EECS 651 Project Report

Design of a CELP coder and analysis of various quantization techniques

Prof. David L. Neuhoff

By:
Awais M. Kamboh
Krispian C. Lawrence
Aditya M. Thomas
Philip I. Tsai

Winter 2005
University of Michigan
Ann Arbor

Table of Contents

List of Figures

Introduction

Speech Coding

Speech coding has been a common area of research in signal processing since the introduction of wire-based telephones. Numerous speech coding techniques have been thoroughly researched and developed, spurned further by the advances in internet, technology and wireless communication. Speech coding is a fundamental element of digital communications, continuously attracting attention due to the increase of demands in telecommunication services and capabilities. Applications of speech coders for signal processing purposes has improved at a very fast pace throughout the years in order to allow it to take advantage of the increasing capabilities of communication technology infrastructure and computer hardware.

This project focuses on the area of speech coding. This particular area has become a fundamental necessity due to the bandwidth limitation of most signal transmission systems. Ideally in speech coding, a digital representation of a speech signal is coded using a minimum number of bits to achieve a satisfactory quality of the synthesized signal whilst maintaining a reasonable computational complexity. Speech coding has two main applications: digital transmission and storage of speech signals. In speech coding, our aim is to minimize the bit-rate while preserving a certain quality of speech signal, or to improve speech quality at a certain bit rate.

Currently, there are various kinds of coders being implemented. This project focuses on the design and implementation of a Code Excited Linear Predictive (CELP) coder. This Linear Predictive Coding (LPC) method performs LP analysis of speech by extracting the LP parameters and coefficients and employs a quantization method to search a codebook and compute the excitation signal. The quantization of the LP parameters, play an important role in the performance of the CELP coder. This project analyzes the performance of the CELP coder by using various quantization methods such as Scalar, vector, DPCM and TSVQ to quantize the LP parameters.

Speech Production

Before proceeding with the handling of digitized speech, it is crucial to have a basic understanding of how speech is produced. Speech is produced when the lungs force the direction of airflow to pass through the larynx into the vocal tract. In normal speech production, the air that is driven up from the lungs is passed through the glottis and vocal tract narrowing resulting in periodic or aperiodic (noise) excitation.

Parts of the mouth's anatomy, such as the jaw, tongue, lips, velum (soft palate) and nasal cavities, act as resonant cavities. These cavities modify the excitation spectrum that is emitted as vibrating sounds. Vowel sounds are produces with an open vocal tract with very little audible obstruction restricting the movement of air. Consonant sounds are produced with a relatively closed vocal tract, from temporary closure or narrowing of air passageway, resulting in high audible effect on the flow of air.

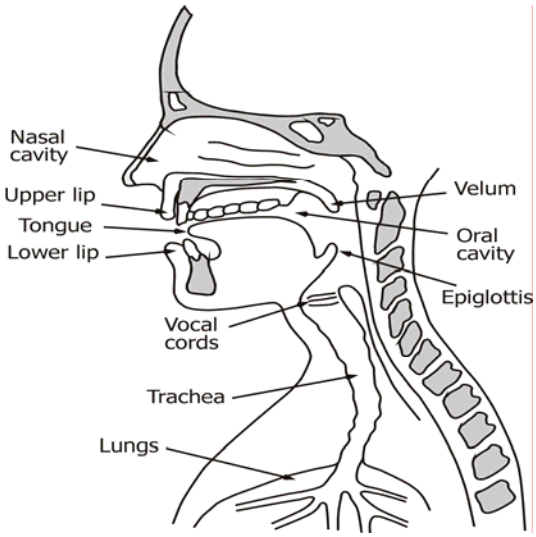


Fig1: Human generation of speech

A very basic model of speech production can be determined by approximating the individual processes of an excitation source, an acoustic filter (the vocal tract response) and the mouth characteristics during speech [1].

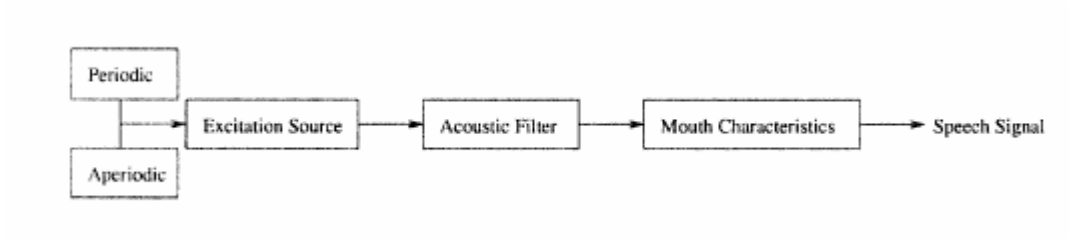


Fig2: basic model of speech production

Speech Signal

Time Domain Representation

Digital signal analysis of speech waves separates the speech into voiced (contains harmonic structure) and unvoiced speech (no harmonics structure, resembles white noise). For voiced speech, the opening and closing of the glottis results in a series of glottal pulses. This excitation possesses a periodic behavior, where each glottal opening-and-closing cycle varies in shape and time period. A string of consecutive glottal pulses, also referred to as pitch pulses, results in a quasi-periodic excitation waveform [2]

An example of speech containing the word [she] can be seen in figure below. Unvoiced segments [sh] do not display any periodic behavior, whereas, the voiced segments [e] contain an obvious periodic behavior in time domain [2].

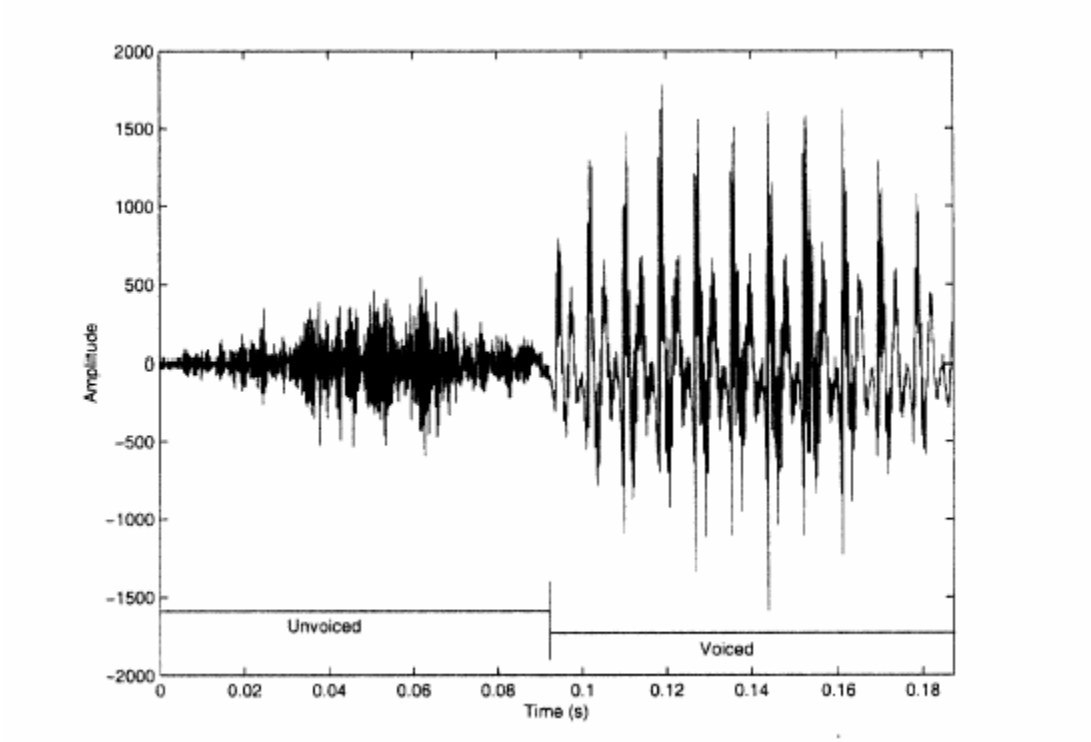


Fig3: comparison of voiced and unvoiced speech

Frequency Domain Representation

In general it is understood that the vocal tract produces speech signals containing all-pole filter characteristics [1]. In speech perception, the human ear normally acts as a filter bank and classifies incoming signals into separate frequency components. In parallel to the behavior of the human speech perception system, discrete speech signals may be analyzed in its frequency domain, where they are transformed into sinusoidal waves located at different frequencies simultaneously.

Technically, the human ear is capable of hearing signals ranging from 16 Hz to 18 kHz, depending on its amplitude. However it is known to be most sensitive for frequencies in the range of 1-5 kHz [3], hence distortion in the high frequency bandwidths is less noticeable to the human hear than distortion of equal amplitude in the low frequency areas. Due to this characteristic, the performance of the CELP coder will not be solely based on the Mean Squared Error (MSE). A 'Perceptual MSE' criterion will be included to evaluate the performance of the coder based on the intelligibility and quality of the reconstructed signal.

Codebook Excited Linear Prediction

Codebook Excited Linear Prediction (CELP) is one of the most widely used class of speech coders which is based on the concept of LPC. The enhancement is that a codebook of different excitation signals is maintained on the encoder and decoder. The encoder finds the most suitable excitation signal sends its index to the decoder which then uses it to reproduce the signal. Hence the name Codebook Excited is given to this coder.

There are many variants of CELP that are in use in various applications. Low Delay CELP (LD-CELP) and Algebraic CELP (ACELP) are generally used in internet voice calls and cell phones. ITU-T has also standardized numerous coders. Main difference in these variants is the generation of excitation signal that is used in the decoder along with other information to reconstruct the speech signal. Other differences include pre-processing and post-processing filters that shape the original and reconstructed signals for a better performance perceptually. These variants work on different bit-rates.

In our project we have designed and implemented a basic CELP coder which takes in speech signals at 8 kHz and outputs the coded speech at 6.7kbps. The output bit-rate can be adjusted according to the channel requirements and the quality of speech desired. The code can also be applied to speech signals sampled at higher rates than 8kHz e.g. 16kHz but that needs a little modification in the code we have written e.g. now the time period has changed and as explained later the sample delay for say 50Hz frequency component changes from 160 samples to 320 samples and so the frame lengths need to be adjusted accordingly.

A block diagram of CELP analysis-by-synthesis coder is shown in the figure. It is called analysis by synthesis because we encode and then decode the speech at the encoder and then find the parameters that minimize the energy of the error signal. First LP analysis is used to estimate the vocal system impulse response in each frame. Then the synthesized speech is generated at the encoder by exciting the vocal system filter. The difference between the synthetic speech and the original speech signal constitutes an error signal, which is spectrally weighted to emphasize perceptual important frequencies and then minimized by optimizing the excitation signal. Optimal excitation sequences are computed over four blocks within the frame duration, meaning that the excitation is updated more frequently than the vocal system filter. In our implementation frame duration of 20ms is used for the vocal-tract analysis (160 samples of an 8 kHz sampling rate) and 5ms block duration (40 samples) for determining the excitation.

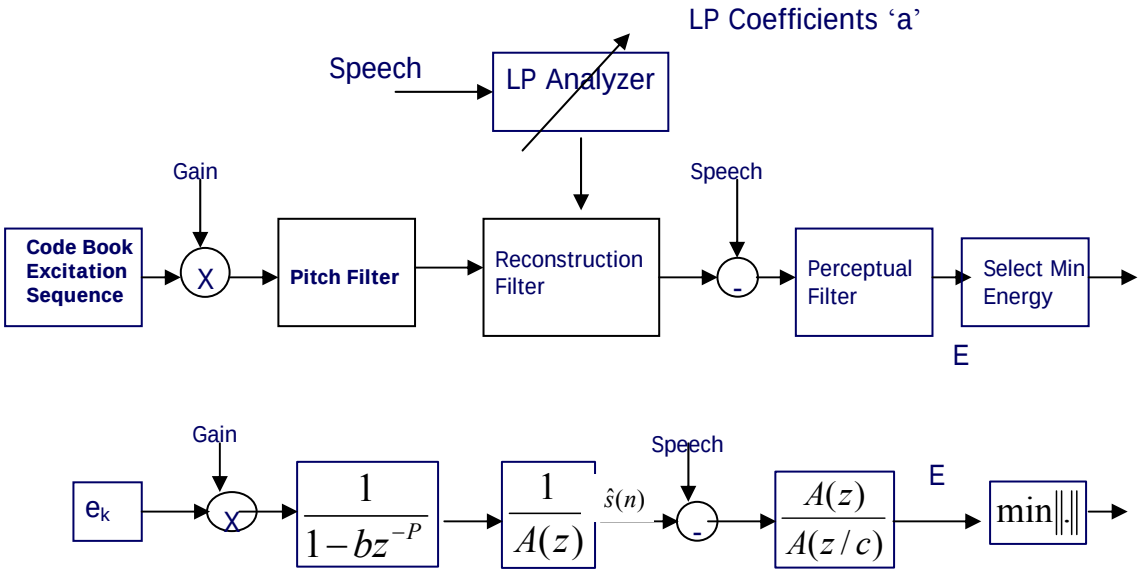


Fig. 4 Block Diagrams of CELP

Required Parameters

Looking at the encoder diagram we see that we need to transmit five pieces of information to the decoder for proper functioning.
The Liner Prediction Coefficients 'a'
The Gain 'G'
The Pitch Filter Coefficient 'b'
The Pitch Delay 'P'
The Codebook Index 'k'

Following is an explanation of all the blocks and how we find these parameters.

LP Analysis

The linear prediction analysis estimates the all-pole (vocal-tract) filter in each frame, used to generate the spectral envelope of the speech signal. The filter typically has 10-12 coefficients. In our implementation it has 10 coefficients. We have used MATLAB's lpc function to obtain these coefficients however they can be obtained by implementing a lattice filter which acts both as a forward and backward error prediction filter. It gives us reflection coefficients which can be converted to filter coefficients. Levinson-Durbin method can be used effectively to reduce complexity of the filter.

$$\hat{y}(n) = \sum_{i=1}^{p} a_i y(n-i)$$

So we define H(z) as the IIR reconstruction filter used to reproduce speech.

$$H(z) = \frac{1}{1 + \sum_{i=1}^{P} a_i z^{-i}} = \frac{1}{A(z)}$$

Perceptual Weighting Filter

The output of the LP filter is the synthetic speech frame, which is subtracted from the original speech frame to form the error signal. The error sequence is passed through a perceptual error weighting filter with system function

$$w(n) = \frac{A(z)}{A(z/c)} = c^M \frac{(p_0 - z)(p_1 - z)....(p_{M-1} - z)}{(cp_0 - z)(cp_1 - z)....(cp_{M-1} - z)}$$

where c is a parameter in the range $0 < c < 1$ that is used to control the noise spectrum weighting. In practice, the range $0.7 < c < 0.9$ has proved effective. In our implementation we use $c = 0.8$. The coefficients of the filter A(z/c) are $a_i c^i$ which can be seen from

$$\begin{aligned} A(z/c) &= 1 - a_1 (z/c)^{-1} - .... - a_M (z/c)^{-M} \\ &= 1 - (a_1 c) z^{-1} - .... - (a_M c^M) z^{-M} \end{aligned}$$

Excitation sequence

The codebook contains a number of Gaussian signals which are used as the excitation signals for the filer. In our implementation we generated a codebook of 512 sequences each of length 5ms i.e. 40 samples. The codebook is known to the encoder as well as the decoder.

The signal e(n) used to excite the LP synthesis filter 1/A(z) is determined every 5 milliseconds within the frame under analysis. An excitation sequence $d_k(n)$ is selected from a Gaussian codebook of stored sequenced, where k is the index. If the sampling frequency is 8 kHz and the excitation selection is performed every 5ms, then the codebook word size is 40 samples. A codebook of 512 sequences has been found to be sufficiently large to yield good-quality speech, and requires 9 bits to send the index.

In literature we find different types of codebooks. Most widely used are adaptive codebooks in conjunction with the fixed codebook. However we stick to fixed codebook which is a collection of Gaussian signals.

Pitch Filter

Human voices have pitch in a few hundred hertz. In our implementation we consider pitch frequencies from 50Hz to 500Hz. For 8 kHz signal these frequencies correspond to pitch delay of 16 to 160 samples. For voiced speech, the excitation sequence shows a significant correlation from one pitch period to the next. Therefore, a long-delay correlation filter is used to generate the pitch periodicity in voiced speech. This typically has the form given by

$$J(z) = \frac{1}{1 - bz^{-P}}$$

Where $0 < b < 1.4$ and P is an estimate of the number of samples in the pitch period which lies in the interval [16,160].

Energy Minimization

Thus, the excitation sequence e(n) is modeled as a sum of a Gaussian codebook sequence $d_k(n)$ and a sequence from an interval of past excitation, that is

$$e(n) = Gd_k(n) + be(n - P)$$

The excitation is applied to the vocal-tract response 1/A(z) to produce a synthetic speech sequence given by

Let

$$F(z) = \frac{1}{A(z)}$$

$$\begin{aligned} \hat{s}(n) &= e(n) * f(n) \\ &= Gd_k(n) * f(n) + be(n - P) * f(n) \end{aligned}$$

Where the parameters G, k, b, and P are selected to minimize the energy of the perceptually weighted error between the speech s(n) and the synthetic speech over small block of time i.e.

$$E(n) = w(n) * (s(n) - \hat{s}(n))$$

Let
$$I(z) = F(z)W(z)$$

then the error signal can be written as

$$\begin{aligned} E(n) &= w(n) * s(n) - Gd_k(n) * I(n) - be(n-P) * I(n) \\ &= E_0(n) - GE_1(n,k) - bE_2(n,P) \end{aligned}$$

Where
$$E_0(n) = w(n) * s(n)$$
$$E_1(n,k) = dk(n) * I(n)$$
$$E_2(n,P) = e(n-P) * I(n)$$

Note here that since P can be greater than subframe length of 40 samples, we need to buffer previous values of e(n) to use at this point.

To simplify the optimization process, the minimization of the energy of error is performed in two steps.

First, b and P are determined to minimize the error energy

$$Y_2(P,b) = \sum_n [E_0(n) - bE_2(n,P)]^2$$

Thus, for a given value to P, the optimum value of b is given by differentiating the equation with respect to b and putting equal to zero. We get the result

$$\hat{b}(P) = \frac{\sum_n E_0(n)E_2(n,P)}{\sum_n E_2^2(n,P)}$$

Which can be substituted for b in the equation for $Y_2(P,b)$ that is

$$Y_2(P,\hat{b}) = \sum_n E_0^2(n) - \frac{[\sum_n E_0(n)E_2(n,P)]^2}{\sum_n E_2^2(n,P)}$$

Hence the value of P minimizes $Y_2(P)$ or, equivalently, maximizes the second term in the above equation. The optimization of P is performed by exhaustive search, which could be restricted to a small range around the initial value obtained from the LP analysis.

Once these two parameters are determined, the optimum choices of gain G and codebook index k are made based on the minimization of the error energy between

$E_3(n) = E_0(n) - \hat{b}E_2(n,\hat{P})$ and $GE_1(n,k)$.

Thus P and k are chosen by an exhaustive search of the Gaussian codebook to minimize

$$Y_1(k,G) = \sum_n [E_3(n) - GE_1(n,k)]^2 ,$$

which is solved in a similar manner as above. Note, that the output of the filters because of the memory hangover (i.e. the output as a result of the initial filter state, with zero input) of previous intervals must be incorporated into the estimation process. And so we need to store final conditions of the filters, the previous values of b and e(n) to be used in the later frames.

Quantization

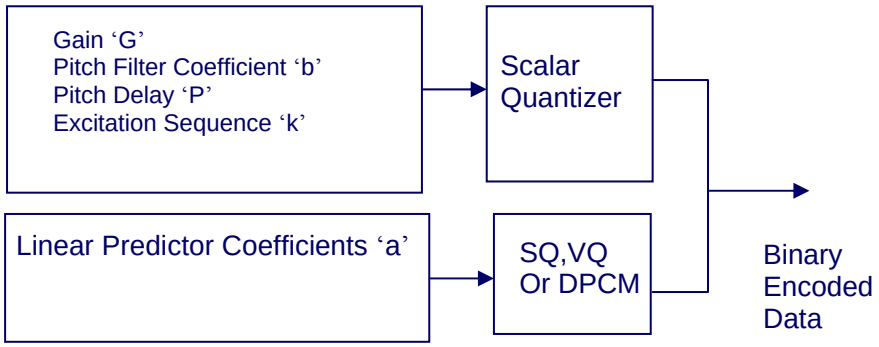


Fig. 5 Different quantization schemes for different parameters.

Though more will be said about quantization in the following chapters, we would like to mention here that in our project we use uniform scalar quantization for the four parameters i.e. Gain, Pitch Delay, Pitch Filter coefficients and Excitation Sequence Index. Linear Prediction coefficients will be quantized using different quantization schemes described later.

Now the speech signal is sampled at 8 kHz, the frame size is 20ms (160 samples), and the block duration for the excitation sequence selection is 5 ms (40 samples). Furthermore, assume that the codebook has 512 sequences which require 9 bit to send the index k, and that the lag of the pitch filter, P, is searched in the range 16 to 160 (equivalent to 50Hz to 500Hz) which require 8 bit to represent exactly. Thus, the quantization procedure only affects the parameters LPC, G, and b.

Per frame breakdown of bits per parameter

Codebook Index	'k'	9x4 bits
Pitch Delay	'P'	8x4 bits
Pitch Filter Coefficient	'b'	5x4 bits
Gain	'G'	5x4 bits
LP Coefficients	'a'	10x5 bits

Total CELP Rate = 0.98 Bit-Rate = 7.9kbps

Here LP coefficients have been allotted 5 bits per coefficient, however when using vector quantization this number can be reduced to 1.5 bits per coefficient in which case

Total CELP Rate = 0.76 Bit-Rate = 6.15kbps

According to most of the literature the LP coefficients are badly affected by quantization and the reconstructed speech signal is highly distorted. So in general these coefficients are transformed into Line Spectral Pair (LSP) coefficients which are comparably more immune to quantization. These are then retransformed into LP coefficients to at the decoder to be used in the filter.

In our implementation we have not transformed our LP coefficients into any other form and have obtained reasonable results. However we believe that such a conversion may help reduce the bit-rate or improve performance.

CELP Synthesizer

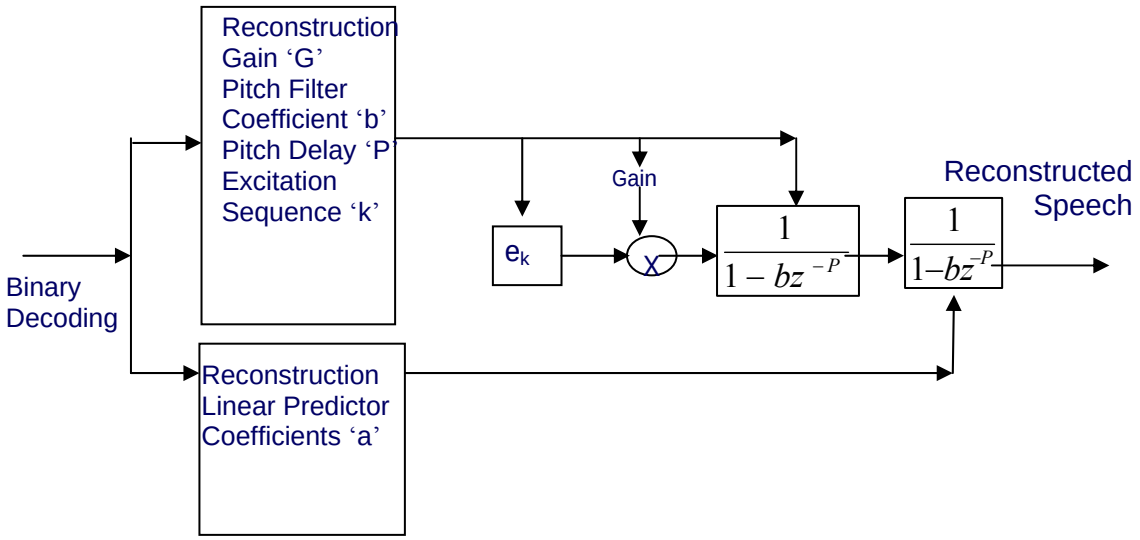


Fig. 6 Block diagram of a CELP synthesizer

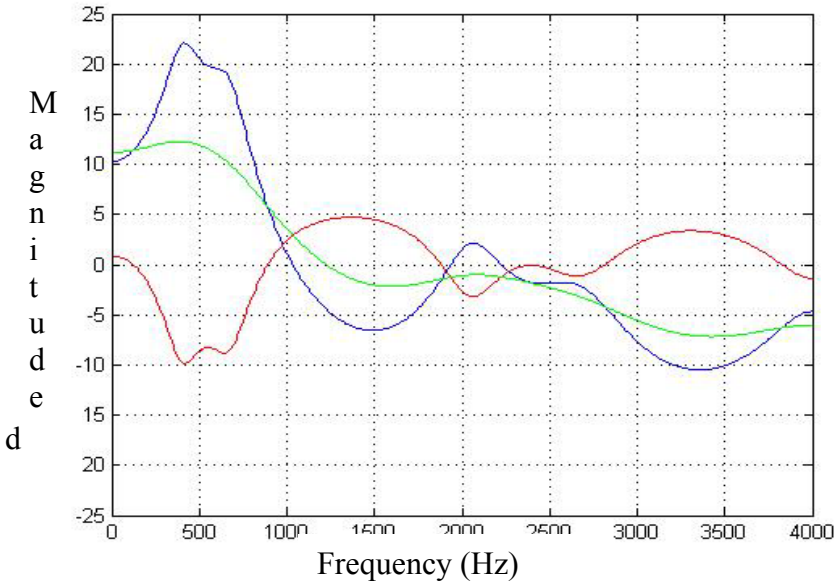
CELP synthesis is relatively simple. The quantized parameters are reconstructed. The excitation signal corresponding to 'k' is selected from the codebook. It is fed into the pitch filter after applying the gain 'G', the Pitch filter coefficient 'b' and pitch delay 'P' are used to introduce the desired correlation into the excitation signal. The resulting signal is then fed into reconstruction filter which is constructed from the filter coefficients 'a'. We get the reconstructed speech signal at the output. As will be explained later, the Mean Squared Error MSE is not a very good criterion to evaluate the quality of reconstruction. We use the concept of perceptual MSE to evaluate our coder. Mean Opinion Score is the method used in this regard.

Perceptual Filter Revisited

Perceptual filter is used to perceptually weight the error of the speech and the reconstructed speech so as to exploit the masking property of human ear. The perceptual filter has the form

$$w(n) = \frac{A(z)}{A(z/c)} = c^M \frac{(p_0 - z)(p_1 - z)....(p_{M-1} - z)}{(cp_0 - z)(cp_1 - z)....(cp_{M-1} - z)}$$

The graph shows the reconstruction filter 1/A(z) which is shown in blue. The green line shows the plot of 1/A(z/c) where c is 0.8. We note that this increases the bandwidth of the filter at the poles so the filter has a relatively smooth transition in-between frequencies. The red line shows the plot of resulting perceptual filter.



$$red = \frac{A(z)}{A(z/c)}$$

$$green = \frac{1}{A(z/c)}$$

$$blue = \frac{1}{A(z)}$$

Fig. 7 Frequency response of perceptual weighting filters

The next figure shows the original filter response and the perceptual filter response with different values of c. Here we notice that changing the value of c results in different spread of filter response which results from the changed bandwidth of filter at the pole locations. Here red line shows the perceptual filter response with c = 0.8 while green line shows the response at c = 0.4 which is then appropriately weighted to clearly see the difference in spread.

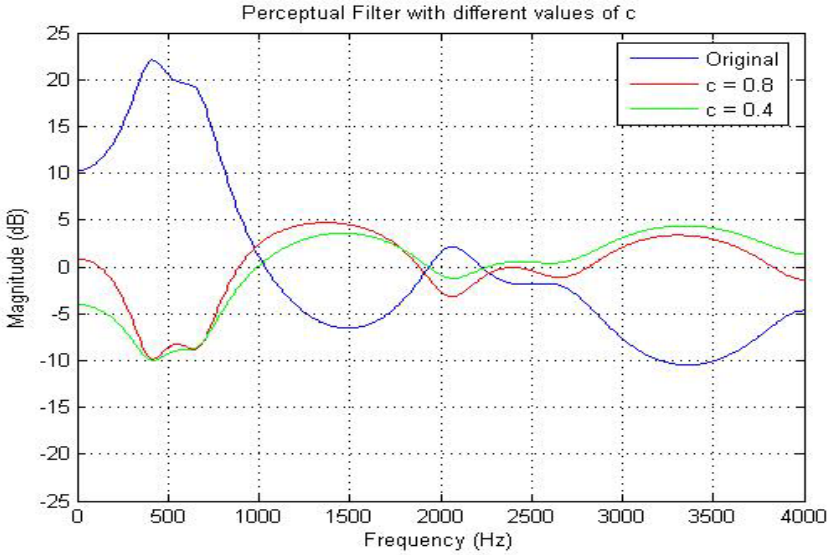
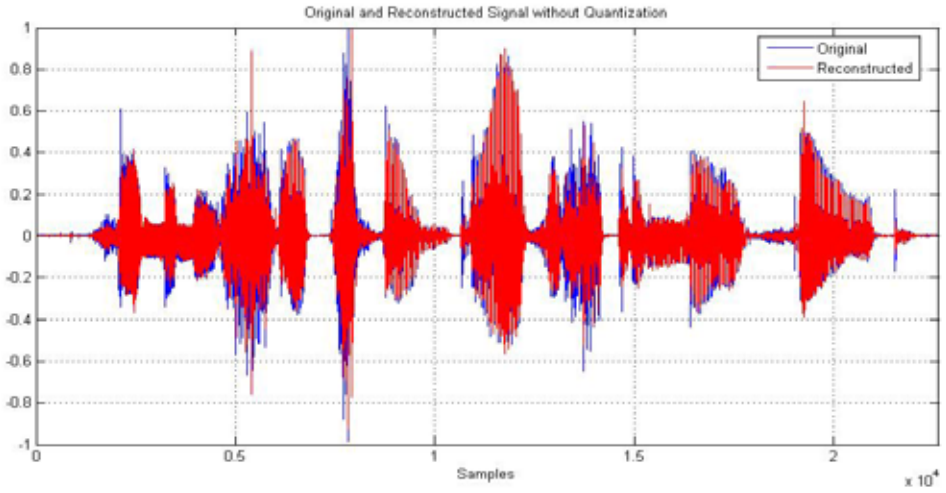


Fig. 8 Frequency response of the perceptual filter with different values of c

Speech Reconstruction

The following graphs show the reconstructed speech signals. The first graph shows the original (blue) and the reconstructed (red) speech signal when no quantization is applied except for the excitation index and pitch. We note that this is a fairly good reproduction of speech signal. Here we are more interested in the speech envelop rather than the MSE. The zoom version of this plot shows that this is a good reconstruction.



Original Speech Unquantized

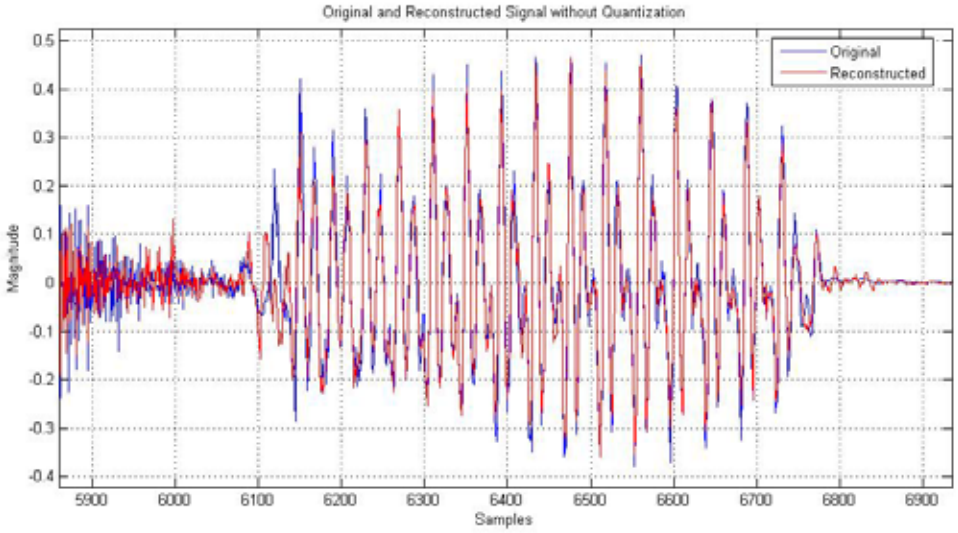


Fig. 9 Waveform of the original and reconstructed speech signals. Only excitation index and pitch are quantized.

The following graphs also show the plots of reconstructed and original speech signal. The difference is that all the parameters except LP coefficients have been quantized according to the bit allocation described above. We note that though the blue component is very much visible it is still a fine reconstruction as envelop of the speech is preserved. We also note that in the zoom picture, the reconstruction looks better in voiced region as compared to unvoiced region in terms of sample by sample comparison. However a qualitative comparison shows that the energy, pitch

and the gain of the signal in those regions is very close to the original signal hence the resulting sound is reasonably good. We can definitely notice degradation in the sound. One solution can be to increase the number of bits to represent the parameters G and b. Another approach can be to use non-uniform scalar quantizers. As a final approach we can use vector quantization to encode these two parameters together. However we have not tried any of these approaches but believe that they may result in better quantization and better reconstruction. We also believe that a post processing low pass filter may also improve the sound quality.

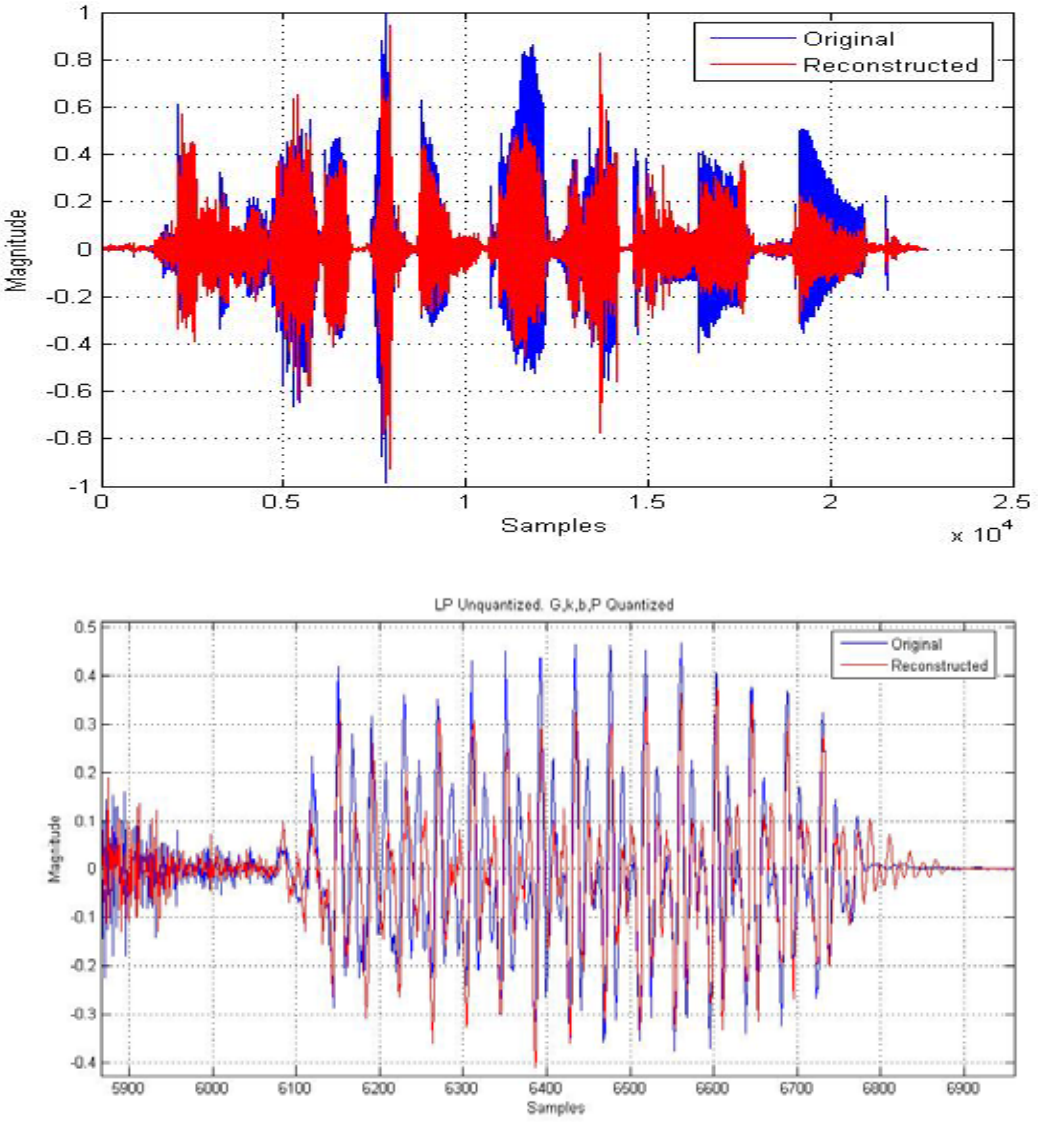


Fig. 10 Original and reconstructed waveforms. Only LP coefficients are unquantized.

Original Speech

Quantized Speech

Quantization

A large number of quantization methods have been developed throughout the years, in this project we investigated the performance of the CELP coder by using

1) Scalar Quantization
2) Differential Pulse Code Modulation
3) Vector Quantization
4) Tree-Structured Vector Quantization

Scalar Quantization

The theory of fixed-rate Scalar Quantization (SQ) has been discussed in detail in class, so we will not repeat them here. We just mention that the OPTA function predicted by Zador's formula is $\sigma^2 \beta_G m_1^* \mathrm{M}^{-2}$, where σ^2 is the source variance, β_G is the Zador factor, m_1^* is the best normalized moment of inertia of a one-dimensional cell, which is 1/12, and M is the size of the quantizers ranging from 2 to 256. For a first-order AR Gaussian source, the Zador factor is $\beta_G = 2\pi 3\sqrt{3} = 6\pi\sqrt{3}$.

To design the codebook, we used the LLOYDS command in MATLAB Communications Toolbox. The initial guess of the codebook is a uniform one with the largest codepoint being the largest value in the training data, and the smallest being the smallest value in the training data.

Differential Pulse Code Modulation

The theory of Differential Pulse Code Modulation (DPCM) has also been discussed in detail during the lectures, so we will skip most of the detail here. We just mention that the OPTA function DPCM is the same as that of SQ except a scaling factor called prediction gain G_N.

From the high-resolution analysis in the lecture notes, we optimize the scalar quantizer inside the DPCM for the random variable $\mathrm{V_i} = \mathrm{X_i} - \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{a_j X_{i-j}}$, where $\mathrm{X_i}$'s are the original training data, $\mathrm{a_j}$'s are the N-th order linear predictor coefficients. We then used the modified training data as input to the LLOYDS command in MATLAB. The initial codebook is still a uniform one, obtained by similar procedures mentioned in designing SQ in the above section.

To design the linear predictor, we need a source model. By assuming the source is AR with order from one to three, we solve different order of Yule-Walker equations as shown below, and got $a_1 = 0.8018$, $\mathbf{a}_2 = [0.7901\ 0.0146]^T$, $\mathbf{a}_3 = [0.7899\ 0.0075\ 0.0089]^T$.

$$R_x[0]\cdot a_1 = R_x[1],\ \begin{bmatrix} R_x[0] & R_x[1] \\ R_x[1] & R_x[0] \end{bmatrix}\mathbf{a}_2 = \begin{bmatrix} R_x[1] \\ R_x[2] \end{bmatrix},\ \begin{bmatrix} R_x[0] & R_x[1] & R_x[2] \\ R_x[1] & R_x[0] & R_x[1] \\ R_x[2] & R_x[1] & R_x[0] \end{bmatrix}\mathbf{a}_3 = \begin{bmatrix} R_x[1] \\ R_x[2] \\ R_x[3] \end{bmatrix}$$

We can see that the source has a high first-order correlation, and very low higher-order correlation. Thus, though no theory indicates that the first-order AR assumption is correct, we still design the DPCM according to this simple assumption.

For a first order AR Gaussian source, the prediction gain is $G_N = 10\log_{10}\frac{1}{1-\rho^2}(\text{dB})$. Here we use the estimated correlation coefficient ρ from solving the 1 by 1 Yule-Walker equation: $\rho = R_x[1]/R_x[0] = 0.8018$, where R_x[n] is the empirical autocorrelation of the test data. Thus the prediction gain G_N is 4.4719 (dB).

Implementation and performance – SQ and DPCM

In this part, we will discuss about the implementation and performance of fixed-rate scalar quantizers (SQ) and differential pulse code modulators (DPCM) for quantizing LP coefficients. We will design both kinds of quantizers with codebook sizes 2, 4, 8, 16, 32, 64, 128, and 256, which are equivalent to quantizer rate from 1 to 8. We put these two together because they are both one-dimensional quantizers. We will see that the linear predictor in DPCM does lower the distortion.

A. Training Data

15000 samples of LP coefficients generated from different speech sources are used to train the codebooks. For quantizers with size 256, every cell has 58 training points on average, and for quantizers with size 2, 7500 training points are assigned to each cell. Thus, the number of training date is enough for designing all of the quantizers mentioned above. Note that we exclude the testing data from the training data.

B. Testing Data

The testing data we used is about 140 LP coefficient vectors with length 10 generated from 3 seconds of speech. For one-dimensional quantizers used here, these data are equivalent to 1400 scalars.

C. Source Model

Speech signals can usually be modeled as first-order AR Gaussian signals. However, there is no apparent reason or intuition to believe that these modeling properties would be preserved for LP coefficients. Indeed, we can design the quantizers using training data and Lloyd's algorithm instead of using source distribution. However, a model is still needed to design the linear predictor in DPCM, and to calculate the performance predicted by Zador's formula. Thus, we still assume the LP coefficients are first-order AR Gaussian anyway. We will discuss the validity of these assumptions later.

Results—SQ and DPCM

Fig. 11 shows various predicted and actual performance. For the rate of 1, the SQ performs better than the predicted OPTA function, which shows the rule of thumb that Zador's analysis is valid only under high-rate situation.

Both SQ and DPCM increase their SNR by 4~6 dB per bit increase, which is different from the 6dB rule under high-resolution analysis. Both types of quantizers achieve the 6dB performance from rate 4 to 6, and tend to improve slower as the rate increase at other rate. Though DPCM outperforms SQ at all rates, the prediction gain is much lower than expected, as shown in fig. 12. In general, the prediction gain tends to decrease as rate increases. The difference of the OPTA

function of DPCM and its actual performance is larger than that of the OPTA function of SQ and its actual performance.

The difference between the actual performance and the predicted results is not surprising if we realized that many assumptions which are not apparently valid have been made. The first-order AR assumption seems reasonable because of the correlation analysis mentioned above. The most questionable assumption is the Gaussian one. If we can analyze the relation between the speech signals and their corresponding LP coefficients, we might get a better model for these coefficients. However, that kind of analysis is usually complex and difficult. Another potential problem is the high-resolution analysis in DPCM. This is also a complex problem which has no clear solution.

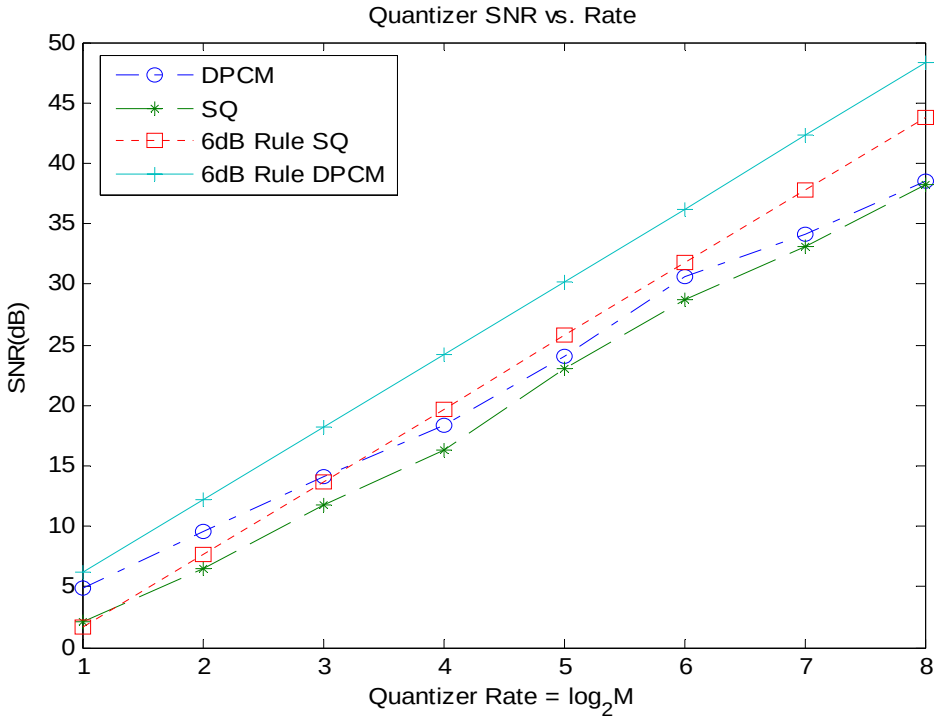


Fig. 11 Performance predicted by Zador's formula and by experimentation.

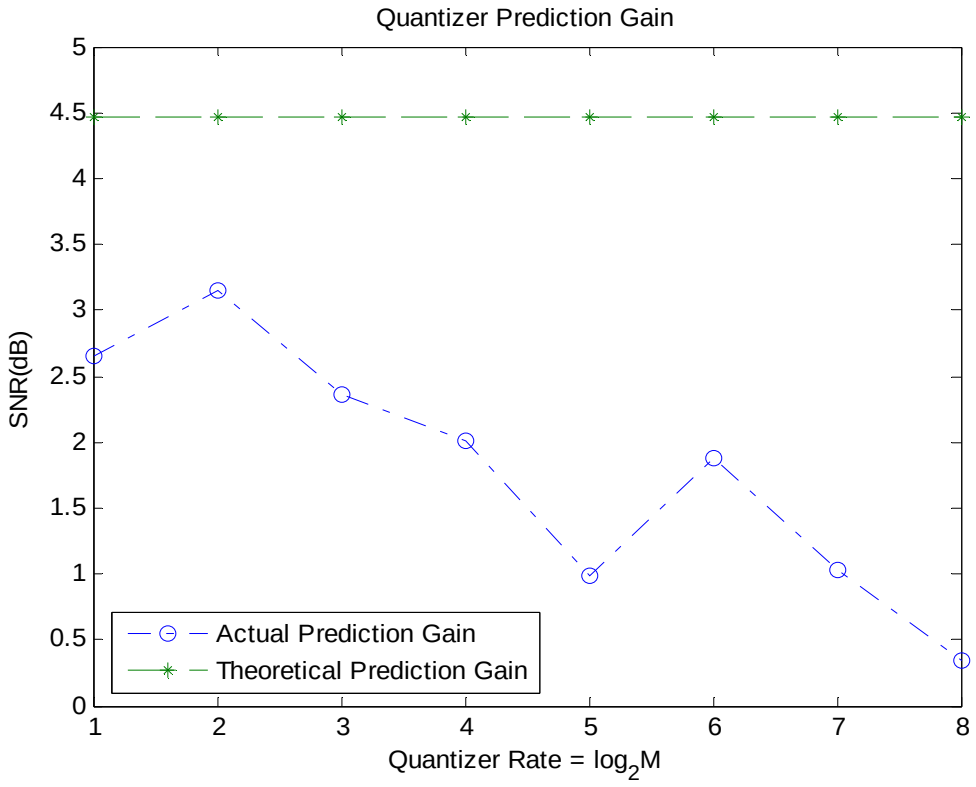


Fig. 12 The prediction gain of DPCM over SQ at different rates.

Vector Quantization

The basic theory for this method of quantization was first introduced by former University of Michigan graduate Claude Shannon, and further developed as a theory of block source coding, with regards to the rate distortion theory. Prominent use of this theory was achieved when Linde, Buzo and Gray first introduced their vector quantization algorithm (LBG algorithm). In this project, the LBG Algorithm was implemented.

Vector quantization (VQ), also known as the block or pattern matching quantization, is a process executed when a set of signal values are quantized jointly as a single vector. It considers a number of samples as a block or vector and represents them for transmission as a single code. VQ offers a significant improvement in data compression algorithms where it minimizes further the data storage required with respect to the methods used in SQ. The disadvantage of this quantization method is that there is a significant increase in computational complexity during the analysis phase or training process. Data base memory would also increase with the introduction of a larger size codebook. Despite its disadvantages, VQ remains a popular method of quantization due to its improvements in encoding accuracy and transmission bit-rate.

VQ encoder maps a sequence of feature vectors to a digital symbol. These symbols indicate the identity of the closest vector to the input vector from the values obtained from a pre-calculated VQ codebook. They are then transmitted as lower bit-rate representations of input vectors. The decoder process uses the transmited symbols as indexes into another copy of the codebook. Synthetic signals can then be calculated from the VQ symbols.

Codebook Computation

The selection criterion of the codebook is the most defining part in designing an effective VQ coder. In determining the codebook, vectors are trained to best represent the data samples, which are specifically designated for the VQ training procedure. The codebook computation procedure involves allocating a collection of vectors into what is referred to as centroids. These centroids represent the signal source and are designed to minimize the quantization distortion across the synthesized signal.

The VQ design procedure can be defined as follows. Given a vector source with its statistical properties known, given a distortion measure, and given the number of code vectors, a code book and a partition which result in the smallest average distortion must be computed

Consider a training sequence consisting of ***M*** source vectors: $T = \{x_1, x_2, \ldots\ldots, x_M\}$

In this project, the training sequence was obtained from a collection of different speech samples simulating different voice and noise characteristics. ***M*** is assumed to be sufficiently large so that all the statistical properties of the source are captured by the training sequence. We assume that the source vectors are ***k***-dimensional

$$\mathbf{x}_m = (x_{m,1}, x_{m,2}, \ldots, x_{m,k}), \quad m = 1, 2, \ldots, M.$$

If ***N*** be the number of codevectors, then $C = \{c_1, c_2, \ldots\ldots, c_N\}$ represents the codebook. Each codevector is ***k***-dimensional, e.g.,

$$\mathbf{c}_n = (c_{n,1}, c_{n,2}, \ldots, c_{n,k}), \quad n = 1, 2, \ldots, N.$$

Let S_n be the encoding region associated with codevector $\boldsymbol{c}_n$ and let $\mathbf{S}_n$

$$\mathcal{P} = \{S_1, S_2, \ldots, S_N\},$$

denote the partition of the space. If the source vector $\mathbf{x}_m$ is in the encoding region , then its approximation is:

$$Q(\mathbf{x}_m) = \mathbf{c}_n, \quad \text{if } \mathbf{x}_m \in S_n.$$

Then the average distortion is given by:

$$D_{ave} = \frac{1}{Mk} \sum_{m=1}^{M} ||\mathbf{x}_m - Q(\mathbf{x}_m)||^2,$$

Optimality Criteria

If C and P are a solution to the above minimization problem, then the following two criteria must be satisfied:

- **Nearest Neighbor Condition:**

$$S_n = \{\mathbf{x} : ||\mathbf{x} - \mathbf{c}_n||^2 \leq ||\mathbf{x} - \mathbf{c}_{n'}||^2 \ \ \forall n' = 1, 2, \ldots, N\}$$

This condition says that the encoding region $\mathbf{S}_n$ should consists of all vectors that are closer to $\mathbf{c}_n$ than any of the other codevectors.

- **Centroid Condition:**

$$\mathbf{c}_n = \frac{\sum_{\mathbf{x}_m \in S_n} \mathbf{x}_m}{\sum_{\mathbf{x}_m \in S_n} 1} \quad n = 1, 2, \ldots, N$$

This condition says that the codevector $\mathbf{c}_n$ should be the average of all those training vectors that are in encoding region $\mathbf{S}_n$. In implementation, one should ensure that at least one training vector belongs to each encoding region (so that the denominator in the above equation is never 0).

LBG Algorithm (obtained from www.data-compression.com)

The LBG VQ design algorithm is an iterative algorithm which alternatively solves the above two optimality criteria. The algorithm requires an initial codebook $\boldsymbol{C}^{(0)}$. This initial codebook is obtained by the splitting method. In this method, an initial codevector is set as the average of the entire training sequence. This codevector is then split into two. The iterative algorithm is run with these two vectors as the initial codebook. The final two codevectors are split into four and the process is repeated until the desired number of codevectors is obtained. The algorithm is summarized below.

1. Given T. Let ε > 0to be a ``small'' number.
2. Let N = 1and

$$\mathbf{c}_1^* = \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m.$$

Calculate

$$D_{ave}^* = \frac{1}{Mk} \sum_{m=1}^{M} ||\mathbf{x}_m - \mathbf{c}_1^*||^2.$$

3. **Splitting:** For i= 1, 2, 3……N, set

$$\begin{aligned} \mathbf{c}_i^{(0)} &= (1+\epsilon)\mathbf{c}_i^*, \\ \mathbf{c}_{N+i}^{(0)} &= (1-\epsilon)\mathbf{c}_i^*. \end{aligned}$$

Set. N = 2N

4. **Iteration:** Let $\boldsymbol{D}_{avg}^{(0)} =$ $\boldsymbol{D}^*_{avg}$. Set the iteration index i=0

5. For m = 1,2,3,….,M, find the minimum value of

$$||[\mathbf{x}_m - \mathbf{c}_n^{(i)}||^2,$$

over all n= 1, 2, 3……N. Let *n** be the index which achieves the minimum. Set

$$Q(\mathbf{x}_m) = \mathbf{c}_{n^*}^{(i)}.$$

i. For n= 1, 2, 3……N, update the codevector

$$\mathbf{c}_n^{(i+1)} = \frac{\sum_{Q(\mathbf{x}_m)=\mathbf{c}_n^{(i)}} \mathbf{x}_m}{\sum_{Q(\mathbf{x}_m)=\mathbf{c}_n^{(i)}} 1}$$

ii. Set i= i+1

iii. Calculate

$$D_{ave}^{(i)} = \frac{1}{Mk} \sum_{m=1}^{M} ||\mathbf{x}_m - Q(\mathbf{x}_m)||^2.$$

iv. If $(D_{ave}^{(i-1)} - D_{ave}^{(i)})/D_{ave}^{(i-1)} > \epsilon$, go back to Step (i).

v. Set $D_{ave}^{*} = D_{ave}^{(i)}$. For $n = 1, 2, \ldots, N$,

set

$$\mathbf{c}_n^{*} = \mathbf{c}_n^{(i)}$$

as the final codevectors.

Repeat Steps 3 and 4 until the desired number of codevectors is obtained

Results

Our simulation of the VQ did not give us expected results. We have not been able to explain this beyond the presence of some errors in our implementation. Hence we have not included any VQ results.

Tree-Structured Vector Quantization

A technique for reducing the search complexity in Vector Quantization is to use Tree-Structured Vector Quantization (TSVQ). In TSVQ, the search is performed in stages. In each stage, a subset of the candidate code vectors is removed from consideration by a relatively small number of operations.

In a m-ary tree search with a balanced tree, the input vector is compared with m test vectors at each stage. The nearest ie minimum distortion test vector determines which of the m paths through the tree to select in order to reach the next stage of testing. Thus at each stage the number of candidate code vectors is reduced to 1/m of the previous set. A m-ary tree with d stages is said to have breadth m and depth d.

Design

Step 1.

We use the training sequence TS to generate a codebook C of size 2 test vectors for the root node (stage 1) of the tree. This training sequence is then partitioned into 2 subsets TS_1, TS_2

Step 2.

For each i, a test codebook C_i of size 2 is designed using the Generalized Lloyd Algorithm applied to TS_i

Step 3.

Each training sequence is partitioned into 2 subsets TS_{ij}, which are used to design the codebooks for the next level

Step 4.

We continue until level d-1 has been reached.

The test vectors at this level constitute the codebook.

Results--TSVQ

Training Data

18000 training sequences of LP coefficients were used to generate the codebook. As before, the testing data was not incorporated into the training sequence.

Testing Data

The testing data we used is about 140 LP coefficient vectors with length 10 generated from 3 seconds of speech.

Source Model

Again, as before we consider the LP coefficients to be modeled as 1st order AR Gaussian. This was required to calculate the performance predicted by Zador's formula.

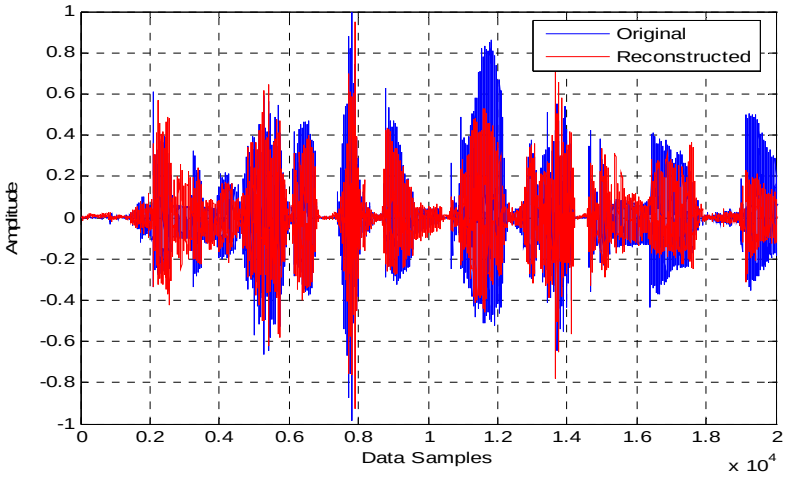
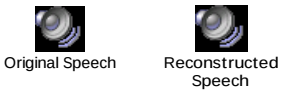


Fig 13. Original and Reconstructed Speech for TSVQ with Rate = 1.2



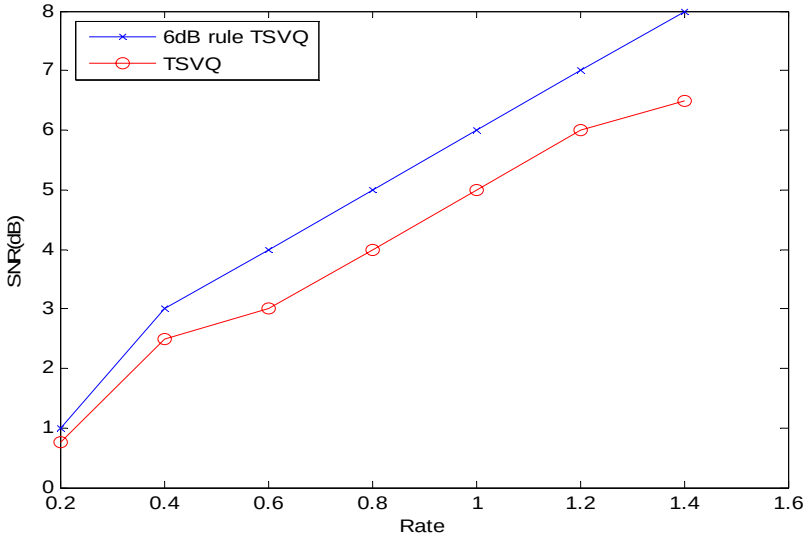


Fig. 14 TSVQ performance by theory and by experimentation

Summary

In this project, the structure of CELP has been studied and analyzed. A basic CELP coder has been designed and implemented in MATLAB. Various quantization schemes have been used to quantize the LP coefficients. The design of these quantizers has been discussed, and their results are shown.

We compared the performance of both one-dimensional quantizers, SQ and DPCM, under the same rate. We have also discussed the validity of the source model we have assumed.

Appendix

Sound files for various reconstructed speeches

Original speech 

Reconstructed Speech using SQ and DPCM to quantize LP coefficients

	SQ	DPCM
Quantizer Rate = 1, Overall CELP Bit Rate = 6.7kbps		
Quantizer Rate = 2, Overall CELP Bit Rate = 7.2kbps		
Quantizer Rate = 3, Overall CELP Bit Rate = 6.7kbps		
Quantizer Rate = 4, Overall CELP Bit Rate = 8.2kbps		
Quantizer Rate = 5, Overall CELP Bit Rate = 8.7kbps		
Quantizer Rate = 6, Overall CELP Bit Rate = 9.2kbps		
Quantizer Rate = 7, Overall CELP Bit Rate = 9.7kbps		
Quantizer Rate = 8, Overall CELP Bit Rate = 10.2kbps		

Reconstructed Speech using TSVQ to quantize LP coefficients, dimension = 10

Quantizer Rate = 0.4, Overall CELP Bit Rate = 6.4kbps 

Quantizer Rate = 0.8, Overall CELP Bit Rate = 6.6kbps 

Quantizer Rate = 1.2, Overall CELP Bit Rate = 6.8kbps 

References

[1] Atal, B. S., Cuperman, V., Gersho, A., *Advances in Speech Coding*, Kluwer Academic Publishers, Boston, 1991

[2] Barnwell III, T. P., Nayebi, K., Richardson, C. H., *Speech Coding: A Computer Laboratory Textbook*, John Wiley and Sons Publishing, New York, 1996

[3] Klejin, W. B., Paliwal, K. K., *Speech Coding and Synthesis*, Elsevier, Amsterdam, 1995

[4] Kondoz. A. M., *Digital Speech: Coding for Low Bit Rate Communications Systems*, John Wiley and Sons Publishing, England, 1994

[5] Quackenbush, S. R., Barnwell III, T. P., Clements, M. A., *Objective Measures of Speech Quality*, Prentice-Hall, Englewood