NEW APPROACH TO SPEECH DIGITIZATION COMBINING TIME-DOMAIN HARMONIC-ETC
AUG 81  J L WELS, A K PANDE

UNCLASSIFIED

DCA100-80-C-0050
FINAL REPORT
NEW APPROACH TO SPEECH DIGITIZATION
COMBINING
TIME DOMAIN HARMONIC SCALING
AND
ADAPTIVE RESIDUAL CODING

Prepared for

Defense Communications Agency
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia 22090

Contract No. DCA 100-80-C-0050

31 August 1981
This report describes the results of a twelve-month study, supported under DCA Contract 100-80-C-0050, of a new speech digitization algorithm combining Time-Domain Harmonic Scaling (TDHS) and Adaptive Residual Coding (ARC). The FORTRAN simulation of this system conducted as part of this study produces high quality speech reproduction at mediumband bit rates of 9.6 kb/s and 16 kb/s. This system also displays excellent robustness characteristics for channel bit error rates as high as 1% and for acoustic background noise. By basing the (over)
required pitch extraction on a three-level clipped signal, the hardware requirements for the system are kept modest.

The combined algorithm has several features which become significant in a full system application. Because the algorithm is a high performance, waveform matching algorithm, extremely good performance in the random configuration with other algorithms is anticipated. Since the technique is basically a waveform reconstruction technique, it will perform well on non-speech signals such as in-band signaling and modem tones.
ABSTRACT

This report describes the results of a twelve-month study, supported under DCA Contract 100-80-C-0050, of a new speech digitization algorithm combining Time-Domain Harmonic Scaling (TDHS) and Adaptive Residual Coding (ARC). The FORTRAN simulation of this system conducted as part of this study produces high quality speech reproduction at medium band bit rates of 9.6 kb/s and 16 kb/s. This system also displays excellent robustness characteristics for channel bit error rates as high as 1% and for acoustic background noise. By basing the required pitch extraction on a three-level clipped signal, the hardware requirements for the system are kept modest.

The combined algorithm has several features which become significant in a full system application. Because the algorithm is a high performance, waveform matching algorithm, extremely good performance in the tandem configuration with other algorithms is anticipated. Since the technique is basically a waveform reconstruction technique, it will perform well on non speech signals such as in-band signaling and modem tones.
PROJECT PERSONNEL

James L. Melsa, principal investigator
Arun K. Pande, research assistant
I

FINAL REPORT
DCA Contract 100-80-C-0050

Abstract

Project Personnel

CHAPTER 1: INTRODUCTION AND OUTLINE OF REPORT

1.1 Introduction
1.2 Algorithm Objectives
1.3 Outline of the Report

CHAPTER 2: TIME DOMAIN HARMONIC SCALING

2.1 Introduction
2.2 Frequency Compression and Expansion
2.3 Sampling Rate
2.4 Window Design
2.5 Compression Ratio
2.6 Pitch Extraction
2.7 Summary

CHAPTER 3: ADAPTIVE RESIDUAL CODING

3.1 Introduction
3.2 The Residual Encoder Structure
3.3 Performance
3.4 Summary

CHAPTER 4: TIME DOMAIN HARMONIC SCALING AND
ADAPTIVE RESIDUAL CODING SYSTEM

4.1 Introduction
4.2 System Configuration
4.3 Simulation
4.4 Coding
4.5 Buffer Control
4.6 Transmission Errors
4.7 Background Noise
4.8 Summary

iii
<table>
<thead>
<tr>
<th>CHAPTER 5: THE 16 KB/S SYSTEM</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction</td>
<td>100</td>
</tr>
<tr>
<td>5.2 Source and Channel Coding</td>
<td>100</td>
</tr>
<tr>
<td>5.3 Buffer Behaviour</td>
<td>103</td>
</tr>
<tr>
<td>5.4 Summary</td>
<td>105</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 6: SUMMARY</th>
<th>110</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>APPENDIX A: FOURIER TRANSFORM OF WINDOW FUNCTIONS</th>
<th>112</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPENDIX B: FLOW CHARTS</td>
<td>115</td>
</tr>
<tr>
<td>APPENDIX C: SOURCE LISTINGS</td>
<td>120</td>
</tr>
</tbody>
</table>

REFERENCES 160
CHAPTER 1

INTRODUCTION AND OUTLINE OF REPORT

1.1 INTRODUCTION

This report describes the results of a twelve-month study supported under DCA Contract 100-80-C-0050 of a new speech digitization algorithm combining Time-Domain Harmonic Scaling (TDHS) and Adaptive Residual Coding (ARC). The FORTRAN simulation of this system conducted as part of this study produces high quality speech reproduction at medium band bit rates of 9.6 kb/s and 16 kb/s. This system also displays excellent robustness characteristics for channel bit error rates as high as 1% and for acoustic background noise. By basing the required pitch extraction on a three-level clipped signal, the hardware requirements for the system are modest.

The TDHS algorithm was developed by Malah (1979) and applied by him (Malah, 1980) to a CVSD system at a transmission rate of 7.2 kb/s. More recently Malah (1981) has combined TDHS with Transform Coding and Sub-band Coding at medium band bit rates. This algorithm consists of properly weighting several adjacent input signal segments of pitch dependent duration by suitable window functions. As a result of this, the number of samples to be transmitted can be reduced by a factor of two. If the bit rate is kept the same, the number of bits allowed per sample is doubled, and the performance of the coder can be improved significantly. The ARC structure was developed by Cohn and Melsa (1975b) and implemented in hardware by CODEX (Qureshi and Forney, 1975). This structure involves the combination of pitch compensating adaptive quantizer (Cohn and Melsa,
1976), sequentially adaptive linear predictor, and adaptive source coding.

The combined algorithm has several features which become significant in a full system application. Because the algorithm is a high performance, waveform matching algorithm, extremely good performance in the tandem configuration with other algorithms is anticipated. Since the technique is basically a waveform reconstruction technique, it will perform well on non-speech signals such as in-band signaling and modem tones.
1.2 ALGORITHM OBJECTIVE

The following objective for the speech coding algorithm have been established from the Statement of Work:

1. The speech processing system shall operate at a medium band transmission data rates of 9.6 kb/s and 16 kb/s.
2. The speech processing system shall produce toll quality speech reproduction.
3. The audio bandwidth of the input speech shall be greater than or equal to 3200 Hz.
4. The speech coder shall produce good quality speech under conditions of a random transmission bit error rate of 1 percent.
5. The speech coder shall produce toll quality speech under 60 dB (reference to 20 μ newtons/meter$^2$) of acoustic background noise such as office noise and good quality speech under 100 dB of acoustic background noise.
6. The computational complexity of the algorithm shall be minimized.
7. The system shall be capable of processing non-speech signals, such as in-band signaling and modem tones.
8. Tandem connection with other algorithms such as CVSD and LPC-10 should cause negligible distortion.
1.3 OUTLINE OF THE REPORT

The design and development of TDHS-ARC algorithm is described in three chapters. Chapter 2 contains the research work pertaining to Time Domain Harmonic Scaling. The recent development of TDHS is described to provide the necessary background material. The research problems such as sampling rate, window design, compression ratio and pitch extraction are addressed in this chapter. The design of an Adaptive Residual Coder for the frequency compressed speech signal is outlined in Chapter 3. Various objective performance measure criteria and a new fixed wordlength source code are also described in this chapter. The complete system structure is presented in Chapter 4. The effect of transmission errors and background noise on the system performance is given. The strategy to control the buffer behaviour and the scheme of extracting pitch at the receiver are also discussed in this chapter. Chapter 5 describes the modifications of the algorithm which are needed to operate at 16 kb/s.

The Fourier Transform of several TDHS window functions are given in Appendix A. The flow charts for the FORTRAN simulation are given in Appendix B while the source listings are given in Appendix C.
CHAPTER 2
TIME DOMAIN HARMONIC SCALING

2.1 INTRODUCTION

In Chapter 1, it was indicated that the system uses time domain harmonic scaling (TDHS) to reduce the number of speech samples to be transmitted without causing excessive distortion. This process also allows an increased number of bits per sample to be available for coding. The TDHS algorithm uses a pitch adaptive window to perform frequency compression or expansion on the speech signal. Such frequency scaling operations are dependent on various factors such as the type of the window and the pitch extraction method used and the amount of frequency compression employed. In this chapter, these and other factors affecting TDHS performance are discussed and results are presented.

2.2 FREQUENCY COMPRESSION AND EXPANSION

The time varying Fourier representation of speech has successfully been used in vocoders. The techniques used for frequency scaling in these vocoders are fairly complex and, therefore, have not been extended to mediumband speech coders. Recently, a time-domain algorithm for frequency scaling was developed by Malah [April, 1979] and applied by him [April, 1980] to CVSD system at a transmission rate of 7200 bits/second.

The algorithm is quite general and involves choices of such parameters as windowing function and scaling factors. One specific, and most common, form of the algorithm is presented below using triangular
windows and 2 to 1 scaling. The TDHS algorithm makes use of the long-term pitch redundancy of speech signals in a manner that is similar to gapped analysis [Melsa, et. al., 1980]. However, by a clever choice of the time-domain windowing function, the TDHS algorithm is able to ensure continuity across the frame boundaries. At the transmitter, the basic concept is to compress two pitch periods of speech into a single pitch period of the same time duration but at half the sampling rate. At the receiver, the compressed signal is frequency multiplied to reconstitute an approximation of the original input signal. Consider first the frequency compression operation.

Suppose that speech samples up to sample number \( k_0 \) have been processed; the corresponding output sample number is \( m_0 = k_0/2 \). The first step is to determine the pitch period associated with the samples following \( k_0 \) by any standard method such as correlation or AMDF. Let the resulting pitch period, in samples, be \( N_p \). The value of \( N_p \) during unvoiced speech or silence can be set arbitrarily. Consider the \( 2N_p \) samples from \( k_0 + 1 \) to \( k_0 + 2N_p \) as shown in Fig. 2.1. Note that these samples need not be pitch synchronous. These \( 2N_p \) samples are frequency compressed into \( N_p \) samples by use of the following TDHS algorithm where \( y(m) \) is the compressed output and \( s(k) \) is the original speech.

\[
y(m_0+i) = s(k_0+i) h(i:N_p) + s(k_0+N_p+i) [1 - h(i:N_p)]
\]

\[
i = 1, 2, ..., N_p
\]  

(2.1)

Equation (2.1) can be rewritten as
Fig. 2.1 TDHS frequency compression for the compression ratio of 2:1.

Fig. 2.2 TDHS frequency expansion for the expansion ratio of 1:2.
\[ y(m_0+i) = s(k_0+N_p+i) + h(i:N_p)[s(k_0+i) - s(k_0+N_p+i)] \]  (2.2)

to indicate that only one multiplication and two additions are required per output sample. The frequency compression operation is illustrated in Fig. 2.1.

As long as the window function \( h(i:N_p) \) satisfies the properties

\[
\begin{align*}
    h(1:N_p) &= 1 \\
    h(N_p:N_p) &= 0
\end{align*}
\]  (2.3)

the following continuity conditions will be satisfied

\[
\begin{align*}
    y(m_0) &= s(k_0) \\
    y(m_0+1) &= s(k_0+1)
\end{align*}
\]  (2.4)

and

\[
\begin{align*}
    y(m_0+N_p) &= s(k_0+2N_p) \\
    y(m_0+N_p+1) &= s(k_0+2N_p+1)
\end{align*}
\]  (2.5)

At the receiver, it is necessary to use a frequency multiplication procedure to regenerate the \( 2N_p \) samples from the \( N_p \) samples of \( y(m) \).

Using the TDHS algorithm this is accomplished as

\[ \hat{s}(k_0+i) = y(m_0+i) h(i:2N_p) + y(m_0-N_p+i)[1 - h(i:2N_p)] \\
    i = 1, 2, 3, \ldots, 2N_p \]  (2.6)

The frequency multiplication operation is illustrated in Fig. 2.2. Once again if the window function satisfies Eq. (2.3), continuity will be ensured across the frame boundary.

The frequency spectrum of the original speech, compressed speech and expanded speech is shown in Figs. 2.3 & 2.4. The plot is for 20 msec of voiced speech. It can be seen that frequency spectrum of original and expanded speech match very well.
2.3 SAMPLING RATE

Samples of analog signals are a unique representation if the analog signal is bandlimited and if the sampling rate is more than twice the Nyquist frequency. Speech signals are not inherently bandlimited, although the spectrum does fall off rapidly at high frequencies. For voiced sounds, the high frequencies are more than 40 db below the peak of the spectrum for frequencies above 4 kHz. On the other hand for unvoiced sounds, the spectrum does not fall off appreciably even above 8 kHz. However, telephone transmission has a bandlimiting effect on speech signals and the maximum frequency in speech signals can be considered as 3.2-3.5 kHz for conversational or "telephone quality" speech.

TDHS algorithms are based on the assumption that the fundamental frequency $F_o$ (the pitch) of the input voiced-speech signal is known. If estimated pitch frequency is $F_p$ then error in frequency estimate is $F_p - F_o$. This error in pitch estimation can be tolerated [Malah, 1979] if

$$\frac{|F_p - F_o|}{F_p} < \frac{1}{2L}$$

where $L = \text{number of harmonics present in bandlimited periodic input signal}$. It is obvious that accuracy in the determination of the pitch period is important. Since a pitch estimator extracts pitch in terms of integer number of samples, the accuracy of the pitch extracted depends on the sampling frequency of the input speech signal. As the sampling frequency is increased, the pitch period resolution is improved. However, by increasing the sampling rate or oversampling the speech signal, fewer bits per sample are available for coding the quantizer levels. For example, if the
sampling rate is 6400 samples/sec (3200 samples/sec for compressed speech) and the transmission rate is 9600 bits/sec, the average number of bits per samples is 3. For a sampling rate of 10000 samples/sec, the pitch period estimation becomes 36% more accurate while the average entropy allowed drops down from 3 bits to 1.92 bits/sample. Hence, there is a trade off between the improvement in performance due to increased pitch accuracy and the degradation due to the decrease in entropy.

To study this trade-off, input speech was sampled at 3 different sampling frequencies, namely: 6.4, 8 and 9.6 kHz. First only frequency compression and expansion operations were considered. Informal listening tests have shown that unvoiced (higher frequency) speech sounds much better for higher sampling rates than lower ones. However, overall speech quality does not differ significantly. When quantization was introduced (TDHS-ARC System), no significant change in quality was noticed for the different sampling rates. As indicated earlier, fewer bits per sample are available for the higher sampling rates. This results in more quantization noise which masks the improvement obtained in the unvoiced sound by higher sampling rates. The sampling frequency for the system with transmission rate of 9.6 kb/s was chosen to be 6400 Hz. With more bits per sample available for coding in 16 kb/s System, a sampling frequency of 8 kHz may be a good choice.

2.4 WINDOW DESIGN

To determine the proper window function to be used, the requirements and the constraints that should be satisfied by this function need to be discussed.
As mentioned earlier, the TDHS algorithm consists of properly weighing several adjacent input signal segments (with pitch dependent duration) by a suitable window function to produce an output segment. Since the pitch period $N_p$ varies, it is necessary that adjacent segments processed with different values of $N_p$ should maintain output signal continuity at the interface between segments. This could be written in equation form as

$$y(m_0) = s(k_0) \quad (2.7)$$

$$y(m_0+1) = s(k_0+1)$$

and

$$y(m_0+N_p) = s(k_0+2N_p) \quad (2.8)$$

$$y(m_0+N_p+1) = s(k_0+2N_p+1)$$

where $k_0$ is the sample number up to which speech samples are processed

$m_0 = k_0/2$ corresponding output sample number

From Eq. (2.1) it is known that

$$y(m_0+i) = s(k_0+i) h(i:N_p) + s(k_0+N_p+i)[1 - h(i:N_p)]$$

for $i = 1, 2, \ldots, N_p$

$$y(m_0+1) = s(k_0+1) h(1:N_p) + s(k_0+N_p+1)[1 - h(1:N_p)] \quad (2.9)$$

for $i = N_p$

$$y(m_0+N_p) = s(k_0+N_p) h(N_p:N_p) + s(k_0+2N_p)[1 - h(N_p:N_p)] \quad (2.10)$$

To satisfy the continuity conditions in Eqs. (2.7) and (2.8),

$$h(1:N_p) = 1$$

and

$$h(N_p:N_p) = 0$$

Another constraint is imposed by the fact that if this algorithm is used for periodic signals, then exact frequency scaling should be obtained. If the signal has period $N_p$ and $h(n:N_p)$ is the window function,
Fig. 2.5 Different window functions.
frequency division by two can be achieved as follows

\[ y(n) = s(n) h(n:N_p) + s(n+N_p) h(n+N_p:N_p) \]

\[ n = 1, 2, 3, \ldots, N_p \]

Since \( s(n) \) is periodic

\[ s(n) = s(n+N_p) \]

hence,

\[ y(n) = s(n)[h(n:N_p) + h(n+N_p:N_p)]. \]

For exact frequency division by two, it is therefore necessary that \( h(n:N_p) \) satisfy

\[ h(n:N_p) + h(n+N_p:N_p) = 1 \quad (2.11) \]

or

\[ h(n+N_p:N_p) = 1 - h(n:N_p) \]

There are various types of window functions possible which satisfy above constraints and hence could be used in TDHS algorithm. Some of the possible window functions are shown in Fig. 2.5.

The choice of window depends upon the simplicity of implementation, the number of computation required and the performance. The performance of a particular window is measured in terms of the quality of output speech produced with that window choice. The best method of measuring the quality of output speech is by listening to it. However, this criterion is subjective and besides, very time consuming to use. The other criteria which are frequently used for measuring the quality of speech are segmental signal-to-noise ratio (SEGSNR) in the time and frequency domain. The SEGSNR in the time domain is the average of SNRs calculated for all the segments of speech. The typical length of the segment is 20 msec. The SEGSNR in the frequency domain is calculated in a similar way except the SNRs are computed for the frequency components of speech samples. The frequency components are obtained by taking a DFT of the input and output speech segments.
It was found that SEGSNR in the frequency domain very closely reflects the quality of output speech and thus, could form a good objective measure for this system. The details are discussed in Chapter 3. Table 2.1 shows various window functions and their performance. The frequency response of these functions are outlined in Appendix A. It can be seen from Table 2.1 that the performance of the TDHS algorithm for different window functions and obtained for a two second male utterance is almost the same. However, the complexity of these windows vary considerably. For example, the triangular window is very simple to implement while its performance is slightly worse than Hanning window which requires more computations.

Figure 2.6 shows the TDHS output speech plot for the word, "CATS" for different types of window functions. For the trapezoidal and Tukey windows the energy fluctuations in transition regions are more accurately reconstructed than the rest. The trapezoidal window function could be an attractive alternative to triangular window function since further savings in multiplication operations could be achieved. This is demonstrated as follows. From Table 2.1, the trapezoidal window function is

\[
h(n:N_p) = \begin{cases} 
1 & 1 < n < N_p/2 \\
2 - 2n/N_p & N_p/2+1 < n < N_p 
\end{cases}
\]

and from Eq. (2.9), the frequency compression operation is given by

\[
y(n) = s(n+N_p) + h(n:N_p)\{s(n)-s(n+N_p)\}
\] for \( n = 1,2,\ldots,N_p \)

For trapezoidal window function, this reduces to

\[
y(n) = s(n) \quad 1 < n < N_p/2
\]

\[
y(n) = s(n+N_p) + h(n:N_p)\{s(n)-s(n+N_p)\} \quad N_p/2+1 < n < N_p
\]
<table>
<thead>
<tr>
<th>Type</th>
<th>$h(n;N_p)$</th>
<th>$#$ of Multiplications per pitch period</th>
<th>$#$ of Additions per pitch period</th>
<th>SEGSNR in frequency domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangular</td>
<td>$1 - \frac{n-1}{N_p-1}$</td>
<td>$N_p$</td>
<td>$1 + N_p$</td>
<td>11.39 dB</td>
</tr>
<tr>
<td>Cosine</td>
<td>$\cos\left(\frac{n(n-1)}{2(N_p-1)}\right)$</td>
<td>$2N_p$</td>
<td>$N_p + 1$</td>
<td>11.69 dB</td>
</tr>
<tr>
<td>Hanning</td>
<td>$\frac{1}{2} + \frac{1}{2} \cos\left(\frac{n(n-1)}{N_p-1}\right)$</td>
<td>$3N_p$</td>
<td>$2N_p + 1$</td>
<td>11.82 dB</td>
</tr>
<tr>
<td>Tukey</td>
<td>$\left{\begin{array}{ll} 1 &amp; 1 \leq n &lt; N_p / 2 \ \frac{1}{2} - \cos\left(\frac{2\pi n}{N_p}\right) &amp; N_p / 2 + 1 \leq n &lt; N_p \end{array}\right.$</td>
<td>$N_p$</td>
<td>$\frac{N_p}{2}$</td>
<td>11.24 dB</td>
</tr>
<tr>
<td>Popoulis</td>
<td>$\frac{1}{\pi} \sin\left(\frac{(n-1)n}{N_p-1}\right) + \left[1 - \frac{n-1}{N_p-1}\right] \cos\left(\frac{n(n-1)}{N_p-1}\right)$</td>
<td>$3N_p$</td>
<td>$3N_p$</td>
<td>11.55 dB</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>$\left{\begin{array}{ll} 1 &amp; 1 \leq n &lt; \frac{N_p}{2} \ 2 - \frac{2n}{N_p} &amp; \frac{N_p}{2} + 1 \leq n &lt; N_p \end{array}\right.$</td>
<td>$N_p$</td>
<td>$\frac{N_p}{2} + 1$</td>
<td>11.28 dB</td>
</tr>
<tr>
<td>Hamming</td>
<td>$0.54 + 0.46 \cos\left(\frac{n(n-1)}{N_p-1}\right)$</td>
<td>$3N_p$</td>
<td>$2N_p + 1$</td>
<td>11.75 dB</td>
</tr>
</tbody>
</table>
Original speech

Triangular window

Trapezoidal window

Papoulis window
Fig. 2.6 TDHS output for the word "CATS" for different window functions.
From the comparisons of the above equations, it can be seen that it requires \(\frac{N_p}{2}\) multiplications and \(N_p\) additions to produce \(N_p\) compressed speech samples for trapezoidal window as compared to \(N_p\) multiplications and \(2N_p\) additions for triangular window.

The quality of output speech generated by using either the triangular or the trapezoidal window is almost the same. Therefore, the choice depends mainly upon the simplicity of implementation in hardware.

2.5 COMPRESSION RATIO

In previous sections, a compression factor of 2 was considered. However, other compression ratios are possible. As the value of this ratio increases, more interharmonic aliasing of pitch harmonics results. Such distortion could be tolerated in certain applications. In speech communication, speech quality is important and therefore, spectral distortions need to be kept small. A compression ratio of 2:1 is acceptable in this study. However, the distortion caused by compression and expansion process can be reduced by employing 3:2 compression.

Fig. 2.7 shows how three pitch periods can be compressed into two and expanded back into three. These operations can be put into equation form as follows. The frequency compressed speech, \(y(n)\) is given by

\[
y(n) = s(n) h(n;2N_p) + s(n+N_p)[1-h(n;2N_p)]
\]

for \(n = 1, 2, ..., 2N_p\) (2.12)

or

\[
y(n) = s(n+N_p) + h(n;2N_p)[s(n)-s(n+N_p)]
\]

\(n = 1, 2, ..., 2N_p\) (2.13)

where \(s(n)\) is the original speech samples, \(N_p\) is the pitch period expressed in terms of number of samples and \(h(n;2N_p)\) is the window function given by
Fig. 2.7(a) TDHS frequency compression for the compression ratio of 3:2.
The frequency expansion operation, with the ratio of 2:3, on speech is performed similar to 1:2 as discussed earlier except for different window function and is given by

\[ \hat{s}(n) = y(n)[1-h(n:3Np)] + y(n+Np)h(n:3Np) \]

or

\[ \hat{s}(n) = y(n) + h(n:3Np)[y(n+Np)-y(n)] \]

\[ n = 1, 2, \ldots, 3Np \]  

where

\[ h(n:3Np) = 1 - \frac{(n-1)}{3Np-1} \quad 1 < n < 3Np \]  

Frequency scaling operations discussed above were simulated. The distortion found in the output speech was less than with the 2:1 compression scheme, as anticipated. This is evident from the results listed in Table 2.2.

**TABLE 2.2**

Comparison of 2:1 and 3:2 compression ratio scheme.

Sentence 1, Male speaker, Block size = 80 samples searching range: 20<T<100

<table>
<thead>
<tr>
<th>Type of Window</th>
<th>SEGSNR (Time domain)</th>
<th>SEGSNR (Frequency domain)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compression Ratio 2:1</td>
<td>Compression Ratio 3:2</td>
</tr>
<tr>
<td>Triangular</td>
<td>8.06 dB</td>
<td>9.34 dB</td>
</tr>
<tr>
<td>Hanning</td>
<td>8.42 dB</td>
<td>10.31 dB</td>
</tr>
<tr>
<td>Tukey</td>
<td>7.30 dB</td>
<td>9.50 dB</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>7.48 dB</td>
<td>9.43 dB</td>
</tr>
</tbody>
</table>
The increase, in segmental SNR in frequency domain, as high as 5 dB could be obtained by employing 3:2 compression scheme. The increase in SEGSNR usually indicates the improvement in speech quality. However, correlation between the extent of such improvement and the increase in SEGSNR is not known and is a separate topic of research [Barnwell, 1979].

Although this scheme looks promising, it has certain drawbacks. The number of bits per sample available for quantization is reduced significantly. For example, for the bit rate of 9.6 kbps and sampling rate of 6.4 kHz, the bits available per sample are reduced from 3 to 2.25 for 3:2 compression ratio scheme. Such reduction in available entropy, not only leads to more quantization noise, but makes fewer bits available for error protection, thus making the system more susceptible to channel noise.

For robust 9.6 kbps system, a compression ratio of 2:1 was thought to be a good choice. However, for higher bit rates and/or less noisy channel, a compression scheme of 3:2 would be an attractive choice. Should the pitch periods \((N_p)\) form the side information, a 2:1 scheme would require transmitting the pitch information every \(2N_p\) samples as against \(3N_p\) samples in a 3:2 scheme. Therefore, a 3:2 scheme would require 33% less bit rate to transmit the pitch than for a 2:1 scheme.

2.6 PITCH EXTRACTION

In earlier sections, it was shown that the TDHS algorithm consists of properly weighing several adjacent input signal segments with pitch dependent duration by a suitable window function, to produce an output segment. In the frequency domain, the time-domain operations are equivalent to shifting the individual pitch harmonics of the quasi-periodic voiced-speech signal according to the center frequency of the subband in which
each harmonic component is located. The number of subbands into which the speech band is divided is pitch dependent. The pitch adaptive nature of the algorithm requires a pitch extraction operation in the system. The choice of a method to be used for extraction depends on how accurate the pitch estimation needs to be.

Since the TDHS algorithm is pitch adaptive, an error in the pitch estimation may cause a distortion in the output speech. Malah, in his recent work [1981], has given the upperbound for an error in the pitch period estimation for the exact reconstruction of the signal harmonics after the frequency compression followed by the frequency expansion. The upperbound, given by Malah, is inversely proportional to the compression ratio and the maximum frequency to be reconstructed exactly. For fairly good quality speech reproduction, at least the second formant frequency of the voiced speech should be reconstructed correctly. This fact can crudely be verified by listening to the speech which is passed through a low-pass filter with different cutoff frequencies. The second formant frequencies for most of the vowels are located below 1.5 to 2 KHz [Rabiner and Schafer, 1978]. With the compression ratio of 2, the allowed pitch-period error for the above frequencies become 0.16 and 0.125 msec respectively. With sampling rate of 6400 Hz, this is equivalent to 0.8 to 1 sample error. In the simulation studies, it was noticed that a pitch error of twice this amount could be tolerated which confirms with Malah’s observation [1981].

Several pitch extraction techniques exist in the literature [Gold & Rabiner 1969; Ross, et. al 1974; Sondhi 1968; Rabiner 1977]. Many of these techniques have some form of a logic for making a voiced/unvoiced
(V/UV) decision as well as for the pitch data smoothing. The algorithm presented here does not need such a complex technique and therefore, only simple techniques will be studied. Two such techniques are autocorrelation [Rabiner, 1977] and AMDF [Ross, et. al 1974]. The pitch was estimated by using the above methods for original speech, center clipped speech [Sondhi, 1968], 3-value center clipped speech [Dubnowski, 1976] and 2-value clipped speech. The autocorrelation and AMDF methods, combined with the four different types of speech input, form essentially eight different techniques. The methods and the results are discussed in the following paragraphs.

Simple methods such as autocorrelation and AMDF, were tried for pitch estimation. These methods estimate the pitch periods quite accurately in the voiced segments of speech, except for double or triple pitch picking. The double pitch-period corresponds to performing the filter bank analysis with twice as many filters. This means that only every other filter contains a pitch harmonic. This algorithm does not need any voiced-unvoiced decision. All these reasons made it possible to use a simple pitch extraction method.

The autocorrelation method involves forming the short time autocorrelation function as in Eq. (2.17)

$$R(\tau) = \sum_{m=0}^{N-1} s(m) s(m+\tau) \quad T_{\text{min}} \leq \tau \leq T_{\text{max}} \quad (2.17)$$

The autocorrelation function representation of the signal is a convenient way of displaying certain properties of the signal. For example, if the signal is periodic with period equal to \(T\) samples, it can be shown that
and \( R(\ell) = R(\ell + T) \)

The pitch determination involves computing \( R(\ell) \) as in Eq. (2.17) for different lags \( \ell \) and locating the maximum. To check the periodicity of the waveform one needs to check at least two periods. Therefore, the searching range for \( \ell \) is of the order of two pitch periods. The blocksize \( N \) should be chosen to give a good indication of the changing properties of the speech signal. A block size on the order of a pitch period was found to be a good choice.

The above method, though simple to implement, involves extensive computations. For example, if the block length is \( N \) and the searching range is \( r \), then for each value of \( r \) there are \( N \) multiplications and \( N \) additions. Hence the total number of multiplications for finding the pitch period becomes \( N \cdot r \); if \( r = 150 \) and \( N = 80 \) then this number is 12000. This is just for one block of samples. This many multiplications consumes significant processing time which may cause problem in a real-time implementation. This led to a search for another technique which is computationally simpler and yet provides accurate results.

This is possible by the use of Average Magnitude Difference Function (AMDF). This technique is based upon the idea that for a truly periodic input of period \( T \), the sequence

\[ d(n) = s(n + k) - s(n) \]

would be zero for \( k = 0, \pm T, \pm 2T \ldots \) For a short-segment of voiced speech, it is reasonable to expect that \( d(n) \) will be small at multiples of the period, but not identically zero. The short-time average magnitude of \( d(n) \) as a function of \( k \) should be small whenever \( k \) is close to the period. The short-time AMDF [Ross, et. al, 1974] is thus defined as
The considerations for the choice of blocksize and the searching range is the same as discussed above. The AMDF function is implemented with subtraction, addition and absolute value operations, in contrast to addition and multiplication operations for the autocorrelation function. With floating point arithmetic, where multiplies and adds take approximately the same time, about the same time is required for either method with the same window length. However, for special purpose hardware, or with fixed point arithmetic, the AMDF appears to have an advantage. In this case, multiplies usually are more time consuming and furthermore either scaling or a double precision accumulator is required to hold the sum of lagged products.

One of the major limitations of the autocorrelation representation is that in a sense it retains too much of the information in the speech signal. As a result, the autocorrelation function has many peaks. Most of these peaks can be attributed to the damped oscillations of the vocal tract response which are responsible for the shape of each period of speech wave. Usually the peak at the pitch period has the greatest amplitude (smallest in case of AMDF). However, rapidly changing format frequencies can cause bigger autocorrelation peaks than those due to the periodicity of the vocal excitation. In such cases, the simple procedure of picking the largest peak in autocorrelation will fail.

To avoid this problem it is again useful to process the speech signal so as to make the periodicity more prominent while suppressing other
distracting features of the signal. Techniques which perform this type of operation on a signal are sometimes called "spectrum flatteners" since their objective is to remove the effects of the vocal tract transfer function, thereby bringing each harmonic to the same amplitude level as in the case of a periodic impulse train. Numerous spectrum flattening techniques have been proposed [Sondhi, 1968]; one technique is called "center clipping" [Sondhi, 1968], and is obtained by a nonlinear transformation

\[ y(n) = C[x(n)] \]

where \( C[ ] \) is shown in Fig. 2.8(a). The operation of center clipper is depicted in Fig. 2.8(b). From a block of speech samples the absolute maximum amplitude \( S_{\text{max}} \) is found; the clipping level \( C_L \) is set equal to a fixed percentage of \( S_{\text{max}} \). It can be seen that for samples above \( C_L \), the output of the center clipper is equal to the input minus the clipping level. For samples below the clipping level, the output is zero.

Clearly, setting the clipping level is important. If the clipping level is large, only a small number of peaks will exceed the clipping level and only a few undesirable pulses will occur in the output. The clearest indication of periodicity is obtained for the highest possible clipping level. There is, however, a difficulty with using a too high a clipping level. It is possible that the amplitude of the signal may vary appreciably across the duration of the speech segment (e.g. at the beginning or end of voicing) so that if the clipping level is set at a high percentage of the maximum amplitude across the whole segment, there is a possibility that much of the waveform will fall below the clipping level and be lost. In the simulation studies, it was found that such situation
Fig. 2.8 (a) Center clipping function

Fig. 2.8 (b) Center clipping operation.
is avoided if clipping level is kept around 30%. This same observation was also made by Sondhi [1968]. If the clipping level is more than 60% the situation noted above usually does occur and the estimation of the pitch period is in error. This is shown in Table 2.3 by the arrows pointed to the wrong pitch periods. A procedure [Rabiner and Schafer, 1978] which permits a greater percentage (60-80%) to be used is to find the peak amplitude in both the first third and the last third of the segment and set the clipping level at a fixed percentage of the minimum of these two maximum levels. This procedure was incorporated in the pitch extraction algorithm and results are shown in Table 2.3. The table shows the analysis intervals and the corresponding pitch periods. The analysis interval is chosen to be 200 samples. The first and the last sample number of this interval is obtained as follows. Suppose the first sample number of an analysis interval is n. The last sample number of the interval becomes n+199 to make the length equal to 200. Let the pitch period for the analysis interval (n to n+199) be N_p. The next interval becomes (n+N_p to n+2N_p +199). Since the pitch is different for different pitch extraction techniques in unvoiced segment of speech, the analysis interval differs. This can also happen due to double or triple pitch picking in the voiced speech. Comparing the results in Table 2.3 and it can be seen that pitch errors due to high clipping level are corrected by using the procedure outlined above. Note that the double or triple pitch periods are not considered pitch errors because they do not degrade the performance.

A simple modification of the center clipping function leads to a great simplification in computation of the autocorrelation function with
TABLE 2.3
The effect of clipping level on the pitch extraction for female speaker.

<table>
<thead>
<tr>
<th>Sample #</th>
<th>Pitch</th>
<th>Sample #</th>
<th>Pitch</th>
<th>Sample #</th>
<th>Pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>7779 - 7970</td>
<td>28</td>
<td>7769 - 7968</td>
<td>28</td>
<td>7781 - 7980</td>
<td>72</td>
</tr>
<tr>
<td>7819 - 8010</td>
<td>25</td>
<td>7869 - 8068</td>
<td>28</td>
<td>7925 - 8124</td>
<td>77</td>
</tr>
<tr>
<td>7869 - 8068</td>
<td>25</td>
<td>7899 - 8099</td>
<td>28</td>
<td>8035 - 8235</td>
<td>29</td>
</tr>
<tr>
<td>7919 - 8119</td>
<td>26</td>
<td>8043 - 8242</td>
<td>29</td>
<td>8093 - 8393</td>
<td>29</td>
</tr>
<tr>
<td>8021 - 8220</td>
<td>28</td>
<td>8181 - 8380</td>
<td>29</td>
<td>8251 - 8450</td>
<td>29</td>
</tr>
<tr>
<td>8073 - 8272</td>
<td>29</td>
<td>8141 - 8340</td>
<td>29</td>
<td>8399 - 8598</td>
<td>29</td>
</tr>
<tr>
<td>8129 - 8328</td>
<td>29</td>
<td>8181 - 8380</td>
<td>29</td>
<td>8437 - 8636</td>
<td>30</td>
</tr>
<tr>
<td>8167 - 8366</td>
<td>29</td>
<td>8239 - 8438</td>
<td>29</td>
<td>8477 - 8676</td>
<td>30</td>
</tr>
<tr>
<td>8245 - 8444</td>
<td>29</td>
<td>8297 - 8496</td>
<td>29</td>
<td>8667 - 8866</td>
<td>30</td>
</tr>
<tr>
<td>8361 - 8560</td>
<td>30</td>
<td>8355 - 8554</td>
<td>30</td>
<td>8727 - 8926</td>
<td>30</td>
</tr>
<tr>
<td>8421 - 8620</td>
<td>30</td>
<td>8415 - 8614</td>
<td>30</td>
<td>8787 - 8986</td>
<td>30</td>
</tr>
<tr>
<td>8481 - 8680</td>
<td>30</td>
<td>8476 - 8674</td>
<td>30</td>
<td>8847 - 8946</td>
<td>30</td>
</tr>
<tr>
<td>8541 - 8740</td>
<td>30</td>
<td>8536 - 8734</td>
<td>30</td>
<td>8907 - 9106</td>
<td>30</td>
</tr>
<tr>
<td>8681 - 8880</td>
<td>30</td>
<td>8595 - 8794</td>
<td>30</td>
<td>9067 - 9267</td>
<td>92</td>
</tr>
<tr>
<td>8661 - 8860</td>
<td>30</td>
<td>8655 - 8854</td>
<td>30</td>
<td>9127 - 9327</td>
<td>92</td>
</tr>
<tr>
<td>8721 - 8920</td>
<td>30</td>
<td>8715 - 8914</td>
<td>30</td>
<td>9187 - 9387</td>
<td>92</td>
</tr>
<tr>
<td>8781 - 8980</td>
<td>30</td>
<td>8775 - 8974</td>
<td>30</td>
<td>9247 - 9447</td>
<td>92</td>
</tr>
<tr>
<td>8841 - 9040</td>
<td>30</td>
<td>8835 - 9034</td>
<td>30</td>
<td>9307 - 9507</td>
<td>92</td>
</tr>
<tr>
<td>8901 - 9100</td>
<td>91</td>
<td>8895 - 9094</td>
<td>91</td>
<td>9367 - 9567</td>
<td>92</td>
</tr>
<tr>
<td>9063 - 9262</td>
<td>31</td>
<td>8977 - 9176</td>
<td>92</td>
<td>9427 - 9627</td>
<td>92</td>
</tr>
<tr>
<td>9145 - 9344</td>
<td>31</td>
<td>9021 - 9221</td>
<td>93</td>
<td>9487 - 9687</td>
<td>92</td>
</tr>
<tr>
<td>9227 - 9426</td>
<td>31</td>
<td>9073 - 9273</td>
<td>93</td>
<td>9547 - 9747</td>
<td>92</td>
</tr>
<tr>
<td>9289 - 9488</td>
<td>31</td>
<td>9125 - 9325</td>
<td>93</td>
<td>9607 - 9807</td>
<td>92</td>
</tr>
<tr>
<td>9351 - 9552</td>
<td>31</td>
<td>9177 - 9377</td>
<td>93</td>
<td>9667 - 9866</td>
<td>30</td>
</tr>
<tr>
<td>9433 - 9633</td>
<td>31</td>
<td>9229 - 9429</td>
<td>93</td>
<td>9727 - 9927</td>
<td>30</td>
</tr>
<tr>
<td>9515 - 9714</td>
<td>31</td>
<td>9281 - 9481</td>
<td>93</td>
<td>9787 - 9987</td>
<td>30</td>
</tr>
<tr>
<td>9545 - 9745</td>
<td>31</td>
<td>9333 - 9533</td>
<td>93</td>
<td>9847 - 9947</td>
<td>30</td>
</tr>
<tr>
<td>9639 - 9838</td>
<td>31</td>
<td>9385 - 9585</td>
<td>93</td>
<td>9877 - 9977</td>
<td>30</td>
</tr>
<tr>
<td>9721 - 9920</td>
<td>31</td>
<td>9437 - 9637</td>
<td>93</td>
<td>9937 - 9937</td>
<td>30</td>
</tr>
<tr>
<td>9805 - 10004</td>
<td>31</td>
<td>9487 - 9687</td>
<td>93</td>
<td>10005 - 10005</td>
<td>30</td>
</tr>
<tr>
<td>10087 - 10286</td>
<td>91</td>
<td>10055 - 10255</td>
<td>91</td>
<td>10055 - 10255</td>
<td>91</td>
</tr>
<tr>
<td>10169 - 10368</td>
<td>31</td>
<td>10237 - 10437</td>
<td>30</td>
<td>10237 - 10437</td>
<td>30</td>
</tr>
<tr>
<td>10311 - 10510</td>
<td>91</td>
<td>10317 - 10517</td>
<td>30</td>
<td>10317 - 10517</td>
<td>30</td>
</tr>
<tr>
<td>10373 - 10572</td>
<td>31</td>
<td>10377 - 10577</td>
<td>30</td>
<td>10377 - 10577</td>
<td>30</td>
</tr>
<tr>
<td>10555 - 10754</td>
<td>31</td>
<td>10437 - 10637</td>
<td>30</td>
<td>10437 - 10637</td>
<td>30</td>
</tr>
<tr>
<td>10497 - 10696</td>
<td>30</td>
<td>10497 - 10696</td>
<td>30</td>
<td>10497 - 10696</td>
<td>30</td>
</tr>
</tbody>
</table>
essentially no degradation in utility for pitch detection [Rabiner, Schafer, Dubnowski, 1976]. This modification is shown in Fig. 2.9. As indicated there, the output of the clipper is +1 if \( x(n) > C_L \) and is -1 if \( x(n) < -C_L \). Otherwise the output is zero. Although this operation tends to emphasize the importance of peaks that just exceed the clipping level, the autocorrelation function is very similar to that of the center clipper of Fig. 2.8.

The computation of the autocorrelation function for a 3-level center clipped signal is particularly simple. The product \( s(k)s(k+\ell) \) in Eq. (2.17) can have only three values.

\[
s(k)s(k+\ell) = \begin{cases} 
0 & \text{if } s(k) = 0 \text{ or } s(k+\ell) = 0 \\
+1 & \text{if } s(k) = s(k+\ell) \\
-1 & \text{if } s(k) \neq s(k+\ell)
\end{cases}
\]

Thus, in hardware terms, only simple combinatorial logic and an up-down counter is required to accumulate the autocorrelation value for each value of \( \ell \). Likewise, the input to the AMDF algorithm could also be center clipped speech.

The two-level or infinite clipper shown in Fig. 2.10, was tried for a hardware simplification to the 3-level clipper described above. However, the pitch period estimation was not accurate. The explanation could be derived from Licklider and Pollack's [1948] experiment. They showed that, whereas speech that has been infinitely peak clipped is highly intelligible, even a few percent of center clipping drastically reduces intelligibility. The reason is infinite peak clipping retains the formants of the speech signal (although it introduces a few secondary "formants"), center
Fig. 2.9 3-level center clipping function

Fig. 2.10 2-level center clipping function
clipping destroys the formant structure, while retaining the periodicity. It is the removal of formant structure that is so important for a good pitch extractor.

Table 2.4 shows the output of the pitch extractor for seven different techniques. The one which uses three level center clipper was found to be accurate enough for the desired applications and hence was selected for its hardware simplicity. Note that, as discussed earlier, the analysis intervals differ for the different pitch estimators. Hence, the care must be taken to compare the pitch period variations for the entire segment of voiced speech instead of one particular analysis interval.

The hardware simplicity of the pitch detector becomes quite important because of the unique feature of this algorithm. The pitch is extracted at the receiver from the reconstructed compressed speech. The two pitch extractors, one at the transmitter and another at the receiver, increase the cost of the system hardware. However, for a 3-level center clipped autocorrelation method, hardware requirements are minimum and hence, the hardware cost is marginal. The pitch extraction at the receiver becomes a little more difficult because the input speech for the pitch detector is noisy compressed speech, possibly with channel errors. The noise in the reconstructed compressed speech is the quantization noise which is small enough to not cause a pitch detection error. However, a smaller number of pitch periods per unit time are available in the compressed speech. This is the major problem to extracting pitch at the receiver. The problem could become serious for male speakers because of the small value of the fundamental frequency. The pitch extraction techniques outlined above can still be used at the receiver. However, a judicious choice of the
TABLE 2.4

Pitch extracted using different pitch estimation techniques. Sentence 11, Female speaker
Block size = 50, Searching range = (20,100). Sampling frequency = 6400 Hz.

<table>
<thead>
<tr>
<th>AUTO CORRELATION</th>
<th>AMDF</th>
<th>AUTO CORRELATION</th>
<th>AMDF</th>
<th>AUTO CORRELATION</th>
<th>AMDF</th>
<th>AUTO CORRELATION</th>
<th>AMDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original speech</td>
<td>original speech</td>
<td>center clipped speech</td>
<td>center clipped speech</td>
<td>J-value center clipped</td>
<td>J-value center clipped</td>
<td>2-value center clipped speech</td>
<td></td>
</tr>
<tr>
<td>sample #</td>
<td>Mp</td>
<td>sample #</td>
<td>Mp</td>
<td>sample #</td>
<td>Mp</td>
<td>sample #</td>
<td>Mp</td>
</tr>
<tr>
<td>9827 - 9278</td>
<td>16</td>
<td>9827 - 9278</td>
<td>16</td>
<td>9827 - 9278</td>
<td>16</td>
<td>9827 - 9278</td>
<td>16</td>
</tr>
<tr>
<td>9827 - 9278</td>
<td>17</td>
<td>9827 - 9278</td>
<td>17</td>
<td>9827 - 9278</td>
<td>17</td>
<td>9827 - 9278</td>
<td>17</td>
</tr>
<tr>
<td>9827 - 9278</td>
<td>18</td>
<td>9827 - 9278</td>
<td>18</td>
<td>9827 - 9278</td>
<td>18</td>
<td>9827 - 9278</td>
<td>18</td>
</tr>
<tr>
<td>9827 - 9278</td>
<td>19</td>
<td>9827 - 9278</td>
<td>19</td>
<td>9827 - 9278</td>
<td>19</td>
<td>9827 - 9278</td>
<td>19</td>
</tr>
<tr>
<td>9827 - 9278</td>
<td>20</td>
<td>9827 - 9278</td>
<td>20</td>
<td>9827 - 9278</td>
<td>20</td>
<td>9827 - 9278</td>
<td>20</td>
</tr>
</tbody>
</table>

36
blocklength and the searching range is necessary. At the transmitter, the blocklength of 80 and searching range of (20,100) samples was found to be a good choice. At the receiver, the blocklength of 50 and the same searching range was found to be satisfactory. These conclusions are based on the results obtained for three sentences. A study for several utterances might be necessary to come up with the best value for the blocklength and the searching range.

Pitch detection at the receiver avoids the transmission of pitch as a side information, thus, making the blocking of the quantizer data and the block synchronization unnecessary. However, channel noise can affect the compressed speech appreciably which in turn would cause wrong pitch estimation. For a bit-error-rate (BER) of 1%, the above situation does occur quite frequently but occurs less frequently if error protection is provided for the quantizer levels. In the simulation studies, the speech quality for the noisy channels with or without pitch transmission was found to be the same. This may be due to the masking effect of the distortion due to the errors in the quantizer levels on the distortion caused by the pitch errors. The results and further discussion of this are postponed until Chapter 4.

2.7 SUMMARY

A time domain harmonic scaling of speech was found to be an effective approach to represent speech with fewer number samples while maintaining excellent quality. A simple triangular window function was used for compression and expansion operations. A compression ratio of 2:1 was found to be a good choice. A 3-value center clipped autocorrelation technique
without any decision logic, V/UV decision or pitch data smoothing was found to be adequate for the purpose. The pitch is extracted both at the transmitter and the receiver. The harmonic distortion produced by TDHS operations could not be noticed due to masking properties of the ear. The TDHS output is coded using an ADPCM technique which is described in the next chapter.
CHAPTER 3
ADAPTIVE RESIDUAL CODER

3.1 INTRODUCTION

In the previous chapter, the compression and expansion operations performed at the transmitter and at the receiver respectively were discussed. Compressed speech, which consists of half the number of original samples for 2:1 compression ratio, needs to be coded and transmitted on the channel. This can be achieved by employing various schemes. Compressed speech is formed by using long-term redundancy in a manner different from the APC technique. Since some of the long-term redundancy in the original speech is already removed in forming the compressed speech, use of APC coder to code it would be inefficient. However, compressed speech could be effectively coded by exploiting the short term redundancy. This is done by using ADPCM coders.

The Adaptive Residual Coder (ARC) System, developed by Cohn and Melsa, [Sept. 1975] is an improved ADPCM system. The following sections describe the system structure of ARC and outline its design and performance for the compressed speech as an input signal. Various performance measures were tested for the ARC system and are also discussed in this chapter.

3.2 THE RESIDUAL ENCODER STRUCTURE

Fig. 3.1 shows the basic ADPCM structure augmented with dashed lines to indicate the flow of information for adaptation in the residual encoder. The underlying design principle of the adaptation procedure is
Fig. 3.1 Adaptive Residual Coder (ARC) System
that all information used in updating the quantizer and predictor be
available both at the transmitter and the receiver. Since the only infor-
mation sent from the transmitter to the receiver is the quantizer output
$q_k$, all adaptation procedures must use quantities derivable from the $q_k$.

As mentioned earlier, the compressed speech samples $y_k$ form the input
to the ARC system. The sampling rate of $y_k$ depends on the sampling fre-
quency of original speech and the compression ratio employed in harmonic
scaling operations. For example, if the input speech bandwidth is 3200 Hz
and it is sampled at the Nyquist rate, and with a 2:1 compression ratio,
then the sampling frequency of $y_k$ becomes 3200 Hz and the bandwidth is
compressed into 1600 Hz.

The system shown in Fig. 3.1 includes the pitch compensating adaptive
quantizer. The object of an adaptive quantizer is to match the quantizer
to the local statistics of the speech rather than to global or "average"
statistics. Thus, if the speaker is talking loudly, a quantizer with a
wide range should be used; but if the speaker is talking quietly, a narrow
range should be used. In fact, the dynamic range of human speech varies
significantly from syllable to syllable.

The usual method of quantizer adaptation is to adjust the range of
the quantizer as a function of the prior quantizer output. Consider the
quantizer structure shown in Fig. 3.2. Here the input is normalized by a
state variable $\sigma_k$ and then quantized with a fixed quantizer. If the ex-
pected range of the input signal is large, $\sigma_k$ should be large; if the ex-
pected range is small, $\sigma_k$ should be small. If the prior quantizer input
was large, then from the correlation between successive samples, one might
Fig. 3.2 Adaptive Quantizer
expect the next input to be large as well. Therefore, if the quantizer output is large, \( \sigma_k \) should be increased. Similarly, if the quantizer output is small, \( \sigma_k \) should be decreased.

Two different approaches have been suggested for quantizer adaptation: forward adaptation and backward adaptation. In a forward scheme, the adaptation decision is based on unquantized data and is communicated to the receiver as side information. Performance tests indicate (Noll, 1974) that the forward method is slightly better if no cost is assessed for the side information. Practical considerations, however, seem to favor backward adaptation.

Several investigators have evaluated adaptive PCM and DPCM systems that incorporate adaptive quantizer (Cummisky, et al., 1973; Gibson, et al., 1974; Castellino, et al., 1974; Stroh, 1971). These systems have consistently shown considerable performance improvements over those which do not use adaptive quantizers. The adaptation procedures do require a mild increase in system complexity; some, however, can be realized with simple shift-and-add operations (Qureshi and Forney, 1975).

Adaptive quantizers have been the subject of intense theoretical study (Goodman and Gersho, 1974; Mitra, 1974a; Mitra, 1975; Cohn and Melsa, 1975). One result of significant intuitive interest unites two apparently different schools of thought on adaptation design. One approach, used in both backward and forward adaptation, normalizes the quantizer input by an estimate of its envelope or RMS level. The other method, first proposed by Jayant (1973), scales the normalization constant according to the prior output; if it is an outer level, the constant is
increased, if it is an inner level, the constant is decreased. Cohn and Melsa (1975a) showed that if some mild conditions are met, the Jayant method is also an envelope estimation.

Initial designs of Jayant quantizers proved to be very sensitive to channel errors. Later work (Cohn and Melsa, 1975a; Goodman and Wilkinson, 1975; Cohn and Melsa, 1975b; Qureshi and Forney, 1975) has shown ways of modifying the algorithm to eliminate this problem.

It is clear then, that adaptive quantizers improve performance because they match their range to the dynamic range of the incoming signal. A pitch compensating quantizer (Cohn and Melsa, 1976) extends this notion by noting that during voiced speech the dynamic range of the input is critically dependent on the proximity of the last pitch pulse.

It is well known that during voiced speech, the signal strength is a local maximum shortly after a pitch pulse and tends to decay towards the end of pitch period. This effect is even more pronounced in the difference signal that is quantized in DPCM systems. Although the predictor may closely approximate the actual speech away from a pitch pulse, it generally cannot predict the next pitch pulse. The design objective of a pitch compensating quantizer is to adapt to both the long term syllable variations in signal strength and to the short term variations.

The backward adapting pitch compensation algorithm uses a quantizer whose outermost levels have been set at higher values than is normal. When the quantizer output is one of these outermost levels, the adaptation algorithm reacts as if a pitch pulse had been detected; the quantizer state variable is significantly increased and then permitted to rapidly decay back to its long-term value.
The pitch compensating quantizer (PCQ) is employed in this algorithm. The prediction error $e(k)$ is the input to the quantizer whose basic design is illustrated in Fig. 3.2. The recommended thresholds are symmetric and are illustrated in Fig. 3.3 and listed in Table 3.1. The level in which the normalized input falls specifies the quantizer output $q(k)$. The inverse quantizer output $e(k)$ is the quantized version of the quantizer input. It is the product of a scaling factor $f(q(k))$ and the state variable $o(k)$. The recommended scale factors are tabulated in Table 3.2. The recommended thresholds were computed to be equidistant between between the scaling factors.

The state variable $o(k)$ is designed to be an approximation to the standard deviation of $e(k)$. Most of the time the scaled average of $|y(k)|$ is an acceptable estimate. However, in voiced speech at the beginning of a pitch period, $e(k)$ is much larger than usual. Therefore, whenever one of the outermost quantizer level occurs, $o(k)$ decays back to the scaled average of $|y(k)|$. Thus $o(k)$ is updated by

$$o(k) = \max\{\text{SMIN} \langle |y(k)| \rangle, \phi[q(k)] o(k-1)\} \quad (3.1)$$

The first term in the braces of Eq. (3.1) usually dominates. This means that quantizer behavior is largely determined by $\text{SMIN} \langle |y(k)| \rangle$ and, hence, by the product of $\text{SMIN}$ and $\text{RMSMIN}$. It is recommended that the scale factor $\text{SMIN}$ be set to 0.25. The second term in the braces only affects performance at the beginning of pitch periods. The quantizer expansion factors $\phi[q(k)]$ are given in Table 3.2.

The choice of output levels, expansion factors and other scalars in the design of quantizer is largely governed by maintaining the average
Fig. 3.3 An adaptive quantizer with thresholds and output levels.
### TABLE 3.1
Quantizer Thresholds

<table>
<thead>
<tr>
<th>i</th>
<th>( T_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>1.05</td>
</tr>
<tr>
<td>3</td>
<td>2.55</td>
</tr>
<tr>
<td>4</td>
<td>5.55</td>
</tr>
<tr>
<td>5</td>
<td>11.55</td>
</tr>
</tbody>
</table>

### TABLE 3.2
Quantizer scaling and expansion factors

<table>
<thead>
<tr>
<th>q(k)</th>
<th>( f[q(k)] )</th>
<th>( \phi[q(k)] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.8</td>
</tr>
<tr>
<td>2,3</td>
<td>0.6</td>
<td>0.9</td>
</tr>
<tr>
<td>4,5</td>
<td>1.5</td>
<td>9.0</td>
</tr>
<tr>
<td>6,7</td>
<td>3.0</td>
<td>1.5</td>
</tr>
<tr>
<td>8,9</td>
<td>6.0</td>
<td>3.0</td>
</tr>
<tr>
<td>10,11</td>
<td>12.00</td>
<td>6.0</td>
</tr>
</tbody>
</table>
source entropy within acceptable limits and at the same time reducing the quantization noise. Usually, the average number of bits available for coding quantizer levels is dependent on bit rate available and the sampling frequency. In this algorithm, fixed length code with run length is used. This requires that quantizer level occupancy statistics be such that probability of run lengths is maintained within a certain range. This makes the average bit rate remain below the allowable limit. The detail discussion may be found in Chapter 4.

The ARC system employs a sequentially adaptive linear predictor. It produces a linear prediction \( p(k) \) given by

\[
p(k) = \sum_{i=1}^{N} a_i(k) y(k-i)
\]

which is to be an estimate of \( y(k) \). The \( y(k-i) \) are the receiver's estimate of \( y(k-i) \). Since \( y(k) \) is frequency compressed to half the original bandwidth, the number of poles required to represent compressed bandwidth would be half that required for original bandwidth. A predictor order of four was found to be a good choice.

If the \( a_i(k) \) accurately model the \( y(k) \), and if the \( y(k-i) \) are close to the \( y(k-i) \), then \( p(k) \) will be a good approximation to \( y(k) \). The \( a_i(k) \) are adaptive, and after \( p(k) \) is formed, they are updated. They are adapted according to steepest descent of \( e^2(k) \) [Melsa & Cohn, 1975]. This is approximated in the system by the following updating algorithm:

\[
a_i(k+1) = \delta b_i + (1-\delta)[a_i(k) + \frac{g \hat{y}(k-i)e(k)}{\langle |y(k)| \rangle^2}]
\]

where \( \langle |y(k)| \rangle \) is a biased exponential time average of \( |y(k)| \).
Thus, the $a_i(k)$ updating algorithm has eight parameters: $\delta, g, a, \text{RMSMIN}$ and $b_i$ for $i=1,2,3$ and 4. It was found in the simulation studies that the effect of channel errors become more severe if the memory in the updating process is increased. This increase is essentially controlled by choice of parameters $\delta, g$ and $a$. In order to minimize the effect of channel errors, the memory time was reduced from what would be optimal in error-free case. This did not significantly degrade performance. The recommended values of these parameters are

$$\begin{align*}
\delta &= 0.05 \\
g &= 0.02 \\
a &= 0.93
\end{align*}$$

The parameters $b_i$ represent the quiescent values of the coefficients $a_i(k)$. The values used are

$$b_i = \begin{cases} 0.7 & i=1 \\ 0 & i=2,3 \text{ or } 4 \end{cases}$$

The quantity RMSMIN is perhaps the most sensitive parameter in the algorithm. It determines the minimum value of $\langle |y(k)| \rangle$ which affects both the adaptive predictor and the adaptive quantizer. As the RMSMIN decreases, the system responds more during low level signals. This reduces granular noise and increases the data rate. The higher data rate means that the buffer fills faster and that buffer control is triggered causing deterioration of speech quality. When $y(k)$ is represented on the interval $[-2048, 2047]$, RMSMIN of 40 produces a good tradeoff.
3.3 PERFORMANCE

The performance of the TDHS-ARC system depends on how well the frequency scaling operations are performed and how well the quantization is done. As mentioned earlier, the ARC system is essentially an ADPCM system and for the given bit rate, its performance can be improved by decreasing the quantization noise. However, a reduction in the quantization noise does not necessarily mean an improvement in the performance. Such situations will be discussed later on in this section.

The bit rate available for coding quantizer levels depends on the bit rate needed for error protection and to transmit the side information, if any. Since it was decided to extract pitch at the transmitter and at the receiver as well, this system has no side information. With (26, 31) Hamming code for error protection [Hamming, 1980], a bit rate of 8 kbs is available to code the quantizer levels generated by ARC. The sampling rate is 3200 Hz, which leaves an average of 2.5 bits per sample. The parameters were designed such that the average value of the source entropy is around 2.5 bits per sample. However, the instantaneous value of the bit rate has to be considered for the buffer overflow problem.

Before discussing the performance of the ARC system, some of the performance indicators are presented. The single most widely used indicator of speech coder performance is the SNR defined by

\[
\text{SNR} = \frac{\langle y^2(k) \rangle}{\langle |y(k) - \hat{y}(k)|^2 \rangle}
\]  

(3.4)

or in dB by
where \( y(k) \) and \( y(k) \) are shown in Fig. 3.4(a) and (b). \( <|1|2> \) denotes the averaging operation. It is well known that the SNR calculated as in Eqs. (3.4) and (3.5) is not a perfect indicator of speech quality and intelligibility. This is because perceived quality and intelligibility depend on the subjective loudness of the quantization noise not just on the quantization noise power, which is the denominator of Eq. (3.4). The numerator of Eq. (3.4) is an average of a square term. Thus, speech with high amplitudes gets more weighing in the calculation of the SNR. Makhoul & Berouti (Feb. 1979) and Atal & Schroeder (June 1979) have shown that due to the auditory masking properties of the human ear, the subjective loudness is determined by the spectrum of the quantization noise and its relationship to the input signal spectrum. The SNR in (3.4) does not reflect these considerations. However, the SNR remains popular since it is simple to compute, and it can be useful if applied prudenty.

Usually, SNR is calculated for the entire utterence. As discussed earlier, the high energy segments of speech get more weighing in the SNR computation than the low energy segments. Hence, SNR values indicate the performance of coder for high energy speech, or voiced speech. The coder performance for low energy, or unvoiced speech can be obtained by computing the SNR in Eqs. (3.4) and (3.5) over many nonoverlapping blocks of data within an utterence [Noll, 1975]. The time variation of the resulting "short-term" SNR provides an indication of how well the coder under consideration is performing on the various blocks of speech data. Based on these short term SNR values, a performance measure SNRSEG can be defined as
Fig. 3.4  
(a) Compressed speech, $y(k)$  
(b) Reconstructed compressed speech, $\hat{y}(k)$  
(c) Quantization error, $e(k)$
Infrequent very large values of SNR\textsubscript{j} tend to show up better in SNRSEG than in SNR in (3.4) [Jayant, 1977]. One slight modification to computation of SNRSEG in (3.6) would be to exclude the SNR\textsubscript{j} values for the blocks which contain silence. Usually, SNR values for silence are zero or negative; those can be excluded from SNRSEG computation.

A distortion measure that is related to the SNR is the SPER given by [Gibson, 1980]

$$\text{SPER} = \frac{<y^2(k)>}{<[y(k) - p(k)]^2>}$$

where \(p(k)\) is the predicted value in Fig. 3.1. The SPER is motivated by a desire to evaluate the predictor performance only, rather than to try to obtain an indicator of the overall system performance which in DPCM includes both the quantizer and the predictor effects. Of course, due to the closed loop nature of DPCM and the presence of the quantizer within the loop, the SPER is also affected by the quantizer.

Another performance indicator for DPCM is the SNRI defined by [McDonald, 1966; O'Neal & Stroh, 1972]

$$\text{SNRI} = \frac{<y^2(k)>}{<[y(k) - p_1(k)]^2>}$$

where

$$p_1(k) = \sum_{i=1}^{N} a_i y(k-i).$$

The SNRI is simply the SPER when it is assumed that \(y(k)=y(k)\) or \(p(k)=p_1(k)\). The SNRI is interpreted as the amount by which linear prediction
can reduce the input signal power and hence is sometimes used as a measure of maximum utility of linear prediction in DPCM.

Although the SNR, SPER and SNRI are easy to calculate and hence very useful for initial system evaluations, they are not absolute indicators of system performance. In fact, the SNR may not rank the coders correctly in terms of speech quality and intelligibility. As a result, it is advisable to augment these previous indicators with subjective listening tests and frequency spectrum plots. By comparing frequency spectrum of the speech coder output with frequency spectrum of the original speech, conclusions can be drawn concerning how well the coder tracks the various formants, reproduces unvoiced or voiced sounds and in frequency quantizer noise in prevalent. By combining all these techniques, coder degradation can be analyzed.

For evaluating the performance of the ARC and also of the entire system, the following three phonetically balanced sentences were used.

1) "Cats and dogs each hate the other" (Male speaker).
2) "Move the vat over the hot fire" (Male speaker).
3) "The pipe began to rust while new" (Female speaker).

The input data were low pass filtered at 2900 Hz (3dB) and sampled at 6.4 kHz. The SNR and the SEGSNR in time and frequency domain are given in Table 3.3. Frequency domain SNR indicates how good the match between input and output speech in frequency domain is. It is calculated by taking DFT of small segment (20 msec) of input and output speech and calculating SNR, as in Eq. (3.9)
TABLE 3.3

ARC Performance

Block length = 128 samples, Sampling freq. = 3200 Hz

<table>
<thead>
<tr>
<th>Sent #</th>
<th>SNR</th>
<th>SEGSNR time</th>
<th>SEGSNR frequency</th>
<th>Entropy H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male 1</td>
<td>20.00 dB</td>
<td>16.39 dB</td>
<td>19.28 dB</td>
<td>2.47 b/sample</td>
</tr>
<tr>
<td>Female 11</td>
<td>19.83 dB</td>
<td>17.57 dB</td>
<td>19.74 dB</td>
<td>2.59 b/sample</td>
</tr>
<tr>
<td>Male 6</td>
<td>20.01 dB</td>
<td>18.12 dB</td>
<td>20.61 dB</td>
<td>2.57 b/sample</td>
</tr>
</tbody>
</table>
where $Y(n)$ and $Y(n)$ are frequency components of $y(k)$ and $y(k)$.

$\text{SNR}_{\text{freq}} = 10 \log_{10} \frac{\langle |Y(n)|^2 \rangle}{\langle [|Y(n)| \cdot |Y(n)|] \rangle}$

(3.9)

Fig. 3.4 shows the compressed speech $y(k)$, reconstructed compressed speech $\hat{y}(k)$, and the quantization error. It can be seen that the ARC system reconstructs the input speech very well, especially in the voiced speech segment. This is more clear in SEGSNR plot of Fig. 3.5, where block to block SNR is plotted for the whole utterance against time. The plot shown in the dashed lines is for the SNR in frequency domain. The distortions in frequency domain can be seen from the frequency spectrum plots of input and output speech as well as input and quantization noise in Figs. 3.6(a) and (b). The frequency spectrum of quantization noise lies much below that of input compressed speech, near the formant frequencies especially for first and second formant. At higher frequencies, however, quantization noise is perceptible since its spectrum lies above the input signal spectrum. In informal listening tests, it was found that this granular noise problem is much less than that in previously developed ADPCM or APC system employing waveform reconstruction techniques. Noise spectral shaping was tried by a number of authors [Atal & Schroder, 1979; Makhoul & Berouti, 1979] to improve the speech quality. However, such spectral shaping adds to the complexity of the system. In the previous research studies for developing PARC system [Melsa, et al., 1980], it was noticed that increases in complexity and levels of adaptation in the system leads to poorer performance in the presence of channel errors. Hence, the compromise between the robustness and speech quality
Fig. 3.6(a) Frequency spectrum of input and output speech of Adaptive Residual Coder.
Fig. 3.6(b) Frequency spectrum of compressed speech and quantization noise.
must be reached in the design of the ARC system. The Fig. 3.7 show the plot of SNR against the signal strength. The desirable performance would be a high value of SNR for all signal strengths. Such performance is unlikely since the predictor performs very poorly for unvoiced speech which is noiselike. However, the parameters were designed such that ARC performance approaches such curve.

The quantizer and the predictor performance for segment to segment of speech is shown in Figs. 3.8(a) and (b). The performance indicator, SNRI in Eq. (3.8), is also plotted in Fig. 3.8(a). It can be seen that the predictor performance, SPER is very close to SNRI. That means that the predictor is performing very well except in transition regions, where its poor performance is anticipated. Total performance (SNR in dB) is the addition of predictor performance (SPER in dB) and the quantizer performance (SNRQ in dB). Hence, increases in SPER and SNRQ lead to an increase in SNR. However, such simplification could be misleading since performance of the predictor depends on that of quantizer. The plots, in Figs. 3.8 often are useful in evaluating the contribution of the predictor and the quantizer in different parts of the speech utterence.

3.4 SUMMARY

The design of adaptive residual coder for coding the compressed speech was presented. An 11-level quantizer and 4th order predictor was used. The signal to quantization noise ratio as high as 20 dB could be obtained. Various performance measures to indicate the speech quality, were presented. It was found that no single criterion indicates the true speech quality. The study of combined system using TDHS and ARC is presented in the next chapter.
Fig. 3.7 Plot of SNR vs. signal strength for the Adaptive Residual Coder.
Fig. 3.8(a) Block-to-block SNR, SNRQ, SPER and SNRI for male speaker.
CHAPTER 4

TIME DOMAIN HARMONIC SCALING (TDHS) and ADAPTIVE RESIDUAL CODER (ARC) SYSTEM

4.1 INTRODUCTION

This chapter describes the complete system employing harmonic scaling and residual coding operations to achieve the desired transmission bit rate. As discussed earlier, time domain harmonic compression reduces the number of samples to be transmitted, and the adaptive residual coder encodes these samples with least possible distortion. The complete system performance depends not just on the TDHS and ARC performance but also on several factors, such as the source code, the error protection employed as well as buffer control strategy. These and other topics were investigated and the results are presented here.

Section 4.2 describes two system configurations; one with pitch as side information and the other with no side information. The source code and the study of the buffer behavior is given in Sections 4.3 and 4.4 respectively. The effects of various transmission bit error rates are examined and reported in Section 4.5. The system performance for background noise is discussed in Section 4.6.

4.2 SYSTEM CONFIGURATION

Fig. 4.1 shows the system structure of TDHS-ARC system. For simplicity the A/D and D/A converters and associated filters are not shown. The speech samples, s(k), bandlimited at 3.2 kHz and taken at Nyquist frequency, form the input. The pitch period, N_p, is extracted from a
Fig. 4.1 TDHS-ARC system.
block of speech samples using the center clipped 3 value autocorrelation method. This value of the pitch period is transferred to the time domain harmonic compression algorithm. This algorithm produces $N_p$ samples of compressed speech $y(k)$, as described in Chapter II. The sampling frequency now becomes half of the original Nyquist frequency. The smoothly varying compressed speech samples are coded using an Adaptive Residual Coder. The output of ARC is a string of quantizer outputs $q(k)$. For every sample or two samples (if two samples form the desired runlength), a 4-bit word is transmitted to the bit buffer. For the bit rate of 9.6 Kb/s and the sampling frequency of 3.2 kHz, the average number of bits per sample transmitted on the channel is 3. Depending on the Hamming code chosen for the error protection, the parity bits are also added to the buffer.

The length of the buffer was chosen to be 1024 bits which corresponds to approximately a tenth of a second delay. Depending on the bit rate generation, the buffer may overflow or underflow. To avoid such drastic situations, buffer content information is passed on to the ARC transmitter to take some action to control the buffer. This is explained in Sec. 4.3. Note that the bits transmitted on the noisy channel represent only the quantizer level information.

Bits thus transmitted are decoded at the receiver correcting any occurrence of single channel error in a Hamming block. The received quantizer level $q(k)$ forms the input to the ARC receiver whose output is the reconstructed compressed speech $y(k)$. The time domain harmonic expansion is performed on these samples. However, the pitch information is needed for the frequency multiplication operation. There are two ways to pass on the pitch
periods to the harmonic expansion operation. One scheme is to extract the pitch periods at the transmitter and send them to the receiver as side information. However, such scheme is associated with a number of disadvantages. The biggest disadvantage is the need for a blocking structure to transmit the quantizer levels. Every \(2N_p\) samples are associated with the pitch period \(N_p\). It is very important that every block of samples at the receiver must be associated with the same pitch period as that at the transmitter. This synchronization can easily be lost unless special error protection is provided for pitch information. However, such error protection as well as pitch period transmission would cost considerable bit rate. This would leave fewer bits per sample for transmitting quantizer levels and therefore, would cause more quantization noise.

In the case of pitch transmission it has been seen that the matching of pitch data and the block of quantizer levels, which is itself pitch dependent, is very sensitive to the transmission errors. Besides, for the 9.6 kb/s system, the extra bit rate required for error protection of the pitch information can not be afforded. This led to the implementation of the other scheme with pitch extraction at the receiver (see Fig. 4.1). Since the pitch period is estimated from the reconstructed compressed speech rather than reconstructed original speech, there are fewer pitch periods per unit time available for pitch extraction. This problem can be handled by using smaller blocksize and the searching range. Table 4.1 shows the pitch extracted at the transmitter and at the receiver. Since the pitch at both ends is extracted from a different type of signal as well as different number of samples, comparison should be made for the
### TABLE 4.1
Comparison of pitch periods extracted at the transmitter and at the receiver.

<table>
<thead>
<tr>
<th>Sample Pitch</th>
<th>Sample Pitch</th>
<th>Sample Pitch</th>
<th>Sample Pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1899 - 1298 27</td>
<td>645 - 744 27</td>
<td>5333 - 5532 29</td>
<td>2864 - 2863 29</td>
</tr>
<tr>
<td>1153 - 1352 22</td>
<td>572 - 711 55</td>
<td>5391 - 5599 3#</td>
<td>2693 - 2892 3#</td>
</tr>
<tr>
<td>1317 - 1516 27</td>
<td>627 - 826 55</td>
<td>6461 - 5655 3#</td>
<td>3723 - 2922 3#</td>
</tr>
<tr>
<td>1371 - 1572 27</td>
<td>692 - 881 55</td>
<td>5511 - 5710 3#</td>
<td>2753 - 2952 3#</td>
</tr>
<tr>
<td>1425 - 1624 22</td>
<td>737 - 926 55</td>
<td>5571 - 5778 3#</td>
<td>2793 - 2992 3#</td>
</tr>
<tr>
<td>1899 - 1709 55</td>
<td>792 - 991 28</td>
<td>5631 - 5831 3#</td>
<td>2813 - 2812 3#</td>
</tr>
<tr>
<td>1699 - 1999 56</td>
<td>829 - 1019 28</td>
<td>5691 - 5891 3#</td>
<td>2843 - 3842 3#</td>
</tr>
<tr>
<td>1899 - 2199 27</td>
<td>848 - 1047 27</td>
<td>5751 - 5951 3#</td>
<td>2873 - 3872 3#</td>
</tr>
<tr>
<td>1863 - 2262 27</td>
<td>875 - 1074 27</td>
<td>5811 - 6011 3#</td>
<td>2903 - 3902 3#</td>
</tr>
<tr>
<td>1917 - 2116 28</td>
<td>982 - 1181 27</td>
<td>5871 - 6071 3#</td>
<td>2933 - 3132 3#</td>
</tr>
<tr>
<td>1973 - 2172 28</td>
<td>929 - 1128 27</td>
<td>5931 - 6131 3#</td>
<td>2963 - 3162 3#</td>
</tr>
<tr>
<td>2229 - 2228 28</td>
<td>956 - 1155 28</td>
<td>5991 - 6191 3#</td>
<td>2993 - 3192 3#</td>
</tr>
<tr>
<td>2885 - 2284 28</td>
<td>984 - 1183 28</td>
<td>6051 - 6251 3#</td>
<td>3023 - 3222 3#</td>
</tr>
<tr>
<td>1012 - 1211 29</td>
<td>1041 - 1241 28</td>
<td>6111 - 6311 3#</td>
<td>3053 - 3252 3#</td>
</tr>
<tr>
<td>1012 - 1211 29</td>
<td>1041 - 1241 28</td>
<td>6171 - 6371 3#</td>
<td>3083 - 3282 3#</td>
</tr>
<tr>
<td>6231 - 6291 3#</td>
<td>6291 - 6491 3#</td>
<td>3113 - 3313 3#</td>
<td></td>
</tr>
<tr>
<td>6351 - 6551 3#</td>
<td>6411 - 6611 3#</td>
<td>3173 - 3373 3#</td>
<td></td>
</tr>
<tr>
<td>6471 - 6671 3#</td>
<td>6471 - 6671 3#</td>
<td>3233 - 3432 3#</td>
<td></td>
</tr>
</tbody>
</table>
voiced and unvoiced speech segment. Remember that the number of samples in any particular voiced segment of compressed speech is half as many as that in a voiced segment of original speech. The pitch estimator does surprisingly well in voiced regions of the utterance. However, in the transition regions between voiced and unvoiced speech pitch estimation at the receiver differs significantly from that at the transmitter. A similar situation occurs at the boundary of the two different sounds. The distortion caused by the above situations is audible in form of buzziness which can be heard only through very high quality headphone. It was thought that such distortion could be tolerated to preserve the robustness and the simplicity of the system.

Fig. 4.2 show the frequency spectrum of a 20 msec segment of input and output speech with and without pitch transmission. The speech is not quantized in order to focus attention on the distortion caused by frequency compression and expansion operation.

4.3 SIMULATION

A program was written in FORTRAN IV to simulate the Time Domain Harmonic Scaling, Adaptive Residual Coding, Coder, Decoder and Buffer control operations. The structure of the simulation is shown in Fig. 4.3. The transmitter program (TRAN·FTN) consists of the simulation of harmonic compression, pitch extraction and the adaptive residual transmitter program. This module produces the quantizer data (QUANT·DAT), pitch data (PITCH·DAT), and the output data (FOR006·DAT) files. The output data file has the record of parameters used, quantizer statistics, predictor quantizer and the ARC transmitter performance. The encoder program reads the
Fig. 4.2 Frequency spectrum of 20 msec of male speech.

- Original speech
- Output speech with pitch transmitted.
- Output speech with pitch not transmitted.
quantizer data from QUANT·DAT file and codes it and writes the code words in ENCO·DAT file. The encoder program also produces the transmitter buffer simulation file (TBUFF·DAT). The channel simulation program reads the code words from file ENCO·DAT and adds the desired channel errors in the bits and writes these corrupted code words in a file, called ERCO·DAT. The decoder program reads this file, decodes the code word into quantizer levels and produces receiver buffer simulation file (RBUFF·DAT) and the decoded quantizer level file (DECO·DAT). The receiver program combines the ARC receiver operation, pitch extraction and the time domain harmonic expansion operation. The reconstructed speech file (SHAT·DAT) and pitch data file (PITCH·DAT;2) is produced for comparisons with those at the transmitter.

Various simulation runs were made for the three utterences described in an earlier chapter. The combined system performance was evaluated by using the criteria outlined in Chapter 3. It was noticed that none of the objective measures described there indicate the true quality of output speech particularly in the case when pitch is extracted at the receiver. Table 4.2 shows the sliding SNR in the time and frequency domain for three utterances, two male and one female speaker. The negative value of SNR does not indicate the bad quality of output speech but it represents the phase difference between input and output speech. Most of the decisions regarding the speech quality were based on the informal listening tests. Fig. 4.4 show that the time plots of input and output speech for the segments of two utterences. It can be seen that the shape of the input speech waveforms is preserved.
<table>
<thead>
<tr>
<th>sentence 1</th>
<th>sentence 6</th>
<th>sentence 11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>male speaker</td>
<td>female speaker</td>
</tr>
<tr>
<td>(dB)</td>
<td>(dB)</td>
<td>(dB)</td>
</tr>
<tr>
<td>4.093</td>
<td>11.189</td>
<td>1.082</td>
</tr>
<tr>
<td>7.702</td>
<td>10.617</td>
<td>3.787</td>
</tr>
<tr>
<td>12.623</td>
<td>10.699</td>
<td>5.160</td>
</tr>
<tr>
<td>15.916</td>
<td>10.736</td>
<td>-1.163</td>
</tr>
<tr>
<td>16.425</td>
<td>0.820</td>
<td>0.900</td>
</tr>
<tr>
<td>16.028</td>
<td>10.970</td>
<td>0.953</td>
</tr>
<tr>
<td>12.721</td>
<td>6.310</td>
<td>6.170</td>
</tr>
<tr>
<td>7.193</td>
<td>9.096</td>
<td>0.610</td>
</tr>
<tr>
<td>6.078</td>
<td>1.703</td>
<td>16.720</td>
</tr>
<tr>
<td>4.920</td>
<td>16.043</td>
<td>0.880</td>
</tr>
<tr>
<td>10.670</td>
<td>16.078</td>
<td>15.679</td>
</tr>
<tr>
<td>5.983</td>
<td>7.921</td>
<td>15.174</td>
</tr>
<tr>
<td>3.942</td>
<td>10.711</td>
<td>7.314</td>
</tr>
<tr>
<td>4.161</td>
<td>11.268</td>
<td>8.043</td>
</tr>
<tr>
<td>13.612</td>
<td>10.687</td>
<td>0.133</td>
</tr>
<tr>
<td>11.933</td>
<td>11.395</td>
<td>0.974</td>
</tr>
<tr>
<td>12.672</td>
<td>12.299</td>
<td>0.885</td>
</tr>
<tr>
<td>13.298</td>
<td>12.306</td>
<td>2.805</td>
</tr>
<tr>
<td>15.903</td>
<td>10.249</td>
<td>2.085</td>
</tr>
<tr>
<td>16.537</td>
<td>10.011</td>
<td>2.085</td>
</tr>
<tr>
<td>15.497</td>
<td>10.734</td>
<td>2.085</td>
</tr>
<tr>
<td>16.464</td>
<td>16.769</td>
<td>12.090</td>
</tr>
<tr>
<td>14.612</td>
<td>15.981</td>
<td>6.448</td>
</tr>
<tr>
<td>14.626</td>
<td>9.125</td>
<td>3.718</td>
</tr>
<tr>
<td>10.263</td>
<td>6.342</td>
<td>2.118</td>
</tr>
<tr>
<td>7.348</td>
<td>9.810</td>
<td>0.885</td>
</tr>
<tr>
<td>13.051</td>
<td>11.927</td>
<td>5.469</td>
</tr>
<tr>
<td>12.435</td>
<td>10.769</td>
<td>3.254</td>
</tr>
<tr>
<td>11.617</td>
<td>7.126</td>
<td>2.272</td>
</tr>
<tr>
<td>8.038</td>
<td>8.308</td>
<td>2.572</td>
</tr>
<tr>
<td>10.322</td>
<td>10.378</td>
<td>0.880</td>
</tr>
<tr>
<td>10.232</td>
<td>16.420</td>
<td>6.448</td>
</tr>
<tr>
<td>18.659</td>
<td>20.675</td>
<td>7.455</td>
</tr>
<tr>
<td>12.470</td>
<td>10.195</td>
<td>5.592</td>
</tr>
<tr>
<td>10.476</td>
<td>10.476</td>
<td>10.360</td>
</tr>
<tr>
<td>10.856</td>
<td>10.461</td>
<td>2.572</td>
</tr>
<tr>
<td>9.704</td>
<td>17.937</td>
<td>8.186</td>
</tr>
<tr>
<td>13.927</td>
<td>20.664</td>
<td>16.127</td>
</tr>
<tr>
<td>14.488</td>
<td>12.399</td>
<td>11.127</td>
</tr>
<tr>
<td>8.251</td>
<td>11.687</td>
<td>2.572</td>
</tr>
<tr>
<td>6.558</td>
<td>10.776</td>
<td>10.360</td>
</tr>
<tr>
<td>0.833</td>
<td>11.269</td>
<td>11.269</td>
</tr>
<tr>
<td>7.740</td>
<td>22.465</td>
<td>22.465</td>
</tr>
</tbody>
</table>
Fig. 4.4(a) Segment of female speech.

Fig. 4.4(b) Segment of male speech.
4.4 CODING

The ARC transmitter produces quantizer levels from 1 to 11. These source symbols need to be coded into channel alphabet for the transmission. The source coding must be such that it uses the bit rate optimally. The probabilities of the quantizer level occupancies are non-uniform. In such case, the bit rate is optimally utilized by assigning variable length code to the quantizer levels. The average code length for Huffman code, matches very well with the average entropy (Gallager, 1969). However, such simple variable length source code tended to cause the buffer to fill up rapidly during segments of voiced speech. When sophisticated variable length code, such as overfull code in PARC system [1980], is employed the effect of channel errors is severe. This is because a single bit error could lead to a string of inaccurate data. This becomes a serious disadvantage for a robust system. This leads to the development of a fixed length code.

With the compressed speech sampling rate of 3200 Hz and the bit rate of 9600 bits per second, the average number of bits per sample are 3. If simple fixed length 3 bit code words are used, only 8-level quantizer can be employed. Fewer quantizer levels cause more quantization noise, thus deteriorating the output speech quality. Therefore, 11-level quantizer was used. This requires 4-bit code word for fixed length coding. There are 16 possible code words out of which 11 are used to represent the quantizer levels and the rest is used to represent the run lengths as shown in Table 4.3. It can be noted there that a 4-bit code word can represent either one sample or two samples. Thus, either 4 bits or 2 bits per sample are transmitted as opposed to the average of 3 bits per
TABLE 4.3
Source code and a quantizer level statistics for a typical two second utterance.

<table>
<thead>
<tr>
<th>Source Alphabet</th>
<th>Codewords</th>
<th>Frequency of Occupancy</th>
<th>Probability of Occupancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0 0 0</td>
<td>516</td>
<td>0.1215</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 1 1</td>
<td>366</td>
<td>0.0862</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0 0 4</td>
<td>351</td>
<td>0.0826</td>
</tr>
<tr>
<td>4</td>
<td>0 0 1 0 2</td>
<td>375</td>
<td>0.0883</td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1 5</td>
<td>265</td>
<td>0.0624</td>
</tr>
<tr>
<td>6</td>
<td>0 0 1 1 3</td>
<td>78</td>
<td>0.0184</td>
</tr>
<tr>
<td>7</td>
<td>0 1 1 0 6</td>
<td>79</td>
<td>0.0186</td>
</tr>
<tr>
<td>8</td>
<td>1 0 1 1 11</td>
<td>5</td>
<td>0.0012</td>
</tr>
<tr>
<td>9</td>
<td>0 1 1 1 7</td>
<td>26</td>
<td>0.0061</td>
</tr>
<tr>
<td>10</td>
<td>1 1 1 1 15</td>
<td>6</td>
<td>0.0000</td>
</tr>
<tr>
<td>11</td>
<td>1 1 1 0 14</td>
<td>7</td>
<td>0.0016</td>
</tr>
<tr>
<td>1, 1</td>
<td>1 0 0 0 8</td>
<td>920</td>
<td>0.2166</td>
</tr>
<tr>
<td>2, 1</td>
<td>1 0 1 0 10</td>
<td>402</td>
<td>0.0946</td>
</tr>
<tr>
<td>2, 2</td>
<td>1 1 0 1 13</td>
<td>248</td>
<td>0.0584</td>
</tr>
<tr>
<td>3, 1</td>
<td>1 0 0 1 9</td>
<td>432</td>
<td>0.1017</td>
</tr>
<tr>
<td>3, 3</td>
<td>1 1 0 0 12</td>
<td>178</td>
<td>0.0419</td>
</tr>
</tbody>
</table>

Probability of runlength occurring - 0.5132
Probability of no runlength - 0.4868
sample. If the probability of occurrence of run length is large enough, then the average number of bits per sample will be 3.

Let the probability of occurrence of run length be \( p \). Hence \((1-p)\) is the probability of no run length occurring.

Hence

\[
\text{Average number of samples/bit} = \frac{2}{4} + (1-p)\frac{1}{4}
\]

or

\[
= \frac{1}{4} (1+p)
\]

It is required that the average number of bits per sample should not exceed three or the average number of samples per bit should be more than one third.

Hence

\[
\frac{1}{4} (1+p) > \frac{1}{3}
\]

or

\[
p > \frac{1}{3}
\]

or the probability of runlength should be greater than 0.33.

If the run lengths occur at least one third of the time, the average bit rate is maintained. The quantizer and the predictor parameters were designed such that lower level quantizer level occupancy is high enough to maintain the probability of run lengths around 0.33. Table 4.3 gives the statistics of the quantizer levels, run lengths and the code words. All the code words are 4 bits long and this prevents the propagation of
transmission error. However, samples can be added or deleted because of such error. Table 4.4 lists all the possible transitions for quantizer levels if the single bit error occurs. The third column shows the probability of the transition from run length to non-run length and vice-versa. To find out the average addition or deletion of samples at the receiver due to channel error, consider probabilities of the quantizer levels and that of the transitions given in Table 4.4.

Let ε be the bit error rate and \( p_i \) be the probability of \( i \)th quantizer level. Similarly, the run length probabilities are \( p_{1,1} \), \( p_{2,1} \), \( p_{2,2} \), \( p_{3,1} \) and \( p_{3,3} \).

The samples added at the receiver per four bits transmitted =

\[
\epsilon \frac{1}{4} \left( p_1 + p_2 + p_3 + p_4 + p_5 + p_8 + p_{10} + p_{11} \right) + \frac{p_8}{4} + \frac{p_{11}}{4} - \frac{1}{2} \left( p_{1,1} + p_{2,1} + p_{2,2} + p_{3,1} + p_{3,3} \right) + \frac{p_{1,1}}{4} + \frac{p_{2,1}}{4} - \frac{1}{2} \left( \frac{1 - p}{4} \right) + \frac{p_6}{4} + \frac{p_7}{4} + \frac{p_9}{4} + p_8 + \frac{p_{11}}{4} - \frac{p}{2} - \frac{p_{1,1}}{4} + \frac{p_{2,1}}{4} + \frac{1 - p}{2}
\]

or

\[
\epsilon \frac{1 - p}{4} - \frac{p}{2} + \frac{1}{4} \left( p_8 + p_{11} - p_{2,1} - p_6 - p_7 - p_9 + p_{1,1} \right)
\]

where

\( p \) - probability of run length

\( (1-p) \) - probability of no run length
<table>
<thead>
<tr>
<th>Quantizer Level</th>
<th>Possible quantizer Level after error</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[2, 4, 3, (1, 1)]</td>
<td>1/4</td>
</tr>
<tr>
<td>2</td>
<td>1, 6, 5, (2, 1)</td>
<td>1/4</td>
</tr>
<tr>
<td>3</td>
<td>5, 7, 1, (3, 1)</td>
<td>1/4</td>
</tr>
<tr>
<td>4</td>
<td>6, 1, 7, (1, 2)</td>
<td>1/4</td>
</tr>
<tr>
<td>5</td>
<td>3, 9, 2, (1, 3)</td>
<td>1/4</td>
</tr>
<tr>
<td>6</td>
<td>4, 2, 9, 8</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>9, 3, 4, 11</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>(1, 2), (2, 1), 10, 6</td>
<td>1/2</td>
</tr>
<tr>
<td>9</td>
<td>7, 5, 6, 10</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>11, (1, 3), 8, 9</td>
<td>1/4</td>
</tr>
<tr>
<td>11</td>
<td>10, (3, 1), (1, 2), 7</td>
<td>1/2</td>
</tr>
<tr>
<td>1, 1</td>
<td>(2, 1), (1, 2), (3, 1), 7</td>
<td>1/4</td>
</tr>
<tr>
<td>2, 1</td>
<td>8, (1, 1), 11, 4</td>
<td>3/4</td>
</tr>
<tr>
<td>2, 2</td>
<td>(3, 1), 10, (2, 1), 5</td>
<td>1/2</td>
</tr>
<tr>
<td>3, 1</td>
<td>2, 8, (1, 3), (1, 1)</td>
<td>1/2</td>
</tr>
<tr>
<td>3, 3</td>
<td>3, 11, (1, 1), (1, 3)</td>
<td>1/2</td>
</tr>
</tbody>
</table>
Consider the specific example, say the sentence 1, this number comes to 0.0006 samples for the BER of 1% or approximately 1.25 samples per second.

The bit string in this system represents only the quantizer levels. If this information is transmitted over noisy channel with BER as high as 0.1% without error protection, the output speech quality is slightly degraded. The higher BERs require error protection. (57, 63) and (26, 31) single-error-correction Hamming codes were tried. The output speech quality was found to be much better for (26, 31) single error correcting Hamming code. However, speech quality for no transmission error is sacrificed.

4.5 BUFFER CONTROL

It was noted in previous sections that the code employed in the system generates a variable number of bits per sample. This causes the buffer content to fluctuate considerably. Since the buffer is of fixed bit length, (1024 bits in this case) it is important to monitor the buffer behavior and control it if the buffer overflows or underflows. For example, in the unvoiced and silence segment of speech, more and more run lengths of quantizer levels are formed, thus generating only 2 bits per sample as against average rate of 3 bits per sample. If such situation continues for long time, the buffer may eventually run out of bits. The buffer overflow situation may occur for the voiced segment of speech where 4 bits per sample are generated as against 3 bits per sample are removed from the buffer.
The simulation of the bit buffer at the transmitter (or sample buffer at the receiver) is carried out by keeping count of the number of net bits (or samples) added to the buffer. The bit rate required for error protection is taken into account by adding \((n-m)\) parity check bits to the transmitter buffer for \(m\) information bits. The sample buffer at the receiver is not affected by the error protection bits. The following equations describe the buffer simulations implementation:

\[ b(k) = b(k-1) + r(k) - \bar{r} \]  \hspace{1cm} (4.1)

where

- \(b(k)\) = Current buffer content
- \(k\) = Time instant
- \(r(k)\) = Instantaneous bit rate
- = 4 bits/sample or 4 bits/2 samples
- \(\bar{r}\) = Average bit rate
- \(= \frac{m \times 3}{n}\) for \((m, n)\) Hamming code

For no run length

\[ b(k) = b(k-1) + 4 - \bar{r} \]  \hspace{1cm} (4.2)

For run length

\[ b(k) = b(k-1) + 4 - 2\bar{r} \]  \hspace{1cm} (4.3)

Similarly, the receiver sample buffer simulation is expressed by

\[ s(k) = s(k-1) + 0.25 - 0.333 \]  \hspace{1cm} (4.4)

or

\[ s(k) = s(k-1) + 0.50 - 0.333 \]  \hspace{1cm} (4.5)
Equation (4.4) is for "no run length" case and Eq. (4.5) expresses "run length" situation. On an average, one sample per 3 bits or 1/3 sample per bit is added to the sample buffer. All the code words are four bit long. The number of samples added to the buffer is either one or two i.e. 0.25 or 0.5 samples per bit.

As mentioned earlier, the transmission buffer tends to underflow when large number of run lengths of quantizer levels is generated. The buffer underflow can be prevented by switching the code when small number of bits are left in it. Such switching can be accomplished by not employing a run length when the buffer contents drop below a certain threshold. In this case, the buffer content is incremented by one bit for every quantizer level. The buffer underflow control can be seen in Fig. 4.5. It shows the transmitter and receiver buffer behavior for an utterance spoken by a male speaker. When the bit count reaches one hundred, switching of code occurs and the bit count remains near or above the threshold.

The buffer overflow situation usually happens in the voiced segment of speech. The probability of run lengths is small when outer quantizer levels are frequently generated. In such case, there is a net addition of a bit for every sample, eventually causing overflow. The buffer overflow should be avoided since it could cause a loss of information. To prevent such situation, a method had to be found to sharply limit the bit generation rate occasionally which did not cause an unreasonable amount of distortion. One such method could be, changing the quantizer level thresholds gradually as the buffer fills beyond certain thresholds. In the simulations, three buffer thresholds and three scalars to change the
Fig. 4.5 Transmitter and receiver buffer behaviour.
quantizer level thresholds, were specified. The quantizer changes as the buffer crosses these thresholds, is shown in Fig. 4.6. This, of course, introduces more quantization noise and hence distortion. The performance of the ARC system with buffer control is outlined in Table 4.5. It can be seen that the distortion introduced is gradual. The transmitter buffer behavior with buffer control, can be seen in Fig. 4.7. The number of bits in the buffer is limited to 1024. The clipping of the buffer at this value indicates buffer overflow situation. With the buffer control strategy described above, more lower quantizer levels are generated thus increasing the run length probability. The Table 4.6 shows the statistics of the quantizer with and without the buffer control. The frequency of occupancies of lower levels is, indeed, increased and hence the probability of the run lengths. The advantage of this scheme is its simplicity since the receiver need not know the quantizer thresholds.

4.6 TRANSMISSION ERRORS

Many toll quality speech links maintain bit-error rates (BER) which are too small (less than $10^{-5}$) to affect the quantizer and hence the coder performance. However, a BER of one tenth of a percent is not uncommon and for bad channels this rate could be as high as one percent. It is important to determine the extent of degradation and if possible how to minimize it.

The FORTRAN simulation program was written to study the effects of transmission errors. The bit manipulations required for the exact simulation of real situation is rather difficult to accomplish easily in
Fig. 4.6 The effect on the quantizer as the transmitter buffer gets full.
<table>
<thead>
<tr>
<th>Sample Number</th>
<th>ARC performance (dB)</th>
<th>Predictor Performance (dB)</th>
<th>Quantizer Performance (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Buffer Control</td>
<td>Buffer Control</td>
<td>No Buffer Control</td>
</tr>
<tr>
<td>4289-4352</td>
<td>11.06</td>
<td>11.06</td>
<td>-8.98</td>
</tr>
<tr>
<td>4353-4416</td>
<td>12.77</td>
<td>12.77</td>
<td>8.86</td>
</tr>
<tr>
<td>4417-4488</td>
<td>11.76</td>
<td>11.69</td>
<td>6.58</td>
</tr>
<tr>
<td>4489-4544</td>
<td>21.69</td>
<td>21.69</td>
<td>4.92</td>
</tr>
<tr>
<td>4609-4672</td>
<td>17.81</td>
<td>17.81</td>
<td>6.31</td>
</tr>
<tr>
<td>4801-4864</td>
<td>10.43</td>
<td>5.54</td>
<td>2.96</td>
</tr>
<tr>
<td>4865-4928</td>
<td>1.21</td>
<td>1.04</td>
<td>1.21</td>
</tr>
<tr>
<td>4929-4992</td>
<td>0.08</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>4993-5056</td>
<td>0.08</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>5057-5128</td>
<td>17.20</td>
<td>17.24</td>
<td>2.34</td>
</tr>
<tr>
<td>5121-5184</td>
<td>18.28</td>
<td>18.28</td>
<td>8.86</td>
</tr>
<tr>
<td>5185-5248</td>
<td>18.85</td>
<td>18.86</td>
<td>4.21</td>
</tr>
<tr>
<td>5313-5376</td>
<td>17.84</td>
<td>17.84</td>
<td>6.47</td>
</tr>
</tbody>
</table>
Fig. 4.7 Transmitter buffer behaviour with buffer control
TABLE 4.6
The statistics of quantizer levels with and without buffer control

<table>
<thead>
<tr>
<th>Source Alphabet</th>
<th>No buffer control</th>
<th>With buffer control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Frequency</td>
<td>Probability</td>
</tr>
<tr>
<td>1</td>
<td>516</td>
<td>.1215</td>
</tr>
<tr>
<td>2</td>
<td>351</td>
<td>.09826</td>
</tr>
<tr>
<td>3</td>
<td>375</td>
<td>.0983</td>
</tr>
<tr>
<td>4</td>
<td>340</td>
<td>.085</td>
</tr>
<tr>
<td>5</td>
<td>79</td>
<td>.0184</td>
</tr>
<tr>
<td>6</td>
<td>.0624</td>
<td>88</td>
</tr>
<tr>
<td>7</td>
<td>.0624</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>.0012</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>.0016</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>.0008</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>929</td>
<td>.2166</td>
</tr>
<tr>
<td>12</td>
<td>482</td>
<td>.0946</td>
</tr>
<tr>
<td>13</td>
<td>248</td>
<td>.0584</td>
</tr>
<tr>
<td>14</td>
<td>432</td>
<td>.1017</td>
</tr>
<tr>
<td>15</td>
<td>176</td>
<td>.0419</td>
</tr>
</tbody>
</table>
NEW APPROACH TO SPEECH DIGITIZATION COMBINING TIME-DOMAIN HARMONIC-ETC(U)
AUG 81 J L MELS; A K PANDA
DCA100-80-C-0050

END
DATE: 11 81
DTIC
FORTRAN. However, the following steps in simulation procedure do represent the real time situation very closely.

1. The encoder program reads a block of 256 quantizer levels, converts them into code words and writes them into a file. This file length is always shorter than the quantizer level file because of run lengths.

2. The channel simulation program reads the code words and asks the BER during run time. Depending on the choice of \( n \) in \((m, n)\) Hamming code, the block of code words is chosen. For example, \((57,63)\) single error-correcting Hamming code, the block of 16 code words is considered. The errors are added in the bit stream according to the bit error rate specified. The single error is corrected, the double or even number of errors are passed uncorrected and for three or more odd numbers of errors additional error is introduced. The single error corrected code words are written in a separate file. The decoder program reads this file and produces a quantizer level file which forms the input to the receiver program.

The simulation described was carried out with the parameters designed to minimize the effect of transmission errors. [Melsa, et al., 1980]. Because of addition and deletion of samples due to the channel errors the system performance cannot be reliably evaluated using objective measure criteria. Informal listening test was used to measure the performance. The output speech is distorted due to two reasons. First, due to the error in quantizer levels and second, due to wrong pitch
As mentioned in an earlier chapter, the pitch is extracted from the reconstructed compressed speech, y(k). If y(k) is changed significantly due to the channel errors, the wrong pitch might be extracted.

Table 4.8 shows the pitch extracted at the transmitter and at the receiver with and without channel errors. It also lists the pitch periods extracted after the error correction. The effect of 1% channel errors on the pitch extraction is not significant. Besides, the distortion caused due to the use of a wrong pitch period is masked by that due to the channel error.

After error correction at the receiver, pitch is extracted generally correctly and thus, pitch extraction at the receiver does not cause any severe degradation in the presence of channel noise.

In the previous section, the effects of [26,31] and [57,63] single error correcting Hamming code on the buffer behavior were discussed. The allocation of significant bit rate for error protection penalizes the error free performance because of the repeated use of the buffer overflow control strategy. However, the system performance in the presence of channel noise is greatly improved. The extent of this improvement can be determined by finding BER after using single error correcting Hamming code.

Let [m,n] be the single error correcting Hamming code and ε be the BER.

\[
\begin{align*}
P(\text{no error per frame}) &= (1-\epsilon)^n \\
P(\text{at least one error per frame}) &= 1 - (1-\epsilon)^n \\
P(\text{single error per frame}) &= n(\epsilon)(1-\epsilon)^{n-1} \\
P(\text{double error per frame}) &= \frac{n(n-1)}{1x2} (\epsilon)^2 (1-\epsilon)^{n-2}
\end{align*}
\]
<table>
<thead>
<tr>
<th>Pitch at the transmitter</th>
<th>Pitch at the receiver No channel error</th>
<th>Pitch at the receiver 1% BER</th>
<th>Pitch at the receiver 1% BER with single error correction</th>
<th>Pitch at the receiver 1% BER with double error correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>6532 - 6532 2#</td>
<td>2664 - 2563 29</td>
<td>2649 - 2548 9#</td>
<td>2676 - 2575 29</td>
<td>2641 - 2541 89</td>
</tr>
<tr>
<td>6539 - 6559 3#</td>
<td>2693 - 2592 3#</td>
<td>2729 - 2628 3#</td>
<td>2769 - 2668 3#</td>
<td>2739 - 2639 39</td>
</tr>
<tr>
<td>6561 - 6571 3#</td>
<td>2723 - 2622 3#</td>
<td>2759 - 2658 3#</td>
<td>2799 - 2699 3#</td>
<td>2769 - 2669 39</td>
</tr>
<tr>
<td>6564 - 6574 3#</td>
<td>2753 - 2652 3#</td>
<td>2789 - 2689 3#</td>
<td>2819 - 2699 3#</td>
<td>2789 - 2689 39</td>
</tr>
<tr>
<td>6566 - 6576 3#</td>
<td>2783 - 2682 3#</td>
<td>2819 - 2689 3#</td>
<td>2849 - 2699 3#</td>
<td>2849 - 2699 39</td>
</tr>
<tr>
<td>6586 - 6596 3#</td>
<td>2813 - 2712 3#</td>
<td>2849 - 2749 3#</td>
<td>2879 - 2749 3#</td>
<td>2879 - 2749 39</td>
</tr>
<tr>
<td>6761 - 6861 3#</td>
<td>2843 - 2742 3#</td>
<td>2879 - 2749 3#</td>
<td>2909 - 2799 3#</td>
<td>2909 - 2799 39</td>
</tr>
<tr>
<td>6811 - 6911 3#</td>
<td>2873 - 2772 3#</td>
<td>2909 - 2799 3#</td>
<td>2939 - 2839 3#</td>
<td>2939 - 2839 39</td>
</tr>
<tr>
<td>6911 - 7011 3#</td>
<td>2893 - 2792 3#</td>
<td>2939 - 2839 3#</td>
<td>2969 - 2879 3#</td>
<td>2969 - 2879 39</td>
</tr>
<tr>
<td>6951 - 7051 3#</td>
<td>2923 - 2822 3#</td>
<td>2969 - 2879 3#</td>
<td>2999 - 2919 3#</td>
<td>2999 - 2919 39</td>
</tr>
<tr>
<td>7091 - 7191 3#</td>
<td>2953 - 2852 3#</td>
<td>2999 - 2919 3#</td>
<td>3029 - 2929 3#</td>
<td>3029 - 2929 39</td>
</tr>
<tr>
<td>7111 - 7211 3#</td>
<td>2983 - 2882 3#</td>
<td>3029 - 2929 3#</td>
<td>3059 - 2969 3#</td>
<td>3059 - 2969 39</td>
</tr>
<tr>
<td>7171 - 7271 3#</td>
<td>3013 - 2912 3#</td>
<td>3059 - 2969 3#</td>
<td>3089 - 3019 3#</td>
<td>3089 - 3019 39</td>
</tr>
<tr>
<td>7231 - 7331 3#</td>
<td>3043 - 2942 3#</td>
<td>3089 - 3019 3#</td>
<td>3119 - 3059 3#</td>
<td>3119 - 3059 39</td>
</tr>
<tr>
<td>7291 - 7391 3#</td>
<td>3073 - 2972 3#</td>
<td>3119 - 3059 3#</td>
<td>3149 - 3099 3#</td>
<td>3149 - 3099 39</td>
</tr>
<tr>
<td>7351 - 7451 3#</td>
<td>3103 - 3002 3#</td>
<td>3149 - 3099 3#</td>
<td>3179 - 3139 3#</td>
<td>3179 - 3139 39</td>
</tr>
<tr>
<td>7411 - 7511 3#</td>
<td>3133 - 3032 3#</td>
<td>3179 - 3139 3#</td>
<td>3209 - 3179 3#</td>
<td>3209 - 3179 39</td>
</tr>
<tr>
<td>7471 - 7571 3#</td>
<td>3163 - 3062 3#</td>
<td>3209 - 3179 3#</td>
<td>3239 - 3219 3#</td>
<td>3239 - 3219 39</td>
</tr>
</tbody>
</table>
If a single error in the frames is corrected, the residual error becomes \([1-(1-\varepsilon)^n-\varepsilon(1-\varepsilon)^{n-1}]\). This error could be equivalent to the BER, \(\delta\), without any error correction. This is expressed in the following equations.

\[
1 - (1-\delta)^n = 1 - (1-\varepsilon)^{n-1} - n \varepsilon (1-\varepsilon)^{n-1}
\]

Let \(n = 31\) and \(\varepsilon = 0.01\) or BER = 1%

\[
[1 - (1-\delta)^{31}] = [1 - (0.99)^{31}] - 31(0.01)(0.99)^{30}
\]

\[
= 0.2676 - .2293 = .03839
\]

\((1-\delta)^{31} = 0.9616\)

\(31 \ln(1-\delta) = \ln 0.9616\)

\(\ln(1-\delta) = -.00126\)

\(1-\delta = 0.9987\) or \(\delta = 0.0012\)

i.e. 1% BER with single error correction using [26,31] Hamming code is equivalent to 0.12% BER without any correction. The similar calculations and the simulation results, for different BERs and Hamming codes, are outlined in Table 4.9. It can be seen that [26,31] single error correcting Hamming code is quite effective against transmission errors. While studying the Table 4.9 it should be kept in mind that the frame size used in the simulation studies, was 64 and 32 bits instead of the actual 63 and 31 bits respectively. Also, it should be noted in the simulation studies that additional errors were introduced if 3 or more odd number of errors were detected in a frame.

It was mentioned earlier that transmission errors cause addition or deletion of samples. This happens because the four-bit code word represent either one or two samples. If the channel error changes the code
Table 4.9

Male speaker, 2 second utterance, sampling frequency = 3200 Hz

<table>
<thead>
<tr>
<th>BER</th>
<th>Hamming Code</th>
<th>Theoretical BER after single error correction</th>
<th>Simulated BER after single error correction</th>
<th>Theoretical BER after double error correction</th>
<th>Simulated BER after double error correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1%</td>
<td>[57,63]</td>
<td>0.0029%</td>
<td>0.074%</td>
<td>0.097%</td>
<td>0.092%</td>
</tr>
<tr>
<td></td>
<td>[26,31]</td>
<td>0.0014%</td>
<td>0.074%</td>
<td>0.0985%</td>
<td>0.092%</td>
</tr>
<tr>
<td>1.0%</td>
<td>[57,63]</td>
<td>0.26%</td>
<td>0.502%</td>
<td>0.714%</td>
<td>0.553%</td>
</tr>
<tr>
<td></td>
<td>[26,31]</td>
<td>0.137%</td>
<td>0.307%</td>
<td>0.85%</td>
<td>0.713%</td>
</tr>
<tr>
<td>5.0%</td>
<td>[57,63]</td>
<td>2.77%</td>
<td>4.8%</td>
<td>2.15%</td>
<td>3.6%</td>
</tr>
<tr>
<td></td>
<td>[26,31]</td>
<td>1.98%</td>
<td>4.16%</td>
<td>2.42%</td>
<td>3.31%</td>
</tr>
</tbody>
</table>
word to represent two samples instead of one, as it should be, there is an addition of samples at the receiver. The deletion of sample occurs if the reverse situation happens. In the earlier discussion of buffer control, it was pointed out that the receiver buffer tends to overflow during silence and underflows during voiced region. A desirable effect of a channel error would be the addition of samples. This might cause deletion of silence or prevent underflowing of the buffer during voiced segment. For the two second utterence, the effect of channel errors on the receiver buffer behavior is not so pronounced. Therefore, the BER of 5% was considered. The receiver buffer plot for five percent bit error rate is shown in Fig. 4.8. It can be clearly seen that the net effect of channel errors is to add samples to the buffer for the code word assignment in Table 4.3.

4.7 BACKGROUND NOISE

The system performance was evaluated for the male and female utterances, spoken in quiet rooms without any background noise. So, the input to the coder is undistorted input. However, this is an unlikely situation since there will certainly be background interference such as other speakers, typewriter noises and the like, whenever a speaker is using the "digital-telephone". It was thought that the periodic background noise would be the worst kind of noise for TDHS-ARC system since the periodicity of the input signal is used for harmonic scaling operations.

The study of background noise is very simple after the algorithm has been implemented in real time. It is just a matter of talking into a handset with noise in the background. The output could be heard through headphones. However, in the FORTRAN simulation, the task is not so
Fig. 4.8 The effect of channel error on a receiver buffer.
straightforward. There is a need for a digital speech file with background noise. Multispeaker files were created by adding two digital speech files with appropriate weight.

In the simulation, the multispeaker file was generated by adding female speech to male speech as in Eq. 4.6

\[ s_{\text{composite}} = s_{11} + k \cdot s_1 \]  

(4.6)

where \( k \) takes values from 0 to 1 thus having varying degree of background noise. It was noticed that pitch extraction loop picks the pitch for the dominant speaker (see Table 4.10) at each short-time interval. Due to the masking properties of the ear, the harmonic distortion in the non-dominant speech is not heard. The ARC and the total system performance for the composite speaker is shown in Table 4.11. As long as the pitch tracking algorithm does not break down, this system should perform very well for background noise.

4.8 SUMMARY

The system presented in this chapter produces a high quality speech at the transmission rate of 9.6 kb/s. The speech quality, however, depends on the bit rate used for error protection. For example, the use of [26,31] single error correcting Hamming code results in an excellent system performance for the bit error rate of 1%. However, noise free performance is degraded because of coarse quantization and/or buffer control operations. The quantizer level runlength is effectively used to employ fix codeword size. The switching of code and changing the quantizer thresholds was found to be the effective strategy to control buffer underflow and overflow respectively. The high bit error rates do affect
### TABLE 4.10
Pitch extraction for composite speaker

<table>
<thead>
<tr>
<th>$S_{11} + 0.5$</th>
<th>$S_{11}$</th>
<th>$S_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Pitch</td>
<td>Sample Pitch</td>
<td>Sample Pitch</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>3953 - 4092</td>
<td>3939 - 4092</td>
<td>3929 - 4079</td>
</tr>
<tr>
<td>4067 - 4256</td>
<td>4079 - 4256</td>
<td>4053 - 4192</td>
</tr>
<tr>
<td>4151 - 4392</td>
<td>4179 - 4392</td>
<td>4169 - 4392</td>
</tr>
<tr>
<td>4299 - 4438</td>
<td>4329 - 4438</td>
<td>4299 - 4424</td>
</tr>
<tr>
<td>4417 - 4556</td>
<td>4433 - 4556</td>
<td>4423 - 4542</td>
</tr>
<tr>
<td>4535 - 4674</td>
<td>4545 - 4674</td>
<td>4523 - 4662</td>
</tr>
<tr>
<td>4665 - 4804</td>
<td>4681 - 4804</td>
<td>4661 - 4792</td>
</tr>
<tr>
<td>4779 - 4918</td>
<td>4871 - 4918</td>
<td>4779 - 4896</td>
</tr>
<tr>
<td>4989 - 5128</td>
<td>4995 - 5128</td>
<td>4979 - 5094</td>
</tr>
<tr>
<td>5243 - 5382</td>
<td>5253 - 5382</td>
<td>5239 - 5368</td>
</tr>
<tr>
<td>5171 - 5310</td>
<td>5171 - 5310</td>
<td>5159 - 5298</td>
</tr>
<tr>
<td>5371 - 5510</td>
<td>5379 - 5510</td>
<td>5369 - 5498</td>
</tr>
<tr>
<td>5429 - 5569</td>
<td>5439 - 5569</td>
<td>5429 - 5558</td>
</tr>
<tr>
<td>5489 - 5629</td>
<td>5499 - 5629</td>
<td>5479 - 5598</td>
</tr>
<tr>
<td>5611 - 5750</td>
<td>5629 - 5750</td>
<td>5609 - 5728</td>
</tr>
<tr>
<td>5671 - 5810</td>
<td>5689 - 5810</td>
<td>5679 - 5798</td>
</tr>
<tr>
<td>5731 - 5870</td>
<td>5749 - 5870</td>
<td>5729 - 5868</td>
</tr>
<tr>
<td>5791 - 5930</td>
<td>5799 - 5930</td>
<td>5789 - 5998</td>
</tr>
<tr>
<td>5911 - 6050</td>
<td>5929 - 6050</td>
<td>5919 - 6188</td>
</tr>
<tr>
<td>5971 - 6110</td>
<td>5999 - 6110</td>
<td>5989 - 6258</td>
</tr>
<tr>
<td>6031 - 6170</td>
<td>6049 - 6170</td>
<td>6039 - 6318</td>
</tr>
<tr>
<td>6131 - 6270</td>
<td>6139 - 6270</td>
<td>6129 - 6388</td>
</tr>
<tr>
<td>6271 - 6410</td>
<td>6279 - 6410</td>
<td>6269 - 6508</td>
</tr>
<tr>
<td>6331 - 6470</td>
<td>6339 - 6470</td>
<td>6329 - 6578</td>
</tr>
<tr>
<td>6391 - 6530</td>
<td>6391 - 6530</td>
<td>6389 - 6648</td>
</tr>
</tbody>
</table>

Note: The table continues with similar entries for each sample pair.
### TABLE 4.11

Performance of ARC and TDHS-ARC system for multispeaker files

<table>
<thead>
<tr>
<th>Sentence #</th>
<th>ARC</th>
<th>TDHS - ARC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SNR (dB)</td>
<td>SPER (dB)</td>
</tr>
<tr>
<td>$S_{11}$</td>
<td>15.29</td>
<td>5.68</td>
</tr>
<tr>
<td>$S_{11} + 0.25S_1$</td>
<td>16.82</td>
<td>6.09</td>
</tr>
<tr>
<td>$S_{11} + 0.5S_1$</td>
<td>17.08</td>
<td>6.11</td>
</tr>
<tr>
<td>$S_1$</td>
<td>13.98</td>
<td>4.60</td>
</tr>
</tbody>
</table>
the pitch extraction at the receiver. However, the distortion caused by improper pitch period was found to be masked by the distortion due to transmission error effects. The system also behaves very well for simultaneous speakers.
CHAPTER 5
THE 16 KB/S SYSTEM

5.1 INTRODUCTION

In the earlier chapters, the TDHS-ARC system for the bit rate of 9.6 kb/s was presented. The same system design can be extended to a bit rate of 16 kb/s with a few modifications. The speech quality and the robustness of the 16 kb/s system are improved with the availability of more bits per sample for coding. The speech quality improvement is achieved by reducing the quantization noise, which can be done by increasing the number of quantizer levels. The amount of error protection available determines the robustness of the system. These two basic issues concerning 16 kb/s system are discussed in detail in this chapter. The other aspects of the system design such as the system configuration, the buffer control strategy and the type of source code, remain the same.

Section 5.2 describes the source and the channel code used in this system. The choice of parameters in the ARC design and the buffer behaviour is discussed in Section 5.3. The effect of transmission errors is also discussed in this section. The results are summarized in Section 5.4.

5.2 SOURCE AND CHANNEL CODING

With the bit rate of 16 kb/s and the sampling frequency of 3.2 kHz of the compressed speech, an average of 5 bits per sample are available for coding. A 31-level quantizer and 5-bit fixed length codewords were used. The system performance was found to be excellent and no distortions could be heard in the informal listening tests. However, the above scheme
of coding is possible only if no error protection is desired. For the noisy channel, a 21-level quantizer with the 5-bit fixed wordsize variable input code was designed. Out of the possible 32 codewords, 21 codewords are used to represent the quantizer levels and the remaining are used to represent the runlengths. The lower quantizer levels (for example: 1, 2, 3, 4) form the runlengths. The code designed to represent the quantizer levels and the runlengths is shown in Table 5.1. The Hamming distance between the codewords representing adjacent quantizer levels is kept to be minimum possible. With the choice of codewords as shown in Table 5.1, the buffer empties by 2.5 bits per sample, everytime the runlength occurs while there is no net addition to the buffer otherwise. If the error protection is used, the check bits are added to the buffer every certain number of the information bits. This may cause a buffer overflow depending on the amount of error protection used. Such situations may be avoided by adjusting the probability of occurrence of runlengths.

If the probability of occurrence of runlength is $p$, then $(1-p)$ becomes the probability of no runlength occurring. Let $[m,n]$ be the Hamming code used, where $m$ are the information bits and $(n-m)$ are the check bits. The bit rate available to code the quantizer levels becomes $16 \cdot m/n$ kb/s and the average number of bits per sample is $5 \cdot m/n$. With the above probabilities,

\[
\text{the average number of samples/bits} = \frac{p \cdot 2}{5} + \frac{(1-p)}{5}
\]

or

\[
= \frac{1}{5}(1+p)
\]

It is required that the average bits per sample be equal or less than $5m/n$ or the average number of samples per bit be equal or more than $n/5m$. 
<table>
<thead>
<tr>
<th>Source Alphabet</th>
<th>Codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00000</td>
</tr>
<tr>
<td>2</td>
<td>10000</td>
</tr>
<tr>
<td>3</td>
<td>00001</td>
</tr>
<tr>
<td>4</td>
<td>10001</td>
</tr>
<tr>
<td>5</td>
<td>00011</td>
</tr>
<tr>
<td>6</td>
<td>10011</td>
</tr>
<tr>
<td>7</td>
<td>00010</td>
</tr>
<tr>
<td>8</td>
<td>10010</td>
</tr>
<tr>
<td>9</td>
<td>00110</td>
</tr>
<tr>
<td>10</td>
<td>10110</td>
</tr>
<tr>
<td>11</td>
<td>00111</td>
</tr>
<tr>
<td>12</td>
<td>10111</td>
</tr>
<tr>
<td>13</td>
<td>00101</td>
</tr>
<tr>
<td>14</td>
<td>10101</td>
</tr>
<tr>
<td>15</td>
<td>00100</td>
</tr>
<tr>
<td>16</td>
<td>10100</td>
</tr>
<tr>
<td>17</td>
<td>01100</td>
</tr>
<tr>
<td>18</td>
<td>11100</td>
</tr>
<tr>
<td>19</td>
<td>01101</td>
</tr>
<tr>
<td>20</td>
<td>11101</td>
</tr>
<tr>
<td>21</td>
<td>01111</td>
</tr>
<tr>
<td>1,1</td>
<td>01000</td>
</tr>
<tr>
<td>1,2</td>
<td>11000</td>
</tr>
<tr>
<td>1,3</td>
<td>01001</td>
</tr>
<tr>
<td>2,1</td>
<td>11001</td>
</tr>
<tr>
<td>2,2</td>
<td>11011</td>
</tr>
<tr>
<td>2,3</td>
<td>11010</td>
</tr>
<tr>
<td>3,1</td>
<td>01011</td>
</tr>
<tr>
<td>3,2</td>
<td>01010</td>
</tr>
<tr>
<td>3,3</td>
<td>01110</td>
</tr>
<tr>
<td>4,4</td>
<td>11110</td>
</tr>
<tr>
<td>5,5</td>
<td>11111</td>
</tr>
</tbody>
</table>
Hence

$$\frac{1}{2}(1+p) > \frac{n}{5m} \quad (5.1)$$

or

$$p > \frac{n-\frac{n}{m}}{m} \quad (5.2)$$

For the Hamming code of [11,15] and [26,31] this probability should be equal to or greater than 0.36 and 0.19 respectively. That means, 36% or 19% of the time runlengths should occur to maintain the average number of bits per sample. The quantizer and the parameters in the ARC system were designed to satisfy Eq. (5.2). Table 5.2 shows the statistics and the parameters used for two different Hamming codes. The block-to-block SNR for ARC is plotted for both the parameter sets in Fig. 5.1. It can be observed from this figure that the heavy error protection leads to more quantization noise since less bit rate is available for the quantizer level coding.

It was discussed for the 9.6 kb/s system that the compromise between the system robustness and the speech quality must be made. It is desired in this project to have a good system performance for the bit-error-rate (BER) of 1%. If [11,15] single error correcting Hamming code is used and the computations as in Section 4.6 are carried out, 1% BER with a single error correlation becomes equivalent to 0.06% BER, with no error correction. For [26,31] Hamming code 1% BER becomes equivalent to 0.12% BER with no error correction. In the simulation studies it was found that the effect of BER of 0.06% was hardly noticeable.

5.3 BUFFER BEHAVIOUR

There are two buffers in the system. The transmitter buffer which is a bit buffer and the receiver sample buffer. The simulation of these buffers
TABLE 5.2


<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AINV = .7, ALP = .93, ALAD = .95, G = .018, N = 4</td>
<td>AINV = .7, ALP = .93, ALAD = .95, G = .018, N = 4</td>
</tr>
<tr>
<td>RMSMIN = 70, SMIN = .28</td>
<td>RMSMIN = 30, SMIN = .28</td>
</tr>
<tr>
<td>Number of quantizer levels = 21</td>
<td>Number of quantizer levels = 21</td>
</tr>
<tr>
<td>Expansion factors: .5, .6, .7, .8, 1.0, 1.0</td>
<td>Expansion factors: .5, .6, .7, .8, 1.0, 1.0</td>
</tr>
<tr>
<td>1.2, 1.4, 2, 4, 6</td>
<td>1.2, 1.4, 2, 4, 6</td>
</tr>
<tr>
<td>Output levels: 0, 0.6, 0.8, 1.1, 1.5, 2, 3, 5, 8, 12, 18</td>
<td>Output levels: 0, .4, .6, .8, 1.0, 1.4, 1.8, 2.5, 5, 8, 12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Alphabets</th>
<th>Probability</th>
<th>Frequency</th>
<th>Source Alphabets</th>
<th>Probability</th>
<th>Frequency</th>
<th>Source Alphabets</th>
<th>Probability</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.0626</td>
<td>296</td>
<td>17</td>
<td>.0051</td>
<td>24</td>
<td>1</td>
<td>.0611</td>
<td>326</td>
</tr>
<tr>
<td>2</td>
<td>.0389</td>
<td>184</td>
<td>18</td>
<td>.0002</td>
<td>1</td>
<td>2</td>
<td>.0504</td>
<td>269</td>
</tr>
<tr>
<td>3</td>
<td>.0395</td>
<td>187</td>
<td>19</td>
<td>.0011</td>
<td>5</td>
<td>3</td>
<td>.0414</td>
<td>221</td>
</tr>
<tr>
<td>4</td>
<td>.0507</td>
<td>240</td>
<td>20</td>
<td>.0000</td>
<td>0</td>
<td>4</td>
<td>.0523</td>
<td>279</td>
</tr>
<tr>
<td>5</td>
<td>.0567</td>
<td>268</td>
<td>21</td>
<td>.0000</td>
<td>0</td>
<td>5</td>
<td>.0515</td>
<td>275</td>
</tr>
<tr>
<td>6</td>
<td>.0615</td>
<td>291</td>
<td>1.1</td>
<td>.1584</td>
<td>769</td>
<td>6</td>
<td>.0484</td>
<td>258</td>
</tr>
<tr>
<td>7</td>
<td>.0647</td>
<td>306</td>
<td>1.2</td>
<td>.0336</td>
<td>159</td>
<td>7</td>
<td>.0517</td>
<td>276</td>
</tr>
<tr>
<td>8</td>
<td>.0543</td>
<td>257</td>
<td>1.3</td>
<td>.0277</td>
<td>131</td>
<td>8</td>
<td>.0583</td>
<td>311</td>
</tr>
<tr>
<td>9</td>
<td>.0524</td>
<td>248</td>
<td>2.1</td>
<td>.0296</td>
<td>140</td>
<td>9</td>
<td>.0684</td>
<td>365</td>
</tr>
<tr>
<td>10</td>
<td>.0444</td>
<td>210</td>
<td>2.2</td>
<td>.0152</td>
<td>72</td>
<td>10</td>
<td>.0544</td>
<td>290</td>
</tr>
<tr>
<td>11</td>
<td>.0372</td>
<td>176</td>
<td>2.3</td>
<td>.0142</td>
<td>67</td>
<td>11</td>
<td>.0549</td>
<td>293</td>
</tr>
<tr>
<td>12</td>
<td>.0319</td>
<td>151</td>
<td>3.1</td>
<td>.0346</td>
<td>161</td>
<td>12</td>
<td>.0534</td>
<td>285</td>
</tr>
<tr>
<td>13</td>
<td>.0262</td>
<td>124</td>
<td>3.2</td>
<td>.0127</td>
<td>60</td>
<td>13</td>
<td>.0435</td>
<td>232</td>
</tr>
<tr>
<td>14</td>
<td>.0104</td>
<td>49</td>
<td>3.3</td>
<td>.0178</td>
<td>84</td>
<td>14</td>
<td>.0452</td>
<td>241</td>
</tr>
<tr>
<td>15</td>
<td>.0097</td>
<td>46</td>
<td>4.4</td>
<td>.0044</td>
<td>21</td>
<td>15</td>
<td>.0422</td>
<td>225</td>
</tr>
<tr>
<td>16</td>
<td>.0000</td>
<td>0</td>
<td>5.5</td>
<td>.0049</td>
<td>23</td>
<td>16</td>
<td>.0064</td>
<td>34</td>
</tr>
</tbody>
</table>
Fig. 5.1 The ARC performance for a male speaker using [11,15] and [26,31] Hamming Code.
is carried out the similar way as in 9.6-kb system. For no runlength case, 5 bits are added to the buffer while the same number of bits are taken out of the buffer. Thus, there is no net addition to the buffer. For the runlength (of two samples) case, 5 bits are added as against 10 bits are taken out of the buffer. In addition to this \((n-m)\) checkbits are added to the buffer for every \(m\) information bits.

The decoder at the receiver decodes the frame of bits, makes the corrections if any, and puts the samples in the receiver buffer. For 16-kb system, the average number of samples per bit taken out of the receiver buffer is \(1/5\). If \([11,15]\) Hamming code was used, for every frame of 15 bits 3 samples are taken out and the number of samples varying from 2.2 to 4.4 are added to the buffer. To work with the integer number of samples, the receiver buffer calculations are done every 75 bits or every 4.67 msec for \([11,15]\) code and every 155 bits (=9.67 msec) for \([26,31]\) code. The transmitter and the receiver buffer plots are shown in Fig. 5.2(a) and (b) for both the codes. For the same set of parameters the transmitter buffer fills faster for \([11,15]\) code than for \([26,31]\) code thus requiring buffer control more frequently.

5.4 SUMMARY

A speech coder was developed for transmission of speech at the bit rate of 16 kb/s using time domain harmonic scaling and adaptive residual coding. The system configuration is the same as 9.6 kb/s system. It was decided to avoid pitch transmission since it retains the simplicity of the system. Besides, the distortion introduced by extracting the pitch information at the receiver, is masked by the quantization noise. The bit rate, which would have been wasted in transmitting and error protecting
Fig. 5.2(a) Transmitter and receiver buffer plots for a male speaker using [26,31] Hamming Code
the pitch information, could be employed to make the system more robust to channel noise. Thus, the use of [11,15] Hamming code was possible. A segmental SNR of more than 20 dB was achieved using 21-level quantizer. The variable input code of 5-bit fixed wordsize was found to be useful since the transmission error effects do not get magnified.

An attempt is made to keep the structure of the 16 kb/s system as simple as 9.6 kb/s system. No side information is transmitted, hence there is no blocking of data or block synchronization problem. The frequency compression and expansion operations by a factor of two are carried out using a triangular window. The hardware implementation of the system should be the same as a 9.6 kb/s system except for the quantizer design and the codeword size.
CHAPTER 6
SUMMARY

A speech coder was developed using a new approach of combining frequency scaling in time domain and adaptive residual coding. The computer simulations of the system were kept very close to real situation. The speech quality was found to be excellent. The system's overall performance could not be measured satisfactorily using existing objective performance measure criterion. In such cases, the output speech quality was evaluated by using informal listening tests. However, there are various objective measure criteria as listed in Chapter 3 to evaluate the performance of DPCM coders. Various block-to-block SNR plots indicate that the adaptive quantizer and the predictor perform very well. Such excellent performance of the ARC system is possible because of smooth varying frequency compressed input signal.

This system is designed for the bit error rate of 1%. However, it can easily withstand bit error rates higher than 5%. In such a severe channel condition, the output speech is considerably distorted but the algorithm does not diverge. In the simulation studies, it was noticed that a compromise must be reached between robustness and transmission error free coder performance. With the telephone modem assuring bit error rates less than $10^{-3}$, output speech quality is very close to toll quality. The effect of channel error is reduced, particularly in this system, because of averaging operations performed at the receiver to get expanded speech. The transmission error effects do not get magnified because of the fixed word size of the codeword. In many coders for bit rate of 9.6 kb/s, entropy coding is used. Variable length codewords, such as Huffman code,
though optimally utilize the available bit rate, are very inefficient if the channel error occurs. One bit error could cause a string of wrong code words to be decoded.

Another coder robustness indicator is its performance in background noise. In real situation, the talker is often talking in the presence of typewriter noise or background conversation. This coder performs very well for the multispeaker case. The waveform coders generally have this advantage over frequency domain speakers. It was observed in our simulation studies that various tones pass through this system with slight or no distortion. This excellent performance for tones is due to the fact that harmonic scaling operations do not introduce any distortion for perfectly periodic signals.
APPENDIX A
FOURIER TRANSFORM OF WINDOW FUNCTIONS

Triangular window

\[
\begin{align*}
\text{w}(t) &= A \left[1 - \frac{|t|}{T_0/2}\right] \\
&= 0 \quad |t| \leq T_0/2 \\
&= 0 \quad \text{Otherwise}
\end{align*}
\]

\[
W(f) = \frac{AT_0}{2} \left(\frac{\sin(\pi T_0 f)}{\pi T_0 f}\right)^2
\]

Cosine window

\[
\begin{align*}
\text{w}(t) &= A \cos \left(\frac{\pi t}{T_0}\right) \\
&= 0 \quad |t| \leq \frac{T_0}{2} \\
&= 0 \quad \text{otherwise}
\end{align*}
\]

\[
W(f) = \frac{2AT_0}{\pi} \left(\frac{\cos \pi T_0 f}{1 - (2 \pi T_0 f)^2}\right)
\]
Hanning window

\[ w(t) = \begin{cases} \frac{A}{2} \left[ 1 + \cos\left(\frac{2\pi t}{T_0}\right) \right] & \text{for } |t| \leq \frac{T_0}{2} \\ 0 & \text{otherwise} \end{cases} \]

\[ W(f) = \frac{AT_0}{2} \frac{\sin(\pi f T_0)}{(\pi f T_0)(1-(\pi f T_0)^2)} \]

Hamming window

\[ w(t) = \frac{0.54+0.46\cos(2\pi t/T_0)}{1} \quad \text{for } |t| \leq \frac{T_0}{2} \]

\[ W(f) = \frac{0.54\pi^2-0.08(\pi f T_0)^2}{\pi f T_0(\pi^2-(\pi f T_0)^2)} \sin(\pi f T_0) \]

Paroulsis window

\[ w(t) = \frac{1}{\pi} \left| \sin\left(\frac{2\pi t}{T_0}\right) + \left(1-\frac{|t|}{T_0/2}\right)\cos\left(\frac{2\pi t}{T_0}\right) \right| \quad \text{for } |t| \leq \frac{T_0}{2} \]

\[ W(f) = 2\pi^2 T_0 \frac{1+\cos(\pi f T_0)}{((\pi f T_0)^2-\pi^2)^2} \]
Tukey window

\[
W(t) = \begin{cases} 
\frac{A}{2} \left[ 1 - \cos \left( \frac{2\pi(t + T_0/2)}{T_0/2} \right) \right], & -\frac{T_0}{2} \leq t \leq -\frac{T_0}{4} \\
A, & -\frac{T_0}{4} \leq t \leq \frac{T_0}{4} \\
\frac{A}{2} \left[ 1 - \cos \left( \frac{2\pi(T_0/2 - t)}{T_0/2} \right) \right], & \frac{T_0}{4} \leq t \leq \frac{T_0}{2} \\
0, & \text{otherwise}
\end{cases}
\]

\[
W(f) = \frac{T_0}{4} \frac{\sin \left( \frac{2\pi f T_0}{4} \right)}{2\pi f T_0} + \frac{T_0}{2} \frac{\sin \left( \frac{2\pi f T_0}{2} \right)}{2\pi f T_0} \\
- \cos \left( \frac{2\pi f T_0}{8} \right) \sin \left( \frac{2\pi f T_0}{8} \right) \cdot \frac{2(2\pi f)}{((2\pi f)^2 - \left( \frac{2\pi}{T_0/2} \right)^2)}
\]
APPENDIX B
FLOW CHARTS

Transmitter

Start

Read original speech samples s(k)

Extract pitch Np and write into PITCH.DAT file

Compute the window function h(n,Np)

Perform frequency compression. y(k)

Is Buffer content more than threshold?

Yes

change the quantizer thresholds appropriately

Perform quantization and write quantizer levels in QUANT.DAT file

No

Are Np Samples done?

Yes

Are all Samples done?

Yes

Stop

No

No

No
Encoder

Start

Read the quantizer levels, q(k)

Does q(k) and q(k+1) form runlength?

yes

Look up the table to select codeword for runlength.

no

Look up the table to select codeword for single level.

4 bits are added & 6 bits are subtracted from buffer.

4 bits are added & 3 bits are subtracted from the buffer.

Write codeword in a CODE.DAT file.

Write buffer content in a TBUFF.DAT file.

Are all quantizer levels read?

no

yes

stop
Channel Error

Start

Read the bit-error-rate.
BER

Read the Hamming Code.
H(m,n)

Read options regarding error
correction.

Read code-
words.

no

is
BER more than 0%
yes

Add uniformly distributed
errors in the bit stream of
frame size N.

Keep count of single errors,
even errors and odd errors.

is
single err. correction
reqd.
yes
Remove if its a
single error. Neg-
lect errors are
even. Add a error
if # of errors are
odd.

no

is
double error correc-
tion reqd.

no

Neglect single err.
Correct double err.
Add extra error if
# of errors per
frame are even(more
than 2).

yes

Write codewords in ERCO.DAT

Are all codewords read

Stop
Read the received codewords

Look up the table for decoding.

Does the codeword generate 2 quantizer levels?

- Yes: Write 2 levels in DECO.DAT file
  - Add 0.5 sample to buffer and subtract 0.33 samples from buffer

- No: Write one level in DECO.DAT file.
  - Add 0.25 samples to the buffer and subtract 0.33 samples from the buffer.
  - Write buffer content in RBUFF.DAT file.

Are all codewords read?

- Yes: Stop
- No: Repeat
Receiver

Start

Read decoded quantizer levels \( q'(k) \)

Inverse quantize to get reconstructed compressed speech \( \hat{y}(k) \). Write into ZHAT.DAT file.

Read \( \hat{y}(k) \)

Extract pitch \((Np)\) and write into PITCH.DAT file

Compute window function. \( h(n;2Np) \)

Perform frequency expansion. \( \hat{s}(k) \)

Write output speech \( \hat{s}(k) \) into SHAT.DAT file.

Are \( 2Np \) samples done?

Are all \( \hat{y}(k) \)'s read?

Stop
APPENDIX C
USER'S GUIDE AND SOURCE LISTINGS

The system software developed for the speech coder for the bit of 9.6 kb/s, consists of six program modules. They are as follows.

1. **Options (OPTION.FTN):** This program asks for various options required for transmitter and receiver programs and create OPT1.DAT and OPT2.DAT files to be used by the transmitter and the receiver program respectively.

2. **Transmitter (TRAN.FTN):** This combines the frequency compression and the quantization using Adaptive Residual Coder. The performance and the statistics are printed on the unit. The program asks for headers to various files and produces the quantizer level file (QUANT.DAT).

3. **Encoder (ENCO.FTN):** This program reads the quantizer levels and generates the codewords which are written into CODE.DAT file. The transmitter buffer (TBUFF.DAT) is also simulated.

4. **Channel Error (CHER5763.FTN and CHER2631.FTN):** There are two modules to simulate noisy channel with the error correction simulation incorporated in them. CHER5763.FTN uses [57,63] Hamming Code and the other module uses [26,31] Hamming Code. These programs ask for the percentage BER, the type of error correction used and output the number of errors per frame. The residual errors and the BER after error correction are also displayed on the terminal.

5. **Decoder (DECO.FTN):** Reads the corrupted and/or corrected code words from ERCO.DAT file and decodes them and puts them into DECO.DAT file. The number of samples added or deleted due to channel error can be found by
noting the number of samples in the tail (which is displayed on the terminal) with and without error. Note that this number is modulo 256. The receiver buffer (RBUFF·DAT) is also simulated in this program.

6. **Receiver (RCVR·FTN):** reads the received quantizer levels, performs inverse quantization, (ZHAT·DAT) does frequency expansion and writes the speech output in the SHAT·DAT file. This program brings all the quantizer levels in the memory simultaneously. Hence, this program will give error for sentences longer than 15600 samples. The longer sentences could be processed by increasing the Q buffer array size.

To run these modules Indirect Command file is written. The task files of the above modules and the ARC parameter file should exist in the same UIC. If the task files are not available, following switches should be used to build them.

```
>ACTFIL = 10
>UNITS = 10
>ASG = SY:6
```

The indirect command file asks options regarding the various files to be retained for further use or to be deleted.
INDIRECT COMMAND FILE FOR
9.6 KBS TDHS-ARC SYSTEM

DATE: JUNE 30, 1981
NAME: ARUN K. PANDE

THIS PROGRAM CONSISTS OF EXECUTION OF FIVE MODULES. THESE MODULES ARE
1. TRANSMITTER TRAN.FTN
2. ENCODER ENCO.FTN
3. CHANNEL (ERROR CORR) CHER5763.FTN
4. DECODER DECO.FTN
5. RECEIVER RCVR.FTN

TO RUN ABOVE INDIRECT COMMAND FILE, IT IS ASSUMED THAT THE TASKS OF
ABOVE MODULES EXISTS IN THE SAME UIC... IF THEY DONOT, BUILD THE TASKS
USING FOLLOWING SWITCHES COMMON TO ALL MODULES.

>ACTFIL=18
>UNITS=18
>ASG=DKI:6

.ENABLE SUBSTITUTION
.SETS S1 "QUANT.DAT;*,CODE.DAT;*,ERCO.DAT;*,DECO.DAT;*,ZHAT.DAT;**
.SETS S2 "FOR06.DAT;**
.SETS S3 "TBUF.DAT;*,RBUF.DAT;**

RUN OPTION
.WAIT

RUN TRAN
.ASK DOPT1 DO YOU WANT TO DELETE TRAN.MITTER OPTION FILE
.IFF DOPT1 .GOTO 18
.WAIT PIP
PIP OPT1.DAT;*/DE

18:
.WAIT
RUN ENCO

28:
.ASK A1 DO YOU WANT TO USE [57,63] HAMMING CODE
.IFF A1 .GOTO 38
.ASK A2 DO YOU WANT TO USE [26,31] HAMMING CODE
.IFF A2 .GOTO 48
.GOTO 28

38:
.WAIT
RUN CHER5763
.GOTO 50
.WAIT
RUN CHER2631
.WAIT
RUN DECO
.WAIT
RUN RCVR
.WAIT PIP
PIP 'S1'/DE
.ASK CHEK DO YOU WANT TO KEEP STATISTICS AND PITCH DATA
.IFF CHEK .GOTO 68
.WAIT PIP
PIP 'S2'/DE
.WAIT

68:
.ASK CHEKB DO YOU WANT TO KEEP TRAN AND RCVR BUFFER FILES
.IFF CHEKB .GOTO 78
.WAIT PIP
PIP 'S3'/DE
.WAIT

78:
.DELAY 11S
.WAIT PIP
PIP RECSAM.DAT;*,NEWBLO.DAT;*/DE
THIS PROGRAM CREATES THE FILES FOR OPTIONS REQUIRE IN TRANSMITTER AND RECEIVER PROGRAMS.

INTEGER*2 FNAME(16), IBT(3)
DIMENSION GAMA(3)

OPEN(UNIT=1, TYPE='NEW', NAME='OPT1.DAT', CARRIAGECONTROL='LIST')

C

TYPE *, 'ENTER THE ORIGINAL SPEECH FILENAME:( < 32 CHAR )';
ACCEPT 1#, FNAME
WRITE(1,*) FNAME

C

TYPE *, 'ENTER CLIPPING LEVEL FOR PITCH EXTRACTION.'
TYPE *, '( 1.8= 18% CLIPPING; 5.5= NO CLIPPING)'
TYPE *, ' TYPICAL VALUE: 5.3 TO 5.6'
ACCEPT *, CLPP
WRITE(1,*) CLPP

C

TYPE *, 'ENTER BLOCK LENGTH (KBLK); SEARCHING RANGE (ITMIN, ITMAX)' 
TYPE *, ' FOR EXAMPLE: KBLK=80, ITMIN=2#, ITMAX=18'
ACCEPT *, KBLK, ITMIN, ITMAX
WRITE(1,*) KBLK, ITMIN, ITMAX

C

TYPE *, 'ENTER ARC PARAMETER FILENAME: ( < 32 CHAR )'
ACCEPT 1#, FNAME
WRITE(1,*) FNAME

C

TYPE *, 'BLOCKLENGTH IN ARC TO CALCULATE SEGSNR, TYPICAL: 64'
ACCEPT *, NBL
WRITE(1,*) NBL

C

TYPE *, 'SELECT THE TYPE OF PITCH ESTIMATOR:'
TYPE *, ' 1 = AUTOCORRELATION; 2 = AMDF; 3 = CLIPPED SP AUTO'
TYPE *, ' 4 = CLIPPED SP AMDF; 5 = 3-VALUE C CLIPPED AUTOCORRELATION'
TYPE *, ' 6 = 3-VALUE C CLIPPED SPEECH AMDF; 7 = 2-VALUE C CLIPPED'
TYPE *, ' SPEECH AUTOCORRELATION; 8 = 2-VALUE C CLIPPED AMDF'
ACCEPT *, IOPT
WRITE(1,*) IOPT

C

TYPE *, 'DO YOU WANT TO HAVE A BUFFER CONTROL? ( 1=YES; 2=NO )'
ACCEPT *, IBCNT
WRITE(1,*) IBCNT

C

IF(IBCNT .NE. 1) GOTO 2#

C

TYPE *, 'ENTER BUFFER_THRESHOLDS FOR BUFFER CONTROL:'
TYPE *, ' FOR EXAMPLE: [ 600, 800, 900 ]'
ACCEPT *, IBT
WRITE(1,*) IBT

C

TYPE *, 'ENTER THE SCALARS TO CHANGE THE QUANTIZER_THRESHOLDS'
TYPE *, ' FOR EXAMPLE: [ 0.5, 0.7, 1.0 ]'
ACCEPT *, GAMA
WRITE(1,*) GAMA

C

CONTINUE

C

TYPE *, 'ENTER HAMMING ERROR CORRECTING CODE:'
TYPE *, ' ( INFO BITS, TOTAL NUMBER OF BITS ); FOR EXAMPLE:(67,63)'
ACCEPT *, INFO, NTOT
WRITE(1,*) INFO, NTOT
OPTIONS FOR THE RECEIVER PROGRAM.

OPEN(UNIT=2,NAME='OPT2.DAT',TYPE='NEW',CARRIAGECONTROL='LIST')

WRITE(2,10)FNAME

TYPE *, 'ENTER THE CLIPPING LEVEL FOR PITCH EXTRACTION AT RECEIVER:
TYPE *, ( 1.0 - 180% CLIPPING; 0.0 - NO CLIPPING )'
TYPE *, 'TYPICAL VALUE= 0.3 TO 0.6'
ACCEPT *, CLP
WRITE(2,*) CLP

TYPE *, 'SELECT THE TYPE OF PITCH ESTIMATOR:
TYPE *, ( 1 = AUTOCORR; 2 = AMDF; 3 = CLPPD SP AUTOCORR
TYPE *, 4 = CLPPD SP AMDF; 5 = 3-VALUE CENTER CLIPPED AUTOCORR
TYPE *, 6 = 3-VALUE CENTER CLIPPED AMDF
TYPE *, 7 = 2-VALUE CENTER CLIPPED AUTOCORR
ACCEPT *, I OPT1
WRITE(2,*) I OPT1

TYPE *, 'ENTER BLOCK LENGTH (KBLK), SEARCHING RANGE (ITMIN, ITMAX):
TYPE *, 'TYPICAL VALUES: 45, 25, 115'
ACCEPT *, KBLK, ITMIN, ITMAX
WRITE(2,*) KBLK, ITMIN, ITMAX

STOP
END
This is a new approach to speech digitization at medium band bit rates of 9.6 to 16 KB/s. The technique is based on a combination of time-domain harmonic scaling (TDHS) and adaptive residual coder (ARC).

TDHS algorithm consists of properly weighting several adjacent input signal segment of pitch dependent duration by suitable window functions. As a result of this, the number of samples to be transmitted can be reduced by a factor of two. If the bit rate is kept the same, the number of bits allowed per sample is doubled, and the performance of the coder can be improved significantly. With the more slowly varying frequency divided signal as input, the prediction and associated quantization in the ARC system will perform better, thus increasing the performance of the system.

Program Name: TRAN.FTN
Date: JUNE 30, 1981

This program reads following options:
1. Block length (KBLK), searching range (ITMIN, ITMAX)
2. Name of the speech file.
3. Name of the parameter file.
4. Headers for quantizer output
5. Option regarding pitch method to be used.
6. Block length NBL for SEGSR.
7. Clipping level to generate clipped speech
8. Buffer control options.
9. Hamming code to be used.

The program reads following options:

INTEGER HD(45), IPICH(450), FNAME(16), SQ, Q, FILEN
INTEGER*2 SPEECH(455), FBCT, OLDQ
DIMENSION Y(256), H(455), SQ(16), IBUFF1(512), IBUFF2(512)
INTEGER*2 FL, F2, F3, F4, STAT
COMMON /PRED/G, ND, RMSMIN, A(12), DVMAT(12), EV,
ISTAT1(45), EP, ALAD, SPERB1(256), SPERB2(256), V(12), SNRB(256)
COMMON /RMS/RMS, NBL, IARG, ENGY1, ENGY2, ENGY3, ENGY4, SPER1, SPER2
COMMON /CLIPP/CLIP
COMMON /ADDN/ FI, F2, F3, F4, BT1, BT2, BT3
1 , JCT, STAT(30), GAMA(4)
COMMON /FN/FILEN(16)

C
 INTEGER MODN(K) = K - (K-1)/16 * 16
 C
OPEN (UNIT=3, TYPE='OLD', NAME='OPT1.DAT') IOPEN the option file.
 C
OPEN THE SPEECH FILE: READ THE HEADER AND THE SPEECH.
 C
READ(3,10) FNAME
10 FORMAT(16A2)
OPEN (UNIT=1, TYPE='OLD', READONLY, NAME=FNAME, SHARED) I OPEN SPEECH FILE
READ(1,20) NSENT, RATE, NSAMP, IBUFF, ILOWR, NTERMS, HD
20 FORMAT(615, 1X, 4BA1)
CONTINUE

CREATE NEW FILE FOR QUANTIZER OUTPUT. WRITE
THE HEADER ON QUANTIZER FILE.

OPEN(UNIT=2, TYPE='NEW', NAME='QUANT.DAT', CARRIAGECONTROL='LIST')
TYPE='.', TYPE THE HEADER FOR THE QUANTIZER FILE:
ACCEPT 50, HD

FORMAT(48A1)
NSPSAM = NSAMP/2
WRITE(2,20)NSAMP, NSPSAM, IUPPR, ILLOWR, INTERMS, HD

READ(3*)ICLP

DEFINE AND INITIALIZE VARIOUS PARAMETERS.

NUMP = 0
READ(3,*)KBLK1, ITMIN, ITMAX
IDFF = 0
ICNT = 0
BUFCNT=0.
J3 = 1
NADD = 0
F1 = 0
F2 = 0
F3 = 0
F4 = 0
JCNT = 1
GAMA(1)=.2
NUMI = KBLK1 + ITMAX
READ(3,10)Filen
FILEN(32)=0
CALL INSTRT

READ(3,*)IOPT2
READ(3,*)FBCNT
IF(FBCNT .NE. 1)GOTO 155
READ(3,*)BT1, BT2, BT3
READ(3,*)GAMA1, GAMA2, GAMA3
GAMA(2)=GAMA1
GAMA(3)=GAMA2
GAMA(4)=GAMA3
GOTO 158
GAMA(2)=1.8
GAMA(3)=1.8
GAMA(4)=1.8
BT1=3000.0
BT2=4500.
BT3=6000.
GOTO 158

CONTINUE
READ(3,*)NUMQ1, NOPAR
NPARI=NOPAR-NUMQ1
AVGBIT=(FLOAT(NUMQ1)/FLOAT(NPARI))*3.0

TRANSFER MAXIMUM LAG (ITMAX) PLUS BLOCKSIZE (KBLK1) SPEECH SAMPLES
TO BUFFER.

CHECK IF NUM1 IS EXACT MULTIPLE OF 16.
IF(MOD(NUM1,16).EQ.0)GOTO 55
IF NOT, MAKE IT EXACT MULTIPLE OF 16.
NUM1=(NUM1/16)*16+16

READ(1,30)(*BUFF1(I), I=1,NUM1)
ICNT = ICNT + NUM1

EXTRACT PITCH FROM SAMPLES IN BUFFER1.

CALL PITCH(IBUFF1, NP, ITMAX, ITMIN, KBLK1, NUMP, NUM1, IOPT2)

IPICH(JJ) = NP
NP2 = NP + NP

56  CONTINUE

------ CHECK IF NUMBER OF SAMPLES IN BUFFER1 IS LESS THAN TWO TIMES PITCH
------ PERIOD SAMPLES; IF LESS, ADD TO IT FROM SPEECH BUFFER AND THEN
------ PROCEED TO FREQUENCY DIVISION.

IF(NUM1 .LT. NP2)GOTO 61

C COMPUTE HOW MANY EXTRA SAMPLES IBUFF1 HAS.

NUM12 = NUM1 - NP2

IS NUM1 EXACTLY EQUAL TO NP2?

IF(NUM12 .EQ. 0)GOTO 73

C IF NOT, TRANSFER EXTRA SAMPLES FROM IBUFF1 TO IBUFF2.

DO 58  I = 1, NUM12
   IBUFF2(I) = IBUFF1(NP2 + I)
58  CONTINUE

GOTO 73

C COMPUTE HOW MANY MORE SAMPLES ARE NEEDED IN IBUFF1 TO MAKE THE
C NUMBER EXACTLY EQUAL TO 2*NP.

MORE = NP2 - NUM1

IS MORE EXACT MULTIPLE OF 16.

IF(MOD(MORE, 16) .EQ. 0)GOTO 73

C IF NOT, FIND THE NUMBER WHICH EXACT MULTIPLE OF 16 AND IS CLOSEST TO

NUM2 = (MORE/16)*16 + 16

READ(1,38,END=533)(SPEECH(I), I = 1, NUM2)

CONTINUE

ICNT = ICNT + NUM2

IF(ICNT .GT. NSAMP)GOTO 999

NUM3 = NUM2 - MORE

DO 64  I = 1, MORE
   IBUFF1(NUM1 + I) = SPEECH(I)
64  CONTINUE

DO 67  I = 1, NUM3
   IBUFF2(I) = SPEECH(MORE + I)
67  CONTINUE

NUM12 = NUM3
GOTO 73

C SINCE "MORE" IS EXACT MULTIPLE OF 16, READ "MORE" SAMPLES FROM
C SPEECH FILE AND PUT IN IBUFF1.

READ(1,39,END=537)(IBUFF1(NUM1 + I), I = 1, MORE)

CONTINUE

ICNT = ICNT + MORE

IF(ICNT .GT. NSAMP)GOTO 999

NUM12 = 0
CALCULATE THE WINDOW FUNCTION.

CALL WINDOW(H,NP) 1 COMPUTE TRIANGULAR WINDOW.

PERFORM FREQUENCY DIVISION OPERATION.

DO 72 I = 1, NP 1 FREQUENCY DIVISION OPERATION.
    NUM4 = NP + 1
    VY(I) = FLOAT(IBUFF1(NUM4)) + H(I) * FLOAT(IBUFF1(I) -
          IBUFF1(NUM4))
    IARG = NADD + I
    VY = VY(I)

CALL ARC SUBROUTINE WHICH RETURNS THE QUANTIZER OUTPUT CORRESPONDING
TO FREQUENCY DIVIDED SPEECH SAMPLE AND ALSO NOISY SPEECH SAMPLE.
CALL ARC(YY,Q,YHAT)

CALL BUFCTL(BUFCNT,JQO,AVGBIT) I BUFFER CONTROL IF OVERFLOW.

WRITE THE QUANTIZER OUTPUT IN THE FILE.
N1 = MODN(IARG)
SQ(N1) = 0 IF(N1 .EQ. 16) WRITE(2,33) SQ

CONTINUE

NADD = NADD + NP
NUM1 = KSLK1 + ITMAX

THERE ARE ALREADY NUM12 SAMPLES IN IBUFF2. PUT (NUM1-NUM12) MORE
SAMPLES IN IT.

NSP = NUM1 - NUM12

MAKE NSP EXACT MULTIPLE OF 16.
NSP = (NSP/16) * 16 + 16
READ(*,END=541)(IBUFF2(NUM12+I), I=1,NSP)

CONTINUE
ICNT=ICNT+NSP
IF(ICNT .GT. NSAMP) GOTO 999

NUMNSP = NUM12 + NSP

EXTRACT PITCH

JJ = JJ + 1
CALL PITCH(IBUFF2,NP,ITMAX,ITMIN,KBLK1, NUMP,NUMNSP,IOPT2)

IF(NUMNSP .LT. NP) GOTO 85

DOES IBUFF2 HAVE SAMPLES LESS THAN 2*NP.

IF(NUMNSP .LT. NP2) GOTO 85

HOW MANY EXTRA SAMPLES IBUFF2 HAS?
NUM12 = NUMNSP - NP2

DOES IBUFF2 HAVE EXACT 2*NP SAMPLES?

IF(NUM12 .EQ. 0) GOTO 82

TRANSFER EXTRA SAMPLES TO IBUFF1.
C

DO 79 I=1,NUM12
IBUFF1(I)=IBUFF2(NP2+I)
79 CONTINUE

GOTO 88

C HOW MANY MORE SAMPLES ARE TO BE ADDED TO IBUFF2 SO THAT IT WILL HAVE 2*NP SAMPLES?
C
C MORE = NP2 - NUM2
C
88 IS MORE EXACT MULTIPLES OF 16?
C
IF(MOD(MORE,16).EQ.0)GOTO 91
C
C IF NOT, MAKE IT EXACT MULTIPLE OF 16.
C
NUM2=(MORE/16) * 16 + 16
READ(1,38,END=544)(SPEECH(I),I=1,NUM2)
544 CONTINUE
ICNT=ICNT+NUM2
IF(ICNT.GT.NSAMP)GOTO 999
C
C HOW MANY EXTRA SAMPLES ARE READ THAN NEEDED.
C
NUM3 = NUM2 - MORE
C
C TRANSFER "MORE" SAMPLES TO IBUFF2 AND NUM3 SAMPLES TO IBUFF1.
C
DO 86 I=1,MORE
IBUFF2(NUMNSP+I)=SPEECH(I)
86 CONTINUE
DO 87 I=1,NUM3
IBUFF1(I)=SPEECH(MORE+I)
97 CONTINUE

C NUM12=NUM3
GOTO 88

91 READ(1,38,END=547)(IBUFF2(I+NUMNSP),I=1,MORE)
547 CONTINUE
ICNT=ICNT+MORE
NUM12=0

88 CALL WINDOW(H,NP)       ! COMPUTE TRIANGULAR WINDOW FUNCTION.
C
C
DO 92 I=1,NP
Y(I) = FLOAT(IBUFF2(NUM4)) + H(I) * FLOAT(IBUFF2(NUM4)) - IBUFF2(NUM4))
92 CONTINUE

IARG = NADD + I
YY = Y(I)

C------ CALL SUBROUTINE ARC.
C
CALL ARC(YY,Q,YHAT)    ! QUANTIZE COMPRESSED SP.
QQQ=Q
CALL BUFCTL(BUCNQ,YQQ,AVGBIT) ! CONTROL BUFFER IF OVERFLOW.
N1 = MODN(IARG)
SQ(N1) = Q
IF(N1.EQ.16) WRITE(2,33)SQ

92 CONTINUE

NADD = NADD + NP
C
C THERE ARE ALREADY NUM12 SAMPLES IN IBUFF1.TRANSFER NSP=NUM1-NUM12
SAMPLES TO IT.

NSP = NUM1 - NUM12
NSP = (NSP/16)*16 + 16
READ(1,30,END=551)(IBUFFI(NUM12+I),I=1,NSP)
CONTINUE
ICNT=ICNT+NSP
IF(ICNT.GT.NSAMP)GOTO 999
NUMNSP=NUM12+NSP

JJ = JJ + 1
CALL PITCH(IBUFFI, NP, ITMAX, ITMIN, KBLKI, NUMP, NUMNSP, ILOPT2)
IPICH(J3) = NP
NP2 = NP + NP
NUM1 = NUMNSP
GOTO 56
CONTINUE

CALL INEND
PRINTOUT ON UNIT 6 ALL THE STATISTICS.
TYPE *, 'TYPE THE HEADER FOR THE PRINTOUT.(40CHAR ONLY)
ACCEPT 775.HD
775 FORMAT(40A1)
WRITE(6,774)HD
774 FORMAT(/,48A1,//)
WRITE(6,776)FORMAT(/,//6X,' SAMPLE NUMBER ',6X,' PITCH PERIOD ///)
ISTRT = 1
IEND = KBLKI+ITMAX
DO 777 I = 1,NUMP
      IP = IPICH(I)
      WRITE(6,778)ISTRT,IEND,IP
    777 CONTINUE
    778 FORMAT(6X,15.1X,'-'.1X,15.18X,13)
    33 FORMAT(1612)
STOP END

***************************************************************************
* PITCH EXTRACTION *
***************************************************************************

SUBROUTINE PITCH(IBUF, NP, ITMAX, ITMIN, KBLKI, NUMP, NBUF, ILOPT2)
DIMENSION A(280), IBUFF(512), IA(280), IBUF(512)
COMMON /CLIPP/CLPP

NUMP = NUMP + 1
M1 = NBUF/3
M2 = M1+M1
DO 5 I = 1,NBUF
      IBUFF(I)=IBUF(I)
CONTINUE
DO 10 I = 1,280
A(i) = S, S
I A(i) = S
18 CONTINUE
C------- CHECK IF CENTER CLIPPING IS ASKED FOR.
C IF(IOPT2 .LE. 2) GOTO 28
C------- FIND OUT ABSOLUTE MAXIMUM OUT OF NBUFF SAMPLES IN THE BUFFER"IBUFF"
C IBIG1 = 8
IBIG2 = 8
DO 126 I = 1, NBUFF
ISPABS = ABS(IBUFF(I))
IF(ISPABS .GT. IBIG1) IBIG1 = ISPABS
126 CONTINUE
C DO 121 I = N2, NBUFF
ISPABS = ABS(IBUFF(I))
IF(ISPABS .GT. IBIG2) IBIG2 = ISPABS
121 CONTINUE
IB = IBIG1 - IBIG2
IF(liB .LE. 0) IBIG = IBIG2
IBIG = 8
C------- ENTER THE CLIPPING LEVEL.
C CL = CLPP * FLOAT(IBIG)
C------- CHECK IF CENTER CLIPPING WITH THREE OR TWO VALUES IS REQUIRED.
C IF(IOPT2 .GT. 4) GOTO 155
C CLIPP THE SPEECH WAVEFORM.
C CLM = CL
DO 146 I = 1, NBUFF
XFLT = FLOAT(IBUFF(I))
IF((XFLT .LE. CL) .AND. (XFLT .GT. CLM)) IBUFF(I) = IBUFF(I) - IFIX(CL)
IF(XFLT .LE. CL) IBUFF(I) = IBUFF(I) + IFIX(CL)
GOTO 146
146 CONTINUE
GOTO 28
155 CONTINUE
IF(IOPT2 .GT. 6) GOTO 36
C CLIPP THE SPEECH WAVEFORM.
C CLM = CL
DO 166 I = 1, NBUFF
XFLT = FLOAT(IBUFF(I))
IF((XFLT .LE. CL) .AND. (XFLT .GT. CLM) .OR. (XFLT .LE. -CL)) IBUFF(I) = 8
IF((XFLT .LE. CL) .AND. (XFLT .GT. CL)) IBUFF(I) = 1
IF((XFLT .LE. -CL)) IBUFF(I) = -1
166 CONTINUE
IF(IOPT2 .GT. 5) GOTO 228
C------- COMPUTE AUTOCORRELATION FUNCTIONS
C IBIG = 188
DO 188 IT = ITMIN, ITMAX
ISUM = 8
DO 178 J = 1, KBLK1
IF((IBUFF(J+IT).LT.CL) .AND. (IBUFF(J).LT.CL)) OR. ((IBUFF(J+IT)) .LT. 8)
178 CONTINUE
C------- COMPUTE AUTOCORRELATION FUNCTIONS
1 .GT.0).AND.( IBUFF(J).GT.0))ISUM=ISUM+1
1 .GT.0).AND.( IBUFF(J).LT.0))ISUM=ISUM-1
1 170 CONTINUE
IA(IT)=ISUM
IF(IBIGG .LT. IA(IT)) IBIGG=IA(IT)
180 CONTINUE
DO 190 I=ITMIN,ITMAX
IF(IBIGG .EQ. IA(I))GOTO 200
190 CONTINUE
200 NP=I
WRITE(S.*)NP
RETURN
C
C ------- CALCULATE AMDF FUNCTIONS
220 CONTINUE
ISMALL=4096
DO 230 IT=ITMIN,ITMAX
ISUM=8
DO 240 J=1,KBLK1
IF(((IBUFF(J+IT).GT.0).AND.( IBUFF(J).LT.0)) OR.
1 (((IBUFF(J+IT).LT.0).AND.(IBUFF(J).GT.0)))ISUM=ISUM+2
1 240 CONTINUE
IA(IT)=ISUM
IF(ISMALL .GE. IA(IT))SMALL=IA(IT)
250 CONTINUE
DO 250 I=ITMIN,ITMAX
IF(ISMALL .EQ. IA(I))GOTO 260
250 CONTINUE
260 WRITE(S.*)NP
RETURN
C
C---- CALCULATE AMDF FUNCTIONS
290 CONTINUE
ISMALL=4096
DO 300 IT=ITMIN,ITMAX
ISUM=8
DO 310 J=1,KBLK1
SUM=8.0
DO 300 J=1,KBLK1
SUM=SUM+FLOAT(IBUFF(J+IT))*FLOAT(IBUFF(J))
300 CONTINUE
A(IT)=SUM
IF(BIG .LT. A(IT))BIG=A(IT)
310 CONTINUE
DO 310 I=ITMIN,ITMAX
IF(BIG .EQ. A(I))GOTO 320
310 CONTINUE
320 NP=I
WRITE(S.*)NP
RETURN
C
C---- CALCULATE AMDF FUNCTIONS
340 CONTINUE
SMALL = 1.0E+09
DO 50 I=ITMIN,ITMAX
SUM = 8.0
DO 50 J=1,KBLK1
SUM = SUM + ABS(FLOAT(IBUFF(J+IT) - IBUFF(J)))
50 CONTINUE
A(IT) = SUM
IF(SMALL .GE. A(IT)) SMALL = A(IT)
60 CONTINUE
DO 70 I=ITMIN,ITMAX
IF(SMALL .EQ. A(I))GOTO 80
70 CONTINUE
C
C ------- CALCULATE AMDF FUNCTIONS
790 CONTINUE
CONTINUE
NP = I
WRITE(5,*),NP
RETURN

CONTINUE

CLIPP THE SPEECH TO TWO VALUES.
CL = 8.8
DO 380 I = 1, NBUFF
     XFLOT = FLOAT(IBUFF(I))
     IF(XFLOT .LE. CL)IBUFF(I) = -1
     IF(XFLOT .GT. CL)IBUFF(I) = 1
380 CONTINUE

CONTINUE
IF(IOPTZ .GT. 7)GOTO 480
IBIGG = -1000
DO 420 IT = ITMIN, ITMAX
     ISUM = 0
     DO 400 J = 1, KBLK1
          IF(IBUFF(J+IT) .EQ. IBUFF(J))ISUM = 1
     400 CONTINUE
     IA(IT) = ISUM
     IF(IBIGG .LT. IA(IT))IBIGG = IA(IT)
420 CONTINUE
480 CONTINUE
NP = I
WRITE(5,*),NP
RETURN

CONTINUE

ISMALL = -4096
DO 580 IT = ITMIN, ITMAX
     ISUM = 0
     DO 490 J = 1, KBLK1
          IF(IBUFF(J+IT) .EQ. IBUFF(J))ISUM = 2
     490 CONTINUE
     IA(IT) = ISUM
     IF(ISMALL .LE. IA(IT))ISMALL = IA(IT)
580 CONTINUE
520 CONTINUE
NP = I
WRITE(5,*),NP
RETURN

-------------------------------------------------------------------------
* WINDOW FUNCTION *
-------------------------------------------------------------------------

SUBROUTINE WINDOW(H, NP)
DIMENSION H(400)
DO 10 I = 1, NP
     HI(I) = 1.8 - FLOAT(I-1)/FLOAT(NP-1)
10 CONTINUE
RETURN
END
**QUANTIZER**

QUANTIZER IS VARIABLE LEVEL QUANTIZER. NUMBER OF LEVELS ARE KQ.

```fortran
SUBROUTINE QUINT(X,Y,I)
INTEGER*2 F1,F2,F3,F4,STAT
COMMON /QUAN/T(20),OUT(20),EXPN(20),SIZE,SMIN,NQ
COMMON /RMS/RMS,NBL,INARG
COMMON /ADDN/F1,F2,F3,F4,BT1,BT2,BT3
1,JCNT,STAT(30),GAMA(4)
C
1X=ABS(X/SIZE)
F=.5
IF(X.LT.0.)F=-.5
I=1
DO 20 K=1,NQ
IF(JCNT.EQ.1)TNEW=T(K)
20 IF(X.GE.TNEW)
I=2*K+.5+F
J=(I-2)/2
Y=2.*F*OUT(J)*SIZE
SIZE=EXPN(J)*SIZE
SIZE=MAXI(SIZE,RMS*SMIN)
RETURN
END
```

**PARAMETERS ARE DEFINED AND INITIALIZED.**

```fortran
SUBROUTINE INSTRT
INTEGER FILEN
COMMON /QUAN/T(20),OUT(20),EXPN(20),SIZE,SMIN,NQ
COMMON /RMS/RMS,NBL,INARG
COMMON /ADDN/F1,F2,F3,F4,BT1,BT2,BT3
1,JCNT,STAT(30),GAMA(4)
COMMON /PRED/INV,MNL,ALP,ALD,ALG,N,N,RMSMIN,SMIN
COMMON /RMS/RMS,NBL,INARG
2,SNRQ(20)
COMMON /ADDN/F1,F2,F3,F4,BT1,BT2,BT3
COMMON /INIT/JI
COMMON /PN/FILEN(16)
C
READ(3,40)FILEN  ! READ PARAMETERS FOR ANN SYSTEM.
FORMAT(16A2)
OPEN(UNIT=8,NAME=FILEN,TYPE='OLD')
READ(8,*)AINV,ALP,ALD,ALG,N,N,RMSMIN,SMIN
WRITE(6,2)AINV,ALP,ALD,ALG,N,N,RMSMIN,SMIN
2 FORMAT(6X,'A16')
WRITE(6,3)IQ
3 FORMAT(6X,'NUMBER OF QUANT LEVELS=' ,I2)
NQ=IQ/2
NQO=IQ/2
READ(8,*)EXPN(I),I=1,NQQ
WRITE(6,4)(1,EXPN(I),I=1,NQQ)
```

134
135

 FORMAT(6X,6('EXPN(','I2.')='F6.2,2X))
 READ(6,*) (OUT(I),I=1,NQQQ)
 WRITE(6,*) (OUT(I),I=1,NQQQ)

 DO 22 I=1,NQ
 30 T(I)=(OUT(I)+OUT(I+I))/2.
  SIZE=1B.

 DO 118 I=1,12
  WHAT(I)=&
  V(I)=&

 118 A(I)=&
  RMS=RMSMIN
  A(I)=AINV
  EY=0.
  EP=0.
  ENGY1=0.
  ENGY2=0.
  JI=0.
  ENGY3=0.
  ENGY4=0.
  SPER1=0.
  SPER2=0.

  READ(3,*)NBL
  DO 621 I=1,NQ

  621 ISTAT1(I)=&
  RETURN
  END

reib A DAP A TIVE RESIDUAL CODER

------- SUBROUTINE ARC RETURNS QUANTIZER OUTPUT AND WHAT.

 SUBROUTINE ARC(Y,Q,WHAT)
 INTEGER Q
 COMMON /PRED/G,N,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),WHAT(12),
 1 EV,ISTAT1(48),EP,ALAD,SPER1(200),SPER2(200),V(12),SNR(200)
 2 .SHR0B(200)
 COMMON /RMS/RMS,NBL,IARG,ENGY1,ENGY2,ENGY3,ENGY4,SPER1,SPER2
 COMMON /INIT/JI

------- PREDICTION.

  PRE=0.
  PRE1=0.
  DO 128 I=1,N

  128 PRE=PRE+ALP*WHAT(I)
  RMS=ALP*(RMS-RMSMIN)+(1.-ALP)*ABS(WHAT(I))+RMSMIN
  ERROR=Y-PRE
  ERROR=Y-PRE1
  CALL QUANT(ERROR,EQ,ISTAT1)-ISTAT1(IOUT)*1
  Q=IOUT
  DO 125 I=1,N

  125 WHAT(J)=WHAT(J-1)
  WHAT(I)=PRE+EQ
  V(I)=Y
YHAT = WHAT(1)

--- ADAPTATION.

ERR = G*EQ/RMS**2
A(1) = A(1) + AINV*(1./ALAD-1.)
DO 130 I = 1, N
130 A(I) = A(I) + ALAD*ERR*YHAT(I+1)

--- UPDATE PAST.

IRM = MOD(1ARG, NBL)
ENGY1 = ENGY1 + Y**2
ENGY2 = ENGY2 + (ERROR-EQ)**2
ENGY3 = ENGY3 - ERROR**2
ENGY4 = ENGY4 + ERROR**2
IF (IRM .NE. 0) GOTO 133
J1 = J1 + 1
EV = EV + ENGY1
EP = EP + ENGY2
IF (ENGY3 .NE. 0.) SPERB1(J1) = 10.*ALOG10(ENGY1/ENGY3)
IF (ENGY4 .NE. 0.) SPERB2(J1) = 10.*ALOG10(ENGY1/ENGY4)
IF (ENGY2 .NE. 0.) SNRB(J1) = 10.*ALOG10(ENGY1/ENGY2)
SPER1 = SPER1 - ENGY3
SREPZ = SREPZ + ENGY4
ENGY1 = 0.
ENGY2 = 0.
ENGY3 = 0.
ENGY4 = 0.
133 CONTINUE
RETURN
END

--- STATISTICS AND RESULTS

--- SUBROUTINE INEND WRITES ALL THE RESULTS SUCH AS SNR, H

--- AND STATISTICS.

SUBROUTINE INEND
REAL PROB(30)
INTEGER*2 STAT, F1, F2, F3, F4
COMMON /PRED/G.NRMSMIN, AINV, KQ, NPSAM, A(12),
1 WHAT(12), EV, ISTAT1(48), EP, ALAD, SPERB1(288), SPERB2(288), V(12)
2 , SNRB(288), SNRGB(288)
COMMON /RMS/RMS, NBL, 1ARG, ENGY1, ENGY2, ENGY3, ENGY4, SPER1, SPER2
COMMON /ADD/F1, F2, F3, F4, BT1, BT2, BT3
1 ,JCNT, STAT(38), GAMA(4)

SNR = 10.*ALOG10(EV/EP)
SREP = 10.*ALOG10(EV/SREP1)
SREP1 = 10.*ALOG10(EV/SREP2)
SNRQ = 10.*ALOG10(SPER1/EP)
ISUM = 0
DO 388 I = 1, KQ
388 ISUM = ISUM + ISTATE1(I)
ARG1 = ISUM
SUM=ALOG(ARG)
DO 500 I=1,KQ
ARG=ISTAT(I)+8.081
500 SUM=SUM-(ARG*ALOG(ARG))/ARG
BITS=SUM/ALOG(2.)
WRITE(6,402)SNR,BITS,(ISTAT(I),I=1,KQ)
402 FORMAT(6X,'SNR IN LOOP=','F7.3,3X,'H=','F4.2,3X,'OP',
1 10.5,1(/22X,10I5)/)
WRITE(6,403)SNR,BITS,(ISTAT(I),I=1,KQ)
WRITE(6,404)
404 FORMAT(///6X,'SAMPLE NUMBER ',6X,'SNR ',6X,'SPEI ',6X,
1 'SNEI ',6X,'SNRQ '///)
C
NB=ISUM/NBL
C
DO 480 I=1,NB
IS=(I-1)*NBL+1
IE=IS+NBL-1
WRITE(6,412)IS,IE,SNRB(I),SPEI(I),SNERB(I),SNRQ(I)
408 CONTINUE
412 FORMAT(6X,15.15,-,15.6X,F7.2,4X,F7.2,4X,F7.2,4X,F7.2)
WRITE(6,416)SPEI,SNRQ
416 FORMAT(///6X,'PREDICTOR PERFORMANCE=','F8.2/6X,'SIGNAL TO NOISE
2 RATIO=','F8.2)"
C
C
C
C
C
BUFFER CONTROL
C
SUBROUTINE BUFCTL(BUFCNT,J,AVGBIT)
INTEGER*2 STAT,F1,F2,F3,F4
COMMON /ADDN/F1,F2,F3,F4
COMMON /JINT/STAT(30),GAMA(4)
C
IF( F1 .EQ. 1)GOTO 50
IF( F2 .EQ. 1)GOTO 50
IF( F3 .EQ. 1)GOTO 50
IF( J .LE. 3) GOTO 80
STAT(J)=STAT(J)+1
BUFCNT=BUFCNT+4.-AVGBIT
F4=4
GOTO 100
50 CONTINUE
IF( J .EQ. 1)GOTO 50
IF( J .EQ. 2)GOTO 153
IF(J .EQ. 3) GOTO 155
STAT(3) = STAT(3) + 1
STAT(1) = STAT(1) + 1
F1 = 0
BUFCT = BUFCT + 4 - AVGBIT
BUFCT = BUFCT + 4 - AVGBIT
F4 = 1
GOTO 180
CONTINUE
STAT(12) = STAT(12) + 1
BUFCT = BUFCT + 4 - AVGBIT - AVGBIT
F4 = 0
F1 = 0
GOTO 180
F2 = 1
STAT(1) = STAT(1) + 1
F1 = 0
BUFCT = BUFCT + 4 - AVGBIT
GOTO 180
F3 = 1
STAT(1) = STAT(1) + 1
F1 = 0
BUFCT = BUFCT + 4 - AVGBIT
GOTO 180
CONTINUE
IF(J .EQ. 1) GOTO 63
IF(J .EQ. 2) GOTO 65
IF(J .EQ. 3) GOTO 67
BUFCT = BUFCT + 4 - AVGBIT
STAT(J) = STAT(J) + 1
BUFCT = BUFCT + 4 - AVGBIT
F4 = 1
F2 = 0
GOTO 180
63
STAT(13) = STAT(13) + 1
BUFCT = BUFCT + 4 - AVGBIT - AVGBIT
F2 = 0
F4 = 0
GOTO 180
65
STAT(14) = STAT(14) + 1
BUFCT = BUFCT + 4 - AVGBIT - AVGBIT
F4 = 0
F2 = 0
GOTO 180
67
STAT(2) = STAT(2) + 1
BUFCT = BUFCT + 4 - AVGBIT
F3 = 1
F2 = 0
F4 = 0
GOTO 180
CONTINUE
IF(J .EQ. 1) GOTO 72
IF(J .EQ. 2) GOTO 74
IF(J .EQ. 3) GOTO 76
STAT(3) = STAT(3) + 1
BUFCT = BUFCT + 4 - AVGBIT
STAT(J) = STAT(J) + 1
BUFCT = BUFCT + 4 - AVGBIT
F3 = 0
F4 = 0
GOTO 180
72
STAT(15) = STAT(15) + 1
BUFCT = BUFCT + 4 - AVGBIT - AVGBIT
F3=0
F4=0
GOTO 188
74
STAT(3)=STAT(3)+1
BUFCNT=BUFCNT+4.-AVGBIT
F2=1
F4=8
F3=0
GOTO 188
76
STAT(16)=STAT(16)+1
BUFCNT=BUFCNT+4.-AVGBIT-AVGBIT
F4=8
F3=8
GOTO 188
88
CONTINUE
IF( J .EQ. 1)F1=1
IF( J .EQ. 2)F2=1
IF( J .EQ. 3)F3=1
188
CONTINUE
IF( BUFCNT .GT. BT1)JCNT=2
IF( BUFCNT .GT. BT2)JCNT=3
IF( BUFCNT .GT. BT3)JCNT=4
IF( BUFCNT .LT. BT1)JCNT=1
IF( BUFCNT .LT. 0.)BUFCNT=8.
RETURN
END
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* ENTPROPY CODING                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

PROGRAM NAME:          ENCO.FTN
DATE:                 JUNE 30, 1981

THIS PROGRAM READS THE QUANTIZER LEVELS, SORTS THEM OUT ACCORDING TO
RUN LENGTH AND PUTS APPROPRIATE CODE WORDS IN OUTPUT FILE.

INTEGER*2 BIN(256),BOUT(512),TBUD(256),HDD(40),BOUT1(256)
DIMENSION IVECT(1)
EQUIVALENCE (BOUT(1),BOUT(1))
DATA IVECT/8,1,4,2,5,3,6,11,7,15,14/
KBIT=8
LATT=8
NEWBLO=8
TYPE=1 , TYPE THRESHOLD TO SWITCH CODE TO PREVENT BUFFER UNDERFLOW
TYPE=2 , TYPICAL VALUE: 100
ACCEPT=* , ITH
IBINT=8

OPEN(UNIT=1,TYPE='OLD',NAME='QUANT.DAT')
OPEN(UNIT=2,TYPE='NEW',NAME='CODE.DAT',CARRIAGECONTROL='LIST')
OPEN(UNIT=3,TYPE='NEW',NAME='TBUF.DAT',CARRIAGECONTROL='LIST')

TYPE=1 , TYPE HAMMING CODE:INFO BITS,CHECK BITS
ACCEPT=* , INFOB,NPARI

READ(1,7)SENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
FORMAT(615,1X,40A1)
TYPE=1 , TYPE THE HEADER FOR BUFFER FILE:
ACCEPT=B,HD
FORMAT(40A1)
WRITE(3,7)SENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
TYPE=1 , TYPE HEADER FOR ENCODER OUTPUT FILE:
ACCEPT=B,HD
WRITE(2,7)SENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD

NBLO = NSAMP/256
I=0
DO 13 I=1,NBLO
READ(1,130,END=333)BIN
FORMAT(16I2)
IF=1
I=0
13 I=I+I
J=I+1
M=BIN(J)
IF(I.EQ.256.OR.KBIT.LE.ITH)GOTO 67
M1=BIN(I+1)
IF(M.GT.3.OR.M1.GT.3)GOTO 67
IF(M.EQ.1.AND.M1.GT.3)GOTO 67
IF(M.EQ.1.AND.M1.EQ.2)GOTO 67
IF(M.EQ.1 .AND. M1.EQ.3)GOTO 69
IF(M.EQ.2 .AND. M1.EQ.3)GOTO 69
IF(M.EQ.3 .AND. M1.EQ.2)GOTO 69
IF(M.EQ.1 .AND. (M1.EQ.1))KCOD=8
IF(M.EQ.2 .AND. (M1.EQ.2))KCOD=13 1 2,1
IF(M.EQ.3 .AND. (M1.EQ.1))KCOD=9 1 3,1
IF(M.EQ.3 .AND. (M1.EQ.3))KCOD=12 1 3,3
IP=2
BOUT(J)=KCOD
KBIT=KBIT+2
DO 78 ILOOP=1,4 1 ADD "NPARI" BITS
IBCNT=IBCNT+1 1 TO BUFFER EVERY "INFOB"
IF(MOD(IBCNT,INFOB)).NE. 0)GOTO 78
KBIT=KBIT+NPARI
IBCNT=0
CONTINUE
78
C
IF(KBIT.GT.1024)KBIT=1024
IF(KBIT.LT.0)KBIT=0
TBU(I)=LAST
GOTO 81
81
TBU(I)=TBU(I-1)
TBU(I+1)=KBIT
GOTO 29
IP=1
BOUT(J)=IVECT(M)
KBIT=KBIT+1
DO 62 ILOOP=1,4
IBCNT=IBCNT+1
IF(MOD(IBCNT,INFOB)).NE. 0)GOTO 62
KBIT=KBIT+NPARI
IBCNT=0
CONTINUE
62
C
IF(KBIT.GT.1024)KBIT=1024
IF(KBIT.LT.0)KBIT=0
TBU(I)=KBIT
CONTINUE
29
C
WRITE(5,328)I,BIN(I),(I=1),BIN(I=1),J,BOUT(J),KCOD,TBU(J)
FORMAT(2X,'BIN(13,')='13,2X,'BIN(13,')='13,2X,'BOUT(13,')=','13,2X,'KBIT=','13)
C
256 OR (I.EQ.255 .AND. IP.EQ.2)GOTO 78
GOTO 13
C
WRITE(3,228)TBU
FORMAT(1615)
LAST=TBU(256)
IF(J.LT.256)GOTO 10
WRITE(2,158)BOUT1
FORMAT(1612)
NEWBLO=NEWBLO+1
KK=J-256
IF(KK.EQ.8)GOTO 93
DO 91 II=1,KK
BOUT(II)=BOUT(256+II)
J=KK
CONTINUE

IF(J.LT.256) GOTO 410
IF(J.GT.256) GOTO 550
GOTO 333

410 LL=256-J
DO 420 L=1,LL
   BOUT(J+L)=IVECT(1)
420 WRITE(2,150)BOUT1
   NEWBLO=NEWBLO+1
   GOTO 333

550 DO 560 I=1,256
560 BOUT(I(I)=IVECT(1)
   KK=J-256
   DO 570 I=1,II
   BOUT(I(I)=BOUT(256+II)
570 GOTO 430

CLOSE (UNIT=1)
CLOSE (UNIT=2)
CLOSE (UNIT=3)

OPEN(UNIT=1,TYPE='NEW',NAME='NEWBLO.DAT')
WRITE(1,666)NEWBLO
666 FORMT(13)
CLOSE(UNIT=1)
STOP
END
PROGRAM NAME: CHER2631.FTN

THIS PROGRAM READS CODE.DAT FILE AND INTRODUCES RANDOM ERRORS IN THE BIT STREAM ACCORDING TO BIT ERROR RATE (BER) IN PERCENT.
THEN IT WRITES A NEW FILE WITH CODEWORD AFFECTED BY CHANNEL ERRORS.
AS OPTION, IT PERFORMS PARITY CHECK CORRECTION AND/OR DOUBLE ERROR CORRECTION.
The FRAME LOOP OF 32 BITS COULD BE ASSOCIATED TO A (31,26)
HAMMING CODE WITH 6*4 + 1 SYNC. + 1 DOUBLE ERR.DETECT. =26

INTEGER*2 BIN(256),BOUT(256),HD(48),FLAG,ISAM(38),DOUB

ACCEPT *,BER
ACCEPT *,FLAG
ACCEPT *,DOUB

OPEN(UNIT=1,TYPE='OLD',NAME='CODE.DAT')
OPEN(UNIT=2,TYPE='NEW',NAME='ERCO.DAT',CARRIAGECONTROL='LIST')
OPEN(UNIT=3,TYPE='OLD',NAME='NEWBLO.DAT')
READ(3,5)NBLO
READ(3,5)NBLO
FORMAT(13)
READ(1,9)NSENT,IRATE,NBBS,IUPPR,ILOWR,NTERMS,HD
FORMAT(615,18X,48A1)
TYPE = ', TYPE HEADER FOR CHERR OUTPUT FILE:'
ACCEPT 11,HD
FORMAT(48A1)
WRITE(2,9)NSENT,IRATE,NBLO,IUPPR,ILOWR,NTERMS,HD

DO 13 JJ=1,10000
CALL RANDU(K1,K2,X)
DO 18 IB=1,NBLO
READ(1,180,END=345)BIN
FORMAT(16I2)
C
DO 20 IR=1,32
1K=0
DO 22 JG=1,38
ISAM(JG)=0
L=0
C
DO 25 IZ=1,8
I=IZ+8*(IR-1)
M=BIN(I)
DO 30 II=1,4
CALL RANDU(K1,K2,X)
IF(X.GT.EX) GOTO 30
J=(Z**((I-1)+.5))
M = Ieor(M,J)
L=L+1
ISAM(L)=I
IKK=IKK+1
CONT=CONT+1.
C
CONTINUE
BOUT(1)=M
25 CONTINUE
IF(IKK.NE.1)GOTO 65
ICS=ICS+1
IF(FLAG.EQ.8)GOTO 20
IX=ISAM(1)
BOUT(IX)=BIN(IX)
ICOR=ICOR+1
GOTO 20
C
65 IF(IKK.EQ.8)GOTO 20
IF(IKK.EQ.2.AND.DOUB.EQ.1)GOTO 75
GOTO 80
75 IDO=IDO+1
IX=ISAM(1)
BOUT(IX)=BIN(IX)
IX=ISAM(2)
BOUT(IX)=BIN(IX)
80 IKK=IKK+1
IF(IKK.EQ.8)ICD=ICD+1
IF(IKK.EQ.1)ICT=ICT+1
IF(IKK.EQ.1.AND.FLAG.EQ.1)GOTO 99
GOTO 20
C
99 OK1=K1
OK2=K2
DO 728 IPLUS=1,58
DO 718 IZ=1,8
I=IZ+8*(IR-1)
C
DO 728 KF=1,38
IFR=ISAM(KF)
IF(IFR.EQ.1)GOTO 718
CONTINUE
C
M=BIN(I)
DO 730 II=1,4
CALL RANDU(K1,K2,X)
IF(X.GT.EX)GOTO 730
J=2**(I1-1) + 8.5
M=IEOR(M,J)
CONT=CONT+1.
GOTO 788
728 CONTINUE
718 CONTINUE
788 CONTINUE
C 788 BOUT(I)=M
1 INSERT ONE ADDITIONAL ERROR : PARITY CHECK FAILS
K1=OK1
K2=OK2
C 28 CONTINUE
C C WRITE(2,158)BOUT
FORMAT(1612)
158 CONTINUE
134 TYPE *, TOTAL # OF ERRORS= ' , CONT
TYPE *, TOTAL # OF SINGLE ERRORS CORRECTED= ' , ICOR
TYPE *, TOTAL # OF DOUBLE ERRORS CORRECTED= ' , INO
TYPE *, TOTAL # OF SINGLE ERR./FRAME= ' , ICS
TYPE *, TOTAL # OF EVEN ERR./FRAME= ' , ICO
TYPE *, TOTAL # OF ODD ERR./FRAME= ' , ICT
RESI=CONT-ICOR-2*IDO
345 TYPE *, # OF RESIDUAL ERRORS= ' , RESI
IF(CONT .NE. 0.)BERNEW=(RESI/CONT)*BER
IF(CONT .NE. 0.)TYPE *, ' NEW BER AFTER ERR CORRECTION = ' , BERNEW
CLOSE(UNIT=1)
CLOSE(UNIT=2)
CLOSE(UNIT=3)
STOP
END
**CHANNEL ERROR**
**WITH PARITY CHECK AND/OR**
**DOUBLE ERROR CORRECTION**

**Program Name:** CHER5763.FTN

This program reads CODE.DAT file and introduces random errors in the bit stream according to bit error rate (BER) in percent. Then it writes a new file with codeword affected by channel errors. As option, it performs parity check correction and/or double error correction.

```fortran
INTEGER*2 BIN(256),BOUT(256),HD(48),FLAG,ISAM(38),DOUB
ACCEP'T TYPE = 'ENTER BER(X)' BER
THREE = 'PARITY CHECK = 1 : NO PAR. CHECK = 0'
ACCEP'T TYPE = 'DOUBLE ERR. CORRECT. = 1 : NO ERR. CORRECT. = 0'
ACCEP'T TYPE = 'DOUB

INTEGER DOB
DOB=0
ICS=0
ICD=0
ICT=0
ICOR=0
IF (BER.LT.1.E-07)BER=0.0
EXT = BER/100.
KI = 773
K2 = 119
CONT=0.
IUNO=1

OPEN(UNIT=1,TYPE='OLD',NAME='CODE.DAT')
OPEN(UNIT=2,TYPE='NEW',NAME='ERCO.DAT',CARRIAGECONTROL='LIST')
OPEN(UNIT=3,TYPE='OLD',NAME='NEWBLO.DAT')
READ(3,5)NBLO
5 FORMAT(I3)
READ(1,9)NSENT,IRATE,NBBS,IUPPR,ILOWR,NTERMS,HD
9 FORMAT(6I5,10X,40A1)
TYPE = 'TYPE HEADER FOR -CHERR OUTPUT FILE:'
ACCEP'T TYPE = 'TYPE HEADER FOR -CHERR OUTPUT FILE:'
WRITE(2,9)NSENT,IRATE,NBLO,IUPPR,ILOWR,NTERMS,HD

DO 13 JJ=1,1000
13 CALL RANDU(KI,K2,X)
DO 18 IB=1,NBLO
18 READ(1,128,END=345)BIN
FORMAT(16I2)
```
C
DO 28 IR=1,16
  IKK=8
DO 22 JG=1,38
  ISAM(JG)=8
  L=8
C
DO 25 IZ=1,16
  I=(IZ+16*(IR-1))
  M=BIN(I)
    DO 38 II=1,4
       CALL RANDU(K1,K2,X)
       IF(X.GT.EXT)GOTO 38
       J = (2.**((II-1) + 8.5))
       M = IEOR(M,J)
       L=L+1
       ISAM(L)=I
       IKK=IKK+1
    CONT=CONT+1.
38 CONTINUE
BOUT(I)=M
25 IF(IKK.NE.1)GOTO 65
   ICS=ICS+1
   IF(FLAG.EQ.0)GOTO 28
   IX=ISAM(1)
   BOUT(IX)=BIN(IX)
   ICOR=ICOR+1
   GOTO 28
C
65 IF(IKK.EQ.0)GOTO 28
   IF(IKK.EQ.2)AND. DOUS.E0.1)GOTO 75
   GOTO 88
75 DO=IDO+1
   IX=ISAM(1)
   BOUT(IX)=BIN(IX)
   IX=ISAM(2)
   BOUT(IX)=BIN(IX)
   IKK=IKK(1)AND(IKK,1UNO)
   IF(IKK.EQ.0)ICD=ICD+1
   IF(IKK.EQ.1)ICT=ICT+1
   IF(IKK.EQ.1.AND.FLAG.EQ.1)GOTO 99
   GOTO 28
C
99 OKI=K1
   OK2=K2
   DO 788 IPRI=1.58
      IEXTRA LOOPS
   DO 718 IZ=1,16
      I=(IZ+16*(IR-1))
    CONTINUE
C
DO 728 KF=1,38
   IPR=ISAM(KF)
   IF(IPR.EQ.1)GOTO 718
728 CONTINUE
C
M=BIN(I)
DO 738 II=1,4
   CALL RANDU(K1,K2,X)
   IF(X.GT.EXT)GOTO 738
   J=2.**((II-1) + 8.5)
   M=IEOR(M,J)
   CONT=CONT+1.
GOTO 788
710 CONTINUE
700 CONTINUE
C
788 BOUT(I)=M       ! INSERT ONE ADDITIONAL ERROR : PARITY CHECK FAILS
     K1=OK1
     K2=OK2
C
20 CONTINUE
C
WRITE(2,150)BOUT
FORMAT(16I2)
C
150 CONTINUE
345 CONTINUE
C
TYPE *,' TOTAL # OF ERRORS='.CONT
TYPE *,' TOTAL # OF SINGLE ERRORS CORRECTED= '.ICOR
TYPE *,' TOTAL # OF DOUBLE ERRORS CORRECTED= '.IDO
TYPE *,' TOTAL # OF SINGLE ERR./FRAME= '.ICS
TYPE *,' TOTAL # OF EVEN ERR./FRAME= '.ICD
TYPE *,' TOTAL # OF ODD ERR./FRAME= '.ICT
RESI=CONT-ICOR-2*IDO
TYPE *,' # OF RESIDUAL ERRORS= '.RESI
IF(CONT .NE. 0.)BERNEW=(RESI/CONT)*BER
IF(CONT .NE. 0.)TYPE *,' NEW BER AFTER CORRECTION= '.BERNEW
CLOSE(UNIT=1)
CLOSE(UNIT=2)
CLOSE(UNIT=3)
STOP
END
**DECODER**

**PROGRAM NAME:** DECO3.FTN

This program reads channel errors and decodes them and writes corresponding quantizer levels in a new file. It also computes receiver buffer occupancy.

```fortran
REAL SAMP(1216), LAST
INTEGER*2 BIN(256), BOUT(1224), ISAMP(256), FNAME(16), KAR(2), BOUT1(256)
INTEGER*2 HD(40), HDI(40)
EQUIVALENCE (BOUT(1), BOUT1(1))

C TYPE "", ENTER THE CODEWORD FILENAME FOR DECODER:
ACCEPT 4, FNAME
FORMAT(16A2)

OPEN(UNIT=1, TYPE='OLD', NAME=FNAME)
OPEN(UNIT=2, TYPE='NEW', NAME='DECO.DAT', CARRIAGECONTROL='LIST')
OPEN(UNIT=3, TYPE='NEW', NAME='RBUF.DAT', CARRIAGECONTROL='LIST')
OPEN(UNIT=4, TYPE='OLD', NAME='NEWBLO.DAT')
READ(4, 5) NBLO
FORMAT(13)

READ(1, 9) NSENT, IRATE, NSOLD, IUPPR, ILOWR, NTERMS, HD
NSAMP=NBLO*256
FORMAT(615, 10X, 40A1)
TYPE "", TYPE HEADER FOR DECODER OUTPUT FILE:
ACCEPT 11, HD
FORMAT(40A1)
WRITE(2, 9) NSENT, IRATE, NSAMP, IUPPR, ILOWR, NTERMS, HD
TYPE "", TYPE HEADER FOR RECEIVER BUFFER OUTPUT FILE:
ACCEPT 11, HDI
WRITE(3, 9) NSENT, IRATE, NSAMP, IUPPR, ILOWR, NTERMS, HDI

C TYPE ".", INITIALIZE THE RECEIVER SAMPLE BUFFER:
ACCEPT "," , WORD
C LAST=360. RECEIVER BUFFER INITIALIZ.
WORD=360.
LAST=WORD
IND=1
END=9
NEW=9

DO 18 IB=1, NBLO
READ(1, 18E, END=345) BIN
FORMAT(1612)
DO 28 I=1, 256
L=BIN(I)
IF(L .EQ. 8) GOTO 35
IF(L .EQ. 18) GOTO 45
IF(L .EQ. 9) GOTO 55
IF(L .EQ. 12) GOTO 53
```

---

149
IF(L.EQ.0)BOUT(IND)=1
IF(L.EQ.1)BOUT(IND)=2
IF(L.EQ.2)BOUT(IND)=4
IF(L.EQ.3)BOUT(IND)=6
IF(L.EQ.4)BOUT(IND)=3
IF(L.EQ.5)BOUT(IND)=6
IF(L.EQ.6)BOUT(IND)=7
IF(L.EQ.7)BOUT(IND)=9
IF(L.EQ.8)BOUT(IND)=10
IF(L.EQ.9)BOUT(IND)=11
IF(L.EQ.10)BOUT(IND)=12
IF(L.EQ.11)BOUT(IND)=13
IF(L.EQ.12)BOUT(IND)=14
IF(L.EQ.13)BOUT(IND)=15
IF(L.EQ.14)BOUT(IND)=16
IF(L.EQ.15)BOUT(IND)=17
IF(L.EQ.16)BOUT(IND)=18

WORD=WORD+.3333

IF(WORD.GT.355.)WORD=355.
IF(WORD.LT.5.)WORD=5.

SAMP(IND)=WORD
IND=IND+1
GOTO 20

35
KAR(1)=1
KAR(2)=1
GOTO 68

40
KAR(1)=2
KAR(2)=1
GOTO 68

45
KAR(1)=2
KAR(2)=2
GOTO 68

50
KAR(1)=3
KAR(2)=1
GOTO 68

53
KAR(1)=3
KAR(2)=3

65
BOUT(IND)=KAR(1)
BOUT(IND+1)=KAR(2)

WORD=WORD+.3333

IF(WORD.GT.355.)WORD=355.
IF(WORD.LT.5.)WORD=5.
IF(IND.NE.1)GOTO 80
SAMP=LAST
GOTO 81

81
SAMP(IND)=SAMP(IND-1)
SAMP(IND+1)=WORD
IND=IND+2

CONTINUE 1 SAMPLE LOOP

WRITE(2,151)BOUT1
FORMAT(1612)
NEWB=NEWB+1

DO 55 I=1,256

55 ISAMP(I)=ISAMP(I)+.5
WRITE(3,151)ISAMP

IF(IND.LE.257)GOTO 75
IF(IND.NE.NBLO)END=1
IF(IND.GT.1924)STOP 'BUFFER TOO SHORT'
KK=IND-257
DO 7# I=1, KK
    BOUT(I)=BOUT(256+I)
7#
    SAMP(I)=SAMP(256+I)
    IND=KK+1
    IF(IND.GT. 257)GOTO 149
    GOTO 18
    IND=1
    LAST=SAMP(256)
C
1# CONTINUE 1 BLOCK LOOP
C
    IF(IND .LT. 257 .AND. IEND .EQ. 0)GOTO 345
C
    WRITE(5,987)IND
    FORMAT(1X, ' # SAMPLES IN THE TAIL = ', I6)
    IF(IND.GE. 257)GOTO 43#
C
    DO 3#1 I=IND,256
    BOUT(I)=1     ! FILL WITH SILENCE
    IF(WORD.GT.355,WORD=355).
    IF(WORD.LT. B,WORD=B.
    SAMP(I)=WORD
3#1
C
    WRITE(2,158)BOUT1
    DO 999 J=1,256
    ISAMP(J)=SAMP(J)+.5
    WRITE(3,151)ISAMP
    999
    NEWB=NEWB+1    ! FINAL # OF BLOCKS
C
    DO 345
    CLOSE(UNIT=1)
    CLOSE(UNIT=2)
    CLOSE(UNIT=3)
    CLOSE(UNIT=4)
    NSAMP=NEWB*256
    OPEN(UNIT=1, TYPE='NEW', NAME='RECSAM.DAT', CARRIAGECONTROL='LIST')
    WRITE(1,888)NSAMP
    345
C
    WRITE(1,888)NSAMP
    888
    FORMAT(15)
    STOP
END
TOHS - ARC SYSTEM

RECEIVER

PITCH IS EXTRACTED AT THE RECEIVER

Quantizer output for frequency divided speech is received. It is inverse quantized and passed through Arc Receiver which gives reconstructed frequency divided speech (YHAT). Frequency multiplication operation is performed on YHAT to get SHAT. To do this pitch period is needed, the values of which are read from PITCH.DAT file. Parameters at transmitter and that at receiver are the same and are read from parameter file.

Program Name: RECRVR.FTN
Date: June 30, 1981

This program asks for
1. Parameter filename: PARA.DAT
2. Pitch estimator option.
3. KBLK itmin, itmax

And produces output file
1. SHAT.DAT (output speech)
2. ZHAT.DAT (reconstructed comprsd sp)

INTEGER HD(40),Q(788),FNAME1(16),PITCH(400),P1,P2,P3,P4,SQ(16)
INTEGER*2 FNAME2(16)
DIMENSION YHAT(364),SHAT(364),H(400),IZHAT(256)
DIMENSION IBUFF1(512),IBUFF2(512)
COMMON /PRED/GND.RMSMINALP,AINV,KQ,NSPSAM,A(12),DVHAT(12),EV,
1 ISTAT1(40),EP,ALAD
COMMON /RMS/RMS
COMMON /CLIPP/CLPP
MODN(K) = K - (K-1)/16 * 16
MODM(K) = K - (K-1)/256 * 256

OPEN(UNIT=8,TYPE='OLD',NAME='OPT2.DAT')
READ(8,10)FNAME1

OPEN(UNIT=1,TYPE='OLD',NAME='DECO.DAT')
OPEN(UNIT=3,TYPE='NEW',NAME='ZHAT.DAT',CARRIAGECONTROL='LIST')
OPEN(UNIT=4,TYPE='OLD',NAME=FNAME1)

READ(1,20)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
FORMAT(6I5,15X,6AI)

OPEN(UNIT=2,TYPE='OLD',NAME='RECSAM.DAT')
READ(2,88)NSAMP
FORMAT(15)
NSPSAM = 2 * NSAMP

C

TYPE = 'TYPE THE HEADER FOR RECEIVER OUTPUT FILE: '

ACCEPT 22,HD
FORMAT(48A1)
WRITE(3,22)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD

C

C
READ(1,29,END=40)(Q(J),J=1,NSAMP)
FORMAT(1612)
30 FORMAT(1615 )
CONTINUE

READ(8,*)CLPP
READ(8,*)IOPT2
READ(8,*)KBLKI,ITMIN,ITMAX

CALL INSTRT
NRMMG=MOD(NSAMP,256)
DO 65 IBGN=1,NSAMP
CALL ARCR(Q(IBGN),YY)
IRI=MODM(IBGN)
IF(YY.GT.5.5)YY=YY+5.5
IF(YY.LT.5.5)YY=YY-5.5
IF(YY.GT.2047.5)YY=2047.5
IF(YY.LT.-2048.5)YY=-2048.5
IF(IBGN.NE.1 .AND. IRI.EQ.1)WRITE(3,30)IZHAT
IZHAT(IBGN)=IFIX(YY)
IF(IBGN.EQ.NSAMP)WRITE(3,35)(IZHAT(K),K=1,NRMNG)
65 CONTINUE
REWIND 3
READ(3,25,END=ES)(0(3) ,3-1,NSAMP)
CONTINUE

CLOSE(UNIT=3)
OPEN(UNIT=3,TYPE='NEW',NAME='SHAT.DAT',CARRIAGECONTROL='LIST')
WRITE(3,25)NSENT,IRATE,NSAMP,UPPR,ILOWR,NTERMS,HD

BUFFER HAS THE RECEIVED COMPRESSED SPEECH. PITCH WILL BE EXTRACTED FROM IT AND WILL BE USED FOR EXPANSION.

NUMP = 10
IOFF = 0
ICNTT = 0
JJ = 1
NADD = 0
NUMI = KBLKI + ITMAX

DO 960 I=1,NUM1
IBUFF1(I) = Q(30*1)
CONTINUE
IOFF = IOFF + NUM1
ICNTT = IOFF
NSINB=NUM1
CONTINUE
CALL PITCH(IBUFF1,NPR,ITMAX,ITMIN,KBLKI,NUMP,NSINB,IOPT2)
PITCH(JJ) = NPR
JJ = JJ + 1
NADD = NADD + NPR
NUM2 = NUM1 - NPR

DO 980 I = 1, NUM2
IBUFF2(I) = IBUFF1(NPR+I)
CONTINUE

NSP = NUM1 - NUM2
DO 990 I = 1, NSP
ICNTT = ICNTT + 1
IF (ICNTT .GT. NSAMP) GOTO 999
IBUFF2(NUM2+I) = Q(388+IOFF+I)
CONTINUE

NSINB=NUM2+NSP
IOFF = IOFF + NSP
CALL PICH(IBUFF2,NPR,ITMAX,ITMIN,KBLK1,NUMP,NSINB,IOPT2)
PITCH(JJ) = NPR
JJ = JJ + 1
NADD = NADD + NPR
NUM2 = NUM1 - NPR

DO 9100 I = 1, NUM2
IBUFFICI) = IBUFF2(NPR.I)
CONTINUE

DO 9110 I = 1, NUM2
ICNTT = ICNTT + 1
IF (ICNTT .GT. NSAMP) GOTO 999
IBUFF1(NUM2+I) = Q(388+IOFF+I)
CONTINUE

NSINB=NUM2+NSP
GOTO 999
CONTINUE

P2 = 388
ICNT = 1, NUMP
NP = PITCH(ICNT)
P1 = P2 - NP
P3 = P2 + NP
P4 = P3 + NP
P2 = P3
DO 88 I = 1, (P4-P1-1)
IP11 = I + P1 + 1
IF (IP11 .GT. NSAMP) GOTO 688
YHAT(I+1) = Q(IP11)
CONTINUE

CONTINUE
NP2 = NP + NP
DO 188 I = 1, NP2
H(I) = 1 - FLOAT(I-1)/FLOAT(NP2-1)
CONTINUE

DO 128 I = 1, NP2
II = I
SHAT(II) = YHAT(II) + H(II) * (YHAT(II+NP) - YHAT(II))
IARG = NADD + I
MI = MODN(IARG)
XX = SHAT(II)
IF(XX .GT. 8.8) XX = XX+8.8
IF(XX .LT. 8.8) XX = XX-8.8
IF(XX .GT. 2847.8) XX = 2847.8
IF(XX .LT. -2048.8) XX = -2048.8
SQ(N1) = IFIX(XX)
120 CONTINUE
IF(N1 .EQ. 16) WRITE(3,3#) SQ
588 CONTINUE
NADD = NADD + NP + NP

688 TYPE *, ' TYPE THE HEADER FOR PRINTOUT. (40 CHAR ONLY)'
ACCEPT 775, HD
775 FORMAT(48A1)
WRITE(6,774) HD
774 FORMAT('/**/ ,48A1, /**/)
WRITE(6,776) HD
776 FORMAT('/**/ , 'SAMPLE NUMBER ', 6X, ' PITCH PERIOD /**/
ISTRT = 1
IEND = KBLK1+ITMAX
DO 777 I=1,NUMP
  JP=PITCH(I)
  WRITE(6,778) ISTRT, IEND, IP
ISTRT=ISTRT+IP
IEND=IEND+JP
777 CONTINUE
778 FORMAT(6X,15.1X, '-', 1X, 15, 18X, 13)
STOP
END
C
C INVERSE QUANTIZER

SUBROUTINE INVQUA(QQ,EQ)
INTEGER QQ
COMMON /QUAN/T(25),OUT(25),EXPN(25),SIZE,SMIN,NQ
COMMON /RMS/RMS
ISIGN=I
IF(MOD(QQ,2).EQ.0) ISIGN=1
J=(QQ+2)/2
EQ=ISIGN*OUT(J)*SIZE
SIZE=EXPN(J)*SIZE
SIZE=AMAXI(SIZE,RMS*SMIN)
RETURN
END

C
C IN INITIALIZATION

SUBROUTINE INSTRT INITIALIZES THE PARAMETERS.
C
SUBROUTINE INSTRT
COMMON /QUAN/T(25),OUT(25),EXPN(25),SIZE,SMIN,NQ
COMMON /RMS/RMS
EC
1, ISTAT(14),EP,ALAD
COMMON /RMS/RMS
READ(4,*) ALAD
COMMON /RMS/RMS
READ(4,*) N,SAM,ALAD,G,N,RMSMIN,SMIN
C
WRITE(6,2) ALAD,G,N,RMSMIN,SMIN
2 FORMAT(/**/, 'AINV=', FS.2,2X,'ALP=', FS.2,2X,'ALAD=', FS.2,2X,'G=',
FS.3,2X,'N=', FS.2,2X,'RMSMIN=', FS.1,2X,'SMIN=', FS.2/)
READ(4,*) EQ
WRITE(6,3) EQ
3 FORMAT(6X,'NUMBER OF QUANT LEVELS=',I2)
   NO=KQ/2
   NOQQ=NO+1
   READ(4,*)((EXPN(I),I=1,NOQQ)
   WRITE(6,4)(I,EXPN(I),I=1,NOQQ)
   WRITE(6,4)(I,OUT(I),I=1,NOQQ)
4 FORMAT(6X,6('EXPN(',I2,')=',F6.2,2X))
   READ(4,*)((OUT(I),I=1,NOQQ)
   WRITE(6,5)(I,OUT(I),I=1,NOQQ)
5 FORMAT(6X,6('OUT(',I2,')=',F6.2,2X))
   SIZE=180.
   DO 119 I=1,12
   VHAT(I)=0.
   RMS=RMSMIN
   A(I)=AINV
   RETURN
   END
   
C ************************************************************
*  AR Cr E C REI V E R  *
*  ************************************************************
C SUBROUTINE ARCR(Q,VHAT)
   INTEGER Q
   COMMON /PRED/G,N,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),VHAT(12),
   1 EV,ISTATI(4B),EP,ALAD
   COMMON /RMS/RMS
C PREDICTION.
   PRE=0.
   DO 125 I=1,N
  125 PRE=PRE+A(I)*VHAT(I)
   RMS=ALP*(RMS-RMSMIN)+(1.-ALP)*ABS(VHAT(I))+RMSMIN
   CALL INVQUA(Q,EQ)
   DO 125 I=1,N
   J=N+2-I
   VHAT(J)=VHAT(J-1)
   VHAT(1)=PRE*EQ
   VHAT1=VHAT(1)
C ADAPTATION.
   ERR=G*EQ/RMS**2
   A(I)=A(I)+AINV*(1./ALAD-1.)
   DO 130 I=1,N
  130 A(I)=A(I)*ALAD+ERR*VHAT(I+1)
C RETURN
   END
   
C ************************************************************
*  PITCH EXTRACTION  *
*  ************************************************************
SUBROUTINE PICH (IBUF, NP, ITMAX, ITMIN, KBLK1, NUMP, NBUFF, IOPT2)
DIMENSION A(200), IBUFF(512), IA(200), IOPT2
COMMON /CLIPP/ CLPP
NUMP = NUMP + 1
N1 = NBUFF/3
N2 = N1 + N1
DO 5 I = 1, NBUFF
IBUFF(I) = IBUF(I)
CONTINUE
DO 10 I = 1, 200
A(I) = 0.0
IA(I) = 15
CONTINUE
C------ CHECK IF CENTER CLIPPING IS ASKED FOR.
C IF(IOPTZ .LE. 2) GOTO 280
C------ FIND OUT ABSOLUTE MAXIMUM OUT OF NBUFF SAMPLES IN THE BUFFER*IBUFF*
IBIG1 = $1
IBIG2 = $1
DO 120 I = 1, N1
ISPABS = ABS(IBUFF(I))
IF(ISPABS .GT. IBIG1) IBIG1 = ISPABS
CONTINUE
DO 123 I = N2, NBUFF
ISPABS = ABS(IBUFF(I))
IF(ISPABS .GT. IBIG2) IBIG2 = ISPABS
CONTINUE
IB = IBIG1 - IBIG2
IF( IB .GE. 0) IBIG = IBIG2
IF( IB .LT. 0) IBIG = IBIG1
C------ ENTER THE CLIPPING LEVEL.
CL = CLPP - FLOAT(IBIG)
C------ CHECK IF CENTER CLIPPING WITH THREE OR TWO VALUES IS REQUIRED.
C IF(IOPT2 .GT. 4) GOTO 155
C------ CLIPP THE SPEECH WAVEFORM.
CLM = CL
DO 145 I = 1, NBUFF
XFLT = FLOAT(IBUFF(I))
IF((XFLT .LE. CL) .AND. (XFLT .GT. CLM)) IBUFF(I) = 0
CONTINUE
GOTO 280
155 CONTINUE
IF(IOPT2 .GT. 6) GOTO 360
CLM = CL
DO 165 I = 1, NBUFF
XFLT = FLOAT(IBUFF(I))
IF((XFLT .LE. CL) .AND. (XFLT .GT. CLM)) IBUFF(I) = 0
IF(XFLT .LT. CL) IBUFF(I) = 1
IF(XFLT .LE. -CL) IBUFF(I) = -1
CONTINUE
IF(IOPT2.GT.5)GOTO 220

C------- COMPUTE AUTOCORRELATION FUNCTIONS

IBIGG=1000
DO 180 IT=ITMIN,ITMAX
   ISUM=0
   DO 170 J=1,KBLK1
      IF(((IBUFF(J+IT).LT.0).AND.(IBUFF(J).LT.0)).OR.((IBUFF(J+IT)
         .GT.0).AND.(IBUFF(J).GT.0)))ISUM=ISUM-1
      IF(((IBUFF(J+IT).LT.0).AND.(IBUFF(J).GT.0)).OR.((IBUFF(J+IT)
         .GT.0).AND.(IBUFF(J).LT.0)))ISUM=ISUM+1
   CONTINUE
   IA(IT)=ISUM
   IF(IBIGG.GT.IA(IT))IBIGG=IA(IT)
CONTINUE
DO 190 I=ITMIN,ITMAX
   IF(IBIGG.EQ.IA(I))GOTO 280
CONTINUE
NP=I
WRITE(*,*)NP
RETURN

C------- CALCULATE AMDF FUNCTIONS

220 CONTINUE
   ISMALL=4996
   DO 230 IT=ITMIN,ITMAX
      J=1,KBLK1
      IF(((IBUFF(J+IT).EQ.0).AND.(IBUFF(J).NE.0)).OR.
         ((IBUFF(J+IT).NE.0).AND.(IBUFF(J).EQ.0)))ISUM=ISUM+1
      IF(((IBUFF(J+IT).EQ.0).AND.(IBUFF(J).NE.0)).OR.
         ((IBUFF(J+IT).NE.0).AND.(IBUFF(J).EQ.0)))ISUM=ISUM+1
   CONTINUE
   IA(IT)=ISUM
   IF(ISMALL.GE.AI(T))ISMALL=IA(IT)
230 CONTINUE
   DO 250 I=ITMIN,ITMAX
      IF(ISMALL.EQ.AI(I))GOTO 260
CONTINUE
NP=I
WRITE(*,*)NP
RETURN

250 CONTINUE
260 CONTINUE
   IF((IOPT2.EQ.2).OR.(IOPT2.EQ.4))GOTO 340
BIG=0.0
   DO 290 IT=ITMIN,ITMAX
      SUM=0.0
      DO 280 J=1,KBLK1
         SUM=SUM+FLOAT(IBUFF(J+IT))*FLOAT(IBUFF(J))
      CONTINUE
      A(IT)=SUM
      IF(BIG.LT.A(IT))BIG=A(IT)
   280 CONTINUE
   DO 310 I=ITMIN,ITMAX
      IF(BIG.EQ.AI(I))GOTO 320
   CONTINUE
NP=I
WRITE(*,*)NP
RETURN
310 CONTINUE
320 CONTINUE
340 CONTINUE
   SMALL = 1.0E-8
   DO 60 IT=ITMIN,ITMAX
      SUM = 0.0
   CONTINUE
60 CONTINUE
DO 50 J = 1, KBLK1
  CONTINUE
  SUM = SUM + ABS(FLOAT(IBUFF(J+IT) - IBUFF(J)))
  A(IT) = SUM
  IF(SMALL .GE. A(IT)) SMALL = A(IT)
  DO 70 I = ITMIN, ITMAX
    IF(SMALL .EQ. A(I)) GOTO 80
  CONTINUE
  CONTINUE
  NP = 1
  WRITE(*,'*'NP
  RETURN
C
CLIPP THE SPEECH TO TWO VALUES.
CL = .8
DO 360 I = 1, NBUFF
  XFLOT = FLOAT(IBUFF(I))
  IF(XFLOT .LE. CL) IBUFF(I) = --
  IF(XFLOT .GT. CL) IBUFF(I) = 1
360 CONTINUE
  IF(IOPT2 .GT. 7) GOTO 480
  IBIGG = -1055
  DO 420 IT = ITMIN, ITMAX
    ISUM = 0
    DO 460 J = 1, KBLK1
      IF(IBUFF(J+IT) .EQ. IBUFF(J)) ISUM = ISUM + 1
      IF(IBUFF(J+IT) .NE. IBUFF(J)) ISUM = ISUM - 1
      IA(IT) = ISUM
    IF(IBIGG .LT. IA(IT)) IBIGG = IA(IT)
    CONTINUE
  DO 420 IT = ITMIN, ITMAX
    ISUM = 0
    DO 460 J = 1, KBLK1
      IF(IBUFF(J+IT) .EQ. IBUFF(J)) ISUM = ISUM + 1
      IF(IBUFF(J+IT) .NE. IBUFF(J)) ISUM = ISUM - 1
      ISMALL = 4896
      DO 500 IT = ITMIN, ITMAX
        ISUM = 0
        DO 490 J = 1, KBLK1
          IF(IBUFF(J+IT) .NE. IBUFF(J)) ISUM = ISUM + 2
100 CONTINUE
  ISMALL = 4896
  CONTINUE
  IA(IT) = ISUM
  IF(ISMALL .GE. IA(IT)) ISMALL = IA(IT)
  DO 520 IT = ITMIN, ITMAX
    IF(ISMALL .EQ. IA(IT)) GOTO 540
  CONTINUE
  IF(IOPT2 .GT. 7) GOTO 480
  NP = 1
  WRITE(*,'*'NP
  RETURN
C
C
REFERENCES


160


