EECS 651- Source Coding Theory

Design of a CELP Speech Coder and Study of Complexity vs Quality Trade-offs for Different Codebooks.

Suresh Kumar Devalapalli
Raghuram Rangarajan
Ramji Venkataramanan

Abstract

In this term project, we study the compression of speech signals using Code Excited Linear Prediction (CELP). We design a basic CELP coder and study its Rate vs. SNR characteristics using MSE as the distortion criterion. The computational complexity of the encoding process is analyzed. We then examine methods to reduce complexity in the encoder by using special types of codebooks viz., binary, ternary and overlapping codebooks. We compare the performance of these codebooks in terms of complexity and SNR of the reconstructed speech. It is found that significant reduction in complexity can be achieved without much degradation in performance. We then compare the different codebooks using Bark Spectral Distortion, a perceptual measure of speech quality. As a final step, we examine if variable rate coding can help reduce the rate of the CELP coder.

Contents

"The most valuable of talents is that of never using two words when one will do."

- Thomas Jefferson

1 Introduction

For long, transmission of speech has been an important application of communication systems the world over. Applications such as transmission of speech over the internet demand real-time coding of speech at low bit rates. In most speech coders, the analog speech signal is sampled at 8000 samples/sec. The aim is to represent these samples with as few bits as possible with the constraint of maintaining a required 'perceived' quality of the reconstructed speech. Thus, speech coding is clearly a lossy compression.

Over the years many speech coding techniques have been developed starting from PCM and ADPCM in the '60s, to linear prediction in the '70s and Code Excited Linear Prediction (CELP) [1] in the late 80s and 90s. Since its introduction, CELP has evolved to become the dominant paradigm for real-time speech compression. In this work, we design a basic CELP coder and study its rate vs. SNR characteristics. We then study various methods to reduce computational complexity of the coder without compromising on the quality of reconstructed speech.

1.1 The CELP concept

The basic principle that all speech coders exploit is the fact that speech signals are highly correlated waveforms. Speech can be represented using an autoregressive (AR) model:

$$X_n = \sum_{i=1}^{L} a_i X_{n-i} + e_n \ . \tag{1.1}$$

Each sample is represented as a linear combination of the previous L samples plus a white noise. The weighting coefficients $a_1, a_2, ..., a_L$ are called Linear Prediction Coefficients (LPCs). We now describe how CELP uses this model to encode speech.

The samples of the input speech are divided into blocks of N samples each, called frames. Each frame is typically 10-20 ms long (this corresponds to $N = 80 - 160$). Each frame is divided into smaller blocks, of k samples (equal to the dimension of the VQ) each, called subframes. For each frame, we choose $a_1, a_2, ..., a_L$ so that the spectrum of $\{X_1, X_2, ..., X_N\}$, generated using the above model, closely matches the spectrum of the input speech frame. This is a standard spectral estimation problem and the LPCs $a_1, a_2, ..., a_L$ can be computed using the Levinson-Durbin algorithm.

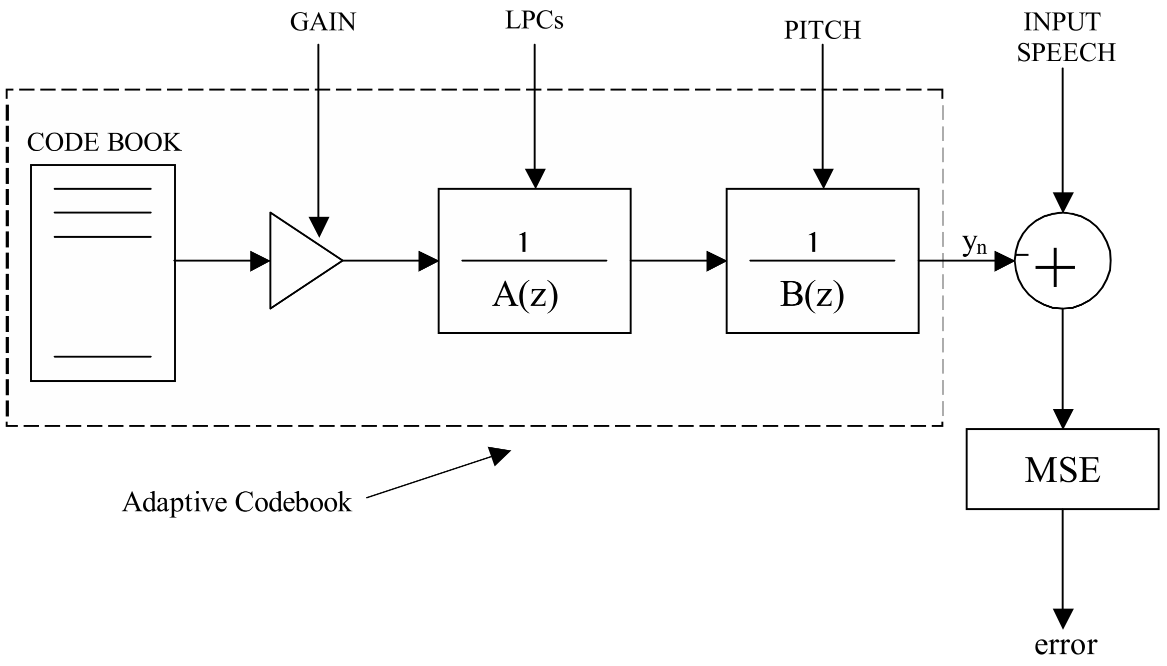


Fig.1: Basic CELP scheme, minimize error by selecting best codebook entry.

Writing Eq.(1.1) in z-domain, we obtain

$$\frac{X(z)}{E(z)} = \frac{1}{1-(a_1 z^{-1} + a_2 z^{-2} + ... + a_L z^{-L})} = \frac{1}{A(z)} \tag{1.2}$$

From Eqs.(1.1) and (1.2), we see that if we pass a 'white' sequence $e[n]$ through the filter $1/A(z)$, we can generate $X(z)$, a close reproduction of the input speech.

The block diagram of a CELP encoder is shown in Fig.1. There is a codebook of size M and dimension k, available to both the encoder and the decoder. The codevectors have components that are all independently chosen from a $N(0,1)$ distribution so that each codevector has an approximately 'white' spectrum. For each subframe of input speech (k samples), the processing is done as follows: Each of the codevectors is filtered through the two filters (labeled $1/A(z)$ and $1/B(z)$) and the output y_k is compared to the speech samples. The codevector whose output best matches the input speech (least MSE) is chosen to represent the subframe.

The first of the filters, $1/A(z)$, is described by Eq.(1.2). It shapes the 'white' spectrum of the codevector to resemble the spectrum of the input speech. Equivalently, in time-domain, the filter incorporates short-term correlations (correlation with L previous samples) in the white sequence. Besides the short-term correlations, it is known that regions of voiced speech exhibit long term periodicity. This period, known as pitch, is introduced into the synthesized spectrum by the pitch filter $1/B(z)$. The time domain behavior of this filter can be expressed as:

$$y[n] = x[n] + y[n-P] \,, \tag{1.3}$$

where $x[n]$ is the input, $y[n]$ is the output and P is the pitch.

The speech synthesized by the filtering is scaled by an appropriate gain to make the energy equal to the energy of the input speech. To summarize, for every frame of speech (N samples) we compute the LPCs and pitch and update the filters. For every subframe of speech (k samples), the codevector that produces the 'best' filtered output is chosen to represent the subframe.

The decoder receives the index of the chosen codevectors and the quantized value of gain for each subframe. The LPCs and the pitch values also have to be quantized and sent every frame for reconstructing the filters at the decoder. The speech signal is reconstructed at the decoder by passing the chosen codevectors through the filters.

An interesting interpretation of the CELP encoder is that of a forward adaptive VQ. The filters are updated every N samples and so we have a new set of codevectors y_k every frame. Thus, the dashed block in Fig.1 can be considered a forward adaptive codebook because it is 'designed' according to the current frame of speech.

In our design, we have used MSE as the criterion to choose the best codevector. However, it is the perceptual quality of the synthesized speech that we should seek to optimize. Does a lower MSE always guarantee better sounding speech? This is in general, but not always, true and so many practical CELP coders use perceptually weighted MSE as the fidelity criterion. In our design, we found MSE to be a reasonable fidelity criterion. In a later section, we examine the correlation between MSE and the perceptual quality of the synthesized speech.

Rate

The rate of the CELP coder is determined by two factors:

1. Rate of the VQ = $R_{VQ} = \dfrac{\log_2 M}{k}$
2. Overhead bits needed to send the quantized values of gain for every subframe and the LPC coefficients for each frame.

The rate of the coder in bits per second is given by

$$R = (R_{VQ} + \#\text{overhead bits/sample}) * 8000$$

In this work, we consider only the rate of the VQ in all our experiments. In practical systems, the number of bits is allocated for the VQ and the overhead values is approximately equal. So the rate of the VQ is roughly half the rate of the coder.

2 Design and Performance Analysis of a CELP Coder

We designed a basic CELP coder in MATLAB with the following parameters:

- Frame size N=80
- L = 10 (10th order AR model)
- Fidelity criterion- MSE
- Dimension k=10 or 5

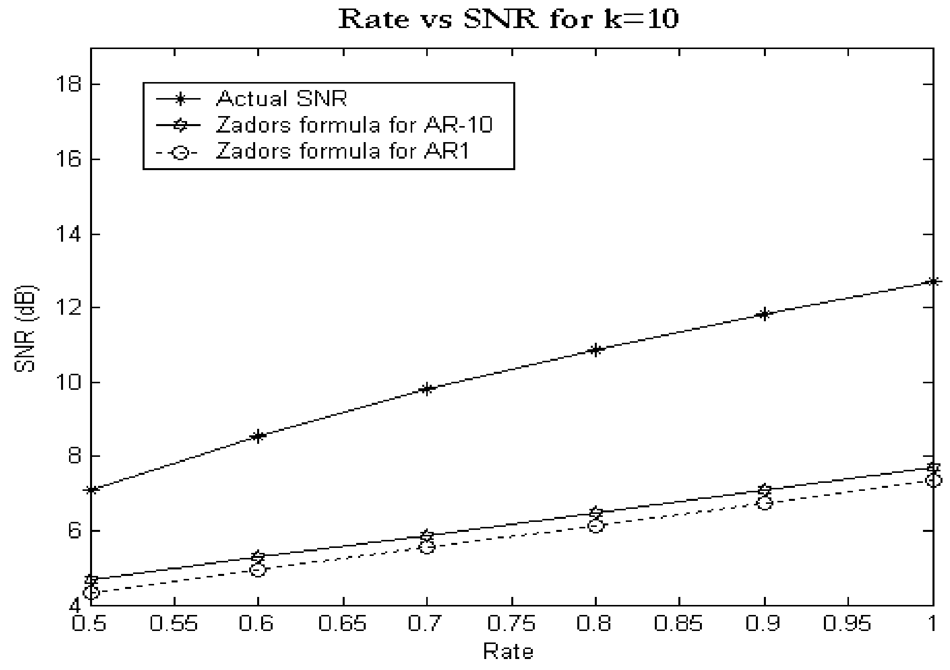
Our input speech sample for all experiments comprised of two sentences each of male and female speech. The SNR of the reconstructed speech was determined at different rates for dimensions k=5 and k=10. The SNR vs. rate curves obtained for $k = 5$ and $k = 10$ are shown in Fig.2. We ran experiments with codebooks of size M = 32, 64, 128, 256, 512 and 1024. This corresponds to a varying rate R_{VQ} from 0.5 to 1 when $k = 10$ and from 1 to 2 when $k = 5$.

The SNR's predicted by Zador's formula are also shown in the graph. The dotted line is the Zador SNR assuming an AR-1 model for speech. This was computed by estimating the AR-1 parameter (correlation coefficient ρ) from the input speech signal. The line above the dotted line represents the Zador SNR assuming a kth order AR model. The Zador factor β_k for this model is given by

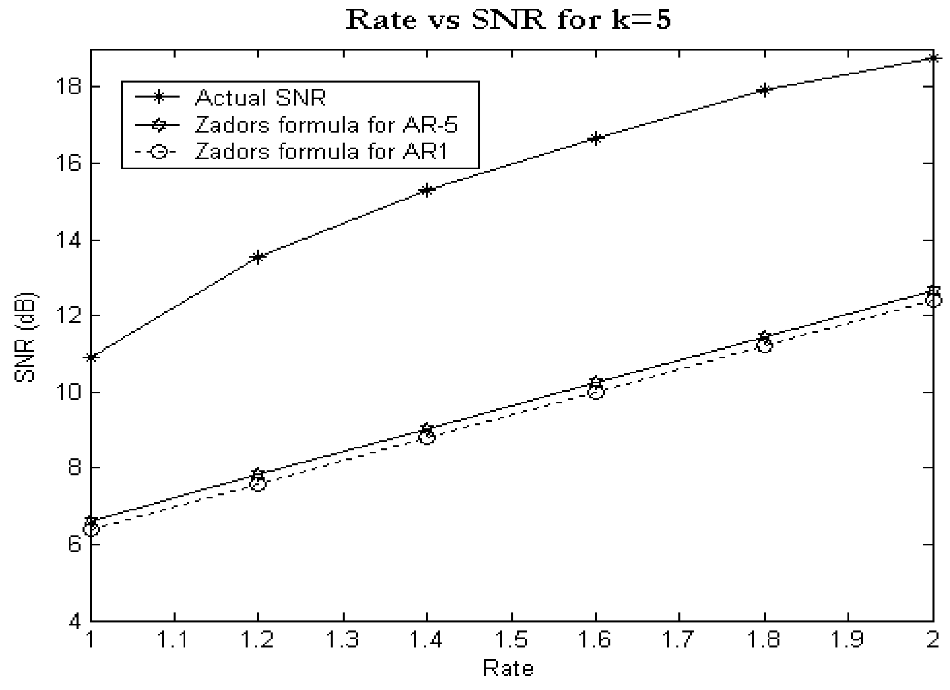
$$\beta_k = \frac{2\pi \left(\frac{k+2}{k} \right)^{\frac{k+2}{2}} |K|^{\frac{1}{k}}}{\sigma^2}$$

The k autocorrelation values required for computing the determinant were estimated from the input speech signal. While it may seem surprising that the SNR of the CELP coder is higher than the Zador SNR, it should be noted that CELP is significantly different from a normal VQ (to which we have applied Zador's formula). CELP has an adaptive codebook that changes every frame according to the input speech. The SNR predicted by Zador's formula is applicable to a 'fixed codebook' VQ for a source that is characterized by the estimated covariance matrix K. Thus, we cannot expect Zador's formula (in the form applied here) to give a reasonable estimate of the SNR. An improved estimate could be obtained by calculating the Zador SNR for every frame based on a covariance matrix that is estimated for that frame.

We also observe that for $k=10$ (lower rates), the rate vs. SNR slope is clearly more than 6 dB/bit; for $k=5$ (higher rates) the slope approaches 6dB/bit. This is consistent with our expectation that SNR increases at approximately 6dB/bit at high rates.



(a)



(b)

Fig.2 SNR vs Rate characteristic of our CELP coder (a) k=5 (b)k=10

3 Complexity of the CELP encoder

Complexity is an important consideration while designing any VQ. The CELP algorithm gives good output speech quality at low bit rates. However, this quality is obtained at the cost of very high complexity. Since CELP coders are used for real-time applications such as the transmission of speech over networks, long delays in encoding are undesirable. In this section, we analyze complexity of the encoder and derive expressions for the number of computations and storage required. We then discuss methods to reduce the complexity of the algorithm without significant degradation in performance.

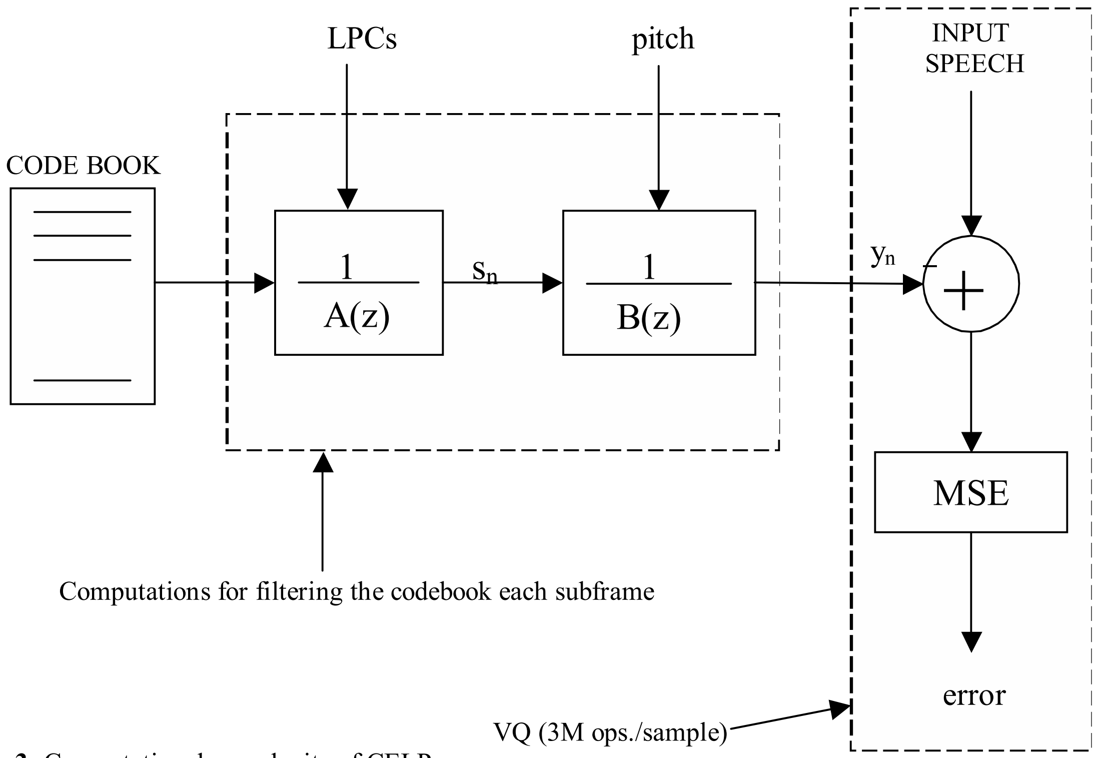


Fig. 3: Computational complexity of CELP

To analyze complexity, it is convenient to look at CELP as a forward adaptive VQ. The following analysis will serve as a baseline for comparison though the actual operations may be performed differently pin real-time applications. From Fig.3, it is clear that the encoding process involves two stages:

1. The filtering operation: Convolution of each codevector with the impulse response of the filter. This has to be done every k samples of speech.
2. Choosing the best codevector from the 'filtered' codebook. This part can be considered a VQ and hence involves 3M ops/sample.

The number of operations in Step 1 can be derived as follows:

While filtering the codevectors in a subframe, we have to take care of the zero-input response (ZIR) at the output of the filters. This is produced due to filtering that occurred in the previous subframe. At the beginning of each subframe, we can subtract the ZIR from the speech. Note that the subtraction needs to be done only once in a subframe, since the ZIR is the same for all codevectors. We can then represent the convolution of each codevector with the impulse response of the LPC filter by a matrix multiplication (i.e. as if we have zero initial state):

$$s = H\, d \quad , \tag{3.1}$$

where d is the codevector, s is the output of the LPC filter and H is the lower triangular matrix :

$$H = \begin{bmatrix} a_0 & 0 & 0 & . & . & . & 0 \\ a_1 & a_0 & 0 & . & . & . & 0 \\ . & & & & & & \\ . & & & & & & \\ a_{k-1} & a_{k-2} & . & . & . & . & a_0 \end{bmatrix}.$$

For each codevector (dimension k), the matrix multiplication requires $k(k-1)/2$ multiplications and $k(k-1)/2$ additions.

The pitch filter equation can be written as

$$y_n = s + g\, y_{n-p} \quad , \tag{3.2}$$

where g is the pitch gain, p is the pitch period and y_n is the filtered codevector. This requires $2k$ operations per codevector. Thus the total number of operations per codevector is

$$op_n / codevector = \frac{k(k-1)}{2} + \frac{k(k-1)}{2} + 2k = k(k+1)$$

Since we have M codevectors and k samples/codevector, the filtering in Step 1 will require $M(k+1)$ operations/sample. The VQ part of the CELP coder will require 3M operations/sample. Thus the overall computation for the encoding process is

$$op_n = M(k+4) \;\; ops./sample \tag{3.3}$$

We have to store the M codevectors; so we need storage corresponding to Mk floating point numbers.

To get an idea of the complexity involved, for $k = 10$, encoding requires $14M$ ops./sample. Out of this, 11M is due to the filtering part and while the VQ part requires 3M operations/sample. Clearly, we would like to reduce the complexity of filtering without sacrificing performance. The codebook that we use consists of codevectors with random numbers drawn from a $N(0,1)$

distribution. In the following subsection, we consider special types of codebooks[2], which reduce the filtering complexity and analyze the performance vs. complexity tradeoffs obtained by using them.

4 Special Codebooks for CELP

4.1 The Binary Codebook

Binary codebooks contain codevectors with only binary components. i.e. zeros and ones. Clearly, the filtering operation does not require multiplications. The filtering of a codevector involves only additions and that too, only when the component of the codevector is a one. The number of ops./sample is computed using a method similar to the previous part.

For the LPC filter, the number of multiplications is zero while the number of additions is half the number of additions in the previous case (there are 50% zeros and 50% ones in the binary codebook). The long-term filter will still involve $2k$ operations per codevector. Thus the total number of operations for the filtering part alone will require

$$op_b / codevector = \frac{k(k-1)}{2.2} + 2k$$

Since we have M codevectors and k samples/codevector, the filtering will require $M\frac{(k-1)}{4} + 2M$ operations/sample. The VQ part of the CELP coder will require 3M operations/sample. Thus the overall computation for the encoding using a binary codebook is

$$op_b = \frac{M(k+7)}{4} + 3M \; ops./sample \tag{4.1}$$

For $k = 10$, this translates to ~ $7M$ ops./sample compared to $14M$ for the normal codebook.

Interestingly, the binary codebook involves no storage when $M = 2^k$ as the index of the codevector in binary notation will be the codevector itself. However, a major disadvantage of using the binary codebook is that the size M can be at most 2^k. This is because for a given k we have only 2^k possible binary sequences. For example, for $k = 5$, M can be at most 32, which is too small a number to get a good reproduction of speech. This is discussed later when we compare the performance of different codebooks.

4.2 The Ternary Codebook

In ternary codebooks [2][3], the components of the codevectors are chosen from the set {-1,0,1}. This offers more flexibility than the binary codebook. Here again, we do not have multiplications; we have only additions and subtractions. The complexity of the filtering can be controlled by varying the percentage of zeros in the codebook. The higher the number of zeros, the lesser the number of adds or subtracts we need to perform. For our experiments, we used 50% zeros. The ternary codevectors were created by first generating a uniform random variable in [-1,1] and all values within ± 0.5 were mapped to zero, those greater than 0.5 to 1 and those less than 0.5 to -1. This ensures that the number of computations in the ternary case is exactly the same as that in the binary case. As before, the number of operations/sample is

$$op_t = \frac{M(k+7)}{4} + 3M \ . \tag{4.2}$$

The storage required corresponds to storing Mk ternary valued numbers. As we will see later, the ternary codebook performs very well- the performance is much better than the binary codebook and almost as good as the normal codebook.

4.3 The Overlapping Codebook

Overlapping codebooks [2][3] have codevectors that are a shifted version of the preceding vector plus some extra components. An example of a 6 dimensional codebook with overlap 1 would be: $w_1=(d_1,d_2,d_3,d_4,d_5,d_6)$, $w_2=(d_2,d_3,d_4,d_5,d_6,d_7)$, $w_3=(d_3,d_4,d_5,d_6,d_7,d_8)$ and so on.

An overlapping codebook of dimension k , size M and overlap 1 would be of the form

$$\begin{bmatrix} (d_1 \ d_2 \ . \ . \ . \ . \ d_k) \\ (d_2 \ d_3 \ . \ . \ . \ . \ d_{k+1}) \\ (d_3 \ d_4 \ . \ . \ . \ . \ d_{k+2}) \\ .. \\ .. \\ (d_M \ d_{M+1} \ . \ . \ d_{k+M-1}) \end{bmatrix}$$

Since the codebook has only M+k-1 distinct entries, we need to store only M+k-1 floating point numbers instead of the usual Mk numbers. We determine the complexity of filtering an overlapping codebook as follows:

Compare the output of the LPC filter corresponding to the i^th^ and (i+1)^th^ codevectors :

1. When the i^th^ codevector is filtered, the output s_i is given by:

$$s_i = \begin{bmatrix} a_0 & 0 & 0 & . & . & . & 0 \\ a_1 & a_0 & 0 & . & . & . & 0 \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ a_{k-1} & a_{k-2} & . & . & . & . & a_0 \end{bmatrix} \begin{bmatrix} d_i \\ d_{i+1} \\ . \\ . \\ . \\ d_{i+k-1} \end{bmatrix} = \begin{bmatrix} a_0 d_i \\ a_1 d_i + a_0 d_{i+1} \\ a_2 d_i + a_1 d_{i+1} + a_0 d_{i+2} \\ . \\ . \\ a_{k-1} d_i + a_{k-2} d_{i+1} + \ldots + a_0 d_{i+k-1} \end{bmatrix} \qquad (4.3)$$

2. When the i+1^th^ codevector is filtered, the output s_{i+1} is given by

$$s_{i+1} = \begin{bmatrix} a_0 & 0 & 0 & . & . & . & 0 \\ a_1 & a_0 & 0 & . & . & . & 0 \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ a_{k-1} & a_{k-2} & . & . & . & . & a_0 \end{bmatrix} \begin{bmatrix} d_{i+1} \\ d_{i+2} \\ . \\ . \\ . \\ d_{i+k} \end{bmatrix} = \begin{bmatrix} a_0 d_{i+1} \\ a_1 d_{i+1} + a_0 d_{i+2} \\ a_2 d_{i+1} + a_1 d_{i+2} + a_0 d_{i+3} \\ . \\ . \\ a_{k-1} d_{i+1} + a_{k-2} d_{i+2} + \ldots + a_0 d_{i+k} \end{bmatrix} \qquad (4.4)$$

Comparing the RHS of Eqs.(4.3) and (4.4), we make the following observation: While computing s_i, we have computed all the additive terms required to determine s_{i+1}, except for those in the last row, already computed in the previous case. If we have a temporary buffer to store all the terms involved in computing s_i, we then only need to compute the terms $a_{k-1}d_{i+1}, a_{k-2}d_{i+2}, \ldots, a_0 d_{i+k}$ and then perform the additions. Thus at the cost of slight increase in temporary storage, most of the multiplications are eliminated. For the last row, we require k multiplications and k-1 additions and for the remaining rows we require $\frac{(k-2)(k-1)}{2}$ additions. The number of operations/codevector for the filtering part is given by:

$$op_o / codevector = \left(2k - 1 + \frac{(k-1)(k-2)}{2} \right) + 2k = \frac{k(k+5)}{2}$$

Taking into account the $3M$ ops./sample for the VQ, the total computation for encoding using the overlapping codebook is given by

$$op_o = \frac{M(k+5)}{2} + 3M \;\; ops./\,sample \tag{4.5}$$

The storage and the complexity involved for each of the codebooks is summarized in Table 1.

CODEBOOK	STORAGE	COMPUTATION (ops/samples)	ops/sample for k=10
Normal (iid Gaussian)	Mk	M(k+1)+3M	14M
Overlapping	M+k-1	M(k+5)/2+3M	10.5M
Ternary	Mk ternary numbers	M(k+7)/4+3M	~7M
Binary	None	M(k+7)/4+3M	~7M

Table 1: Storage and complexity for various codebook structures.

4.4 Performance of the Special Codebooks

Having evaluated the complexity of the different codebooks, we now compare their performance. The SNR of the reconstructed speech was determined at different rates for each of the codebooks. The SNR vs. rate curves obtained for dimension $k = 10$ are shown in Fig.4.

Clearly, the performance of the binary codebook is the worst among the four. This is not surprising as the codevectors contain only ones and zeros. We recall that the codevectors are supposed to be zero-mean white sequences. The binary valued codevectors are not zero-mean and do not have a white spectrum. One possible scope for improvement would be to consider codevectors with binary components 1 and -1. Though the number of codevectors would still be limited by the dimension of the codebook, the codevectors would be zero mean and might give a better SNR than {0, 1} binary codebook..

Remarkably, the ternary codebook, which is a slight modification over the binary codebook, produces much higher SNR (the SNR curve is just a little lower than the normal codebook). Since we have used a ternary codebook with 50% zeros, the number of computations is same as in the binary case. Despite its structured nature, the overlapping codebook performs almost as well as the normal codebook. An intuitive explanation could be given to justify this. Although the

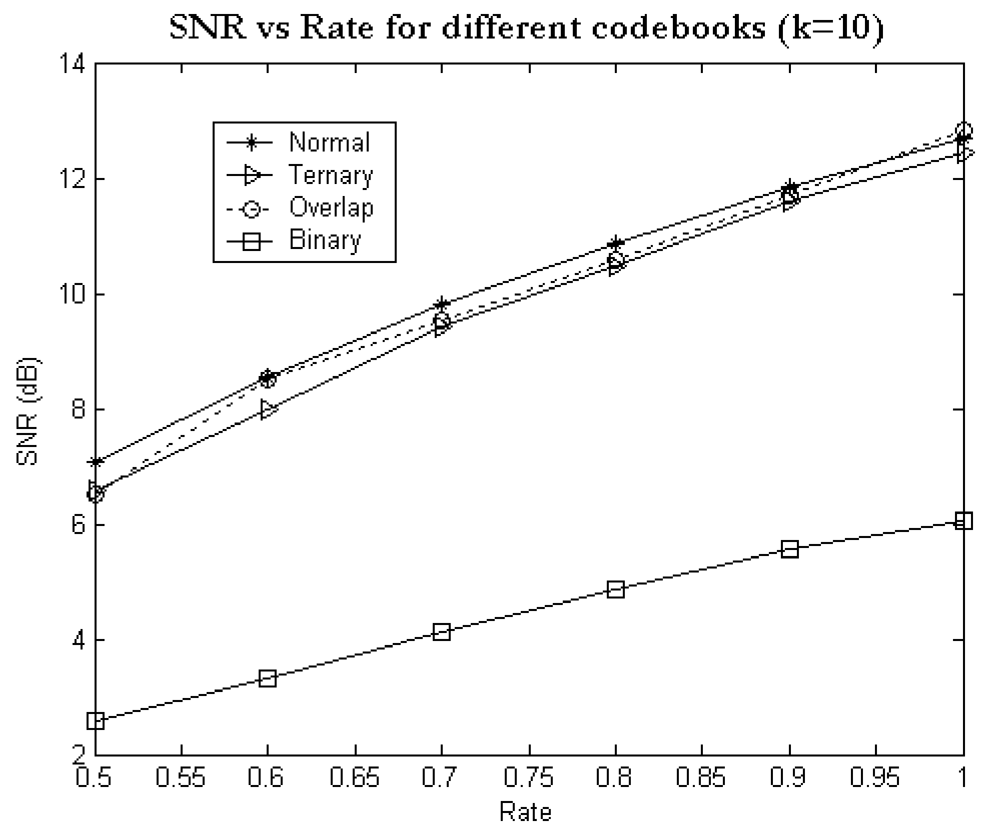


Fig.4. Comparison of different codebooks

codevectors overlap, each of them still has a 'white' spectrum since it consists of independent variables drawn from an $N(0,1)$ distribution. When these codevectors are filtered, the outputs could be considerably different. Therefore, there need not be any structure in the adaptive codebook although the codevectors have overlapping components,

To summarize, we see that the ternary and overlapping codebooks perform almost as well as the normal codebook. Both are excellent alternatives to the normal codebook that reduce the encoding complexity significantly. The overlapping codebook performs slightly better than the ternary. From Table 1, we see that this is achieved at a cost of higher computational complexity. There is a tradeoff between computational complexity and SNR and we can choose a codebook suited to the constraints imposed on the encoder.

5 Comparison of CELP coders based on perceptual quality

Upto this point, the performance of the coders was analyzed based on MSE as the distortion criterion. A few questions we may ponder about are, “How good is MSE as the distortion criterion?” “Does a coder with low SNR always sound worse than a coder with high SNR?” “How do we compare the performance of different coders?” These are questions we would like to address in this sub-section.

Mean Opinion Score (MOS)

The most widely accepted criterion for the performance of a coder is Mean Opinion Score (MOS). This is a subjective test and it maps the perceived level of distortion on to a scale of numerical values in the range 5-1. The rating scale employed in MOS testing is illustrated in the following table.

Rating	**Speech Quality**	**Level of Distortion**
5	Excellent	Imperceptible
4	Good	Just perceptible but not annoying
3	Fair	Perceptible and slightly annoying
2	Poor	Annoying but not objectionable
1	Unsatisfactory	Very annoying and objectionable

Table 2 Rating Scale of Mean Opinion Score

MOS permits the ranking of coders according to their subjective quality. But it involves lengthy listening tests which are usually costly and time consuming. So, an objective measure that predicts the subjective quality of the coder would be useful. Bark Spectral Distortion [4] is one such measure that has been shown to be highly correlated with Mean Opinion Scores.

Bark Spectral Distortion (BSD)

BSD tries to measure the subjective quality by considering the features of perceptual processing of speech sounds by the human ear. The steps involved in the computation of BSD are briefly explained below.

- ***Critical band filtering***: In this step, the speech is passed through band of filters whose center frequencies and bandwidths increase with frequency. This takes care of the changing sensitivity of the ear as the frequency varies. The human ear is more sensitive at lower frequencies than at higher frequencies.

- ***Equal loudness pre-emphasis:*** This step incorporates the fact that the ear is not equally sensitive to stimulations at different frequencies. For e.g., a 100Hz tone needs to be up to 35dB more intense than a 1000Hz tone, for the two to sound equally loud. After this step, the spectrum is loudness equalized so that the relative loudness imitates the perceptual loudness.
- ***Computation of BSD:*** Both the original and the reconstructed speech are processed through the two steps described above. The BSD is then computed as the Euclidean distance between the two.

Mapping BSD to MOS

To map the BSD to Mean Opinion Score, we used the data in Table 3. A second-order equation was obtained by performing least square regression on the data.

BSD	MOS
9.725	1.58
3	3.07
0.475	4.00
0.412	4.07

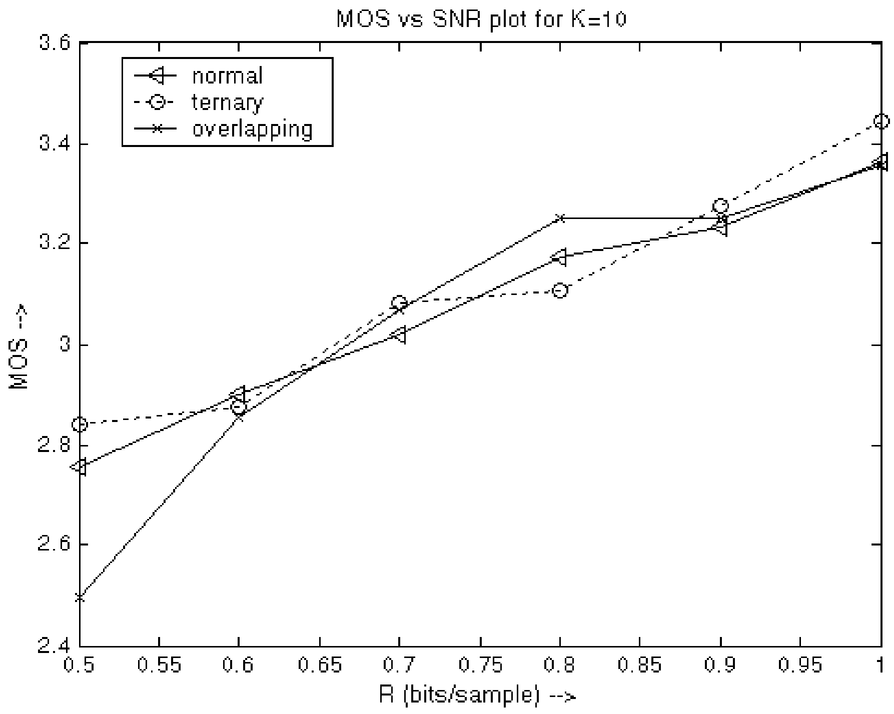
Table 3 MOS for different values of BSD

(Note: This data was obtained from Dr. Robert Yantorno, Temple Univ., PA)

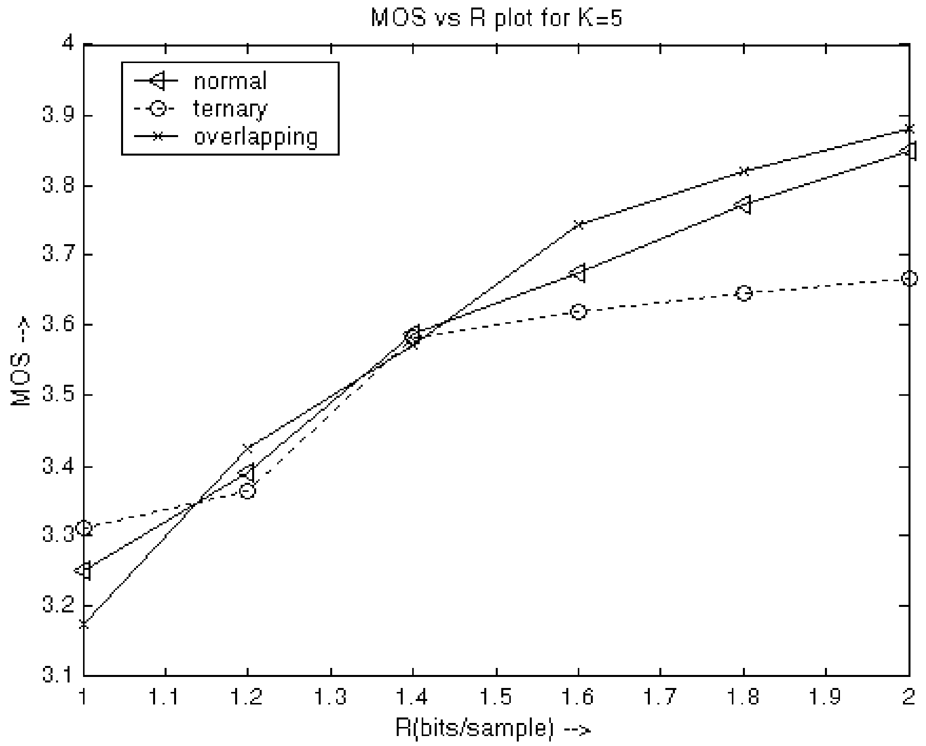
The equation that we obtained to estimate MOS from BSD is:

$$\text{MOS} = 0.0169 - 0.4359*\text{BSD} + 4.2253*\text{BSD}^2$$

Using this, MOS was estimated for various rates for the special codebooks discussed in the previous section to compare their subjective performances. Fig.5 in the next page compares their MOS obtained for various codebooks. From the graphs we see that at low rates ($k = 10$), all the codebooks perform more or less same. But at higher rates ($k = 5$), overlap and normal codebooks are 'perceptually' better than the ternary codebook.



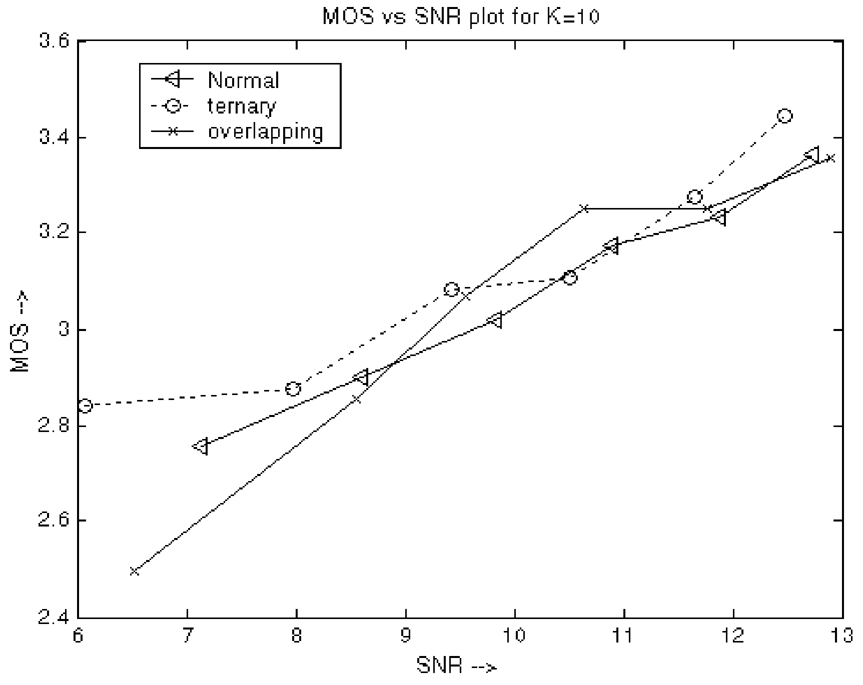
(a)



(b)

Fig.5 MOS vs. Rate for different codebooks. (a) k=10 (b) k=5

We now try to examine the validity of using MSE as a measure of perceptual distortion. Fig.6 shows the relation between SNR and MOS for the various codebooks considered. We see that MOS increases with SNR and the relationship is almost linear in the case of the normal codebook. Hence, we conclude that MSE is a reasonably good fidelity criterion, though not the best, for the designing CELP coders.



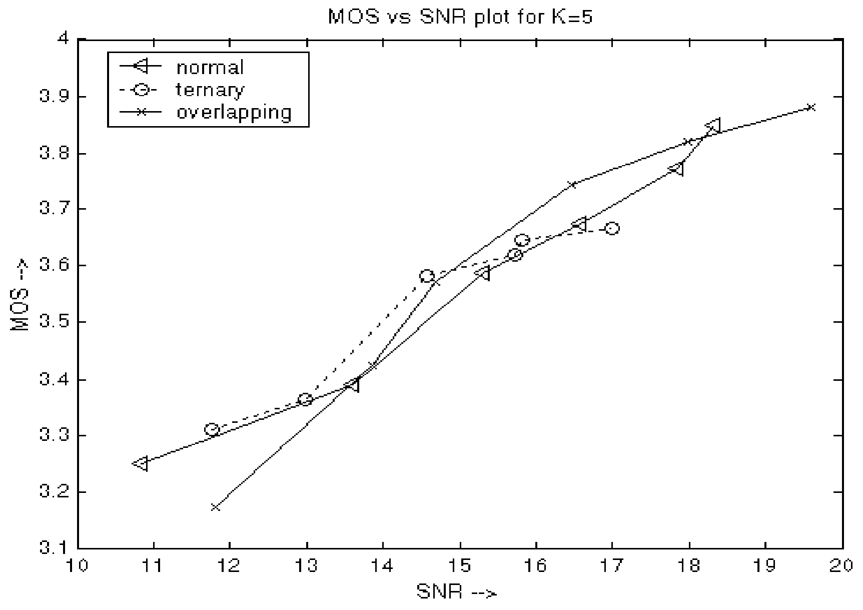


Fig.6 Correlation between MOS and SNR for different codebooks.

6 Variable rate coding

We conducted an experiment to see whether variable rate coding would be help improve the rate in a CELP coder. We estimated the probability of each of the code vector in a codebook with $k = 10, M = 512$. This was done by encoding a large number of speech samples and finding the frequency of occurrence of each codevector. With these probabilities, the entropy of the source was found to be 0.88. This suggests that all the code vectors are more or less equally probable. Hence, there is not much to gain by using a variable rate code in CELP.

7 Conclusions

In this term project, we designed a simple CELP coder that produces good quality speech at low bit rates. Its performance at different rates was studied and its computational complexity analyzed. To reduce its complexity, various "special codebooks" were used and their performance was evaluated. We found that some of these codebooks achieve significant computational savings without compromising a great deal on performance. We used BSD as a measure to compare the subjective quality of the different codebooks, since it is closely correlated with MOS. Using this, observed that MOS and SNR are closely correlated which suggests that MSE is not a bad distortion criterion. From our last experiment, we concluded that there is not much to gain by using variable rate coding in CELP.

REFERENCES

[1] M. R. Schroeder and B. S. Atal, "Code-excited linear prediction (CELP): High quality speech at very low bit rates", in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Mar 1985, pp 937-940.

[2] W. B. Kleijn, D. J. Krasinski and R. H. Ketchum, " Fast Methods for the CELP speech coding algorithm", in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Aug 1990, pp 1330-1342.

[3] R.Goldberg and L.Riek, "A practical handbook of Speech Coders", CRC Press, 2000.

[4] Shihua Wang, Andrew Sekey, and Allen Gersho, " An Objective Measure for Predicting Subjective Quality of Speech Coders", *IEEE Journal on selected areas in communications, June 1992*, pp 819-829.