

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

# AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/90D-38

# RULE-BASED FREQUENCY DOMAIN SPEECH CODING

THESIS

Vance M. McMillan
Captain, USAF

AFIT/GE/ENG/90D-38



Approved for public release; distribution unlimited.

AFIT/GE/ENG/90D-38

# RULE-BASED FREQUENCY DOMAIN SPEECH CODING

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Vance M. McMillan, B.S.E.E
Captain, USAF

December, 1990

## *Preface*

The "AFIT experience" was a difficult experience for me, however it was not nearly as difficult as it would have been without the tremendous amount of help and support I received. Dr Kabrisky, Maj Rogers, and Capt George supplied the intellectual support and doubled as my advisory committee. They pulled the best out of me and Dr K kept me from panicking when things didn't go as well as I expected. The computer support people throughout AFIT were great but a special thanks goes to Dan Zambon and Dave Doak for not laughing too much at my computer illiteracy and making sure I didn't hang myself too many times. The academic help came from a group of friends in the Comm-Radar track, namely Scott S., Becky, Joe, Al and Scott T.. They also supplied the motivation when the "Stop Me When I Have a B" attitude set in. The most important support came from my family. Thanks Jason, Megan, and Anne for knowing that I love you even when I didn't speak to you for a week. Thanks Debbie for putting up with a husband who for 18 months was grumpy the little time he was around.

Vance M. McMillan

# *Table of Contents*

## *List of Figures*

AFIT/GE/ENG/90D-38


## *Abstract*

A speech processing system is designed to simulate the transmission of speech signals using a speech coding scheme. The transmitter portion of the simulation extracts a minimized set of frequencies in Fourier space which represents the essence of each of the speech timeslices. These parameters are then adaptively quantized and transmitted to a receiver portion of the coding scheme. The receiver then generates an estimate of the original timeslice from the transmitted parameters using a sinusoidal speech model. After initial design, the thesis investigates how each of the design parameters affect the human perceived quality of speech. This is done with listening tests. The listening tests consist of having volunteers listen to a series of speech reconstructions. Each reconstruction is the result of the coding scheme acting on the same speech input file with the design parameters varied. The design parameters which are varied are: number of frequencies used in the sinusoidal speech model for reconstruction, number of bits to encode amplitude information, and number of bits used to code phase information. The final design parameters for the coding scheme were selected based on the results of the listening tests. Post design listening tests showed that the system was capable of 4800 bps speech transmission with a quality rating of five on a scale from zero (not understandable) to ten (sounds just like original speech).

# RULE-BASED FREQUENCY DOMAIN SPEECH CODING

## *I.* Introduction

The search for more robust and bandwidth efficient ways to transmit speech signals has recently led researchers into the speech analysis/synthesis area (1, 2, 9). These systems have an advantage over the established analog speech transmission systems in the area of noise rejection; therefore, they are more robust. However, the analog transmission techniques have the more modest bandwidth requirements of about 8 KHz for speech. Speech analysis/synthesis techniques usually must be coupled with digital signal transmission systems that typically require a greater system bandwidth to communicate the same information in digital format as compared to an analog format (13:4). Therefore the new analysis/synthesis coding schemes must be able to code speech more efficiently than the analog coding systems or they will not be a viable alternative to the analog systems. Therefore, the next step in the analysis/synthesis research area is to ensure that the coding schemes are as bandwidth efficient as possible.

### *1.1* Background

In March 1989, Nadeem A. Bashir published his thesis on the reduction of noise in speech signals. In his thesis, Bashir reconstructed speech signals by first estimating the speakers glottal frequency, then transmitting the amount of energy measured in frequency bands corresponding to exact harmonics of the glottal frequency estimate. The problem was, if exact harmonics of the glottal frequency were chosen for reconstructing the speech, a noise, which sounded like music playing in

the background corrupted the reconstructed speech (see Literature Review for an in-depth analysis of the Bashir system). To eliminate the "musical noise" effect, he selected the highest energy frequency within the nearest four neighbors of the harmonics of the glottal frequency estimate. He then reconstructed the speech from this spectra. After implementing this algorithm, Bashir was able to reconstruct speech signals which were perceived to be indistinguishable from the original speech by human listeners (2). The ability of this system to reconstruct pristine speech from selected frequency components made it a candidate for research as a possible midband or lowband rate speech coding system.

In March 1990, Capt. M. F. Alenquer was successful in adapting the noise reduction scheme, demonstrated by Bashir, as a speech coding and transmission scheme. He also began the search for the lowest possible data rate for the system. The expression for finding the data rate of the system was found to be:

$$DataRate = 2L/T(A + P + 8) \tag{1}$$

Where:

- L is the number of components per frame.
- A is the number of bits transmitted to convey amplitude information.
- P is the number of bits transmitted to convey the phase information.
- T is the frame time duration.

Alenquer reduced the system data rate by minimizing the three parameters A, P and L through subjective listening test. He was able to obtain speech reconstruction data rates as low as 18 Kbps (1).

## 1.2 Problem

The objective of this thesis is to implement a speech analysis/synthesis coding system based on the work of Bashir and Alenquer which transmits low bit rate speech signals at near toll quality.

## 1.3 Research Questions

The following is a list of research questions which will be addressed in this thesis:

1. Can the system be modified to accept a lower data rate by transmitting the glottal frequency (8 bits), then transmitting the relative position of the next selected frequency (2 bits)? If this is possible the data rate equation will change from Equation (1) to:

$$DataRate = 2/T(A + P + 8) + 2/T(L - 1)(A + P + 2), \qquad (2)$$

   leading to a substantial data rate reduction.

2. How does the perceived reconstucted speech quality change as a function of the design parameters?

3. Can an adaptive quantization scheme be developed which matches itself to the expected value of the amplitudes of the spectral components?

## 1.4 Assumptions

The following assumptions are made in this thesis:

1. As described by Alenquer, the best speech reconstruction results occur while using the four nearest neighbors, frequency-selection algorithm (1:4-2).

2. Alenquer's algorithm performs as he intended and produces the effect on speech as described in his thesis (1).

3. Any noise which occurs when the system design parameters of the algorithm are varied is the result of information loss and not the result of Hamming window leakage.

## *1.5* Scope

A speech transmission system is designed based on the work of Alenquer and Bashir using frequency domain analysis techniques. The speech is reconstructed from the frequency, amplitude, and phase of selected spectral components (1:1-5). The spectrum of the speech is obtained by taking a 1024-point DFT of time framed speech. The signal is reconstructed by inverse transforming the vectors containing the selected frequency components.

## *1.6* Approach

The first step of the thesis will be to design a speech coding system which is based on a combination of the characteristics of Alenquer and Bashir's work. The speech processing system will be software designed to simulate the transmitter and receiver in a digital communications link. The coding system will exploit the characteristics of the speech spectra including:

1. Speech energy rolls off at 6 dB per octave above 600 Hz.(12)
2. Speech spectra are stationary for up to 40 ms time periods.(10)
3. The basic shape of the spectra remains approximately constant for all voiced speech signals, only the locations of the maxima change.

The second step of this thesis will be to establish a high fidelity audio tape containing several reproductions of an utterance. Each reproduction will be the result of the speech processing system operating on the same utterance, but with the system parameters varied. The system parameters to be varied are components per frame (L), bits transmitted for amplitude information (A), and bits transmitted

for phase information (P). The tape will then be played for a number of listeners who will be asked to rate each utterance on a scale form zero(could not understand the utterance) to ten (sounded just like original speech). This information will be used to develop a family of curves which will show how the quality of reproduced speech is affected by varying each of the design parameters one parameter at a time.

In the final step of this thesis, a "best guess" will be made of the parameters A, L, and P which will minimize the data rate of the system without greatly diminishing the quality of speech based on the listening test results. These parameters will be implemented in the speech processing system final design.

## *1.7* Definitions

1. **Glottis** is the opening between the vocal cords. (7)
2. **Voiced** sounds are sounds produced by forcing air through the vocal cords, thereby producing quasi-periodic pulses of air which excite the vocal tract. (11)
3. **Unvoiced or Fricative** sounds are sounds produced by forcing air through a constriction in the vocal tract, thereby producing turbulence. The resulting broad-spectrum noise excites the vocal tract. (11)
4. **Formants** are resonances in the vocal tract. (7)
5. **Glottal or Fundamental Frequency** is the rate at which the glottis opens and closes during voice speech. (7)

## *1.8* Sequence of Presentation

Chapter II presents a literature review of recent work in the area of speech transmission systems at both AFIT and MIT Lincoln Laboratory.

Chapter III gives a description of the speech reconstruction processing system hardware and software environment.

Chapter IV describes the algorithm.

Chapter V describes the System Testing.

Chapter VI presents the results, of the tests. Chapter VII gives the final conclusions and recommendations.

The Appendices contain sample data and the computer program source code.

# *II.* Literature Review

## *2.1* Introduction

Since the beginning of the use of electrical signaling techniques, one of the questions which has concerned the telecommunications engineer is how to send more information on the limited resources available. Cherry describes one telecommunications engineer who found a way to better use existing resources. His name was Samuel Morse. After inventing the telegraph, Morse analyzed the use of the alphabet to determine which letters were used the most. He then assigned the shortest codes to the letters most often used, thus maximizing the use of the limited telegraph resources.(3:37) In 1928, Hartley showed that to transmit a "quantity of information," in a given amount of time required a certain minimum amount of bandwidth (3:43,44). With this discovery, the telecommunications engineer was doomed forever to consider the tradeoff between time of transmission and channel bandwidth. The quest to find more ingenious ways to use the resources available intensified.

One area of particular interest for maximizing the use of resources is the speech transmission area. Cherry writes:

> ...in order to obtain more economic transmission of speech signals, in view of the bandwidth x time law, something drastic had to be done to the speech signals themselves to remove those elements which do not contribute markedly to the speech intelligibility. These considerations led to what is known as the vocoder, an instrument for analyzing, and subsequently resynthesizing, speech; a "talking machine" which requires only control signals to be transmitted and received in order to reproduce intelligible speech (3:45).

Vocoders have been refined and made more efficient since Cherry wrote this passage. However, the search for the most efficient means to transmit voice signals continues because todays communications engineers are faced with the same dilemma as their

predecessors: find better ways to use existing transmission resources or go through the costly and time consuming effort of installing new systems.

This literature review will examine some of the important factors involved in efficient speech encoding. Section 1.2 will describe how voice coders are categorized into one of two broad categories; waveform coders or synthesis/analysis coders. Section 1.3 will cover the spectral properties of speech which make it possible to code speech for transmission by selecting certain limited number of parameters for transmission to a receiver. Section 1.4 will review recent research efforts in the field of sinusoidal encoding of speech. Finally, the chapter will conclude with a brief summary of the Literature Review.

## 2.2 Speech Coding

Speech coding systems can be categorized as either waveform coders or synthesis/analysis coders(13:649). Waveform coders estimate the input signal amplitude during a given time interval. The coder then transmits the estimate to the decoder at the receiving end of the system and an estimate of the original signal is reproduced. Examples of waveform encoders include Pulse Code Modulation (PCM) coders and Differential Pulse Code Modulation (DPCM) coders (1:1-2).

Synthesis/analysis coders approach signal estimation in a different way. In this scheme, the characteristics of the power spectra of the signal to be transmitted are exploited. If the power spectrum of the signal is constant over a short period of time, the coder can select the characteristics of the short-term spectra which contain the speech information. These characteristics may then be transmitted to the decoder at the receiving end of the system and the signal is reproduced from its spectra. Systems which use the synthesis/analysis to transmit speech signals are called vocoders (13:64$^{Q}$).

The first vocoder was described by H. Dudley of Bell Labs and was demonstrated at the 1939 New York Worlds Fair (13:651). The *channel coder* consisted of a

bank of narrowband filters, detectors, and averagers, which form a local estimate of the power distribution over the frequency band (13:651,652). The speech was then reconstructed from the estimates of the power distribution.

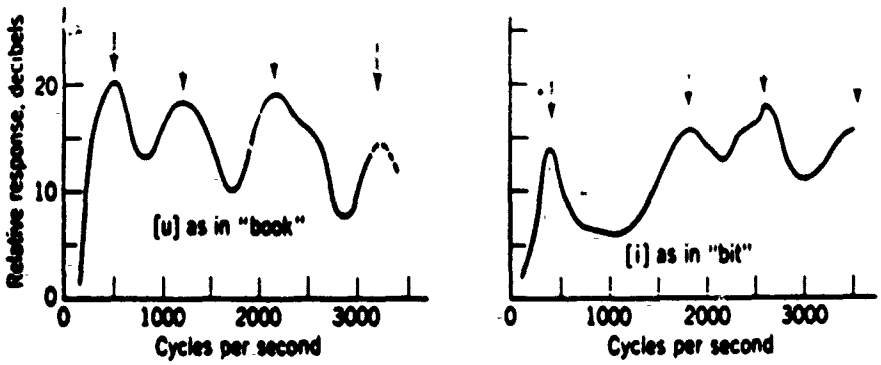
## 2.3 Spectral Properties of Speech

As described by Cherry, voiced speech may be modeled as a two part process. In the initial part of the process, the vocal cords generate a steady tone. The tone is periodic and therefore has a periodic Fourier (line) spectrum of harmonics. The frequency spacing between the lines in the spectra depends on the speaker. Males have a lower frequency voice; thus, the line spacing is closer than in a female voice. In the second part of the process, the tone generated by the vocal cords is acted upon by the resonances of the vocal tract. These resonances vary depending on the voiced sound being generated. The frequencies where the resonances occur are called formants (3:155,156).

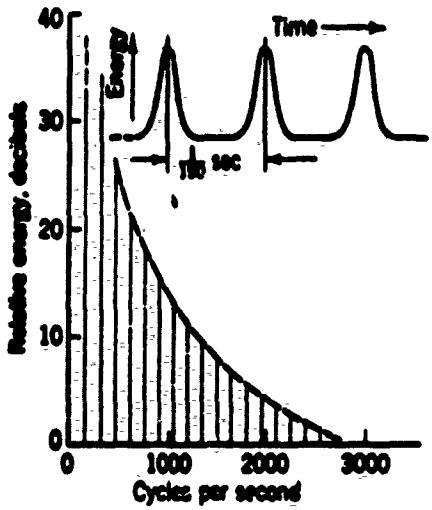
When generating unvoiced speech, humans do not use the larynx to generate a glottal tone. Therefore, unvoiced speech does not have the periodicity of voiced speech. The energy spectra of unvoiced speech closely resembles the uniform distributed spectra of noise. An example of unvoiced speech is the "s" sound (5).

Figure 1a shows the "selective characteristics" of the vocal tract for the vowels "u" and "i". Cherry obtained these characteristics by measuring the response of the vocal tract as the vowel sounds were being whispered. While whispering, the airstream sets up a acoustic spectrum of uniform energy; thus, the response is due solely to the selective characteristics of the vocal tract (3:155). Figure 1b shows the approximate spectrum of the larynx and Figure 1c shows the output spectrum of the voiced vowel "u".

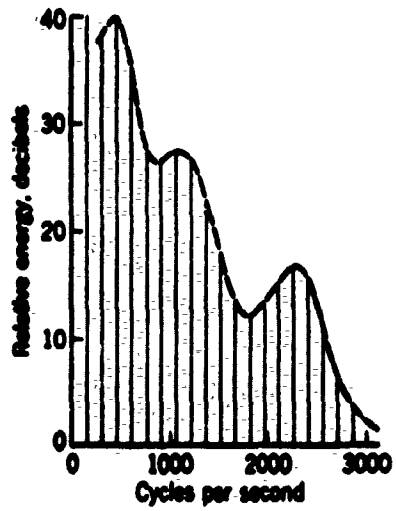
As seen in Figure 1, each vowel has a distinct vocal tract frequency response. This concept has been extended to words and sentences. That is, the frequency response of the vocal tract has a distinct frequency response in time. That this is




(*a*) The "selective characteristics" of the vocal tract, for two sustained vowels, showing formants (corresponding to the energy spectra of the *whispered* vowels)




(*b*) Wave form of larynx source energy and its approximate spectrum




(*c*) Spectrum of phonated (voiced) vowel [u]

Figure 1. Spectra of Sustained Vowels (Male Voice)(3:156)

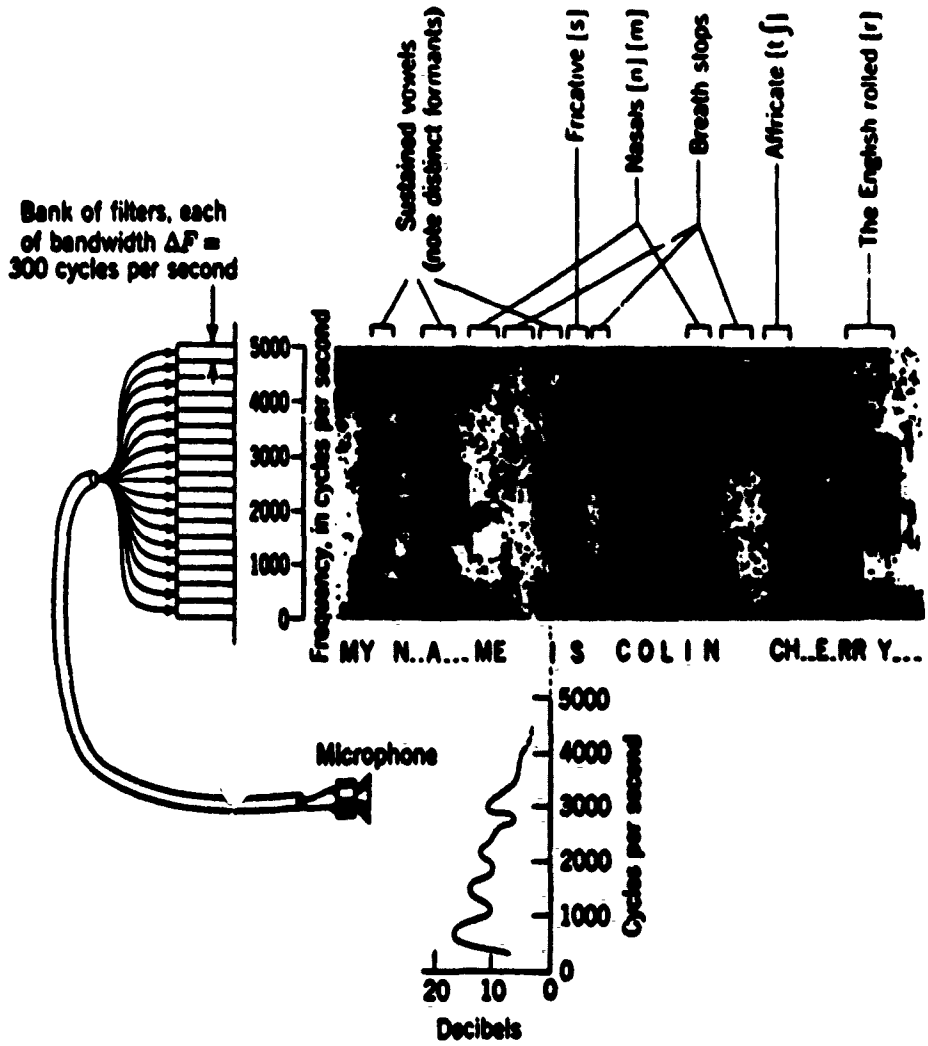



Figure 2. A "Visible Speech" Spectogram (Sonogram)(3:148)

true may be demonstrated with spectograms. Spectograms are representations of the energy spectra of speech for discrete periods of time (3:146). A sample spectogram is shown in Figure 2. One vertical slice of the spectogram in Figure 2 would represent the energy spectrum of Figure 1. It is in the shape of the frequency response of the vocal tract that the information lies (5).

Sundberg states:

> Moving the articulary organs is what we do when we speak and sing; in effect we chew the standing waves of our formants to change their frequencies. Each articulary configuration corresponds to a set of formant frequencies, which in turn is associated with a particular sound. (14:84)

Thus, we have information lying in the continuously changing frequency response of the human vocal tract. The spectral properties of this speech appear to be stationary for 20 to 50 ms time intervals (13:649). There is an inherent redundancy in speech which allows the listener to understand mutilated speech or speech corrupted by noise (2:1-1). The problem that must now be solved is how to select characteristics of speech for transmission which carry the the necessary information, but are not redundant (5). This reduced set of information may then be sent on the transmission facilities, thus maximizing the use of the voice transmission facilities.

## *2.4* Sinusoidal Encoding of Speech

One model which exploits the characteristics of the spectrum of speech is the sinusoidal speech model. According to McAulay and Quatieri, this model is characterized by speech that is assumed to be the result of passing the glottal excitation input through a linear time invariant filter which models the response of the vocal tract. Thus, the output speech waveform $s(t)$ may be estimated by the well known linear time invariant system model:

$$s(t) = \int_0^t h(t-\tau;t)e(\tau)$$

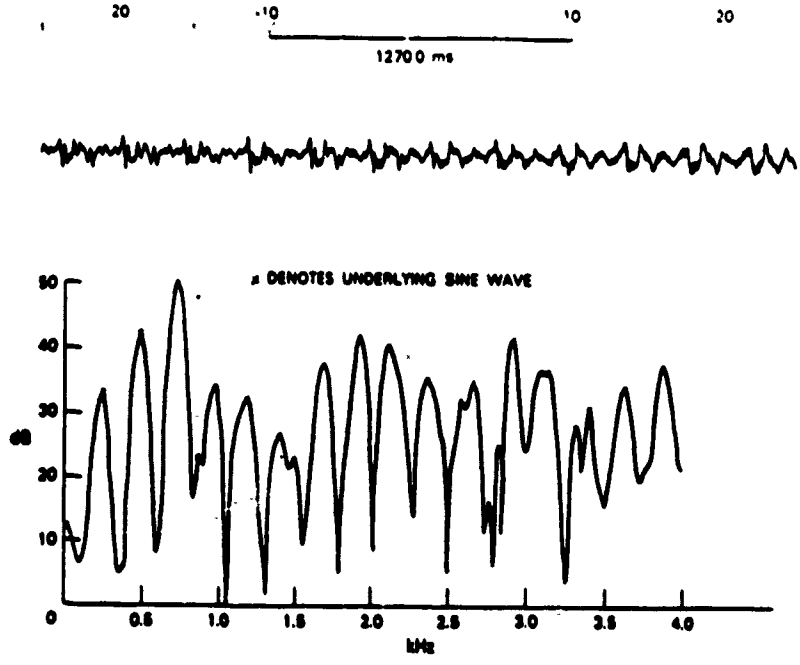



Figure 3. Peak Selection in McAulay and Quatieri System(9:747)

where $h(t)$ is the impulse response of the vocal tract and $e(t)$ is the waveform generated by the glottal excitation (9:744,745). The sinusoidal speech model has been proven to be effective in reproducing speech which is almost indistinguishable from the original speech (1, 2, 9).

Using the sinusoidal speech model, there are several methods of choosing which frequencies in the speech spectogram carry the necessary information. McAulay and Quatieri generated spectograms for intervals of speech 10 ms long. They then extracted the amplitudes, phases, and frequencies of the sinusoidal components of each interval of speech by using a 512-point Fast Fourier Transform (FFT) and an adaptive Hamming window of 2.5 times the average pitch. Frequency selection was accomplished by choosing the peaks within the spectogram as illustrated in Figure 3. This selection technique was used for both voiced and unvoiced speech (9).

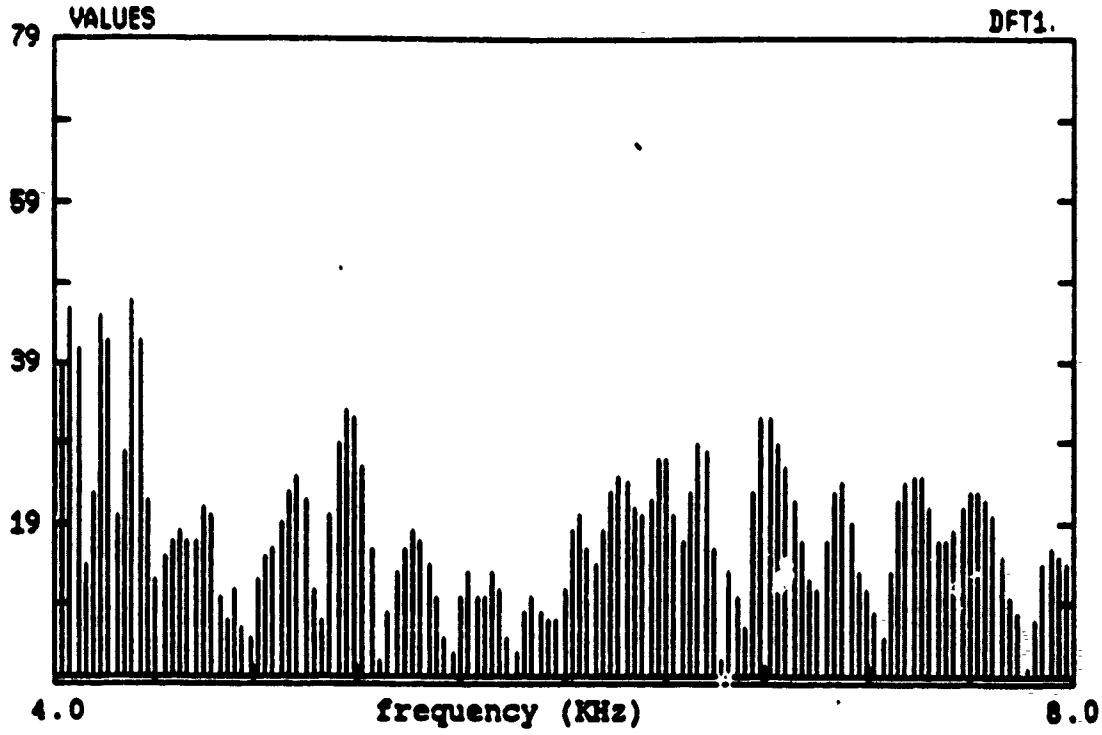
McAulay and Quatieri also used used a sinusoidal *birth/death* philosophy for

frame-to-frame peak matching. In this system, a frequency in frame $k$ is compared to all frequencies in frame $k+1$ in an attempt to find a frequency in frame $k+1$ within a "matching interval" $\Delta$. If a match is found, then the frequency continues to be a selected frequency peak. If more than one match is found, a tiebreaking system defines which frequency is the best match. If a match is not found, the frequency is declared *dead*. After all frequencies from frame $k$ are matched to the frequencies in frame $k+1$, or are declared dead, any left over frequencies in frame $k+1$ are declared *born*. (9:748)

Alenquer and Bashir choose to use two different frequency selection techniques: one for voiced and another for unvoiced speech. Initially, the speech was digitized using an analog to digital converter. This data was then Hamming windowed and a 512-point DFT taken. Frequency selection for voiced speech was based on an estimate of the speaker's glottal frequency. The rule based selection began by finding the highest energy frequency in a region where the glottal frequency was expected to be. Subsequent frequencies were chosen by selecting the highest energy frequency within four nearest neighbors of harmonics of the glottal frequency estimate. A typical spectogram of speech before and after frequency selection are shown in Figure 4. Unvoiced speech frequency selection was accomplished by choosing all frequencies above a variable energy threshold. The threshold was varied up and down until the desired number of frequencies were selected. The threshold also was attenuated 6 dB per octave above 625 Hz to compensate for the fact that energy in speech falls off at this rate (1, 2).

## *2.5* Summary

This chapter has shown that speech coding may be done by several different methods. These methods can be categorized as either analysis/synthesis coding or waveform coders. Coders which use the spectral content of speech to perform analysis/synthesis coding are called Vocoders. Speech spectrum analysis can be



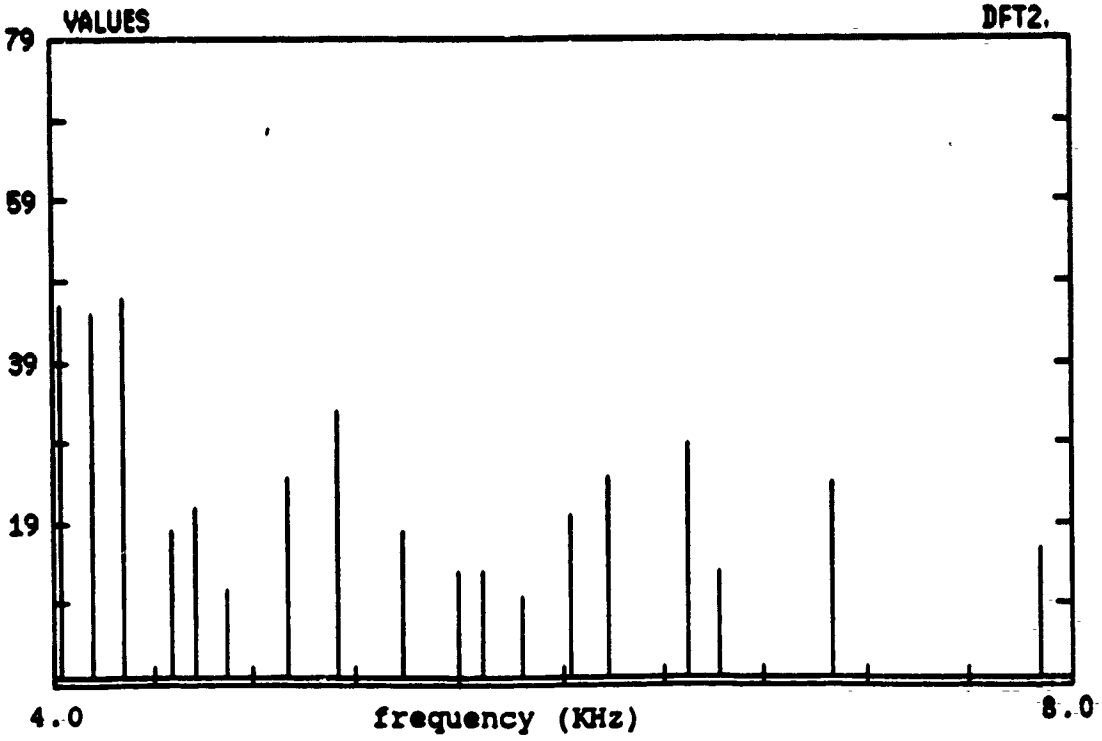




Figure 4. Harmonics Selection(2:3-10)

performed in many ways. McAulay and Quatieri choose to select the spectral peaks for speech analysis. Alternatively, Alenquer and Bashir choose to select frequencies which were neighbors of harmonics of the glottal frequency. Chapter three will explain the experiment environment for this thesis.

# *III.* Development Environment

## *3.1* Introduction

This chapter describes the hardware and software environment used in the development of the speech processing system. The first section describes the setup used for speech recording and processing. The software development environment is explained in the following section. The final section shows the environment for playing back the reconstructed and original speech files. A block diagram of the environment is included in the final section.

## *3.2* Speech Recording and Processing

Analog speech signals were input to the system through a noise reducing microphone. A Rockland Model 852 filter set to low pass filter at 6000 Hz was placed between the microphone and Digital Sound Corporation Analog to Digital (A/D) Converter. The 6000 Hz setting prevented aliasing at the 16,000 samples per second sampling rate. The A/D converter digitized the analog signal to integer format from -32,768 to 32,767. The data was then stored in a VaxStation II for input to the speech processing system.

## *3.3* Software Environment

The speech processing system was written in the 'C' program language on a VAXWorkstation II running the VMS version 5.1 operating system. The program was modularly designed for ease of program redesign and intermediate data extraction. Some computer code Alenquer and Barmore wrote for their thesis efforts is modified and used in the speech processing system. The computer code is presented in appendix B.

## 3.4 Speech Playback

After the speech files were run on the system the reproduced files were played back for listening tests. The speech files stored on the VAXstation II were input to the DSC 200 Digital to Analog (D/A) Converter for playback. The playback signal was routed through the Rockland filter. The filter was set to low pass filter at 4000 Hz because the final design of the speech processing system does not output any frequencies above 4000 Hz. The output was played on either a speaker or headphones.

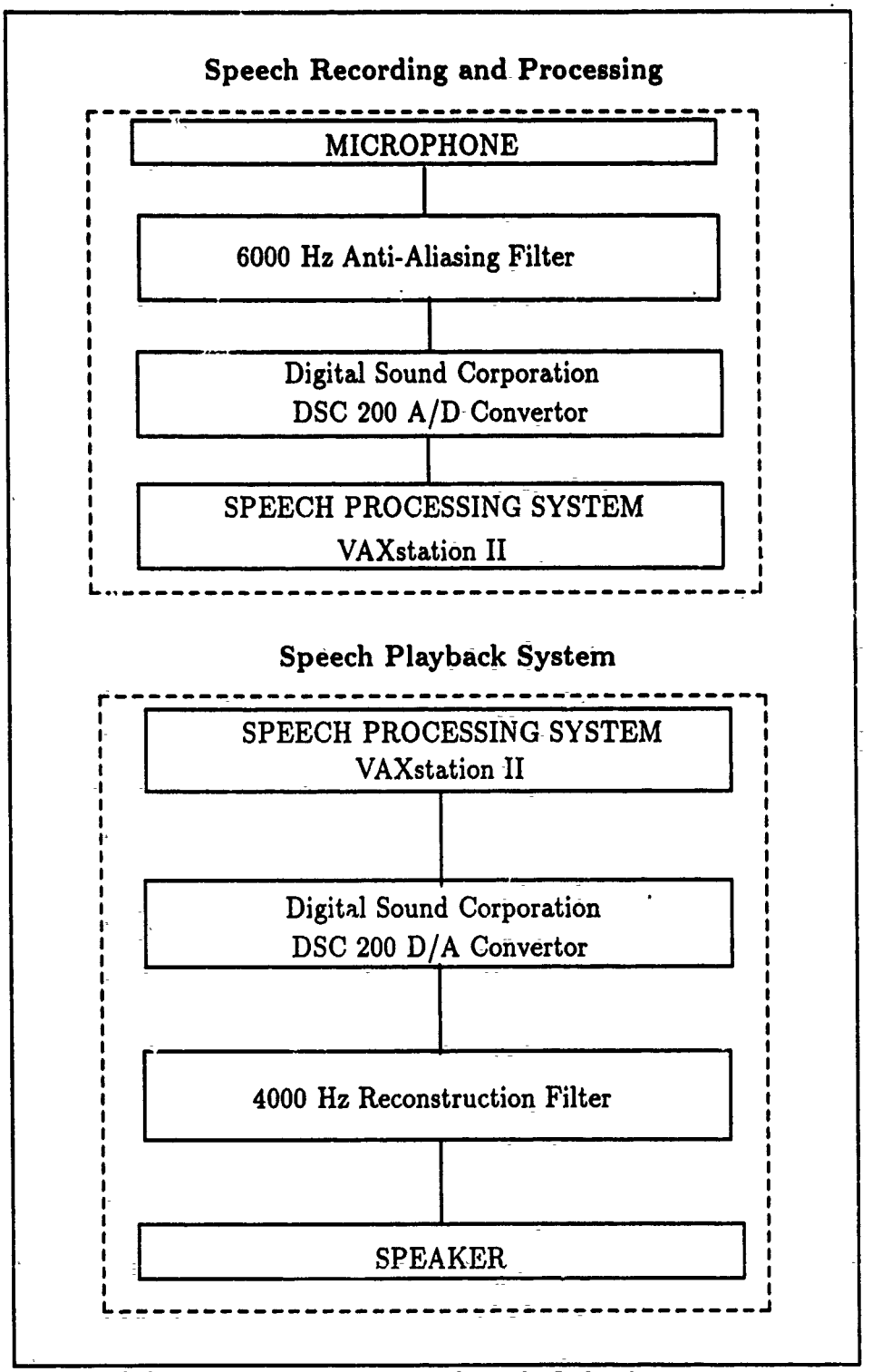



Figure 5. System Block Diagram

# *IV.* Speech Processing System

## *4.1* Introduction

The speech processing system is a computer program designed to simulate the transmission of speech signals at low data rates. The system simulates the transmitter and receiver but not the effects of the transmission channel. The system was designed in three major modules: analysis, coding, synthesis. The analysis module begins by breaking the speech signal into many small time slice signals. The time slices are then Fourier transformed and analyzed to extract a minimized amount of information in the frequency domain. The coding module then quantizes the information extracted by the analysis module for digital signal transmission. The final module, synthesis, then regenerates an estimate of the original speech signal from the reduced set of information.

*4.1.1* **Time Frame Generation** Sample rates of 16 thousand samples per second and 8 thousand samples per second were tried in the development of the system. Most speech spectral energy is generally considered to exist at frequencies below 4 kHz. However, since the system was evaluated on its ability to reproduce quality speech, as much of the original speech energy as possible was desired to ensure all important spectral components were passed. Therefore, energy spectral components above 4 kHz were passed. As expected, aliasing occurred when spectral components above 4 kHz were sampled at 8,000 samples per second. Therefore, the 16 thousand samples per second sample rate was used. In addition, the input speech signals were low pass filtered at 6 kHz to prevent any possibility of aliasing when using the higher sample rate.

Time slices of 30 to 40 ms were desired because speech spectral components are generally stationary for this period of time (5). In order to obtain these time slice periods at a 16 thousand samples per second rate, 512 to 640 samples were taken to

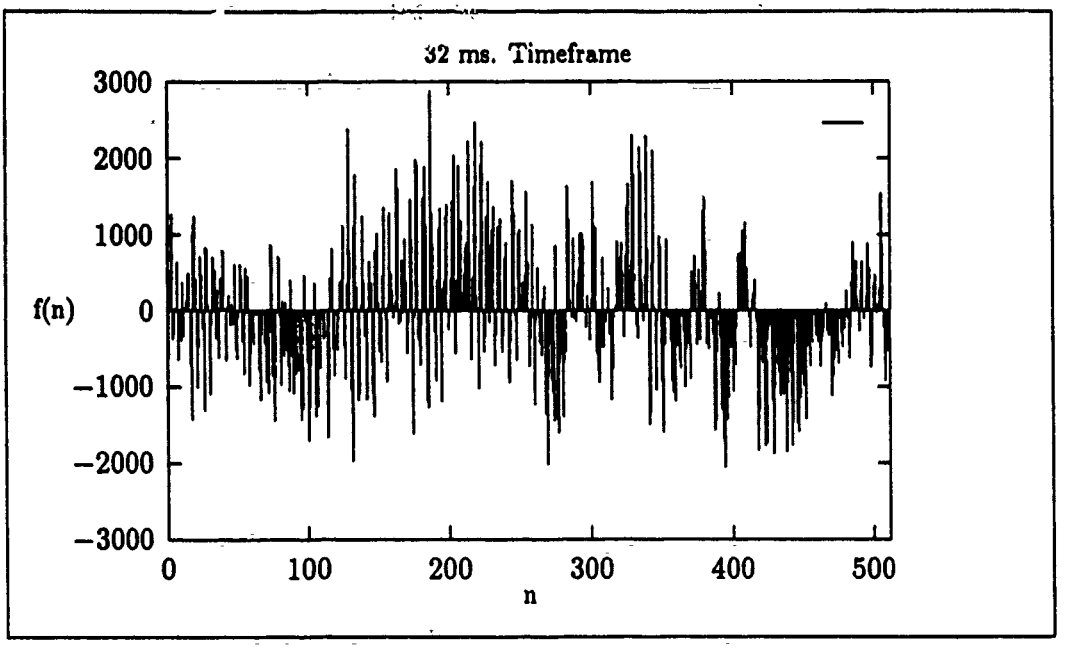



Figure 6. Time Slice Consisting of 512 Sample Points

be one time slice. Each time slice therefore represented 32 to 40 ms of speech. An example of a 512 sample time frame is shown in Figure 6.

*4.1.2* **Windowing** Each time slice was windowed both at the input to the system and at the output of the system. The input windowing was accomplished to prepare the data for the Fast Fourier Transform (FFT). Windowing before the FFT helps reduce side lobe spectral leakage which is due to the finite observation interval. The window chosen for the input was the sine window function seen in Figure 7.

$$W(n) = sin(n\pi/N) \tag{3}$$

Where:

$n = 0, 1, 2, ...N - 1$

Each time slice was also windowed at the output. This window was needed because the system is a non-linear rule based system; therefore, each time slice is

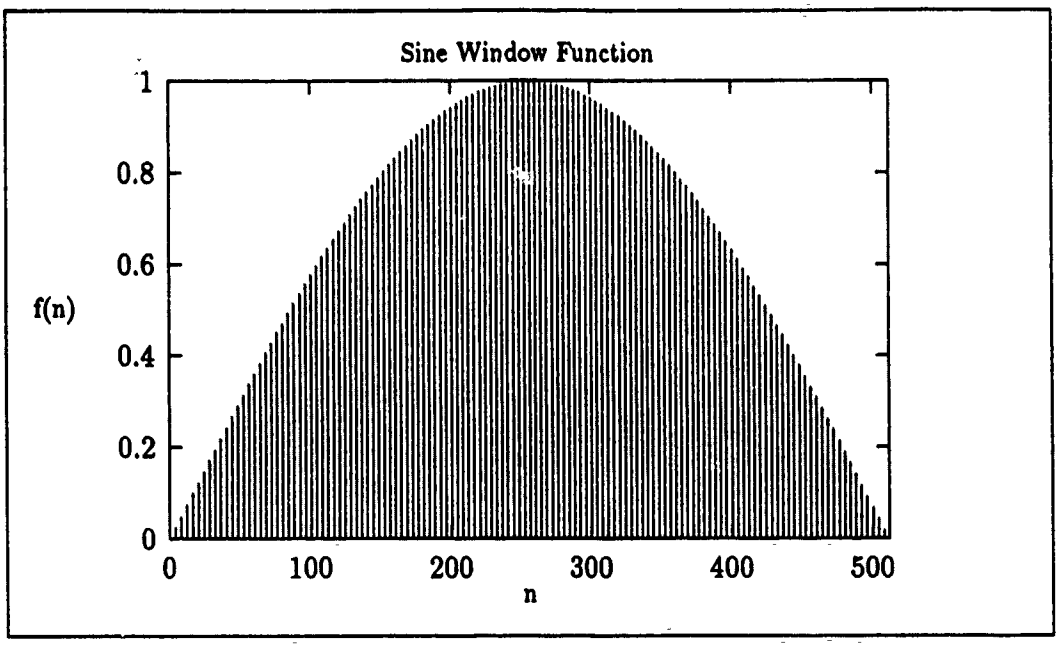



Figure 7. Sine Window Function with 512 Points

acted upon differently by the system. This causes amplitude discontinuities when the time slices are added back together. Without output windowing, the discontinuities at the time slice additions caused a 60 Hz "purring noise" in the reconstructed speech. The 60 Hz noise corresponds to a 16 ms occurrence rate; the rate at which the time slices were added together. The window function gradually adds in each new time slice, thereby smoothing the discontinuities.

The original window function chosen for the system was the hamming window.

$$W(n) = 0.54 - 0.46cos(2\pi n/N) \tag{4}$$

Where:

$n = 0, 1, 2, ...N - 1$

The Hamming window, shown in Figure 8, had the desired effect of reducing the spectral leakage when used at the input to the system. However, when this window function is used as the window function for the input and output, the repro-

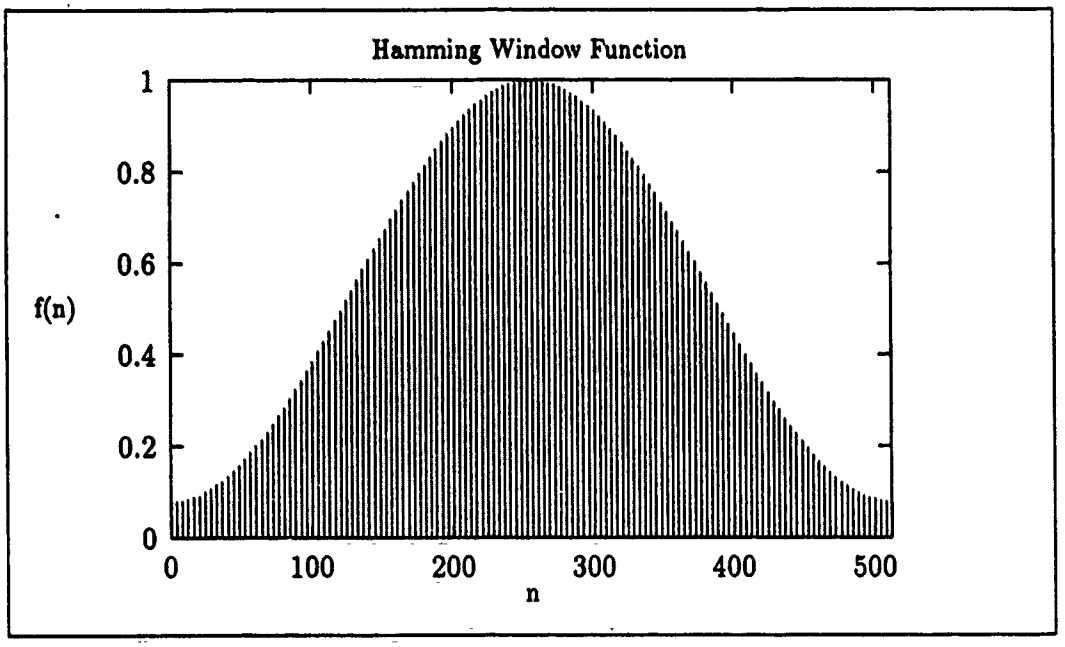



Figure 8. Hamming Window Function with 512 Points

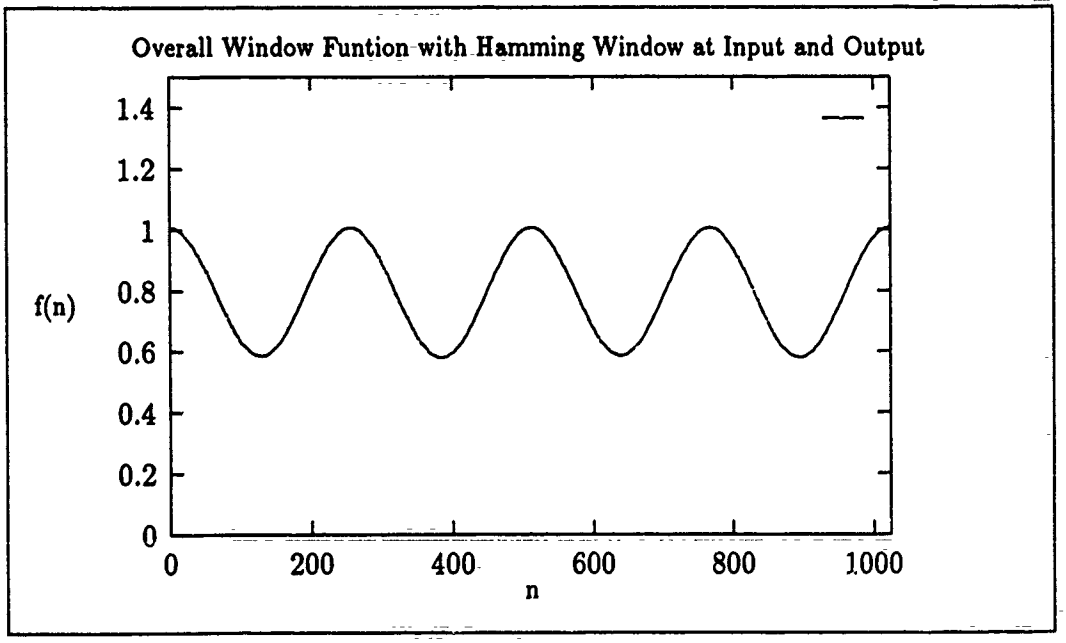



Figure 9. Overall Hamming Window Scheme Function

duced speech was modulated with a 60 Hz sinusoid. This effect is explained by the calculation shown in Figure 9. The calculation represents the overall effect of the windowing scheme. The window scheme consists of the window function squared, due to the multiplication at the input and output, and added to itself with a 50 percent overlap (50 percent overlap is explained in the next section). As shown, the resulting function is a 60 Hz sinusoid. Because of the distortion introduced by the Hamming window, a different function whose overall window function which did not add noise to the system, namely the sine window.

The overall effect on the system by the sine window is multiplication by unity. The sine window effect was verified by reconstructing speech with all frequency components. The speech was indistinguisable from the original recorded speech.

*4.1.3* **Frame Overlapping** A 50 percent frame overlap was used in the speech analysis. The overlap is needed because of the characteristics of the windowing function. Since the window function is near zero at the endpoints, data at the endpoints is ignored if some type of overlapping scheme is not used. The most common overlap schemes are 33 percent and 50 percent. Since the data rate of the system is inversely proportional to the overlap, to help minimize the data rate, the 50 percent overlap was used.

*4.1.4* **Fast Fourier Transform** After the time frames were generated, overlapped, and windowed, they were input into a Fast Fourier Transform (FFT) to obtain the spectogram of the time slice. During design 256, 512, and 1024-point FFT's were tried in the system. Sample rate, number of samples per frame, and FFT size interact throughout the system. According to Harris (4) FFT bin size is defined to be:

$$\Delta f = \alpha(f_s/N) \tag{5}$$

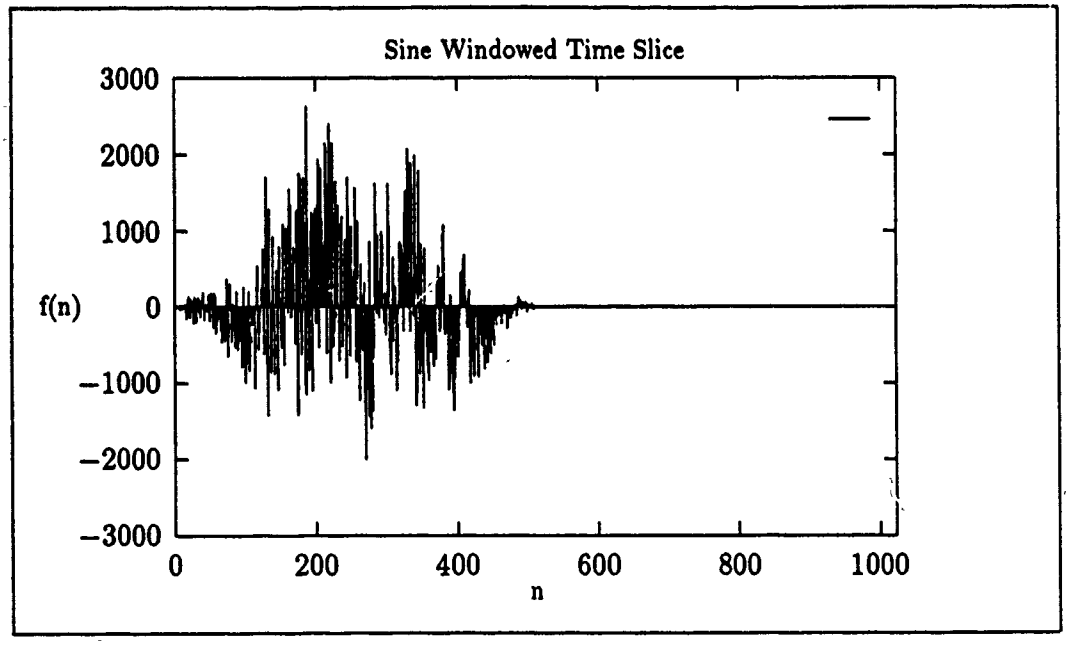



Figure 10. Windowed Timeslice with Zero Stuffing

Where:

- $\Delta f$ is the FFT bin width.
- $\alpha$ is coefficient reflecting bandwidth increase due to windowing.
- $f_s$ is the sample rate.
- $N$ is the FFT size.

The system goal was detection and estimation of important spectral components of speech. Therefore, a narrow bin width was desired. In order to shrink the bin size, a large FFT size was needed. However, the 32 to 40 ms timeslice duration times and the 16 thousand sample per second sample rate fixed the number of samples per time slice between 512 and 640 samples. Thus, either the FFT size must be 512 or zero padding had be used.

As discussed by Marple, zero padding causes the discrete-time Fourier series to interpolate transform values between the original N values. Thus, not only does zero stuffing match the transform size to the number of input values, it also gives

an "apparent" increase in transform resolution (8). In addition, subjective listening tests showed that the 1024 point FFT performed best at reconstructing the speech. Therefore the 1024 point FFT was chosen for the system. An example of a sine windowed and zero padded input to the FFT is shown in Figure 10.

*4.1.5* **Frequency Selecting** Bashir reconstructed speech by estimating the glottal frequency, then selecting all harmonics of the glottal frequency for use in reconstruction. He found that using exact harmonics of the glottal frequency caused a "musical noise" to corrupt the reconstructed speech. In order to eliminate this effect, he used a "nearest neighbor" rule to prevent exact harmonics of the glottal frequency from always being selected for reconstruction. The "nearest neighbor" rule consist of estimating the glottal as before, then searching the N nearest neighbors of the glottal harmonics to see which frequency component has the highest energy. The highest energy frequency component in the area is then selected for reconstruction of the speech. Bashir was able to reconstruct speech which was indistinguishable from the original recorded speech using this selection scheme. (2)

Alenquer initially selected the same components for speech reconstruction as Bashir by using the nearest neighbor rule. Alenquer's goal was to code speech with as low a data rate as possible. Therefore, his next step was to eliminate as many of the components selected as possible without severely distorting the speech. Alenquer chose to send the first N, N being the number of components sent per time slice, harmonics of the glottal frequency chosen using the nearest neighbor rule. Alenquer found that the system performed best using a four nearest neighbor rule. He was able to reconstruct speech using 16 frequency components and a 18 kbps data rate. (1)

As Alenquer did, this thesis began by pruning the frequency components which would later be used to select a smaller number of components for transmission. Many different frequency component selection schemes were tried in the system including:

- Choose greatest energy frequencies in the spectogram.
- Choose greatest energy frequencies from a low pass filtered (smoothed) spectogram.
- Approximate the glottal pitch; then choose glottal pitch harmonics using four nearest neighbor rule.
- Approximate the glottal pitch; then choose highest energy glottal pitch harmonics using the nearest neighbor rule and using compensation factors to account for 6 dB per octave rolloff in speech energy above 600Hz.

Limited research time necessitated the choice of one selection scheme for research in the system. Again, preliminary subjective listening tests were used to select which scheme performed the best. The selection scheme using the compensation factor performed best and was selected for the system.

The 6dB per octave decrease in energy in speech was compensated for by generating a decision vector. The decision vector was generated by initially copying the amplitude spectra vector into the decision vector. The vector was then multiplied by a compensation vector, shown in Figure 11, which consisted of components increasing exponentially at 6 dB per octave. The functions used to generate the compensation vector are:

$$\begin{aligned} c(n) &= 1 && n = 0, 1..37 \\ c(n) &= e^{.0355(n-38)} && n = 38, 39, ..76 \\ c(n) &= 4e^{.018(n-77)} && n = 77, 78, ..154 \\ c(n) &= 16e^{.009(n-154)} && n = 155, 156, ..256 \end{aligned} \tag{6}$$

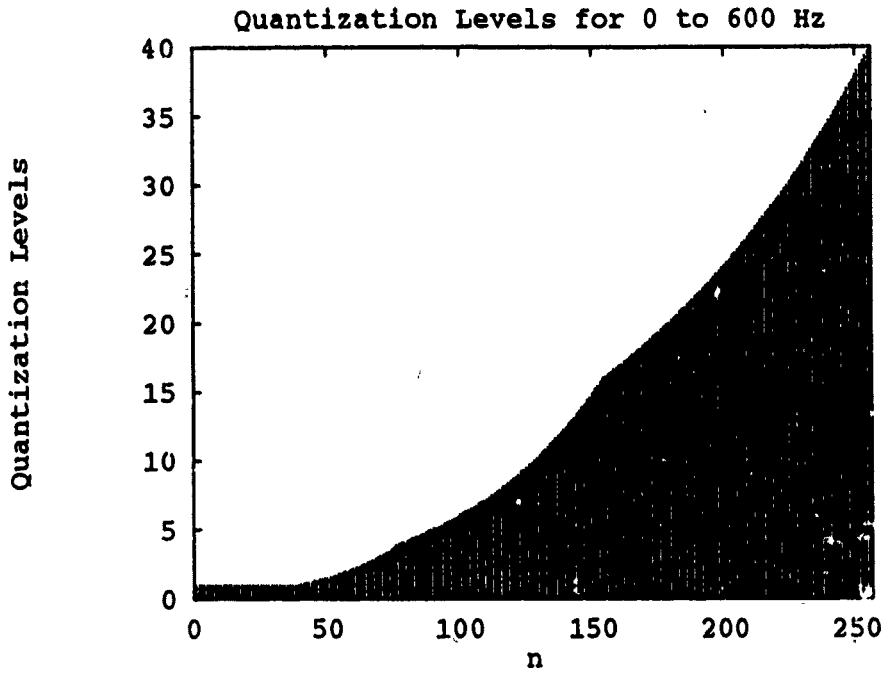



Figure 11. Compensation Vector to Compensate for Energy Drop-off in Speech

An additional problem was found while experimenting with selection schemes. That is, strong formants were usually several selected harmonics wide. Therefore many adjacent components were selected representing the same formant, causing the spectra to be inadequately sampled. This problem was greatly reduced by the use of a neighborhood inhibiting scheme. The neighborhood inhibiting scheme consists of selecting components from the reduced frequency set using the decision vector then multiplying the four nearest neighbors of the selected frequency by .75 to inhibit them from being selected. Examples of spectra and frequencies selected for voiced and fricative speech are shown in Figures 12, 13, 14 and 15.

## *4.2* Coding System

The second portion of the speech processing system is the coding module. The components which are to be coded are:

1. The frequency of the selected components.

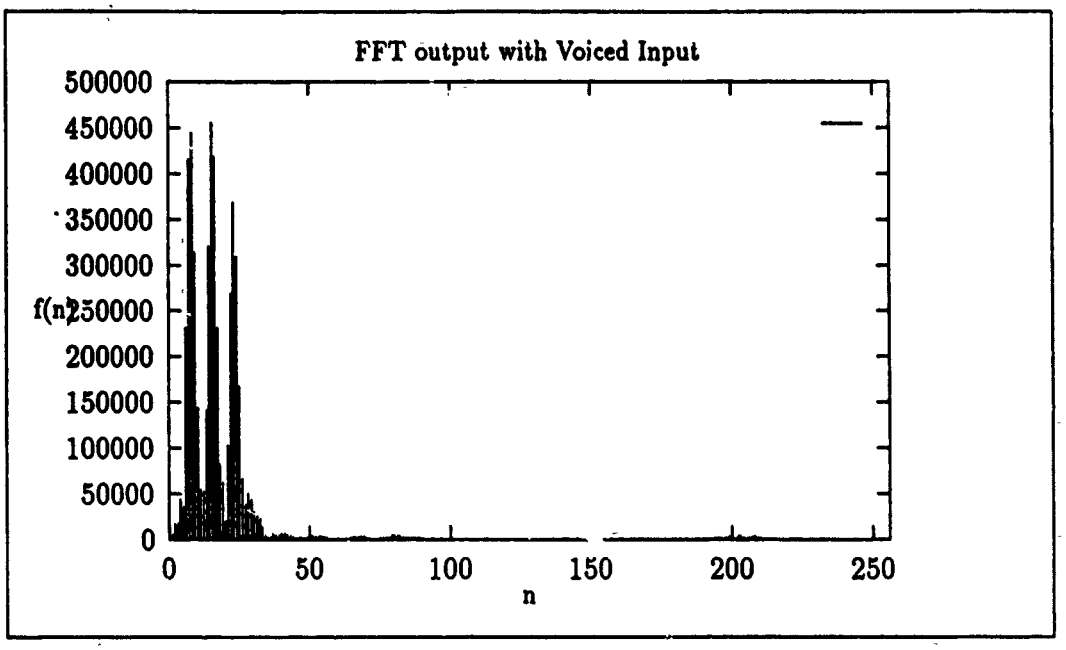



Figure 12. FFT Output with Voiced Input

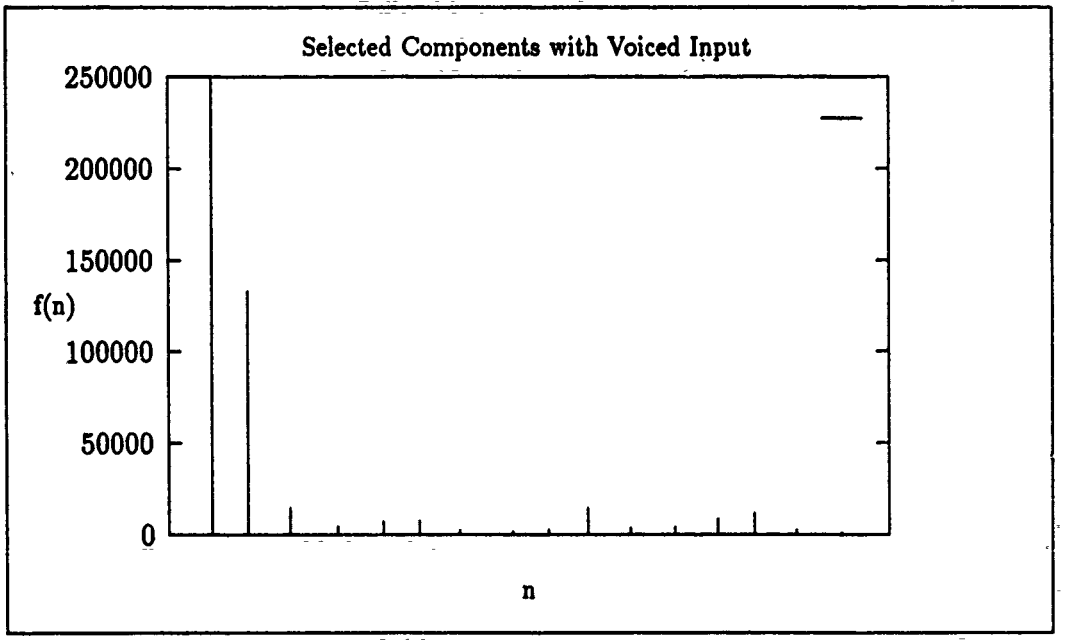



Figure 13. Voiced Frequency Selection

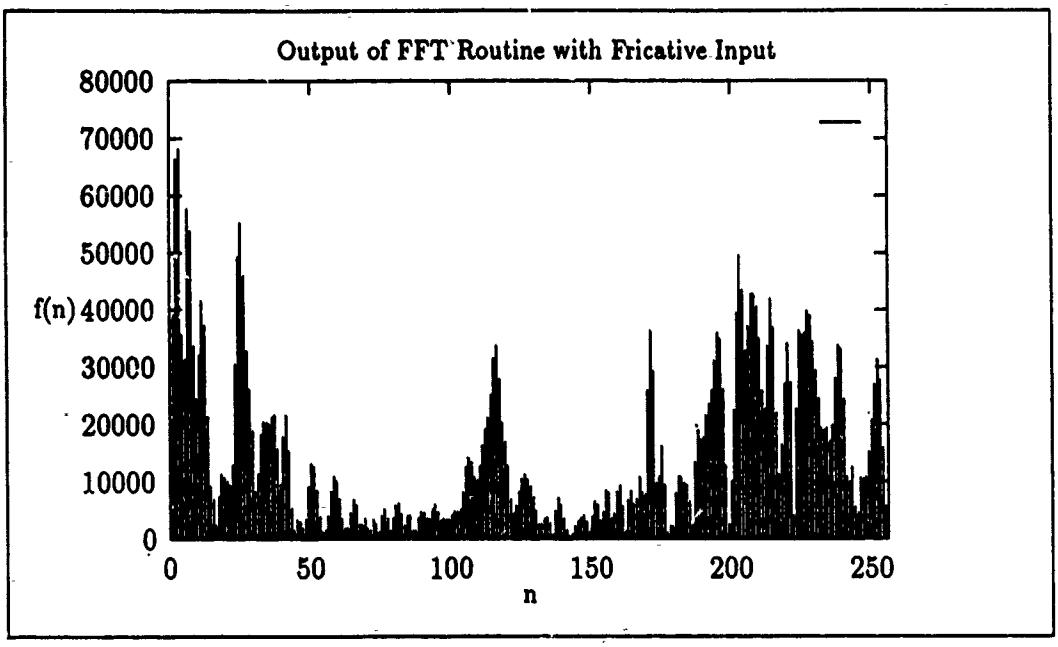



Figure 14. FFT Output with Fricative Input

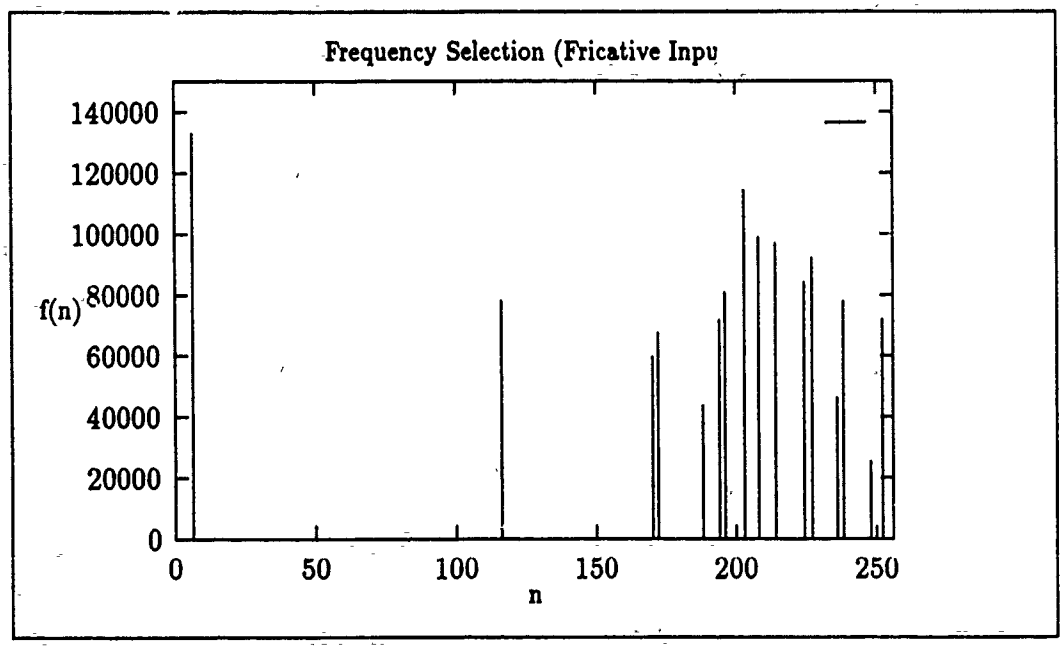



Figure 15. Fricative Frequency Selection

2. The amplitude of the transmitted frequency components.

3. The phase of the transmitted frequency components.

*4.2.1* **Amplitude Quantization** The amplitude of the transmitted frequency components were linearly quantized as integers between zero and an empirically determined maximum value. For the system described in this thesis, the maximum value was found to be 800,000. This value is a function of the multiplicative constants of the FFT and other subroutines.

Again, the fact that speech energy drops off at a rate of 6dB per octave was used to match the speech processing system to speech characteristics. This was done by adaptively adjusting the quantization step size of each frequency by the inverse value of the compensation value used in the frequency selection section of the program. Thus, decreasing the step size and the maximum quantization error for the higher frequency values. An example of quantization step size rolloff is shown in Figure 16. As shown, if the quantization step size were 400 for 0 to 600 Hz, the step size would decrease to 100 at the end of the first octave (1200 Hz) and would further decrease to 25 by the second octave (2400 Hz).

*4.2.2* **Phase Quantization** The phase value of the transmitted frequency components were linearly quantized between 0 and $2\pi$. For the final system, all phase components are quantized and sent. However several different phase transmission combinations were tried including:

1. Sending the phase of the first component and setting all other phase values to zero.

2. Sending the phase of the four lowest frequency components and setting all other phase values to zero.

3. Sending the phase values of all components other than the four lowest frequencies and setting all other phase values to zero.

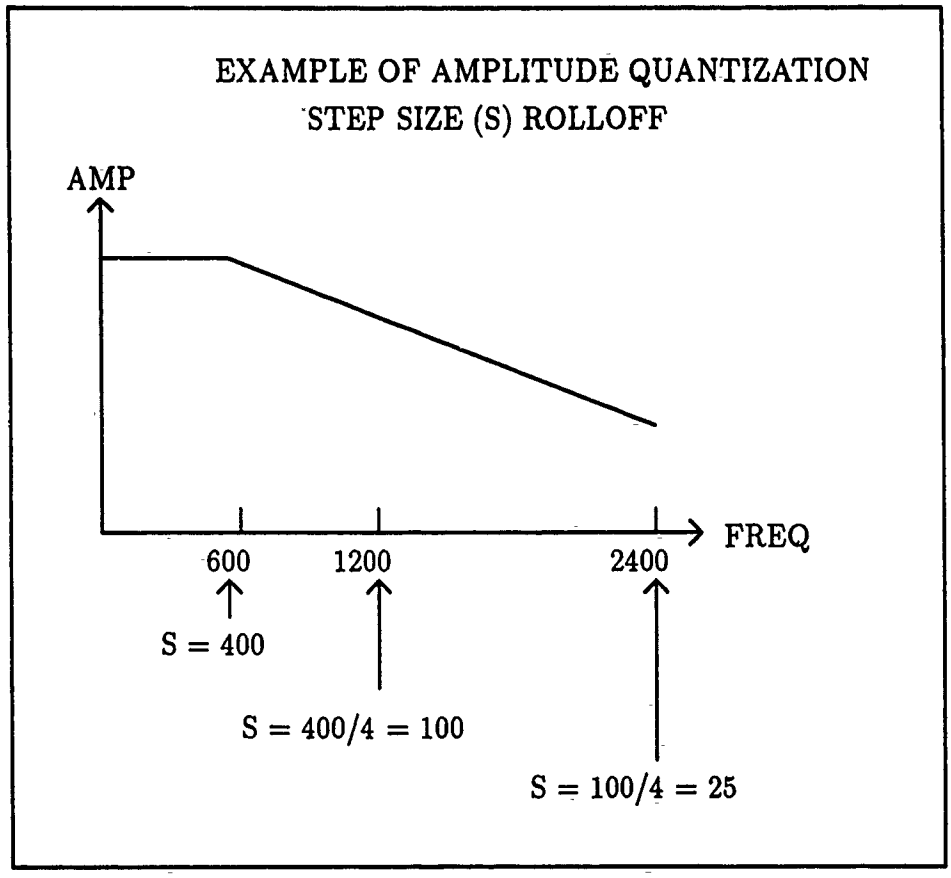



Figure 16. Example of Adaptive Amplitude Quantization Scheme

All of the above schemes caused the reproduced speech to sound as if the speaker was hoarse from a bad cold.

*4.2.3* **Frequency Coding** In order to transmit the frequency location of the selected component both the harmonic and the jitter components had to be transmitted. As described in the frequency selection section, the speech analysis system searches for the highest energy frequency in the neighborhood of glottal frequency harmonics. The frequency location of the selected component is the number of the selected harmonic and the amount of difference from the exact harmonic.

$$bin = (h * g) + j \tag{7}$$

where:

- h = The number of harmonics of the glottal bin between the transmitted harmonic and the previous transmitted harmonic.
- g = The glottal frequency bin for the timeslice.
- j = The jitter or difference between the exact harmonic and the frequency selected.

For example, if the glottal frequency bin is found to be the 8th bin, a possible frequency would be the 10th harmonic with a jitter of $-1$. In this example the frequency bin would be interpreted to be $(8 * 10) - 1 = 79$th frequency bin.

In order to transmit the frequency in this manner the output signal had to be limited to a maximum number of bins. In order to transmit the harmonic number with 5 bits the harmonic number had to be limited to:

$$(c - 1) * 7 + 32 * 7 \tag{8}$$

Equation 8 represents the worst case frequency transmission situation. In the situation all but one frequencies are as close together as possible and the last frequency is as far away from the others as possible. Each 7 in the equation represents the lowest bin in which the system searches for the glottal frequency. The first 7 represents the glottal frequency being found in the 7th FFT bin. The 32*7 represents the maximum distance frequency bin which can be selected with 5 bits and the the glottal bin = 7. $c$ represents the number of components sent. The $(c-1)$ calculates the smallest number of bins covered by the first $c-1$ frequencies. If the system is not limited to this number of bins, it is possible for one of the frequencies selected to be more than 32 multiples of the glottal frequency away from the previous selected frequency, thus causing an error at the receiver. The system typically used 8 components and was therefore limited to 273 frequency bins. Each bin is $15.625Hz$ wide therefore limiting to 273 bins corresponds to 4265 Hz bandlimiting. The final system design limited the output speech energy to frequencies below 4000 Hz. The maximum number of bins was therefore limited to 256, well below the 273 maximum derived above.

### *4.3* Energy Conservation

Speech signals which were regenerated from a small number of spectral components had a fluctuation in volume. The reason for the fluctuation was determined to be that the amount of energy in the selected frequency components varied greatly from frame to frame. The volume fluctuation was particularly evident at voiced-unvoiced boundaries. The spectra of the voiced frames had small areas where the energy of the frame was concentrated. However, the unvoiced frame spectra had a relatively uniform distribution. When small numbers of frequency energies were selected for transmission, a significantly smaller amount of energy was selected from the unvoiced frames. The result was weak sounding fricatives and as mentioned before a fluctuation in the volume from frame to frame.

To compensate for the volume fluctuations, the amount of energy in the chosen

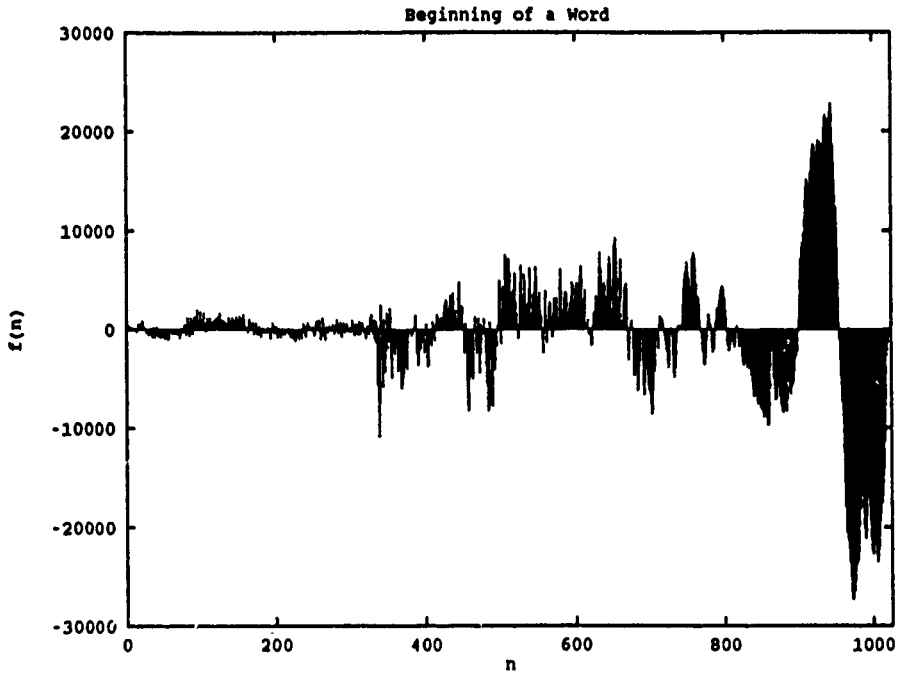



Figure 17. Beginning of Word Reproduced with all Frequency Components

spectral components was amplified so that the amount of energy in the reconstructed speech was equal to the energy in the original timeframe. The amount of amplification varied from frame to frame and was calculated to be:

$$d = \sqrt{\frac{\sum_{n=0}^{255} c_N^2}{\sum_{k=0}^{N-1} c_k^2}} \tag{9}$$

Where:

$N$ is the number of spectral components selected for the timeslice.

$\sum_{n=0}^{255} c_n^2$ is the energy in the original timeslice.

$\sum_{k=0}^{N-1} c_k^2$ is the energy in the selected spectral components.

The result of this operation was higher volume fricatives.

## 4.4 Voiced-Unvoiced Thresholding

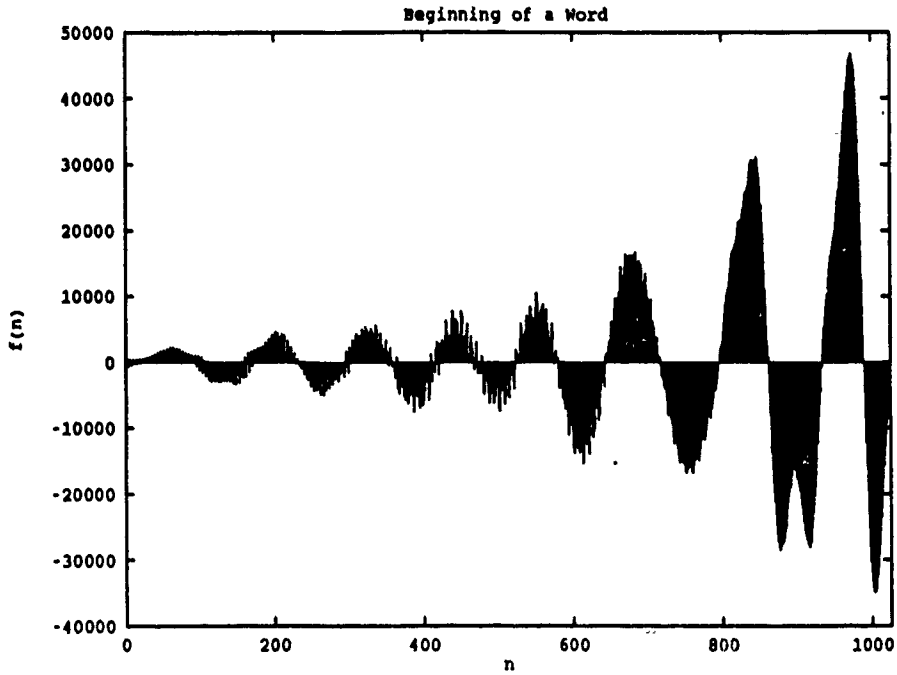



Figure 18. Beginning of Word Reproduced with Three Frequency Components

During preliminary subjective listening tests, listeners complained about a ringing noise during unvoiced portions of the reproduced speech. Inspection of the waveform in unvoiced regions revealed that fricative signals were riding on top of lower frequency components which were not evident in the original time waveform (see Figures 17 and 18). The lower frequency on which the fricatives were riding was approximately the same as the glottal frequency. The speech processing system always transmits a frequency which was believed to be the glottal frequency and this was the corrupting lower frequency in the fricatives. To reduce the ringing in the fricatives, lower frequencies had to be eliminated from the unvoiced portion of the signal. To eliminate the ringing a decision had to be made as to whether each timeslice was voiced, unvoiced, or blank space. The decision was based on the amplitude of the amplitude of the glottal frequency estimate which was made in all three classes of time waveform. The decision thresholds were empirically determined to be:

1. If the amplitude of the glottal frequency estimate is above 100000 the timeslice is determined to be voiced speech.
2. If the amplitude of the glottal frequency estimate is between 35000 and 100000 the region is determined to be unvoiced speech.
3. If the amplitude of the glottal frequency estimate is below 35000 the region is blank space.

Thresholding also eliminated noise which had been generated in the blank space due to the energy conserving subroutine putting all of the frames energy into a few frequencies. Only voiced and unvoiced speech frames were sent into the energy conservation subroutine. Thresholding decreased noise, however, it did not completely eliminate the ringing noise. The decision failed if a timeslice was half voiced and half unvoiced.

### *4.5* Data Rate

The data rate of the system was found to be:

$$\frac{S_r * L * (A + P + F)}{R * N_s} \tag{10}$$

where:

$S_r$ is the sample rate of the A to D converter.

$L$ is the number of selected frequency components used for reconstructing.

$A$ is the number of bits used to quantize amplitude information.

$P$ is the number of bits used to quantize phase information.

$A$ is the number of bits used to quantize frequency information.

$R$ is the percent overlap used for the FFT (50%).

$N_s$ is the number of sample points taken to be one timeslice.

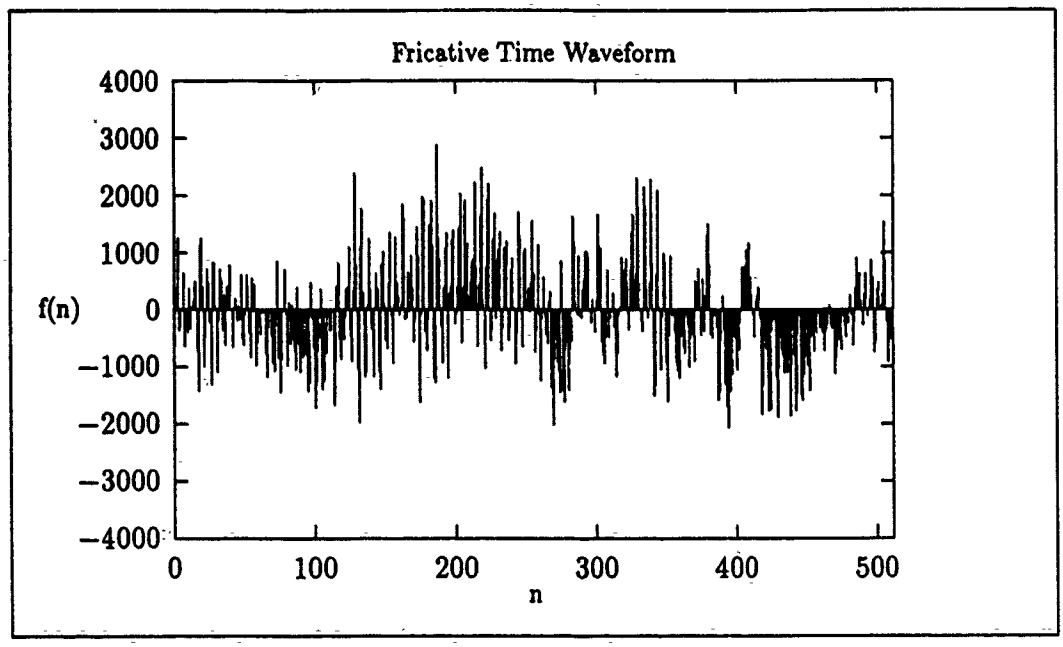



Figure 19. Original Fricative Waveform

## 4.6 Synthesis System

After the signal was analyzed and coded, the estimate of the speech was generated with the sinusoidal speech model described by Quatieri and McAulay (9). The estimate was generated using the reconstruction model

$$s(n) = \sum_{t=0}^{N-1} \sum_{f=0}^{maxf} A_f \cos(\frac{2\pi tf}{1024} + \phi) \tag{11}$$

where:

$s(n)$ is the reconstructed speech signal.

$maxf$ is the maximum allowed frequency bin for the system.

$A_f$ is the quantized amplitude of the selected frequency component.

$\phi$ is the quantized phase of the selected frequency component.

$t$ is the time index for the timeslice.

$f$ is the frequency index.

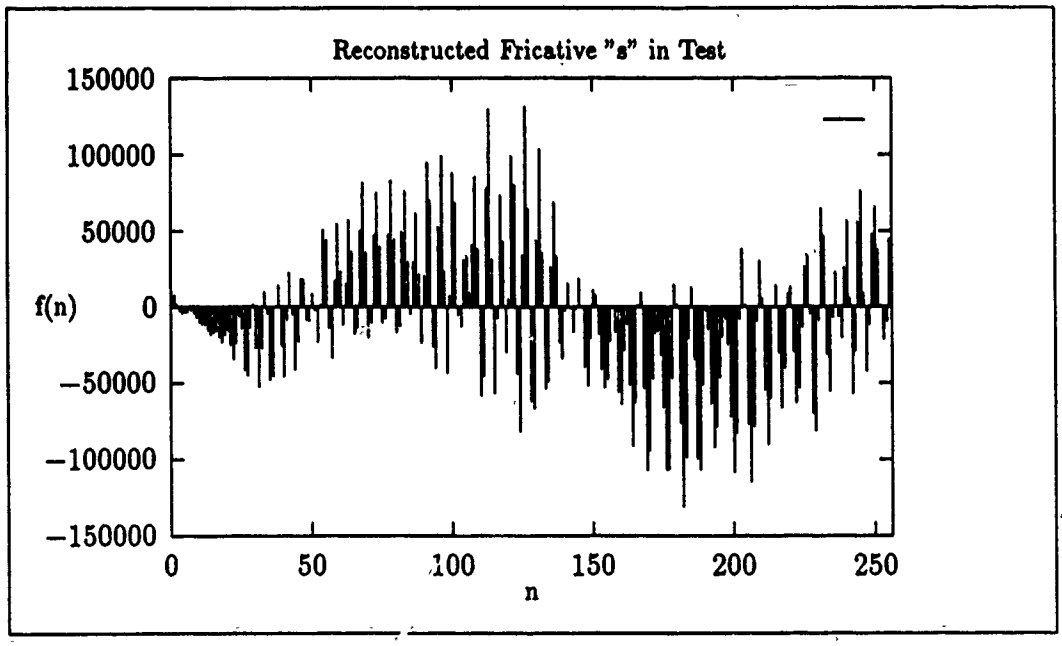



Figure 20. Reconstructed Fricative Waveform

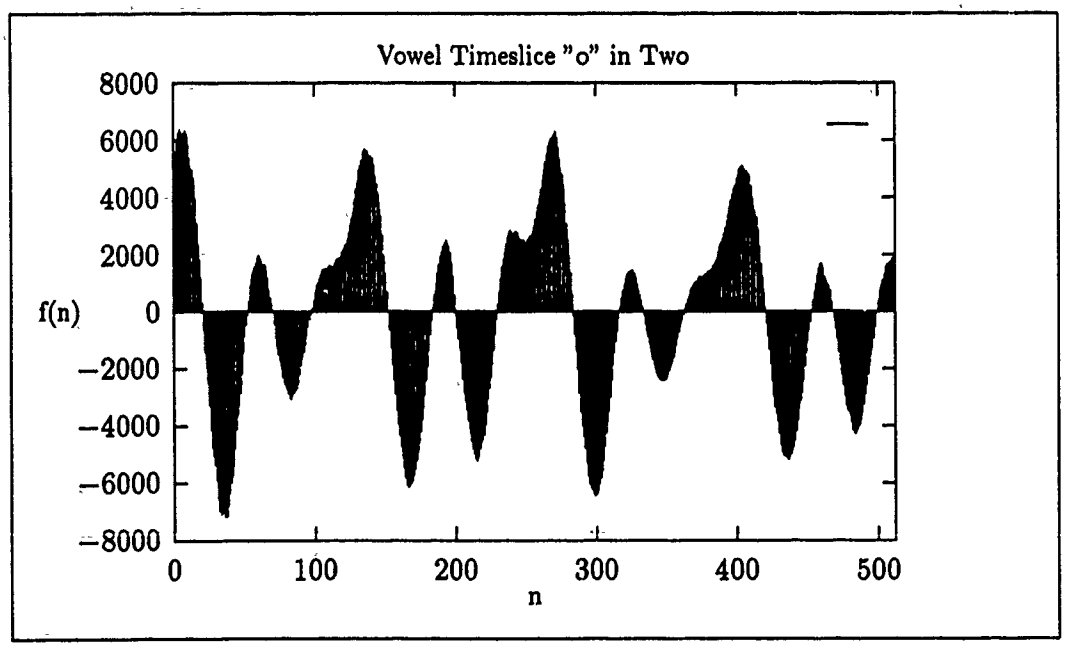



Figure 21. Original Vowel Waveform

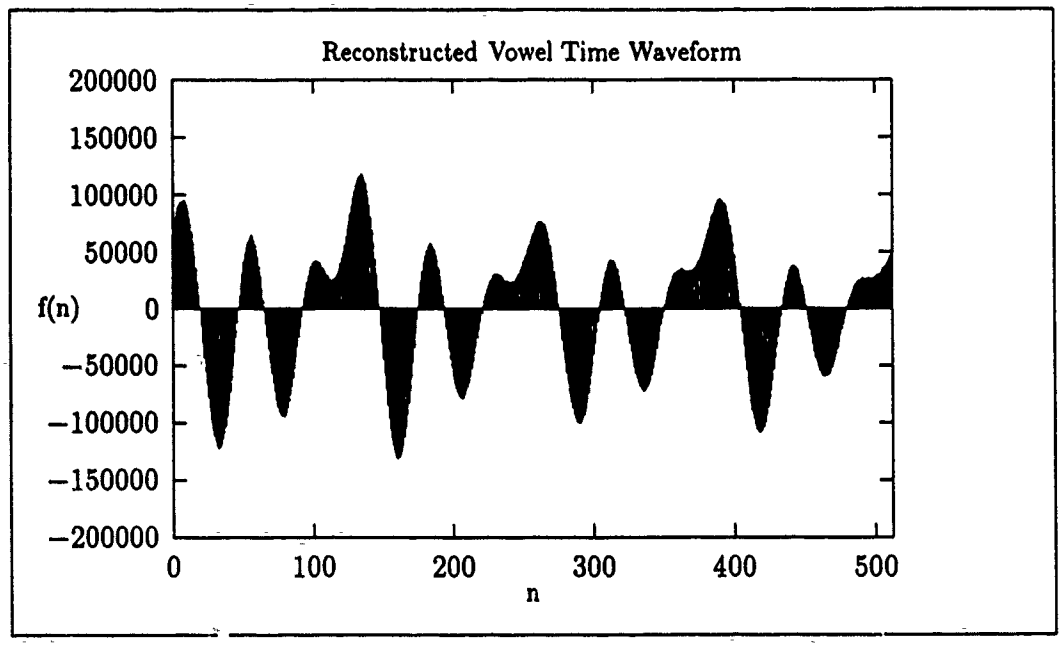



Figure 22. Reconstructed Vowel Waveform

$N$ is the number of samples per timeslice.

Examples of an original and reproduced voiced waveform are shown in Figures 21 and 22. The reconstructed vowel was generated using 16 frequency components. Note that the reconstruction is not exact, however the waveforms are very similar. An example of an original fricative waveform and its reconstruction are shown in Figures 19 and 20. Here the waveforms are not as similar as the reconstructed vowel is to its original waveform, yet the fricatives in the utterance are clear.

# *V.* System Testing

## *5.1* Introduction

There are several types of speech reproduction success measures. Some of these attempt to measure the success of reconstruction by calculating the absolute error in the signal reconstruction. These measures are useful because a very low error in signal reconstruction would have a very good chance of being accepted by the human listener. However, there could be cases of a large signal error which does not insult the human ear at all. There are two measures which describe the success of a coding system's ability to interface with human listeners. The first is "quality" of reproduction. This is a measure of how natural the reproduction sounds to listeners; including how much the reproduction sounds like the speaker. The second measure is "intelligibility" of the reproduction. Intelligibility is the ability of the system to reproduce speech such that the listener can distinguish what word was sent from other similar words. For example, can the listener tell the difference between "cat" and "sat" being sent.

There is no single test which measures both speech quality and intelligibility. The test which measures speech intelligibility is the Modified Rhyme Test (MRT). Due to the size of the files required to run the MRT on the speech processing system, it was not feasible to run the MRT on the system. A set of subjective listening tests, designed to measure the quality of the speech reconstruction verses the design parameters were run on the system. In these listening tests, listeners were asked to rate the quality of the reproduced speech with different design parameter settings. How the parameters were varied will be discussed later in this chapter. The parameter settings were selected because by the performance of the parameters in the listening tests.

## 5.2 Listening Tests

Listening tests were performed on the speech processing system to measure how each of the design parameters affected the human perceived quality of speech. The tests consisted of having seven people listen to a test tape which consisted of several different reproductions of the same utterance. Each of the reproductions were the result of the speech processing system operating on the same digitized speech input file.

*5.2.1* **Test Tape** As described in the data rate section of this thesis, the data rate of the system was found to be:

$$\frac{S_r * L * (A + P + F)}{R * N_s} \tag{12}$$

where:

$S_r$ is the sample rate of the A to D converter.

$L$ is the number of selected frequency components used for reconstructing.

$A$ is the number of bits used to quantize amplitude information.

$P$ is the number of bits used to quantize phase information.

$F$ is the number of bits used to quantize frequency information.

$R$ is the percent overlap used for the FFT (50

$N_s$ is the number of sample points taken to be one timeslice.

The parameters to be adjusted in the listening tests were L, A, P, and F.

The test tape was partitioned into three sections. The first section tested the variation of the parameter L, the number of frequency components selected to reconstruct the timeslice. In this section of the tape the amplitude and phase (A and P) were transmitted unquantized. Therefore all of the error in the reconstructed

speech signal was due to the lack of certain frequency components. The section began with a reconstruction of the utterance "testing one two three" using only two frequency components. The section continued with reproductions using 3, 4, 5, 6, 7, 8, 9, 10, 15 and 30 frequency components.

The second portion of the tape was the section which tested the response to the variation of the parameter P, the number of bits used to quantize the phase information. In this section the amplitude was unquantized and the number of frequency components was set to 30. The phase variation section of the tape began with a reconstruction of the speech signal using one phase bit, or zero crossing phase. The section continued with reconstructions using 2, 3, 4, 5, 6, 7, and 8 bit phase.

The final section of the test tape was the variation of the parameter A, the number of bits used to quantize the amplitude information. In this section the phase information was unquantized and the number of frequency components selected was set to 30. Like the phase variation portion, the amplitude portion of the tape began with a speech reconstruction using only one bit, or on/off amplitude information. The section continued with reconstructions using 2, 3, 4, 5, 6, 7, and 8 bit amplitude quantization.

*5.2.2* **Test Procedure** The listening test was run on seven volunteers. Each volunteer selected had not been a preliminary listing test subject or had they repeatedly listened to the speech reproductions for any other reasons. After volunteering they were given the following instructions.

> Thank you for volunteering for the listening test portion of the Frequency Domain Speech Coding System. The test will take approximately 5 minutes each day for three days. The following is a list of the information which will be helpful in performing the test.
>
> 1. There are 27 utterances which have been reproduced by a data reduction coding system and recorded on a cassette tape. The tape

will be played for you. Each utterance is separated from the next utterance by about one second.

2. After each utterance reproduction, rate the quality of the reproduction on the rating sheet provided. Please note the scale on the rating sheet; that is, 0 means you could not understand the speech, 7 means the reproduction sounds as good as telephone quality.
3. Please remember that you are asked to return to perform the tests three times.

Again thank you for volunteering.

Vance McMillan

After reading the instructions they were given a rating sheet and asked if they understood the instructions and rating sheet. No other instructions or information was given. An example grading sheet is shown on page 45.

| Utt Num | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | |
| 2 | | | | | | | | | | | |
| 3 | | | | | | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | | | | | | | | | | | |
| 6 | | | | | | | | | | | |
| 7 | | | | | | | | | | | |
| 8 | | | | | | | | | | | |
| 9 | | | | | | | | | | | |
| 10 | | | | | | | | | | | |
| 11 | | | | | | | | | | | |
| 12 | | | | | | | | | | | |
| 13 | | | | | | | | | | | |
| 14 | | | | | | | | | | | |
| 15 | | | | | | | | | | | |
| 16 | | | | | | | | | | | |
| 17 | | | | | | | | | | | |
| 18 | | | | | | | | | | | |
| 19 | | | | | | | | | | | |
| 20 | | | | | | | | | | | |
| 21 | | | | | | | | | | | |
| 22 | | | | | | | | | | | |
| 23 | | | | | | | | | | | |
| 24 | | | | | | | | | | | |
| 25 | | | | | | | | | | | |
| 26 | | | | | | | | | | | |
| 27 | | | | | | | | | | | |

Scale

0 = Could not understand

7 = Telephone quality

10 = Sounds like origional utterance

# *VI.* Test Results

## *6.1* Introduction

In this chapter the overall results of the listening tests are presented. The complete set of listening test results are given in appendix A. The overall result curves represent the average test results from all seven listeners and all three test measurements.

## *6.2* Quality Verses Number of Frequencies used in Reconstruction

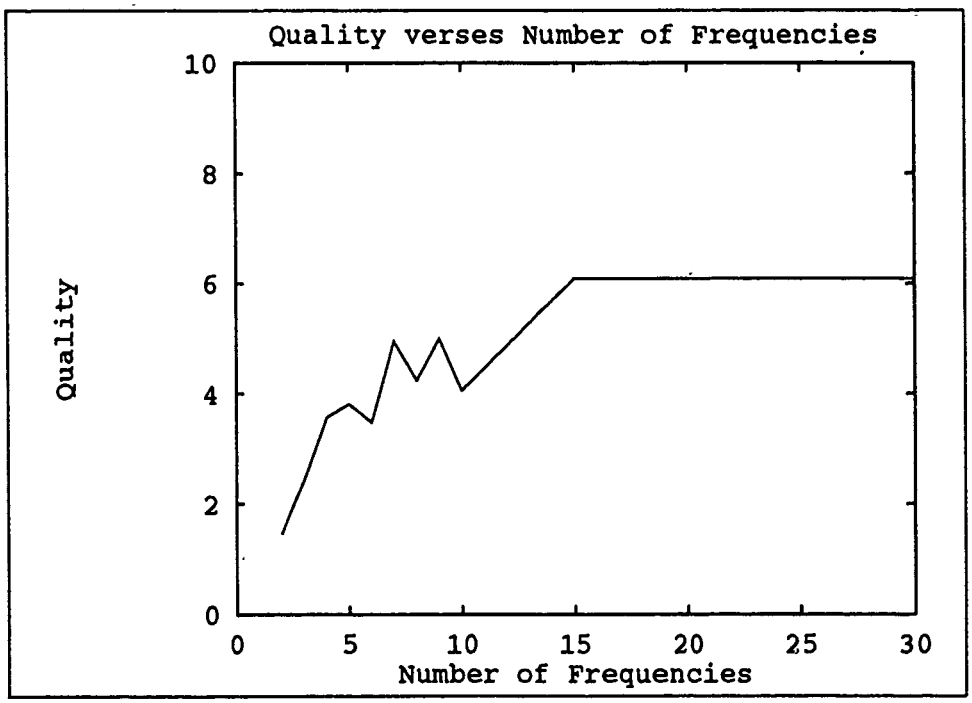



Figure 23. Quality vs Frequency Components Curve

Figure 23 shows the results of the portion of the listening tests aimed at measuring the quality of reconstruction as a function of the number of frequencies used in reconstruction. The overall shape of the curve is increasing as expected, however there are some erratic points on the curve which were not expected. These data points may be an artifact of the order of the reproductions on the test tape. During

the second and third iterations of the listening tests, the order of the reproductions was scrambled to prevent the listeners from memorizing the pattern of the response sheet, and thus biasing the results. With the order changed, the listeners tended to rate the first few utterances lower than the first time through the test. Some of the points on the curve which are lower than may be expected correspond to the first utterances on the scrambled version of the test tape.

The quality verses frequencies plot shows a sharp increase in quality until the number of frequencies reaches about eight. After this point only about one point in quality is gained by sending up to thirty frequencies. Based on these observations, the design parameter, number of frequencies used in reconstruction (L), was set to eight frequencies.

### *6.3* Quality Verses Number of Phase Bits

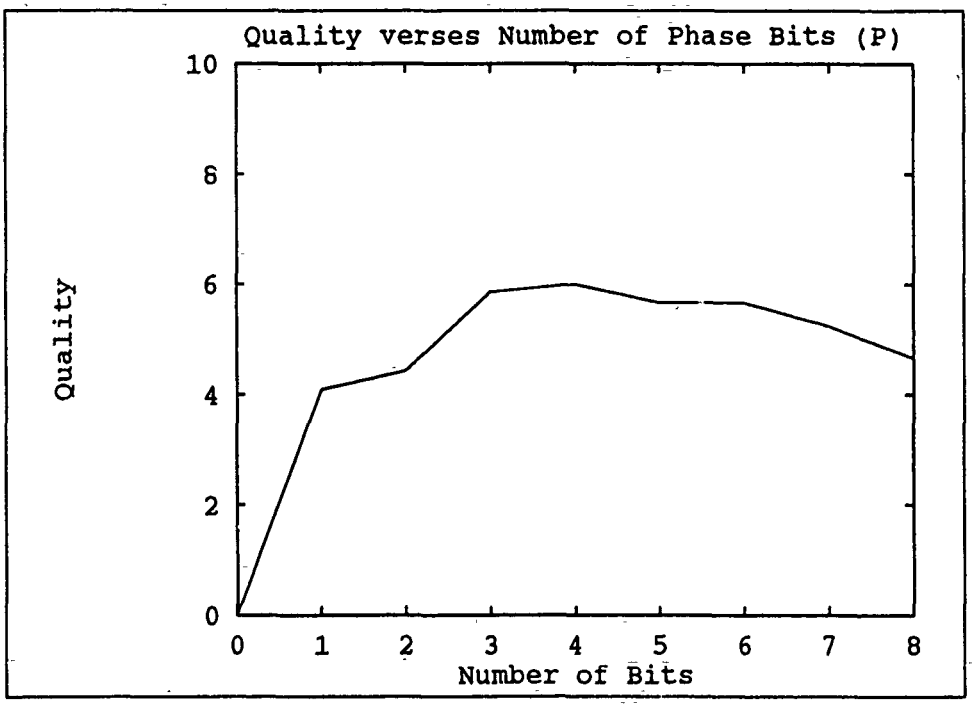



Figure 24. Quality vs Number of Phase Bits Curve

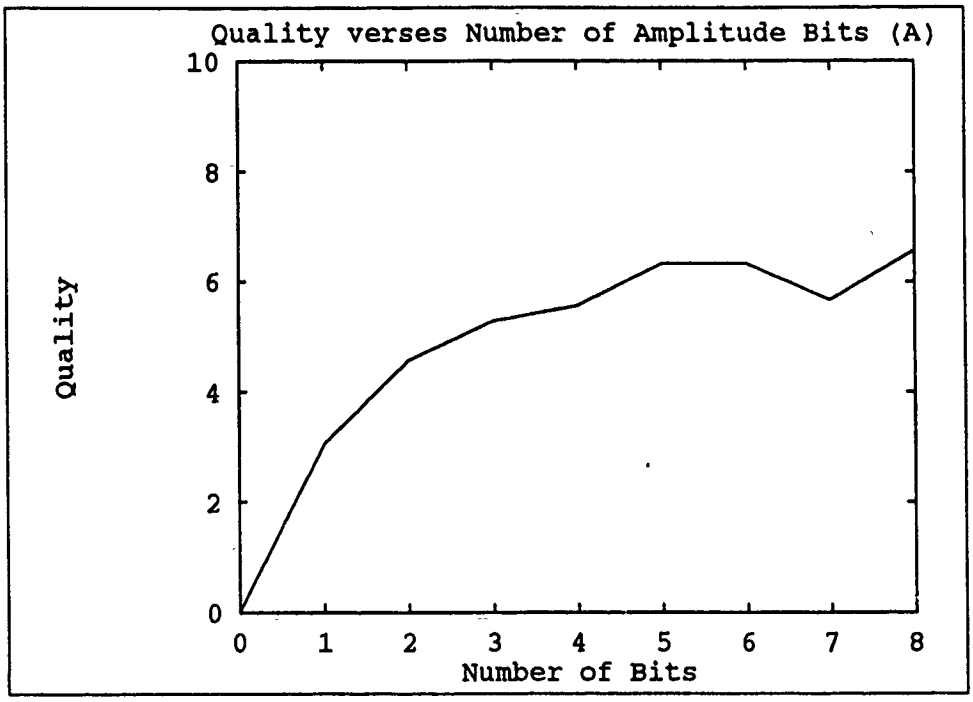



Figure 25. Quality vs Number of Amplitude Bits Curve

The quality verses phase bits curve shown in Figure 24 increases very quickly to its maximum value at about three bits, then remains approximately constant through eight bits. The curve shows that low quality phase information gives high quality reproduction. The design parameter (P) was set to three bits.

### *6.4* Quality Verses Number of Amplitude Bits

The final curve, shown in Figure 25, shows the relationship between quality and the number of bits representing amplitude information. Like the phase curve, the amplitude curve rises very quickly. The quality has nearly reached its maximum value by the two bit quantization point. This shows the amplitude quantization scheme performs well as rated by listeners. The design parameter (A), number of amplitude bits, was set to two bits based on the information shown in the quality vs amplitude bits curve.

## 6.5 System Design Results

The final system design parameters were L = 8, A = 2, and P = 3. In addition to these parameters the timeframe size was increased to 40 ms for the final design. This parameter was not one of the parameters included in the listening tests and therefore had to be selected by quality comparisons by people not involved in the formal listening tests. Increasing the size above 40 ms did have a slurring affect on the speech according to the listeners. However, little slurring was noted for timeframes below 40 ms.

These parameters resulted in a bit rate of 4800 bps in the final design of the speech processing system. The reproduced speech from the design sounded very good in the voiced regions and good in the unvoiced regions. The system did have the musical noise at the boundaries of voiced-unvoiced regions. There are several hypothesis as to why the musical noise is present at these boundaries.

1. The spectra of the speech is changing quickly at the voiced-unvoiced boundaries and therefore require a higher sample rate to prevent a time aliasing affect. This could also be explained in the time domain by realizing that these boundary signals are too complicated to be approximated from a few frequency components
2. The human brain is able to fill in the spectra in the slowly changing regions. However, the brain is unable to fill in the spectra for fast changing speech. The musical noise is the brain's unsuccessful attempt to fill the spectra and therefore the noise is not even present in the waveform(5).
3. The speech processing system makes good frequency selections to represent voiced or unvoiced speech but makes poor decisions when the timeframe is a combination of the two.

A system design validation test was run on the system with the final design parameters installed. The test consisted of having three of the listeners used in the

listening tests rate three utterances using the same scale used in the listening tests. The three utterances were: the original speech file, the system operating with the final design parameters, and the 30 frequency reconstruction with no quantization of the amplitude or phase information. The results were 8, 5.0, and 5.7 respectively. The results show that the original speech file does not rate a ten as might be expected. This may be the result of the recording and playback headphone distortion. The 30 frequency reconstruction was used as a comparator to see how the results correlated with the listening test results. This reproduction rated 6.2 in the listening test compared to the 5.7 in the validation test, well within the expected value of change by eliminating four of the listeners. The 5.0 rating for 4800 bps speech is encouraging. The result is only 2 rating points below toll quality speech at 4800 bps.

# *VII.* Conclusions and Recommendations

This thesis has been successful in refining the design of the sinusoidal based model speech coding scheme. The adaptive amplitude quantization design was validated by the listening test quality verses amplitude bit curve. The system performed well at reproducing speech at 4800 bps but the quality rating did not reach toll quality because of the noise in the voiced-unvoiced speech boundaries. Without the noise it may be possible to transmit 4800 bps toll quality speech.

The next step in the refining process of frequency domain speech coding should be to isolate the cause of the musical noise. If the noise is caused by the brain trying to fill in the spectra in the quickly changing speech then it may be eliminated by filling in the spectra at the receiver with an spectrum filling function, a function which fills in the spectra based on the limited components available. Efforts in this thesis included trying to fill the spectra with a sinc filling function which failed. Gaussian filling functions were also tried however finding a standard deviation which does not introduce noise into the speech was not accomplished.

If the noise is due to decisions by the system at voiced-unvoiced boundaries then a new set of rules may need to be developed. These rules must make good frequency selection choices at the boundaries. Additionally, a method of determining when the boundaries occur would need to be developed.

Finally, the most likely reason for the noise is that the system is trying to reconstruct a timeslice which is generally made of high frequency components during one half of the (unvoiced) and lower frequency components in the other half(voiced). The reconstruction is attempted with a small number of frequency components. The fact that the signal is changing quickly and the signal is sparsely sampled creates a " time aliasing" effect. One method of eliminating or reducing the effect would be to find a way to sense that the timeslice is a voiced-unvoiced timeslice and implement a

new set of frequency selection rules to cancel the aliasing effect. The new rules would need to sample the spectra more often in the quickly changing speech timeframes. A sampling scheme which knows how fast the waveform is changing will be extremely difficult to develop because the system must be non-causal. The delay due to the non-causality must remain small enough to prevent the system users from realizing that there is a delay or the " real time" appearance of the system will be lost. Another approach to solving the voiced-unvoiced boundary problem could be to have a method of adaptively adjusting the timeslice size based on the input speech. This could be accomplished by having a wavelet system looking ahead of the Fourier system. The wavelet system would recognize the different phonemes and would make speech timeslice boundaries accordingly. this type system is particularly attractive because it would not only solve the voiced-unvoiced problem it would possibly extend the average timeslice times. Thus decreasing the data rate considerably.

The Fourier based speech coding system has performed well in reconstructing low bit rate speech signals. It has been shown that it is capable of reproducing low bit rate (4800 bps) speech at near toll quality. With further research, the Fourier system will probably perform as well as Linear Predictive Coding (LPC) speech coding systems and possibly will perform better than these systems.

# Appendix A. **Listening Test Result Charts**

The data collected in the listening tests described in Chapter 5 are presented here. Each plot represents the average data collected from each volunteer. The listeners initials are shown in parentheses to the right of the plot title. All plots of quality verses frequency are presented. The plots for quality verse phase bits follow. Finally, the quality verses amplitude bits curves are presented.

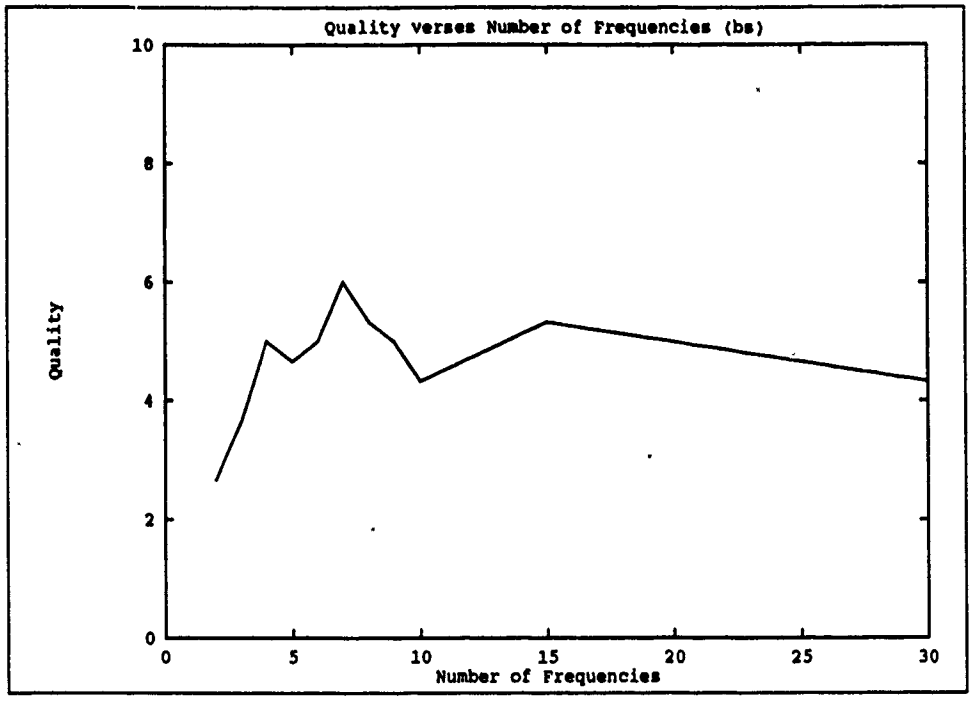



Figure 26. Average Quality Score verses Number of Frequency Components (L)

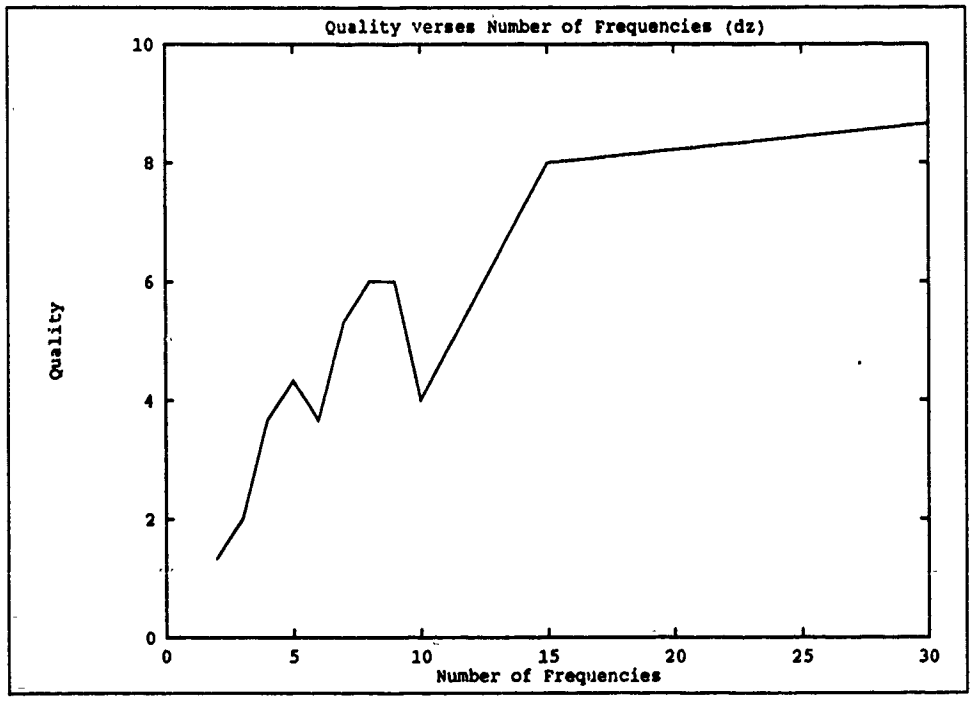



Figure 27. Average Quality Score verses Number of Frequency Components (L)

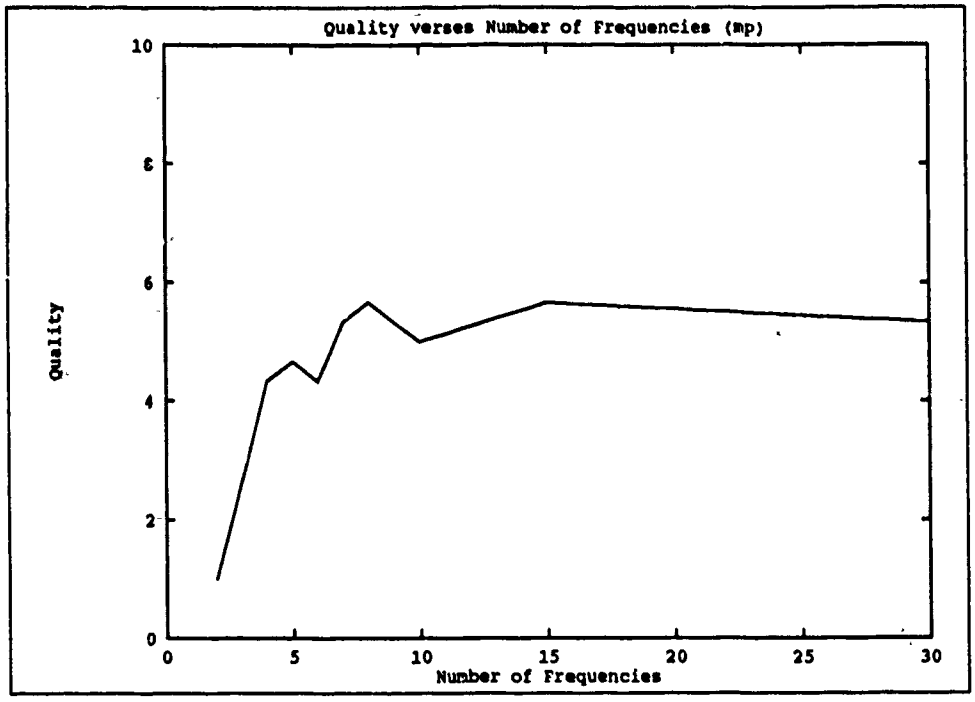



Figure 28. Average Quality Score verses Number of Frequency Components (L)

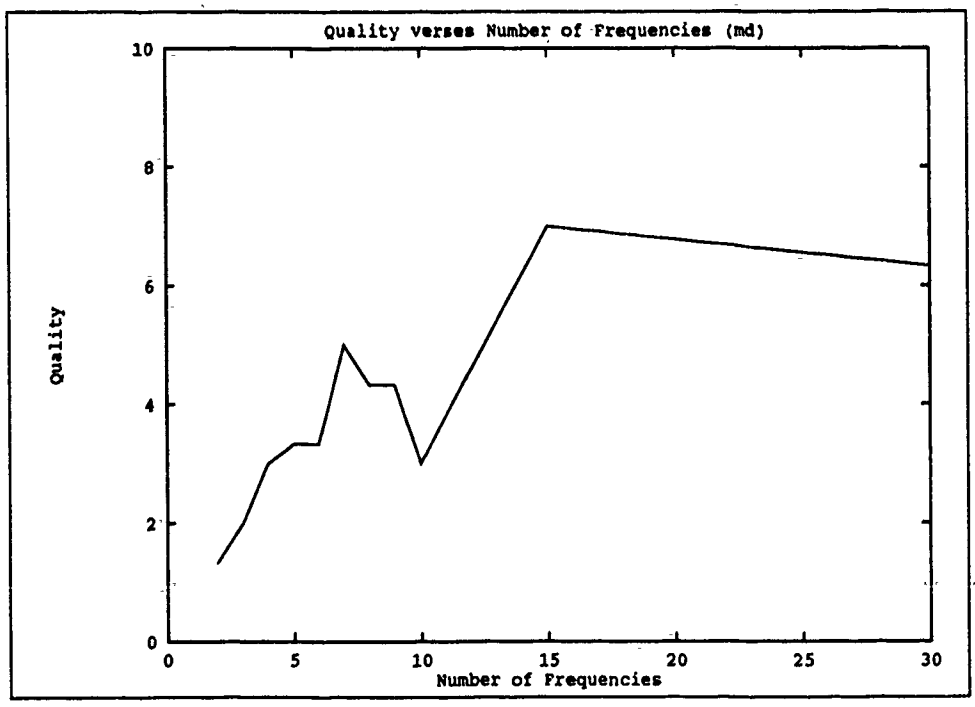



Figure 29. Average Quality Score verses Number of Frequency Components (L)

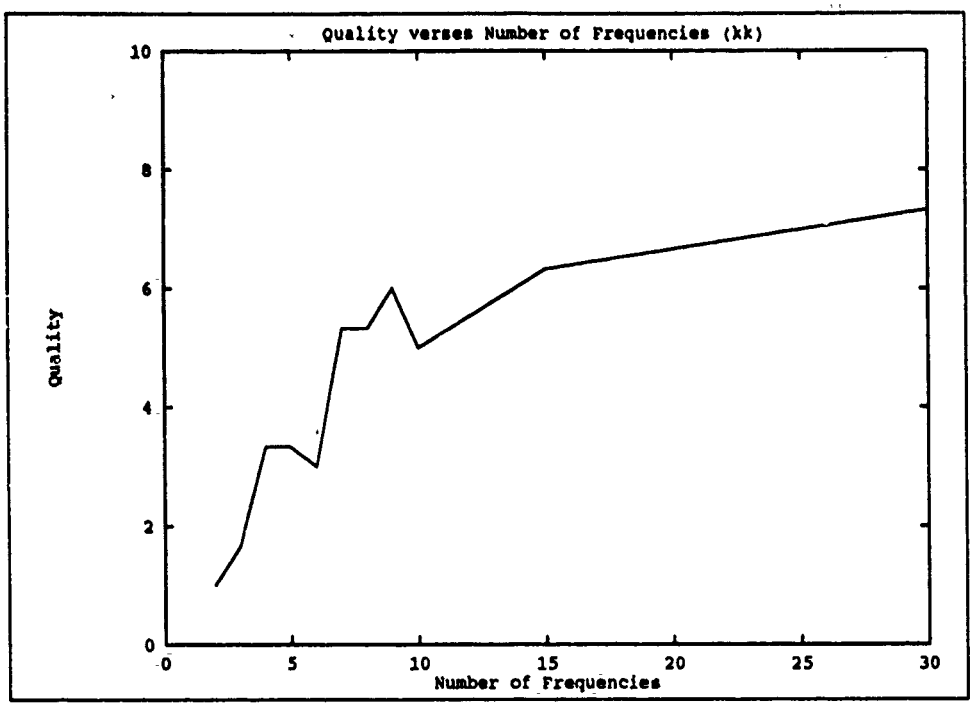



Figure 30. Average Quality Score verses Number of Frequency Components (L)

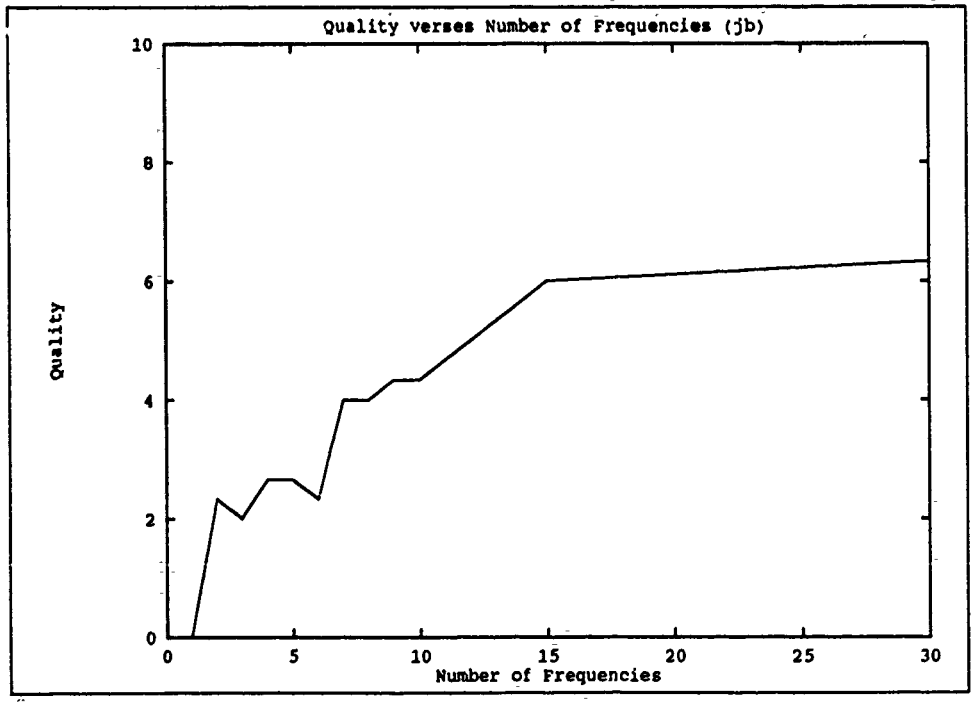



Figure 31. Average Quality Score verses Number of Frequency Components (L)

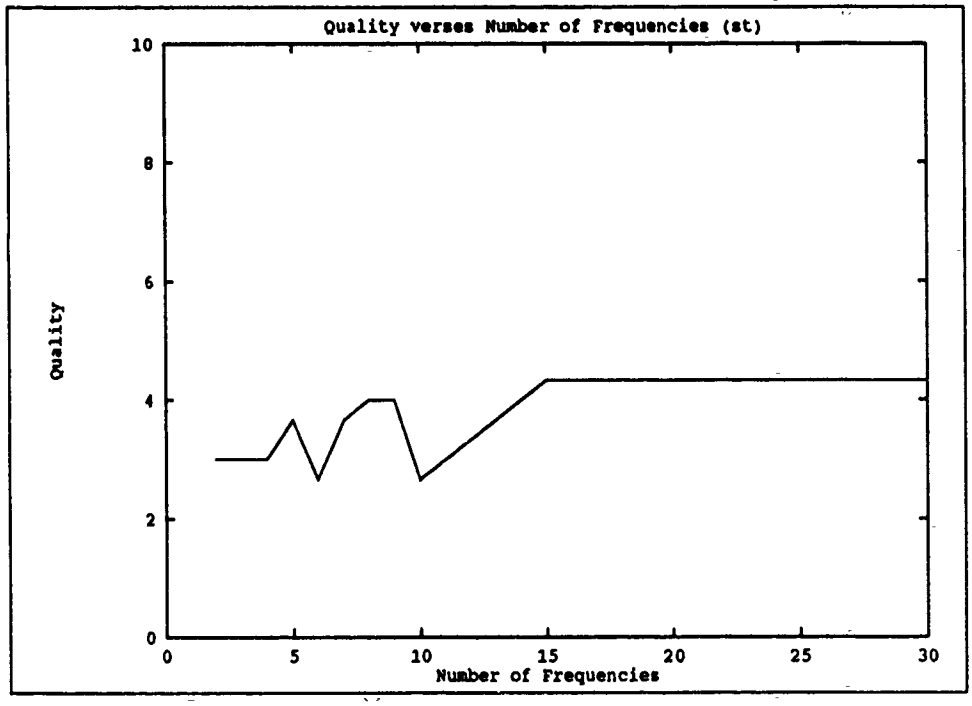



Figure 32. Average Quality Score verses Number of Frequency Components (L)

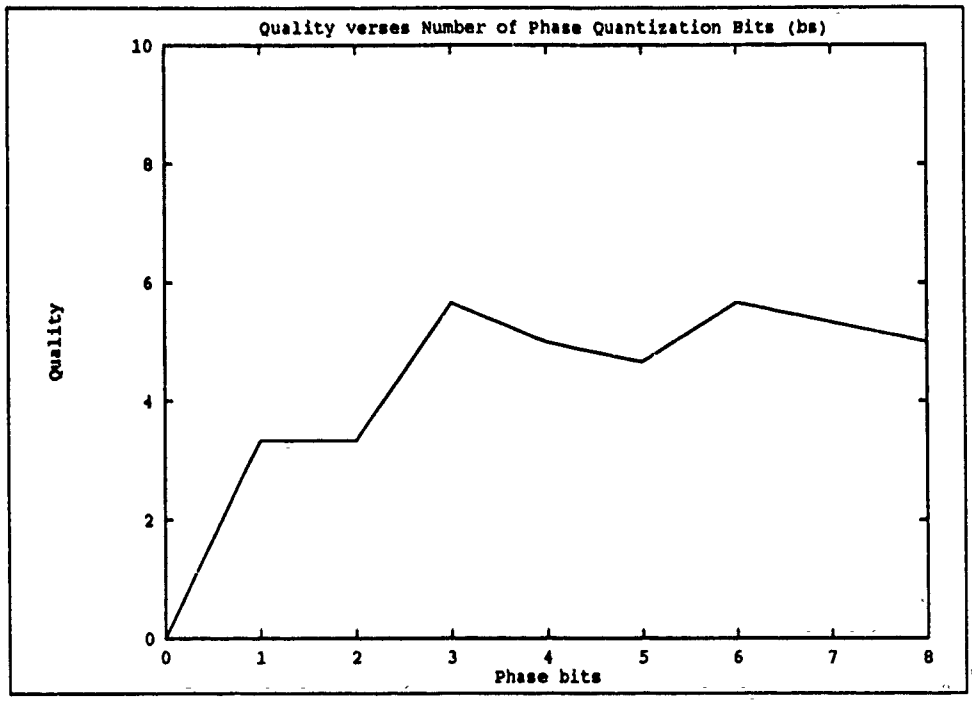



Figure 33. Average Quality Score verses Number of Phase Bits (P)

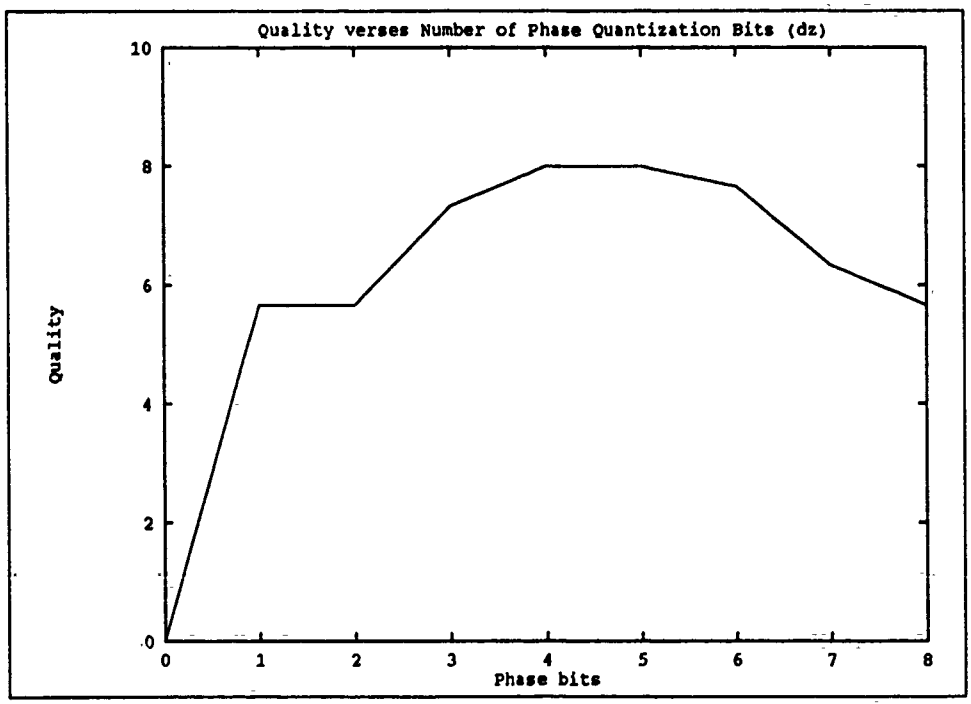



Figure 34. Average Quality Score verses Number of Phase Bits (P)

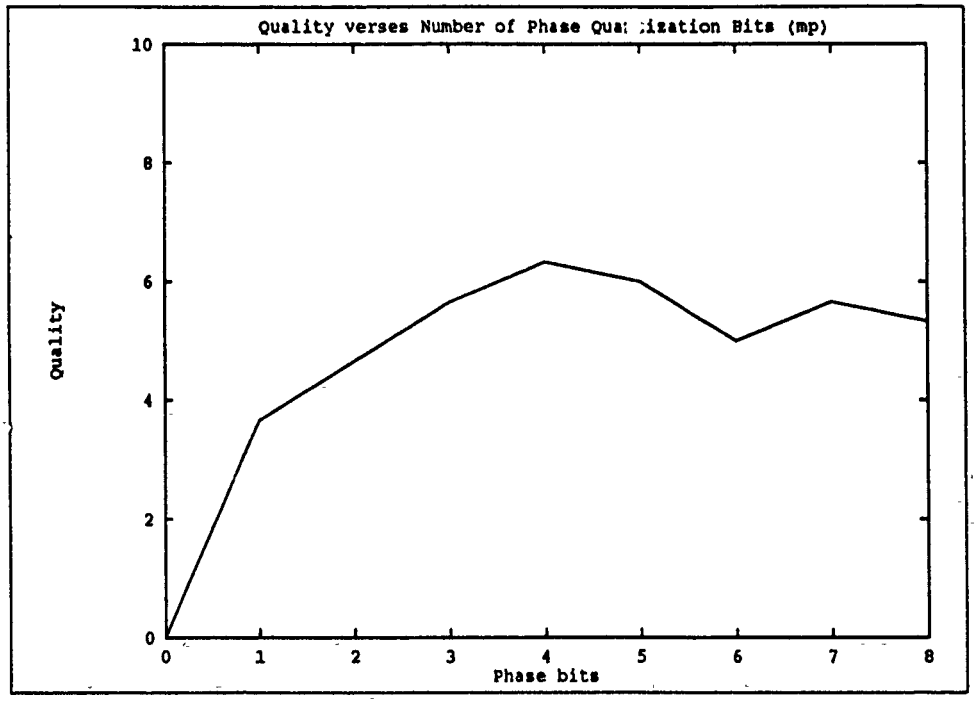



Figure 35. Average Quality Score verses Number of Phase Bits (P)

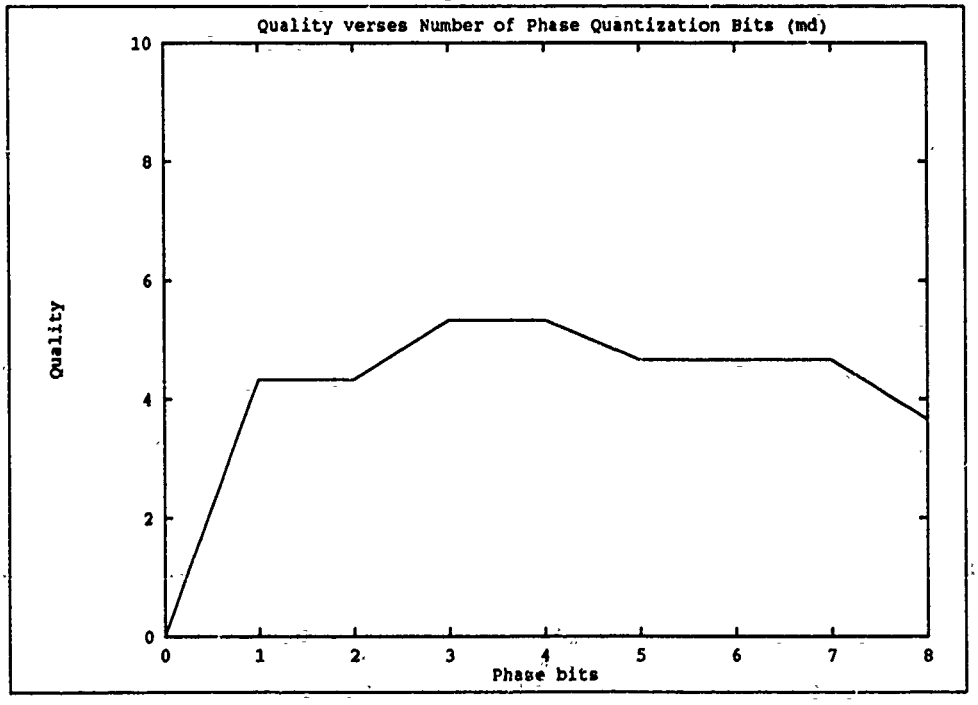



Figure 36. Average Quality Score verses Number of Phase Bits (P)

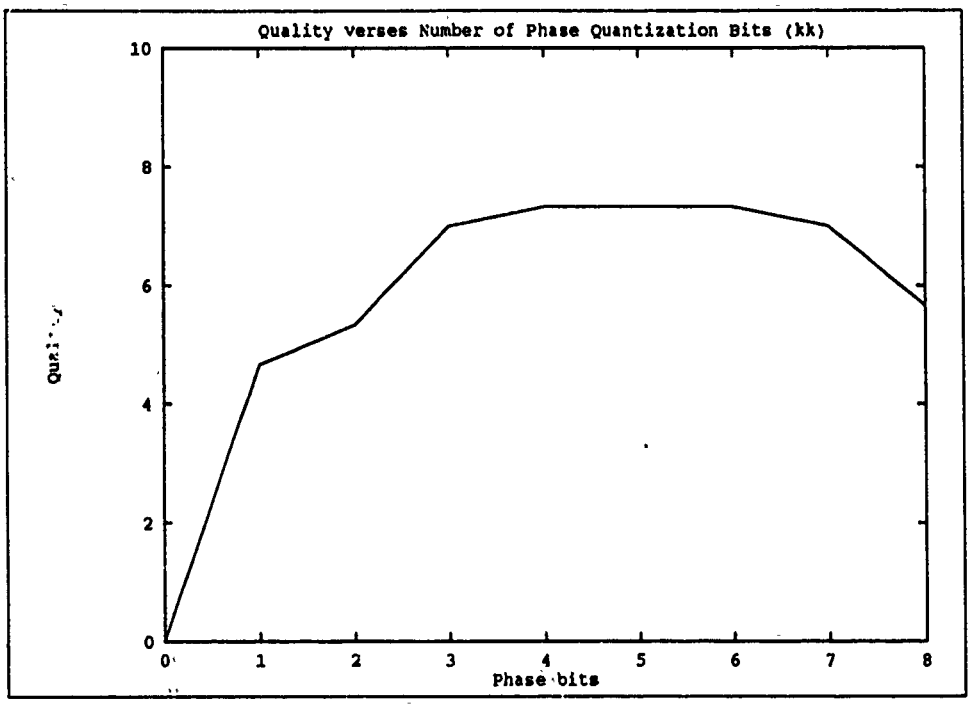



Figure 37. Average Quality Score verses Number of Phase Bits (P)

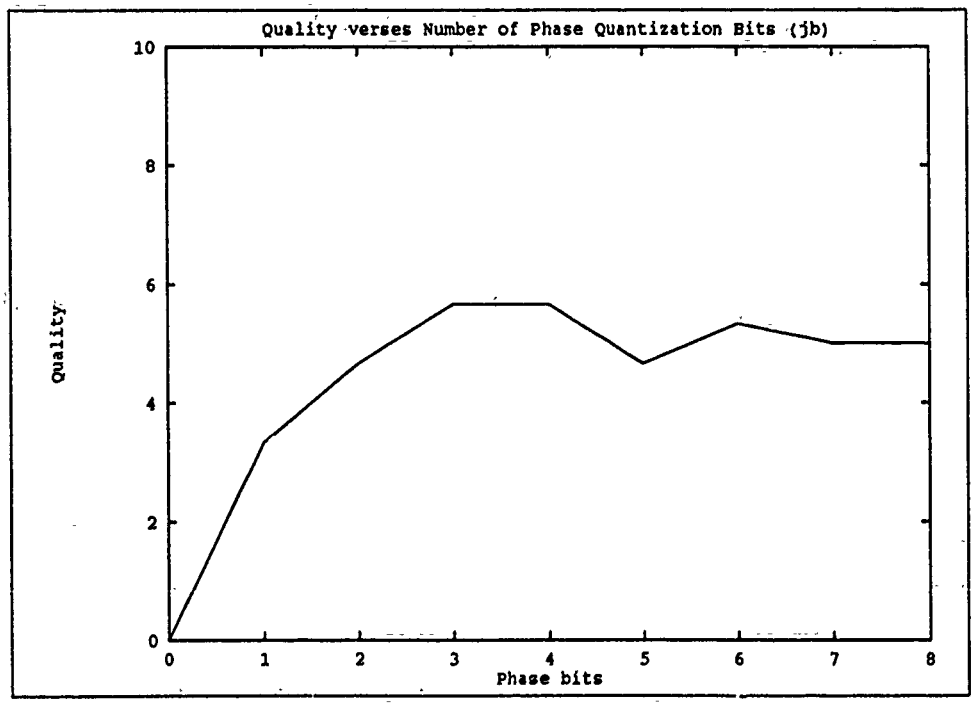



Figure 38. Average Quality Score verses Number of Phase Bits (P)

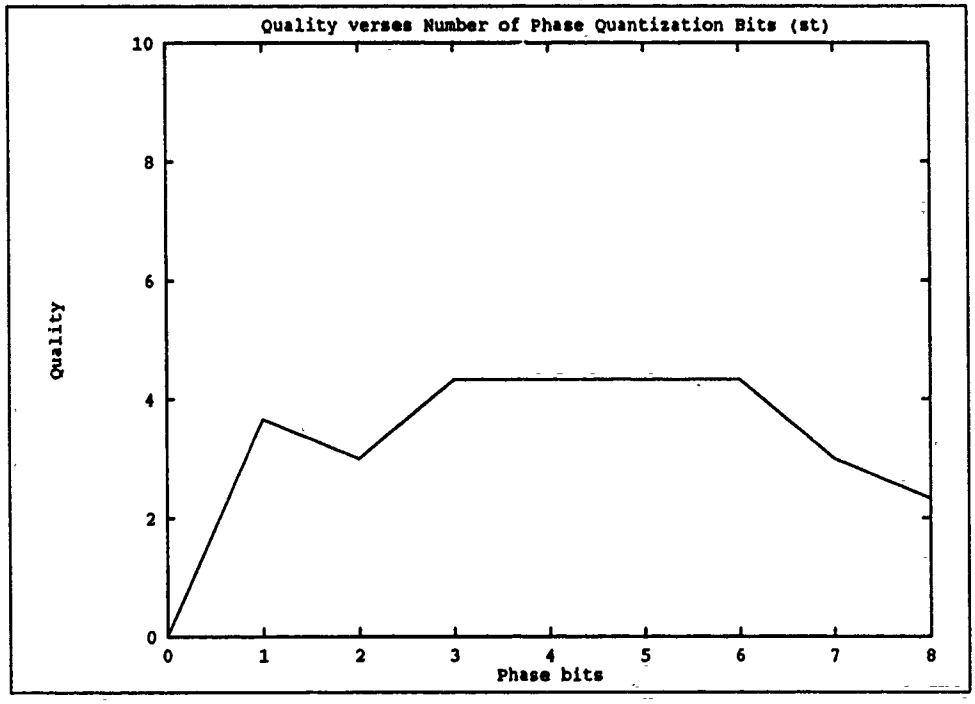



Figure 39. Average Quality Score verses Number of Phase Bits (P)

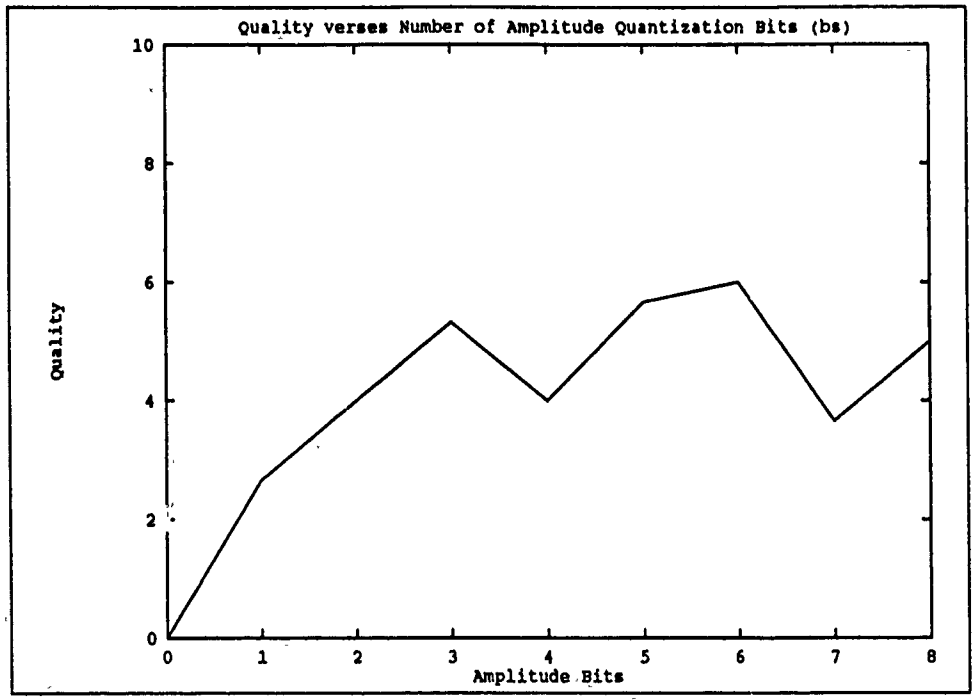



Figure 40. Average Quality Score verses Number of Amplitude Bits (A)

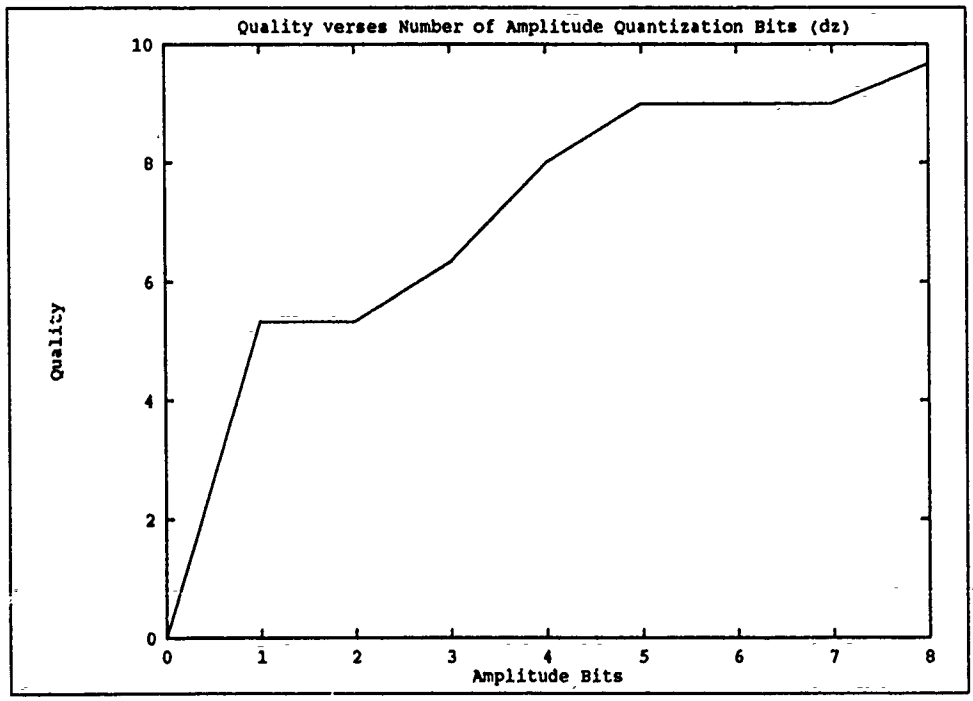



Figure 41. Average Quality Score verses Number of Amplitude Bits (A)

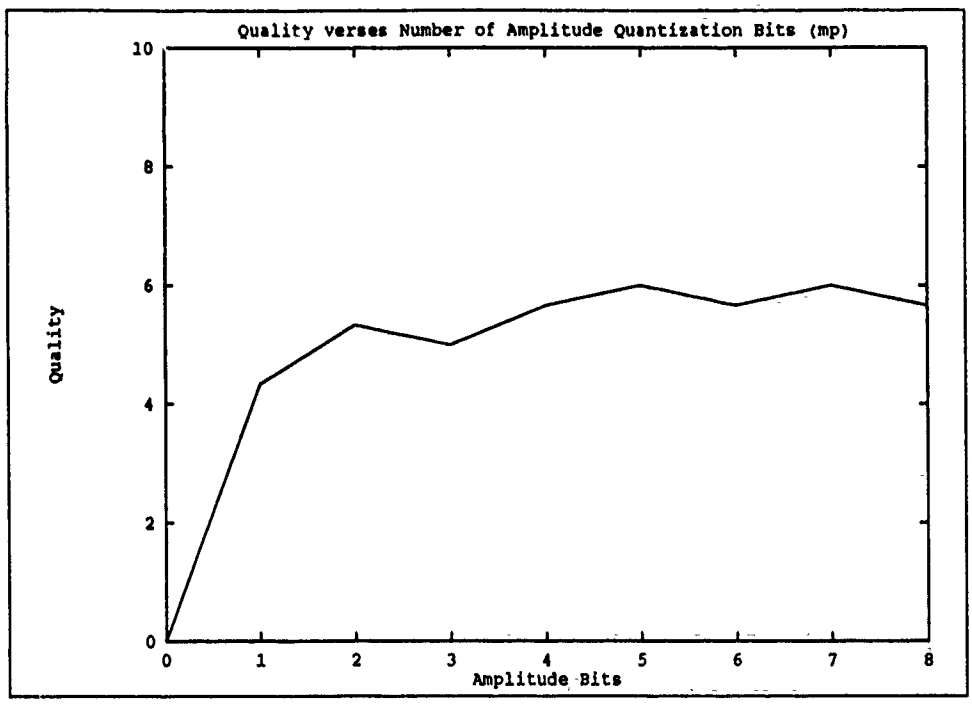



Figure 42. Average Quality Score verses Number of Amplitude Bits (A)

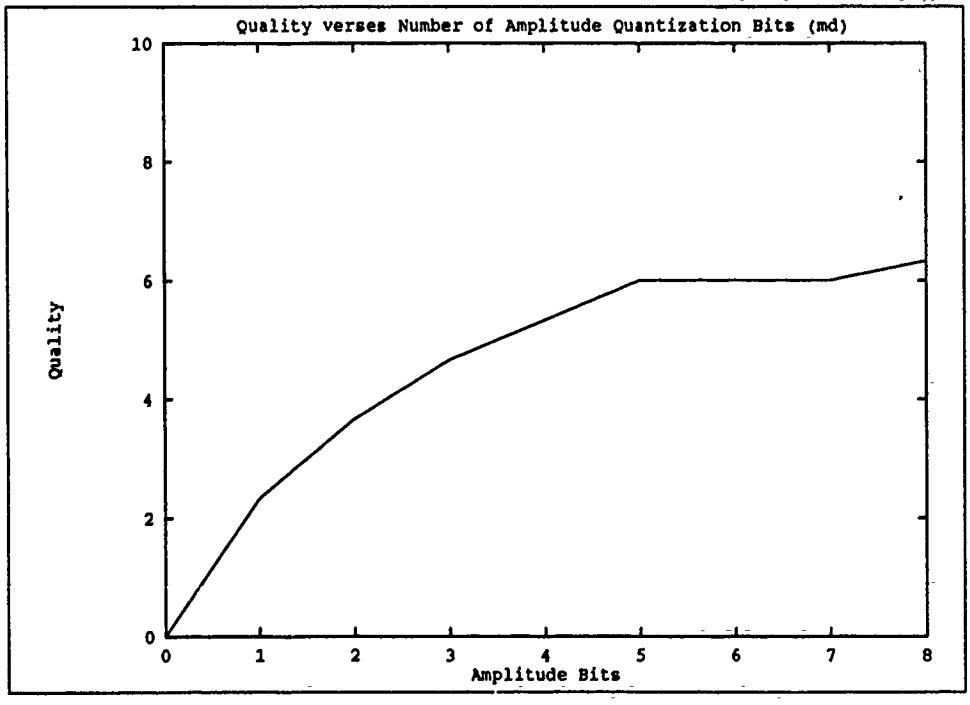



Figure 43. Average Quality Score verses Number of Amplitude Bits (A)

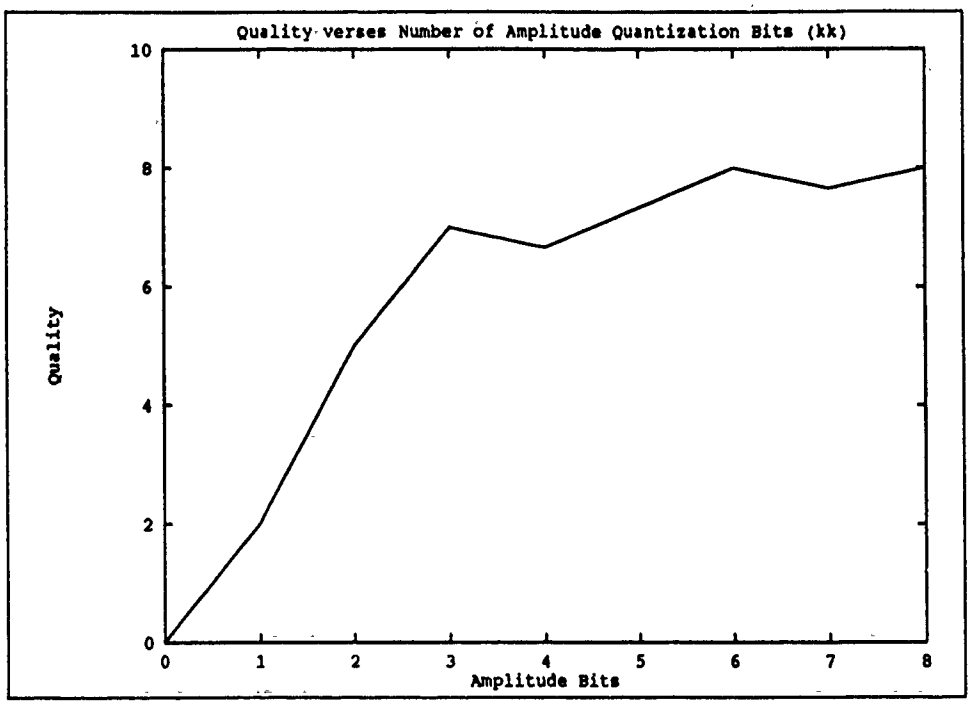



Figure 44. Average Quality Score verses Number of Amplitude Bits (A)

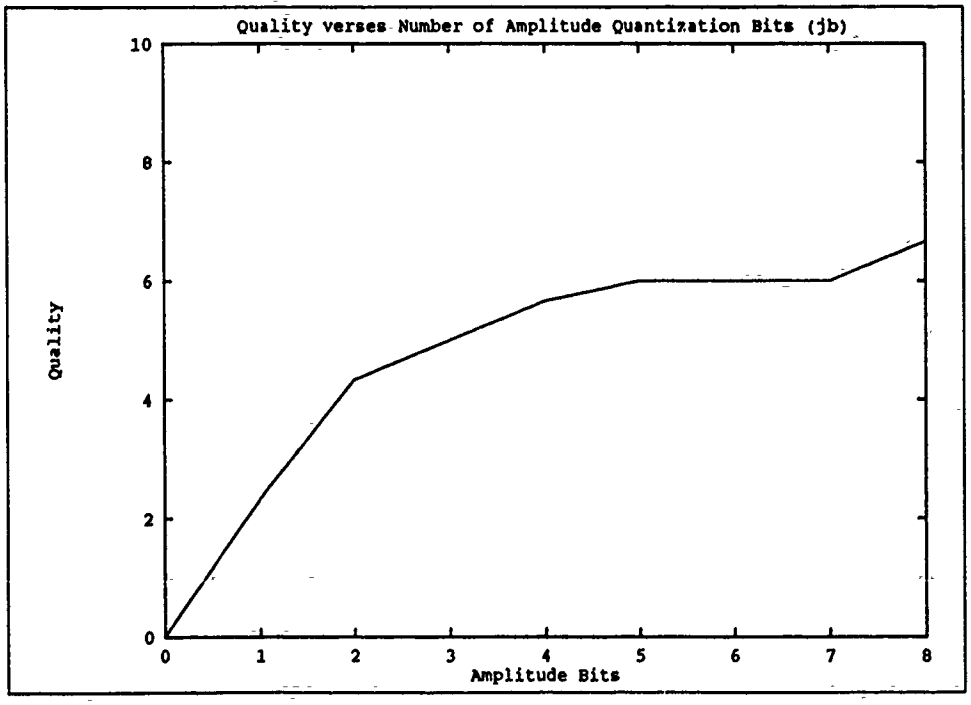



Figure 45. Average Quality Score verses Number of Amplitude Bits (A)

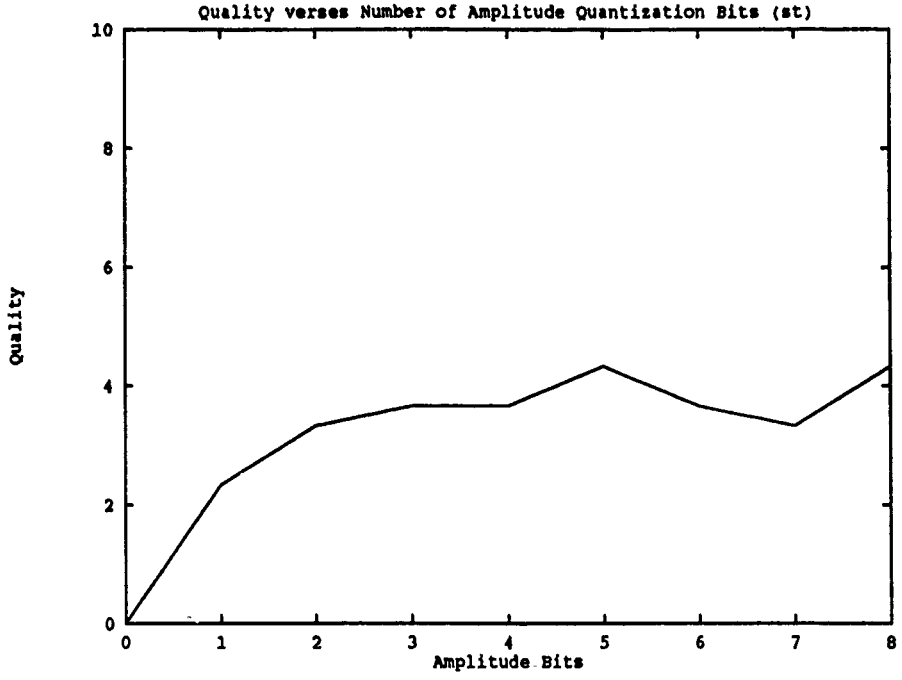



Figure 46. Average Quality Score verses Number of Amplitude Bits (A)

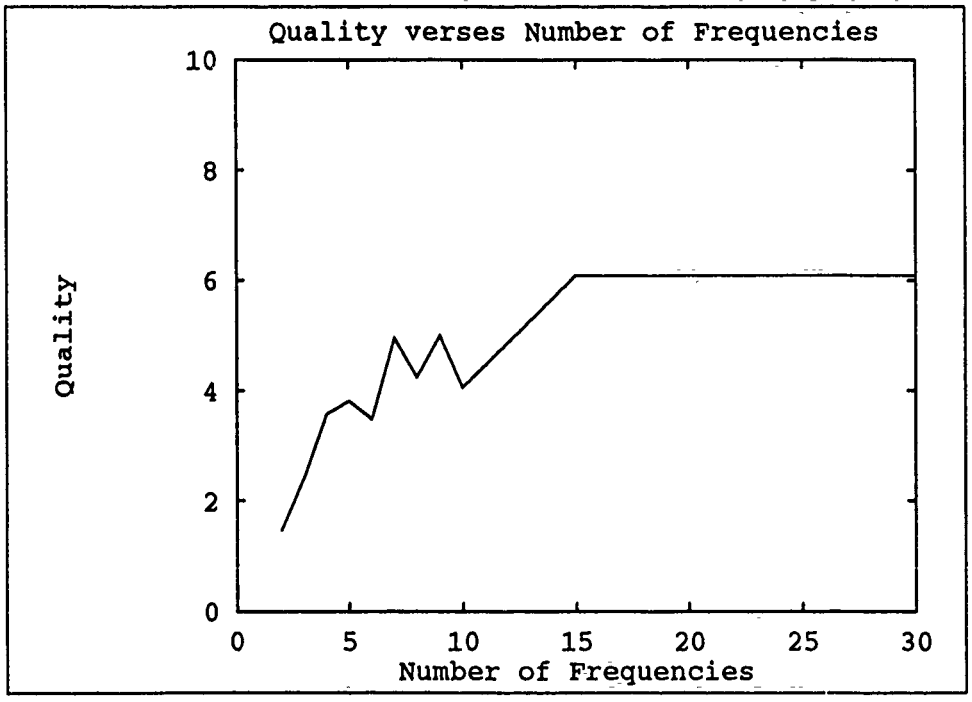



Figure 47. Average Quality Score verses Number of Frequencies (L) All Listeners

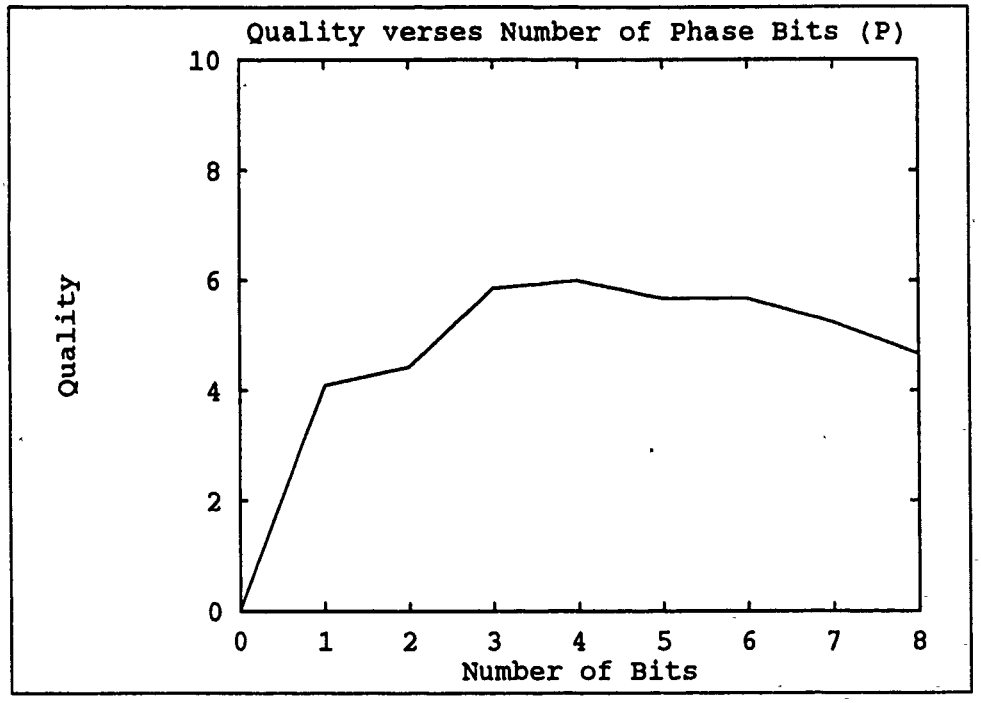



Figure 48. Average Quality Score verses Number of Phase Bits (P) All Listeners

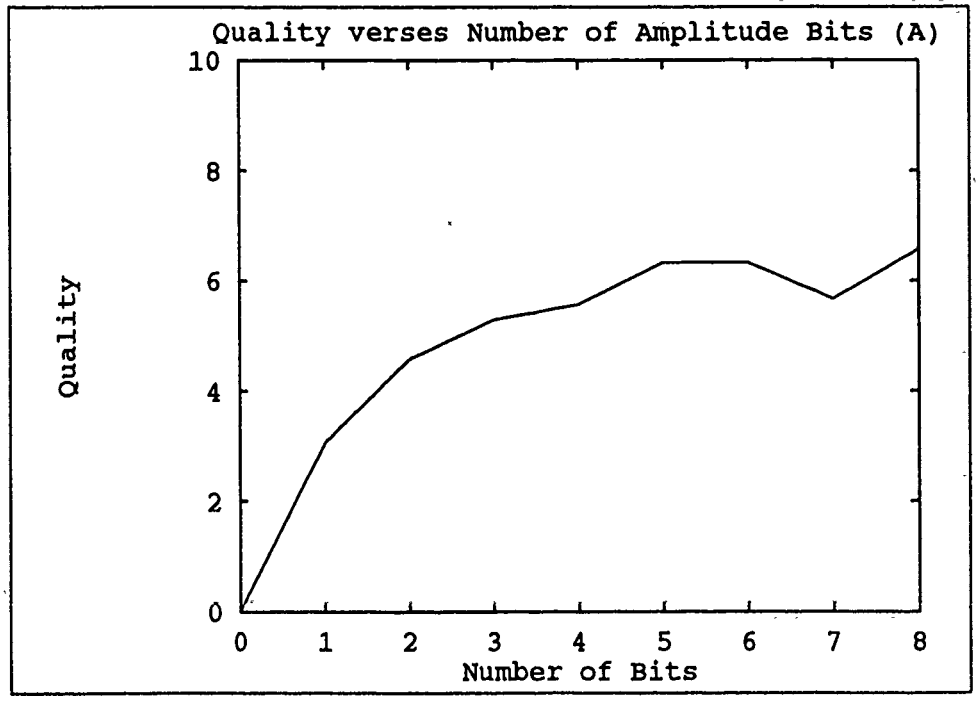



Figure 49. Average Quality Score verses Number of Amplitude Bits (A) All Listeners

# Appendix B. Computer Source Code

```
/*********************************************************/
/**                                                     **/
/**              SPEECH PROCESSING SYSTEM               **/
/**                                                     **/
/*********************************************************/

# include stdio
# include math
# include stdlib
# define pi  3.1415926535
# define e  2.71828182846
# define hw 1
# define conv 0
# define slice 640
# define jitter 2

double  max, maxfreq, compensate_vector[256], quant_vector[256];
double  qexp_amp_vector[256], count_vector[512], vcutoff;
double  bcutoff;
int freq_vector[256], vecsize;

main()
{
double   temp_even[slice], temp_odd[slice], txamp_vector[256];
double   output_vector[slice], txphase_vector[256];
double   time_vector[slice], exp_amp_vector[512], check;
double   hamming_frame[1024], amp_vector[512], phase_vector[512];
double   timeframe_vector[slice], exp_phase_vector[512];
int      i, j, k, r, n, x, b1, b2, b3, h1, size, begin;
int      putfile, end, count, harm[256], sum;
int      amp_bits, phase_bits, window_size, components;
```

```
char     Temp_in[25], Temp_in_file[22],Temp_out_file[22];
char     Temp_out[25], temp_file[25];
FILE     *input_ptr, *output_ptr, *data_ptr, *temp_ptr;



printf (" Enter input file name less (.snd) \n");
scanf ("%s", Temp_in_file);
printf (" Enter output file name less (.out) \n");
scanf ("%s", Temp_out_file);
printf(" Enter the number of amplitude bits\n");
scanf("%d", &amp_bits);
printf(" Enter the number of phase bits\n");
scanf("%d", &phase_bits);
/*printf(" Enter the voiced cutoff threshold \n");
scanf("%f", &vcutoff);
printf(" Enter the blank space cutoff threshold \n");
scanf("%f", &bcutoff);*/
printf("Enter the number of spectral components to be chosen \n");
scanf("%d", &components);
sprintf (Temp_in, "%s.dat", Temp_in_file);
sprintf (Temp_out, "%s.out", Temp_out_file);
input_ptr = fopen (Temp_in, "r");
output_ptr = fopen (Temp_out, "wr");

i = window_size % 2;
i=1;
if (i !=1 )
{
  ' window size must be odd \n  Enter window size\n");
scanf( %d", &window_size);
}

/*  open file, read to end of file  */
```

```
max = 0;
maxfreq = 0;
i=1;
begin = 1;
sum = 0;

setup_quant(amp_bits);

setup_compvec(compensate_vector);

/* read in header */

for (r = 0; r < 128; ++r)
{
b3 = getw(input_ptr);
putw(b3,output_ptr);
}

while (!feof(input_ptr))
  {

/* take pointer past header data and read in data
   with 50 percent overlap */

j = slice + (i-1)*(slice);
    fseek (input_ptr,j,0);

/*  read in speech time frame vectors  */

    for ( k = 0;  k < slice && !feof(input_ptr);  ++k )
      {

/* read in byte one and byte two then convert msd */
```

```
        b1 = getc(input_ptr);
        b2 = getc(input_ptr);
        h1 = b2*256 + b1;

/* convert negative numbers by two's compliment */

          if (h1 >= 32767)
              h1 = h1 - 65536;

          timeframe_vector[k] = (float)h1;
       } /** end for k **/


      hamming_window(timeframe_vector,hamming_frame);

      fft(1024,hamming_frame,amp_vector, phase_vector);

      freq_select(amp_vector, phase_vector, txamp_vector,
         txphase_vector, freq_vector);

      comp_select(txamp_vector, txphase_vector, freq_vector,
      harm, components, i);

      if (txamp_vector[0] > 0)
      energy_conserve(amp_vector, txamp_vector, components);

      expand(txamp_vector, txphase_vector, freq_vector, harm,
             exp_amp_vector, exp_phase_vector, components);

      quantphase(exp_phase_vector, phase_bits);

      quantamp(exp_amp_vector, qexp_amp_vector, amp_bits);

/*      convolve(exp_amp_vector, exp_phase_vector, harm,
```

```
            freq_vector, components);*/

    freq_to_time(qexp_amp_vector, exp_phase_vector, time_vector);

    reduce(time_vector, output_vector, i, temp_even, temp_odd);

 putfile = i % 2;

    if (putfile == 0)
        normalize(output_vector, output_ptr);

begin = 0;

 ++i;

  } /** end while (!eof) **/

 }/** end main **/


/*****************************************************************/
/*                                                               */
/*          SUBROUTINE SETUP COMPENSATE  VECTOR                  */
/*                                                               */
/*    This routine sets up a vector which is used to             */
/*    multiply the amplitude vector for decision making.         */
/*    The vector compensates for the 6 dB per octave             */
/*    rolloff in energy in speech signals.                       */
/*                                                               */
/*****************************************************************/


setup_compvec(compensate_vector)
```

```
double  compensate_vector[];
{
int  n;

for (n = 0; n < 38; ++n)
compensate_vector[n] = 1;

for (n = 38; n < 77; ++n)
compensate_vector[n] = pow(e, .0355*(n-38));

for (n = 77; n < 154; ++n)
compensate_vector[n] = 4*pow(e, .018*(n-77));

for (n = 154; n < 256; ++n)
compensate_vector[n] = 16*pow(e, .009*(n-154));

}


/*********************************************************/
/*                                                       */
/*                  SUBROUTINE WINDOW                    */
/*                                                       */
/*      This subroutine sets up a vector called window   */
/*      for multiplication with each time frame vector.  */
/*      Windowing prevents the spectral leakage caused   */
/*      by multiplying the time waveform by a rect.      */
/*                                                       */
/*********************************************************/


hamming_window (timeframe_vector, hamming_frame)

double timeframe_vector[],hamming_frame[];
```

```
{
int z, y, s;
double  ham_vector[slice];

for (s = 0; s < 1024; ++s)
    hamming_frame[s] = 0;

for (y=0; y < slice; ++y)
ham_vector[y] = sin(pi*y/(slice-1));


for (z = 0; z < slice; ++z)
   {
    hamming_frame[z] = timeframe_vector[z]*ham_vector[z];
   }
} /** end hamming window **/


/*************************************************************/
/*                                                           */
/*                    SUBROUTINE FFT                         */
/*                                                           */
/*      This subroutine takes the digitized speech           */
/*      timeframes and performs a fast Fourier transform     */
/*      on the data.  The output is two vectors:             */
/*      amp_vector represents the magnitude of the           */
/*      transform and phase_vector represents the phase      */
/*      of the transform                                     */
/*                                                           */
/*************************************************************/


fft(n, hamming_frame, amp_vector, phase_vector)
```

```
int  n;
double  hamming_frame[], amp_vector[], phase_vector[];

{
    int nv,nm,i,j,k,m,le,ld,p;
    double  xi[1024],ur,ui,rt,it,wr,wi,up;


        for (i = 0; i < n; i++)
        {
            xi[i] = 0;
        }
        nv = n/2; nm = n - 1; j = 1;
        m = log((n + 1)* 1.0)/log(2.0);

        for (i = 1; i <= n; i++)
            {
              hamming_frame[n-i+1] = hamming_frame[n-i];
              xi[n-i+1] = xi[n-i];
            }

        for ( i = 1; i <= nm; i++ )
            {
              if (i<j)
                 {
                    rt = hamming_frame[j];
                    it = xi[j];
                    hamming_frame[j] = hamming_frame[i];
                    xi[j] = xi[i];
                    hamming_frame[i] = rt;
                    xi[i]=it;
                 }
                k= nv;
```

```
        while (k<j)
          {
            j-=k;
            k/=2;
          }
        j+=k;
     }

for (k=1; k<=m; k++)
  {
    ld = pow(2.0,k*1.0);
    le = ld/2;
    wr = cos(pi/le);
    wi = -sin(pi/le);
    ur = 1.0;
    ui = 0.0;

    for (j = 1; j <= le; j++ )
        {

        for( i = j; i <= n; i+=ld)
           {
           p = i + le;
           rt = hamming_frame[p]*ur-xi[p]*ui;
           it = hamming_frame[p]*ui+xi[p]*ur;
           hamming_frame[p] = hamming_frame[i]-rt;
           xi[p] = xi[i] - it;
           hamming_frame[i] = hamming_frame[i] + rt;
           xi[i] = xi[i] + it;
           }

        up = ur*wr - ui*wi;
        ui = ui*wr + ur*wi;
        ur = up;
```

```
            }
        }
        for (i = 1; i <= n; i++ )

            {
                hamming_frame[i-1] = hamming_frame[i];
                xi[i-1] = xi[i];
            }
            for (j = 0; j < 512; ++j)
                {
                amp_vector[j] = sqrt(hamming_frame[j]*hamming_frame[j] +
                                xi[j]*xi[j]);
                count_vector[j]=count_vector[j]+amp_vector[j]/60;
                if (hamming_frame[j] == 0 && xi[j] == 0)
                phase_vector[j] = 0;
                else
                phase_vector[j] = atan2(xi[j],hamming_frame[j]);
                }

} /** end fft **/


/*****************************************************************/
/*                                                               */
/*                   SUBROUTINE FILTER                           */
/*                                                               */
/*      This subroutine takes in the amplitude vector from       */
/*      the FFT and smoothes the curve by averaging all          */
/*      values within a window size.  The routine passes         */
/*      back a smoothed amplitude vector.                        */
/*                                                               */
/*****************************************************************/
```

```
filter(window_size, amp_vector)

int   window_size;
double  amp_vector[];

{

int   i, j;
double   temp, filtered_amp[256];

for (i = 0;  i < 256; ++i)
    filtered_amp[i] = amp_vector[i];


for (i=0; i<(256-window_size); ++i)
  {

    temp = 0;

    for (j = i; j < i+window_size; ++j)
       {
         temp = temp+amp_vector[j];
       } /** end for j **/

    filtered_amp[i+(window_size+1)/2] = temp/window_size;

  } /** end for i **/
for (i = 0; i < 256; ++i)
   amp_vector[i] = filtered_amp[i];

} /** end filter **/
```

```
/*********************************************************************/
/*                                                                   */
/*              SUBROUTINE  FREQUENCY SELECT                         */
/*                                                                   */
/*      This subroutine takes the spectral components of the         */
/*      timeframe and selects the critical components for            */
/*      transmission to the receiver.                                */
/*                                                                   */
/*********************************************************************/



freq_select (amp_vector, phase_vector, txamp_vector,
              txphase_vector, freq_vector)

double   amp_vector[], phase_vector[] ,txamp_vector[];
double   txphase_vector[];
int      freq_vector[];

{
double  glottal_amp, harm_amp, decide_vector[256];
double  compen_vector[256];
int     n;
int     glottal_bin, glot_harm, harm_bin,i,x,y;

glottal_amp = 0.0;
glottal_bin = 0;


  for (i = 0; i < 256; ++i)
{
```

```
decide_vector[i] = amp_vector[i]*compensate_vector[i];
txamp_vector[i] = 0;
}

   for (i = 5; i < 16; ++i)
    {
        if (abs(decide_vector[i]) >= abs(glottal_amp))
          {
            glottal_amp = decide_vector[i];
            glottal_bin = i;
           } /* end if */
    } /* end for */

        txamp_vector[0] = amp_vector[glottal_bin];
        freq_vector[0] = glottal_bin;
        txphase_vector[0] = phase_vector[glottal_bin];
        y = 2;

    if (glottal_bin ==0)
        glottal_bin = 7;

    while (y*glottal_bin < 256)
    {
     harm_amp = 0.0;
     harm_bin = 0;
     for (x = ((y*glottal_bin)-jitter); x <= ((y*glottal_bin)+jitter); ++x)
      {
        if (abs(decide_vector[x]) >= abs(harm_amp))
          {
            harm_amp = decide_vector[x];
            harm_bin = x;
          } /* end if */
      } /* end for */
    txamp_vector[y-1] = amp_vector[harm_bin];
```

```
    txphase_vector[y-1] = phase_vector[harm_bin];
    freq_vector[y-1] = harm_bin  - (y*glottal_bin);
    ++y;
    } /* end while */
    vecsize = y;
} /* end freq_select */



/*****************************************************************/
/*                                                               */
/*              SUBROUTINE COMPONENT SELECT                      */
/*                                                               */
/*      This subroutine takes the vector which is comprised      */
/*      of the harmonics of the estimated glottal frequency      */
/*      (txamp_vector) multiplies it by the compensation         */
/*      vector then select the n greatest quantities for         */
/*      speech reconstruction.  n is chosen by the user.         */
/*                                                               */
/*****************************************************************/


comp_select(txamp_vector, txphase_vector, freq_vector, harm,
            components, y)

double   txamp_vector[], txphase_v   or[];
int      components, harm[], freq_vector[], y;


{

double   temp1[256],temp2[256],temp3[256], temp;
double   decide_vector[256], compensa_vector[256];
int      x, i, a, c, d, n;
```

```
c = freq_vector[0];
for (i = 1; i < vecsize; ++i)
  {
        d = c*(i+1) + freq_vector[i];
        temp1[i]= txamp_vector[i];
        temp2[i]= txphase_vector[i];
        temp3[i]= freq_vector[i];
        txamp_vector[i] = 0;
        txphase_vector[i] = 0;
        freq_vector[i] = 0;
        decide_vector[i]=temp1[i]*compensate_vector[d];
   } /*  end for i */


for (x = 1; x < components; ++x)
  {
    temp = 0;

     for(i = 1; i < vecsize; ++i)
     {
        if (abs(decide_vector[i]) >= abs(temp))
         {
          harm[x]=i;
          temp = decide_vector[i];
         } /* end if */
      } /*  end for i */

    a = harm[x];
    txamp_vector[x] = temp1[a];
    txphase_vector[x] = temp2[a];
    freq_vector[x] = temp3[a];
    decide_vector[a] = 0;
    decide_vector[a+1] = .7*decide_vector[a+1];
```

```
   decide_vector[a-1] = .7*decide_vector[a-1];
   decide_vector[a-2] = .85*decide_vector[a-2];
   decide_vector[a+2] = .85*decide_vector[a+2];
   printf("harmonic chosen = %d",harm[x]);
 } /* end for x */

} /* end component select */



/*****************************************************************/
/*                                                               */
/*              SUBROUTINE PHASE QUANTIZATION                    */
/*                                                               */
/*      This subroutine takes the continuous phase vectors       */
/*      obtained from the FFT and quantizes them.  This          */
/*      simulates the process which would be taken to encode     */
/*      the phases into binary code.                             */
/*                                                               */
/*****************************************************************/

quantphase(exp_phase_vector, phase_bits)

double   exp_phase_vector[];
int      phase_bits;

{

int  i, num;
double  step_size, rem;

for (i = 0; i < 256; ++i)
{
  if (exp_phase_vector[i] < 0)
```

```
    exp_phase_vector[i] = exp_phase_vector[i]+2*pi;
}

step_size = 2*pi/(pow(2,phase_bits));

for (i = 0; i < 256; ++i)
  {
    if(exp_phase_vector[i] != 0)
     {
       num = exp_phase_vector[i]/step_size;
       rem = exp_phase_vector[i]-num*step_size;
         if(rem > step_size/2)
           {
              exp_phase_vector[i] = (num+1)*step_size;
           }
         else
           {
            exp_phase_vector[i] = num * step_size;
           }
     }
   }
}


/*****************************************************************/
/*                                                               */
/*              SUBROUTINE AMPLITUDE QUANTIZATION                */
/*                                                               */
/*      This subroutine takes the continuous amplitude vectors   */
/*      obtained from the FFT and quantizes them.  This          */
/*      simulates the process which would be taken to encode     */
/*      the amplitudes into binary code.                         */
/*                                                               */
/*****************************************************************/
```

```
quantamp(exp_amp_vector,qexp_amp_vector, amp_bits)

double exp_amp_vector[], qexp_amp_vector[];
int    amp_bits;

  {

int  x, steps, num;
double comp_quant, rem, step_size;

steps = pow(2,amp_bits);
step_size = 800000/(steps-1);

for (x = 0; x < 256; ++x)
   qexp_amp_vector[x] = 0;

for (x = 0; x < 256; ++x)
 {
 if(exp_amp_vector[x] > 1)
  {
  comp_quant = step_size/compensate_vector[x];
  num = exp_amp_vector[x]/comp_quant;
  rem = exp_amp_vector[x]- num*comp_quant;
   if(rem > comp_quant/2)
      qexp_amp_vector[x] = (num + 1)*comp_quant;
   else
      qexp_amp_vector[x] = num*comp_quant;
 } /* end if */
  } /* end for x */
}


/*****************************************************/
```

```
/*                                                                  */
/*              SUBROUTINE ENERGY CONSERVATION                      */
/*                                                                  */
/*      This subroutine compensates for the fact that               */
/*      different amounts of energy is eliminated from frame        */
/*      to frame by the frequency select subroutine.                */
/*      Compensation is accomplished by calculating the             */
/*      energy in the origional frame and the tx frame, then        */
/*      a compenstion factor is calculated for multiplication       */
/*      of the spectral components of the tx amplitude vector.      */
/*                                                                  */
/********************************************************************/

energy_conserve(amp_vector, txamp_vector,    onents)

double  amp_vector[], txamp_vector[];
int     components;

{

int   i;

double  temp1, temp2, comp_factor;


temp1=0;
temp2=0;
comp_factor=0;

    for (i = 0; i < 256; ++i)
      {
        temp1 = temp1 + pow(amp_vector[i],2);
      } /* end for i */
```

```
    for (i = 0; i < components; ++i)
      {
        temp2 = temp2 + pow(txamp_vector[i],2);
      }

   if (temp1 > 1700000000.0)
     comp_factor = sqrt(temp1/temp2);
   else
     comp_factor = 1;

     for (i = 0; i < 256; ++i)
        txamp_vector[i] = txamp_vector[i]*comp_factor;

} /* end energy conserve */



/*****************************************************************/
/*                                                               */
/*                    SUBROUTINE EXPAND                          */
/*                                                               */
/*      This subroutine expands the transmitted amplitude        */
/*      frame from its origional size to the size which          */
/*      matches the origionl amplitude vector                    */
/*                                                               */
/*****************************************************************/

expand(txamp_vector,txphase_vector, freq_vector, harm,
      exp_amp_vector, exp_phase_vector, components)

double   txamp_vector[], txphase_vector[], exp_amp_vector[];
double   exp_phase_vector[];
int      freq_vector[], harm[], components;
```

```
{

int  x, y, z;
double   sinc_vec[11], temp_vec[11], hold;
int  c;
int  d;
int  i;

int  h;

sinc_vec[5] = 1;
for (c = 6; c < 11; ++c)
sinc_vec[c] = sinc_vec[10-c] =sin(pi*(c-5)/2)/(pi*(c-5)/2);


x = freq_vector[0];

for (y = 0; y < 256; ++y)
{
   exp_phase_vector[y] = 0;
   exp_amp_vector[y] = 0;
} /** end for y **/

if (conv == 1)
{
hold = txamp_vector[0];

for (c = 0; c < 11; ++c)
{
exp_amp_vector[c+(x-5)]= hold*sinc_vec[c];
exp_phase_vector[c+(x-5)] = txphase_vector[0];
}

for (y = 1; y < components; ++y)
```

```
{
z = x*(harm[y]+1)+ freq_vector[y];

hold = txamp_vector[y];
for (i = 0; i < 11; ++i)
{
exp_amp_vector[i+(z-5)]=hold*sinc_vec[i];
exp_phase_vector[i+(z-5)] = txphase_vector[y];
}
} /** end for y **/
}
else
{
exp_amp_vector[x] = txamp_vector[0];
exp_phase_vector[x] = txphase_vector[0];

for (y = 1; y < components; ++y)
{
z = x*(harm[y]+1) + freq_vector[y];
exp_amp_vector[z] = txamp_vector[y];
exp_phase_vector[z] = txphase_vector[y];
} /** end for y **/
} /** end else **/

} /** end expand **/


/*********************************************************/
/*                                                       */
/*              SUBROUTINE CONVOLVE                      */
/*                                                       */
/*      This subroutine takes the reduced speech         */
/*      spectra and convolves it with a sinc function.   */
/*                                                       */
```

```
/*********************************************************


convolve(exp_amp_vector, exp_phase_vector, harm, freq_vector, components)

double   exp_amp_vector[], exp_phase_vector[];
int      harm[], freq_vector[], components;

{
double   sinc_vec[11], temp_vec[11], hold;
int  c;
int  d;
int  i;
int  h;

sinc_vec[5] = 1;
for (c = 6; c < 11; ++c)
sinc_vec[c] = sinc_vec[10-c] =sin(pi*(c-5)/2)/(pi*(c-5)/2);


i = freq_vector[0];

for (c = 0; c < 11; ++c)
temp_vec[c] = exp_amp_vector[i]*sinc_vec[c];

exp_amp_vector[i] = 0;

for (c = i-5; c < i+6; ++c)
{
exp_amp_vector[c] = temp_vec[c-(i-5)];
exp_phase_vector[c] = exp_phase_vector[i];
printf("tv = %f",temp_vec[c-(i-5)]);
}
```

```
for (c =1; c < components; ++c)
  {
     h=e*(harm[c]+1)+freq_vector[c];
     hold = exp_amp_vector[h];
     exp_amp_vector[h] = 0;
       for( d = 0; d < 11; ++d)
           temp_vec[d] = hold*sinc_vec[d];
       for (d = h-5; d < h + 6; ++d)
          {
           exp_amp_vector[d] = temp_vec[d-(h-5)];
           printf("TV2 = %f",temp_vec[d-(h-5)]);
           exp_phase_vector[d] = exp_phase_vector[h];
          }
     }
} /** end convolve **/



/*****************************************************************/
/*                                                               */
/*          FREQUENCY TO TIME DOMAIN CONVERSION                  */
/*                                                               */
/*      This subroutine takes the selected spectral              */
/*      component's amplitude, phase and frequency and           */
/*      reconstucts an approximation of the origional speech     */
/*      using the sinusoidal model of speech signals.            */
/*                                                               */
/*****************************************************************/


freq_to_time(exp_amp_vector, exp_phase_vector, time_vector)

double exp_amp_vector[], exp_phase_vector[], time_vector[];
```

```
{

int  a, b, c, d;

   for (a = 0; a < slice; ++a)
     {
        time_vector[a] = 0;

          for (b = 0; b < 256; ++b)
            {
              if (abs(exp_amp_vector[b]) > 100)
                time_vector[a] = time_vector[a] +
                  (exp_amp_vector[b]*cos((2*pi*b*a/1024)
                  + exp_phase_vector[b]));
            }
     }
}


/*****************************************************************/
/*                                                               */
/*                  SUBROUTINE REDUCE                            */
/*                                                               */
/*      This routine reduces the 2n-1 overlapped frames          */
/*      to the origional number of frames (n).   The frames      */
/*      are multiplied by the sin window function before         */
/*      reduction to smooth the effects of discontinuities       */
/*      from one frame to the next.                              */
/*                                                               */
/*****************************************************************/



reduce(time_vector ,output_vector, i, temp_even, temp_odd)
```

```
double time_vector[], output_vector[], temp_even[], temp_odd[];
int  i;

{

int x, k;


if (i == 1)
   {
      for (x = 0; x < slice; ++x)
         temp_odd[x] = time_vector[x]*sin(pi*x/(slice-1));
    }
else
    {
      k = i % 2;

      if (k == 0)
         {
            for (x = 0; x < slice; ++x)
                temp_even[x] = time_vector[x]*sin(pi*x/(slice-1));

            for (x = 0; x < slice/2; ++x)
              {
                 output_vector[x] = temp_odd[x];
                 output_vector[x+(slice/2)] = (temp_odd[x+(slice/2)] +
                    temp_even[x]);
              }
             for (x = 0; x < slice; ++x)
                  temp_odd[x] = 0;
            }
       else
            {
```

```
            for (x = 0; x < slice; ++x)
                temp_odd[x] = time_vector[x]*sin(pi*x/(slice-1));

            for (x = 0; x < slice/2; ++x)
                 temp_odd[x] = temp_odd[x] + temp_even[x+(slice/2)];

            for (x = 0; x < slice; ++x)
                 temp_even[x] = 0;
            }
    }
}


/*************************************************************************/
/*                                                                       */
/*                    SUBROUTINE NORMALIZE                               */
/*                                                                       */
/*      This subroutine takes the approximation of the origional         */
/*      speech waveform and normalizes the data to make it               */
/*      compatable with the DSC digital to analog convertor.             */
/*      The normalized data is then written to an output file            */
/*      specified by the user.                                           */
/*                                                                       */
/*************************************************************************/


normalize(output_vector, output_ptr)

double  output_vector[];
FILE    *output_ptr;

{

int     byte1, byte2, intnumber, i;
```

```
char    bytechar1, bytechar2;
double   number;

                for (i = 0; i < slice; ++i)
                 {
                  intnumber = (int) output_vector[i]/225;
                    if (intnumber < 0)
                        intnumber = intnumber + 65536;


                  byte1 = (int) intnumber % 256;
                  byte2 = (int) intnumber/256;
                  bytechar1 = (char) byte1;
                  bytechar2 = (char) byte2;

                  putc(bytechar1, output_ptr);
                  putc(bytechar2, output_ptr);
               }

}
```

# *Bibliography*


1. Alenquer, Capt. Luis M. F. *Rule Based Sinusoidal Encoding of Speech.* Master's Thesis, Air Force Institute of Technology, 1990.
2. Bashir, Flt. Lt. Nadeem A. *Phonem Adjustment in Enhanced Speech.* Master's Thesis, Air Force Institute of Technology, 1989.
3. Cherry, Colin. *On Human Communication (Second Edition).* Cambridge, Massachusetts and London, England: The M.I.T. Press, 1966.
4. Harris, Frederick J. "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform." *Proceedings of the IEEE*, Vol. 66 No. 1 : 51-83 (January 1978).
5. Kabrisky, Matthew, Professor. Personal Discussion. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH., February 1990.
6. Ludeman, Lonnie C. *Fundamentals of Digital Signal Processing.* New York: John Wiley and Sons, 1986.
7. Markel, John D. and Augustine H. Gray Jr. *Linear Prediction of Speech* New York: Springer-Verlag, 1976.
8. Marple S. Lawrence Jr. *Digital Spectral Analysis with Applications* Englewood Cliffs NJ: Prentice-Hall Inc., 1987.
9. McAulay, Robert J. and T. F. Quatieri. "Magnitude-only Reconstuction using a Sinsoidal Model of Speech." *IEEE Transactions of Acoustics, Speech, and Signal Processing.* : 489-492 (March 1985).
10. Parsons, Thomas W. *Voice and Speech Processing.* New York: McGraw Hill Book Company, 1987.
11. Rabiner, Lawrence R. and Ronald Schafer. *Digital Processing of Speech Signals.* New Jersey: Prentice-Hall Inc., 1978.
12. Sivian, L. J. "Speech Power and Its Measurement." *Bell Systems technical Journal.* : 646-661 (1929).
13. Sklar, Bernard. *Digital Communications Fundamentals and Applications.* New Jersey : Prentice Hall, 1988.
14. Sundberg, Johan. "The Acoustics of the Singing Voice," *Scientific American*, 82-91+ (March 1977).

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | December 1990 | Master's Thesis |

**4. TITLE AND SUBTITLE**

RULE-BASED FREQUENCY DOMAIN SPEECH CODING

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**

Vance M. McMillan, Capt, USAF

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Air Force Institute of Technology, WPAFB OH 45433-6583

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT/GE/ENG/90D-38

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

Approved for Public Release; Distribution Unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

A speech processing system is designed to simulate the transmission of speech signals using a speech coding scheme. The transmitter portion of the simulation extracts a minimized set of frequencies in Fourier space which represents the essence of each of the speech timeslices. These parameters are then adaptively quantized and transmitted to a receiver portion of the coding scheme. The receiver then generates an estimate of the original timeslice from the transmitted parameters using a sinusoidal speech model. After initial design, the thesis investigates how each of the design parameters affect the human perceived quality of speech. This is done with listening tests. The listening tests consist of having volunteers listen to a series of speech reconstructions. Each reconstruction is the result of the coding scheme acting on the same speech input file with the design parameters varied. The design parameters which are varied are: number of frequencies used in the sinusoidal speech model for reconstruction, number of bits to encode amplitude information, and number of bits used to code phase information. The final design parameters for the coding scheme were selected based on the results of the listening tests. Post design listening tests showed that the system was capable of 4800 bps speech transmission with a quality rating of five on a scale from zero (not understandable) to ten (sounds just like original speech).

**14. SUBJECT TERMS**

Speech Coding, Speech Compression, Rule-Based Coding

**15. NUMBER OF PAGES**

104

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |