3 - 4 dB reduction in SQNR is experienced.
Microphone in a Telephone handset used.

Fig. 7.15: Dynamic Range of the hybrid encoders.
Fig. 7.16: Step Size Histograms
Fig. 7.17: SQNR as a function of Maximum Step Size.
7.2.7 Step Size Histograms

Fig. 7.16 shows the step size histograms for the three codecs. The interesting feature is that whereas the SHCDM and MSHCDM encoders show a more-or-less uniform decrease (in number of occurrences) as the step size increases, the HCDM has a much wider spread of step sizes in the -100 to +100 range.

Referring to fig. 7.17 (the variation of the SQNR as a function of the maximum permissible step size) it is significant that for the MSHCDM encoder no increase in SQNR is achieved by increasing the permissible maximum step size beyond the -300 to +300 range. Both the HCDM and SHCDM encoders appear to offer an improvement in performance if the maximum step sizes are limited to the -400 to +400 range. However, taking into account the fact that the peaks in the curves for these two encoders are likely to vary with different speakers and speech segments it is probably better to consider only the general trend exhibited by these curves (i.e., for the SHCDM encoder a maximum step size of 700 could be used, while for the HCDM encoder a value of 900 could be used).

The significance of this is that for the MSHCDM encoder an 8-bit step size word (giving a maximum step size of $\gamma_{\max} = 256$) could be used without significantly reducing the quality of the decoded speech. The HCDM and SHCDM encoders would require a
10-bit step size word length (giving a maximum step size of $|\gamma_{\text{max}}| = 1024$).
CHAPTER 7 : SYSTEM PERFORMANCE

7.3 Subjective Results.

7.3.1 Significance of Results.

Once the subjective tests have been performed and the various scores for the three codecs have been determined it is necessary to be able to evaluate the significance of the different statistical results. In other words, it is desirable to be able to determine whether the differences in the statistics for the three codecs are significant or not.

The test for significance is performed using the Z-score test\(^{35}\). The Z-score is defined as:

\[
Z = \frac{(P_1 - P_2)}{\sigma_z},
\]

where

- \(P_1\) is the proportion of listeners who (a) heard the sentence correctly, (b) recognised the speaker, or (c) judged the test sentence to be better than the reference for hybrid codec 1,
- \(P_2\) is the proportion for hybrid codec 2,
- \(\sigma_z = \sqrt{pq(1/N_1 + 1/N_2)}\),

where

- \(N_1\) is the number of tests (ie. listeners x sentences) for hybrid codec 1,
- \(N_2\) is the number of tests for hybrid codec 2,
CHAPTER 7: SYSTEM PERFORMANCE

\[ p = \left( \frac{N_1 \cdot P_1 + N_2 \cdot P_2}{N_1 + N_2} \right) \]

\[ q = 1 - p. \]

In all cases the table listed in Appendix G is used and a value of 5\% is used for the degree of significance. In other words, we can be sure that, with a 5\% margin of confidence, our assumptions regarding the significance of the statistical results are correct.

7.3.2 Intelligibility Test.

The results of the intelligibility tests are given in Table 7.2.

Table 7.2: Codec intelligibility scores.

<table>
<thead>
<tr>
<th>CODEC</th>
<th>CORRECT</th>
<th>PARTLY CORRECT</th>
<th>INCORRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCDM</td>
<td>239</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>SHCDM</td>
<td>197</td>
<td>45</td>
<td>8</td>
</tr>
<tr>
<td>MSHCDM</td>
<td>203</td>
<td>44</td>
<td>3</td>
</tr>
</tbody>
</table>

The values listed above reflect the product of the number of listeners and the number of sentences heard correctly, partly correct or incorrectly for each listener. Hence, for 50 listeners and 5 sentences per codec the maximum possible value is 250.

From these results it would appear at first that the HCDM codec is significantly better than the other two. However, in the majority
CHAPTER 7: SYSTEM PERFORMANCE

of cases for the sentences which were scored 'PARTLY CORRECT' the
listeners heard one word incorrectly, and this was invariably the
first word of the sentence. In all such cases the listeners
interpreted the sentence correctly (i.e. they understood the
semantic content). In Table 7.3 the 'CORRECT' and 'PARTLY
CORRECT' scores have been combined.

Table 7.3: Modified intelligibility scores.

<table>
<thead>
<tr>
<th>CODEC</th>
<th>CORRECT &amp; PARTLY CORRECT</th>
<th>INCORRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCDM</td>
<td>249</td>
<td>1</td>
</tr>
<tr>
<td>SHCDM</td>
<td>242</td>
<td>8</td>
</tr>
<tr>
<td>MSHCDM</td>
<td>247</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 7.4: Z-scores for codec comparisons.

<table>
<thead>
<tr>
<th>COMPARISON</th>
<th>P_1</th>
<th>P_2</th>
<th>N_1</th>
<th>N_2</th>
<th>Z-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCDM &amp; SHCDM</td>
<td>0.996</td>
<td>0.968</td>
<td>250</td>
<td>250</td>
<td>2.355</td>
</tr>
<tr>
<td>HCDM &amp; MSHCDM</td>
<td>0.996</td>
<td>0.988</td>
<td>250</td>
<td>250</td>
<td>1.004</td>
</tr>
<tr>
<td>SHCDM &amp; MSHCDM</td>
<td>0.968</td>
<td>0.988</td>
<td>250</td>
<td>250</td>
<td>1.524</td>
</tr>
</tbody>
</table>

Then, using the Z-score method, and referring to Table 7.4, it can
be seen that there is a significant difference between the scores
of the SHCDM and HCDM codecs. The main purpose of the intelligi-
bility test was however to establish that this measure was at a
satisfactory level for the isopreference tests, and not so much
CHAPTER 7: SYSTEM PERFORMANCE

for the purpose of comparing the three codecs. From the results shown in Tables 7.2 and 7.3 it can be seen that the three codecs do have a sufficiently high level of intelligibility to facilitate the isopreference tests.

7.3.3 Speaker Recognition

As with the intelligibility tests the main purpose of this test was to establish that each of the codecs in question had a satisfactory level of speaker-identity retention. It should be noted that none of the speakers were known to the listeners before the tests and hence the results are likely to be 'coloured' by a certain learning factor (i.e. as the listeners become more familiar with the voices so it becomes easier for them to identify the speaker).

The results presented in Table 7.5 reflect the number of times the speakers were identified correctly or incorrectly for each codec. The 'NOT SURE' score was used to represent the cases where, had the listener been more familiar with the speakers, a 'CORRECT' result would possibly have been recorded.

Using the Z-score as a measure of the significance (Table 7.6) it can be seen that there is no significant difference between the three codecs as far as 'CORRECT' and 'INCORRECT' sentences are
Table 7.5: Speaker recognition scores.

<table>
<thead>
<tr>
<th></th>
<th>CORRECT</th>
<th>NOT SURE</th>
<th>INCORRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCDM</td>
<td>148</td>
<td>55</td>
<td>47</td>
</tr>
<tr>
<td>SHCDM</td>
<td>153</td>
<td>55</td>
<td>42</td>
</tr>
<tr>
<td>MSHCDM</td>
<td>139</td>
<td>76</td>
<td>35</td>
</tr>
</tbody>
</table>

concerned and that only in the case of the MSHCDM is there any significant difference with the other two, for the 'NOT SURE' score. However, considering that it is quite common for listeners to have difficulty recognising even familiar voices (using telephone handsets), and taking into account the fact that, as

Table 7.6: Z-scores for speaker recognition tests.

<table>
<thead>
<tr>
<th>COMPARISON</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$N_1$</th>
<th>$N_2$</th>
<th>Z-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORRECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCDM &amp; SHCDM</td>
<td>0.592</td>
<td>0.612</td>
<td>250</td>
<td>250</td>
<td>0.457</td>
</tr>
<tr>
<td>HCDM &amp; MSHCDM</td>
<td>0.592</td>
<td>0.556</td>
<td>250</td>
<td>250</td>
<td>0.814</td>
</tr>
<tr>
<td>SHCDM &amp; MSHCDM</td>
<td>0.612</td>
<td>0.556</td>
<td>250</td>
<td>250</td>
<td>1.270</td>
</tr>
<tr>
<td>NOT SURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCDM &amp; SHCDM</td>
<td>0.220</td>
<td>0.220</td>
<td>250</td>
<td>250</td>
<td>0.000</td>
</tr>
<tr>
<td>HCDM &amp; MSHCDM</td>
<td>0.220</td>
<td>0.304</td>
<td>250</td>
<td>250</td>
<td>2.136</td>
</tr>
<tr>
<td>SHCDM &amp; MSHCDM</td>
<td>0.220</td>
<td>0.304</td>
<td>250</td>
<td>250</td>
<td>2.136</td>
</tr>
<tr>
<td>INCORRECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCDM &amp; SHCDM</td>
<td>0.188</td>
<td>0.168</td>
<td>250</td>
<td>250</td>
<td>0.584</td>
</tr>
<tr>
<td>HCDM &amp; MSHCDM</td>
<td>0.188</td>
<td>0.140</td>
<td>250</td>
<td>250</td>
<td>1.449</td>
</tr>
<tr>
<td>SHCDM &amp; MSHCDM</td>
<td>0.168</td>
<td>0.140</td>
<td>250</td>
<td>250</td>
<td>0.867</td>
</tr>
</tbody>
</table>
CHAPTER 7: SYSTEM PERFORMANCE

mentioned previously, the speakers were totally unfamiliar to the listeners, the results of these tests indicate a satisfactory reproduction of the speakers' voices by the three codecs.

Having now established that the retention of the speaker's identity and the intelligibility for the three codecs are at a satisfactory level the results of the isopreference tests are presented.

7.3.4 Isopreference Tests.

The results for these tests are presented in both graphical and tabular form (Fig. 7.18 & Table 7.7). In Table 7.7 the differences between the three hybrid codecs have been tested for significance for each level of the SNR of the reference system. This was done because in the isopreference tests themselves the three codecs were not compared directly with each other, but to the reference system. From the method outlined in chapter 6 it should be clear that the most important region of the curves of fig. 7.18 is the area in which the codecs are isopreferent with the reference system (this area has been shaded).

Referring to fig. 7.18 it can be seen that the three curves have the characteristics that one would expect - ie. when the SNR of the reference system is very low the majority of the listeners
Table 7.7: Results of Isopreference tests.

<table>
<thead>
<tr>
<th>COMPARISON</th>
<th>SNR (dB)</th>
<th>P1</th>
<th>P2</th>
<th>N1</th>
<th>N2</th>
<th>Z-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCDM &amp; SHCDM</td>
<td>-6</td>
<td>1.00</td>
<td>0.96</td>
<td>50</td>
<td>50</td>
<td>1.429</td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>0.88</td>
<td>1.00</td>
<td>50</td>
<td>50</td>
<td>2.526</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.50</td>
<td>0.80</td>
<td>50</td>
<td>50</td>
<td>3.145</td>
</tr>
<tr>
<td></td>
<td>+2</td>
<td>0.24</td>
<td>0.50</td>
<td>50</td>
<td>50</td>
<td>2.693</td>
</tr>
<tr>
<td></td>
<td>+4</td>
<td>0.32</td>
<td>0.28</td>
<td>50</td>
<td>50</td>
<td>0.436</td>
</tr>
<tr>
<td></td>
<td>+6</td>
<td>0.22</td>
<td>0.06</td>
<td>50</td>
<td>50</td>
<td>2.306</td>
</tr>
<tr>
<td></td>
<td>+8</td>
<td>0.20</td>
<td>0.10</td>
<td>50</td>
<td>50</td>
<td>1.400</td>
</tr>
<tr>
<td></td>
<td>+10</td>
<td>0.08</td>
<td>0.04</td>
<td>50</td>
<td>50</td>
<td>0.842</td>
</tr>
<tr>
<td>HCDM &amp; MSHCDM</td>
<td>-6</td>
<td>1.00</td>
<td>0.98</td>
<td>50</td>
<td>50</td>
<td>1.005</td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>0.88</td>
<td>0.98</td>
<td>50</td>
<td>50</td>
<td>1.960</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.50</td>
<td>0.76</td>
<td>50</td>
<td>50</td>
<td>2.693</td>
</tr>
<tr>
<td></td>
<td>+2</td>
<td>0.24</td>
<td>0.84</td>
<td>50</td>
<td>50</td>
<td>6.019</td>
</tr>
<tr>
<td></td>
<td>+4</td>
<td>0.32</td>
<td>0.56</td>
<td>50</td>
<td>50</td>
<td>2.417</td>
</tr>
<tr>
<td></td>
<td>+6</td>
<td>0.22</td>
<td>0.38</td>
<td>50</td>
<td>50</td>
<td>1.746</td>
</tr>
<tr>
<td></td>
<td>+8</td>
<td>0.20</td>
<td>0.32</td>
<td>50</td>
<td>50</td>
<td>1.368</td>
</tr>
<tr>
<td></td>
<td>+10</td>
<td>0.08</td>
<td>0.32</td>
<td>50</td>
<td>50</td>
<td>3.000</td>
</tr>
<tr>
<td>SHCDM &amp; MSHCDM</td>
<td>-6</td>
<td>0.96</td>
<td>0.98</td>
<td>50</td>
<td>50</td>
<td>0.586</td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>1.00</td>
<td>0.98</td>
<td>50</td>
<td>50</td>
<td>1.005</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.80</td>
<td>0.76</td>
<td>50</td>
<td>50</td>
<td>0.483</td>
</tr>
<tr>
<td></td>
<td>+2</td>
<td>0.50</td>
<td>0.84</td>
<td>50</td>
<td>50</td>
<td>3.615</td>
</tr>
<tr>
<td></td>
<td>+4</td>
<td>0.28</td>
<td>0.56</td>
<td>50</td>
<td>50</td>
<td>2.836</td>
</tr>
<tr>
<td></td>
<td>+6</td>
<td>0.06</td>
<td>0.38</td>
<td>50</td>
<td>50</td>
<td>3.862</td>
</tr>
<tr>
<td></td>
<td>+8</td>
<td>0.10</td>
<td>0.32</td>
<td>50</td>
<td>50</td>
<td>2.701</td>
</tr>
<tr>
<td></td>
<td>+10</td>
<td>0.04</td>
<td>0.32</td>
<td>50</td>
<td>50</td>
<td>3.644</td>
</tr>
</tbody>
</table>

prefer the test system (e.g., for a SNR of -6 dB P2 for the MSHCDM encoder is 0.98 - i.e. 49 listeners preferred the test system), while on the other hand, when the SNR of the reference system is increased sufficiently the majority of the listeners prefer the reference system.
Fig. 7.18: Psychometric curves for the Isopreference test.
CHAPTER 7: SYSTEM PERFORMANCE

From the curves shown in fig. 7.18 and from the results shown in Table 7.7 it should be clear that the MSHCDM encoder's performance is significantly better than the other two. The MSHCDM codec has a subjective SNR of approximately 4.8 dB compared to the SNR's of the HCDM and SHCDM codecs of 0 dB and 2 dB respectively. Hence, both the hybrid codecs using Song Voice instantaneous adaptation show an improvement in quality over the HCDM codec of Un et al.

7.4 Summary

The objective results show that the performance of the MSHCDM codec is, at worst, equivalent to that of the HCDM codec. The Granular Signal-to-Noise ratios indicate that a moderate improvement is to be had using the MSHCDM codec. If used to predict the outcome of the subjective tests it is significant that the SNG and SNGSEG measures indicate that the best of the three codecs is the MSHCDM codec, followed by the SHCDM codec, and then the HCDM codec. As has been discussed, this has in fact been born out by the subjective results. Comparing the SNG and SNGSEG curves of the three codecs it is noticeable that for low RMS levels the HCDM codec is better. Essentially this means that the 'idle channel' performance of this codec is better. However, despite this fact, subjectively the SHCDM and MSHCDM codecs rated better. This would appear to indicate that the improvement in 'idle channel' performance offered by the HCDM codec is not perceptually
CHAPTER 7: SYSTEM PERFORMANCE

significant.

Overall, the most significant results are those obtained from the subjective tests. The results of these indicate a clear advantage offered by the MSHCDM codec. This also correlates with the results of the SNG tests in that the codec producing the least amount of granular noise is likely to be preferred to the others (assuming of course that all the codecs have been previously optimized).

Basically the difference in granular noise powers is caused by the relatively slower change in step size of the Song Voice compandor as compared to the Constant Factor compandor (the Song Voice step size varies linearly whereas the Constant Factor step size varies exponentially). The important point arising out of the difference between the natures of the step size adaptations is that:

(i) During silent periods, or very low amplitude signal periods, the Song Voice compandor produces a binary output signal 110011001100 ... 110011. ... (resulting in a step size sequence +1 +2 -1 -2 +1 +2 -1 -2 ... ) whereas the Constant Factor compandor produces a binary output sequence 1010101010 (resulting in a step size sequence of +1 -1 +1 -1 +1 -1 ...). Hence, during these periods the MSHCDM codec produces more granular noise than the HCDM codec.

PAGE 7-45
(ii) On the other hand, during large signal amplitude periods the exponential step size adaptation of the Constant Factor compandor produces considerably more noise than the linearly adapting Song Voice compandor.

Considering the fact that voiced speech is subjectively more susceptible to the addition of granular noise than unvoiced speech (cf. chapter 2), and the fact that voiced speech has a higher probability of occurrence than unvoiced speech (chapter 2), the performance of the codecs during speech portions with high RMS levels is probably more significant.

7.4.1 Implementation Considerations.

The implementation of the three algorithms is another significant factor in deciding between them. The stated aim of the research project is to produce an integrated circuit version of the delta modulation speech codec. It is important therefore that, in order to reduce the cost of a system using the codec, the codec algorithm be efficient in terms of hardware as well. In this regard the MSHCDM codec offers a significant advantage over the HCDM codec in that one less multiplication operation is required. This is due to the fact that the instantaneous companding factor of the MSHCDM codec is generated by addition as opposed to multiplication - the multiplication of $A$ by $\gamma_0$ and $b_{n-1}$ is very
simple as \( y_0 \) is 1 and \( b_{n-1} \) is either +1 or -1 (the instantaneous companding is given by: \( y_n = 1 y_{n-1} b_n + y_0 \Lambda b_{n-1} \)). Also, the multiplication by the factor \( \Lambda \) (i.e. \( \Lambda = \alpha E \)) is much simpler to achieve in the MSHCDM case than the HCDM case. This is because to implement a multiplication by 0.25 requires a simple two-place arithmetic shift whereas to implement a multiplication by 0.8 is far more complicated.

Based on the results of the objective and subjective tests, and considering that the MSHCDM codec involves a significantly less complex hardware implementation, we consider this codec to offer an improved method for implementing a hybrid companding delta modulator in digital hardware.
CHAPTER 8

PROSPECTIVE FUTURE DEVELOPMENT

Having established the superiority of the MSHCDM codec it is necessary, in conclusion, to outline the direction in which future research developments could take place.

8.1 Extending to a Variable-Rate System

There are basically two possible methods for extending the improvements in performance offered by the hybrid companding schemes discussed in this work. Overall system improvement could be achieved at the expense of considerable complexity, either to achieve an improvement in the output speech quality, or in order to reduce the transmission bandwidth while maintaining the speech quality. An alternative technique, which can be implemented with a moderate increase in complexity, involves using variable-rate sampling.

It has been found that by combining the hybrid companding technique with a variable-rate sampling scheme an improvement of between 3 and 4 dB can be realised.\(^4\) The sampling rate of the encoder is adjusted according to the input speech activity. By using a buffer the encoder binary output signal can be transmitted at a fixed rate.

By extending this variable-rate system to a multisubscriber system an improvement in SQNR of approximately 10 dB, over the conven-
tional single-subscriber HCDM system, can be achieved\cite{46}. This improvement is realised by employing a dynamic buffer control algorithm which utilises statistical multiplexing; and a variable-rate sampling scheme which adjusts the sampling rate according to both the speech activity and the buffer occupancy.

An important element of the variable-rate system design is the buffer size. If the buffer is too small overflow will occur frequently during high speech activity periods. However, if the buffer is too large, the resultant delay experienced at the decoder gives rise to unnatural output speech. The use of statistical multiplexing (which is similar in concept to TASI) necessitates the design of a buffer control scheme that reduces the probability of overflow.

Based on the findings of Un and Kim et al\cite{45,46}, as regards the multisubscriber variable-rate system, it was envisaged that a similar degree of improvement for the MSHCDM system could be attained if the fixed-rate version, developed to date, were extended to a variable-rate one. Preliminary work on this has already been carried out (by S.C.Hall, as part of a postgraduate research project) with the initial results confirming the findings of Un et al\cite{45}. Further work still needs to be done on developing a suitable efficient buffer control algorithm.
8.2 Optimization of the Algorithm Implementation.

All investigations to date have involved implementing the codec algorithms as detailed previously in this work. However, because the eventual aim of this research is to implement an Application Specific Integrated Circuit (ASIC) version of the codec it will be necessary to optimize the design in terms of, for instance, the number of gates required to realise a certain function.

It is envisaged that the syllabic compandor used in the three hybrid codecs could be replaced by a simpler rectifier-low-pass filter combination.

ie. the syllabic companding function

\[ \Lambda = \alpha E = \alpha \sum_{i=1}^{M-1} (X_i - X_{i-1})^2 / (M - 1) \]

could be replaced by a simple low-pass function

\[ \Lambda = \alpha E_t \]

which is updated at the syllabic rate.

where

\[ E_t = L \cdot E_{t-1} + A_t \]

which is updated at the sampling rate.

\( A_t \) is the absolute value of \( X_t \) (the approximation signal),

and

\[ L = \exp(-\beta_t T) \],

\( \beta_t \) being the inverse of the syllabic low-pass filter time constant.
CHAPTER 8: PROSPECTIVE FUTURE DEVELOPMENT

Work would need to be done on optimising and evaluating the design of the module to perform this function.

8.3 Correlating the Results of Objective and Subjective Tests.
Perhaps the single most important development in the field of speech codec performance analysis is the development of a single test procedure that would enable different codecs to be rated on a single quality scale according to their relative performances. Ideally it would be desirable to have a single objective measure from which the subjective analyses results could be accurately predicted.

The subjective test procedure used here followed the recommendations of the IEEE closely and was, we believe, adequate enough to illustrate the superiority of the MSHCDM codec over the HCDM codec. However, in the course of the tests a few points arose which should be highlighted to prevent possible confusion.

Firstly, it is common practice for the speech signals used for the purpose of both subjective and objective tests to be recorded using high quality microphones. This practice would be acceptable if the eventual system were to be incorporated in a high quality dedicated communications link (where it is not unlikely that high quality microphones might be used). However, if the system were
CHAPTER 8: PROSPECTIVE FUTURE DEVELOPMENT

to be incorporated into an extensive commercial communications network it is more meaningful to conduct these tests using a microphone, and playback equipment, characteristic of the devices used in the network. In this way more meaningful objective results would be obtained - ie. the results would be indicative of the quality of a realistic implementation of the system under discussion.

8.3.1 Inclusion of Network Parameters.

In this work the performance of the codecs alone have been considered. However, when the codecs are included in a network a number of factors are likely to affect their performance.

Firstly, if a packet switched network is used, the effect of packet losses on the codecs' performance would need to be ascertained. Secondly, if the variable-rate system mentioned previously is implemented then the effect of buffer overflows would need to be investigated and the buffer control algorithm modified accordingly. It is common for simple bit error performance ratings to be quoted. However, in the network systems mentioned above the channel bit error rate is not likely to be the most significant influence on performance. In any event, it is quite common for digital channels with bit error rates of $10^{-7}$, or better, to be used, and most waveform codecs perform adequately...
under these conditions.

Another factor that could be investigated is the design of a suitable protocol for voice packet switching networks.

In conclusion it should be stated that the primary objective at this stage is to characterise the performance of the hybrid codecs using a variable-rate system. Once that has been concluded the ASIC implementation should be finalised. This, in turn, will enable the scope of the research project to be extended to network considerations, such as variable-rate multisubscriber systems, packet switched protocols, etc.
REFERENCES

- publication of the Defence Communication Agency, Department of Defence, U. S. A.

(2) T. S. Lamba, M. N. Faruqui : "Intelligible voice communication through adaptive delta modulation at bit rates lower than 10 kbps."

(3) R. E. Crochiere, J. L. Flanagan : "Current perspectives in digital speech."


(5) J. L. Flanagan : "Speech analysis, synthesis and perception."
- Springer-Verlag, 1965.


(7) N. R. F. MacKinnon : "The development of speech encipherment."

(8) N. S. Jayant : "Digital coding of speech waveforms : PCM, DPCM and DM quantizers."
(9) J. W. McVay, J. D. Gibson: "Analyses and experiments on the DM into LPC tandem."

(10) B. S. Atal, S. L. Hanauer: "Speech analysis and synthesis by linear prediction of the speech wave."

(11) B. S. Atal: "Predictive coding of speech at low bit rates."


(13) R. Steele: "Delta modulation systems."
- Pentech Press, London.

(14) A. Tomozawa, H. Kaneko: "Companded delta modulation for telephone transmission."

(15) J. A. Greefkes, K. Riemens: "Code modulation with digitally controlled companding for speech transmission."

C. K. Un, H. S. Lee, J. S. Song: "Hybrid companding delta modulation.


H. R. Schindler: "Delta modulation.

J. O. Smith, J. B. Allen: "Variable bandwidth adaptive delta modulation.


N. S. Jayant: "Adaptive delta modulation with a one-bit memory.

(25) J. B. O'Neal: "Waveform encoding of voiceband data signals."


(27) "IEEE Recommended practice for speech quality measurements."

(28) D. J. Goodman, B. J. McDermott, L. H. Nakatani: "Subjective evaluation of PCM coded speech."

(29) N. S. Jayant, A. E. Rosenberg: "The preference of slope overload to granularity in the delta modulation of speech."

(30) M. H. L. Hecker, C. E. Williams: "Choice of reference conditions for speech preference tests."


(33) W. A. Munson, J. E. Karlin: "Isopreference method for evaluating speech transmission circuits."


(35) M. Nakatsui, P. Mermelstein: "Subjective speech-to-noise ratio as a measure of speech quality for digital waveform coders."

(36) M. R. Schroeder: "Reference signal for signal quality studies."


(38) M. R. Spiegel: "Theory and problems of statistics (in SI units)."

(39) M. E. Hogan: "Voice packet switching."

(40) J. A. Eva, A. Day: "SABus user manual."
- Digital Postgraduate Laboratory, Dept. Electrical and Electronic Engineering, University of Cape Town.

(41) T. J. Terrel: "Introduction to digital filters."

(43) B. McDermott, C. Scaglioni, D. Goodman: "Perceptual and objective evaluation of speech processed by adaptive differential PCM."

(44) P. Mermelstein: "Evaluation of a segmental SNR as an indicator of the quality of ADPCM coded speech."


(46) N. M. Kim, C. K. Un, J. R. Lee: "A multisubscriber variable-rate sampling HCDM system with dynamic buffer control."
APPENDIX A

In this appendix the derivation of the equations for the Song Voice Adaptive Delta Modulation (SVADM) encoder is given.

Referring to fig. 3.8 it should be noted that the predictor (in the encoder) can only use the past output sequence

\[ b_{k-1}, b_{k-2}, \ldots, b_1 \] (which shall be represented as \( b^{k-1} \))

and the past predicted sequence

\[ x_{k-1}, x_{k-2}, \ldots, x_1 \] (which shall be represented as \( x^{k-1} \))

to predict the sample of the source \( Y_k \). The estimator (in the decoder) on the other hand can use \( b^k \) and \( X^k \) to estimate \( Y_k \).

To optimise the predictor Song et al. minimised the mean square of the difference between \( Y_k \) and \( X_k \). Noting that \( X_k \) is a function of \( b^{k-1} \) and \( X^{k-1} \) (i.e. \( X_k = X_k(b^{k-1}, X^{k-1}) \)) the mean square of the difference can be represented by:

\[ E((Y_k - X_k(b^{k-1}, X^{k-1}))^2). \] (1)

If the output of the estimator is represented by \( R_k \) then the optimum estimator is obtained by minimising

\[ E((Y_k - R_k(b^k, X^k))^2). \] (2)

In the system of Song et al the two past samples were used and hence \( b^{k-1} \) and \( X^{k-1} \) can be given by \( b_{k-1}, b_{k-2} \) and \( x_{k-1}, x_{k-2} \) respectively.

The minimisation of (1) and (2) leads to the equations:

\[ X_k = E(Y_k | x_{k-1}, x_{k-2}, b_{k-1}, b_{k-2}) \] (3)
APPENDIX A

\[ R_k = E( Y_k \mid X_k, X_{k-1}, b_k, b_{k-1} ) \]  

(4)

In the derivation of Song et al it was assumed that a Markov source was used, with samples given by:

\[ Y_k = \rho Y_{k-1} + \lambda_{k-1}. \]  

(5)

If each \( \lambda_{k-1} \) is uncorrelated, normal with a zero mean and standard deviation \( \sigma \), then it can be shown that

\[ X_{k+1} = \rho R_k. \]  

(6)

The result in (6) can be derived by substituting (5) into (3).

Using the two past samples, the estimator equation is given by:

\[ R_k = \int_{-\infty}^{\infty} Y_k p( Y_k \mid X_{k-1}^k, b_{k-1}^k ) dY_k \]

which has solutions:\footnote{1}:

(7a) \( b_k = +1, \ b_{k-1} = -1 \):

\[ R_k = \int_{-\infty}^{\infty} \left[ \rho q'(z_{k-1}) Y_{k-1} + \left( \frac{\sigma}{\sqrt{2\pi}} \right) \exp\left( -\frac{1}{2} z_{k-1}^2 \right) \right] P(Y_{k-1}) dY_{k-1} \]

(7b) \( b_k = +1, \ b_{k-1} = +1 \):

\[ R_k = \int_{-\infty}^{\infty} \left[ \rho q'(z_{k-1}) Y_{k-1} + \left( \frac{\sigma}{\sqrt{2\pi}} \right) \exp\left( -\frac{1}{2} z_{k-1}^2 \right) \right] P(Y_{k-1}) dY_{k-1} \]

(7c) \( b_k = -1, \ b_{k-1} = +1 \):

\[ R_k = \int_{-\infty}^{\infty} \left[ \rho q'(z_{k-1}) Y_{k-1} + \left( \frac{\sigma}{\sqrt{2\pi}} \right) \exp\left( -\frac{1}{2} z_{k-1}^2 \right) \right] P(Y_{k-1}) dY_{k-1} \]

(7d) \( b_k = -1, \ b_{k-1} = -1 \):

\[ R_k = \int_{-\infty}^{\infty} \left[ \rho q'(z_{k-1}) Y_{k-1} + \left( \frac{\sigma}{\sqrt{2\pi}} \right) \exp\left( -\frac{1}{2} z_{k-1}^2 \right) \right] P(Y_{k-1}) dY_{k-1} \]

where
To simplify the results \( q'(y) \) is approximated by:

\[
q'(y) \approx \begin{cases} 
\exp(-ay^2) & \text{for } y > 0 \\
1 - \exp(-ay^2) & \text{for } y < 0 
\end{cases}
\] (8)

It has been found that the value of 'a' which results in the greatest simplification of (7) is \( a = 0.5 \).

If the input samples are highly correlated (\( \rho > 0.9 \)) then (7) can be reduced to:

(9a) \( b_k = +1, b_{k-1} = -1 \):
\[
R_k = \frac{2(1/\sqrt{2\pi})\sigma}{\sqrt{2\pi}} \left\{ X_k - \frac{\sigma \exp(-\frac{1}{2}y_k^2)}{\sqrt{2\pi} q(y_k)} \right\}
\]

(9b) \( b_k = +1, b_{k-1} = +1 \):
\[
R_k = \frac{2(1/\sqrt{2\pi})\sigma}{\sqrt{2\pi}} \left\{ X_k - \frac{\sigma \exp(-\frac{1}{2}y_k^2)}{\sqrt{2\pi} q(y_k)} \right\}
\]

(9c) \( b_k = -1, b_{k-1} = +1 \):
\[
R_k = \frac{2(1/\sqrt{2\pi})\sigma}{\sqrt{2\pi}} \left\{ X_k - \frac{\sigma \exp(-\frac{1}{2}y_k^2)}{\sqrt{2\pi} q(y_k)} \right\}
\]

(9d) \( b_k = -1, b_{k-1} = -1 \):
\[
R_k = \frac{2(1/\sqrt{2\pi})\sigma}{\sqrt{2\pi}} \left\{ X_k - \frac{\sigma \exp(-\frac{1}{2}y_k^2)}{\sqrt{2\pi} q(y_k)} \right\}
\]
where
\[
y_k = (\rho X_{k-1} - X_k) / \sigma
\] (9e)

Since \( \rho \) is approximately equal to 1,

\[
R_k = \rho X_{k+1} \approx X_{k+1}
\]
and
\[ y_k \approx (X_{k-1} - X_k)/\sigma, \]
which allows (9) to be re-written as:
\[ -y_{k+1} = \pm \sqrt{2/\pi} + \left( \exp\left(-\frac{1}{2}y_k^2\right) \right)/\left( q'(y_k) \sqrt{2\pi} \right) \quad b_{k+1} = +1 \]  
\[ -y_{k+1} = \pm \sqrt{2/\pi} - \left( \exp\left(-\frac{1}{2}y_k^2\right) \right)/\left( q'(y_k) \sqrt{2\pi} \right) \quad b_{k+1} = -1 \]  

The system described by (9) is highly non-linear and not easily implementable using digital integrated circuits.

From equation (5), using the approximation \( \rho \approx 1 \):
\[ \sigma^2 = \text{E}( Y_k - \rho Y_{k-1} )^2 \approx \text{E}( Y_k - Y_{k-1} )^2 \]
which, considering that \( X_k \) is the estimate of \( Y_k \), modifies to
\[ \sigma^2 \approx \text{E}( X_k - X_{k-1} )^2. \]
If a further approximation is made, namely
\[ \sigma^2 \equiv ( X_k - X_{k-1} )^2 \]
then
\[ y_k \approx \text{sgn}( X_{k-1} - X_k ). \]  

Using equations (7e) and (11) the equations (9) simplify to:
\[
\text{for } X_k - X_{k-1} > 0 \\
R_k = \begin{cases} 
X_k + 0.57( X_k - X_{k-1} ) & b_k = +1, \quad b_{k-1} = -1 \\
X_k + 1.15( X_k - X_{k-1} ) & b_k = +1, \quad b_{k-1} = +1 \\
X_k - 0.57( X_k - X_{k-1} ) & b_k = -1, \quad b_{k-1} = +1 \\
X_k - 1.15( X_k - X_{k-1} ) & b_k = -1, \quad b_{k-1} = -1 
\end{cases} \]
APPENDIX A

and for $X_k - X_{k-1} < 0$

$$R_k = X_{k+1} = \begin{cases} 
X_k - 0.51(X_k - X_{k-1}) & b_k = +1, b_{k-1} = -1 \\
X_k - 1.15(X_k - X_{k-1}) & b_k = +1, b_{k-1} = +1 \\
X_k + 0.51(X_k - X_{k-1}) & b_k = -1, b_{k-1} = +1 \\
X_k + 1.15(X_k - X_{k-1}) & b_k = -1, b_{k-1} = -1
\end{cases}$$

(13a) \hspace{1cm} (13b) \hspace{1cm} (13c) \hspace{1cm} (13d)

From equations (13a) and (13b), for $b_k = +1 (X_{k+1} - X_k) > 0$. If

we let $k = n - 1$, then when $b_{n-1} = +1 (X_n - X_{n-1}) > 0$. However,

this contradicts the condition for equations (13a) - (13d).

Hence, equations (13b) and (13c) cannot exist. Similarly,

equations (12a) and (12d) can be ignored, and so the system

simplifies to :

$$R_k = X_{k+1} = \begin{cases} 
X_k + 0.815|X_k - X_{k-1}| - 0.3|X_k - X_{k-1}| & b_k = +1, b_{k-1} = -1 \\
X_k + 0.815|X_k - X_{k-1}| + 0.3|X_k - X_{k-1}| & b_k = +1, b_{k-1} = +1 \\
X_k - 0.815|X_k - X_{k-1}| + 0.3|X_k - X_{k-1}| & b_k = -1, b_{k-1} = +1 \\
X_k - 0.815|X_k - X_{k-1}| - 0.3|X_k - X_{k-1}| & b_k = -1, b_{k-1} = -1
\end{cases}$$

(14a) \hspace{1cm} (14b) \hspace{1cm} (14c) \hspace{1cm} (14d)

$R_k$ can then be written as :

$$R_k = X_{k+1} = X_k + g + h.$$

where

$g = g(b_k, X_k - X_{k-1})$, and

$h = h(b_{k-1}, X_k - X_{k-1})$. 

PAGE A-5
APPENDIX B

This appendix contains the circuit diagrams for the hardware implementation of the Song Voice Adaptive Delta Modulation (SVADM) system described in Chapter 4.
This appendix contains the software for implementing the full duplex communications link described in Chapter 4.
START

INITIALISE MEMORY STACK

SET PPI MODE OF OPERATION

DISABLE INTERRUPT MODE

ENTER SAMPLING FREQUENCY

SET TIMER MODE

ENABLE INTERRUPT MODE

INPUT DATA FROM CODECS

OUTPUT TO RECEIVING CODECS

CONSOLE INTERRUPT?

Y

N
VOICE PACKET SWITCHING

J.M.IRVINE
3rd JUNE 1983

TELPHN.A85: A Program to give direct Full Duplex communication

Test for char in USART 0.
Prints a string on the console.
Input from the console.
Temporary storage.
Start of Data Stack.
ASCII <CR>.
ASCII <LF>.
Mask programming byte.
Appendix C - Duplex Communication Program

MODULE PAGE 2

LOC OBJ LINE SOURCE STATEMENT

29 ;***********************************************************************;
30 ; MAINLINE OF THE PROGRAM.
31 ; ================
32 ; INPUTS : BYTES OF DATA FROM THE DELTA MODULATOR, CONTROL
33 ; : CHARACTERS, AND CHARACTERS FROM THE CONSOLE.
34 ; OUTPUTS : BYTES OF DATA TO THE DEMODULATOR.
35 ; CALLS : RATE, SETUP, UOTEST.
36 ; DESTROYS : NONE.
37 ; DESCRIPTION : The outputs from the modulators of the two telephones
38 ; : must be relayed on to the respective demodulators.
39 ;
40 ;***********************************************************************;
41 ;BEGIN:
42 ;
43 1800 F3 44 DI ; Disable all interrupts.
44 1801 31001F 45 LXI SP,STAK ; Initialise an independent stack.
45 1804 CD2A18 46 CALL SETUP ; Initialise the interface card of the SABUS kit.
46 1807 00 47 LOOP: NOP ;
47 1808 C30718 48 JMP LOOP ;
48 50 ;
50 ;
51 ;
52 ;
53 ;
54 ;
55 ;
56 ;
57 ;
58 180B F5 59 STROBE: PUSH PSW ; Save the relevant registers.
59 180C 85 60 PUSH H ;
60 180D FB 61 EI ; Enable the interrupts again.
61 180E DBFC 62 ;
62 63 IN ØFCH ; Input the most recent byte from Tel. 2.
Appendix C - Duplex Communication Program

LOC OBJ LINE SOURCE STATEMENT

1810 32001C 64 STA BUFFER ; Store for later output.
1813 DBF8 65 IN 0F8H ; Input the most recent 8 bits from the DM.
1815 00 66 ;
1816 00 67 NOP ; Because the RST 7.5 interrupt and
1817 00 68 NOP ; the output (parallel-serial) register
1818 00 69 NOP ; "load" are simultaneous it is necessary
1819 00 70 NOP ; to ensure that the load is complete before
181A D3FD 71 NOP ; outputting new information to this register.
181C 3A001C 72 ;
181F D3F9 73 OUT 0F9H ; Output to Tel. 2.
1821 CD3F00 74 LDA BUFFER ; Recover the input from Tel. 2.
1824 C43218 75 OUT 0F9H ; Output to Tel. 1.
1827 E1 76 ;
1828 F1 77 CALL UOTEST ; Test for interrupt from the console.
1829 C9 78 CNZ RATE ;
1829 C9 79 ;
1827 E1 80 POP H ; Restart the registers.
1828 F1 81 POP PSW ;
1829 C9 82 RET ;
83 $EJECT
Appendix C - Duplex Communication Program

LOC OBJ LINE SOURCE STATEMENT

84 ;***********************************************************************;
85 ;
86 ; PROCEDURE : SETUP.
87 ; INPUTS : NONE.
88 ; OUTPUTS : CONTROL WORDS TO INTERFACE CARDS.
89 ; CALLS : CLOCKS.
90 ; DESTROYS : A, F/F's.
91 ; DESCRIPTION : Control words are sent to the interface card in order;
92 ; to set it in the desired mode. The 8255 Parallel I/O device is set to have two input channels, two output:
93 ; channels, and two channels for handshaking.
94 ;
95 ;***********************************************************************;
96 ;
182A 3EB4 98 SETUP: MVI A,0B4H ; Configure both PPI Interfaces (8255's) for Mode 1
182C D3FB 99 OUT 0FBH ; i.e. Port A Input, Port B Output, Port C Handshake.
182E 3EBC 100 MVI A,0BCH ; Port C0 PC6/7 = output, Port C1 PC6/7 = input.
1830 D3FF 101 OUT 0FFH ;
1831 102 ;
1832 F3 103 RATE: DI ; Disable interrupts while changes are made.
1833 3E0D 104 MVI A,0DH ; Disable the CODEC clocks.
1835 D3FB 105 OUT 0FBH ;
1837 3E0E 106 MVI A,0EH ; Reset the CODEC F/F's.
1839 D3FB 107 OUT 0FBH ;
1840 108 ;
183B 3ECC 109 MVI A,0CCH ; Equivalent to zero signal.
183D D3F9 110 OUT 0F9H ; Initialise the DM decoder.
1841 111 ;
183F CD5818 112 CALL CLOCKS ; Set the programmable interval timer.
1842 3E0F 113 ;
1844 D3FB 114 MVI A,0FH ; Set the F/F Clear lines.
1846 3E0C 115 OUT 0FBH ;
1848 D3FB 116 MVI A,0CH ; Enable the CODEC clocks.
117 ;
118 ;
### Duplex Communication Program

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
<th>STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>184A</td>
<td>DBFA</td>
<td>119</td>
<td>STRB:</td>
<td>IN $0FAH; Because of indeterminate starting conditions</td>
</tr>
<tr>
<td>184C</td>
<td>E520</td>
<td>120</td>
<td>ANI</td>
<td>$20H; the interface registers are loaded</td>
</tr>
<tr>
<td>184E</td>
<td>CA4A18</td>
<td>121</td>
<td>JZ</td>
<td>STRB; and read before enabling the Codec.</td>
</tr>
<tr>
<td>1851</td>
<td>DBF8</td>
<td>122</td>
<td>IN</td>
<td>$0F8H; Reset the IBF line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1853</td>
<td>3E3B</td>
<td>124</td>
<td>MVI</td>
<td>A, MASK; Interrupt mask enabling the RST 7.5 interrupt.</td>
</tr>
<tr>
<td>1855</td>
<td>30</td>
<td>125</td>
<td>SIM</td>
<td>Set the mask.</td>
</tr>
<tr>
<td>1856</td>
<td>FB</td>
<td>126</td>
<td>EI</td>
<td>Future input/output is interrupt controlled.</td>
</tr>
<tr>
<td>1857</td>
<td>C9</td>
<td>127</td>
<td>RET</td>
<td></td>
</tr>
</tbody>
</table>

$\text{\textbackslash$ Ej\textbackslash$ect}$
Appendix C - Duplex Communication Program

LOC OBJ LINE SOURCE STATEMENT

129 ;***********************************************************************;
130 ;
131 ; PROCEDURE : CLOCKS.
132 ; INPUTS  : CHARACTERS FROM THE CONSOLE.
133 ; OUTPUTS : USER PROMPTS TO CONSOLE SCREEN PLUS CONTROL
134 ;          : INFORMATION TO THE PROGRAMMABLE INTERVAL TIMER.
135 ; CALLS  : GETECH, OUTMSG.
136 ; DESTROYS : A,F/F's.
137 ; DESCRIPTION : The 8253 Programmable Timer can be set to operate
138 ;                      : at the desired bit rate with a second clock being
139 ;                      : used for strobing. This second timer runs at one
140 ;                      : eighth the bit rate.
141 ;
142 ;***********************************************************************;

143 ;
1858 216819 144 CLOCKS: LXI H,SCREEN; String to clear the screen.
1858 CD7200 145 CALL OUTMSG;
185E 21C018 146 LXI H,MSG ; Start of ASCII string.
1861 CD7200 147 CALL OUTMSG ; Send the string to the console screen.
148 ;
1864 CD6000 149 RESP: CALL GETECH ; Get the user's response.
1867 218618 150 LXI H,TOPA ;
186A FE41 151 CPI 41H ; Check for ASCII 'A'.
186C CA9818 152 JZ INIT ; User chose first option.
186F 21B818 153 LXI H,TOPB ;
1872 FE42 154 CPI 42H ; Check for ASCII 'B'.
1874 CA9818 155 JZ INIT ; Second option chosen.
1877 21BA18 156 LXI H,TOPC ;
187A FE43 157 CPI 43H ; Check for ASCII 'C'.
187C CA9818 158 JZ INIT ; Third option chosen.
187F 21BC18 159 LXI H,TOPD ;
1882 FE44 160 CPI 44H ; Check for ASCII 'D'.
1884 CA9818 161 JZ INIT ; Fourth option chosen.
1887 21BE18 162 LXI H,TOPE ;
188A FE45 163 CPI 45H ; Check for ASCII 'E'.


### Appendix C - Duplex Communication Program

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>188C</td>
<td>CA9818</td>
<td>164</td>
<td>JZ INIT ; Last option chosen.</td>
</tr>
<tr>
<td>188F</td>
<td>213819</td>
<td>165</td>
<td>LXI H,ERROR ; Invalid option has been selected.</td>
</tr>
<tr>
<td>1892</td>
<td>CD7200</td>
<td>166</td>
<td>CALL OUTMSG ;</td>
</tr>
<tr>
<td>1895</td>
<td>C36418</td>
<td>167</td>
<td>JMP RESP ; Keep looping until valid option chosen.</td>
</tr>
<tr>
<td>1898</td>
<td>3E76</td>
<td>170</td>
<td>INIT: MVI A, 76H ; Mode Word for Counter #1.</td>
</tr>
<tr>
<td>189A</td>
<td>D3F7</td>
<td>171</td>
<td>OUT $F7H ; Mode 3, 16-bit binary counter.</td>
</tr>
<tr>
<td>189C</td>
<td>7E</td>
<td>172</td>
<td>MOV A,M ; Set the output clock rate for Counter #1.</td>
</tr>
<tr>
<td>189D</td>
<td>D3F5</td>
<td>173</td>
<td>OUT $F5H ;</td>
</tr>
<tr>
<td>189F</td>
<td>23</td>
<td>174</td>
<td>INX H ; Get the MSB.</td>
</tr>
<tr>
<td>18A0</td>
<td>7E</td>
<td>175</td>
<td>MOV A,M ;</td>
</tr>
<tr>
<td>18A1</td>
<td>D3F5</td>
<td>176</td>
<td>OUT $F5H ; MSB sent last.</td>
</tr>
<tr>
<td>18A3</td>
<td>3EB6</td>
<td>177</td>
<td>MVI A, 0B6H ; Mode Word for Counter #0.</td>
</tr>
<tr>
<td>18A5</td>
<td>D3F7</td>
<td>178</td>
<td>OUT $F7H ; Same mode as Counter #1.</td>
</tr>
<tr>
<td>18A7</td>
<td>3E08</td>
<td>179</td>
<td>MVI A, 08H ; Set the output clock rate for Counter #0.</td>
</tr>
<tr>
<td>18A9</td>
<td>D3F6</td>
<td>180</td>
<td>OUT $F6H ; It is 1/8th the rate for Counter #1.</td>
</tr>
<tr>
<td>18AB</td>
<td>3E00</td>
<td>181</td>
<td>MVI A, 00H ;</td>
</tr>
<tr>
<td>18AD</td>
<td>D3F6</td>
<td>182</td>
<td>OUT $F6H ; Send the MSB last.</td>
</tr>
<tr>
<td>18AF</td>
<td>216D19</td>
<td>183</td>
<td>LXI H,MSG ; Send a message to the console screen.</td>
</tr>
<tr>
<td>18B2</td>
<td>CD7200</td>
<td>184</td>
<td>CALL OUTMSG ;</td>
</tr>
<tr>
<td>18B5</td>
<td>C9</td>
<td>185</td>
<td>RET ;</td>
</tr>
</tbody>
</table>

186 $EJECT
### Appendix C - Duplex Communication Program

<table>
<thead>
<tr>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>187 ;*********************************************************;</td>
</tr>
<tr>
<td>188 ;</td>
</tr>
<tr>
<td>189 ;</td>
</tr>
<tr>
<td>190 ;</td>
</tr>
<tr>
<td>191 ;*********************************************************;</td>
</tr>
<tr>
<td>192 ;</td>
</tr>
<tr>
<td>18B6 A0</td>
</tr>
<tr>
<td>193 TOPA: DB 0A0H ; Corresponds to 9600 baud, DM sample clock.</td>
</tr>
<tr>
<td>18B7 00</td>
</tr>
<tr>
<td>194 DB 00H ;</td>
</tr>
<tr>
<td>18B8 60</td>
</tr>
<tr>
<td>195 TOPB: DB 60H ; Corresponds to 16000 baud.</td>
</tr>
<tr>
<td>18B9 00</td>
</tr>
<tr>
<td>196 DB 00H ;</td>
</tr>
<tr>
<td>18BA 30</td>
</tr>
<tr>
<td>197 TOPC: DB 30H ; Corresponds to 32000 baud.</td>
</tr>
<tr>
<td>18BB 00</td>
</tr>
<tr>
<td>198 DB 00H ;</td>
</tr>
<tr>
<td>18BC 1C</td>
</tr>
<tr>
<td>199 TOPD: DB 1CH ; Corresponds to 56000 baud.</td>
</tr>
<tr>
<td>18BD 00</td>
</tr>
<tr>
<td>200 DB 00H ;</td>
</tr>
<tr>
<td>18BE 18</td>
</tr>
<tr>
<td>201 TOPE: DB 18H ; Corresponds to 64000 baud.</td>
</tr>
<tr>
<td>18BF 00</td>
</tr>
<tr>
<td>202 DB 00H ;</td>
</tr>
<tr>
<td>203 ;</td>
</tr>
<tr>
<td>18C0 54484520</td>
</tr>
<tr>
<td>204 MSG: DB 'THE POSSIBLE CODEC BAUD RATES ARE :',CR,LF</td>
</tr>
<tr>
<td>504F5349</td>
</tr>
<tr>
<td>49424C45</td>
</tr>
<tr>
<td>20434F44</td>
</tr>
<tr>
<td>45432042</td>
</tr>
<tr>
<td>41554420</td>
</tr>
<tr>
<td>52415445</td>
</tr>
<tr>
<td>53204152</td>
</tr>
<tr>
<td>45203A0D</td>
</tr>
<tr>
<td>0A</td>
</tr>
<tr>
<td>18E5 28612920</td>
</tr>
<tr>
<td>205 DB ' (a) '9600',CR,LF</td>
</tr>
<tr>
<td>20393630</td>
</tr>
<tr>
<td>300D0A</td>
</tr>
<tr>
<td>18F0 28622920</td>
</tr>
<tr>
<td>206 DB ' (b) 16000',CR,LF</td>
</tr>
<tr>
<td>31363030</td>
</tr>
<tr>
<td>300D0A</td>
</tr>
<tr>
<td>18FB 28632920</td>
</tr>
<tr>
<td>207 DB ' (c) 32000',CR,LF</td>
</tr>
<tr>
<td>33323030</td>
</tr>
<tr>
<td>LOC</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>300D0A</td>
</tr>
<tr>
<td>35363030</td>
</tr>
<tr>
<td>36343030</td>
</tr>
<tr>
<td>52205448</td>
</tr>
<tr>
<td>45205245</td>
</tr>
<tr>
<td>51554952</td>
</tr>
<tr>
<td>4544204F</td>
</tr>
<tr>
<td>5054494F</td>
</tr>
<tr>
<td>4E203E</td>
</tr>
<tr>
<td>1967</td>
</tr>
<tr>
<td>1968</td>
</tr>
<tr>
<td>1969</td>
</tr>
<tr>
<td>196A</td>
</tr>
<tr>
<td>196B</td>
</tr>
<tr>
<td>196C</td>
</tr>
</tbody>
</table>
Appendix C - Duplex Communication Program

LOC OBJ  LINE  SOURCE STATEMENT

221 
196D 0D0A54F  222  MESG:  DB  CR,LF,'TO UPDATE THE SAMPLING RATE TYPE ANY CHARACTER >'

199F 00  223  DB  00H ;
1F00  224 
1F00  225  ORG  1F00H 
1F00  226  STAK:  DS  1 ;
1F00  227 
1FFD  228  ORG  1FFDH 
1FFD C30B18  229  JMP  STROBE ; 1FFD is the trap address for RST 7.5.

0 error(s) detected

User symbols

BEGIN  1800 A  BUFFER 1C00 A  CLOCKS 1858 A  CR  000D A  ERROR 1938 A  GETECH 0060 A  INIT 1898 A
LF  000A A  LOOP 1807 A  MASK 003B A  MESSG 196D A  MSG 18C0 A  OUTMSG 0072 A  RATE 1832 A
RESP  1864 A  SCREEN 1968 A  SETUP 182A A  STAK 1F00 A  STRB 184A A  STROBE 180B A  TOP 1C01 A
TOPA  1896 A  TOPB 1898 A  TOPC 182A A  TOPD 18BC A  TOPE 18BE A  UOTEST 003F A
APPENDIX D

The simulation package described in chapter 5 is contained on a floppy disc (5 1/4 inch) labelled "CODSIM: Codec Simulation Package". It is possible for the system to operate using only the two built-in disc drives of the HP9836C desk-top computer, but if a hard disc (such as the HP9134XV) is available then it is advisable to copy the system on to this disc as the access times are considerably shorter and generally the available storage space is greater.

D.1 Using a hard disc.

If a hard disc is used then it will be necessary for the user to alter the software accordingly. The changes necessary are very simple to make.

In each of the programs in the package there are statements specifying which disc drives are used. These statements occur in the initialisation section of the software (in the initialisation routine — in the subroutine section) and are:

\[
\text{Drive}_0 = "\text{:\ itertools,4,0}"
\]

\[
\text{Drive}_1 = "\text{:\ itertools,4,1}"
\]

If the two built-in drives are to be used then these need not be changed. However, if a hard disc is to be used then these statements should be changed to:

\[
\text{Drive}_0 = "\text{:\ HP9134,700}"
\]

\[
\text{Drive}_1 = "\text{:\ HP9134,700}".
\]

or a similar appropriate designation depending on which disc
APPENDIX D

Drive is used (refer to the operating/user manual for the drive concerned). The "700" in the designation indicates that interface "7" is to be used and the drive is at address "00".

D. 2 Getting started.

The computer is switched off and you wish to commence working. In order to boot the appropriate system the following procedure should be followed.

(i) Insert the system disc, labelled "Codsim: System Disc", into the right hand disc drive and close the drive's door.

(ii) Switch on the computer, the Multiprogrammer (HP6942A) and any other relevant peripheral devices.

(iii) When the computer has completed booting remove the disc from the drive and replace it with the disc labelled "Codsim: System Extensions".

(iv) Now type:

```
LOAD BIN"AP2_1" EXECUTE
```

When this has completed type:

```
LOAD BIN"GRAPH2_1" EXECUTE
```

If the HP9134XV hard disc (or any other disc with the 'XV' appendage) is to be used then type:

```
LOAD BIN"XV2_1" EXECUTE
```

(v) If necessary the changes mentioned in D. 1 should be made at
this stage. The programs in the simulation library that need to be adjusted are:

RECORD
PLAYBACK
CODECS
MEASURE
POLATE

and, as mentioned before, are kept on the "CODSIM : Codec Simulation Package" disc. This disc should be inserted in the right hand disc drive. If a hard disc is to be used the programs should be loaded into memory, edited and then stored on the hard disc.

eg. LOAD "MASTER" EXECUTE

edit the program as described in section D.1,

STORE "MASTER:HP9134,700" EXECUTE

Once this has been done the default mass storage device should be changed:

MSI ":HP9134,700" EXECUTE

(Only to be done if the built-in disc drives are not to be used.)

(vi) The system should now be in a usable form. To commence using the simulation package type:

LOAD "MASTER", 1 EXECUTE

This will start the system by displaying a menu.
D. 3 **Using the simulation package.**

The system has been designed to interact with the user via a set of menus. The main menu contains a list of the possible subsystems that can be accessed. The menu is shown below:

Select a function from the following:

1) Acquire speech data at 8 kHz
2) Interpolate 8 kHz data
3) Process interpolated data
4) Determine objective quality measures
5) Replay speech data
6) Terminate

Enter the required option >

The user selects the desired option by typing the corresponding number. When the selected function has completed the main menu will again be displayed and the user can enter a new option.

D. 3.1 **Acquire speech data at 8 kHz.**

Before options 2-5 can be successfully chosen it is necessary to record some speech and store it. This is done by selecting option 1. The program will chain to the input interface program. When the system has completed the interface configuration the memory configuration is carried out. The length of the speech segment that will be recorded is 10 seconds (above this value lack of memory space becomes a problem. Once all configurations have been
completed the system will be ready to commence recording speech segments and the message

Press CONTINUE when ready to record

will be displayed. When the CONTINUE key is pressed the system commences the sampling, quantizing and digitizing process, stopping when the 10 second segment has been recorded.

The RMS level of the signal will be calculated and displayed, along with the peak signal values. The user is then asked whether this segment should be saved or discarded:

Do you wish to save this segment (Y or N) ?

If the 'Y' option is selected the computer then asks for the name of the file in which the data must be stored.

Enter the desired file name >

The file name can be entered using either upper or lower case alphanumeric characters (lower case characters are converted to upper case by the program). A maximum of eight characters can be used for the file name. If the file name already exists the user is asked if that file should be overwritten and, if not, a new name will be prompted for.

Once the data has been stored or discarded the user is asked if any more data is to be recorded. If the response to this is in the affirmative then the process begins again. If not, the user
APPENDIX D

is returned to the main menu.

D.3.2 **Interpolate 8 kHz data.**

If option two is selected the computer needs to know three things -

- the name of the source file,
- the required effective sampling rate,
- the name of the destination file.

All files containing 8 kHz data have the suffix 

"_0" appended to the name chosen by the user (to indicate that the file contains 'original' data - i.e. unprocessed). This suffix need not be specified by the user as the computer automatically affixes the correct suffix. In response to the prompt

**Enter the name of the source file >**

a name, containing at most eight characters, must be entered.

Next the new effective sampling rate is entered.

**Enter the sampling rate (Possible values : 16, 24, 32, 64) >**

These values are in kHz. Any values other than those shown will be rejected.

The third piece of information is the name of the file in which the interpolated data is to be stored. As in previous cases the user responds to the prompt
Enter the destination file name >

by entering a name (maximum of eight characters). If no name is
specified the computer uses the name of the source file, the only
difference being the suffix appended. All processed files are
given the "_P" appendage to indicate 'processed' data.

e.g. If no destination file name is specified and the source
file name was "TEST_O", then the destination file will be
given the name "TEST_P"

Once the interpolation is complete the user is returned to the
main menu.

D.3.3 Process interpolated data.

In describing this section it is assumed that the user is familiar
with the designs of the various codecs. There are seven codecs
that have been implemented in this system (once the user is
thoroughly familiar with the simulation system's design details it
will be possible to implement other codec designs).

The first thing the user is asked is which codec should be used to
process the speech data. It has been made possible to select a
range of codecs - i.e. one segment of speech could be processed by
all or just some of the seven codecs.
APPENDIX D

Codec Type
1) LDM
2) CVSD
3) CFDM
4) SVADM
5) SHCDM
6) MSHCDM
7) HCDM

<table>
<thead>
<tr>
<th>Start</th>
<th>Stop</th>
<th>Increment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A codec type '0' returns to main menu.

Enter the desired value >

The value that is to be entered by the user is shown on the screen in inverse video. If no value is entered then the default value displayed on the screen will be assumed. Once a value is entered the relevant value in the table is updated and the inverse video mode is turned off with the next value now being displayed in inverse video. Once all the values have been entered the user is asked:

Do you wish to keep these values (Y or N) ?

The default answer is 'Y'. If the 'N' option is selected then the 'Start' value is shown in inverse video and the values can be
altered as before.

eg. If values for 'Start', 'Stop' and 'Increment' of 2, 6 and 2 respectively are chosen then the speech will be processed by the CVSD, SVADM and MSHCDM codecs.

There are various design parameters affecting the codecs' performance. These include the leakage time constants (for the leaky integrators), maximum and minimum step sizes, maximum signal range, etc. Depending on the codecs selected, these parameters are presented to the user in a similar fashion to that described above. If a design parameter is not associated with the selected codec it will not be presented to the user for modification. Initially the default parameter values are displayed and it is up to the user to change them.

As in previous cases the user must specify the source file that is to be processed. It should be noted that the codecs implemented here have been designed to allow the user to decide whether 16, 24, 32 or 64 kHz data is to be processed (the choice is reflected in the parameter default values).

Lastly, the user enters the name of the files in which the processed data is to be stored.

Enter the destination file name >
If a range of codecs, or parameter values, had been selected then the file name is given a numerical appendage to indicate the separate files.

eg. If the name 'TEST' had been selected then the processed data will be stored in 'TEST1_P', 'TEST2_P', 'TEST3_P', etc.

D.3.5 Determine objective quality measures.

This section is virtually identical to the previous one except that the various signal-to-noise ratios are calculated and the processed data is not stored simply because the routines to calculate the noise components of the signal destroy the original signal.

Before asking for the source file name the user is asked if the granular and overload SNR's must be calculated.

Do you require SNG (Y or N) ?

and

Do you require SNO (Y or N) ?

The user is also asked

Do you require the segmental values (Y or N) ?

The relevant results will subsequently be sent to the printer.
D.3.6 Replay speech data.

The first thing the user must tell the program is the sampling rate to be used. Because of the interface hardware the maximum sampling rate is limited to 16 kHz. The user will be asked:

Possible output data rates are 8 kHz and 16 kHz.

Enter the required data rate (kHz)>

Next the user will be asked for the source file name:

Enter the name of the file from which the data is to be read>

As in previous cases only the name need be specified and not the "_P" suffix (the computer will do this). The data is divided into 14 sections and the user may specify the start and finish sections. The default values are 1 and 14 respectively. If a value for the finish section is specified to be less than the start then the default value is assumed.

<table>
<thead>
<tr>
<th>Start</th>
<th>Finish</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>YES</td>
</tr>
</tbody>
</table>

Enter the desired value or toggle value (ie. Y/N).

As before the value that must be entered is highlighted on the screen by being shown in inverse video. The 'Continuous' parameter specifies whether or not the selected sections are output repeatedly or just once.

Once the relevant speech segment commences outputting it can be
interrupted by pressing any key on the keyboard.

To terminate the output hit any key on the keyboard.

If this is done the following prompt will be displayed.

Do you wish to output new sections of present segment (Y/N) ?

If 'Y' is selected then the values shown in the table above can be
updated and the process continues as described. Otherwise, the
user is asked:

Do you wish to output a new speech segment (Y/N) ?

In this case, if 'Y' is selected the entire process described in
this section repeats itself. If 'N' is selected then the user is
returned to the main menu.

D. 3. 7 Terminate.

This option gets the user out of the simulation system and back to
the normal operating system.
This appendix contains the logic flowcharts and the software listings of the simulation system described in Chapter 5 and Appendix D.
START

INITIALISE SYSTEM

DISPLAY MENU

INPUT REQUIRED OPTION

VALID OPTION?

Y

B C D E F

FINISH

N
MASTER LIBRARY FOR SPEECH DATA ACQUISITION AND PROCESSING

CODSIM : Codec Simulation Package.
Author : J.M. Irvine.
Department of Electrical and Electronic Engineering
University of Cape Town.
Unit Number : 0.

Programs developed for the Hewlett Packard 2000 Series desk-top computers. All development work done on an HP9836C computer with the interfacing between the analogue and digital environments done using the HP6942A Multiprogrammer.

This program provides the key to the data acquisition and processing system. Access is provided through a main menu listing the possible options available. Program control is then transferred to the software unit corresponding to the selected option.

The default printer device is the monitor.
Turn the key label display off.
Set the default alpha colour to Cyan.
Clear the alpha display.
Turn the graphics display off.
Allocate space for the selected command.
Control character to stop program execution.
Control character to start program execution.

Setup:

PRINTER IS CRT
CONTROL 1,12;1
CONTROL 1,5;140
PRINT USING "$";
GRAPHICS OFF
DIM Function$[40]
Stop$=" !"
Run$=" R"
DISPLAY THE MASTER MENU

PRINT TABXY(10.5),CHR$(132);
PRINT "Select a function from the following :";CHR$(128)
PRINT
PRINT TAB(10),"(1) Acquire speech data at 8 kHz"
PRINT
PRINT TAB(10),"(2) Interpolate 8 kHz data"
PRINT
PRINT TAB(10),"(3) Process interpolated data"
PRINT
PRINT TAB(10),"(4) Determine objective quality measures"
PRINT
PRINT TAB(10),"(5) Replay 8 kHz data"
PRINT
PRINT TAB(10),"(6) Terminate"
PRINT "Enter the required option >"
ON ERROR GOTO Check_error
!
! Poll the controller keyboard for a response from the user.

Wait_answer: !
ON KBD GOTO Get_answer
GOTO Wait_answer
!
Get_answer: !
Answer=VAL(KBD$)
OFF KBD
!
SELECT Answer
CASE 1
  DISP "Function 1 selected"
  Function$="LOAD ""RECORD:INTERNAL,4,0"" X"
  OUTPUT KBD:Stop$:Function$:Run$;
 CASE 2
  DISP "Function 2 selected"
  Function$="LOAD ""POLATE:INTERNAL,4,0"" X"
  OUTPUT KBD:Stop$:Function$:Run$;
 CASE 3
  DISP "Function 3 selected"
  Function$="LOAD ""CODECS:INTERNAL,4,0"" X"
  OUTPUT KBD:Stop$:Function$:Run$;
 CASE 4
  DISP "Function 4 selected"
  Function$="LOAD ""MEASURE:INTERNAL,4,0"" X"
  OUTPUT KBD:Stop$:Function$:Run$;
 CASE 5
  DISP "Function 5 selected"
  Function$="LOAD ""PLAYBACK:INTERNAL,4,0"" X"
  OUTPUT KBD:Stop$:Function$:Run$;
 CASE 6
  GOTO Finish
CASE ELSE
BEEP 1000,.1
DISP CHR$(11);CHR$(129);"INVALID OPTION !!!";CHR$(128)
WAIT 1.5
DISP CHR$(11);"Please re-enter the required option >"
GOTO Wait_answer
END SELECT
!

Finish: !
PRINTER IS CRT
CONTROL 1,5;139
CONTROL 1,12;0
MASS STORAGE IS ":INTERNAL,4,0"
STOP
!

Check_error: !
IF ERRN=32 THEN
  GOTO 1030
END IF
!
END
ENTER SAMPLING RATE

SET UP DATA AND CONTROL CHANNELS

INITIALISE INTERFACE UNIT

INITIATE SAMPLING

CALCULATE RMS LEVEL

SAVE SPEECH SEGMENT?

SAVE DATA ON DISC

RECORD ANOTHER SEGMENT?
PROGRAMS developed for the Hewlett-Packard 2000 Series desk-top computers. All development work done on an HP9836C computer with the interfacing between the analogue and digital environments done using the HP6942A Multiprogrammer.

This program controls the buffered A/D process. The multiprogrammer is initialised to perform continuous data acquisition. The input analogue signal is sampled and converted to digital form at a rate of 8000 samples per second. This data is transferred to the controller via memory cards on the multiprogrammer and when sampling is complete the data is transferred to one of the mass storage devices (the default device is the HP9134 hard disc). Once the storage is complete the user may then listen to the stored data. The length of the sampled speech segment is 3 seconds which is sufficiently long for phonetically balanced sentences such as the Harvard sentences.

<<<<< DEFINE THE VARIABLES USED BY THE PROGRAM >>>>>>>>>>>

OPTION BASE 1

INTEGER Card,Mistake,A_d_slot,D_a_slot ! Multiprogrammer card variables.
INTEGER Mem_in_slot,Tb_slot,Slot_no
INTEGER Complete,No_of_errors,No_of_cards
INTEGER Card_variable,Intr_card,Next_in_buf
INTEGER Loop,Loop1,Differential,Underflow
REAL Intr_svc_delay,Transfer_oh,Total_delay ! System performance parameters.
REAL Transfer_rate,Reinit_time,Head_room
REAL Block_size,Reference_word,Time_available
REAL Transfer_time,Additional_rdgs,Trigger_rate
DIM Function$[40],Error$[40],Mr_command$[30],File_name$[10]
ALLOCATE INTEGER Configuration(0:16.5)

Stop$=" !" ! Control characters to start & Run$=" R" ! stop program execution.
GOSUB Initialise
    ! Reset the mode of the controller.
GOSUB Sample_rate
    ! Set up the variables for 8 kHz sampling.
GOSUB Io_paths
    ! Set up data and control paths.
Xfer(Op$)
    ! Load BINARY subprogram.

! Determine the multiprogrammer card configuration.
Multi_set_up(Configuration(*),Tb_slot,Mem_in_slot,A_d_slot)

GOSUB Re_init
    ! Set the multiprogrammer cards in the
    ! correct operating state.

NOW THE SPEECH SEGMENTS ARE SAMPLED AND STORED

Input_speech:
GOSUB Clear_disp
    ! Re-initialise the VDU displays.
GOSUB Speech_adc
    ! Sample and store the speech using the
    ! ADC (Analogue-to-Digital Conversion)
    ! module.
GOTO Input_speech
SUBROUTINES

SPECIFY THE DEFAULT MASS STORAGE DEVICES

Initialise:

Drive_0$=":INTERNAL,4,0"
Drive_1$=":INTERNAL,4,1"

PRINTER IS CRT
CONTROL 1,12;1
No_of_buffers=8

CLEAR THE ALPHANUMERIC AND GRAPHICS DISPLAYS

Clear_disp:

PRINT USING "@"
PRINT CHR$(128);CHR$(140);
ALPHA ON
DISP " "
RETURN

CALCULATE MEMORY AND TIMING REQUIREMENTS

Sample_rate:

Trigger_rate=1/8000
Intr_svc_delay=.0229
Transfer_oh=.0013
Total_delay=Intr_svc_delay+Transfer_oh
Transfer_rate=7.5355E-5
Reinit_time=.00091

Head_room=Total_delay/Trigger_rate
Reference_word=INT(4085-Head_room)
Time_available=(Reference_word*Trigger_rate-Reinit_time)
Block_size=4085
Transfer_time=Block_size*Transfer_rate
Calculate:

Additional_rdgs=Transfer_time/Trigger_rate
IF Additional_rdgs>=1 THEN
Block_size=Block_size+Additional_rdgs
Transfer_time=Additional_rdgs*Transfer_rate
GOTO Calculate
END IF
Block_size=INT(Block_size)

! <<<<<<<<<<<<<< PRINT OPERATIONAL PARAMETERS ON SCREEN >>>>>>>>>>>>>>>>
!
370 GOSUB Clear_disp
380 !
390 PRINT TABXY(5,3):"CONTINUOUS MODE OPERATIONAL PARAMETERS"
400 PRINT TABXY(8,5):"TRIGGER RATE (microsecs):";Trigger_rate*1.E+6
410 PRINT TABXY(8,6):"BLOCK SIZE (words):";Block_size
420 PRINT TABXY(8,7):"REFERENCE WORD :";Reference_word
430 IF Time_available>1 THEN
440 PRINT TABXY(8,8):"TIME AVAILABLE (secs):";TIME USING "D.DDD";Time_available
450 ELSE
460 PRINT TABXY(8,8):"TIME AVAILABLE (msecs):";TIME USING "DDD.DD";Time_available
470 END IF
!
480 ! <<<<<<<<<<<<<< MEMORY SPACE IS NOW RESERVED >>>>>>>>>>>>>>>>
!
490 !
500 A total of 8 buffers, each representing approximately 1.3 seconds worth
510 of speech, are used to store the sampled speech signal.
520 !
530 ALLOCATE INTEGER Buf1(Block_size),Buf2(Block_size)
540 ALLOCATE INTEGER Buf3(Block_size),Buf4(Block_size)
550 ALLOCATE INTEGER Buf5(Block_size),Buf6(Block_size)
560 ALLOCATE INTEGER Buf7(Block_size),Buf8(Block_size)
570 ! Another 4 buffers are used for calculating the RMS level of the input
580 ! speech signal.
590 !
600 ALLOCATE REAL Buffer1(Block_size),Buffer2(Block_size)
610 ALLOCATE REAL Buffer3(Block_size),Buffer4(Block_size)
620 !
630 RETURN
!
640 ! <<<<<<<<<<<<<< SET UP THE DATA TRANSFER PATHS >>>>>>>>>>>>>>>>
!
650 !
660 Io_paths:
670 ASSIGN @Multi TO 823
680 ASSIGN @Mr_readback TO 82305;FORMAT OFF
690 ASSIGN @Intr_list TO 82312
700 ASSIGN @Intr_cause TO 82310
710 RETURN

!*************************************************************************
! <<<<<<<<<<<<<< SET UP THE DATA TRANSFER PATHS >>>>>>>>>>>>>>>>
!*************************************************************************
340 !~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ INITIALIZE THE SYSTEM CARDS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

350 !---------------------------------------------------------

370 Re_init:!

380 !

390 OUTPUT @Multi;"CC";Tb_slot;"T"
300 OUTPUT @Multi;"WC";Tb_slot:0;"T"
310 OUTPUT @Multi;"WF";Tb_slot+.2;1;"T"
320 OUTPUT @Multi;"CC";A_d_slot;"T"
330 OUTPUT @Multi;"SF";A_d_slot;"1.0.005T"
340 OUTPUT @Multi;"CC";Mem_in_slot;"T"
350 OUTPUT @Multi;"CC";Mem_in_slot+1;"T"
360 OUTPUT @Multi;"WF";Mem_in_slot+1:0;"T"
370 OUTPUT @Multi;"WF";Mem_in_slot+.1;21;"T"
380 !

390 ! SET THE TIME BASE CARD'S CLOCK FREQUENCY
400 !----------------------------------------

410 OUTPUT @Multi;"WF";Tb_slot:Trigger_rate*1.E+6/2;"UT"
420 !

430 ! PROGRAM THE DIFFERENTIAL COUNTER OF THE MEMORY CARDS
440 !----------------------------------------------------------

450 OUTPUT @Multi;"WF";Mem_in_slot+1;Reference_word;"T"
460 !

470 ! INSTRUCT MEMORY ON NUMBER OF READINGS TO BE TAKEN
480 !-----------------------------------------------

490 OUTPUT Mr_command$;"MR";Mem_in_slot:Block_size;"T"
500 !

510 ! ENABLE THE BUFFERED ANALOGUE TO DIGITAL CONVERSION
520 !------------------------------------------------------

530 OUTPUT @Multi;"WF";Mem_in_slot+.1;1;"T"
540 !

550 ! ARM THE MEMORY CARD
560 !------------------------

570 OUTPUT @Multi;"AC";Mem_in_slot+1;"T"
580 RETURN
590 !*************************************************************************
THE ANALOGUE-TO-DIGITAL CONVERSION ROUTINE

Speech_adc:
PRINT TABXY(0.20);"When ready hit the ";CHR$(129);" CONTINUE ";CHR$(128);" key"
PAUSE
Next_in_buf=1
ON INTR 8 GOTO Interrupt
ENABLE INTR 8;2
GOSUB Clear_disp
OUTPUT @Multi:"CY";Tb_slot;"T"
DISP "Stop speaking at the next tone..."
Sampling:
GOTO Sampling
400 !INTERRUPT SERVICE ROUTINE !
410 !
420 !
430 Interrupt:!
440 IF SPOLL(@Multi)=64 THEN ! Multiprogrammer has interrupted.
450 !
460 ENTER @Intr_cause:Complete,No_of_errors,No_of_cards
470 ENTER @Intr_list:Error$
480 IF No_of_cards>0 THEN
490 OUTPUT @Multi:Mr_command$
500 ON Next_in_buf GOTO L1,L2,L3,L4,L5,L6,L7,L8
510 !
520 L1: ENTER @Mr_readback:Buf1(*),Differential,Underflow
530 GOTO Data_is_in
540 !
550 L2: ENTER @Mr_readback:Buf2(*),Differential,Underflow
560 GOTO Data_is_in
570 !
580 L3: ENTER @Mr_readback:Buf3(*),Differential,Underflow
590 GOTO Data_is_in
600 !
610 L4: ENTER @Mr_readback:Buf4(*),Differential,Underflow
620 GOTO Data_is_in
630 !
640 L5: ENTER @Mr_readback:Buf5(*),Differential,Underflow
650 GOTO Data_is_in
660 !
670 L6: ENTER @Mr_readback:Buf6(*),Differential,Underflow
680 GOTO Data_is_in
690 !
700 L7: ENTER @Mr_readback:Buf7(*),Differential,Underflow
710 GOTO Data_is_in
720 !
730 L8: ENTER @Mr_readback:Buf8(*),Differential,Underflow
740 BEEP 3000,1
750 GOSUB Re_init
760 GOSUB Store_on_disc
770 DISP "Do you wish to record more speech? >"
780 Input_answer:!!
790 ON KBD GOTO Response
800 GOTO Input_answer
810 Response:!!
820 R$=UPCS(KBD$)
830 OFF KBD
840 IF R$="N" THEN
850 Function$="LOAD ""MASTER:INTERNAL,4,0"" X"
860 OUTPUT KBD;Stop$;Function$;Run$;
870 ELSE
880 RETURN ! Return to the program mainline.
890 END IF
900 !
910 ELSE
920 DISP "SOME OTHER MULTI INTERRUPT HAS OCCURED"
930 BEEP 500,1
940 STOP
950 END IF
960 !
970 ELSE
980 DISP "SOME OTHER HPIB DEVICE HAS INTERRUPTED"
390 BEEP 500,1
400 STOP
410 END IF
420 !
430 Data_is_in:
440 IF (Differential=4095) OR (Underflow<>0) THEN Problems
450 Next_in_buf=(Next_in_buf MOD No_of_buffers)+1
460 ENABLE INTR 8;2
470 GOTO Sampling
480 !*************************************************************************
090 !~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ STORE SPEECH SEGMENT ON DISC ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
100 ! ---------------------------------------------
110 !
120 Store_on_disc:!
130 Drive$=Drive_1$
140 ! Use the default disc drive.
150 !
160 PRINT "Maximum sampled value = " :MAX(Buf1(*),Buf2(*),Buf3(*),Buf4(*),Buf5(*),Buf6(*),Buf7(*),Buf8(*))
170 PRINT "Minimum sampled value = " :MIN(Buf1(*),Buf2(*),Buf3(*),Buf4(*),Buf5(*),Buf6(*),Buf7(*),Buf8(*))
180 !
190 GOSUB Calc_rms ! Evaluate the RMS signal level.
200 !
210 DISP "Do you wish to store that Speech Segment ? >"
220 Get_answer:ON KBD GOTO Answer
230 GOTO Get_answer
240 Answer:R$=UPCS$(KBDS$)
250 !
260 IF R$="Y" THEN
270 ON ERROR GOSUB Wrong_disc
280 MASS STORAGE IS Drive$
290 !
300 INPUT "Enter file name under which data is to be stored >",File_name$
310 !
320 File_name$=File_name$&"_0"
330 !
340 !~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ SPECIFY FILE NAME AND SIZE ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
350 !
360 !
370 Create:
380 !
390 ON ERROR GOTO Duplicate
400 CREATE BDAT File_name$,8*Block_size,2
410 OFF ERROR
420 !
430 !~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ SPECIFY I/O PATH NAME ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
440 !
450 !
460 DISP "Transferring data to disc."
470 ASSIGN @Data TO File_name$
480 !
490 FOR Next_in_buf=1 TO No_of_buffers
500 !
510 !~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ OUTPUT DATA TO DISC ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
520 !
530 SELECT Next_in_buf
540 CASE 1 OUTPUT @Data;Buf1(*)
550 CASE 2 OUTPUT @Data;Buf2(*)
560 CASE 3 OUTPUT @Data;Buf3(*)
570 CASE 4 OUTPUT @Data;Buf4(*)
580 CASE 5 OUTPUT @Data;Buf5(*)
590 CASE 6 OUTPUT @Data;Buf6(*)
600 CASE 7 OUTPUT @Data;Buf7(*)
610 CASE 8 OUTPUT @Data;Buf8(*)
620 END SELECT
630 !
640 NEXT Next_in_buf
ASSIGN @Data TO *
END IF
MASS STORAGE IS Drive_0$
RETURN

Close the I/O path.
RECOVER FROM A SYSTEM ERROR

```plaintext
10 !!!!!!!!!!!!!!!!!! RECOVER FROM A SYSTEM ERROR !!!!!!!!!!!!!!!!!!!!
20 !
30 !
40 Wrong_disc:!
50 IF ERRN=72 THEN ! Attempted to store data on non-existant disc.
60 BEEP 500,.2
70 PRINT "Hard disc is not present, so use default devices."
80 PRINT "Place an initialised disc in the LEFT drive."
90 PRINT "Hit ";CHR$(129);": CONTINUE ";CHR$(128);": when ready."
100 Drive_1$=":INTERNAL,4,1"
110 PAUSE
120 ELSE
130 DISP ERRN,ERRM$
140 STOP
150 END IF
160 RETURN
170 !
180 Duplicate: !A duplicate file name has been specified.
190 BEEP 1000,.1
200 DISP "A file of that name already exists. Must it be overwritten (Y/N)?"
210 ON KBD GOTO Overwrite
220 !
230 Spin: !
240 GOTO Spin
250 !
260 Overwrite: !
270 R$=UPC$(KBD$)
280 OFF KBD
290 !
300 IF R$="Y" THEN
310 DISP "Purging old version of the file."
320 PURGE File_name$
330 ELSE
340 INPUT "Enter a new file name for the data >",File_name$
350 File_name$=File_name$&"_0"
360 END IF
370 GOTO Create
380 !*************************************************************************
```
Calculate the RMS signal value.

`Calc_rms`:

100

`Disp "Calculating the RMS signal value."`

110

120 `MAT Buff1 = Buf1`

130 `MAT Buff2 = Buf2`

140 `MAT Buff3 = Buf3`

150 `MAT Buff4 = Buf4`

160 `MAT Buff1 = Buff1 . Buff1`

170 `MAT Buff2 = Buff2 . Buff2`

180 `MAT Buff3 = Buff3 . Buff3`

190 `MAT Buff4 = Buff4 . Buff4`

200 `MAT Buff2 = Buff1 + Buff2`

210 `MAT Buff4 = Buff3 + Buff4`

220 `MAT Buff4 = Buff2 + Buff4`

230 `Signal_pwr = SUM(Buff4)`

240 `MAT Buff1 = Buf5`

250 `MAT Buff2 = Buf6`

260 `MAT Buff3 = Buf7`

270 `MAT Buff4 = Buf8`

280 `MAT Buff1 = Buff1 . Buff1`

290 `MAT Buff2 = Buff2 . Buff2`

300 `MAT Buff3 = Buff3 . Buff3`

310 `MAT Buff4 = Buff4 . Buff4`

320 `MAT Buff2 = Buff1 + Buff2`

330 `MAT Buff4 = Buff3 + Buff4`

340 `MAT Buff4 = Buff2 + Buff4`

350 `Signal_rms = SQR(Signal_pwr + SUM(Buff4)/(8*Block_size))`

360 `PRINT USING "Input signal RMS level = " , 4D.DD ; Signal_rms`

370 `DISP`

380 `RETURN`

**************************************************************************
Problems:

160 IF Differential = 4095 THEN
170 BEEP 1500,.1
180 PRINT TABXY(8,9):CHR$(129):"MEMORY OVERFLOW HAS OCCURRED"
190 PRINT TABXY(11,10):"PROGRAM HAS BEEN HALTED";CHR$(128)
200 END IF
210 IF Underflow<>0 THEN
220 BEEP 1500,.1
230 PRINT TABXY(8,9):CHR$(129):"MEMORY UNDERFLOW HAS OCCURRED"
240 PRINT TABXY(11,10):"PROGRAM HAS BEEN HALTED";CHR$(128)
250 END IF
260 DISP " "
270 STOP
280 END
290 !*************************************************************************
SUB Xfer(Op$)  ! Transfer the special instruction from the supplemental instruction set kept on disc.
    CLEAR 823
    WAIT 4
    ENTER 82310;Dummy, Dummy, Dummy
    ALLOCATE Ascii$[80]
    Name$=Op$&"Z:INTERNAL,4,0"
    ASSIGN @Disc_file TO Name$
    ON END @Disc_file GOTO Eof

Rd_file: ENTER @Disc_file; Ascii$
    OUTPUT 823; Ascii$
    GOTO Rd_file

Eof: OFF END @Disc_file
    ASSIGN @Disc_file TO *
    DEALLOCATE Ascii$
    PRINT TABXY(5,11); "<MR> SUPPLEMENTAL INSTRUCTION LOADED INTO MULTI"
    CLEAR SELF TEST SRQ
    ENTER 82310; Complete, No_of_errors, No_of_cards
    WAIT 1
    DISP " "

SUBEND

********************************************************************************

SUB Multi_set_up(INTEGER Configuration(*), Tb_slot, Mem_in_slot, A_d_slot)
    OPTION BASE 1
    PRINT TABXY(5,13); "DETERMINING MULTI CONFIGURATION"
    FOR Slot_no=15 TO 0 STEP -1
        OUTPUT 823; "RF"; Slot_no; "T"
    NEXT Slot_no
    FOR Card_variable=1 TO 5
        ENTER 82304 USING "%,K"; Configuration(Slot_no, Card_variable)
        NEXT Card_variable
        SELECT Configuration(Slot_no, 1)
        CASE 1
            Tb_slot=Slot_no
        CASE 6
            Mem_in_slot=Slot_no
        CASE 52
            A_d_slot=Slot_no
        CASE ELSE
            END SELECT
    NEXT Slot_no
SUBEND

********************************************************************************
PROGRAM FOR INTERPOLATION FROM 8 kHz DATA TO A HIGHER RATE

CDDSIM : Codec Simulation Package.
Author : J.M. Irvine.
Department of Electrical and Electronic Engineering
University of Cape Town.
Unit Number : 2.

Programs developed for the Hewlett Packard 200 Series desk-top computers. All development work done on an HP9836C computer with the interfacing between the analogue and digital environments done using the HP6942A Multiprogrammer.

This program interpolates the sampled speech data from 8 kHz to an effective sampling rate which is selected by the user. Only integer multiples of the base sampling rate are realisable. The possible sampling rates available with this package are 16, 24, 32, 48 and 64 kHz. For a 10 second speech segment it is advisable to have a hard disc available as the mass storage device (a 5 1/4 inch discs cannot store sufficient data).

DEFINE THE VARIABLES USED BY THE PROGRAM

OPTION BASE 1

INTEGER I,J,K,Rate,Position,Last_flag,Rate_blocks,Block_size
DIM In_file$(8),Out_file$(8),File_name$(12)
DIM Function$(40),Error$(40)

ALLOCATE INTEGER Speech(1:10283) ! The buffer for the input speech.
ALLOCATE INTEGER Voice(1:8,1:10283) ! The buffer for the output speech.
ALLOCATE REAL Vbuff(1:10283) ! The filter operates on real values.
ALLOCATE REAL Window(1:121) ! The window length for filtering.

Run$="R" ! Control character to start program execution.
Stop$="!" ! Control character to stop program execution.
GOSUB Initialise  ! Reset the mode of the controller.
GOSUB General_param  ! Input the necessary file information, etc.
Rate_blocks=Rate/8  ! There are 8 blocks of data for 8 kHz
                      ! sampled data.
GOSUB Open_file  ! Open the speech data files.
FOR I=1 TO 8  ! Interpolate the 8 blocks of 8 kHz
              ! data.
              ! Read a block of input data.
              ! Insert null samples.
GOSUB Get_input  ! For J=1 TO Rate_blocks
Stretch(Rate,Speech(*),Voice(*))  ! Interpolate the 8 blocks of 8 kHz
Next I  ! data.
FOR K=1 TO Block_size  ! If the data is interpolated to 16
  Vbuff(K)=Voice(J,K)  ! kHz then only twice the number of
Next K  ! samples are produced. If to 24,
Filter(PI,Position,Last_flag,Rate,Window(*),Vbuff(*))  ! then three times, etc.
MAT Speech= Vbuff
GOSUB Save_output
NEXT J
NEXT I
ASSIGN @Source TO *  ! Close the source file.
ASSIGN @Destination TO *  ! Close the output file.
GOSUB Clear_disp  ! Clear the display screen.
! Return the user to the main menu of the simulation system.
Finish:  !
Function$="LOAD ""MASTER:INTERNAL,4,0"" X"
OUTPUT KBD;Stop$;Function$;Run$;
SUBROUTINES

SPECIFY THE DEFAULT MASS STORAGE DEVICES

Initialise:

Drive_0$=":INTERNAL,4,0"

Drive_1$=":INTERNAL,4,1"

PRINTER IS CRT

CONTROL 1,12;1

Initialise the values of the simulation system control parameters.

Block_size=10283

In_file$="TEST2"

Out_file$="TEST2"

Rate=16

Rate_blocks=8

Position=60

Last_flag=0

MAT Voice= (0)

MAT Vbuff= (0)

MAT Speech= (0)

MAT Window= (0)

Clear the arrays used for processing the speech data.

CLEAR THE ALPHANUMERIC AND GRAPHICS DISPLAYS

Clear disp:

GCLEAR

CONTROL 1,12;1

ALPHA OFF

PRINT USING "@"

PRINT CHR$(128):CHR$(140);

ALPHA ON

DISP " "

RETURN

Clear the graphics display.

Turn the key label display off.

Clear the Alpha display.

Select normal video with Colour = Cyan.
OPEN THE FILE CONTAINING THE SOURCE DATA

Open file:

Drive$ = Drive_0$

FILE STORAGE IS Drive$

File_name$ = In_file$ &"_0"

ASSIGN @Source TO File_name$

Create and open a file into which the interpolated data can be stored.

File name$ = Out_file$ &" P"

CREATE BDAT File_name$ , Rate_blocks*Block_size, 2

ASSIGN @Destination TO File_name$

READ THE DATA FROM THE FILES STORED ON DISC

Get input:

File $ ="Reading speech data off disc."

ENTER @Source;Speech(*)

STORE THE INTERPOLATED SPEECH DATA

Save output:

DISP "Writing interpolated data to disc."

OUTPUT @Destination;Speech(*)

RETURN
30 !<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
CODEC SIMULATION PROGRAMS FOR SUBJECTIVE ANALYSIS

CODSIM : Codec Simulation Package.
Author : J.M. Irvine.
Department of Electrical and Electronic Engineering
University of Cape Town.
Unit Number : 3.

Programs developed for the Hewlett Packard 200 Series desk-top computers. All development work done on an HP9836C computer with the interfacing between the analogue and digital environments done using the HP6942A Multiprogrammer.

This program processes the speech data, stored on disc, using the simulated delta modulators (i.e. the LDM, CVSD, CFDM, SVADM, SHCDM, MSHCDM, and HCDH coders). Once the data has been processed by the relevant codec it is stored on disc, thus enabling subjective tests to be performed using this processed data.

<<<<<<<<<<< DEFINE THE VARIABLES USED BY THE PROGRAM >>>>>>>>>>>>

OPTION BASE 1
INTEGER Syllable,Err1,Err2,Err3,So
REAL Alpha,Beta,Step,New_step
REAL Big_alpha,Signal,New_signal
REAL Max_step,Max_cf,Max_sig,Min_sig
INTEGER Position,Processor_no,So_start,So_fin
INTEGER Syll_start,Syll_fin,Proc_start,Proc_fin
INTEGER So_inc,Syll_inc,Proc_inc
REAL Rate,Alpha_start,Alpha_fin,Beta_start
REAL Beta_fin,Alpha_inc,Beta_inc

DIM In_file$[8],Out_file$[8],File_name$[12]
DIM Function$[40],Error$[40]

ALLOCATE INTEGER Speech(1:10283) ! The buffer for the input speech.
ALLOCATE INTEGER P_speech(1:10283) ! The buffer for processed speech.
ALLOCATE REAL Voice(1:10283) ! Codecs operate on real values.
ALLOCATE REAL Syllab(1:1000) ! Syllabic companding filter.
ALLOCATE REAL Window(1:60) ! The window length for filtering.

Run$="R" ! Control character to start program execution.
Stop$="!" ! Control character to stop program execution.
GOSUB Initialise
  ! Reset the mode of the controller.
GOSUB Set_parameters
  ! Set the parameters controlling the codec
type and the parameter ranges.
File_count=0
  ! Initialise output file count parameter.
GOSUB Open_file
  ! Open the file containing the speech data.

FOR Processor_no=Proc_start TO Proc_fin STEP Proc_inc
  ! It is possible to process the speech data using a range of different
  ! parameter settings. This enables the effects of the parameters to be
evaluated subjectively.

FOR Alpha=Alpha_start TO Alpha_fin STEP Alpha_inc
  FOR Beta=Beta_start TO Beta_fin STEP Beta_inc
    FOR So=So_start TO So_fin STEP So_inc
      FOR Syllable=Syll_start TO Syll_fin STEP Syll_inc

        RESET @Source
        ! Reset the source file to start.
        File_count=File_count+1
        ! The number of output files.
        GOSUB Open_outfile
        ! Open a file for the output data.
        MAT Syllab= (0)
        ! Clear the syllabic filter.
        MAT Window= (0)
        ! Clear the digital output filter.
        GOSUB Print_message
        ! Output a status message to VDU.
        FOR Blocks=1 TO Rate_blocks
          DISP "Processing ";Blocks:" out of ";Rate_blocks;" of ";Rate;"
hz data."
          GOSUB Get_input
          ! Read the input speech data off disc.
          MAT Voice= Speech
          ! Copy from Integer to Real.

          ! The procedure "Codecs", which performs the simulation,
          ! consists of a compiled PASCAL module. The listing of this
          ! module is contained in Unit no. 6.

          Codecs(So,Err1,Pos,Err2,Err3,Processor_no,Syllable,Beta,Rate,A
          Apha,Signal,Max_cf,Max_sig,Min_sig,New_step,Max_step,Big_alpha,New_signal,Syllab
            IF Filter_flag=1 THEN
              Filter(Rate,Window(*),Voice(*))
          END IF
          !
          MAT P_speech= Voice
          ! Make an integer copy of the speech.
          GOSUB Save_output
          ! Copy the processed data into a file.
          NEXT Blocks
          ASSIGN @Destination TO * ! Close the file containing output.
          NEXT Syllable
          NEXT So
          NEXT Beta
          NEXT Alpha
          NEXT Processor_no
ASSIGN @Source TO *  ! Close the source file.
GOSUB Clear_disp  ! Clear the display screen.

! Return the user to the main menu of the simulation system.

Function$="LOAD ""MASTER:INTERNAL.4,0"" X"
OUTPUT KBD;Stop$;Function$;Run$;
!
SUBROUTINES

SPECIFY THE DEFAULT MASS STORAGE DEVICES

Initialise:

Drive_0$=":INTERNAL,4,0"
Drive_1$=":INTERNAL,4,1"
PRINTER IS CRT
CONTROL 1,12;1

Initialise the values of the simulation system control parameters.

in_file$="TEST2"
out_file$="TEST2"

Rate=16
Rate_blocks=8
Proc_start=1
Proc_fin=1
Proc_inc=1
So_start=1
So_fin=1
So_inc=1
Alpha_start=.25
Alpha_fin=.25
Alpha_inc=.01
Beta_start=1.53
Beta_fin=1.53
Beta_inc=.01
Syll_start=5
Syll_fin=5
Syll_inc=1
Max_step=1000
Max_cf=1.5^17
Min_sig=-32768
Max_sig=+32767

Clear disp:

GCLEAR
CONTROL 1,12;1
ALPHA OFF
PRINT USING "@"
PRINT CHR$(128);CHR$(140);
ALPHA ON
DISP " ",
RETURN

Clear the graphics display.
Turn the key label display off.
Clear the Alpha display.
Select normal video with Colour = Cyan.
! OPEN THE FILE CONTAINING THE SOURCE DATA >>>>>>>>>>>>>>>>>>>>

OPEN_FILE:

Drive$=Drive_0$
MASS STORAGE IS Drive$
File_name$=In_file$"P"
ASSIGN @Source TO File_name$
RETURN

READ THE DATA FROM THE FILES STORED ON DISC >>>>>>>>>>>>

GET_INPUT:

DISP "Reading speech data off disc."
ENTER @Source;Speech(*)
DISP
RETURN

OPEN A FILE FOR THE PROCESSED SPEECH DATA >>>>>>>>>>>>

OPEN_OUTFILE:

Drive$=Drive_1$
MASS STORAGE IS Drive$
!
Update the name of the output file.
File_name$=Out_file$"P"
CREATE BDAT File_name$,Rate_blocks*Block_size,2
ASSIGN @Destination TO File_name$
RETURN

TRANSFER THE PROCESSED DATA TO THE OUTPUT FILE >>>>>>>>

SAVE_OUTPUT:

DISP "Transferring processed speech to disc"
OUTPUT @Destination;P_speech(*)
DISP
RETURN

*************************************************************************
PRINT A MESSAGE ON THE VDU FOR THE USER

GOSUB Clear_disp

PRINT "Processing the speech data contained in ";In_file$
PRINT "Processing using a ";
SELECT Processor_no
CASE 1
  PRINT "Linear Delta Modulator";
CASE 2
  PRINT "Continuously Variable Slope Delta Modulator";
CASE 3
  PRINT "Constant Factor Delta Modulator";
CASE 4
  PRINT "Song Voice Adaptive Delta Modulator";
CASE 5
  PRINT "Song Hybrid Companding Delta Modulator";
CASE 6
  PRINT "Modified Song Hybrid Companding Delta Modulator";
CASE 7
  PRINT "Hybrid Companding Delta Modulator";
END SELECT
PRINT
IF Filter_flag=1 THEN
  PRINT "Digital filtering > YES."
ELSE
  PRINT "Digital filtering > NO."
END IF
PRINT
RETURN
SET THE CONTROL PARAMETERS OF THE SYSTEM

Set_parameters:

This subroutine enables all the variable and parameter values to be set. First the codecs that are to be used are enabled and then the variables relating to each are set. Also, the source and destination files are specified, as well as the data rate.

DISPLAY THE CODEC-TYPE MENU

- Set_parameters:

  ! This subroutine enables all the variable and parameter values to be set.
  ! First the codecs that are to be used are enabled and then the variables
  ! relating to each are set. Also, the source and destination files are
  ! specified, as well as the data rate.

- Set_parameters:

  ! Set the control parameters of the system.

DISPLAY THE CODEC-TYPE MENU

  ! This subroutine enables all the variable and parameter values to be set.
  ! First the codecs that are to be used are enabled and then the variables
  ! relating to each are set. Also, the source and destination files are
  ! specified, as well as the data rate.

- Set_parameters:

  ! Set the control parameters of the system.

DISPLAY THE CODEC-TYPE MENU

  ! This subroutine enables all the variable and parameter values to be set.
  ! First the codecs that are to be used are enabled and then the variables
  ! relating to each are set. Also, the source and destination files are
  ! specified, as well as the data rate.

- Set_parameters:

  ! Set the control parameters of the system.

DISPLAY THE CODEC-TYPE MENU

  ! This subroutine enables all the variable and parameter values to be set.
  ! First the codecs that are to be used are enabled and then the variables
  ! relating to each are set. Also, the source and destination files are
  ! specified, as well as the data rate.

- Set_parameters:

  ! Set the control parameters of the system.
PRINT TABXY(19,15);Proc_start:TAB(30);Proc_fin:TAB(42);Proc_inc
DISP "Do you wish to keep these values (Y/N) ?"

ON KBD GOTO Answer

Wait:!
GOTO Wait

Answer:!
R$=UPC$(KBD$)
IF (R$="Y") OR (R$=" E") THEN
    IF (Proc_start<>0) OR (Proc_fin<>0) THEN
        GOTO Finish
    END IF
ELSE
    GOTO Alter_values
END IF

!------------------------------------------------------------------------

! Update the various design parameters of the different codecs.

GOSUB Clear_disp

Design_param:

PRINT TABXY(18,2):CHR$(129);"Codec algorithm parameters: ":;CHR$(128);CHR$(10)
PRINT TAB(21);CHR$(132);"Minimum step size (So)";CHR$(128);CHR$(10)
PRINT TAB(18);CHR$(132);"Start";CHR$(128);TAB(30);CHR$(132);"Stop";CHR$(128)
TAB(41);CHR$(132);"Increment";CHR$(128)
PRINT TAB(19);CHR$(129);So_start:CHR$(128);TAB(31);So_fin:TAB(42);So_inc
DISP "Enter the desired value"
GOSUB Keyboard_input
IF Value>O THEN
    So_start=Value
END IF
PRINT TABXY(19,7);So_start:TAB(29);CHR$(129);So_fin:CHR$(128);TAB(42);So_inc
GOSUB Keyboard_input
IF Value>O THEN
    So_fin=Value
END IF
PRINT TABXY(19,7);So_start:TAB(29);So_fin:TAB(40);CHR$(129);So_inc:CHR$(128)
INPUT "Enter the desired value",Increment
IF Increment<>O THEN
    So_inc=Increment
    Increment=O
END IF
PRINT TABXY(19,7);So_start:TAB(29);So_fin:TAB(40);So_inc

!------------------------------------------------------------------------
510 PRINT TABXY(18,12);CHR$(132);"Leakage time constant (Beta)";CHR$(128);CHR$(120)
520 PRINT TAB(18);CHR$(132);"Start";CHR$(128);TAB(30);CHR$(132);"Stop";CHR$(128)
530 PRINT TAB(41);CHR$(132);"Increment";CHR$(128)
540 Value=0
550 INPUT "Enter the desired value",Value
560 IF Value<>0 THEN
570 Beta_start=Value
580 Value=0
590 END IF
600 PRINT TABXY(17,15);Beta_start;TAB(27);CHR$(129);Beta_fin;CHR$(128);TAB(41)
610 INPUT "Enter the desired value",Value
620 IF Value<>0 THEN
630 Beta_fin=Value
640 Value=0
650 END IF
660 PRINT TABXY(17,15);Beta_start;TAB(27);Beta_fin;CHR$(129);Beta_inc;CHR$(128)
670 INPUT "Enter the desired value",Value
680 IF Value<>0 THEN
690 Beta_inc=Value
700 Value=0
710 END IF
720 PRINT TABXY(17,15);Beta_start;TAB(27);Beta_fin;TAB(39);CHR$(129);Beta_inc;CHR$(128)
730 INPUT "Enter the desired value",Value
740 IF Value<>0 THEN
750 Beta_inc=Value
760 Value=0
770 END IF
780 PRINT TABXY(17,15);Beta_start;TAB(27);Beta_fin;TAB(39);Beta_inc
790 DISP "Do you wish to keep these values (Y/N)?"
800 IF Value<>0 THEN
810 Beta_inc=Value
820 Value=0
830 END IF
840 PRINT TABXY(17,15);Beta_start;TAB(27);Beta_fin;TAB(39);Beta_inc
850 PRINT TABXY(17,15);Beta_start;TAB(27);Beta_fin;TAB(39);Beta_inc
860 ON KBD GOTO Answer_2
870 Wait_2:!!
880 GOTO Wait_2
890 Answer_2:!!
900 R$=UPC$(KBD$)
910 IF R$="N" THEN
920 GOTO Design_param
930 END IF
940!!
950!!
The parameters for the hybrid codecs are only updated if the range of of the codec types, chosen previously, includes one or more of the hybrid codecs.

IF (Proc_start<4) AND (Proc_fin<4) THEN
    GOTO General_param
END IF

GOSUB Clear disp

Hybrid_param: !

PRINT TABXY(23,2):CHR$(129):"Hybrid Design Parameters :";CHR$(128):CHR$(10)
PRINT TAB(18):CHR$(132):"Syllabic Companding Constant (Alpha)";CHR$(128):CHR$(10)
PRINT TAB(23):CHR$(132):"Start";CHR$(128):TAB(35):CHR$(132):"Stop";CHR$(128)
PRINT TAB(46):CHR$(132):"Increment";CHR$(128)
PRINT TAB(23):CHR$(129):Alpha_start;CHR$(128):TAB(35):Alpha_fin;TAB(46):Alpha_inc
PRINT "Enter the desired value",Value
IF Value<>0 THEN
    Alpha_start=Value
    Value=0
END IF
PRINT TABXY(23,7):Alpha_start:TAB(33):Alpha_fin:CHR$(129):TAB(44):Alpha_inc
PRINT "Enter the desired value",Value
IF Value<>0 THEN
    Alpha_fin=Value
    Value=0
END IF
PRINT "Enter the desired value",Value
IF Value<>0 THEN
    Alpha_inc=Value
    Value=0
END IF

340 PRINT TABXY(24,12);CHR$(132);"Syllabic Companding Period";CHR$(128);CHR$(128);CHR$(128);CHR$(128)
350 PRINT TAB(23);CHR$(132);"Start";CHR$(128);TAB(35);CHR$(132);"Stop";CHR$(128);TAB(46);CHR$(132);"Increment";CHR$(128)
360 PRINT TABXY(23,15);CHR$(129);Syll_start;CHR$(128);TAB(35);Syll_fin;TAB(47)
Syll_inc
370 VALUE=0
380 INPUT "Enter the desired value (milliseconds)",Value
390 IF Value<>0 THEN
400 Syll_start=Value
410 Value=0
420 END IF
430 PRINT TABXY(23,15);Syll_start;TAB(33);CHR$(129);Syll_fin;CHR$(128);TAB(47)
Syll_inc
440 INPUT "Enter the desired value (milliseconds)",Value
450 IF Value<>0 THEN
460 Syll_fin=Value
470 Value=0
480 END IF
490 PRINT TABXY(23,15);Syll_start;TAB(33);Syll_fin;TAB(45);CHR$(129);Syll_inc;R$(128)
500 INPUT "Enter the desired value (milliseconds)",Value
510 IF Value<>0 THEN
520 Syll_inc=Value
530 Value=0
540 END IF
550 PRINT TABXY(23,15);Syll_start;TAB(33);Syll_fin;TAB(45);Syll_inc
560 DISP "Do you wish to keep these values (Y/N) ?"
570 !
580 ON KBD GOTO Answer_3
590 !
600 Wait_3:
610 GOTO Wait_3
620 !
630 Answer_3: !
640 R$=UPC$(KBDS)
650 IF R$="N" THEN
660 GOTO Hybrid_param
670 END IF
680 !
690 !-------------------------------------------------------------------
The general parameters are updated at this stage - eg. Maximum step sizes, maximum output signal ranges, source file names, destination file names, etc.

General_param:

GOSUB Clear_disp

PRINT TAB(22);CHR$(132);"Speech file information :";CHR$(128);CHR$(10)
PRINT TAB(20);"Source file name :";CHR$(129);In_file$;CHR$(128);CHR$(10)
PRINT TAB(20);"Destination file name :";Out_file$;CHR$(10)
PRINT TAB(20);"Maximum signal value :";Max_sig;CHR$(10)
PRINT TAB(20);"Minimum signal value :";Min_sig;CHR$(10)
PRINT TAB(20);"Maximum step size :";Max_step;CHR$(10)
PRINT TAB(20);"Sampling rate (kHz) :";Rate;CHR$(10)
PRINT TAB(10);CHR$(129);"The default setting is selected by pressing 'ENT'";CHR$(128)

INPUT "Enter the source file name", In_file$
In_file$=UPC$(In_file$)
PRINT TABXY(20,4);"Source file name :";In_file$;CHR$(10)
PRINT TAB(20);"Destination file name :";CHR$(129);Out_file$;CHR$(128)
INPUT "Enter the destination file name", Out_file$
Out_file$=UPC$(Out_file$)
PRINT TABXY(20,6);"Destination file name :";Out_file$;CHR$(10)
PRINT TAB(20);"Maximum signal value :";CHR$(129);Max_sig;CHR$(128)

Value=0
INPUT "Enter the desired value", Value
IF Value>0 THEN
Max_sig=Value
Value=0
END IF
PRINT TABXY(20,8);"Maximum signal value :";Max_sig;CHR$(10)
PRINT TAB(20);"Minimum signal value :";CHR$(129);Min_sig;CHR$(128)

INPUT "Enter the desired value", Value
IF Value>0 THEN
Min_sig=Value
Value=0
END IF
PRINT TABXY(20,10);"Minimum signal value :";Min_sig;CHR$(10)
PRINT TAB(20);"Maximum step size :";CHR$(129);Max_step;CHR$(128)

INPUT "Enter the desired value", Value
IF Value>0 THEN
Max_step=Value
Value=0
END IF
PRINT TABXY(20,12);"Maximum step size :";Max_step;CHR$(10)
PRINT TAB(20);"Sampling rate (kHz) : ";CHR$(129);Rate;CHR$(128)
INPUT "Enter the sampling rate (kHz) : 16, 24, 32, 48, or 64",Value
IF (Value=16) OR (Value=24) OR (Value=32) OR (Value=48) OR (Value=64) THEN
  Rate=Value
END IF
PRINT TABXY(20,14);"Sampling rate (kHz) : ";Rate;CHR$(10)
PRINT TAB(80)
! DISP "Do you require the processed speech to be filtered (Y/N) ?"
ON KBD GOTO Answer_4
Wait_4:
GOTO Wait_4
Answer_4:
R$=UPC$(KBD$)
IF R$="N" THEN / Filter_flag=0
END IF
Rate_blocks=Rate_blocks*Rate/8
Syll_start=INT(Rate*Syll_start) ! Convert from a time value to
Syll_fin=INT(Rate*Syll_fin) ! a number of samples (ie. the
Syll_inc=INT(Rate*Syll_inc) ! no. of samples in the syllabic
RETURN
!************************************************************************
Keyboard_input: !
ON KBD GOTO Get_response
Waiting: !
GOTO Waiting
Get_response: !
ON ERROR GOTO Waiting
Value=VAL(KBD$)
OFF KBD
OFF ERROR
RETURN
!************************************************************************
END
CODEC SIMULATION PROGRAMS FOR OBJECTIVE ANALYSIS

CODSIM


Author : J.M. Irvine,
Department of Electrical and Electronic Engineering
University of Cape Town.

Unit Number : 4.

Programs developed for the Hewlett Packard 200 Series desk-top computers. All development work done on an HP9836C computer with the interfacing between the analogue and digital environments done using the HP6942A Multiprogrammer.

This program processes the speech data, stored on disc, using the simulated delta modulators (ie. the LDM, CVSD, CFDM, SVADM, SHCDM, MSHCDM, and HCDM coders). Once the data has been processed by the relevant codec the output is filtered (if required) by a digital filter and the noise signals extracted. The signal-to-noise ratio (SNR) and the segmental signal-to-noise ratio (SNRSEG) are calculated and, if specified, the respective granular and segmental values are evaluated.
DEFINE THE VARIABLES USED BY THE PROGRAM >>>>>>>>>>>>>>>

OPTION BASE 1

INTEGER Syllable,Err1,Err2,Err3,So
REAL Alpha,Beta,Step,New_step
REAL Big_alpha,Signal,New_signal
REAL Max_step,Max_cf,Max_sig,Min_sig

INTEGER Position,Processor_no,So_start,So_fin
INTEGER Syll_start,Syll_fin,Proc_start,Proc_fin
INTEGER So_inc,Syll_inc,Proc_inc
REAL Rate,Alpha_start,Alpha_fin,Beta_start
REAL Beta_fin,Alpha_inc,Beta_inc

REAL Signal_pwr,Seg_signal_pwr,Total_noise
REAL Segmental_noise,Total_gran,Segmental_gran
REAL Total_over,Segmental_over
REAL Sig_cnt,Seg_signal_cnt,Noise_cnt
INTEGER Seg_noise_cnt,Gran_cnt,Seg_gran_cnt
INTEGER Over_cnt,Seg_over_cnt

DIM In_file$(8),File_name$(12)
DIM Function$(40),Error$(40)

ALLOCATE INTEGER Speech(1:10283) ! The buffer for the input speech.
ALLOCATE REAL Voice(1:10283) ! Codecs operate on real values.
ALLOCATE REAL Voice2(1:10283)
ALLOCATE REAL Syllab(1:1000) ! Syllabic companding filter.
ALLOCATE REAL Window(1:60) ! The window length for filtering.

ALLOCATE Seg_sig(1:515) ! Array for segmental calculations.
ALLOCATE Seg_noise(1:515),Seg_gran(1:515),Seg_over(1:515)

Run$=" R" ! Control character to start program execution.
Stop$=" !" ! Control character to stop program execution.

************************************************************************
GO SUB Initialise ! Reset the mode of the controller.
GO SUB Set_parameters ! Set the parameters controlling the codec
                    ! type and the parameter ranges.
GO SUB Open_file   ! Open the file containing the speech data.
GO SUB Print_heading ! Print a heading for the results.
FOR Processor_no=Proc_start TO Proc_fin STEP Proc_inc
    ! It is possible to process the speech data using a range of different
    ! parameter settings. This enables the effects of the parameters to be
    ! evaluated subjectively.
    !--------------------------------------------------------------~-------
FOR Alpha=Alpha_start TO Alpha_fin STEP Alpha_inc
    FOR Beta=Beta_start TO Beta_fin STEP Beta_inc
        FOR Scale=Scale_start TO Scale_fin STEP Scale_inc
            RESET @Source ! Reset the source file to start.
            MAT Syllab= (0) ! Clear the syllabic filter.
            MAT Window= (0) ! Clear the digital output filter.
            GO SUB Print_message ! Output a status message to
                                   VDU.
            FOR Blocks=1 TO Rate_blocks
                DISP "Processing ";Blocks:" out of ";Rate_blocks:" of ";Rate
            GO SUB Get_input ! Read the input speech data off disc.
            MAT Voice= Speech ! Copy from Integer to Real.
                ! Amplify the speech signal.
            MAT Voice= (10^Scale)*Voice
                ! Update the lumped and segmental signal powers.
            00 Noises(Signal_sgn,Sig_cnt,0,0,Seg_sgn_cnt,Samples_per_seg,Sig
                              al_pwr,Seg_signal_pwr,Seg_sgn(•),Voice(•))
            ! The procedure "Codecs", which performs the simulation,
            ! consists of a compiled PASCAL module. The listing of
            ! this module is contained in Unit no. 6.
            !----------------------------------------------------------
00 Codec(So,Err1,Pos,Err2,Err3,Processor_no,Syllable,Beta,Rate
            Alpha,Signal,Max_cf,Max_sig,Min_sig,New_step,Max_step,Big_alpha,New_signal,Syl
            IF Filter_flag=1 THEN
            END IF
Extract the noise signal.

MAT Voice = Voice-Speech
MAT Voice2 = Voice

Update the lumped and segmental noise powers.

Noises(Noise_sgn, Noise_cnt, 0, 0, Seg_noise_cnt, Samples_per_seg, Total_noise, Segmental_noise, Seg_noise(*), Voice(*))

Update the lumped and segmental granular noise powers.

IF Gran_flag = 1 THEN
Noises(Noise_sgn, Gran_cnt, 1, 0, Seg_gran_cnt, Samples_per_seg, Total_gran, Segmental_gran, Seg_gran(*), Voice(*))

END IF

MAT Voice = Voice2
Noises(Noise_sgn, Over_cnt, 0, 1, Seg_over_cnt, Samples_per_seg, Total_over, Segmental_over, Seg_over(*), Voice(*))

END IF

GOSUB Ratios

NEXT Blocks

PRINT USING "#.D,2X,D,DD,2X,D,DD,2X,D,DD,2X,6D,4X,D,DD,3X"; Processor_no, Alpha, Beta, INT(Syllable/Rate), So, Max_step, Scale
PRINT USING "DD,D,3X,DD,D,2X,D,DD,3X,DD,D,2X,D,DD,3X,DD,D"; Sqn, Snr_seg, Sng, Sng_seg, Sno, Sno_seg

ASSIGN @Destination TO *! Close the file containing output.

NEXT Scale
NEXT Syllable
NEXT So
NEXT Beta
NEXT Alpha
NEXT Processor_no

ASSIGN @Source TO *! Close the source file.

GOSUB Clear Disp

RETURN the user to the main menu of the simulation system.

Finish:

Function$ = "LOAD ""MASTER:INTERNAL.4.0"" X"
OUTPUT KBD;Stop$;Function$;Run$;

***********************************************************************
SUBROUTINES

SPECIFY THE DEFAULT MASS STORAGE DEVICES

Initialise:

Drive_9$=":INTERNAL,4,0"
Drive_1$=":INTERNAL,4,1"
PRINTER IS CRT
CONTROL 1,12;1

Initialise the values of the simulation system control parameters.

In_file$="TEST2"
Out_file$="TEST2"

Rate=15
Rate_blocks=8
Proc_start=1
Proc_fin=1
Proc_inc=1
So_start=1
So_fin=1
So_inc=1
Alpha_start=.25
Alpha_fin=.25
Alpha_inc=.01
Beta_start=1.53
Beta_fin=1.53
Beta_inc=.01
Syll_start=5
Syll_fin=5
Syll_inc=1
Scale_start=0
Scale_fin=0
Scale_inc=1
Max_step=1000
Max_cf=1.5,17
Min_sig=-32768
Max_sig=+32767
Filter_flag=1
Gran_flag=1
Clear the graphics display.
Clear the Alpha display.
Select normal video with Colour = Cyan.

Open file:
Drive$=Drive_0$
MASS STORAGE IS Drive$
File_name$=In_file$&".P"
ASSIGN @Source TO File_name$
RETURN

Open the file containing the source data >>>>>>>>>>>>>>>>>>>

Get input:
DISP "Reading speech data off disc."
ENTER @Source:Speech(*)
DISP
RETURN

*************************************************************************

**************************************************************************
Print_message:
GOSUB Clear_disp
PRINT "Processing the speech data contained in " : In_file$;
SELECT Processor_no
CASE 1
PRINT "Linear Delta Modulator":
CASE 2
PRINT "Continuously Variable Slope Delta Modulator":
CASE 3
PRINT "Constant Factor Delta Modulator":
CASE 4
PRINT "Song Voice Adaptive Delta Modulator":
CASE 5
PRINT "Song Hybrid Companding Delta Modulator":
CASE 6
PRINT "Modified Song Hybrid Companding Delta Modulator":
CASE 7
PRINT "Hybrid Companding Delta Modulator":
END SELECT
IF Filter_flag = 1 THEN
PRINT "Digital filtering > YES.
ELSE
PRINT "Digital filtering > NO.
END IF
PRINT
RETURN
Print_heading:
PRINTER IS CRT
PRINT "CODSIM: Objective evaluation of codec performance."
PRINT "Filtering : ";
IF Gran_flag = 1 THEN
PRINT "Y ";
ELSE
PRINT "N ";
END IF
PRINT " Granular and overload SNR's : ";
IF Gran_flag = 1 THEN
PRINT "Y"
ELSE
PRINT "N"
END IF
PRINT "P Alph Beta. Syll So Maxstt Scale SQNR SNRSEG SNG SNGSEG
SNO SNOSEG"
FOR I = 1 TO 80
PRINT "-"; NEXT I
PRINT PRINTER IS CRT
RETURN
CALCULATE THE RESPECTIVE SEGMENTAL VALUES

Snr_seg = 0
Sng_seg = 0
Sno_seg = 0

FOR I = 1 TO 515
  IF Segmental_noise(I) <> 0 THEN
    Snr_seg = Snr_seg + 10 * LGT(Seg_signal_pwr(I) / Segmental_noise(I))
  END IF
  IF Gran_flag = 1 THEN
    IF Segmental_gran(I) <> 0 THEN
      Sng_seg = Sng_seg + 10 * LGT(Seg_signal_pwr(I) / Segmental_gran(I))
    END IF
    IF Segmental_over(I) <> 0 THEN
      Sno_seg = Sno_seg + 10 * LGT(Seg_signal_pwr(I) / Segmental_over(I))
    END IF
  END IF
NEXT I

Snr_seg = Snr_seg / 515
Sng_seg = Sng_seg / 515
Sno_seg = Sno_seg / 515

MAT Seg_signal_pwr = (0)
MAT Segmental_noise = (0)
MAT Segmental_gran = (0)
MAT Segmental_over = (0)

RETURN
Set_parameters:

This subroutine enables all the variable and parameter values to be set. First, the codecs that are to be used are enabled and then the variables relating to each are set. Also, the source and destination files are specified, as well as the data rate.

DISPLAY THE CODEC-TYPE MENU

Alter_values: Enter the required codec types.

A codec type '0' returns to main menu.

"Enter the desired value"

"Enter the desired value", Increment

Increment=0

Increment>0 THEN

Proc_inc=Increment

Increment=0

END IF
4610 "Do you wish to keep these values (Y/N)?"
4620 ON KBD GOTO Answer
4630 Wait:
4640 GOTO Wait
4660 !
4670 Answer: !
4680 RS=UPC$(KBDS)
4690 IF (RS="Y") OR (RS="E") THEN ! A value of zero for a codec
4700 IF (Proc_start=0) OR (Proc_fin=0) THEN ! type returns the user to
4710 GOTO Finish ! the main menu.
4720 END IF
4730 ELSE
4740 GOTO Alter_values
4750 END IF

! Update the various design parameters of the different codecs.
4780 Gosub Clear_disp
4790 Design_par:\!
4800 !
4810 PRINT TABXY(18,2):CHR$(129):"Codec algorithm parameters:";CHR$(129):CHR$(
4820 PRINT TAB(21):CHR$(132):"Minimum step size (So)";CHR$(128):CHR$(10)
4830 PRINT TAB(18):CHR$(132):"Start";CHR$(128);TAB(30);CHR$(132);"Stop";CHR$(12
4840 PRINT TAB(41);CHR$(132);"Increment";CHR$(128)
4850 DISP "Enter the desired value"
4860 Gosub Keyboard_input
4870 IF Value>0 THEN
4880 So_start=Value
4890 END IF
4910 Gosub Keyboard_input
4920 IF Value>0 THEN
4930 So_fin=Value
4940 END IF
4960 INPUT "Enter the desired value",Increment
4970 IF Increment>0 THEN
4980 So_inc=Increment
4990 Increment=0
5000 END IF
5020 !
5080 PRINT TABXY(18,12);CHR$(132);"Leakage time constant (Beta)";CHR$(128);CHR$(10)
5090 PRINT TAB(18);CHR$(132);"Start";CHR$(128);TAB(30);CHR$(132);"Stop";CHR$(128)
5100 PRINT TAB(41);CHR$(132);"Increment";CHR$(128)
5110 Value=0
5120 INPUT "Enter the desired value".Value
5130 IF Value<>0 THEN
5140 Beta_start=Value
5150 Value=0
5160 END IF
5170 PRINT TABXY(17,15);Beta_start;TAB(27);CHR$(129);Beta_fin;CHR$(128);TAB(41)
5180 INPUT "Enter the desired value".Value
5190 IF Value<>0 THEN
5200 Beta_fin=Value
5210 Value=0
5220 END IF
5230 PRINT TABXY(17,15);Beta_start;TAB(27);Beta_fin;TAB(39);CHR$(129);Beta_inc
5240 INPUT "Enter the desired value".Value
5250 IF Value<>0 THEN
5260 Beta_inc=Value
5270 Value=0
5280 END IF
5290 PRINT TABXY(17,15);Beta_start;TAB(27);Beta_fin;TAB(39);Beta_inc
5300 DISP "Do you wish to keep these values (Y/N)?"
5310 !
5320 ON KBD GOTO Answer_2
5330 !
5340 Wait_2!:!
5350 GOTO Wait_2
5360 !
5370 Answer_2: !
5380 RS=UPC$(KBDS)
5390 IF RS="N" THEN
5400 GOTO Design_param
5410 END IF
5420 !
5430 !--------------------------
The parameters for the hybrid codecs are only updated if the range of
of the codec types, chosen previously, includes one or more of the
hybrid codecs.

IF (Proc_start<4) AND (Proc_fin<4) THEN
  GOTO General_param
END IF

GOSUB Clear_disp

Hybrid_param:

PRINT TABXY(23.2);CHR$(129);"Hybrid Design Parameters ":CHR$(128);CHR$(10)
PRINT TAB(18);CHR$(132);"Syllabic Companding Constant (Alpha)":CHR$(128):CHR$(10)
PRINT TAB(23);CHR$(132);"Start":CHR$(128):TAB(35):CHR$(132);"Stop":CHR$(128)
PRINT TAB(46);CHR$(132);"Increment":CHR$(128)
PRINT TAB(23);CHR$(129);Alpha_start:CHR$(128):TAB(35);Alpha_fin:TAB(46):Alpha_inc

Value=0
INPUT "Enter the desired value",Value
IF Value<>0 THEN
  Alpha_start=Value
  Value=0
END IF

INPUT "Enter the desired value",Value
IF Value<>0 THEN
  Alpha_fin=Value
  Value=0
END IF

INPUT "Enter the desired value",Value
IF Value<>0 THEN
  Alpha_inc=Value
  Value=0
END IF

------------------------------------------------------------------------
5810 PRINT TABXY(24,12);CHR$(132);"Syllabic Companding Period";CHR$(128);CHR$(10)
5820 PRINT TAB(23);CHR$(132);"Start";CHR$(128);TAB(35);CHR$(132);"Stop";CHR$(128)
5830 PRINT TAB(46);CHR$(132);"Increment";CHR$(128)
5840 VALUE=0
5850 INPUT "Enter the desired value (milliseconds)".Value
5860 IF Value<>0 THEN
5870 Syll_start=Value
5880 Value=0
5890 END IF
5900 PRINT TABXY(23,15):Syll_start;TAB(33):CHR$(129):Syll_fin;CHR$(128):TAB(47)
5910 INPUT "Enter the desired value (milliseconds)".Value
5920 IF Value<>0 THEN
5930 Syll_fin=Value
5940 Value=0
5950 END IF
5960 PRINT TABXY(23,15):Syll_start;TAB(33):Syll_fin;TAB(45);CHR$(129):Syll_inc:CHR$(128)
5970 INPUT "Enter the desired value (milliseconds)".Value
5980 IF Value<>0 THEN
5990 Syll_inc=Value
6000 Value=0
6010 END IF
6020 PRINT TABXY(23,15):Syll_start;TAB(33):Syll_fin;TAB(45):Syll_inc
6030 DISP "Do you wish to keep these values (Y/N) ?"
6040!
6050 ON KBD GOTO Answer_3
6060!
6070 Wait_3:!
6080 GOTO Wait_3
6090!
6100 Answer_3:!
6110 RS=UPC$(KBD$)
6120 IF RS="N" THEN
6130 GOTO Hybrid_param
6140 END IF
6150!
6160!-----------------------------------------------
The general parameters are updated at this stage – eg. Maximum step sizes, maximum output signal ranges, source file names, destination file names, etc.

General_params:

GOSUB Clear_disp

PRINT TAB(22);CHR$(132);"Speech file information : ";CHR$(128);CHR$(10)
PRINT TAB(20);"Source file name : ";CHR$(129);In_file$;CHR$(128);CHR$(10)
PRINT TAB(20);"Destination file name : ";Out_file$;CHR$(10)
PRINT TAB(20);"Maximum signal value : ";Max_sig;CHR$(10)
PRINT TAB(20);"Minimum signal value : ";Min_sig;CHR$(10)
PRINT TAB(20);"Maximum step size : ";Max_step;CHR$(10)
PRINT TAB(20);"Sampling rate (kHz) : ";Rate;CHR$(10)
PRINT TAB(10);CHR$(129);"The default setting is selected by pressing 'ENT': CHR$(128)

INPUT "Enter the source file name", In_file$
In_file$=UPCS(In_file$
PRINT TABXY(20,4);"Source file name : ";In_file$;CHR$(10)
PRINT TAB(20);"Destination file name : ";CHR$(129);Out_file$;CHR$(128)
PRINT TABXY(20,5);"Destination file name : ";Out_file$;CHR$(10)
PRINT TAB(20);"Maximum signal value : ";CHR$(129);Max_sig;CHR$(128)

Value=0
INPUT "Enter the desired value", Value
IF Value>0 THEN
Max_sig=Value
Value=0
END IF

PRINT TABXY(20,8);"Maximum signal value : ";Max_sig;CHR$(10)
PRINT TAB(20);"Minimum signal value : ";CHR$(129);Min_sig;CHR$(128)
INPUT "Enter the desired value", Value
IF Value<0 THEN
Min_sig=Value
Value=0
END IF

PRINT TABXY(20,10);"Minimum signal value : ";Min_sig;CHR$(10)
PRINT TAB(20);"Maximum step size : ";CHR$(129);Max_step;CHR$(128)
INPUT "Enter the desired value", Value
IF Value>0 THEN
Max_step=Value
Value=0
END IF

PRINT TABXY(20,12);"Maximum step size : ";Max_step;CHR$(10)
7250 Rate_blocks=Rate_blocks*Rate/8 ! Convert from a time value to
7260 Syll_start=INT(Rate*Syll_start) ! a number of samples (i.e. the
7270 Syll_fin=INT(Rate*Syll_fin) ! no. of samples in the syllabic
7280 Syll_inc=INT(Rate*Syll_inc) ! period).
7290
7300 Samples_per_seg=Rate*20 ! The no. of samples in 20 msec.
7310 RETURN
7320 !*********************************************************************
7330 !
7340 Keyboard_input: !
7350 ON KBD GOTO Get_response
7360 !
7370 Waiting: !
7380 GOTO Waiting
7390 !
7400 Get_response: !
7410 ON ERROR GOTO Waiting
7420 Value=VAL(KBD$)
7430 OFF KBD
7440 OFF ERROR
7450 !
7460 RETURN
7470 !*********************************************************************
7480 END
BUFFERED D/A PROGRAM FOR SUBJECTIVE SIGNAL ANALYSIS

CODSIM : Codec Simulation Package.
Author : J.M. Irvine.
Department of Electrical and Electronic Engineering
University of Cape Town.
Unit Number : 5.

Programs developed for the Hewlett Packard 2000 Series desk-top computers. All development work done on an HP9836C computer with the interfacing between the analogue and digital environments done using the HP6942A Multiprogrammer.

This program reads the data stored on the discs and outputs it, via a memory buffer, to the D/A card in the Multiprogrammer. Once the data has been transferred from the mass storage device it is transferred to the memory cards in the multiprogrammer and then clocked out to the D/A card. The speech segment can be output once or repeatedly and it can be output in shorter segments which can form the whole or part of the original segment. There are two possible output data rates : 8 kHz for the original sampled data, and 16 kHz for the processed or interpolated versions of the sampled data.

DEFINE THE VARIABLES USED BY THE PROGRAM

OPTION BASE 1
INTEGER Card,Mistake,A_d_slot,D_a_slot ! Multiprogrammer card variables.
INTEGER Mem_in_slot,Tb_slot
INTEGER Complete,No_of_errors,No_of_cards,Slot_no
INTEGER Card_variable
REAL Intr_svc_delay,Transfer_oh,Total_delay ! System performance parameters.
REAL Transfer_rate,Reinit_time,Head_room
REAL Block_size,Reference_word,Time_available
REAL Transfer_time,Additional_rdgs,Trigger_rate

DIM Function$[40],Error$[40],Mw_command$[30],File_name$[10];
ALLOCATE INTEGER Configuration(0:16,5)

Run$=" R" ! Control character to start program execution.
Stop$=" !" ! Control character to stop program execution.
GOSUB Initialise ! Reset the mode of the controller.
GOSUB Sample_rate ! Set up the output data rate.
GOSUB Io_paths ! Set up data and control paths.
Xfer(Op$) ! Load BINARY subprogram.

! Determine the multiprogrammer card configuration.
Multi_set_up(Configuration(*),Tb_slot,Mem_in_slot,D_a_slot)
GOSUB Clear Disp
GOSUB Re_init ! Set the multiprogrammer cards in the
! correct operating state.

!THE SPEECH DATA IS READ OFF DISC AND OUTPUT

Output_speech:
GOSUB Read_data ! Transfer the speech data from disc.
New_sections:
GOSUB Sections ! Determine which portions to be listened to.
ON KBD GOTO Response ! Check for interrupt from keyboard.

DISP "To terminate the output hit any key on the keyboard."

Output_data ! Send the speech data to the multiprogrammer.

Response:
OFF KBD
DISP "Do you wish to output a new speech segment ? (Y/N)"
ON KBD GOTO Reply1

Wait:
GOTO Wait1

Reply1:
R$=UPC$(KBD$)
OFF KBD
IF R$="N" THEN
DISP "Do you wish to output a new speech segment ? ( Y/N )"
ON KBD GOTO Reply2
10 Wait2:!
20 GOTO Wait2

20 Reply2:!
30 R$=UPC$(KBD$)
40 OFF KBD
50 IF R$="N" THEN
60 )Function$="LOAD ""MASTER:INTERNAL,4.0"" X"
70 OUTPUT KBD;Stop$;Function$;Run$;
80 ELSE
90 GOTO Output_speech
100 END IF
120 ELSE
120 GOTO New_sections
130 END IF
140 !************************************************************************
SUBROUTINES

SPECIFY THE DEFAULT MASS STORAGE DEVICES

Initialise:

Drive_0$=":INTERNAL,4,0"
Drive_1$=":INTERNAL,4,1"
PRINTER IS CRT
CONTROL 1,12;1
No_of_buffers=14

! Set the default mass storage devices.
! Use the VDU for displays.
! Turn the key label display off.
! The number of arrays used to store
! the sampled speech data.

CLEAR THE ALPHANUMERIC AND GRAPHICS DISPLAYS

Clear disp:

GCLEAR
ALPHA OFF
PRINT USING "@"
PRINT CHR$(128):CHR$(140);;
ALPHA ON
DISP "" RETURN

Clear the graphics display.
Clear the Alpha display.
Select normal video with
Colour = Cyan.

DETERMINE THE OUTPUT DATA RATE

Sample_rate:

Op$="MW"
PRINT CHR$(129):TABXY(0,20);
PRINT "Possible output data rates are 8 kHz and 16 kHz."
BEEP 2500,.1
Wrong:INPUT "Enter the required data rate (kHz) >",Sample_rate
IF (Sample_rate<8) AND (Sample_rate>16) THEN
   Error$="INVALID ENTRY -- PLEASE RE-ENTER"
   GOTO Wrong
ELSE
   PRINT TABXY(0,20):"
END IF
CALCULATE MEMORY AND TIMING REQUIREMENTS

Bench marks:

```
Trigger_rate = 1/(Sample_rate*1.E+3) ! The sampling period.
Block_size = 10283 ! Size of the original data blocks.
Block_size = 8*2*Block_size/14 ! Size of sub-segments of speech.
Total_delay = 0.0229 + 0.0013
Head_room = Total_delay/Trigger_rate
Reference_word = INT(4085 - Head_room)
```

```
MEMORY SPACE IS NOW RESERVED
```

A total of 14 buffers, each representing approximately 0.8 seconds of speech, are used to store the sampled output speech signal.

```
ALLOCATE INTEGER Buf1(Blocksize), Buf2(Blocksize), Buf3(Blocksize)
ALLOCATE INTEGER, Buf4(Blocksize), Buf5(Blocksize), Buf6(Blocksize)
ALLOCATE INTEGER Buf7(Blocksize), Buf8(Blocksize), Buf9(Blocksize)
ALLOCATE INTEGER Buf10(Blocksize), Buf11(Blocksize), Buf12(Blocksize)
ALLOCATE INTEGER Buf13(Blocksize), Buf14(Blocksize)
```

```
Io_paths:
ASSIGN @Multi TO 823 ! Multiprogrammer address.
ASSIGN @Mw_readout TO 823;FORMAT OFF ! Buffer to Memory path.
ASSIGN @Intr_list TO 82312 ! Interrupt list.
ASSIGN @Intr_cause TO 82310 ! Interrupt cause.
```

```
RETURN
```
320 ! INITIATE THE SYSTEM CARDS >>>>>>>>>>>>>>>>>>>>>>>
330 !
340 Re_init:
350 OUTPUT @Multi;"CC";Tb_slot;"T"  ! Clear Timer/Pacer card. 
360 OUTPUT @Multi;"WC";Tb_slot;0;"T"  ! Turn off Timer/Pacer. 
370 OUTPUT @Multi;"WF";Tb_slot+2;1;"T"  ! Continuous mode timer. 
380 OUTPUT @Multi;"CC";D_a_slot;"T"  ! Clear the D/A card. 
390 OUTPUT @Multi;"SF";D_a_slot;"2,1,1T"  ! Program the D/A mode. 
400 OUTPUT @Multi;"CC";Mem_in_slot;"T"  ! Clear Memory card #1. 
410 OUTPUT @Multi;"CC";Mem_in_slot+1;"T"  ! Clear Memory card #2. 
420 OUTPUT @Multi;"WF";Mem_in_slot+1;0;"T"  ! Clear the REF REGISTER. 
430 OUTPUT @Multi;"WF";Mem_in_slot+1;21;"T"  ! Disable Memory card #1.
440 ! SET THE TIME BASE CARD'S CLOCK FREQUENCY 
450 OUTPUT @Multi;"WF";Tb_slot;((1.E+3/Sample_rate)/2);"UT" 
460 OUTPUT @Multi;"CY";Tb_slot;"T"  ! CYCLE THE TIME BASE CARD 
470 ! PROGRAM THE DIFFERENTIAL COUNTER OF THE MEMORY CARDS 
480 OUTPUT @Multi;"WF";Mem_in_slot+1;Reference_word;"T" 
490 ! INSTRUCT MEMORY ON SIZE OF THE DATA BLOCK 
500 OUTPUT Mw_command$;"MW";Mem_in_slot;Blocksize;"T" 
510 ! ENABLE THE BUFFERED DIGITAL TO ANALOGUE CONVERSION 
520 OUTPUT @Multi;"WF";Mem_in_slot+1;1;"T" 
530 ! ARM THE MEMORY CARD 
540 OUTPUT @Multi;"AC";Mem_in_slot+1;"T" 
550 RETURN
DETERMINE SECTION OF SPEECH TO BE OUTPUTTED

Sections:
Start=1
Finish=14
New_start=0
New_finish=0
Continuous$="Y"
New_continuous$="Y"

PRINT TABXY(3,5);CHR$(132);"Speech segments to be listened to:";CHR$(128)
PRINT TABXY(5,7);"Start Finish Continuous"
PRINT TABXY(6,9);CHR$(129);Start;CHR$(128);TAB(24);Finish;TAB(42);
IF Continuous$="Y" THEN
  PRINT " YES ";CHR$(128)
ELSE
  PRINT " NO ";CHR$(128)
END IF

PRINT TABXY(6,9);"Enter the desired value or toggle value (ie. Y/N)."

INPUT New_start
IF (New_start>0) AND (New_start<15) THEN
  Start=New_start
END IF
PRINT TABXY(6,9);Start;TAB(22);CHR$(129);Finish;CHR$(128);
IF Continuous$="Y" THEN
  PRINT TAB(42);" YES ";CHR$(128)
ELSE
  PRINT TAB(42);" NO ";CHR$(128)
END IF

INPUT New_finish
IF (New_finish>=Start) AND (New_finish<15) THEN
  Finish=New_finish
END IF
PRINT TABXY(6,9);Start;TAB(22);Finish;TAB(40);CHR$(129);
IF Continuous$="Y" THEN
  PRINT " YES ";CHR$(128)
ELSE
  PRINT " NO ";CHR$(128)
END IF
INPUT New_continuous$
New_continuous$=UPC$(New_continuous$
IF (New_continuous$="Y") OR (New_continuous$="N") THEN
Continuous$=New_continuous$
END IF
IF Continuous$="Y" THEN
PRINT TABXY(6,9);Start;TAB(22);Finish;TAB(40);" YES 
END IF
IF Continuous$="N" THEN
PRINT TABXY(6,9);Start;TAB(22);Finish;TAB(40);" NO 
END IF
!
DISP "Do you wish to keep these values ?"
ON KBD GOTO Answer
Get_answer: !
Answer: !
R$=UPC$(KBD$)
OFF KBD
IF R$="N" THEN
GOTO Sections
END IF
PRINT TABXY(0,21);"
RETURN
!*************************************************************************
Read data:

Drive$=Drive_1$

ON ERROR GOTO Wrong_disc

Assign_device:

MASS STORAGE IS Drive$

Get_file:

GOSUB File_name

ASSIGN @Test_file TO File_name$

DISP "Reading speech data off disc."

OFF ERROR

ENTER @Test_file; Buf1(*)
ENTER @Test_file; Buf2(*)
ENTER @Test_file; Buf3(*)
ENTER @Test_file; Buf4(*)
ENTER @Test_file; Buf5(*)
ENTER @Test_file; Buf6(*)
ENTER @Test_file; Buf7(*)

The speech data has been divided into small sections, each about 0.8 sec in length. If the original version of the sampled speech is to be listened to then only 7 sections are needed, whereas if the processed version is needed then 14 sections are required.

IF File_type$="P" THEN

ENTER @Test_file; Buf8(*)
ENTER @Test_file; Buf9(*)
ENTER @Test_file; Buf10(*)
ENTER @Test_file; Buf11(*)
ENTER @Test_file; Buf12(*)
ENTER @Test_file; Buf13(*)
ENTER @Test_file; Buf14(*)

If the file is a processed version of the original sampled data (ie. the data has been processed by a DM) then the extra speech data must be read off the disc.
ELSE
MAT Buf8= Buf1
MAT Buf9= Buf2
MAT Buf10= Buf3
MAT Buf11= Buf4
MAT Buf12= Buf5
MAT Buf13= Buf6
MAT Buf14= Buf7
END IF

ASSIGN @Test_file TO *

MASS STORAGE IS Drive_0$
RETURN

! If the speech segment is the original sampled version then only half as much data is available (as compared with interpolated and processed files) and so it is repeated.
Determine the name of the disc file containing the data >>>>>>>>>

File_name:

INPUT "Enter name of file from which the data is to be read >", File_name$
File_name$=UPC$(File_name$)
DISP "Does this file contain the original or processed data?"
ON KBD GOTO File_type
Spin:
GOTO Spin
File_type:
R$=UPC$(KBD$)
OFF KBD
IF R$="P" THEN
File_name$=File_name$&"_P" ! Add the suffix indicating processed data.
File_type$="P"
ELSE
File_name$=File_name$&"_O" ! Add the suffix indicating original data.
File_type$="O"
END IF
RETURN
!
80 !<<<<<<<<<<<<<< SEND THE SPEECH DATA TO THE MULTIPROGRAMMER >>>>>>>>>>>>>>
80 !
90 Output_data:
10 DISP "Press ";CHR$(129);" CONTINUE ";CHR$(128);" to commence output."
20 PAUSE
30 !
40 DISP
50 Transfer_data:
60 FOR Buffer_no=Start TO Finish
70 OUTPUT @Multi;Mw_command$
80 WAIT .005
90 SELECT Buffer_no
100 CASE 1
110 OUTPUT @Mw_readout;Buf1(*)
120 CASE 2
130 OUTPUT @Mw_readout;Buf2(*)
140 CASE 3
150 OUTPUT @Mw_readout;Buf3(*)
160 CASE 4
170 OUTPUT @Mw_readout;Buf4(*)
180 CASE 5
190 OUTPUT @Mw_readout;Buf5(*)
200 CASE 6
210 OUTPUT @Mw_readout;Buf6(*)
220 CASE 7
230 OUTPUT @Mw_readout;Buf7(*)
240 CASE 8
250 OUTPUT @Mw_readout;Buf8(*)
260 CASE 9
270 OUTPUT @Mw_readout;Buf9(*)
280 CASE 10
290 OUTPUT @Mw_readout;Buf10(*)
300 CASE 11
310 OUTPUT @Mw_readout;Buf11(*)
320 CASE 12
330 OUTPUT @Mw_readout;Buf12(*)
340 CASE 13
350 OUTPUT @Mw_readout;Buf13(*)
360 CASE 14
370 OUTPUT @Mw_readout;Buf14(*)
380 END SELECT
390 NEXT Buffer_no
40 !
50 IF Continuous$="Y" THEN
60 GOTO Transfer_data
70 END IF
80 !
90 RETURN
50 !*************************************************************************
170 !<<<<<<<<<<< RESOLVE ERRORS ARISING FROM DISC OPERATIONS >>>>>>>>>>>>
80 !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
90 !
200 Wrong disc:!!
210 IF ERRN=72 THEN
220  BEEP 500,.1
230  PRINT "Hard disc is not present !"
240  PRINT "Place the appropriate floppy disc in the LEFT hand disc drive."
250  PRINT "Hit ";CHR$(129);" CONTINUE ";CHR$(128);" when ready."
260  Drive$=":INTERNAL,4,1"
270  PAUSE
280  GOTO Assign_device
290 ELSE
300  IF ERRN=56 THEN
310   BEEP 500,.1
320   DISP "Invalid file name !! Please re-enter >"
330   WAIT 2
340   GOTO Get_file
350 ELSE
360   DISP ERRN,ERRMS$
370   STOP
380  END IF
390 END IF
400 STOP
410 END
420 !*************************************************************************
430 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! BINARY LOADER SUB-PROGRAM >>>>>>>>>>>>>>>>>>>>>>>
440 !
450 !
460 SUB Xfer(Op$) ! Transfer the special instruction
470 CLEAR 823 ! from the supplemental instruction
480 WAIT 4
490 ENTER 82310;Dummy,Dummy,Dummy
500 ALLOCATE Ascii$[80]
510 !
520 Name$=Op$&"Z:INTERNAL,4,0"
530 ASSIGN @Disc_file TO Name$
540 ON END @Disc_file GOTO Eof
550 !
560 Rd_file;ENTER @Disc_file;Ascii$
570 OUTPUT 823;Ascii$
580 GOTO Rd_file
590 !
600 Eof:OFF END @Disc_file
610 ASSIGN @Disc_file TO *
620 DEALLOCATE Ascii$
630 PRINT TABXY(5,8);"<MR> SUPPLEMENTAL INSTRUCTION LOADED INTO MULTI"
640 ! CLEAR SELF TEST SRQ
650 ENTER 82310:Complete,No_of_errors,No_of_cards
660 WAIT 1
670 DISP " "
680 SUBEND
690 !******************************************************************************
700 !<)))))))))))))))))) DETERMINE MULTIPROGRAMMER CARD CONFIGURATION >>>>>>>>
710 !
720 !
730 SUB Multi_set_up(INTEGER Configuration(*),Tb_slot,Mem_in_slot,D_a_slot)
740 OPTION BASE 1
750 PRINT TABXY(5,9);"DETERMINING MULTI CONFIGURATION"
760 FOR Slot_no=15 TO 0 STEP -1 ! Read data format of each card.
770 OUTPUT 823;"RF";Slot_no;"T" ! Start at slot #15.
780 !
790 FOR Card_variable=1 TO 5 ! Five variables returned per
800 ! card - Card ID, Data type,
810 ! LSB, Size, Limit.
820 !
830 ENTER 82304 USING ";,K";Configuration(Slot_no,Card_variable)
840 NEXT Card_variable
850 SELECT Configuration(Slot_no,1)
860 CASE 1
870 Tb_slot=Slot_no
880 CASE 6
890 Mem_in_slot=Slot_no
900 CASE 48
910 D_a_slot=Slot_no
920 CASE ELSE
930 END SELECT
940 NEXT Slot_no
950 SUBEND
960 !******************************************************************************
IMPLEMENT

PROCEDURE Codecs;

CONST
Block_size = 10283;  { The number of samples per array. }

VAR
Leakage,        { The prediction filter constant. }
Input_signal    : REAL;  { The actual sampled input signal. }

PROCEDURE LDM;  { Linear Delta Modulation. }

VAR
I              : INTEGER;

BEGIN
FOR I := 1 TO Block_size DO
BEGIN
Signal := New_signal;
Input_signal := Voice[ I ];

{ Update the two-level quantizer binary output signal. }
{--------------------------------------------------------------------}

IF Input_signal >= Signal
THEN
  Bn := +1
ELSE
  Bn := -1;

{ Update the coder approximation signal - which is }
{ equivalent to the decoder output signal under ideal }
{ channel conditions. }
{--------------------------------------------------------------------}

New_Signal := Signal + So * Bn;
IF New_signal > Max_sig
THEN
  New_signal := Max_sig
ELSE
  IF New_signal < Min_sig
  THEN
    New_signal := Min_sig:
  Voice[ I ] := New_Signal;
END;

WRITE( '.' );

END;
{**************************************************************************}
PROCEDURE CVSD ;  { Continuously Variable Slope Delta Modulation. }  
{ The algorithm used here is based on the algorithms described in }  
{ references [18], [21], and [22]. For further details refer to }  
{ chapter 3 section 2.2. }  

CONST  
B = 0.99 ;  { Step size leakage constant. }  
H = 3 ;  { Gain of the prediction integrator. }  

VAR  
I,V : INTEGER ;  

BEGIN  
FOR I := 1 TO Block_size DO  
BEGIN  
Bn_2 := Bn_1 ;  
Bn_1 := Bn ;  
Step := New_step ;  
Signal := New_signal ;  
Input_signal := Voice[ I ] ;  

{ Update the two-level quantizer binary output signal. }  
{-----------------------------------------------}  

IF Input_signal >= Signal  
THEN  
Bn := +1 ;  
ELSE  
Bn := -1 ;  

{ Update the coder step size according to the syllabic  
{ companding algorithm. }  
{-----------------------------------------------}  

IF ( Bn_2 = Bn_1 ) AND ( Bn_1 = Bn )  
THEN  
V := 150 ;  
ELSE  
V := 0 ;  

New_step := B * Step + ( 1-B ) * ( V + So ) ;
IF New_step > Max_step
    THEN
        New_step := Max_step
    ELSE
        IF New_step < -Max_step
            THEN
                New_step := -Max_step
            ELSE
                IF (New_step > 0) AND (New_step < So)
                    THEN
                        New_step := So
                    ELSE
                        IF (New_step > -So) AND (New_step < 0)
                            THEN
                                New_step := -So;

{ Update the coder approximation signal - which is equivalent to the decoder output signal under ideal channel conditions. }  
New_Signal := Leakage * Signal + Bn * H * (1-Leakage) * New_step;

IF New_signal > Max_sig
    THEN
        New_signal := Max_sig
    ELSE
        IF New_signal < Min_sig
            THEN
                New_signal := Min_sig;

    Voice[I] := New_Signal;

END;

WRITE( ' ');

END;  
{*****************************************************************************}
PROCEDURE CFDM ;  { Constant Factor Delta Modulation. }

{ The algorithm used here corresponds to Jayant’s ADM (cf. reference (23)). For further details refer to chapter 2 section 2.3. }

CONST

P = 1.5 ;  { The constant factors used by Jayant. }

VAR

I : INTEGER :

BEGIN

FOR I := 1 TO Block_size DD

BEGIN

Bn_1 := Bn ;
Step := New_step ;
Signal := New_signal ;
Input_signal := Voice[ I ] ;

{ 'Update the two-level quantizer binary output signal. }
{---------------------------------------------------------}

IF Input_signal >= Signal
THEN

Bn := +1 ;
ELSE

Bn := -1 ;
{ Update the coder step size according to the Constant }
{ Factor companding algorithm. }
{---------------------------------------------------------}

IF Bn_1 = Bn
THEN

New_step := Step × P
ELSE

New_step := Step / P ;

IF New_step > Max_step
THEN

New_step := Max_step
ELSE

IF New_step < So
THEN

New_step := So ;
{ Update the coder approximation signal - which is }
{ equivalent to the decoder output signal under ideal }
{ channel conditions. }
{---------------------------------------------------------}

New_Signal := Leakage × Signal + Bn × New_step ;

IF New_signal > Max_sig
THEN

New_signal := Max_sig
ELSE

IF New_signal < Min_sig
THEN

New_signal := Min_sig ;
{ Check that the output of }
{ the decoder is within }
{ the permissible range of }
{ of the output signal. }

Voice[ I ] := New_Signal ;

END ;
WRITE( ' ' ) ;

END ;
PROCEDURE SVADM ;          { Song Voice Adaptive Delta Modulation. }  
VAR
  I         : INTEGER ; 
BEGIN
  FOR I := 1 TO Block_size DO
  BEGIN
    Bn_1 := Bn ;
    Step := New_step ;
    Signal := New_signal ;
    Input_signal := Voice[ I ] ;

    { Update the two-level quantizer binary output signal. }
    IF Input_signal >= Signal
    THEN
      Bn := +1
    ELSE
      Bn := -1 ;

    { Update the coder step size according to the Song Voice }
    { Adaptive algorithm. }
    New_step := ABS( Step ) * Bn + So * Bn_1 ;
    IF New_step > Max_step
    THEN
      New_step := Max_step
    ELSE
      IF New_step < -Max_step
      THEN
        New_step := -Max_step ;

    { Update the coder approximation signal - which is }
    { equivalent to the decoder output signal under ideal }
    { channel conditions. }
    New_Signal := Leakage * Signal + New_step ;
    IF New_signal > Max_sig
    THEN
      New_signal := Max_sig
    ELSE
      IF New_signal < Min_sig
      THEN
        New_signal := Min_signal ;
    END ;

    WRITE( '.' ) ;
  END ;
END ;
{********************************************************************}
PROCEDURE SHCDM ;  { Song Hybrid Companding Delta Modulation. }
{ The algorithm used here was developed as part of the research project. It is described in detail in chapter 3, section 2.6. }
{-----------------------------------------------------------}

VAR
Sum,
Last_step,
Shift_out : REAL ;
I : INTEGER ;
BEGIN
Sum := 0 ;
FOR I := 1 TO ( Syllable-1 ) DO
Sum := Sum + SQR( Syll_filter[ I+1 ] - Syll_filter[ I ] ) ;
END FOR
FOR I := 1 TO Block_size DO
BEGIN
Bn := Bn ;
Last_step := Step ;
Signal := New_signal ;
Input_signal := Voice[ I ] ;

{ Update the two-level quantizer binary output signal. }
{---------------------------------------------------------}
END IF
ELSE
Bn := -1 ;

{ Update the syllabic step size factor. }
{---------------------------------------------------------}
BEGIN
Shift_out := Syll_filter[ Pos ] ;
Syll_filter[ Pos ] := Signal ;
END IF
ELSE
Sum := Sum + SQR( Signal - Syll_filter[ Syllable ] )
END IF
END FOR
Pos := Pos + 1 ;
IF Pos > Syllable
THEN
Pos := 1 ;
ELSE
Pos := Pos + 1 ;
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
END IF
END FOR
END IF
Sum := Sum - SQR(Shift_out - Syll_filter[ Pos ]); 

IF Pos = Syllable 
THEN 
BEGIN 
Big_alpha := Alpha * SQRT( Sum/Syllable ); 
IF Big_alpha < So 
THEN 
Big_alpha := So ;
END ;

{ Update the coder instantaneous step factor according to } 
{ the Song Voice Adaptive algorithm. } 
{---------------------------------------------------------}

Step := ABS( Last_step ) * Bn + Bn_1 * So ;

{ Update the coder step size by combining the instantaneous and syllabic step factors. } 
{---------------------------------------------------------}

New_step := Big_alpha * Step ;

IF New_step > Max_step 
THEN 
New_step := Max_step 
ELSE 
IF New_step < -Max_step 
THEN 
New_step := -Max_step ;

{ Limit the step size to the specified range. }
{---------------------------------------------------------}

New_signal := Leakage * Signal + New_step ;

IF New_signal > Max_sig 
THEN 
New_signal := Max_sig 
ELSE 
IF New_signal < Min_sig 
THEN 
New_signal := Min_sig ;

{ Check that the output of the decoder is within the permissible range of the output signal. }

Voice[ I ] := New_signal ;

END ;

WRITE( '.', );

END ;
PROCEDURE MSHCDM ;  { Modified Song Hybrid Companding DM. }

{ The algorithm used here was developed as part of the research }  
{ project. It is descibed in detail in chapter 3, section 2.6. }  
{--------------------------------------------------------------}

VAR

    Sum, 
    Shift_out : REAL ; 
    I : INTEGER ;

BEGIN

    Sum := 0 ;
    FOR I := 1 TO ( Syllable-1 ) DO 
        { Update the syllabic energy. }
        Sum := Sum + SQR( Syll_filter[ I+1 ] - Syll_filter[ I ] ) ; 
    FOR I := 1 TO Block_size DO
        BEGIN
            Bn:= Bn ;
            Step := New_step ;
            Signal := New_signal ;
            Input_signal := Voice[ I ] ;

            { Update the two-level quantizer binary output signal. } 
            {----------------------------------------------------------}
            IF Input_signal >= Signal
                THEN
                    Bn := +1
                ELSE
                    Bn := -1 ;

            { Update the syllabic step size factor. } 
            {----------------------------------------------------------}
            Shift_out := Syll_filter[ Pos ] ;
            Syll_filter[ Pos ] := Signal ;
            IF Pos = 1
                THEN
                    Sum := Sum + SQR( Signal - Syll_filter[ Syllable ] )
                ELSE
                    Sum := Sum + SQR( Signal - Syll_filter[ Pos-1 ] ) ;
            Pos := Pos + 1 ;
            IF Pos > Syllable
                THEN
                    Pos := 1 ;
                Sum := Sum - SQR( Shift_out - Syll_filter[ Pos ] ) ;
IF Pos = Syllable THEN BEGIN
    Big_alpha := Alpha * SQRT( Sum/Syllable );
    IF Big_alpha < So THEN
        Big_alpha := So;
END;

{ The instantaneous companding - Song Voice Adaptive. In this application the syllabic factor is included at this stage. }

New_step := ABS( Step ) * Bn + Bn_1 * Big_alpha * So ;

IF New_step > Max_step THEN
    New_step := Max_step
ELSE
    IF New_step < -Max_step THEN
        New_step := -Max_step ;

{ Update the coder approximation signal - which is equivalent to the decoder output signal under ideal channel conditions. }

New_signal := Leakage * Signal + New_step :

IF New_signal > Max_sig THEN
    New_signal := Max_sig
ELSE
    IF New_signal < Min_sig THEN
        New_signal := Min_sig :

Voice[ I ] := New_signal ;

END ;
WRITE( '.' ) ;

END ;
PROCEDURE HCDM ; 
{ Hybrid Companding Delta Modulation. }

{ The algorithm used in this procedure is the one described by Un et al [17], [18]. The syllabic companding is combined with Constant Factor companding. For further details refer to chapter 3, section 2.4. }

VAR
Sum, Shift_out : REAL :
I : INTEGER :

BEGIN
Sum := 0.:
FOR I := 1 TO ( Syllable-1 ) DO 
{ Update the syllabic energy. }
Sum := Sum + SQR( Syll_filter[ I+1 ] - Syll_filter[ I ] ) ;

FOR I := 1 TO Block_size DO
BEGIN
Bn_2 := Bn_1 ;
Bn_1 := Bn ;
Step := New_step ;
Signal := New_signal ;
Input_signal := Voice[ I ] ;

{ Update the two-level quantizer binary output signal. }
{---------------------------------------------------------}
IF Input_signal >= Signal
THEN
Bn := +1
ELSE
Bn := -1 ;

{ Update the syllabic step size factor. }
{---------------------------------------------------------}
Shift_out := Syll_filter[ Pos ] ;
Syll_filter[ Pos ] := Signal :

IF Pos = 1
THEN
Sum := Sum + SQR( Signal - Syll_filter[ Syllable ] )
ELSE
Sum := Sum + SQR( Signal - Syll_filter[ Pos-1 ] ) ;

Pos := Pos + 1 :
IF Pos > Syllable
THEN
Pos := 1 ;
Sum := Sum - SQR(Shift_out - Syll_filter[ Pos ]) :

IF Pos = Syllable
    THEN
        BEGIN
            Big_alpha := Alpha * SQRT( Sum/Syllable ) ;
            IF Big_alpha < So
                THEN
                    Big_alpha := So ;
        END ;

    { The instantaneous companding - Constant Factor. }

    {---------------------------------------------------------}

    IF ( Bn = Bn_1 ) AND ( Bn_1 = Bn_2 )
        THEN
            Step := Step * 1.5
        ELSE
            IF Bn <> Bn_1
                THEN
                    Step := Step / 1.5 ;
        IF Step < 1
            THEN
                Step := 1 ;
        IF Step > Max_cf
            THEN
                Step := Max_cf ;

    { Combine syllabic & instantaneous companding factors. }

    New_step := Big_alpha * Step ;
    IF New_step > Max_step
        THEN
            New_step := Max_step ;

    { Update the coder approximation signal - which is }
    { equivalent to the decoder output signal under ideal }
    { channel conditions. }

    New_signal := Leakage * Signal + Bn * NEW_STEP ;

    IF New_signal > Max_sig
        THEN
            New_signal := Max_sig
        ELSE
            IF New_signal < Min_sig
                THEN
                    New_signal := Min_sig ;

    Voice[ I ] := New_signal ;

END ;

WRITE( ".");

END ;
BEGIN

IF Beta > 1000 THEN
Leakage := 1 ELSE
Leakage := EXP( -1 / ( Beta * Rate ) );

CASE P_no OF
1 : LDM ; ( Linear Delta Modulation. )
2 : CVSD ; ( Continuously Variable Slope Delta Modulation. )
3 : CFDM ; ( Constant Factor Delta Modulation. )
4 : SVADM ; ( Song Voice Adaptive Delta Modulation. )

{ In the case of the hybrid delta modulators the code perform- }
{ ing the syllabic companding has been repeated in each of the }
{ procedures simulating the hybrids. Although this is not the }
{ most efficient use of the code it does help to reduce the }
{ amount of processing time occupied by subroutine calls. }

5 : SHCDM ; ( Song Hybrid Companding Delta Modulation; )
6 : MSHCDM ; ( Modified Song Hybrid Companding DM. )
7 : HCDM ; ( Hybrid Companding Delta Modulation. )

END ;

END { Of Procedure Codecs. } ;
END { Of Module CODERS. }.

{********************************************************************}
MODULE NOISE

{--------------------------------------------------------------------}

{ CODSIM     : Codec Simulation Package.}
{ M.Sc Thesis : The use of Delta Modulation for Low Bit-Rate Digital Speech Coding.}
{ Author      : J.M. Irvine.}
{ Department of Electrical and Electronic Engineering University of Cape Town.}
{ Unit Number : 7.}
{ Latest Update : 1st December, 1984.}

{--------------------------------------------------------------------}

$STACKCHECK OFF$   { Disable stack overflow checks. }
$RANGE OFF$        { Disable array and CASE index range checks. }
$OVLFLCHECK OFF$   { BASIC & PASCAL allow different INTEGER ranges. }
$IOCHECK OFF$      { No error checks after calls on I/O routines. }
$DEBUG OFF$        { Disable array and CASE index range checks. }

$SEARCH 'HARb:CSUBDECL':$   { BASIC-PASCAL conversion code. }
IMPORT CSUBDECL ;   { Simplifies parameter passing to BASIC. }

EXPORT   { The EXPORT statement includes the TYPE and PROCEDURES to }
{ be accessed as compiled subroutines by the BASIC program. }

TYPE
SEGMENT_NOISE = ARRAY [ 1..2057 ] OF REAL ;
VOICE_BUFFER = ARRAY [ 1..10283 ] OF REAL ;

PROCEDURE Noises( VAR Present_sgn,
                    Sample_count,
                    Granular_flag,
                    Overload_flag,
                    Segment_count,
                    Samples_per_segment : BINTVALTYPE ;

VAR Total_noise,
    Segmental_noise : REAL ;
    S_Pointer : DIMENTRYPTR ;
VAR Segmental : SEGMENT_NOISE ;
VAR Voice : VOICE_BUFFER ;

IMPLEMENT

PROCEDURE Noises ;

CONST
Block_size = 10283 ;   { The number of samples per array. }

VAR
Input_signal : REAL ;   { The actual sampled input signal. }

I,
Last_sgn : INTEGER ;
PROCEDURE Granular_noise;
VAR J : INTEGER;
BEGIN
FOR J := 1 TO Block_size DO
BEGIN
  Last_sgn := Present_sgn;
  IF Voice[ J ] < 0
    THEN
      Present_sgn := -1
    ELSE
      Present_sgn := +1;
  IF Last_sgn = Present_sgn
    THEN { If two equal signs are detected }
      Voice[ J ] := 0; { then the noise is Overload noise. }
  END;
END;
{*****************************************************************************}

PROCEDURE Overload_noise;
VAR J : INTEGER;
BEGIN
FOR J := 1 TO Block_size DO
BEGIN
  Last_sgn := Present_sgn;
  IF Voice[ J ] < 0
    THEN
      Present_sgn := -1
    ELSE
      Present_sgn := +1;
  IF Last_sgn <> Present_sgn
    THEN { If two unequal signs are detected }
      Voice[ J ] := 0; { then the noise is Granular noise. }
  END;
END;
{*****************************************************************************}
BEGIN

{ This module calculates the total noise energy and the segmental noise energy contained in the decoded speech signal. The noise signal, either filtered or unfiltered, is passed to this module by the BASIC program. The segmental noise powers are calculated using 20 msec segments. If the Granular or Overload flags are set then the respective noise energies are calculated. If neither of these flags are set then the values calculated correspond to the lumped noise and segmental noise energies. }

IF Granular_flag = 1 THEN
    Granular_noise ; ( Extract the Granular noise signal. )
ELSE
    IF Overload_flag = 1 THEN
        Overload_noise ; ( Extract the Overload noise signal. )

For I := 1 TO Block_size DO
    BEGIN
        Total_noise := Total_noise + SQR( Voice[ I ] ) ;

        IF Sample_count = Samples_per_segment THEN
            BEGIN
                Sample_count := 0 ;
                Segmental[ Segment_count ] := Segmental_noise ;
                Segment_count := Segment_count + 1 ;
                Segmental_noise := 0 ;
            END ;

            Segmental_noise := Segmental_noise + SQR( Voice[ I ] ) ;
            Sample_count := Sample_count + 1 ;
        END ;

    END ;

END { Of Procedure Codecs. } ;
END { Of Module CODERS. } ;
{********************************************************************}
MODULE STRETCHER

{--------------------------------------------------------------------
/ { CODSIM Codec Simulation Package. } { M.Sc Thesis : The use of Delta Modulation for Low Bit-Rate } 
/ { Author : J.M. Irvine. } } } } } } } } } } } } } } } } } } } } } } } } } } } } } } } } 
/ { Department of Electrical and Electronic Engineering } 
/ { University of Cape Town. } 
/ { Unit Number : 8. } 
/ { Latest Update : 4th December, 1984. } 
/ {--------------------------------------------------------------------

$STACKCHECK OFF$ { Disable stack overflow checks. } 
$RANGE OFF$ { Disable array and CASE index range checks. } 
$OVLFLCHECK OFF$ { BASIC & PASCAL allow different INTEGER ranges. } 
$IOCHECK OFF$ { No error checks after calls on I/O routines. } 
$DEBUG OFF$ 

$SEARCH 'HARD:CSUBDECL'$; { BASIC-PASCAL conversion code. } 

IMPORT CSUBDECL ; { Simplifies parameter passing to BASIC. } 

EXPORT { The EXPORT statement includes the TYPE and PROCEDURES to } 
{ be accessed as compiled subroutines by the BASIC program. } 

TYPE 
SPEECH_BUFFER = ARRAY [ 1..10283 ] OF BINTVALTYPE : 
VOICE_BUFFER = ARRAY [ 1..8,1..10283 ] OF BINTVALTYPE ; 

PROCEDURE STRETCH( VAR Sample_rate : BINTVALTYPE ; 
S_pointer : DIMENTRYPTR ; 
VAR Speech : SPEECH_BUFFER ; 
V_pointer : DIMENTRYPTR ; 
VAR Voice : VOICE_BUFFER ) ; 

IMPLEMENT 

PROCEDURE STRETCH ; 

CONST 
Block_size = 10283 ; { The number of samples per array. } 

VAR 
I,J,K, 
Rate_factor : INTEGER ; 

{--------------------------------------------------------------------}
BEGIN

{ The effective sampling rate determines the sample spacing. If }
{ a sampling rate of 32 kHz is required then the 8 kHz sampled }
{ data must have two null samples inserted between every sample. }
{---------------------------------------------------------------}

J := 0 ; { The subscripts controlling the }
K := 1 ; { position of the output samples. }

Rate_factor := Sample_rate DIV 8 ;

FOR I := 1 TO Block_size DO
BEGIN

J := J + Rate_factor ; { Determine the position of }
IF J > Block_size { the next sample in the new }
THEN { speech segment. }

K := K + 1 ;
J := J - Block_size ;

END ;


END ;

WRITE( '.' ) ;

END ; { Of Procedure Stretch. }
END. { Of Module Stretcher. }
{*******************************************************************}
The digital filter simulated by this procedure is responsible for the removal of the out of band components of the noise produced by the delta modulation process. The purpose of this is to implement the digital equivalent of the low-pass filter at the output of the decoder so as to enable the correct determination of the signal-to-noise ratio of the DM codec. This filter is also used for the interpolation of the speech data to higher effective sampling rates. The filter is designed to operate at 16, 24, 32, 48 and 64 kHz sampling rates. A window length of 60 has been used.

```pascal
$STACKCHECK OFF$  { Disable stack overflow checks. }
$RANGE OFF$        { Disable array and CASE index range checks. }
$0VFLCHECK OFF$    { BASIC & PASCAL allow different INTEGER ranges. }
$IOCHECK OFF$      { No error checks after calls on I/O routines. }
$DEBUG OFF$        { Disable stack overflow checks. }

$SEARCH 'HARD:CSUBDECL'$;  { BASIC-PASCAL conversion code. }

IMPORT CSUBDECL ;    { Simplifies parameter passing to BASIC. }

EXPORT { The EXPORT statement includes the TYPE and PROCEDURES to be accessed as compiled subroutines by the BASIC program. }

TYPE
  WINDOW_BUFFER = ARRAY [1..121] OF REAL;
  VOICE_BUFFER = ARRAY [1..10283] OF REAL;

PROCEDURE FILTER( VAR PI Position, Last_flag, Sample_rate : BINTVALTYPE;
  W_pointer : DIMENTRYPTR;
  VAR Window : WINDOW_BUFFER;
  V_pointer : DIMENTRYPTR;
  VAR Voice : VOICE_BUFFER ) :
```
IMPLEMENT

PROCEDURE FILTER ;

CONST
Window_length = '60 ; { The no. of elements in the filter. }
Block_size = 10283 ; { The number of samples per array. }
F_interval = 250 ; { Frequency spacing for the filter. }

VAR
D_filter,
Filter_in : ARRAY[ -Window_length..+Window_length ]
OF REAL ;

Filter_out : REAL ;

J,K,
Input_position : INTEGER ;

PROCEDURE Initialize_filter ;

VAR
Theta,
Hamming : REAL ;

I,J,
Freq_samples : INTEGER ;

BEGIN
Freq_samples := ( Sample_rate * 1000 ) DIV ( F_interval ) ;
FOR I := -Window_length to +Window_length DO
  Filter_in[ I ] := Window[ I+61 ] ;
FOR I := 0 TO Window_length DO
  BEGIN
    \{ Determine the impulse response weighting sequence. \}
    \{---------------------------------------------------------\}
    D_filter[ I ] := 1 + COS( 20*PI*I / Freq_samples ) ;
    FOR J := 1 TO 9 DO \{ The number of frequency samples in \}
       \{ the signal pass band. \}
      BEGIN
        \{ A Hamming window has been used to reduce the 'Gibson' \}
        \{ effect - ie. the overshoot caused by the finite length \}
        \{ of the window of the filter. \}
        \{---------------------------------------------------------\}
        Theta := ( 2*PI*I*J ) / ( Freq_samples ) ;
        D_filter[ I ] := D_filter[ I ] + 2 * COS( Theta ) ;
      END ;
  \{********************************************************************\}
END ;

END :
BEGIN

Initialize_filter;
FOR J := 1 TO Block_size DO
BEGIN
    Filter_in[ Position ] := Voice[ J ];
    Filter_out := 0;
    Inp_Position := Position;
    FOR K := - Window_length TO + Window_length DO
    BEGIN
        Inp_Position := Inp_Position + 1;
        IF Inp_Position > Window_length THEN
            Inp_Position := - Window_length;
        Filter_out := Filter_out + D_filter[ K ] * Filter_in[ Inp_Position ];
    END;
    Voice[ J ] := Filter_out;
    Position := Position + 1;
    IF Position > Window_length THEN
        Position := - Window_length;
END;
WRITE( '.', );

{ Update the Filter Window buffer before returning to the BASIC parent-program. }
{-----------------------------------------------}
FOR J := - Window_length to + Window_length DO
    Window[ J+1 ] := Filter_in[ J ];
END; { Of Procedure Filter. }
END. { Of Module FIR. }
{*******************************************************************************}
APPENDIX F

The digital filter described here was used for the interpolation function as well as for the objective quality measures. A finite impulse response (FIR) filter was used because of its inherent stability and the fact that the phase and amplitude characteristics may be arbitrarily specified.

Having specified the amplitude and phase characteristics the impulse response weighting sequence can be determined. If the amplitude response function is sampled at $1/NT$ Hz intervals then the frequency samples ($G_r$) are related to the impulse response $g(i)T$ by the discrete Fourier Transform (DFT) given by:

$$G_r = \sum_{i=0}^{N-1} g(i)T \cdot e^{-j\frac{2\pi r i}{N}}$$  \hfill (F1)

Hence, using the inverse DFT the impulse response weighting sequence can be determined.

$$g(i)T = \left( \sum_{r=0}^{N-1} G_r \cdot e^{j\frac{2\pi r i}{N}} \right) / N$$  \hfill (F2)

where

$$i = 0, 1, 2, \ldots, N-1.$$

However, by truncating the Fourier series to $N$ values gives rise to oscillations in the actual amplitude response of the filter. These oscillations, known as the Gibbs phenomenon, can be reduced by applying a window function to the finite series. The three most commonly used window functions for non-recursive digital
filters are the Hamming, Hanning and Blackman windows $^{(4)}$.

(a) Hamming window: $w_r = 0.54 + 0.46 \cos(\pi r/I)$.

(b) Hanning window: $w_r = 0.50 + 0.50 \cos(\pi r/I)$.

(c) Blackman window: $w_r = 0.42 + 0.50 \cos(\pi r/I)$

$$+ 0.08 \cos(2\pi r/I).$$

where

'I' is the number of terms in the impulse response weighting sequence on either side of $g(0)T$.

The filter used in the simulation package had the characteristics specified in fig. F1.

![Diagram](image)

**Fig. F1:** (a) Amplitude and (b) Phase characteristics of the FIR filter used in the simulation package.

The filter will first be designed for a sampling frequency, $f_s$, of 16 kHz and a cut-off frequency, $f_c$, of 2.5 kHz. If the frequency
APPENDIX F

samples are separated by 250 Hz then

\[ N = 16.10^3 / 250 = 64. \]

If \( \{ G_0, G_1, G_2, \ldots, G_n \} \) is represented by \( \{ G \} \), and noting that the value of \( G \) at a point of discontinuity is given by the average of \( G_{r-1} \) and \( G_{r+1} \), then, from fig. F1(a),

\[ \{ G \} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, \ldots, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1). \]

Representing \( e^{-j2\pi/N} \) as \( W \), from equation (F2)

\[ g(i)T = \left( \sum_{i=0}^{N-1} W G_i \right) / N \]

Knowing \( \{ G \} \), and from equation (F3):

\[ g(i)T = \left[ 1 + W^{-1} + W^{-2} + W^{-3} + \ldots + W^{-9} + \frac{1}{4} W^{-10} + \frac{1}{4} W^{-14} + W^{-15} + \ldots + W^{-8} + W^{-9} \right] / 64 \]

NOTE: \( W^{-31} = W^{-(64-31)} \)

\[ = e^{j2\pi(64-31)/64} \]

\[ = e^{j2\pi 1} e^{-j2\pi 1/64} \]

\[ = 1 e^{-j2\pi 1/64} \]

\[ \therefore W^{-31} = W^1. \]

\[ \therefore W^{-1} + W^{-31} = 2 \text{ Re}(W^1) = 2 \cos(2\pi 1/64) = 2 \cos(\varphi) \]

Therefore,

\[ g(i)T = \left[ 1 + 2 \cos(\varphi) + \ldots + \cos(10\varphi) \right] / 64 \]

The values for the impulse response weighting filter are given
Table F1 : Impulse response weighting sequence values - g(i)T.

<table>
<thead>
<tr>
<th>i</th>
<th>g(i)T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.3125</td>
</tr>
<tr>
<td>1</td>
<td>0.2645</td>
</tr>
<tr>
<td>2</td>
<td>0.1466</td>
</tr>
<tr>
<td>3</td>
<td>0.0205</td>
</tr>
<tr>
<td>4</td>
<td>-0.0555</td>
</tr>
<tr>
<td>5</td>
<td>-0.0612</td>
</tr>
<tr>
<td>6</td>
<td>-0.0917</td>
</tr>
<tr>
<td>7</td>
<td>0.0243</td>
</tr>
<tr>
<td>8</td>
<td>0.0377</td>
</tr>
<tr>
<td>9</td>
<td>0.0184</td>
</tr>
<tr>
<td>10</td>
<td>-0.0172</td>
</tr>
<tr>
<td>11</td>
<td>-0.0256</td>
</tr>
<tr>
<td>12</td>
<td>-0.0165</td>
</tr>
<tr>
<td>13</td>
<td>0.0041</td>
</tr>
<tr>
<td>14</td>
<td>0.0176</td>
</tr>
<tr>
<td>15</td>
<td>0.0143</td>
</tr>
<tr>
<td>16</td>
<td>0.0000</td>
</tr>
<tr>
<td>17</td>
<td>-0.0118</td>
</tr>
<tr>
<td>18</td>
<td>-0.0118</td>
</tr>
<tr>
<td>19</td>
<td>-0.0023</td>
</tr>
<tr>
<td>20</td>
<td>0.0074</td>
</tr>
</tbody>
</table>

In this case 20 terms have been used either side of g(0)T and hence the three window functions become:

(a) Hamming : \( W_r = 0.54 + 0.46 \cos \left( \frac{\pi t}{20} \right) \)

(b) Hanning : \( W_r = 0.50 + 0.50 \cos \left( \frac{\pi t}{20} \right) \)

(c) Blackman : \( W_r = 0.42 + 0.50 \cos \left( \frac{\pi t}{20} \right) + 0.08 \cos \left( \frac{\pi t}{10} \right) \).

The values for the impulse response weighting sequence are then modified to the values shown in Table F2 (overleaf).
The filter's pulse transfer function is then given by:

\[
G(z) = g(20)T \cdot z^{-20} + \ldots + g(1)T \cdot z^{-1} + g(0)T \cdot z^{0} + g(1)T \cdot z^{1} + \ldots + g(0)T \cdot z^{20}
\]  

where

\[z = e^{i\omega T}\]  

The amplitude vs frequency and the phase vs frequency plots are given in fig's. F2 and F3. Fig. F3 also shows the effect of changing the parameter 'I' and the sampling frequency (f_s). Fig. F4 gives an indication of the relative effects of using the
different window functions.

It should be noted that the filter design specified by equation (F4) is not a real-time implementation. In order to make the filter realisable in real time there should be no terms of $Z$ raised to a positive power. However, because this filter was used in the simulation package (which itself is not a real-time system) purely for interpolation and measurement purposes it was preferable to use the non-real time version as in this manner it is possible to achieve a zero phase shift characteristic.

**Finite Impulse Response Filter**

![Finite Impulse Response Filter](image)

Fig. F2: (a) Amplitude and (b) Phase response of the FIR filter specified by equation F4.
APPENDIX F

Finite Impulse Response Filter

Fig. F3: Variation of the amplitude (a) and phase (b) characteristics for different values of 'I'.

Sampling frequency = 16000
Filter cut-off frequency = 2500
Type of window = HAMMING
Fig. F4: Effect of the different window functions on the (a) amplitude and (b) phase characteristics of the filter.
This appendix contains the table which gives the critical values of Z for various levels of significance. This table is used in conjunction with the Z-score test to test the significance of results from the subjective tests.
Table G.1.

<table>
<thead>
<tr>
<th>Level of Significance</th>
<th>0.10</th>
<th>0.05</th>
<th>0.01</th>
<th>0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical values of Z for</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One-Tailed Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.28</td>
<td>-1.645</td>
<td>-2.33</td>
<td>-2.58</td>
</tr>
<tr>
<td></td>
<td>1.28</td>
<td>1.645</td>
<td>2.33</td>
<td>2.58</td>
</tr>
<tr>
<td>Critical values of Z for</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two-Tailed Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.645</td>
<td>-1.96</td>
<td>-2.58</td>
<td>-3.08</td>
</tr>
<tr>
<td></td>
<td>1.645</td>
<td>1.96</td>
<td>2.58</td>
<td>3.08</td>
</tr>
</tbody>
</table>

One-tailed, or one-sided, tests are used when testing values to one side of the mean. The procedure for testing a hypothesis that one system (e.g., one codec) is better than another involves the use of one-tailed tests. This is different to testing whether one system is better or worse than another.