

Linear Prediction, Synthesis, and Spectrum Estimation (Project 2)

Ashwin Swaminathan (ashwins@umd.edu)

Aswin C Sankaranarayanan (aswch@umd.edu)

ENEE624: Advanced Digital Signal Processing

Instructor: Dr. Min Wu

December 18, 2003

1 Introduction

Speech coding is a major issue in digital speech processing. The objective behind speech coding is to represent the speech waveform with as small a number of bits as possible so that the resultant signal has no perceptual difference from the actual signal.

In this project, we explore the usage of Linear Predictive coding (LPC) [1] in representing and coding speech signals. The basic principle in LPC coding is to minimize the sum of the squared differences between the original and the estimated speech signal, with the estimated signal being represented as a linear combination of its past.

This project is divided into 4 parts. In task 1, we first estimate the average duration of the speech signal within which the signal is approximately wide-sense stationary (also called the frame length). We propose two different algorithms based on the order statistics of the process. We then build a p^{th} order linear predictor to estimate the speech signal and show the error due to prediction. We then use these model co-efficients and

the residue to reconstruct the signal. In task 2, we explore the effects of quantization on the model co-efficients. We determine the maximum compression that we can achieve so that the resultant signal has no perceptual difference from the actual signal.

In task 3, we use a non-parametric approach to estimate the spectrum of the given input signals and explain the results obtained in task 1 and 2 based on them. In task 4, we further develop an algorithm to de-noise the given input signals.

2 Task 1: Linear Predictive Coding and Associated Parameter estimation

2.1 Finding the Frame Length

Speech signal is a slowly varying time signal in the sense that it is can almost be approximated as a stationary signal over short durations of time. We first design an algorithm to find the frame length of the given signal and then sub-divide the speech signals into these frame lengths so that the speech signal can be considered as quasi-stationary in these short intervals of time so that we can employ Linear Prediction Coding to represent the speech signal.

We develop 2 different methods to estimate the frame length based on the order 1 and order 2 statistics of the process.

2.1.1 Estimating the Frame length based on Order 1 statistics

For a wide sense stationary process the mean of the process is a constant independent of time. That is

$$E(x[n]) = m \tag{1}$$

We use this essential principle to estimate the frame length. Let L be the frame length. We initially subdivide the original $x[n]$ process into frames $y[n, k]$ defined as shown below

$$y[k, n] = x[nL + k] \quad \forall n = 1, 2, \dots, N \text{ and } k = 1, 2, \dots, L \tag{2}$$

where N is the number of frames. We compute the mean of the resultant $y[.,.]$ process for each k .

$$m(k) = \sum_{n=1}^N y[k, n] \quad (3)$$

For the process to be wide-sense stationary, the mean must be independent of k and must be as constant as possible. This would mean that a good estimate of the *frame length* would be that value of L that minimizes the variance of the $m[k]$ series.

2.1.2 Estimating the Frame length based on Order 2 statistics

For a wide sense stationary process the autocorrelation of the process depends only on the difference of time. That is

$$E(x[n]x[n+k]) = r_x[k] \quad (4)$$

In this method, we again sub-divide the signal $x[n]$ into frames $y[n, k]$ defined as in (2). We then compute the autocorrelation of the process as

$$r_x[k, d] = \sum_{n=1}^N y[k, n]y[k+d, n] \quad (5)$$

A best estimate of the *frame length* would be that value of L at which the $r_x[k, d]$ series is independent of k and dependent only on d . This would mean that the variance of the $r_x[k, d]$ series is as close to zero as possible for all given d 's.

2.1.3 Results of simulation

The two algorithms described in the previous section were implemented and the results are shown in table (1). The plot of the variance of the mean and the average variation of the autocorrelation function are shown in Fig.(1)

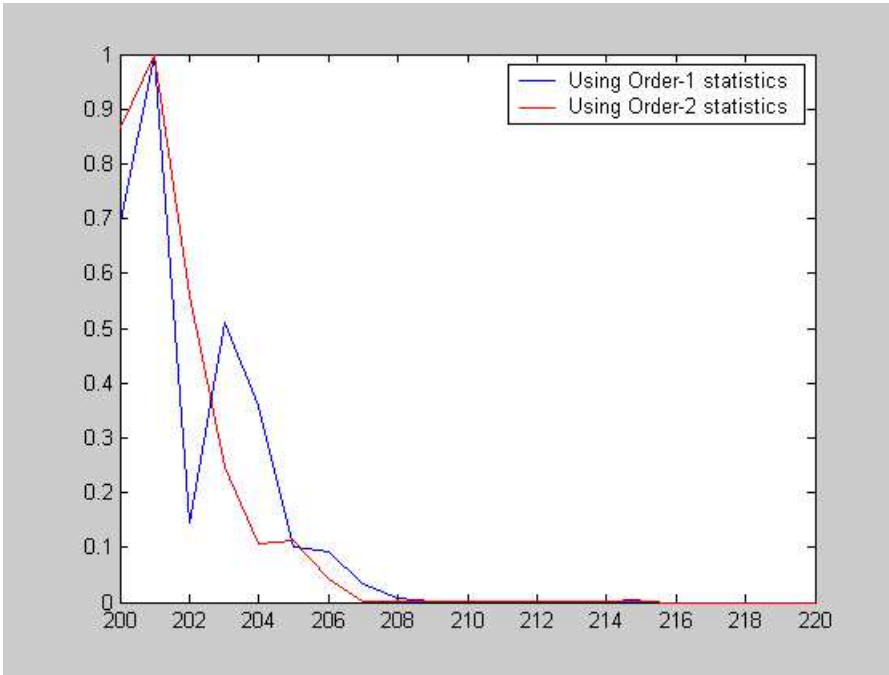


Figure 1: The frame length estimated using Order-1 and Order-2 statistics

2.2 Linear Predictive Coding

Linear prediction analysis is historically one of the most important speech analysis techniques. The $(n + 1)^{th}$ sample is estimated using the past p samples (p being the order of the Linear predictor).

$$x[n + 1] = \sum_{k=1}^p a[k]x[n - k] \quad (6)$$

We solve for the optimal filter co-efficients by minimizing the least squares error.

Audio file	Frame length estimate	Frame length estimate	Frame length in ms
	Order 1 method	Order 2 method	
test1.wav	211	217	13.56
test2.wav	103	113	14.12
test3.wav	90	80	10
test4.wav	90	90	11.25

Table 1: Comparison of the frame length estimates obtained for different input signals using the methods described above

This leads to a set of linear equations in the filter parameters. The auto-correlation is estimated using the biased estimate as shown in (7). The biased estimator was used as it assures the stability of the estimated filter. The filter parameters can be optimally solved using a $O(n^2)$ called the Levinson-Durbin algorithm.

$$r[k] = \frac{1}{N} \sum_{i=1}^{N-k} x[i]x[k+i] \quad (7)$$

2.2.1 Results And Analysis

The file *test1.wav* was taken as input and the LPC parameters were computed for it. These parameters were used to reconstruction. The following things were observed.

1. Qualitative Analysis: The predicted (reconstructed) audio, when listened to, was clear enough for the speech to be distinguished clearly, though there was a presence of noise.
2. Subjective Analysis: The waveforms of the original and reconstructed signal were compared and the error as a function of time calculated. These are shown in figure (2).

The fact that the error magnitude is as high as one-third the original signal magnitude should not be a cause of worry as this waveform as no information content as such, in the sense that the error waveform or the residue as we'll call it henceforth can be quantized using lesser number of bits, without appreciable loss of quality. This property is established and used in Task 2.

2.3 Effect of the order of the Linear Predictor

The algorithm shown above was implemented with different prediction orders. The average error in prediction in a particular frame of the signal is shown in Fig. (3). It is seen that that the average prediction error reduces from its maximum value at $p = 1$ to a minimum at round $p = 10$. The error however remains almost a constant for $p > 10$. Similar statistics is observed over all frames. We also have a trade-off between the prediction order and the amount of compression we achieve. Lower the prediction

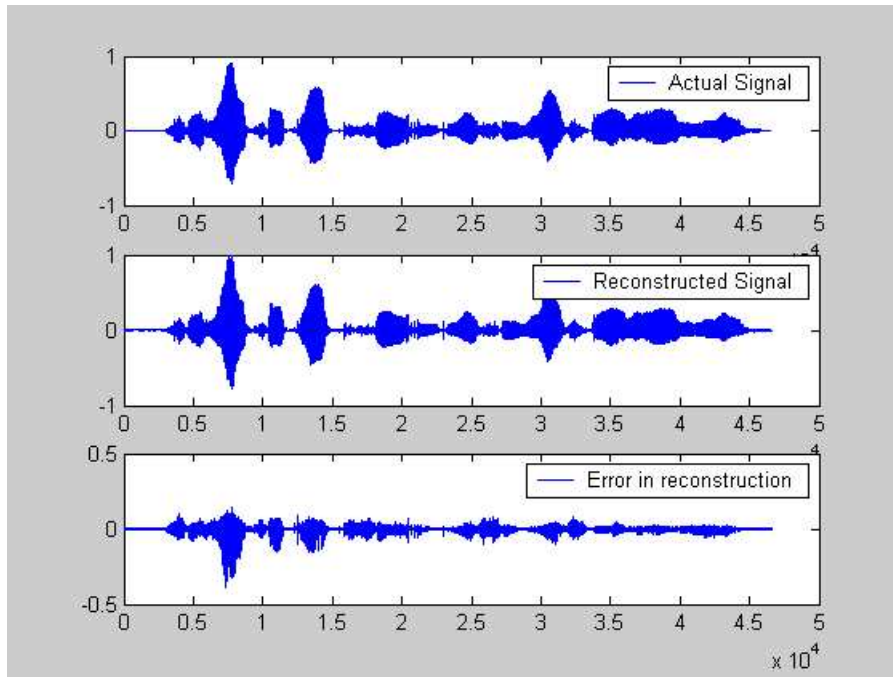


Figure 2: **The original signal, reconstructed signal and the error in reconstruction**

order the more compressed the final output. Thus we could choose a prediction order of 10 beyond which the mean-square error is almost a constant.

2.4 Reflection parameters

The AR parameters can be equivalently represented as the reflection co-efficients through the Levinson-Durbin algorithm. A lattice structure can be used in implementing the model. This model is shown in Fig. (4). This model gives results similar to those obtained in the previous sections. However, the reflection co-efficients are less susceptible to quantization effects and the lattice structure is easy to implement and so this method is generally preferred in practical applications.

3 Task 2: Compression Scheme and Results

For the second task, a storage scheme was developed which enabled writing all the required information into a binary file (as a bit-stream). This is read by the decoder,

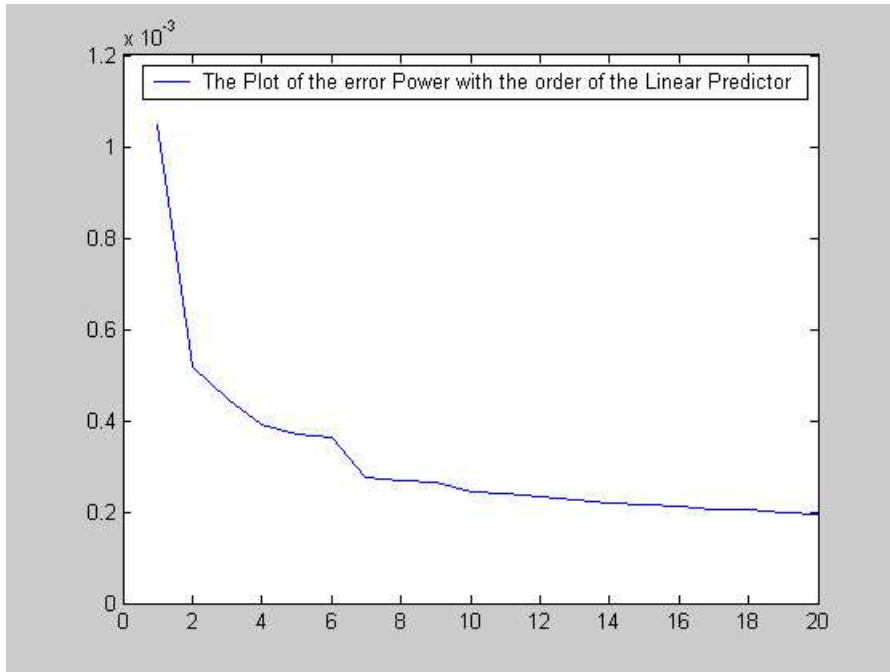


Figure 3: The order of error power with the order of the linear predictor

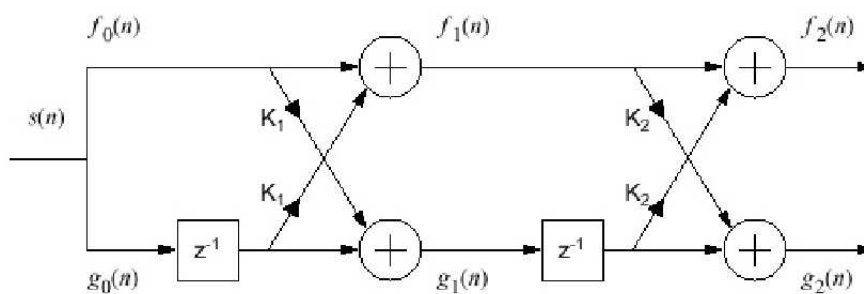


Figure 4: The lattice structure for error prediction

which retrieves all the information from the binary file. The storage format was developed. Other than this, a coding scheme was developed which would decide when to update and transmit the model parameters and when not to.

3.1 Pre-emphasis

The lower frequency components of the speech signal usually contain more energy than the higher frequency components of the speech signal. The pre-emphasis filter is usually used to boost the high frequency components. The pre-emphasis filter is an all-pole AR(1) filter given by (8)

$$s[n] = s[n] - b.s[n - 1] \quad (8)$$

It can be shown that the optimal value of the parameter b that minimizes the error can be written as

$$b = \frac{r[1]}{r[0]} \quad (9)$$

The typical values of b lie in the range 0.95 to 0.99.

3.2 Quantization

We represent the given speech signal in terms of the compressed versions of the linear prediction co-efficients, the residue and the first p values of the signal in each frame. These parameters are individually quantized and stored. The various quantization schemes used to quantize each one of these signals are discussed below.

1. **Quantization of the Linear Prediction parameters:** First and the most straight forward approach is to quantize and store the linear prediction parameters $a[k]$. However, it was observed that a slight change in the linear prediction parameters can make the system unstable and also give erroneous prediction. To circumvent this problem, the LP parameters were first converted to reflection coefficients which are an equivalent representation of the LP parameters. The reflection coefficients were then quantized and stored. It was observed that the reflection co-efficients are less susceptible to slight changes caused due to quantization.

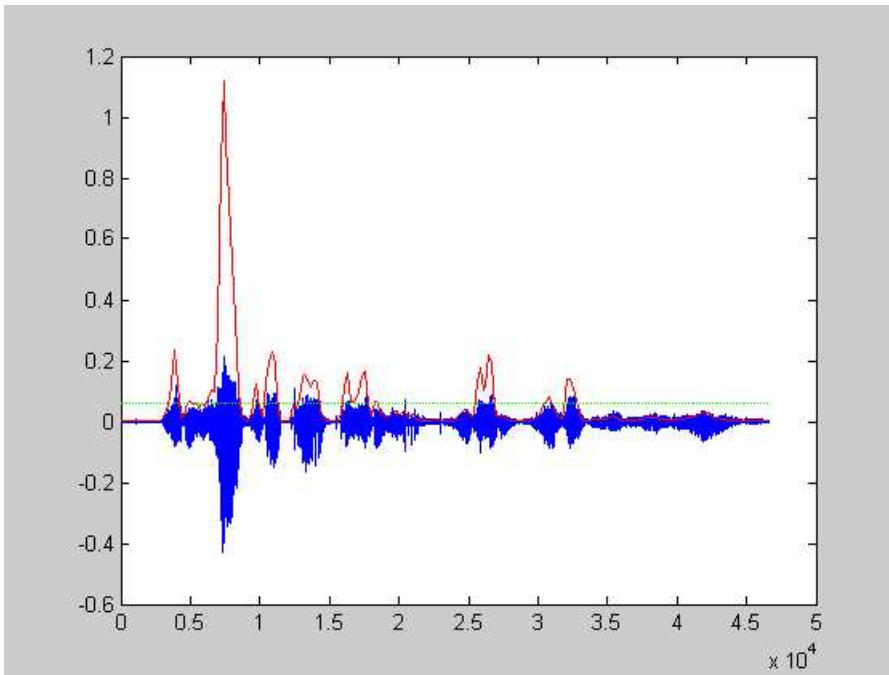


Figure 5: **The plot of error power for every frame using the 10 order linear predictor**

2. **Quantization of the residue:** The performance of the system depends to a large extent on the actual representation of the residue. The efficiency in the compression depends on how fewer bits we use to compress the residue. We use the following algorithm to compress the residue.

- **Dividing the error signals:** We first divide compute the mean error power P_p caused due to linear prediction for each frame. The plot of the error power for every frame is shown in Fig. (5). The speech can be divided into *voice* and *non-voice* components. It is observed that the error power is comparatively large for voice components and small for non-voice components. We there for divide the error into 2 parts

- Type 1: the error for the voice components of the speech signal. This happens when the error power is greater than a certain *threshold*.
- Type 2: the error for the non-voice components of the speech signal.

This happens when the error power is lesser than the *threshold*.

The *threshold* is chosen as the mean of the error power over all frames. This leads to a storage of one additional bit per frame to indicate to the receiver if the residue is Type 1 or 2. Speech signals are inherently low-pass in nature. Hence for an accurate representation of the resultant signal, we need less errors in the low pass components of speech signal and we can tolerate more errors in the high pass component. Therefore, we would require more bits to store the type-2 error signal. Inadequate representation of the type-2 error signal would lead to a uniform background noise in the final compressed signal.

- **The Discrete Cosine transform:** The Discrete cosine transform is a loss-less transform that can be used in the representation of the signal. To eliminate the errors in the frequency domain representation of the signal, we quantize the DCT of the error signals instead of the error signals itself. We quantize the DCT of type-1 error signals with $NBits_1$ bits and the type-2 error signal with $NBits_2$ bits with $NBits_2 > NBits_1$ and store them into the file. For reconstruction, we take the inverse discrete cosine transform and then use the overhead to classify the error as type-1 or type-2 from which the original residue can be reconstructed.

3. **Quantizing the first p signal values of each frame** The first p signal values of each frame are quantized uniformly and stored.

3.3 Model Update

Further reduction in the number of bits could be achieved if we do not transmit the reflection co-efficients for all frames. We transmit the reflection co-efficient for a frame only if the difference in reflection co-efficient between subsequent frames is greater than a threshold ϵ . In formal terms, this reduced to

$$\sum_{i=1}^p |\Gamma_i^n - \Gamma_i^{n-1}|^2 < \epsilon,$$

where Γ_i^n is the i^{th} reflection coefficient for the n^{th} frame. If the above equation is satisfied then we do not update the model parameters. The decoder is informed of this by a bit-stream of single bit values which take the value 1 if there is no model parameter update. By doing this, we save on transmitting a lot of bits and instead need only to transmit one additional bit per frame to indicate model update.

3.4 Quantization Schemes: Uniform vs Non-uniform

The DCT of the residue waveform had to be quantized before storage and transmission. There is a choice between using uniform and non-uniform. The pros and cons of the 2 are discussed below.

- **Representation and Distortion:** Non-uniform quantization can adjust the quantizer levels to represent the zones with more samples better, resulting in lesser distortion. Uniform quantization will represent signals immaterial of the nature of the signal — this leading to higher distortion at times.
- **Storage requirement:** Given the same number of bits for quantization, the overall storage of the quantized data is the same for the both the methods. But the overheads are much lesser in the case of the uniform quantizer. The uniform quantizer needs to store only 3 values, the maximum value, the minimum value and the number of bits used. With this, the quantizer codebook can be constructed. The non-uniform quantizer, on the other hand, need to store the entire codebook which can lead to an exponential increase in the overhead storage with the increase in the number of bits used to perform the quantization.

After carefully consideration, uniform quantization was chosen for the task.

3.5 The Gandalf Storing Scheme

The list of parameters that need to be transmitted to the decoder is described below.

- The order of the AR model p $\langle 8 \text{ bits} \rangle$.
- The length of each frame L $\langle 8 \text{ bits} \rangle$.

- Number of frames $n <16 \text{ bits}>$.
- Sampling frequency of the original waveform $f_s <32 \text{ bits}>$.
- Number of bits used to represent residues of the 2 types, $NBits_1$ and $NBits_2$.
- The number of bits used to represent the reflection coefficient $NBits_R$.
- To build the codebooks for the quantized reflection coefficients, we need the maximum and minimum values taken by each reflection coefficient set (one set of p values for each frame). Each of the values are stored with $<32 \text{ bits}>$.
- Similarly, for the same reason as above we need the maximum and minimum values taken by the residue signals $<32 \text{ bits each}>$.
- Residue Signal lengths $<16 \text{ bits}>$.
- Frame Residue type information, which tell if the frame is of type 1 or 2 $<1 \text{ bit per frame}>$.
- Frame Model Parameter updation information $< 1 \text{ bit per frame}>$.
- The first p samples of each frame for initial values in reconstruction.
- Residue Signals of type 1 $<NBits_1 \text{ bits each}>$.
- Residue Signals of type 2 $<NBits_2 \text{ bits each}>$.
- Reflection Coefficients $<NBits_R \text{ bits each}>$.

A proprietary storing scheme, called was developed *Gandalf*. Two functions were written called `write2bitstream.m` and `readfrombitstream.m`, construct a logical bitstream over an array of numbers which are filled and read bit-by-bit. Each function can read a pre-specified number of elements of arbitrary precision from a array, from any location in the array.

Shown in table (2) is the framing digram of the scheme.

The output is written in the “results” folder under the extension `.gandalf`.

p (8)	L (8)	n (16)	f_s (32)	
$NBits_1$ (8)	$NBits_2$ (8)	$NBits_R$ (8)	Ref. max (32)	Residue type I length len_1 (8)
Ref. min (32)		Type 1. max (32)		
Type 1. min (32)		Type 2. max (32)		
Type 2. min (32)		Residue type II length len_2 (8)		
Type I/II classification ($n \times 1$)				
Model Updation Information ($n \times 1$)				
Initial values ($n.p \times 32$)				
Type I residue ($NBits_1 \times len_1$)				
Type II residue ($NBits_2 \times len_2$)				
Quantized Reflection Coefficients				

Table 2: Framing Schemata for storage. Note that sizes of box need not necessarily represent the size of the field. The bracketed number shows the number of bits used for storage of that field

3.6 Results and Analysis

In results, we present the compression achieved with the file *test1.wav*, with appreciable playback quality. To determine the quality of reconstruction (playback), we considered the following.

1. The difference in waveforms between original and reconstructed.
2. Playback quality as to the human ear should not be appreciably poor, when compared to the original speech.

Some of the key figure are shown in table (3).

4 Task 3a: Analysis for the remaining audio signals

The results are shown in table (4).

Feature	Value
Original File Size	93354
Compressed File Size	32918
Frame length	217
Order of estimation	10
No. of Model Updation	177 / 215
Compression Ratio	35.3 %

Table 3: Performance with Test1.wav

Feature	Test 2	Test 3	Test 4
Original File Size	25356	3244	15884
Compressed File Size	11874	1899	8092
Frame length	217	217	217
Order of estimation	10	10	10
No. of Model Updation	83/112	20/ 20	49/88
Compression Ratio	46.8 %	58.5%	50.9%

Table 4: Performance for Test2, Test3 and Test4

5 Task 3b: Spectrum Estimation

We first used the non-parametric approach to determine the nature of the spectrum. This would give us the necessary information to find a model that could describe the underlying spectrum better.

5.1 Non parametric estimation

Three methods we tried out, namely

1. **Periodogram:** The periodogram is defined as

$$\hat{P}_{per}(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j2\pi f n} \right|^2$$

The periodogram is the simplest of all estimation techniques. But it is not a consistent estimator.

2. **Averaged Periodogram:** The periodogram can be viewed as the fourier transform of the autocorrelation of the signal. The high variance of the periodogram can be attributed to the fact that there is very little averaging in estimation of some of the lags of the autocorrelation. To get a More reliable estimator we divided a signal into frames of length L . Let x^1, x^2, \dots, x^n , each of the elements of length L . The averaged periodogram computes the periodogram of each and averages them. This way the variance is reduced, but at the cost of the biasing of the spectrum. The periodogram is defined as

$$\hat{P}_{avg}(f) = \frac{1}{n.L} \sum_{i=1}^n \left| \sum_{k=0}^{L-1} x^i[k] e^{-j2\pi f k} \right|^2$$

3. **Minimum Variance Spectrum Estimation:** Both of the above methods fit a general model without considering the nature of signal. MVSE takes into account the nature of the signal before estimating the psd. The expression for PSD is given as:

$$\hat{P}_{MVSE}(f) = \frac{1}{\mathbf{e}^H \mathbf{R}_{xx}^{-1} \mathbf{e}},$$

where R_{xx} is the autocorrelation of order M (some positive integer), and \mathbf{e} is a column matrix whose entries are,

$$\mathbf{e} = \begin{pmatrix} 1 \\ e^{j2\pi f} \\ e^{j4\pi f} \\ e^{j6\pi f} \\ \cdot \\ \cdot \\ \cdot \\ e^{j2\pi f(M-1)} \end{pmatrix}$$

Each of the 4 test audio signals are estimated by the above 3 methods. Fig. (6) and (7) show the estimated PSD for the four signals by the above techniques.

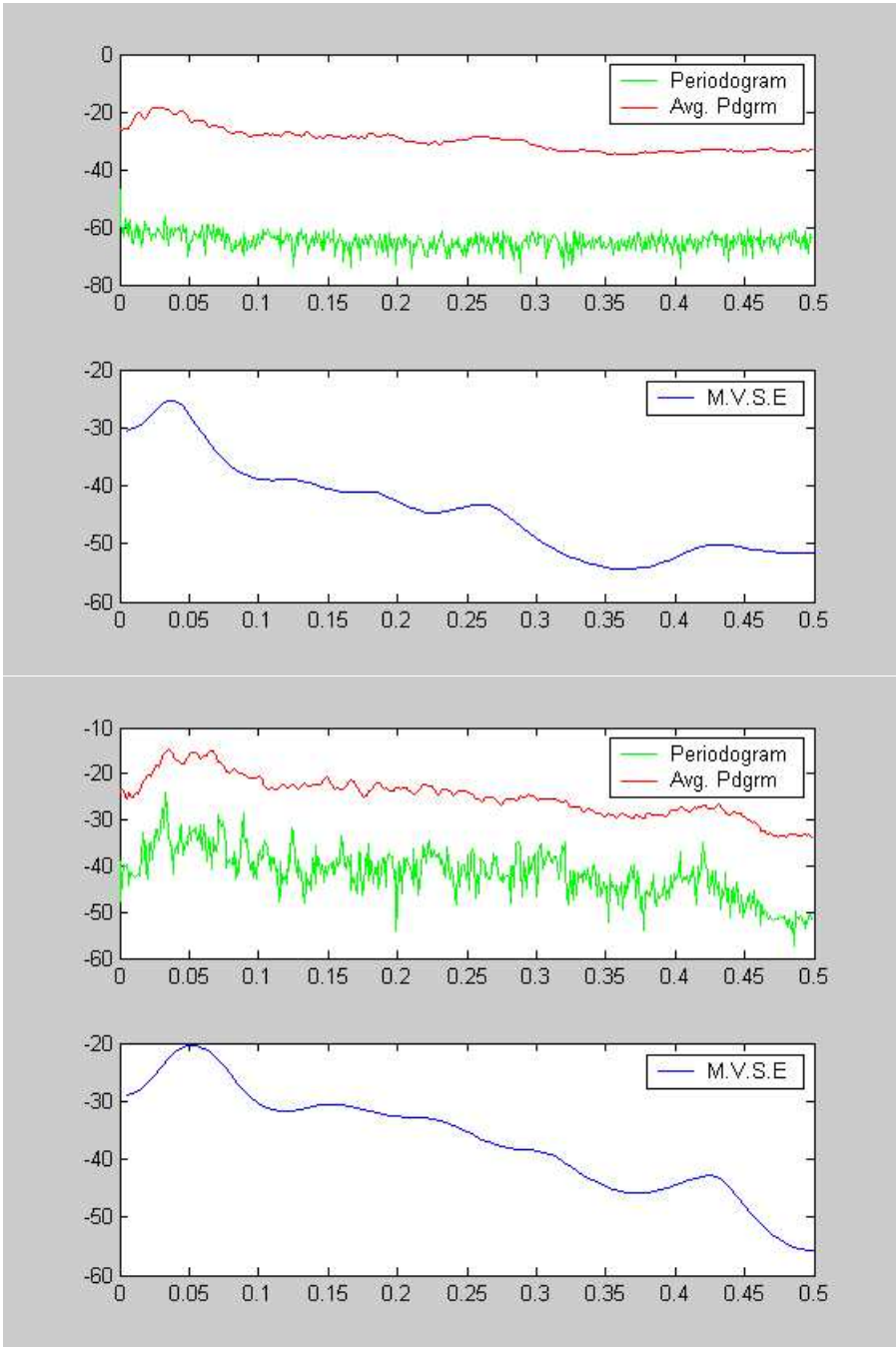


Figure 6: **Periodogram, Average periodogram and the MVSE for test1.wav and test2.wav**

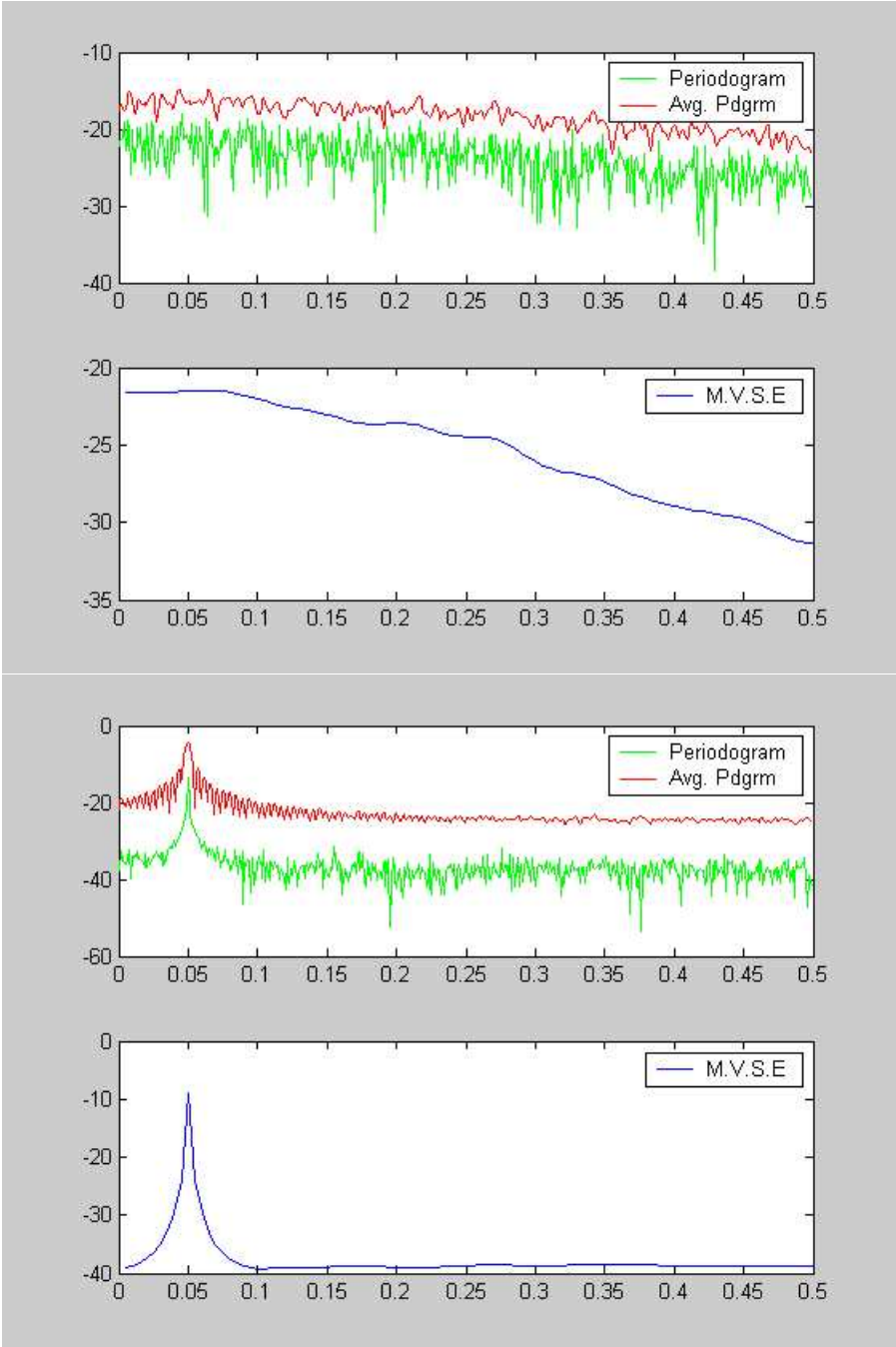


Figure 7: Periodogram, Average Periodogram and the MVSE for test3.wav and test4.wav

5.2 Analysis

Test1 and 2, from their spectrum show that they are ARMA signals. An ARMA(p,q) model can be used to fit their spectrum. Test4.wav is a mono-tone signal (or a dual-tone with 2 frequencies close to each other). For better resolution of this signal we can use parametric techniques with assume the model as sinusoids buried in noise [2]. The signal was modeled as a sum of sinusoids buried in white noise. MUSIC algorithm was used to determine the location of the peaks. The PSD, according to MUSIC is given as,

$$\hat{P}_{MUSIC} = \frac{1}{\sum_{i=p+1}^M |\mathbf{e}^H \hat{\mathbf{v}}_i|^2},$$

where $\hat{\mathbf{v}}_i$ is are the eigenvectors corresponding to the noise.

Different values were tried for p , the number of sinusoids. The values most suited was $p = 2$, as the number of peaks didnot increase beyond that. The estimated spectrum is shown in figure .

6 Conclusion

In this project, we first did LPC coding of a speech signal after analysing and determining parameters of representation such as frame length, order of representation, quantization of residue and storage schemes. An encoder and a decoder was developed was compressing and retrieving this signal. The DCT of the residue was compressed and sent. Reflection coefficients were used to represent the AR model, given their relatively lesser susceptibility to quantization noise.

The spectrum of the signal were estimated by three methods - namely, periodogram, averaged periodogram and minimum variance spectrum estimator algorithm and the results discussed.

References

- [1] Deller Jr., J. R., Proakis, J. G., Hansen, J. H. L., "Discrete-Time Processing of Speech Signals," Macmillian Publishing Company, 1993

- [2] Kay, S. M., "Modern Spectral Estimation: Theory and Application," Prentice-Hall, Englewood Cliffs.