NEAR EAST UNIVERSITY

GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

LINEAR PREDICTIVE CODING

Burak Alacam

Master Thesis

Department of Electrical and Electronic Engineering

Nicosia - 2002

Burak Alacam: Linear Predictive Coding

Approval of Director of the Graduate School of Applied and Social Sciences

Prof. Dr. Fakhraddin Mamedov
Director

We certify that this thesis is satisfactory for the award of the degree of Master of Science in Electrical and Electronic Engineering

Examining Committee in Charge:

Assist. Prof. Dr. Kadri Büründcük,	Chairman of Committee, Electrical and Electronic Engineering Department, NEU
Prof. Dr. Parviz Ali Zada,	Committee Member, Electrical and Electronic Engineering Department, NEU
Assoc. Prof. Dr. Doğan Ibrahim,	Committee Member, Computer Engineering Department, NEU

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Prof. Dr. Fakhraddin Mamedov and my committee members,Assist. Prof. Dr. Kadri Bürüncük, Assoc. Prof. Dr. Doğan İbrahim Akay and Prof. Dr. Parviz Ali Zada for carefully reviewing my thesis and providing valuable suggestions. I would also like to thank Assoc. Prof. Dr. Adnan Khasman for his help and useful suggestions.

I would like to also thank fellow students working in Near East University who have made my three semesters of stay at University a great learning experience in every aspect. I would also like to thank Mr. Cemal Kavalcıoğlu and Mr.Hani for their contribution and useful suggestions .

Finally, I would like to thank my family and my good friends from Cyprus for their constant encouragement and emotional support during my entire graduate school study.

ABSTRACT

Speech coding is important in the effort to make more efficient use of digital telecommunication networks, particularly wireless systems, and to reduce the memory requirements in speech storage systems. The desire for a low-rate digital representation of speech is often contrary to the demand for a high quality speech reconstruction. Linear Predictive Coding (LPC) is one of the most powerful speech analysis techniques, and one of the most useful methods for encoding good quality speech at a low bit rate. It provides extremely accurate estimates of speech parameters, and is relatively efficient for computation.

In this thesis we implement a Linear Predictive Coding (LPC) technique designed for good quality speech coding at bit rates as low as 2.4 kb/s. Windowing and preemphasis are important in the accurate determination of the speech parameters. The computation of the LPC parameters is explanined. Besides the reflection coefficients,there are other sets of parameters carrying exactly the same information.There are derived from the reflection coefficients and are used in cases where their properties can lead to better speech quality.

The software implementation of a 2.4 kb/s LPC coder is described. The main features of the implemented LPC coder are: Pre-emphasis Filtering to remove the natural frequency roll-off in speech, Data Windowing ,AR Parameter Estimation, Pitch Period and Gain Estimation to determine whether the block in question was voiced or unvoiced ,Quantization ,Decoding and Frame Interpolation.

TABLE OF CONTENTS

LIST OF FIGURES

Page

1. INTRODUCTION

1.1 Overview

In communications, digital signals become increasingly important. Digital signals have some advantages over conventional analog signals: they can be compressed more efficiently, messages can be secured more easily against unwanted reception by others and digital signals are more robust to channel errors when proper error correction is performed. A disadvantage is that a larger channel bandwidth is required. For analog signals, the minimum required channel bandwidth equals the highest frequency in the signal. For digital signals, the required channel increases with the number of bits used to represent each sample and the sampling frequency.

There are numerous applications where the communication of digital speech signals is involved. Some examples are: mobile satellite communications, cellular mobile radio, teleconferencing, cordless telephones, mobile telephony. In many applications it is necessary to reduce the data rate. For example, the number of potential users of mobile telephones is very large, so it is important to keep the number of bits as low as possible, because the channel capacity is limited. Other motivations for lowering the bit rate are transmission and storage costs.

Frequencies up to about 20 kHz are audible by humans, but in speech the majority of the information is contained in the frequency band up to about 4 kHz. Speech signals consequently can be represented more efficiently than audio signals: a lower sampling frequency can be used and hence the bit rate is lower. For telephone-bandwidth speech a sampling frequency of 8 kHz is applied. If 16 bits per sample are used, telephone bandwidth speech has a bit rate of 128 kbit/s and audio signals have a bit rate of about 700 kbit/s (per channel); too high for many applications. For these applications, coding systems have to be developed which represent the digital signals by binary code numbers with a lower bit rate.

At low rates, there is always a loss in information and the goal of the coding systems is either to maximise the quality for a given bit rate or to minimise the bit rate for a given quality.

The bit rate can be brought down much further for speech signals than for audio signals. One reason is that the quality requirements are usually higher for audio signals than for speech signals. Another reason is that speech has some specific properties that can be exploited. A simple speech production model is available that assumes the speech to be formed by the spectral shaping by the vocal tract of either a quasi-periodic pitch pulse excitation for voiced speech or a noise-like excitation for unvoiced speech. Audio signals form a much broader class of signals.

A class of speech coders which has been applied successfully is formed by the Linear Prediction based speech coders. In these coders, the coded speech is synthesised by the excitation of a time-varying all-pole filter. The filter coefficients are obtained with an autoregressive estimation method and describe the spectral envelope of the signal. Linear Prediction forms the core of many coders at various bit rates.

Examples are: the Multi Pulse coder , operating at around 16 kbit/s, the Regular Pulse coder (13 kbit/s), Code Excited Linear Prediction [14] (4-8 kbit/s). These Linear Predictive Coding (LPC) algorithms have led to several standards.

The main difference between different Linear Predictive coders is the method that is used for coding the excitation of the Linear Prediction synthesis filter and in particular the number of bits used for this purpose. As the bitrate decreases, an accurate estimation,interpolation and quantisation of the autoregressive model parameters becomes increasingly important, because at low rates errors in the LPC model cannot be easily compensated for by the excitation. An accurate representation of the model is very important for the quality.

Because the Linear Prediction model is such an important part of modern coders, this thesis is devoted mainly to the various aspects of Linear Prediction: estimation, interpolation and quantisation.

The main purpose of the research was not to develop new coding techniques, but rather to gain theoretical and practical insight in the advantages and disadvantages of existing Linear Predictive Coding technique.

The objective of this project is to firstly study the spectral features of the speech signals and then derive a general framework to do analysis and LPC coding of speech signal. In this framework, we will try different methods to do each part of the job. Although there have been already several successful standards in speech coding which are used extensively in mobile communications, we don't intend to implement any standards here because of the detailed complexity of implementation.

I prefer to do some research for the principles of speech coding and try to compare some of the method by discussing the strength or disadvantage of each method and hope to derive some insights for future works.

This thesis is organised as follows. In chapter 2 several speech coding algorithms are discussed and give a little bit literature survey of it , most of which use Linear Prediction. In chapter 3 a brief review of autoregressive theory will be given. Mainly results are given which are necessary for a proper understanding of the contents of later parts of the thesis on estimation, interpolation and quantisation in matlab implementation. Different autoregressive estimation methods are discussed and it is shown that the well-known and often used autocorrelation method is very sensitive to edge effects and is therefore not a suitable method. A tapered data window reduces this sensitivity but increases the variance of the models. Other methods are available that do not need a window.In chapter 4 the implementation of the LPC coding are discussed and algoritm is given.Implementation of speech coding algorithm is programmed in matlab software packages and using its toolbox. The thesis is concluded with a summary of our work in Chapter 5, along with suggestions for future work.

2. SPEECH CODING

2.1 Overview

In this chapter an overview is given of speech coding techniques at several bit rates. Most of them use Linear Prediction. This overview is not meant to be complete; its purpose is to make the reader somewhat familiar with Linear Predictive Coding which is necessary for a proper understanding of later chapters. Section 2.2 treats the subject of quantisation and coding. In section 2.3 a description of speech production and speech sounds is given. Coders based on linear prediction can be considered as being based on a simple speech production model. This model is explained in section 2.4. Section 2.5 describes various speech coding algorithms and techniques. Section 2.6 briefly describes some measures for the quality of coded speech.

2.2 Quantisation and Coding

The two main forms in which a signal can be transmitted are *analog* and *digital*. An analog signal can have any value in a continuous range. If the signal has a bandwidth of W Hertz, all its information is contained in 2W samples per second of the signal and the original continuous time signal can be exactly recovered from this discrete time signal [4][9][33]. A digital signal can be obtained by *quantisation* of the samples taken from an analog signal, i.e. limiting them to a discrete set of possible values. The use of digital signals has some advantages over the use of analog signals: digital signals can be compressed more efficiently, messages can be more easily secured against unwanted reception by others and digital signals are more robust to channel errors when coding is properly performed. *Coding* of a quantised number (or vector of numbers) means that it is represented by a binary code number. In a coding system, these code numbers are transmitted to the decoder which can make a reconstruction on the basis of the information provided by the code numbers. In this thesis, the distinction between quantisation and coding will not always be made. A drawback of digital signals

is that the bandwidth of the signal is larger, and a channel must support this larger bandwidth.

2.2.1 Scalar Quantisation

The simplest method to quantise a digital signal is *Scalar Quantisation (SQ)*. In SQ each sample of the signal is independently represented by a b bit binary number. This binary number is the code for one of the 2 possible levels the digital signal may have. If these b levels are equally spaced, the quantisation is called *uniform*. Uniformly spaced levels are optimal for the scalar quantisation of uniformly distributed random variables (in the sense of the Mean Squared Error). If the samples of the signal have a different distribution, e.g. a Gaussian distribution, the levels of the optimal scalar quantiser are non_equally spaced. Such a quantiser with non_equally spaced levels is called a *non_uniform quantiser*.

2.2.2 Vector Quantisation

Scalar quantisation does not take into account any correlations or dependencies that may be present in the signal. These dependencies can be exploited to increase the efficiency, i.e. lower the distortion at the same bit rate or decrease the number of bits for the same distortion. One way to do this is to remove the correlation or dependencies from the signal and quantise the resulting uncorrelated signal.

This signal will have a smaller variance and can be coded with a smaller error. A method for removing correlation from a signal which has proved to be very successful in speech coding is Linear Prediction. Section 2.5 contains the descriptions of coding techniques based upon Linear Prediction. An other way to exploit the correlations and dependencies in a signal is Vector Quantisation (VQ).

This section will explain the basic principles of VQ..In VQ several random variables are grouped together into a *target vector* and this entire vector is coded. This means that there is a set of *code vectors* or *representation vectors*, which form a *code book* that is known both at the encoder and the decoder. The target vector is compared with all code vectors in the code book by means of a certain *distortion measure*. The code vector which has the smallest distortion with respect to the target vector is the *winning* code vector. It is represented by a binary index. This number is sent to the

decoder and there the winning vector can be picked from the code book. If b bits are used to code a vector, the code book contains 2 code vectors. The number of bits need not be an integer multiple of the vector dimension, which means that fractional bit rates are possible, i.e. the *average* number of bits per vector element need not be an integer number and may be even less than one.

Perhaps the main advantage of VQ over SQ is that the joint probability density function of the vector elements is taken into consideration with VQ [7][19][20]. Consider figure 2.1 where a scatter plot is shown of samples taken from a two_dimensional distribution function. The distribution function of the variable 'x' covers the range from -1 to +1 and the distribution function of the variable 'y' the range from about -0.2 to +1.2. If one would quantise x and y independently with SQ, the *entire* area would be covered by rectangles. The centre of each rectangle represents one of all possible combinations of quantised values of x and y.

All combinations of x and y that are in a certain rectangle are represented by the central point of the rectangle. There are, however, areas that are not covered by the *joint* distribution function of x and y and there will never be a combination of x and y in these areas. Hence SQ is spoiling bits by covering regions which are empty. With VQ, this can be avoided because the code vectors can be placed exclusively in regions which are covered by the joint probability density function. In this way, correlations and dependencies between vector elements can be taken into consideration.

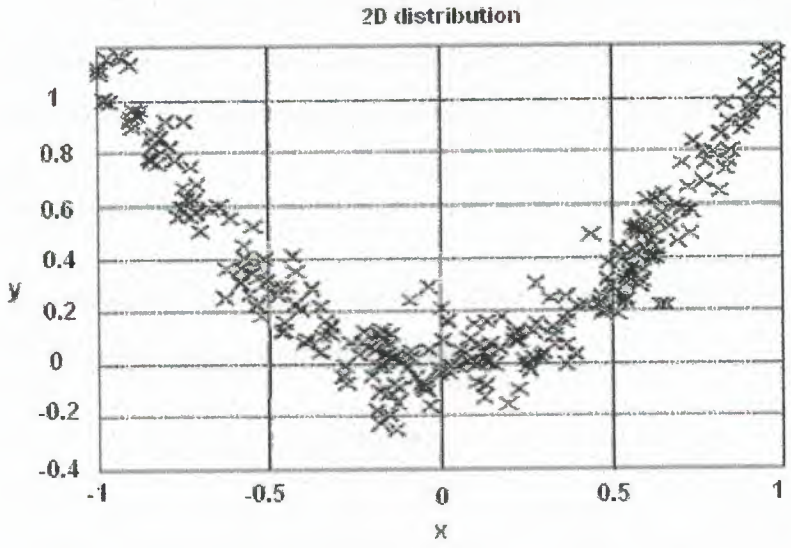


Figure 2.1 Scatter plot of samples of a two-dimensional distribution vector. Quantisation can efficiently exploit dependencies in the joint distribution of random variables. These dependencies are neglected in Scalar Quantisation. [26].

Even if the random variables are uncorrelated and there are no dependencies, VQ has an advantage over SQ. This advantage has to do with the quantisation *cell shape*. A quantisation cell or *Voronoi region* is a volume around a code vector. The quantisation cell for a certain code vector is defined as the set of all target vectors that would be assigned to that code vector in a quantisation procedure. The cells are defined by the locations of the code vectors and by the rule or distortion measure that is used to select a code vector. In SQ, the cells are rectangular boxes, in VQ the cells can have all kinds of shapes. There is some gain because of this freedom in cell shape in higher dimensions.

A drawback of VQ is its complexity in terms of storage space and computational effort. If N random variables x are quantised using scalar quantisation with b bits per variable, one has to store N2 levels, if different levels are used for each variable. For VQ of the vector of these N variables, the codebook which uses the same total number of bits as used for SQ has a size of 2 , which is generally much much larger than N2 . The search complexity is of course also much higher. In the past the use of VQ was limited to applications where a coarse quantisation with a small code book was sufficient. The development of complexity reduction techniques has made VQ useful for

applications were an accurate quantisation is necessary. These complexity reduction techniques include special code book structures and fast search methods.Another drawback of VQ is that it is more sensitive to channel errors than scalar quantisation.

2.2.3 Rate Distortion Theory

For a stationary correlated normally-distributed stochastic process x, the minimum distortion for a given bit rate is given by:

$$D_{\min} = \sigma_x^2 \gamma_x^2 2^{-2R} \tag{2.1}$$

$D_{\min}$ is the lower bound for the expectation of the squared error between x and its coded version $\hat{x}$, R is the rate, that is; the number of bits per sample of x^2, σ is the variance of x and γ_x^2 is the spectral flatness measure, given by:

$$\gamma_x^2 = \frac{\exp\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(x(w))dw\right]}{\frac{1}{2\pi}\int_{-\pi}^{\pi}X(w)dw} \tag{2.2}$$

where X(w) is the spectrum of x.

The spectral flatness measure has a value between zero and one. For an uncorrelated process with variance σ^2, γ_x^2 is equal to one and (2.1) becomes:

$$D_{\min} = \sigma^2 2^{-2R} \tag{2.3}$$

For high rates, there exists theoretical bounds on the performance of quantisers . For the coding of d dimensional vectors of normally identically distributed variables with b bits, it can be shown that the following bound exists for the mean squared distortion D:

$$D \geq \theta(d) f(d) \sigma^2 2^{-2R} \tag{2.4}$$

where σ^2 is the variance of the vector elements, R=b/d is the rate and

$$\theta(d)=\left(\prod_{i=1}^{d}\mu_i\right)^{\frac{1}{d}}\times\left(\sum_{i=1}^{d}\mu_i/d\right)^{-1} \tag{2.5}$$

$$f(d)=\left(\frac{d\Gamma(d/2)}{2}\right)^{2/d}\frac{2}{d}\left(\frac{d+2}{d}\right)^{\frac{d}{2}} \tag{2.6}$$

The coefficients μ_i are the eigenvalues of the covariance matrix of the vector elements and Γ is the gamma function. If the vector elements are not correlated with each other, their covariance matrix is a diagonal matrix and all eigenvalues are equal to σ^2 and $\theta(d)$ is equal to one. If scalar quantisation is applied, the dimension is one and the bound is:

$$D_{scalar}\geq 2.72\sigma^2 2^{-2R} \tag{2.7}$$

In the limit that the vector dimension goes to infinity while the rate is kept constant, the bound becomes identical to the rate distortion bound (2.3). This illustrates the advantage of a higher dimension that VQ has over scalar quantisation.

2.3 The Speech Signal

This section briefly explains how speech is produced and gives a short overview of different classes of speech.

Speech is formed by the flow of air from the lungs. The air flows through the *larynx*, which contains the *vocal cords*, to the *pharynx* (throat cavity) and next leaves the head via the oral cavity and the lips or via the nasal cavity and the nostrils. Both the oral cavity and the nasal cavity can be closed. The tube leading from the larynx to the pharynx and from there on to the oral and nasal cavities is called the *vocal tract*.

The two main mechanisms with which speech sounds can be formed are *voiced excitation* and *voiceless excitation*. Voiced excitation arises when the air flow causes a

vibration of the vocal cords. By the influence of Bernoulli forces the vocal cords open and close quasi-periodically. The average vibration frequency, the *pitch* frequency, is about 100 Hz for males and twice that for females. Sounds for which the vocal cords are vibrating are called voiced sounds. All vowels are voiced (unless whispered), but also some consonants.

For example, the words "Roman", "yellow" and "wiring" are composed entirely of voiced sounds.

A second important mechanism of speech production is turbulence caused by a constriction in the oral cavity. Voiceless sounds such as in the words "flat" and "sound" are the result of this turbulence. Both mechanisms can occur simultaneously, as happens in the words "voice" and "zip".

Now some classes of speech sounds will be mentioned. Vowels are voiced and are produced without any constriction in the oral cavity. The nasal tract often is closed; if it is open the vowel is called *nasalised.*

Vowels can be further subdivided into so-called pure vowels, which can be generated without a movement in the vocal tract, and *diphthongs*, which are a combination of two vowels. For diphthongs, the vocal tract changes from the position corresponding to the first vowel to the position corresponding to the second vowel, as takes place e.g. in the words "say", "boy" or "new".

Consonants are always produced with a narrowing of the vocal tract. Nasal consonants arise when only the nasal tract is open, as occurs in the words "man", "him" and "wing" [18][25][32][35].

If both the oral and the nasal tract are closed, no air can flow from the lungs. The pressure increases and when the constriction is suddenly opened, sounds called *plosive consonants* or *stop consonants* are formed. They can be both voiced ("by", "day", "go") or unvoiced ("pi", "to", "kiss").

Fricative consonants are produced due to a turbulent airflow at a constriction and also may be voiced ("voice", "zoo", "that") or unvoiced ("fit", "see", "thin").

This survey is not complete: there are still more types of sounds, such as *glides* ("you","we"), semi-vowels ("ray", "lay") and affricatives ("chew", "jar"). However, the main categories are covered.

2.4 A Simple Speech Production Model

The two main mechanisms of speech production, i.e. voiced excitation and turbulent airflow through a constriction, can be captured in a simple speech production model. In this model, speech is assumed to be formed by the excitation of the vocal tract by either a periodic pitch-pulse sequence for voiced speech or a noise-like signal for unvoiced speech. This model is shown schematically in figure 2.2. In the figure, the pitch pulses are shown as vertical arrows, but in reality they are roughly triangular in shape and there spectrum decays with about 12 dB per octave. The influence of the lip radiation is approximately a 6 dB per octave increase in the spectrum with higher frequencies. The net result is that for voiced speech, the spectrum has a tilt and decreases with about 6 dB per octave. This speech model may be a realistic model for the production of many sounds, although for e.g. plosives it will be of limited validity.

In Linear Predictive Coding (LPC) algorithms, the influence of the pulse shape, the vocal tract and lip radiation are combined into one filter. The coding algorithm has to provide the synthesis filter with a proper excitation. LPC algorithms differ in the way the excitation is found and in the number of bits that are spent on it. Next, an overview of some important speech coding algorithms is presented. Most of them use Linear Prediction.

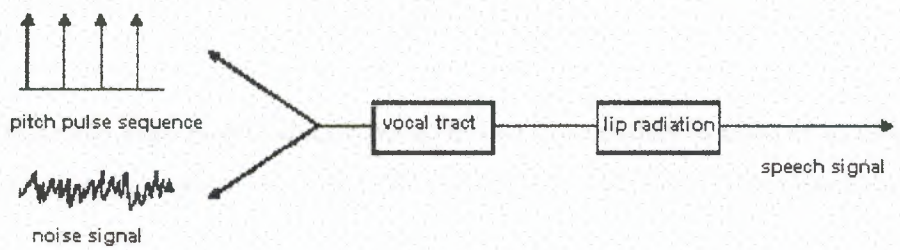


Figure 2.2 Simple speech production model. Speech is assumed to be generated by a spectral shaping of either a periodic pitch pulse sequence or a noise excitation. [25].

2.5 Speech Coding Algorithms

2.5.1 Pulse Code Modulation

Pulse Code Modulation (PCM) is the simplest of all coding algorithms. It does not assume or use any speech production mechanism and it does not use Linear Prediction. In fact, PCM is just a digital representation of the original analog signal: the signal is sampled and each sample is quantised with a fixed number of bits. In A-law and μ-law PCM, the quantisation steps are not of equal size but the quantiser characteristic is roughly logarithmic. This leads to a higher quality than a uniform characteristic (steps of equal size) and has the additional advantage that the quantiser is less sensitive to large variations is signal level.

For telephone applications, a sampling frequency of 8 kHz is normally used and 8 bits per sample. Hence the bitrate is 64 kbit/s.

2.5.2 Adaptive Differential Pulse Code Modulation

PCM does not make use of correlation or dependencies in the signal. Exploiting these redundancies can largely increase the coding efficiency. One way to exploit redundancy is to use Linear Prediction (LP). In LP a prediction $\hat{s}(n)$of a signal s(n) is made on the basis of a weighted sum of preceding signal values:

$$\hat{s}(n) = -\sum_{i=1}^{P} a_i s(n-i) \tag{2.8}$$

The minus sign is introduced because this convention is used in autoregressive literature.The *prediction error* is the difference between prediction and predicted signal:

$$e(n) = s(n) - \hat{s}(n) \tag{2.9}$$

If the prediction is good, the variance of e(n) is much smaller than the variance of s(n). Therefore, if e(n) and s(n) are quantised with the same number of bits, and the quantisation step size is adapted to the variance of the signal, then the absolute quantisation error in e(n) is much smaller than in s(n), although the signal to quantisation noise ratio is the same for both signals. If the prediction error e(n) is

quantised in a coder, the decoder must make a reconstruction $\tilde{s}(n)$ on the basis of this quantised prediction error $\tilde{e}$:

$$\tilde{s}(n) = -\sum_{i=1}^{p} a_i \tilde{s}(n-i) + \tilde{e}(n) \tag{2.10}$$

The reconstruction error (s(n)- $\tilde{s}(n)$) is not equal to the quantisation error (e(n)- $\tilde{e}$ (n)), because $\tilde{s}(n)$ not only depends on $\tilde{e}$ (n) but also on previous values of the reconstruction.

Therefore, a propagation of quantisation errors occurs. The quantisation of the prediction error, without taking into account the propagation of quantisation errors, is called *open_loop quantisation.*

A way to circumvent the propagation of quantisation errors is to use *closed_loop quantisation* of the prediction error, that is; a feed_back loop is put around the quantiser, as is depicted in figure 2.3. In this figure, the Linear Prediction polynomial A(z) is defined by [5][10][15][16]:

$$A(z) = 1 + \sum_{i=1}^{p} a_i z^{-i} \tag{2.11}$$

where z is the forward time-shift operator, e.g. z s(n)=s(n+1). This way of coding is known as Differential Pulse Code Modulation (DPCM). A closed_loop configuration has the advantage that no propagation of quantisation errors occurs because the *prediction* $\hat{s}(n)$ is made on the basis of previous *reconstructed* values:

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

$$\tilde{s}(n) = \hat{s}(n) + \tilde{e}(n) \tag{2.12}$$

In contrast, in an open_loop configuration, the prediction (2.8) at the encoder is made on the basis of the original signal. The reconstruction error in closed-loop DPCM is equal to the quantisation error in the prediction error:

$$s(n)-\tilde{s}(n)=s(n)-\hat{s}(n)-\tilde{e}(n)=e(n)-\tilde{e}(n) \tag{2.13}$$

Both for an open_loop and a closed_loop configuration, the optimal predictor depends on the signal characteristics. For an open_loop configuration, the optimal predictor by definition minimises the variance of e(n) in (2.9) and depends only on the signal characteristics. For a closed_loop configuration the optimal predictor depends both on the signal and on the quantiser, because predictions are made on the basis of the reconstruction.

In speech, the characteristics of the signal are changing with time. To maintain a good prediction, the predictor has to be adapted to the signal. DPCM with adaption of the predictor is known as Adaptive DPCM (ADPCM). The adaption of the predictor can be performed in two ways: *forward adaption* or *backward adaption*. In forward adapting schemes the parameters of the predictor are obtained from blocks of the input signal. For speech signals an updating interval of about 10-30 ms is appropriate, because the signal can be considered more or less as stationary on such intervals. In backward adapting schemes the parameters are obtained in an adaptive manner from the reconstructed signal. An advantage of backward adaption is that no bits have to be spent on coding of the predictor parameters, because both the encoder and decoder use the same reconstructed signal to obtain the predictor. Another advantage is that the coding delay will be smaller because only one sample or a small number of samples of the reconstructed signal is needed for the adaption, instead of a whole block as is the case in forward adapting schemes.

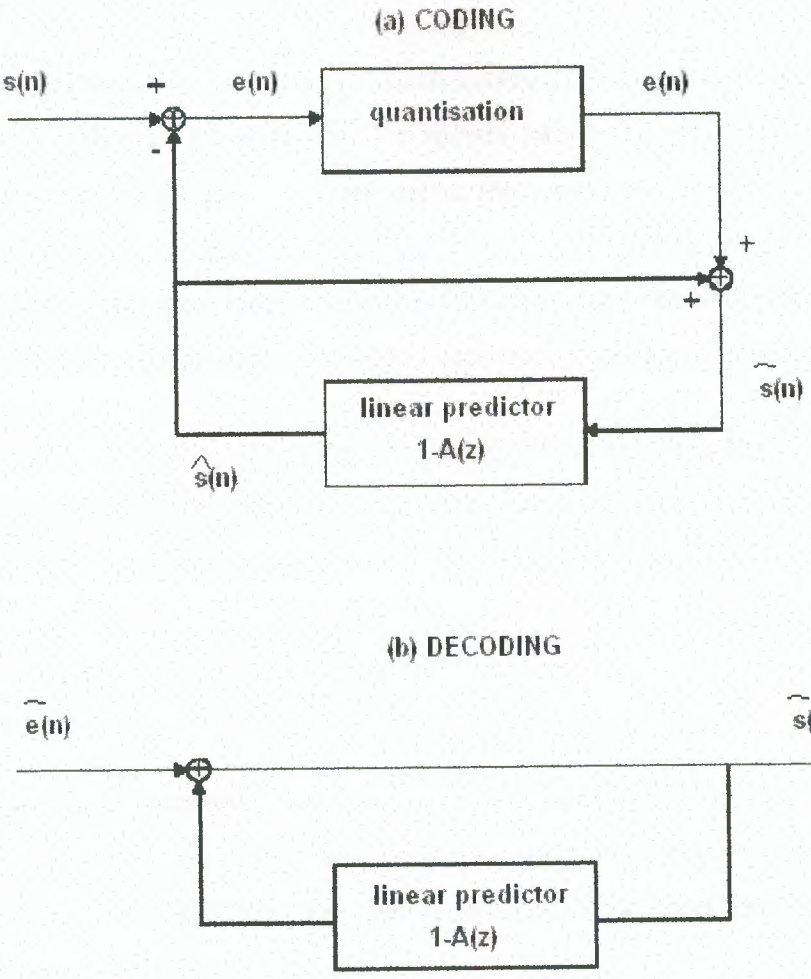


Figure 2.3 Differential Pulse Code Modulation (DPCM) scheme. DPCM uses Linear Prediction and a closed-loop quantisation of the prediction error. In Adaptive DPCM the LPC parameters are updated at regular intervals and are also sent to the decoder. [14].

Disadvantages of backward adaption are that the quality of the predictor is lower, because it has to be obtained from the reconstructed speech which contains coding noise. Therefore, more bits are needed for quantisation of the prediction error to obtain the same quality as for forward adaption.

A typical bitrate for an ADPCM scheme is 32 kbit/s with about the same quality as A-law or μ-law PCM.

2.5.3 Adaptive Predictive Coding

The predictor order p used in predictive coding schemes for speech is usually not very large. A value of 10 is typical for speech sampled at 8 kHz. Therefore, only correlation over small distances in time, called *short term correlation*, is exploited. The voiced speech signal is quasi-periodic due to the pitch pulse excitation and therefore also has a correlation over longer distances, which is not exploited when the predictor order is small. This correlation over longer distances is called *long term correlation.*

In Adaptive Predictive Coding (APC) both short term and long term correlation are exploited with a linear predictor. An APC scheme is shown in figure 2.4. The long term predictor or *pitch predictor* polynomial has the following form:

$$P(z) = 1 + \sum_{i=-j}^{+i} b_i 2^{-M-i} \tag{2.14}$$

where M corresponds to the pitch period in samples. Usually, one or three tap pitch predictor filters are employed. The parameters of the pitch prediction filter are usually updated at a higher rate than the parameters of the short term prediction filter, for example every 5-10 ms.

2.5.4 Analysis-by-Synthesis Coding

Accurate quantisation of the parameters of the short and long term prediction filters can be performed with roughly about 80 bits per 25 ms, which comes down to an average bit rate of less than one half bit per sample for a sampling frequency of 8 kHz.

For the scalar quantisation of the prediction error in APC schemes at least one bit per sample is needed.

The majority of the bits is therefore spent on quantisation of the prediction error signal. The bit rate for quantisation of the error signal can be lowered significantly by coding the error signal in blocks, i.e., applying a form of VQ. The vector length cannot be taken too large, because in that case the complexity becomes a problem. The quantised vector of prediction errors is used in the decoder as an excitation for the long and short term synthesis filters.

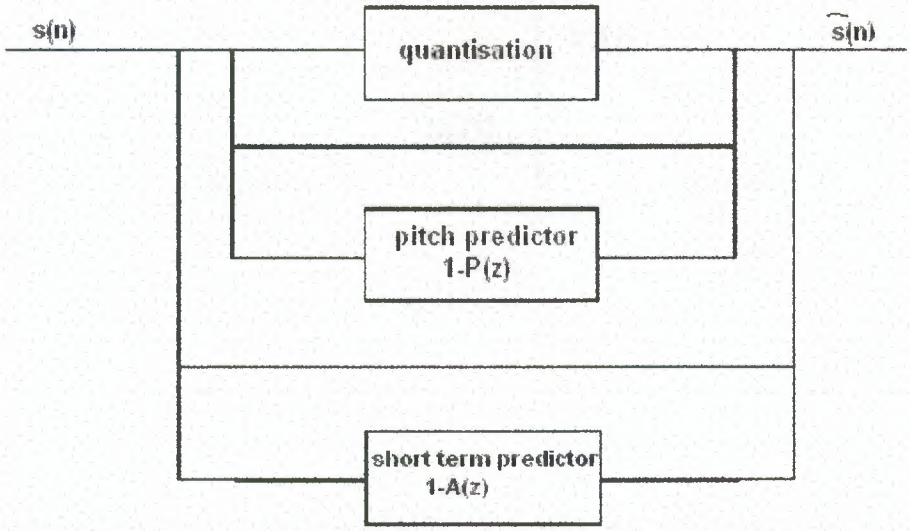


Figure 2.4 Adaptive Predictive Coding (APC) scheme. APC is similar to ADPCM (figure 2.3), but a pitch predictor is added to remove the long term correlation. [14].

Therefore, the coding of the vector of prediction errors is called *vector excitation coding*.It is not recommendable to code the excitation vector on the basis of a direct comparison with candidate excitation vectors, e.g., with a mean squared error measure, because quantisation errors will accumulate in the reconstructed signal, as was mentioned earlier in section 2.5.2 after (2.10). A much more efficient way to code the excitation is *Analysis-by-Synthesis*.

The Analysis-by-Synthesis scheme is shown in figure 2.5. Most high-quality low bit rate LPC coders use Analysis-by-Synthesis for coding the excitation. In Analysis-by-Synthesis, a candidate reconstruction is synthesised for each candidate excitation vector. The choice of a certain excitation vector is made on the basis of the error between original signal and reconstruction [21][30][34].

2.5.5 Error Weighting

In Analysis-by-Synthesis coding, the error between candidate reconstructions and original signal is the basis for a selection criterion. The error is not used directly but is spectrally shaped with a *perceptual weighting filter* W(z). The excitation with the smallest weighted error is chosen. The weighting filter makes use of the frequency domain masking properties of the human auditory system.

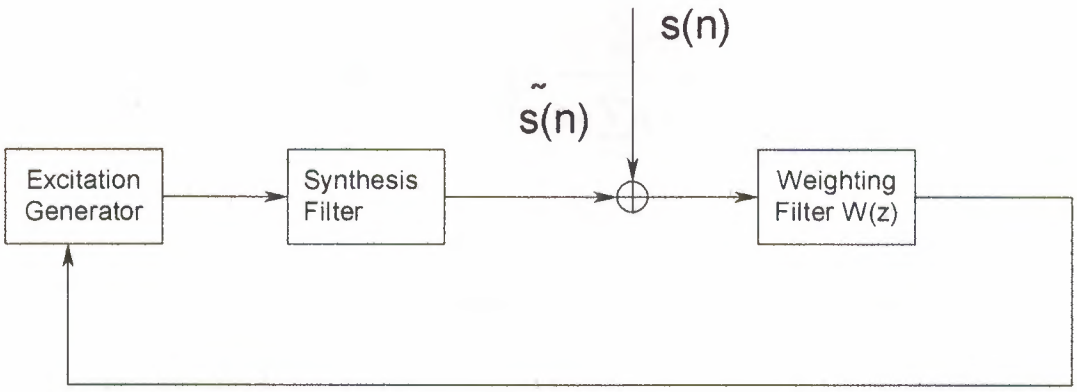


Figure 2.5 Analysis-by-Synthesis coding scheme. In Analysis-by-Synthesis a reconstruction is made with each candidate excitation. That excitation is selected that gives the lowest weighted reconstruction error. [14].

For example, large peaks in the spectrum of a signal may mask nearby weaker tones so that they are not audible. If the bit rate is high enough, the coding noise in the reconstruction is approximately white and has a flat spectrum. The weighting filter shapes the noise in such a way that more coding noise is put under the peaks in the spectrum where it can be masked. A simple and effective way to find a weighting filter is to derive it directly from the LPC filter. The LPC filter is regularly adapted to the signal and its parameters are obtained by applying a standard autoregressive estimation method. A property of autoregressive modelling is that the model describes the spectral envelope of the signal. This property will be explained in chapter 3 where autoregressive modelling and estimation will be discussed, but now it suffices to merely state it. The LPC synthesis filter has the following transfer function [13]:

$$\frac{1}{A(z)} = \frac{1}{1+\sum_{i=1}^{p} a_i z^{-i}} \quad , \quad z = e^{jw} \tag{2.15}$$

An appropriate weighting filter W(z) that is often used, has the form [2]:

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 + \sum_{i=1}^{p} a_i z^{-i}}{1 + \sum_{i=1}^{p} a_i \gamma^i z^{-i}} \tag{2.16}$$

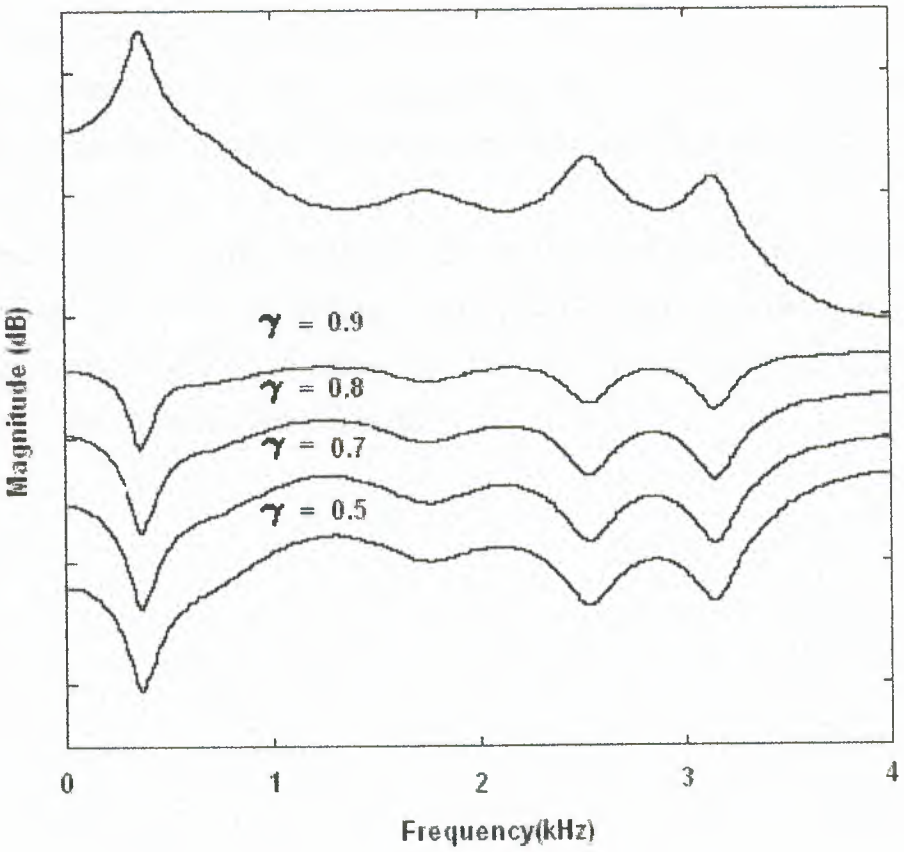


Figure 2.6 LPC model spectrum and perceptual weighting filter spectrum for several values of the perceptual weighting factor γ. The aim of the perceptual weighting filter is to put more coding noise under formants where it may be masked. For the sake of clarity, the spectra are separated by vertical shifts. [28].

Where γ is the *perceptual weighting factor* which has a value between zero and one. A suitable value for the perceptual weighting factor is between 0.8 and 0.9 for a sampling frequency of 8 kHz. The parameters of A(z/ γ) are easily found by multiplying the i_th parameter a_i of A(z) by γ^i as is shown in (2.16). The poles of 1/A(z/ γ) are at the same argument angles in the complex plane as the poles of 1/A(z), but their radii are

multiplied by γ and therefore their bandwidth is expanded. The net effect of the weighting filter is that he frequencies in the signals corresponding to peaks in the spectrum are de-emphasized ring the excitation selection procedure and hence more noise is put in the places where he model spectrum |1/A(z)|^2 is large. The noise is shaped in a perceptually beneficial way. The weighting filter is not applied at the decoder. In figure 2.6 an LPC spectrum is shown, and the spectrum corresponding to the weighting filter W(z) for some values of γ

This kind of error weighting is applied in many LPC coding algorithms. Error weighting decreases the signal to noise ratio somewhat but increases the subjective quality considerably.

The complexity of the Analysis-by-Synthesis excitation selection can be reduced by a rearrangement of the LPC synthesis and weighting filters, as is shown in figure 2.7. This rearrangement has the advantage that the input speech has to be weighted only once prior to the excitation determination.

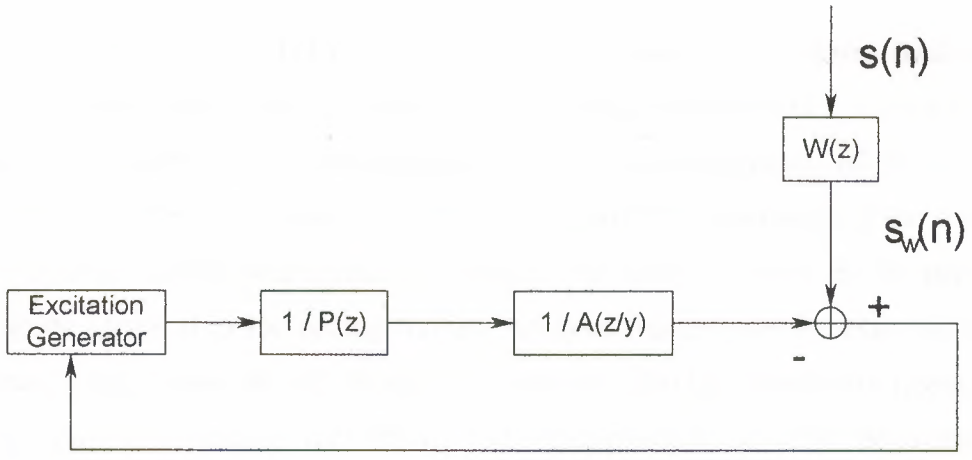


Figure 2.7 The complexity of Analysis-by-Synthesis (figure 2.5) can be lowered by a rearrangement of short term synthesis and weighting filters. In this way, the input speech has to be weighted only once prior to the excitation selection. [14].

2.5.6 Postfiltering

For algorithms operating at relatively high rates, such as ADPCM, the assumption that the coding noise is white is quite accurate (if no noise-shaping is applied). Because the coding noise level is low, a large amount of noise can be masked by the signal by application of a suitable noise-weighting filter. At lower bitrates, a much smaller fraction of the noise can be masked. Moreover, the coding noise can no longer be assumed to be white. This makes error weighting less effective, because the weighting filters assume the coding-noise to be white. In the spectral valleys there will be noise that is audible most of the time. A postfilter can be used to increase the quality at the cost of a decrease in signal to noise ratio. The idea of a postfilter is to enhance the formant peaks of the spectrum of the reconstructed speech with respect to the valleys where most of the audible noise is present. Postfilters are applied only on the reconstruction at the decoder. Postfiltering increases the subjective quality without increasing the bit rate.

2.5.7 Interpolation

LPC based Analysis-by-Synthesis coders operate in a blockwise fashion. The speech is divided into blocks of about 25 ms, called *coding frames*. For each coding frame, the encoder provides the decoder with the coded parameters of short term and long term prediction filters and the coded excitation. The parameters of the short term LPC model are determined in *analysis frames*. The analysis frames do not necessarily have to be identical to the coding frames. Analysis frames may for example overlap, whereas coding frames do not. Analysis frames may also be shifted with respect to the coding frames to reduce the delay. The determination of LPC parameters and excitations is synchronised: each coding frame is divided into a fixed number of subframes, usually four, in which the excitation is determined. This synchronisation ensures that the bit rate is fixed; the same number of bits is sent for every coding frame. Another advantage of synchronising the determination of model parameters and excitations is that the changes in model parameters occur only at subframe boundaries and this is beneficial for the complexity of the selection of the excitation. The parameters of the long term prediction filter are adapted at a higher rate than those of the short term prediction model, for example, every subframe.

It is important for the quality of low bitrate coders that the LPC models vary smoothly with time. Large changes at frame boundaries may give audible distortions. This becomes more important at lower rates, where less bits are available for the excitation to compensate for transition effects. The transition effects can be reduced by applying interpolation of the short term LPC model. LPC models from consecutive frames are interpolated on a subframe basis. A suitable transformation of the LPC parameters is made, and this transformation is linearly interpolated. Several transformations are introduced in chapter 3, interpolation increases the quality without increasing the bit rate.

It is possible to interpolate the pitch predictor parameters as well. For this application, a so-called *fractional delay pitch predictor* is more suitable. A fractional delay pitch predictor has only one tap. The pitch delay is determined from an upsampled version of the signal. This gives an increased resolution which is beneficial because the pitch period is never exactly equal to an integer number of samples. The pitch delay is specified in terms of a number of samples of the upsampled signal, and may contain a non-integer number of samples of the original signal. Next, some analysis-by-synthesis coding algorithms are discussed. They differ in the way the candidate excitations are generated, and in bit rate.

2.5.8 Multi-Pulse and Regular-Pulse Excitation Coding

In this section the multi-pulse and regular pulse excitation coders are described. In the multi-pulse excitation coder, the excitation is represented by just a small number of pulses at non-regular intervals. The amplitudes and the pulse locations have to be coded. About 5 pulses per 5 ms are needed for an acceptable quality. Finding the optimal combination of pulse locations and amplitudes with Analysis-by-Synthesis is a very complex problem. Therefore, suboptimal procedures are often used where the pulse locations and amplitudes are found one at a time. Multi-pulse coders operate at a bitrate of about 16 kbit/s.

In the regular-pulse excitation coder, the pulses are uniformly spaced. Hence, only the position of the first pulse and the amplitudes of all pulses have to be coded. For a certain position of the first pulse, the amplitudes of all pulses are found by solving a linear set of equations. About 10 pulses per 5 ms are needed for a good quality.

A version of the regular-pulse excitation coder is recommended by the "Groupe Spéciale Mobile (GSM)" for digital cellular radio in Europe.

2.5.9 Code-Excited Linear Prediction

A successful speech coding technique for low bitrates is Code-Excited Linear Prediction (CELP). This technique yields good speech quality at bitrates of about 4-8 kilobits per second for a sampling frequency of 8 kHz. The basic scheme of the coder is shown in figure 2.8.

The parameters of short term prediction filter are obtained directly from the input speech signal with a standard autoregressive estimation method. This is an open-loop method. The parameters of the pitch predictor can also be obtained directly from the speech signal. A more efficient way is to use a closed-loop Analysis-by-Synthesis method to determine the parameters of the long term predictor. With this method, the parameters of the long term prediction filter are obtained from the speech signal and a selected part of the most recent past excitation. All possible candidate pitch periods in a specific range are considered.

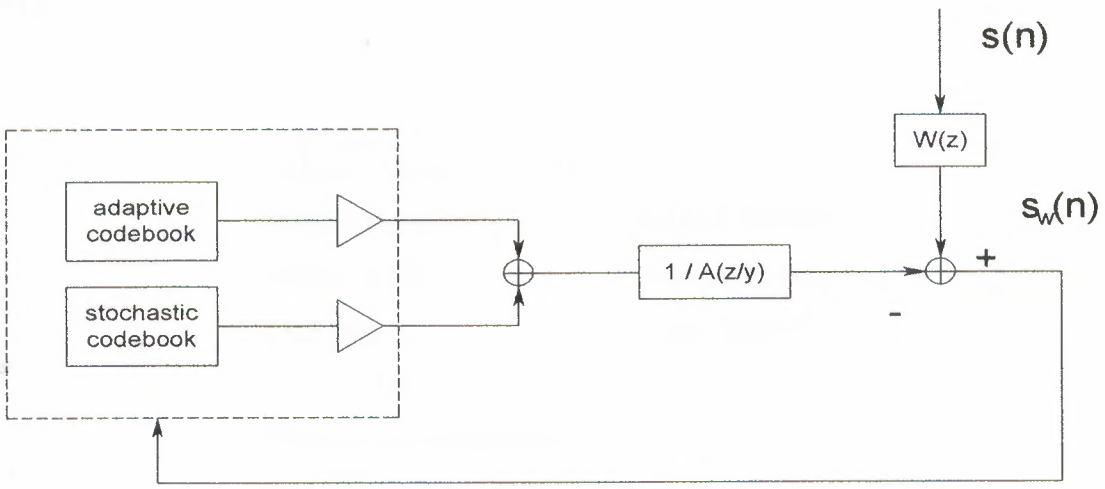


Figure 2.8 Code-Excited Linear Prediction (CELP) scheme. CELP is an Analysis-by-Synthesis coding algorithm. The pitch prediction filter is usually implemented as an adaptive codebook. Furthermore, a fixed 'stochastic' codebook is used. [14]

This range is typically between 20 and 147 samples. If the candidate pitch period is M, a part of length M of the most recent past excitation is used in the Analysis-by-Synthesis procedure. The past excitation is also available at the decoder. If the candidate pitch period M is shorter than the subframe length, the considered part of the past excitation has not a sufficient length. In this case, the last M values of the considered past excitation are periodically repeated up to a length equal to the subframe length. This structure is called an *adaptive codebook* and it is equivalent to a filter if the pitch period is longer than the subframe length. The parameters of the long term prediction filter, which are the gains of the adaptive input, are updated every subframe. M denotes what part of the past input is used as an excitation for A(z). In order to make a reconstruction, the decoder needs the quantised parameters of A(z), the value of M, the energy of the current frame and the adaptive and "stochastic" excitations and gains. The stochastic excitation is called this way because it is selected from a fixed codebook of noise-like excitation vectors, known both at the encoder and decoder. This codebook of noise-like signals is called stochastic codebook. The code number of the winning excitation in the codebook and its gain are sent to the decoder.

An example of how the frame length and bit allocation may be chosen is given below. This is the bit allocation that is used in the U.S. Department of Defense 4.8 kbit/s standard [29]:

Federal Standard FS1016	
Sample frequency:	8 kHz 4,800 bits/s
Frame length:	30 ms / 240 samples
4 subframes	bits/frame
LPC model	34
Adaptive codebook index	28
Adaptive codebook gain	20
Stochastic codebook index	36
Stochastic codebook gain	20
Error correction and sync.	06 +
Total	144

In CELP the LPC parameters, the adaptive excitations and the stochastic excitation are determined sequentially. Although CELP achieves high quality speech at low bitrates, its sequential procedure is certainly not optimal in terms of SNR. The optimal procedure would find the best of all possible combinations of parameters and adaptive and stochastic excitations. Finding the very best combination is impossible in practice because the complexity is enormous.

2.5.9.1 Complexity Reduction

Analysis-by-Synthesis coding of the excitation is very complex because all candidate excitations (adaptive and stochastic) have to be filtered before they can be compared with the (weighted) speech signal. Several complexity reduction techniques are proposed in literature.

The adaptive codewords have a large overlap, because they consist of parts of the recent past excitation. Consecutive adaptive codewords for which the corresponding candidate pitch period is larger than the subframe length, have all but the first and last sample in common. If the candidate pitch period is smaller than the subframe length, there is less similarity because parts of different lengths of the past excitation are then periodically repeated, but consecutive adaptive codewords still have many samples in common. This can be used to reduce the complexity considerably. If the filtered version of one codeword is known, the filtered version of the next codeword can be obtained efficiently with some simple end-point corrections. Fast search methods for obtaining the stochastic excitation can also be developed in the autocorrelation domain or in the frequency domain. The stochastic codebook may also be overlapping or have another special structure. For example, the codebook may be *sparse* or *centre-clipped*. In a sparse codebook a large fraction of the excitation samples is zero. In a centre-clipped codebook the non-zero values are equal to either plus or minus one. For example, the Department of Defense 4.8 kbit/s standard [29] uses an overlapping, sparse, centre-clipped stochastic codebook.

The complexity of CELP coders may be further reduced by the application of an alternative error weighting filter proposed in 1992. This weighting filter is obtained by replacing the denominator $A(z/\gamma)$ of $W(z)$ in (2.15) by $H(z/\gamma)$, where $H(z)$ is a *pre-emphasis* filter (this is a first order high-pass filter):

$$H(z) = 1 - \mu z^{-1} \quad , \quad \mu = \frac{\hat{R}(1)}{\tilde{R}(0)} \tag{2.17}$$

R(0) and R(1) are sample autocorrelation coefficients of the speech data (see chapter 3. The complexity reduction comes from the fact that the filtering action is now obtained by a more simple linear recursion.

2.5.10 The Vocoder

The name Vocoder (for *Voice Coder*) is a generic term for coding systems in which the excitation and vocal tract transfer functions are treated separate. A Linear Prediction filter may be used for the synthesis. In the simplest form, the excitation consists of a periodic pitch pulse sequence for voiced speech and a noise signal for unvoiced speech.

The Vocoder, however, is not able to code nonstationary parts of the speech signal, like transition segments, with sufficient quality. The excitation coding of CELP is much more flexible and can cope with these segments. The decoder needs the coded LPC parameters, an unvoiced-voiced decision bit, the energy of a frame and the pitch period in the case of voiced speech. It may operate at a bit rate of 2400 bits/s or even lower, but the coded speech is not of very high quality.

2.5.11 Improved Quality at Lower Rates

There is continuous progression towards good quality at lower rates. One possible approach is to adapt the coding method to the properties of the signal. The Analysis-by-Synthesis schemes that were described use the same algorithm and number of bits, independent of the type of signal under analysis. However, unvoiced speech needs far less bits for an acceptable quality than voiced speech, and the main reason is that no pitch predictor is needed in unvoiced speech. Non-speech segments, where only background noise is present, need even less bits. Adaption of the coding method to the signal characteristics leads to coders with a varying bit rate.

In *phonetic segmentation* speech segments are classified into phonetically distinct categories and the coding mechanism and bit rate are tailored to each class. In

this way the average bitrate can be lowered to about 3 kbit/s with a quality at least as good as the 4.8 kbit/s U.S. Federal Standard 1016 (CELP) algorithm.

Another promising approach is *generalised Analysis-by-Synthesis coding*. The idea is that not the speech signal itself is coded, but a modified version which sounds the same. This modified version has the property that it can be coded more efficiently than the original signal. Generalised Analysis-by-Synthesis can be used to improve the efficiency of the pitch predictor.

Improved efficiency may also be obtained by using more extensive interpolation techniques: only parts of the signal are coded and missing parts are synthesised by interpolation. These techniques are applied only to voiced speech and hence the coders need an unvoiced/voiced decision.In 1995 the Kleijn and Haagen have presented a coding algorithm which avoids the unvoiced/voiced decisison. This coder also uses Linear Prediction and performs at least as well as the U.S. Federal Standard 1016, but at only half the bitrate.

2.6 Quality Measures

2.6.1 Objective Quality Measures

An objective quality measure that is used in many fields of signal processing is the *Signal-to- Noise Ratio (SNR)*. It is defined as the ratio of the signal power to the (coding) noise power, expressed in decibels:

$$SNR = 10^{10} \log\left\{\frac{\sum_n s^2(n)}{\sum_n \{s(n) - \tilde{s}(n)\}^2}\right\}(dB) \qquad (2.18)$$

The SNR has several disadvantages for speech coding. One important disadvantage is that the SNR may be determined mainly by the segments with the highest energy and the influence of, for example, important transition segments may be underestimated. The SNR can be improved somewhat by computing the SNR over short segments of, say, 15 ms, and averaging these SNR values. In this way the *Segmental SNR (SSNR)* is obtained. SSNR and SNR can also be computed in the weighted domain,

i.e. computed after filtering signal and reconstruction with the weighting filter W(z) of (2.16). At low bit rates SNR and SSNR do not predict accurately the subjective quality of speech. Several measures have been developed which better predict the performance of speech and audio coding systems.

2.6.2 Subjective Quality - The Mean Opinion Score

The most often used subjective quality measure for high quality low bit rate coding systems is the *Mean Opinion Score (MOS)*. The MOS is determined by formal listening tests where experienced listeners rank the reconstructed speech with a value between one and five. The average of all listener scores for all speech data is the MOS. The values of MOS mean the following:

1 bad 2 poor 3 fair 4 good 5 excellent

High quality Analysis-by-Synthesis coders have MOS scores between 3 and 4. A MOS score of 4 is considered as near transparent quality over a telephone line.

2.7 Summary

In this chapter the various coding algorithms has been given. Only algorithms have been considered that use Linear Prediction and waveform matching: the error between original and reconstructed signal (in the weighted domain) is minimised. This means that phase information is taken into account. The predictor is an all-pole (autoregressive) predictor. In principle, a pole-zero (autoregressive-moving average) predictor could be used or a non-linear predictor, but it is not yet clear if these predictors will improve the coding quality. However, autoregressive-moving average predictors can be applied successfully in speech synthesis.

Two alternative approaches at low rates are the *Multi Band Excitation (MBE)* coder and the *Sinusoidal Transform Coder (STC)*. In the MBE algorithm, the excitation is divided into several frequency bands and in each band an unvoiced/voiced decisison is made. In the unvoiced bands, phase information is discarded. MBE coders operate at rates of 2.4-4.8 kbit/s. In STC, the synthesised speech is a sum of sinusoidal signals. The frequencies, amplitudes and phases are efficiently coded.

3. AUTOREGRESSIVE MODELLING OF SPEECH

3.1 Overview

In the previous chapter several speech coding algorithms based on linear prediction have been described. An advantage of linear prediction is that the model has a frequency domain interpretation. This frequency-domain interpretation makes it possible to use techniques such as error weighting and postfiltering and to use objective measures with a frequency domain interpretation for the evaluation of the quality of quantised or interpolated models. For high quality coding, the accurate estimation, interpolation and quantisation of the models is very important. Some of the questions involving these issues can be answered by using properties of model parameters that follow from autoregressive theory. In this chapter a short overview of autoregressive theory is given. Furthermore, it is shown that the well known autocorrelation method for estimation of the parameters is not a useful method. Its shortcomings can be cured to a large extent by the application of tapered data windows, but other methods are available that do not need a window and even perform better without it in stochastic signals.

3.2 Autoregressive Estimation

3.2.1 Estimation Methods

In a K-th order autoregressive process the signal x(n) is described by a weighted sum of preceeding signal values plus an independent identically distributed noise signal $\varepsilon(n)$ with variance.

$$\varepsilon(n) = x(n) + \sum_{i=1}^{K} a_i x(n-i) \tag{3.1}$$

The coefficients a_i are the autoregressive parameters, called LPC parameters in speech coding.

The best known autoregressive estimation methods are the autocorrelation or Yule Walker method, the covariance or one-sided least squares method, the modified covariance or two-sided least squares method and the Burg method. All these methods estimate the parameters from N samples of a signal.

The autocorrelation method assumes the signal to be exactly zero outside the interval of observation and estimates the AR parameters of a p-th order model by minimising the residual sum of squares $s^2_{[P]}$ of the forward residuals from minus to plus infinity:

$$s^2_{[P]} = \frac{1}{N}\sum_{n=-\infty}^{\infty}\{x(n)+a_1x(n-1)+.....+a_px(n-p)\}^2 \tag{3.2}$$

This is equivalent to the minimisation of the following expression:

$$s^2_{[P]} = a^T\hat{R}a \tag{3.3}$$

where **a** = [1 a ... a] is the vector of AR parameters to be found. The elements R(i,j) of the *autocorrelation matrix* $\hat{R}$ are the *autocorrelation coefficients* of the data and are defined by:

$$\hat{R}(i,j) = \frac{1}{N}\sum_{n=-\infty}^{\infty}x(n-i)x(n-j) = \hat{R}(|i-j|) \tag{3.4}$$

Because of the infinite sum, these autocorrelation coefficients are dependent only on the absolute value of the difference between i and j, i.e. $\hat{R}(i,j) = \hat{R}(|i-j|)$. This means that the autocorrelation matrix $\hat{R}$ in (3.3) has a persymmetric Toeplitz structure for the autocorrelation method. This structure allows the parameters to be found efficiently from the autocorrelation coefficients with the Levinson-Durbin algorithm. The Levinson-Durbin algorithm transforms an autocorrelation function R(k) of a process to the autoregressive parameters of that process:

$$k_m = -\frac{R(m) + \sum_{j=1}^{m-1} a_j^{[m-1]} R(m-j)}{s_{[m-1]}^2} \quad , \text{m=1,.....,p} \tag{3.5}$$

$$s_m^2 = s_{m-1}^2 \left(1 - k_m^2\right) \tag{3.6}$$

$$a_j^{[m]} = a_j^{[m-1]} + k_m a_{m-j}^{[m-1]} \quad , \quad a_m^{[m]} = k_m \tag{3.7}$$

Where k_m is the m-th *reflection coefficient*, $a_j^{[m]}$ is the j-th parameter of an m-th order model and s_m^2 is the residual variance for the m-th order model. The reflection coefficient k_m can be interpreted as the negative of the *partial correlation coefficient* [6][27][31] between x(n-m) and x(n). The partial correlation coefficient between two random variables which both are correlated with a third random variable is defined as the correlation between them after the correlation with the third one has been removed from both of them. A reflection coefficient k_m can thus be interpreted as the negative of the correlation between two samples x(n-m) and x(n) of a time series when the correlation with the samples x(n-m+1) ... x(n-1) has been removed. The reflection coefficients owe their physical names to an acoustic tube model of the vocal tract. They describe the reflection coefficients for the forward and backward travelling waves in that model.

The $\hat{R}$ (k)'s in the autocorrelation method are biased estimates of the theoretical autocorrelation coefficients R(k) of the process, because $\hat{R}$ (k) contains only N-k non-zero terms, but is normalised by 1/N.

The bias in the sample autocorrelation coefficients for the autocorrelation method is caused by the way this method handles edge effects. It will be shown that the edge effects in the autocorrelation method give poor results if there are reflection coefficients present in the process close to plus or minus one.

The covariance method uses only data within the segment to minimise the residual sum of squares of forward residuals:

$$\sigma_{[p]}^2 = \frac{1}{N-p}\sum_{n=p+1}^{N}\{x(n)+a_1x(n-1)+.....+a_px(n-p)\}^2 \qquad (3.8)$$

The modified covariance method minimises the sum of squares of forward and backward residuals within the segment:

$$\sigma_{[p]}^2 = \frac{1}{N-p}\sum_{n=p+1}^{N}\{x(n)+a_1x(n-1)+.....+a_px(n-p)\}^2 +$$

$$\frac{1}{N-p}\sum_{n=1}^{N-p}\{x(n)+a_1x(n+1)+.....+a_px(n+p)\}^2 \qquad (3.9)$$

The residual sums of squares (3.2), (3.8) and (3.9) are estimates of the innovation variance σ^2.

The equations (3.8) and (3.9) can be expressed in the same form as (3.3), but the definition of $\hat{R}$ for each method is slightly different. The corresponding matrices $\hat{R}$ are symmetric but not Toeplitz.

Another well-known method is the Burg method. The Burg method does not estimate AR parameters with (3.3), but this method estimates reflection coefficients directly and uses the Levinson recursion (3.7) to obtain AR parameters. The reflection coefficient $\hat{k}_m$ is the negative of the correlation coefficient of the forward and backward residual signals $\hat{e}_{m-1}^f$ and $\hat{e}_{m-1}^b$, respectively, of the (m-1)-th order model [12]:

$$\hat{k}_m = \frac{-2\sum_{n=m+1}^{N}\hat{e}_{m-1}^f(n)\hat{e}_{m-1}^b(n-1)}{\sum_{n=m+1}^{N}\left((\hat{e}_{m-1}^f(n))^2+(\hat{e}_{m-1}(n-1))^2\right)} \qquad (3.10)$$

$$\hat{e}_m^f(n) = \hat{e}_{m-1}^f(n)+\hat{k}_m\hat{e}_{m-1}^b(n-1) \qquad n = m+2,.....,N$$

$$\hat{e}_m^b(n) = \hat{e}_{m-1}^b(n-1)+\hat{k}_m\hat{e}_{m-1}^f(n) \qquad n = m+1,.....,N-1 \qquad (3.11)$$

where $\hat{e}_0^f(n) = \hat{e}_0^b(n) = x(n)$.

The estimate of the residual variance and the parameters are found subsequently with (3.6) and (3.7). The modified covariance method also uses forward and backward residuals, but not the Levinson recursion. The Burg method is a constrained minimisation of the residual sum of squares (3.9) in the sense that the autoregressive predictor parameters of the (m-1)-th order model are assumed to have been estimated already. The parameters of the m_th order model are obtained with the Levinson algorithm (3.7) from the (m-1)-th order parameters and $\hat{k}_m$. The modified covariance method finds parameters by an unconstrained minimisation of (3.9).

These four methods differ only in the way the edge effects are handled. If p zeros are added to the observation interval both at the begin and at the end, these four methods are *identical*.

3.1.2 Stability

In speech coding it is necessary that the LPC models are stable, i.e., the poles of the autoregressive transfer function are within the unit circle. This is because otherwise the output of the synthesis filter may diverge at the decoder. Even if Analysis-by-Synthesis is used for coding of the excitation this can occur because errors in the received bits due to a poor channel can cause the decoder to use an excitation different from the one that was intended. Furthermore, Analysis-by-Synthesis is generally applied in the weighted domain, while no weighting filter is applied for reconstruction at the decoder.

The bias in the sample autocorrelation function for the autocorrelation method ensures that the models obtained with this method are always stable. The covariance method and the modified covariance method do not guarantee a stable model, although instability will occur less frequently with the modified covariance method than with the covariance method.These methods are therefore not suitable for use in a speech coder.

There exist some modifications of the covariance method that do ensure a stable model [2].

A convenient stability criterion exists for the reflection coefficients. If the absolute value of all reflection coefficients is smaller than one, the model is guaranteed to be stable. From (3.10) it follows that the reflection coefficients as they are computed in the Burg method satisfy this stability criterion.

3.2 Asymptotic Theory

The four main estimation methods that have been described in the previous section differ in the way edge effects are handled. If the model order p is small enough relative to the number of observations N used to estimate the parameters, and the poles of the process are not too close to the unit circle, all methods perform similarly. In asymptotic theory no difference is made between estimation methods. The residual sums of squares are estimates of the innovation variance σ^2.

The expectation of the residual sum of squares is called the *residual variance*. The residual variance is the expectation of the minimum of (3.3) and equals (1-p/N) σ^2 asymptotically; slightly smaller than σ^2. This is only true if the model order p is higher than or equal to the true process order K. If parameters that are found by minimising the residual sum of squares for one set of observations, are substituted in another independent set of observations of the same process, the "residuals" are called *prediction errors*. If, for a given estimated parameter vector, the expectation of the prediction error variance is taken for independent sets of observations, the result is a measure for the quality of the estimated parameters. This measure is called *Prediction Error (PE)*. It can be computed by substituting in (3.3) the true autocorrelation matrix of the process, because the parameters are assumed to be obtained from another independent set of observations:

$$PE = \hat{a}^T R \hat{a} \tag{3.12}$$

If the model order is higher than or equal to the process order, the expectation of the prediction Error over the estimated parameters equals (1+p/N) σ^2; slightly larger than σ^2. In chapter 2, no distinction has been made between residuals and prediction errors, because it was not discussed in much detail there how the predictor parameters are obtained. One application where it is essential to make a clear distinction between residuals and prediction errors is order selection. In that case, also differences between estimation methods must be taken into account. In speech coding, differences between estimation methods are also important, because a stable model is necessary. The expressions for the variance of the residuals and the prediction errors are accurate when the model order p/N is small and when there are no poles that are too close to the unit

circle. If there are reflection coefficients close to plus or minus unity, the autocorrelation method may suffer from undesirable edge effects. In section 3.3 these edge effects will be investigated.

In (3.12) the parameters are assumed to be estimated from a realisation of the process with correlation matrix **R**. Often a quality measure is needed where no assumptions are made about the source of the parameters. For instance, when one has a model that has to be quantised, the parameter vector $\hat{a}$ can be a test vector from a code book. Then the matrix **R** is the autocorrelation matrix of the process that is described by the reference model, i.e., the autocorrelation matrix corresponding to the model that has to be quantised. The *Likelihood Ratio (LR)* is such a quality measure, defined by:

$$LR = a^T R_{p+1} a \tag{3.13}$$

R_{p+1} is the (p+1)×(p+1) correlation matrix of the reference process, normalised by σ^2 Likelihood Ratio can also be expressed as follows [3]:

$$LR = 1 + \Delta a^T R_p \Delta a \tag{3.14}$$

Where $\Delta a = \lfloor \Delta a ..... \Delta a^T \rfloor$ is the vector of differences of the parameters of test and reference models. If the parameter vector **a** is obtained by estimation from a realisation of the reference process, the Likelihood Ratio is equal to PE/σ^2, and its expectation is 1+p/N. The Likelihood Ratio can also be expressed in the frequency domain:

$$LR = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{\hat{A}(e^{jw})}{A(e^{jw})} \right|^2 dw \tag{3.15}$$

where $A(e^{jw}) = 1 + a_1 e^{jw} + a_2 e^{2jw} + ... + a_p e^{pjw}$, and $\hat{A}(e^{jw})$ is defined in a similar way. The spectrum of the autoregressive process is given by σ^2 /|A(e)| . From these expressions an important conclusion can be drawn: if the model order p is relatively small in comparison with the number of observations used to estimate the parameters, but high enough to catch all relevant parameters, an autoregressive model can give an accurate description of a signal in both the time and frequency domain. An accurate spectral description means a small prediction error in the time domain, and vice versa.

In speech coding algorithms at high rates, such as ADPCM, use is made of the good predicting capabilities of the autoregressive model. Algorithms at lower bit rates use both the good predicting capabilities (for Analysis-by- Synthesis excitation coding) and the spectral interpretation (for error weighting, postfiltering and quantisation).

3.2.1 Spectral Distortion Measure

The Likelihood Ratio is an LPC distortion measure that is used frequently in speech processing. There are many other LPC distortion measures. Perhaps the most often used distortion measure is the *Spectral Distortion (SD)*. The Spectral Distortion is defined in the frequency domain as a squared relative distortion measure of the LPC log-spectra:

$$SD = \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\log\frac{\left|\hat{A}\left(e^{jw}\right)\right|^2}{\left|A\left(e^{jw}\right)\right|}\right|^2 dw} \tag{3.16}$$

It can be expressed in the time domain as an infinite sum as follows:

$$SD = \sqrt{\sum_{-\infty}^{\infty}\{c_k - \hat{c}_k\}^2} \tag{3.17}$$

Where c_k and $\hat{c}_k$ are the *cepstral coefficients* corresponding to A and $\hat{A}$, respectively. These cepstral coefficients can be computed recursively from the LPC parameters. Spectral Distortion is usually expressed in decibels. The value in dB can be obtained by multiplying (3.16) or (3.17) with 10 log e.In 1976 it has been state that for values of Spectral Distortion smaller than about 2 dB, Spectral Distortion and Likelihood Ratio are for all practical purposes identical [23]:

$$SD^2 \approx 2(LR-1) = 2\Delta a^T R_p \Delta a \tag{3.18}$$

Spectral Distortion is often used to express the quality of a quantiser of the LPC model. An accurate quantisation of the LPC model is of much importance for the

quality of low bit rate speech coders. Current research aims at *transparent quantisation*, that is; the effects of quantisation are not audible.

3.2.2 Other Representations

The Levinson-Durbin algorithm transforms an autocorrelation function to the autoregressive parameters. The reflection coefficients show up as an intermediate set of coefficients. The parameters, the reflection coefficients and the autocorrelation function (normalised with respect to σ^2) are one-to-one transformations of each other and all describe the same model.

Such a one-to-one transformation of the parameters which completely describes the model is called a *representation* of the model. The cepstral coefficients are also a representation of the LPC model. To compute the LPC coefficients from the autocorrelation function, only R(0) through R(p) are needed. These are the elements of the first row (or column) of $\mathbf{R}_{p+1}$ in (3.13). This representation will be denoted COR. How COR can be computed from the LPC parameters is described in [1]. The normalised autocorrelation function (NCOR) is obtained by dividing COR by R(0). NCOR is also a representation of the model. Although different representations describe the same model, their properties are quite different. The specific properties of representations are of great importance for their interpolation and quantisation behaviour. Other representations of interest are the *Log Area Ratios (LARs)* l_i and the *Arcsine of Reflection Coefficients (ASRCs)* m_i [24]. These are scalar transformations of the reflection coefficients and are defined as follows:

$$l_i = \log\frac{1+k_i}{1-k_i} \quad , \qquad m_i = \sin^{-1} k_i \quad , \qquad i = 1,\ldots\ldots, p \tag{3.19}$$

A uniform scalar quantisation of the LARs or the ASRCs is similar to a non-uniform quantisation of the reflection coefficients with more closely spaced levels for values of the reflection coefficients near plus or minus one. This is necessary because quality measures such as Likelihood Ratio and Spectral Distortion become more sensitive to changes in reflection coefficients if these are close to one in absolute value. These *Reflection Coefficient based representations* have been used frequently in the

past for scalar quantisation because stability is easily guaranteed, in contrast to, e.g., the LPC parameters or the (normalised) autocorrelation function.

Another representation which has been applied successfully for both quantisation and interpolation is formed by the *Line Spectrum Frequencies (LSFs)*. It is defined as the locations of the zeros of two polynomials. These polynomials are formed by adding to an LPC polynomial an extra reflection coefficient equal to +1 or -1, respectively. Consider an LPC polynomial $A_p(z)$. It can be factorised as p a product of first order polynomials:

$$A_p(z) = 1 + a_1 z^{-1} + ... + a_p z^{-p} = \prod_{i=1}^{p} \left(1 - q_i z^{-1}\right) \tag{3.20}$$

The coefficients q_i are the zeros of $A_p(z)$. From this expression it is readily seen that the last parameter a_p , which is equal to the last reflection coefficient, is the product of all zeros:

$$a_p = \prod_{i=1}^{p} q_i \tag{3.21}$$

When the artificially an extra reflection coefficient, k_{p+1} , is added to $A_p(z)$, with a value equal to plus or minus one. The result is two polynomials of order p+1: an antisymmetric polynomial P(z) for k_{p+1} =-1 and a symmetric polynomial Q(z) for k_{p+1} =+1. The product of all zeros of these polynomials is equal to the extra reflection coefficient, and it follows that all zeros of P(z) and Q(z) are located on the unit circle. The frequencies corresponding to the locations of the zeros are the LSFs. The term "Line Spectrum" stems from the fact that $|1/P(e^{jw})|$ and $|1/Q(e^{jw})|$ describe line spectra in the frequency domain. It can be shown that P(z) and Q(z) can be written for even order LPC polynomials as:

$$P(z) = (1 - z^{-1}) \prod_{i=1}^{p/2} (1 - 2\cos w_{2i} z^{-1} + z^{-2}) \tag{3.22}$$

$$Q(z) = (1 + z^{-1}) \prod_{i=1}^{p/2} (1 - 2\cos w_{2i-1} z^{-1} + z^{-2})$$

where the w_i, i=1,2,...,p, are the Line Spectrum Frequencies. The LSFs can be defined in a similar way for p odd, but the polynomials P(z) and Q(z) then look slightly different. A simple stability criterion for the polynomial $A_p(z)$ can be expressed in terms of the LSFs. $A_p(z)$ is stable if and only if the following *ordering property* is fulfilled:

$$0 < w1 < w2 < ..... < w_p < \pi \tag{3.23}$$

Another name that is often used for the LSFs is *Line Spectrum Pairs (LSPs)* because the even and odd LSFs are obtained from the coefficients of a pair of antisymmetric and symmetric polynomials. Another property of LSFs is that if there is a peak in the spectrum $1/|A(e^{jw})|$, then in the vicinity of this peak LSFs tend to be close together. The smaller the bandwidth of the peak, the closer the LSFs are. Figure 3.1 shows an example of an LPC spectrum and the corresponding LSFs.

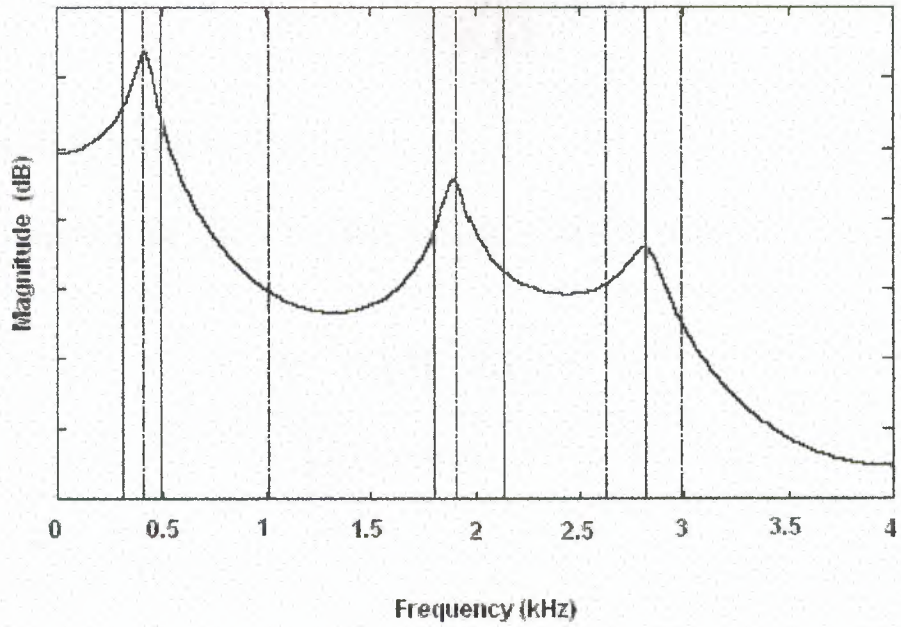


Figure 3.1 LPC spectrum and corresponding Line Spectrum Frequencies (shown as vertical lines). [17].

Another property of LSFs is that a small change in one of the LSFs gives a change in the autoregressive spectrum mainly in the neighbourhood of that LSF. This is known as the *localisation property*. Similar properties hold for odd order polynomials.

A representation which is closely related to the LSFs is formed by the *Immitance Spectral Pairs (ISPs)*. The ISPs of a p-th order autoregressive polynomial $A_p(z)$ are formed by a 'gain' and p-1 frequency parameters. They are defined in the following way. For autoregressive polynomials, the following recurrent relation exists:

$$A_p(z) = A_{p-1}(z) + k_p z^{-1} B_{p-1}(z) \tag{3.24}$$

$$B_p(z) = z^{-1} B_{p-1}(z) + k_p A_{p-1}(z)$$

where $B_p(z)$ is equal to $z^{-p} A_p(z^{-1})$ and a *Immitance function* $I_p(z)$

$$I_p(z) = \frac{A_p(z) - B_p(z)}{A_p(z) + B_p(z)} \tag{3.25}$$

and give the following stability theorem:

$A_p(z)$ is stable if and only if $I_p(z)$ can be written, when p=2m, as:

$$I_{2m}(z) = \frac{K(1 - z^{-2}) \prod_{i}^{m-1} (1 - 2\cos_{2i} z^{-1} + z^{-2})}{\prod_{i=1}^{m} (1 - 2\cos w_{2i-1} z^{-1} + z^{-2})} \tag{3.26}$$

and when p=2m+1 as:

$$I_{2m+1}(z) = \frac{K(1 - z^{-1}) \prod_{i=1}^{m} (1 - 2\cos_{2i} z^{-1} + z^{-2})}{(1 + z^{-1}) \prod_{i=1}^{m} (1 - 2\cos w_{2i-1} z^{-1} + z^{-2})} \tag{3.27}$$

with p parameters that are real and satisfy:

$$0 < w1 < w2 < ..... < w_{p-1} < \pi \quad , \quad k > 0 \quad , k_p = \frac{K - 1}{K + 1} \tag{3.28}$$

The polynomials that occur in this stability theorem are very similar to the polynomials in (3.22) for the LSFs. It can have shown that the w_i in (3.28) are the LSFs of a (p-1)-th order model. In other words, the ISP frequency parameters for $A_p(z)$ are actually the LSFs for A (z). Together with K, the transformation in (3.28) of the p-th reflection coefficient k_p of A_{p-1}(z), they form a representation of the autoregressive parameters. It is clear that ISPs and LSFs are closely related.

3.2.3 Statistics

In this section the statistics of different LPC representations are considered. These statistics will turn out to be of importance for the interpolation and quantisation behaviour of the representations

3.2.3.1 Covariance matrices in AR processes

Consider the covariance matrix of estimated AR parameters :

$$C_a = \Im\{(\hat{a} - \Im\hat{a})(\hat{a} - \Im\hat{a})^T\} \tag{3.29}$$

where $\Im$ is the expectation operator and $\hat{a} = \lfloor \hat{a}_1 \hat{a}_2 ..... \hat{a}_p{}^T \rfloor$ is the vector of estimated parameters.

Asymptotic autoregressive theory states that the covariance matrix C_a of the estimated parameters is given by:

$$C_a = \frac{1}{N} R_p{}^{-1} \tag{3.30}$$

where N is the number of observations that is used to estimate the parameters and R_p is the p×p autocorrelation matrix of the AR process, normalised by σ^2. R_p can be computed recursively from the true parameters of the process. Expressions in a closed form exists for the elements of C_a.

Asymptotically, the theoretical covariance matrix C_λ for stationary AR processes of a different representation λ can be computed from that of the LPC parameters:

$$C_\lambda = L C_a L^T \tag{3.31}$$

The symbol denotes a transformation of the LPC parameters, such as reflection coefficients, LARs, LSFs, etc. The (i,j)-th element of the (p×p) matrix **L** is the partial derivative of the i-th coefficient λ_i with respect to the j-th LPC parameter.

The theoretical covariance matrix of estimated reflection coefficients can be found with a recursive procedure. The covariance matrices of LAR and ASRC follow from that of the reflection coefficients by using the matrix of partial derivatives of the LARs and ASRCs with respect to the reflection coefficients. This matrix is a diagonal matrix, because the LARs and ASRCs are scalar transformations of the reflection coefficients.

A general property of the theoretical covariance matrix of estimated reflection coefficients is that all estimated reflection coefficients of the orders greater than or equal to the true process order K, are uncorrelated with all other estimated reflection coefficients of the orders below K. In other words, for a K-th order process, the covariance between the i-th estimated reflection coefficient, $\hat{k}_i$, and the j-th estimated reflection coefficient, $\hat{k}_j$, is always zero it either i or j is greater than or equal to K (and i and j are different). The situation is different for the lower order estimated reflection coefficients: if there are true reflection coefficients close to plus one or minus one, the lower estimated reflection coefficients can have large variances and covariances, in contrast to the LPC parameters.

For example, for third order processes the variances of the first estimated reflection coefficient, $\hat{k}_1$, and the first estimated LPC parameter, $\hat{a}_1$, are equal to:

$$\operatorname{var}\hat{k}_1 = \frac{1}{N}\frac{\left(1-k_1^{\,2}\right)\left(1-k_2\right)^2\left(1+2k_1k_3+k_3^{\,2}\right)}{\left(1-k_3^{\,2}\right)\left(1-k_2^{\,2}\right)} \tag{3.32}$$

$$\operatorname{var}\hat{a}_1 = \frac{1}{N}\left(1-k_3^{\,2}\right)$$

Theoretically, the variance of $\hat{k}_1$ may become very large, but the variance of $\hat{a}_1$ is always smaller than 1/N.

Due to their large variances and covariances, estimated reflection coefficients may differ much from their theoretical values, but still yield good models in terms of Likelihood Ratio. This is also true for LAR and ASRC. Despite their names, the *estimated* coefficients of the *Reflection Coefficient* based representations have therefore no physical meaning in the presence of large reflections. This is one of the reasons why the diameters of the sections in an acoustic tube model cannot be found directly from the speech signal with an autoregressive model.

Symmetry relations may be used to show that even and odd LSFs are mutually uncorrelated when estimated from stationary autoregressive processes. If the polynomials P(z) and Q(z) in (3.22) are written in the same form as $A_p(z)$, then the parameters follow from the Levinson recursion (3.7) with k_{p+1} equal to plus or minus one. In this way two sets of equations are obtained connecting the LSFs w_i to the LPC parameters a_i. From these relations it can be readily seen that the LSFs are functions of the parameters of P(z) and Q(z), which themselves are differences and sums of the LPC parameters:

$$P(z) = a_j^{[p+1]} = a_j^{[p]} - a_{p+1-j}^{[p]}$$

$$Q(z) = a_j^{[p+1]} = a_j^{[p]} + a_{p+1-j}^{[p]} \tag{3.33}$$

The even LSFs are functions of the differences of LPC parameters and the odd LSFs of the sums of LPC parameters only. Therefore, the following symmetry relations hold for the elements of the LPC to LSF derivative matrix:

$$\frac{\partial w_j}{\partial a_i} = -\frac{\partial w_j}{\partial a_{p+1-i}} \quad \text{, j even}$$

$$\frac{\partial w_j}{\partial a_i} = +\frac{\partial w_j}{\partial a_{p+1-i}} \quad \text{, j odd} \tag{3.34}$$

The covariance matrix R_p of an autoregressive process has a persymmetric Toeplitz structure, so R(i,j)=R(j,i)=R(p+1-i,j)=R(p+1-j,i). These symmetries also hold for the correlation matrix C_a of the LPC parameters. Using these symmetries in C_a and the symmetry relations of (3.34) it follows from (3.31) that the even and odd LSFs are mutually uncorrelated. Analytical computation of the covariance matrix of the LSFs for process orders up to p=6 showed that *all* LSFs are uncorrelated for these orders, i.e., **C** is a diagonal matrix. For higher orders, analytical computation of the covariance matrix is rather straightforward but very tedious. However, numerical computation of the covariance matrix of LSFs for many different processes indicated that LSFs are also uncorrelated for higher order processes. In a similar way it turns out that the ISPs are also uncorrelated.

3.2.3.2 Distributions in speech

When parameters are estimated from a stationary AR process asymptotic theory says that their correlation matrix is given by (3.30) and their mean is given by the true parameters, apart from a bias of order 1/N. In chapter 2 it was mentioned that Vector Quantisation makes use of correlations and dependencies to code random vectors in an efficient way. For coding of the LPC parameters in speech, it is not only the correlations expressed by (3.30) or (3.31) that are exploited. Speech is a continuously changing signal and may at best be considered as a concatenation of stationary pieces of about 25 ms. The correlations as expressed by (3.31) are therefore changing all the time and the poles of the models could, in theory, be located anywhere inside the unit circle. In practice, more models are found in certain areas of parameter space than in others. It is these distributions in speech that are mainly used to code the LPC models efficiently by designing codebooks which put more codevectors in areas which are covered by speech. A problem is that the distributions are different for different languages and speakers and hence it is difficult to obtain efficient and robust codebooks.

3.3 Bias Propagation in the Autocorrelation Method

3.3.1 Introduction

Many low bitrate speech coders use an autoregressive linear prediction model to describe the speech spectral envelope. All standard techniques for estimation asymptotically have a similar performance on stochastic signals. Some well-known methods have been described in section 3.1.1. For small samples, clear differences exist between these methods because the edge effects are handled. It is known that a tapered data window improves the performance of the autocorrelation method in speech analysis. In this section, a time-domain analysis of the autocorrelation method will be presented, that gives a clear insight in the behaviour of this method, with and without windowing.

For *stochastic* signals, the behaviour of estimation methods is expressed in terms of bias and variance of the model parameters. In 1983 they study the asymptotical bias of parameters obtained with different estimation methods by means of asymptotic Taylor expansions. In 1985 it can be show theoretically that for strongly correlated second order autoregressive processes, the estimate of the residual variance for the autocorrelation method is severely biased. They express the bias in terms of the poles of the autoregressive process. In 1988 they give expressions for the bias of the covariance and autocorrelation methods in terms of the parameters. Although these studies describe the bias quite accurately, the formulas do not give much insight in what causes the bias and what are the consequences of it for higher order processes. In this section, it is shown that this bias in the estimate of the residual variance is caused by edge effects and that it occurs if a reflection coefficient is close to one in absolute value. This bias in the residual variance propagates with the Levinson-Durbin recursion to the estimates of higher order reflection coefficients and leads to a large bias in them as well. A tapered data window may improve the performance of the autocorrelation method because it smooths the edge effects. All estimation methods perform similarly with a tapered data window, but for the least squares methods and the Burg method a tapered data window is theoretically superfluous for stochastic signals and even undesirable because the variance of the estimated models is increased.

In 1980 the suitability of the Burg method has been investigated for speech analysis and synthesis applications. Their conclusions, based mainly on *deterministic* signals (responses to periodical pulse sequences), are that the autocorrelation method and Burg method perform very similar with a data window, and slightly better than the Burg method without a window.

It can be report that an LPC-VQ system using the Burg method performs both qualitatively and quantitatively better than similar systems that use the autocorrelation method.

Results of autoregressive estimation methods are dependent on the location of a pulse within the analysis frames. Windowing reduces the sensitivity of estimation methods to the location of the pulses. Hence, for real speech two opposite effects of windowing exist: a deterministic advantage because the sensitivity to the location of pitch pulses is decreased, but a stochastic disadvantage because the variance of the models is increased [22].

3.4 Summary

The autogressive modelling of speech is given in this chapter. The types of autocorrelation method is discussed such as Levinson-Durbin, Burg and Covariance method.

The autocorrelation method of autoregressive estimation is not suitable if reflection coefficients are close to plus or minus one. This result is of practical interest because for sampled band-limited signals the first reflection coefficient is close to minus one unless the sample frequency is low. The poor performance of the autocorrelation method is due to edge effects; incomplete terms in the residual cause a large bias in the residual variance, this bias is propagated via the denominator of the Levinson-Durbin recursion and causes higher order reflection coefficients to be seriously biased as well. A data window decreases the edge effects and reduces the differences between the autocorrelation method and other estimation methods.

4. LINEAR PREDICTIVE CODING

4.1 Overview

In previous chapter, we discuss little bit literature survey of the speech coding. In this chapter, the implementation and of the Linear predictive coding of speech will be given. We will discuss the practical aspects and implementation of the LPC coding.

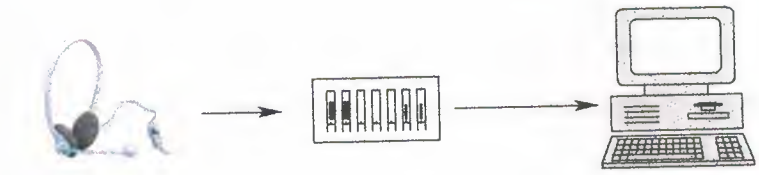
As we know frequency content of typical speech is within the range 300-3300 Hz. According to Nyquist's Sampling Theorem, speech has to be sampled at 6.6 kHz to avoid aliasing and ensure perfect reconstruction. However, 8 kHz has become the standard sampling frequency because it provides for a margin of error.

Uncompressed speech using 8-bit quantization requires a total bit rate of 64 kbits/s. This rate is far too high for most applications and can be reduced considerably through various compression schemes. Among these schemes, linear predictive coding (LPC) has been found to give high compression ratios.

LPC accomplishes high compression by modeling speech as an AR process and encoding the model parameters rather than the speech.

4.2 Speech Signals

In this thesis, I use my own speech signal and other speech signals taken from the internet sites [36]. In order to record and edit of the speech samples, we need a microphone (or sound source such a tape recorder), sound card and sound recorder program. I used Quake SPQ-03V type head-microphone, PCL sound card and a COLEA program to record my speech in wave format in my PC, which is shown in figures 4.1.



Quake SPQ - 03V Headphone

Soundcard CMI8738/c3dx PCL

AMD Athlon 800MHz. PC

Fig 4.1 recording of our speech sample

Then I convert my own speech samples into matlab format by using ' **[Data]=wavread('buraka.wav') ;** ' command.

4.3 Spectral Analysis of Speech Signals

The analysis of the speech signal is always the foundation of related processing techniques. So we first studied the spectral features of speech signals.

4.3.1 Features of Speech Spectrum

Since speech signal is time-varying, the analysis should be a time-frequency analysis. We always assume WSS and hope to take length of samples as long as possible to obtain a low-bias. Besides, bias variance tradeoff is always what we hope to control. In speech signal, however, we have to cut the whole signal into blocks to obtain short time stationary. Typically the block is 20-30 ms long. Short-time FT (STFT) is applied in the spectral analysis for speech.

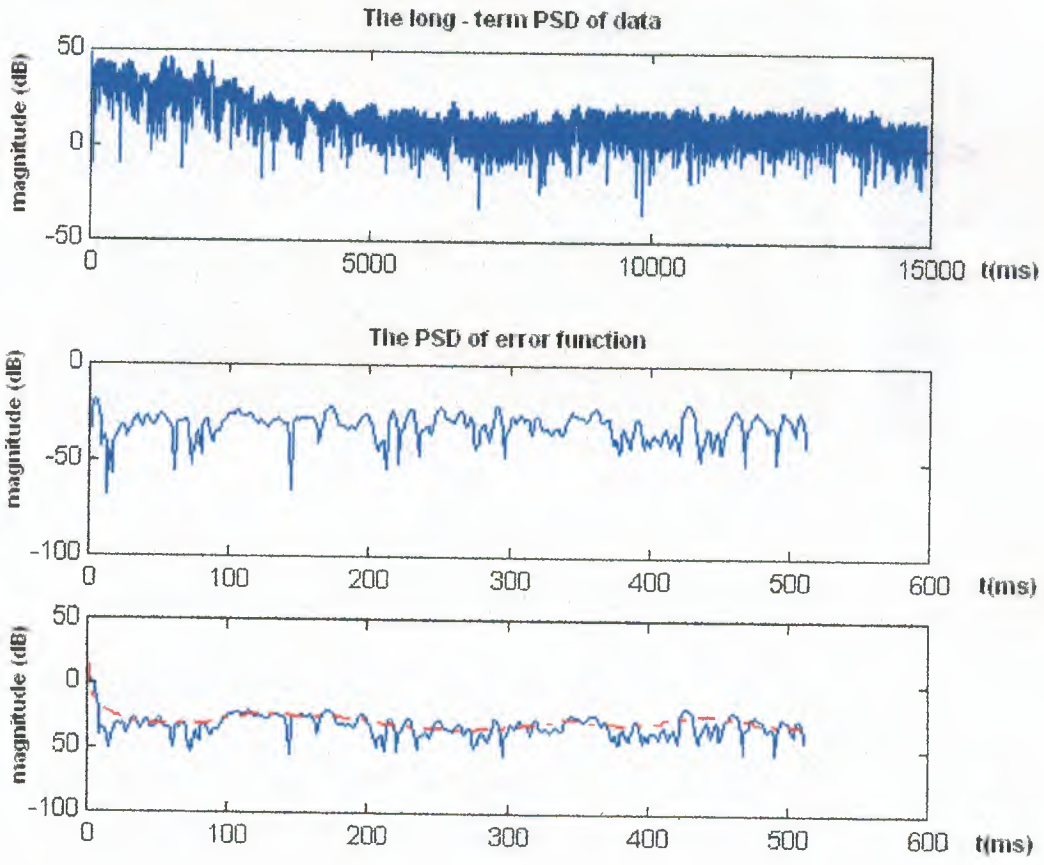


Fig. 4.2 Long-term spectrum vs. Short-time spectrum. (a). shows the long-term spectrum. (b) is the predictor error spectrum; (c) The red line is the AR model spectrum, and the blue line is STFT of one block.

To test this, I did a bunch of experiments to draw the different spectrum of speech. We found from the results that long-term spectrum is non-sense for speech. The short-time spectrum is quite well fitted by AR model and gives us a good structure to represent the signal. Spectrogram is applied to show the time-varying feature of speech spectrum.

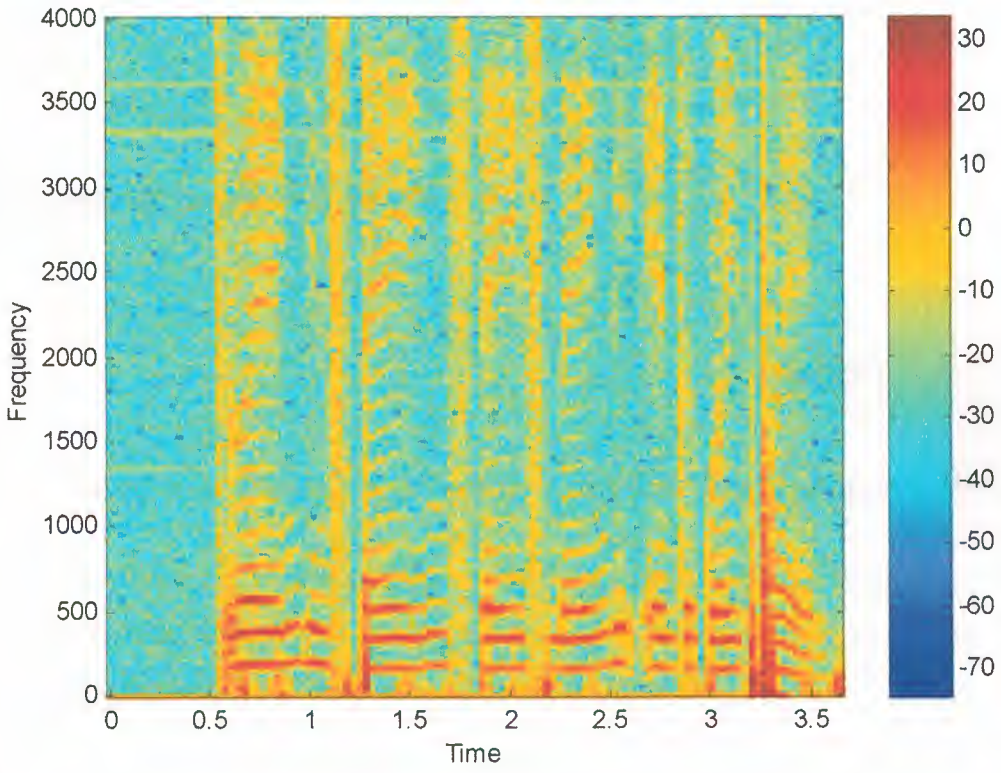


Fig. 4.3 The spectragram style and colorbar of my speech.

4.3.2 Voiced/Unvoiced Spectrum

Speech can be generally divided into voiced and unvoiced. We studied both the spectrum of typical voiced and unvoiced block. The spectrum in Fig 4.4 shows that for a voiced speech, the time series have obvious periodic. The spectrum of voiced speech is featured as some fine spectrum with formant envelope. The fine peaks mean the pitch period and the formants reflect the vocal tract feature. While for the unvoiced case, the

signal looks much like a white noise. The spectra lose the pitch period but keeps some formant peaks.

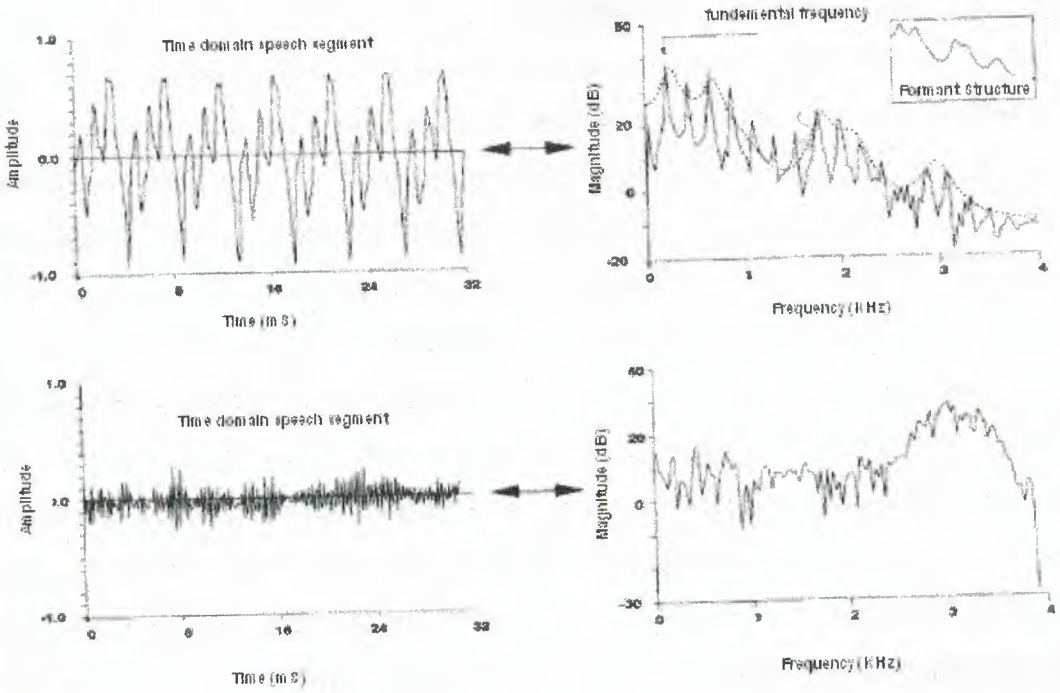


Fig.4.4 Voiced and Unvoiced segments and their short time spectra. [14]

The figure below shows that the AR model fits the speech signal quite well. By passing the speech through a predictor filter A(z), the spectrum is much more flatten (whitened). But it still containes some fine details.

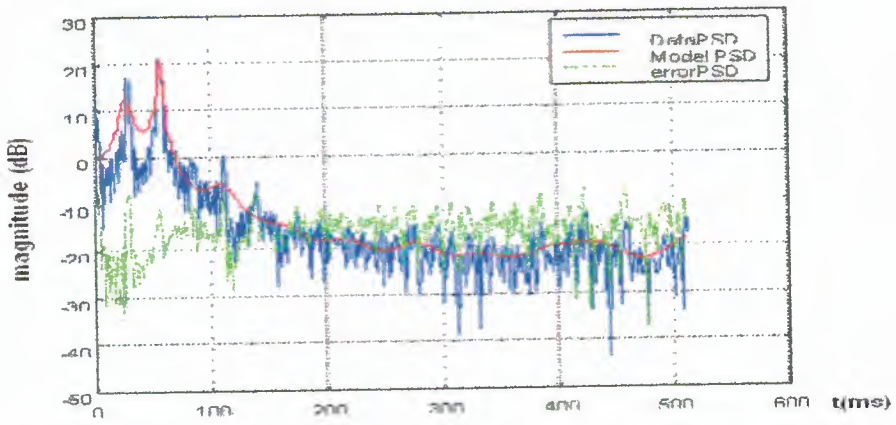


Fig 4.5 Data PSD,Model PSD,Error PSD

4.4 Voice Model

When we speak, we physically change the way our vocal tract (mouth, tongue, and trachea) is shaped. The vocal tract (basically a lossless, non-uniform tube), can be modeled as a linear mechanical system that is (relatively) slowly varying in time. A good model for a lossless tube is an all pole system, in the case of the vocal tract a model order of around ten is appropriate. The input to this system is an excitation signal that depends on what sound we are trying to make.

Our speech is made up of two basic components, voiced (vowels), and unvoiced (consonants). For example, when somebody is saying the word "fast", the 'f' is unvoiced, the short 'a' is voiced, and the 's' and 't' are unvoiced. When we make an unvoiced sound, we constrict part of our vocal tract to a very small passage way, and push air through this small opening. The resulting airflow is very turbulent and is modeled in our system input as white (Gaussian) noise. To make a voiced sound, our vocal cords vibrate against the glottis (entrance to the vocal tract) periodically interrupting the airflow.The vocal tract anatomy is shown in appendix A. This is modeled by input into our system of a series of impulses. The time in between these impulses,called the pitch period,varies with the vowel we are trying to form.Our linear system model of speech is shown in the Figure below.We will make the assumption that over short periods of time (30 ms), our speech is a stationary random process. So, for 30 ms time intervals, we use the speech data to estimate model parameters, and then code these parameters. We also need to code whether the sound made during the 30 ms was voiced (in which case we also need the pitch period) or unvoiced.

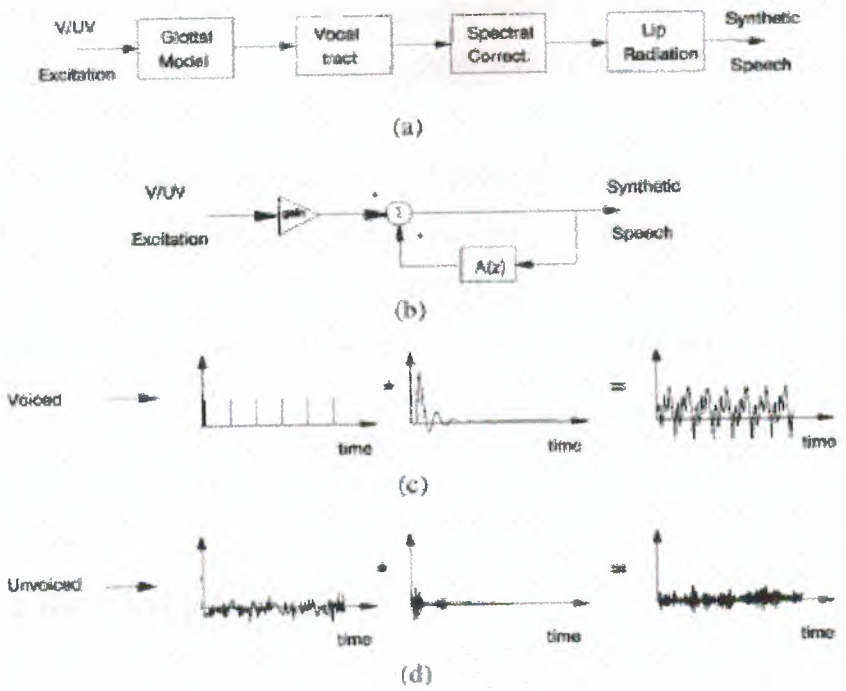


Fig 4.6 Linear speech models and voiced/unvoiced speech representations.(a) Fant's speech production model.(b)All-pole source-system model.(c) Graphical representations of voiced speech production. (d) Graphical representation of unvoiced speech production. [14].

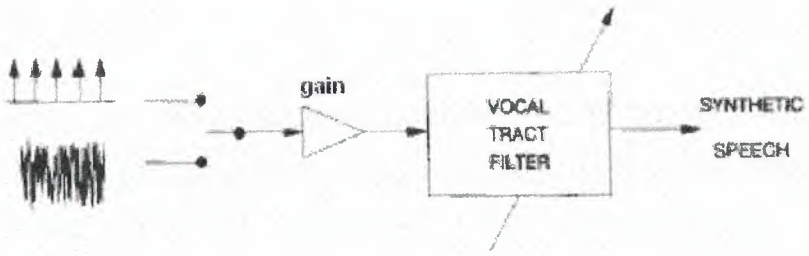


Fig 4.7 The engineering model for speech synthesis. [14]

To reconstruct (synthesize) the speech, either an impulse train (for voiced) or white noise (for unvoiced) was passed through a filter made up from the coded vocal tract parameters.

The model of the vocal tract as an all pole linear system is surprisingly accurate. In fact, LPC works extremely well at very low bit rates for unvoiced signals.However, the quality of LPC for voiced signals is limited due to the approximation of the voiced driving signals. LPC performance depends heavily on the accuracy of the pitch period estimation.

4.5 LPC Coder Architecture

For a two-state LPC vocoder, the system block is almost fixed. What makes difference is the application of different algorithms to do all these parts of work. In brief, there are three big problems, i.e. **(1). linear prediction, (2). voice/unvoiced decision and (3).pitch detection**. So we first set up the framework of LPC coder. Then we focused our efforts on algorithms for each part and carried out features, hoping to derive some insights for future work.

To improve the performance, people will also assume more complex production models(e.g. mixed excitation and residual excitation).

4.5.1 Encoder

Based on the analysis of the previous sections, the system of our two-state LPC coder follows the following block. In the encoder, we first did some kinds of pre-processing which include pre-emphasis, segmentation and windowing. We did Linear prediction analysis to extract the linear prediction parameters for each block, which include reflection coefficients, alfa coefficients, Gain and prediction error. Then we applied some kind of voiced/unvoiced decision algorithm to determine the voice/unvoiced feature and some kind of pitch detection algorithms to extract the pitch period. Before transmission, the LP coefficients, pitch period, gain and uvdecision(one bit) result were encoded. For simplicity, we here only discussed the quantization feature of linear prediction coefficients.

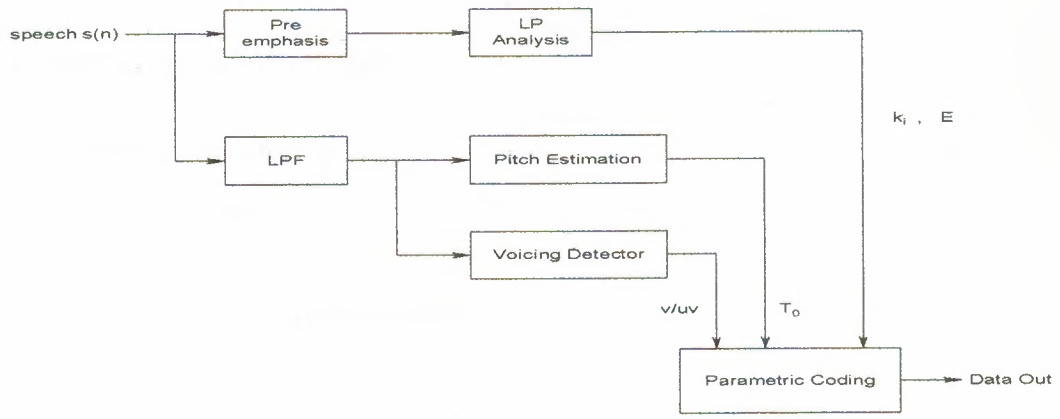


Fig 4.8 LPC Encoder (Federal Standard FS1015). [14].

4.5.2 Decoder

Based on the simplest two state speech production model, the decoder is as the following figure shows. Unvdecision is a switch to determine whether the excitation should be white noise or an impulse train with pitch period. The reflection coefficients were conversed to direct coefficients as the parameters of the all-pole inverse filter. Gain is used to reconstruct the energy of the speech. Then some post-processing procedures were applied to align the output speech.

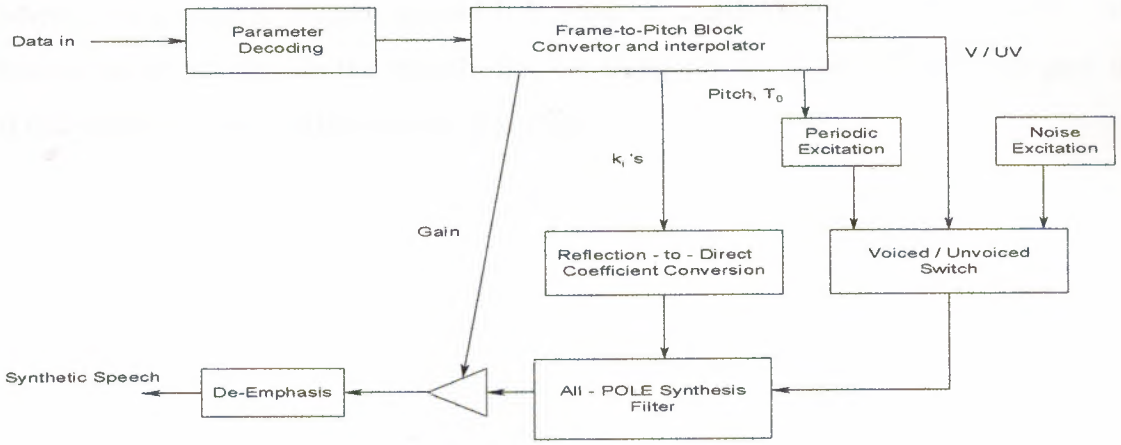


Fig 4.9 LPC Decoder (Federal Standard FS1015). [14].

4.6 LPC Coding implementation

LPC consists of the following steps

1. Pre-emphasis Filtering
2. Data Windowing
3. AR Parameter Estimation
4. Pitch Period and Gain Estimation
5. Quantization
6. Decoding and Frame Interpolation

4.6.1 Pre-emphasis Filtering

When we speak, the speech signal experiences some spectral roll off due to the radiation effects of the sound from the mouth (see the Figure below). As a result, the majority of the spectral energy is concentrated in the lower frequencies. However, the information in the high frequencies is just as important to us understanding the speech as the low frequencies; we would like our model to treat all frequencies equally. To have our model give equal weight to each, we need to apply a high-pass filter to the original signal. This is done with a one zero filter, called the pre-emphasis filter. The filter has the form:

$$y[n] = 1 - a\, x[n] \tag{4.1}$$

where **a** is generally a value around 0.9. Most standards use **a** = 15/16 = .9375. Of course, when we decode the speech, the last thing we do to each frame is to pass it through a de-emphasis filter to undo this effect.

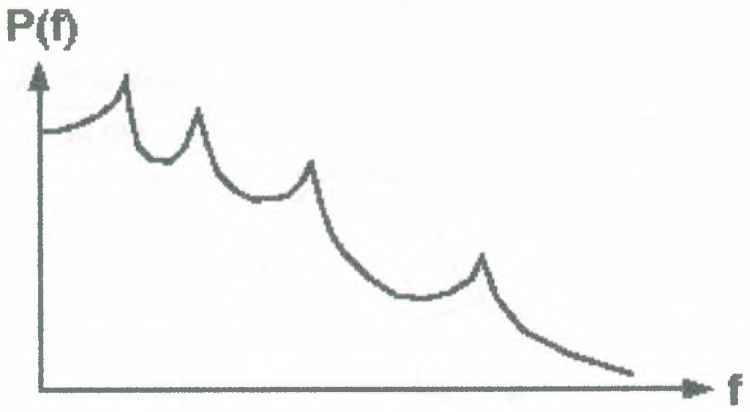


Fig 4.10 Typical spectral envelope of voiced sound

4.6.2 Data Windowing

We will assume that a speech signal is a stationary AR process over a short amount of time. However, to avoid discontinuities in the model, we will use overlapping data frames. As the frame size gets larger and larger, our bit rate gets lower and lower, but of course our assumption of the process being stationary over a frame becomes more and more precarious. For the LPC coder implemented in this thesis, the original speech signals were sampled at 8kHz. The signal was cut into 240 sample 'blocks' which overlapped the previous and succesive blocks by 60 samples. These blocks were Hamming windowed before further processing was done.

Because sometimes it is desirable to window the data to lower the variance of the autocorrelation matrix estimate. The standard bias-variance trade-offs for the AR estimate (autocorrelation estimate) occur for different data windows. In this project we used a Hamming window, as shown in the figure below.

In total, each block represented 180 data points, or 22.5ms. This is referred to as the 'frame period' in LPC coding.

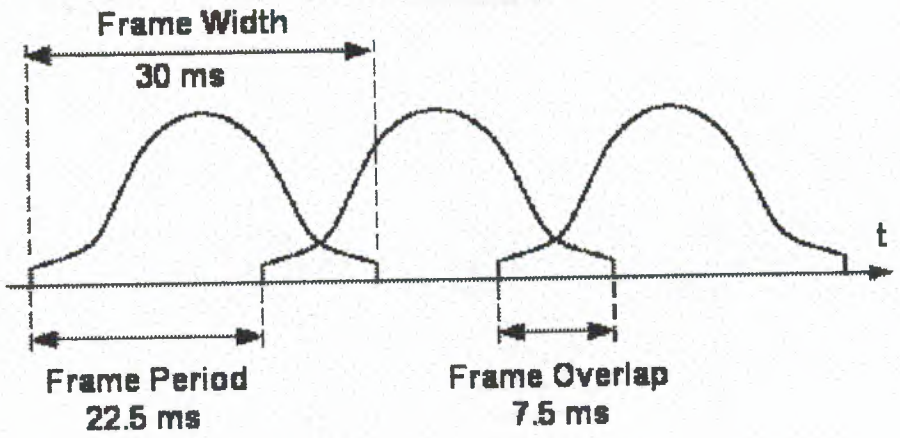


Fig 4.11 Window placement and frame overlapping

4.6.3 Linear prediction (AR parameter estimation)

There are various kinds of formulations for linear prediction problem. Two important algorithms are Levinson Algorithm and Burg Algorithm. Although a bunch of functions to solve AR model problem can be found in matlab (such levinson, ar, lpc,…) the interfaces are always too complex . I also implement my own implementation of algorithms to understand and help for drawing some graphs. In Coder implementation each block of data was modeled as an AR process using the method. If the block was voiced, a tenth order model was used. For unvoiced speech, we used a fourth order model. The AR reflection coefficients were taken instead of the standard 'a' parameters, as this model is more stable when quantized. I hope this study more helpful for the deeper understanding of the literature. Moreover, the expertise built through this work can help us easily write a C or asm language program for real-time application. We also carried out comparisons of these two algorithms.

4.6.3.1 Levinson algorithm

In a p-th order forward linear predictor, the current sample of speech is predicted from a linear combination of the p-past samples. Levinson algorithm is achieved by minimizing the MSE of forward prediction error. And then by applying the Toeplitz

feature of the auto- correlation matrix, a fast algorithm based on order-recursion is derived by levinson. The recursion is formulated as Fig below:

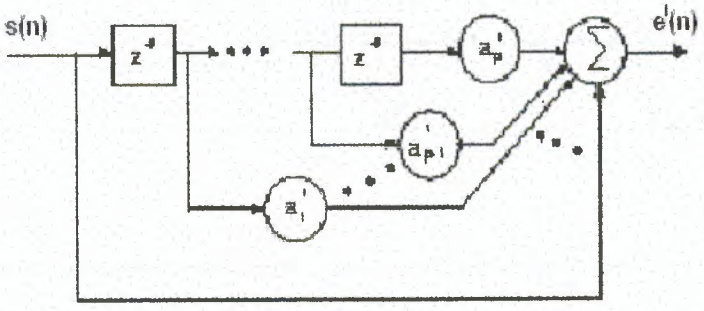


Fig 4.12 Linear prediction realizations, Direct forward LP analysis. [14].

4.6.3.2 Burg Algorithm

Burg algorithm is a kind of forward-backward prediction. Based on the Levinson recursion and this prediction, we derived the following recusive computation:

$$ef_p(n) = ef_{p-1}(n) + k_p \times eb_{p-1}(n-1)$$

$$eb_p(n) = eb_{p-1}(n-1) + k_p \times ef_{p-1}(n) \tag{4.2}$$

In which the reflection coefficients are computed with this expression:

$$\hat{k}_m = \frac{2\sum_{n=m+1}^{N} e_{m-1}^{f}(n) e_{m-1}^{b}(n-1)}{\sum_{n=m+1}^{N} \left|e_{m-1}^{f}(n)\right|^2 + \sum_{n=m+1}^{N} \left|e_{m-1}^{b}(n-1)\right|^2} \tag{4.3}$$

The figure below shows a clear diagram of the recursive computation of Burg algorithm.

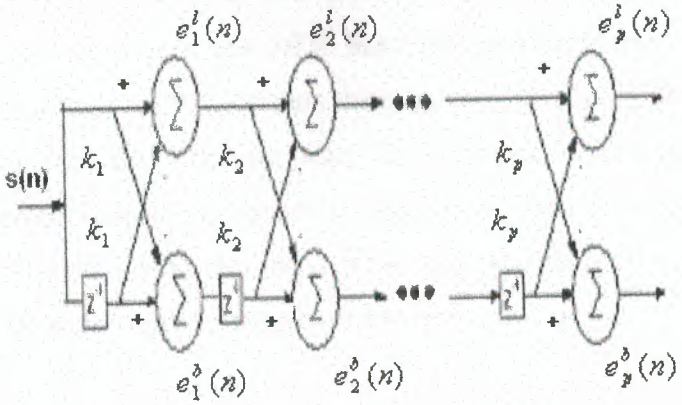


Fig 4.13 Linear prediction realizations, Lattice forward-backward predictor.The input is the speech signal; the output is the residual error. [14].

I carried out a comparison between these two algorithms. Summarily, they have the following differences:

1. Computation form: Note that Levinson algorithm has a direct form while Burg algorithm has a Lattice form. In burg algorithm, the reflection coefficients can be computed directly in different stages without the computation of the prediction filter coefficients. We also highlight here that the lattice form is a representative direction in filtering theory.
2. Although both algorithms guarantee stability in theory, in practice, Levinson is more sensitive than Burg. Some kinds of preprocessing (preemphsizing, windowing) will improve the performance. But for burg, this procedure can be saved.
3. For Levinson algorithm, the computation of Autocorrelation function is required. In burg algorithm, there is no need for this.

4.6.3.3 Order Selection

Order selection is very important for AR model. We did experiments for both levinson and burg with AIC and MDL criterion. AIC is expressed as AIC(k)=Nln(MSE(k))+2*k , MDL is computed as MDL(k)= N* ln(MSE(k))+k*ln(N); The left figure below is the result of levinson and the right burg. Solid line is AIC and dashed line is MDL. From the result, we found that MDL has a clearer valley for both

levinson and burg algorithm. AIC tends to overestimate the order since the valley is not clear even for order 20. This means that MDL has a better performance in the selection of order. This maybe because MDL considers the number of bits in the second term. We can also see clear from the result that an order 10 is enough for voiced while order 4 looks good for unvoiced speech(result not shown here). That is why we can typical LPC coder LPC-10. We also found that the cost function decreases faster than Levinson. This may also support the strength of Burg over Levinson.

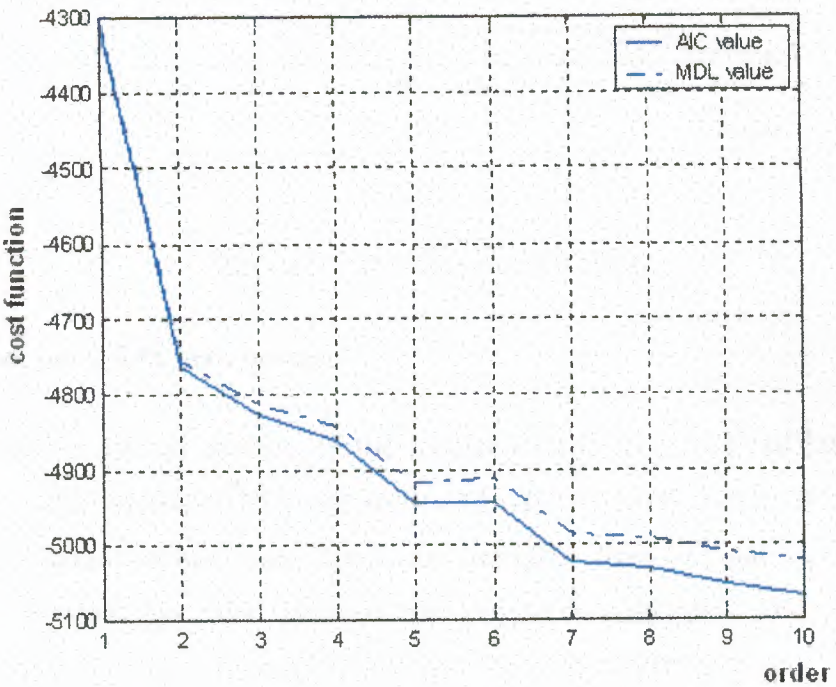


Fig 4.14 Order selection for Levinson.

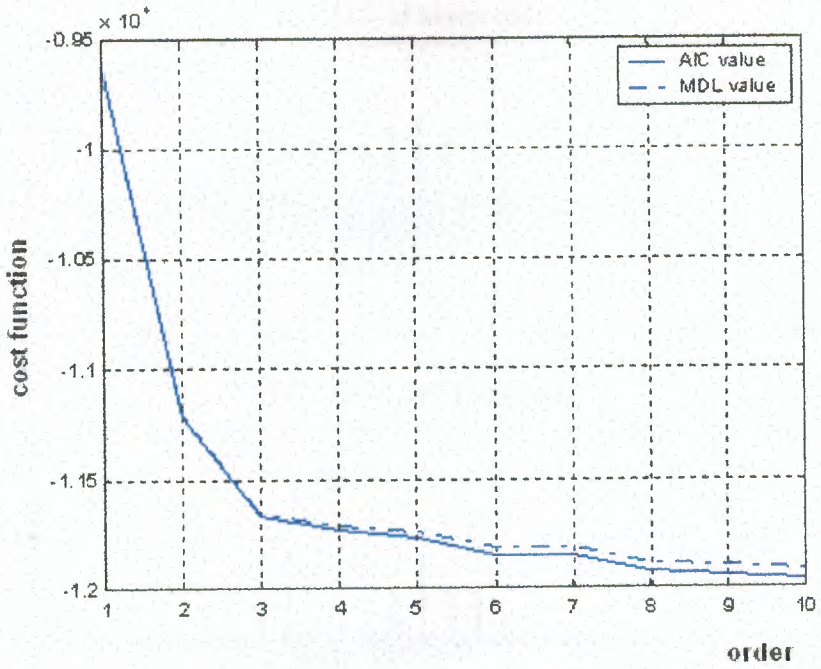


Fig 4.15 Order selection for Burg.

4.6.3.4 Application of LPC parameters

We mainly studied features of the predictor error. The ACFs of both original signals and residual are plotted for one section of typical voiced speech. It is clear that the ACF of residual shows more significant periodics than original signal. This is because the residual has been reduced the long-term correlation and made more whitened. So it may be used for pitch detection with better performance.

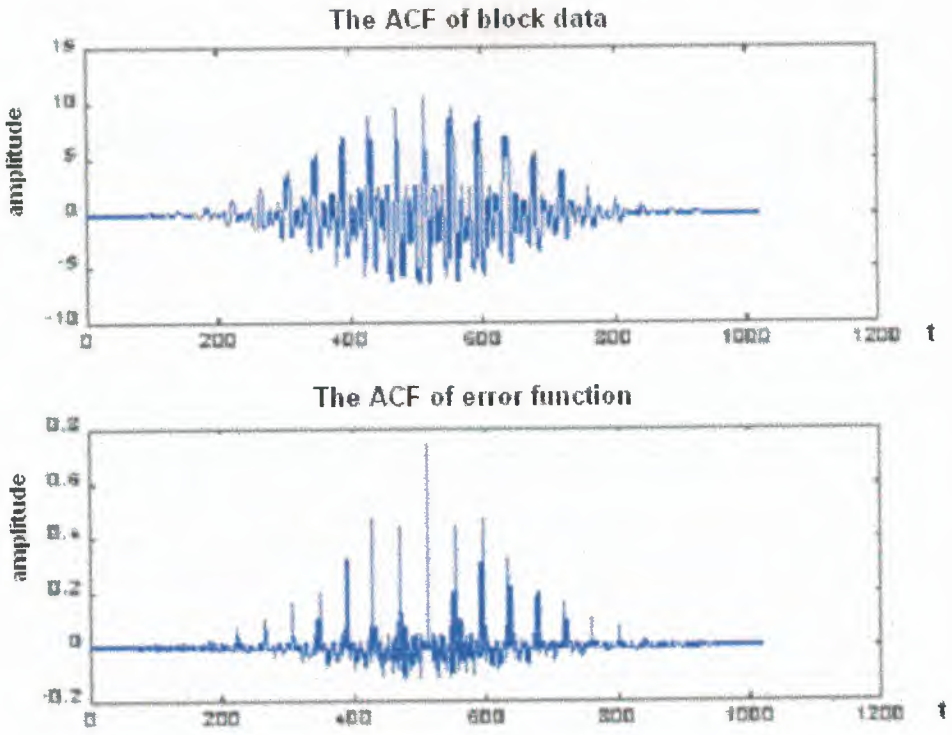
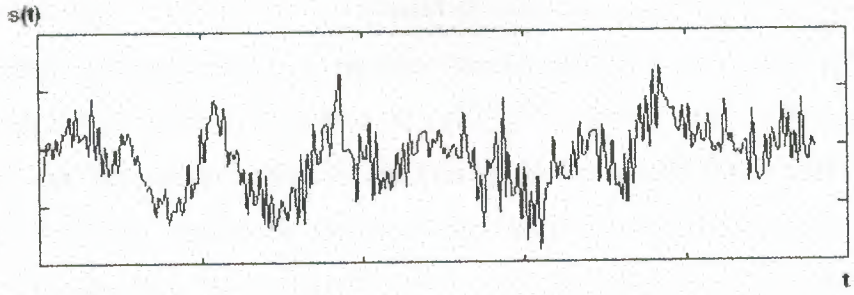


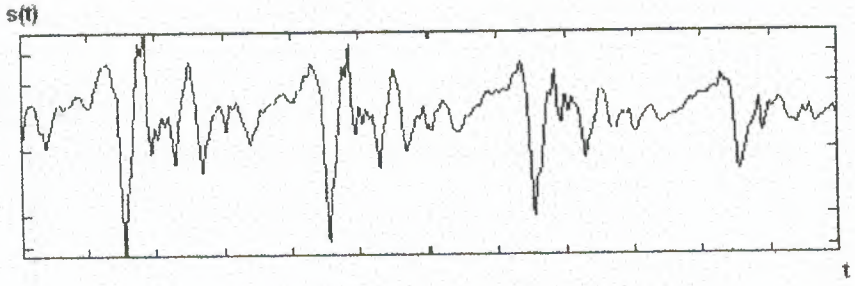
Fig 4.16 Autocorrelation Function of block data and residual of speech.

4.6.4 Determining Pitch Period and Voiced/Unvoiced Decision

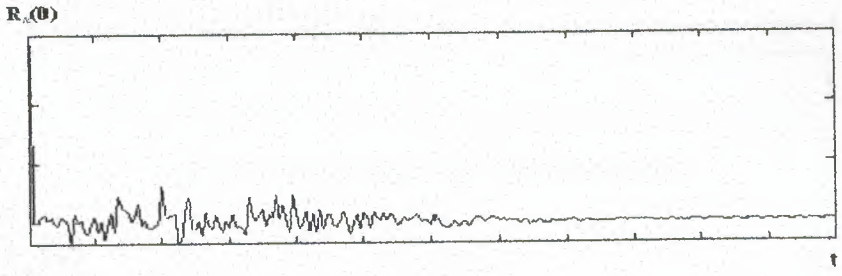
For each frame, we must determine if the speech is voiced or unvoiced. We do this by searching for periodicities in the residual (prediction error) signal. In the figures below, we see the residuals for two typical frames, one voiced and one unvoiced. Clearly, the unvoiced frame is very *noise-like*, but the periodicities in the voiced residual are not easy to see. Therefore, we compute the autocorrelation of both residuals. In the unvoiced frame, the autocorrelation is near zero except for the spike at $\mathbf{R_x(0)}$, as we expect for white noise. However, the autocorrelation for the voiced frame clearly displays the periodicities.



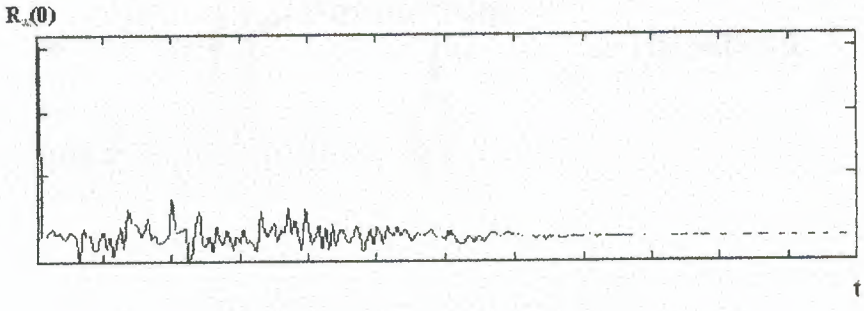
(a)



(b)



(c)



(d)

Fig 4.17 Residuals for two typical frames.(a) unvoiced ;(b) voiced ;(c) autocorrelation of unvoiced ;(d) autocorrelation of voiced.

To determine if the frame is voiced or unvoiced, we apply a threshold to the autocorrelation (shown below). Typically, this threshold is set at $\mathbf{R_x(0)} * \mathbf{0.3}$. If no values of the autocorrelation sequence exceed this threshold, then we declare the frame unvoiced. If we have periodicities in the data (as in the second figure), then we should see spikes which do exceed the threshold; in this case we declare the frame voiced. Notice that the distance between spikes in the autocorrelation function is equivalent to the pitch period of the original signal.

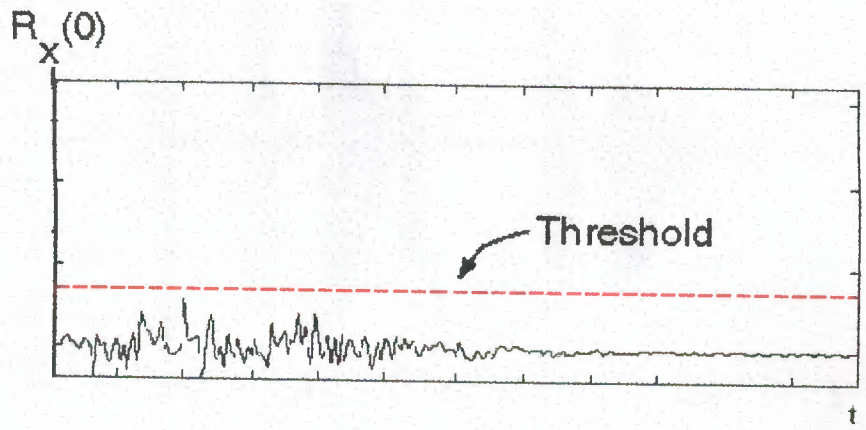


Fig 4.18 Unvoiced – Autocorrelation.

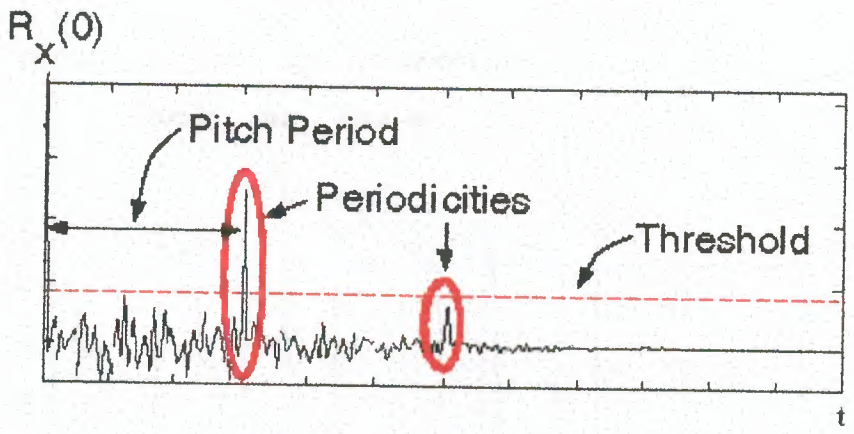
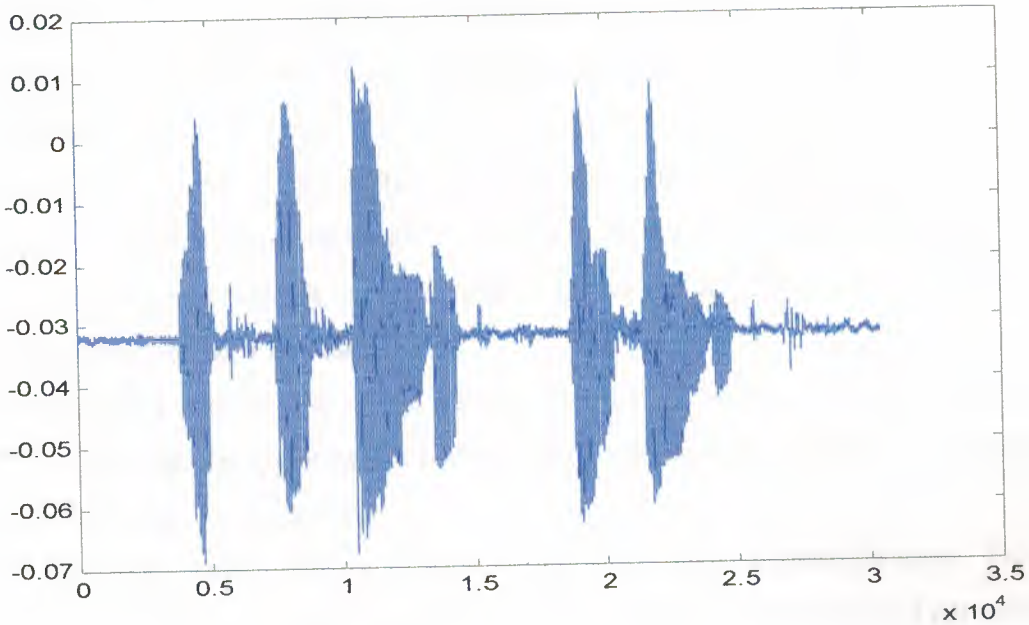


Fig 4.19 Voiced – Autocorrelation.

The figures below demonstrate the voiced and unvoiced decisions for the phrase *look for sounded unsounded.* Notice the large unvoiced regions which correspond to *l* and *s*. Also note the pitched regions which correspond to the vowels.



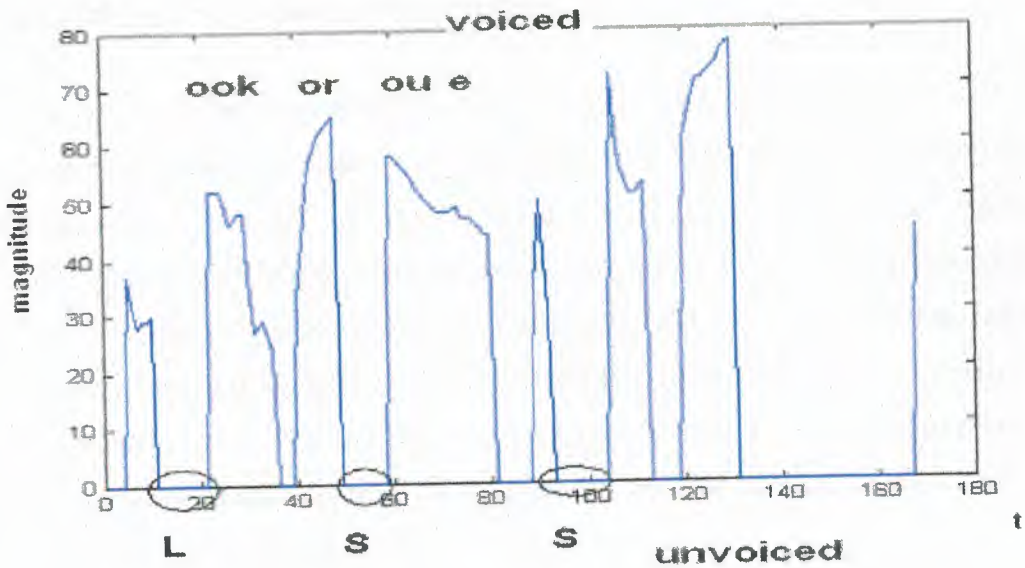


Fig 4.20 My Speech signal and Voiced – Unvoiced decision with pitchperiod.

4.6.5 Quantization

Quantization was simulated through scaling and rounding. For example, four bit quantization was simulated by scaling the output between -8 and 8, rounding, and scaling back. Note that this is not strictly 4 bit quantization (17 possible values, not 16), but is still very close.

Each AR reflection coefficient was quantized to four bits. The pitch period and gain (volume) were scaled to six and seven bits respectively. Unvoiced speech was coded by a pitch period of zero.

This leads to 53 bits/frame for voiced speech, and 29 bits/frame for unvoiced, with respective compression ratios of 27x and 49x, or bit rates of 2.4 kbits/s and 1.3 kbits/s. For our actual speech samples, with both voiced and unvoiced components, compression was between 35x and 42x.

Although 4 bits for the model parameters may be seem to be a bit coarse, it should be noted that the quantization had very little effect on the quality of the speech that was synthesized by the decoder.

Many LPC systems operate in such a way as to require constant bit rates. These systems use extra space in the unvoiced frame models for error detection and correction. Note that 2.4 kbits/s is the standard for LPC10 coding, the US standard in this area.

4.6.6 Decoding and Frame Interpolation

Now that we have a voiced/unvoiced decision for each frame, along with the appropriate pitch periods, we must send this information to the decoder and reconstruct the residual signals. Below are the reconstructed residuals (for a voiced frame - unvoiced is simply white noise) using pulses, doublets, and the government test signal.

The pulses and the government test signal both yield equivalent sounding results, but the government test signal more closely recreates the original signal visually (no impact on sound quality). The last figure is the actual residual we are trying to recreate. As you can see, we capture some of the information, but this inadequate reconstruction leads to a mechanical sounding result.

Remember that our frames are overlapping. We therefore cannot just concentrate our results for each frame. Instead we interpolate the time series output by windowing each frame with an interpolation window (we used a trapezoidal window, pictured in

the Figure below) and putting the frames in their appropriate positions (adding the overlap between frames). It is desirable that the interpolation windows add to 1.

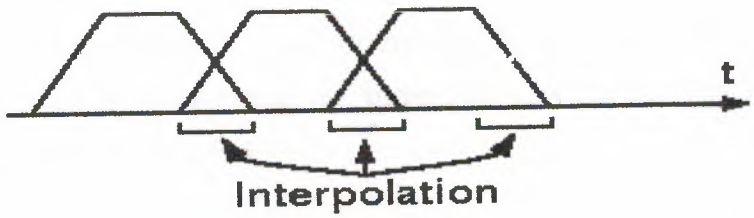
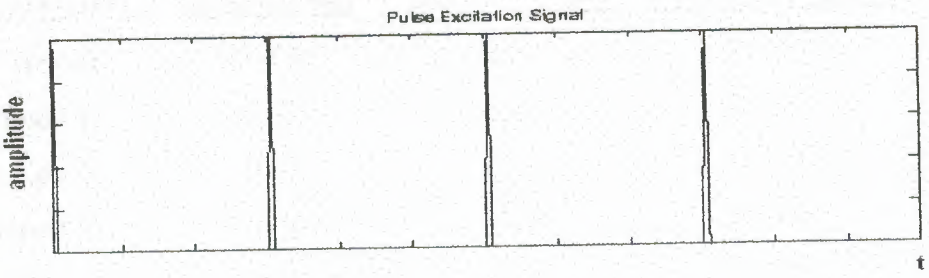
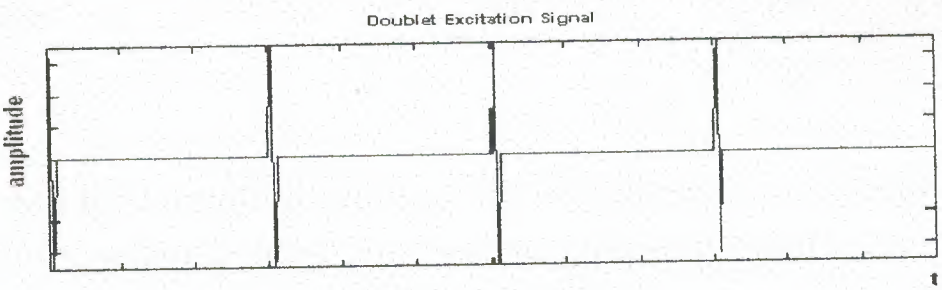


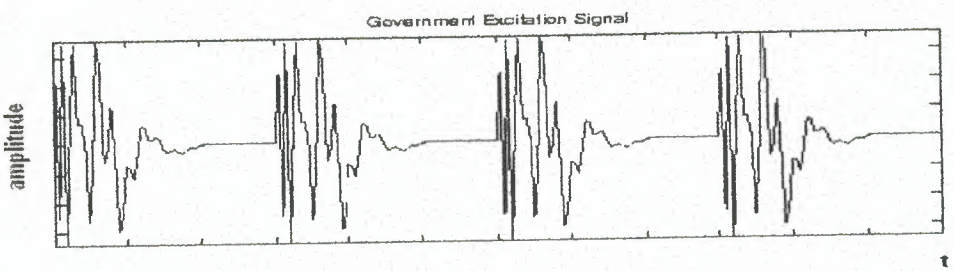
Fig 4.21 Representation of interpolation window .



(a)



(b)



(c)

Fig 4.22 Typical excitation signal. [8]

4.7 LPC Performance

In this thesis, we quantized each of the 10 AR parameters (reflection coefficients) with 4 bits. Thus, the voiced/unvoiced and pitch period information was encoded with 4 bits and 6 bits. Also, we used 7 bits to encode the gain for each frame. Then we reconstruct the speech signal using lpc techniques. So the performance of our lpc implementation is shown in below.

Table 4.1 LPC performance

Speech samples	Original bits	Our coded bits	Compression ratio	Bit rate
My speech	239040	6622	36.0979	1.773 kb/s
Female 1	162720	4845	33.5851	1.905 kb/s
Female 2	146880	4166	35.2568	1.815 kb/s
Female 3	154080	4719	32.6510	1.960 kb/s
Male 1	218880	5904	37.0732	1.726 kb/s
Male 2	177120	4607	38.4458	1.665 kb/s
Male 3	210240	4986	42.1661	1.517 kb/s

As a result typical LPC techniques that we implement , result in rates of about 2.0 kbits/s, a considerable improvement over the uncompressed rate of 64 kbit/s.

4.8 Summary

Linear predictive coding with Autogressive Modelling and its implementation are discussed in this chapter. Other practical aspects is considered and shown also.

We code the speech signal with such consideration and then we reconstruct it using LPC techniques.The reconstructed speech is very mechanical. This is due to the poorly reconstructed error residuals, which are used to drive the output models for each frame. However, even when each frame is driven with white noise (totally ignoring the pitch periods and voiced/unvoiced decisions), the speech is still understandable. This demonstrates that the AR model did a good job of removing the structure from the original signal. Therefore, we suspected that we could improve the quality by doing a better job of encoding the error residuals. This is what is done in the Residual Excited Linear Predictor (RELP).

5. CONCLUSION

5.1 Overview

This thesis has focused on different aspects of autoregressive modelling of speech, for the application in LPC. The LPC system yielded varying qualitative results, yet it had desirable characteristics. The LPC system resulted in very intelligible speech at very low bit rates, yet the voices had a very mechanical sound to them, a typical artifact of this type of coder. Overall bit-rates for the LPC were around .2 bits/sample, a compression ratio of roughly **27:1**.

The LPC coder would not be very useful in trying to get high quality speech. It was determined that there is a signal degradation using LPC regardless of the number of bits used to quantize the signal before putting it on the channel. LPC is the system of choice when bit-rate is an important issue.

5.2 Summary of Contribution

Digital communication systems become increasingly important. To reduce costs and bandwidth, data compression is necessary. An important signal is speech and an enormous effort is put into the development of efficient coding algorithms, because of the commercial applications.

Linear Prediction has been applied very successfully in speech coding and many coders are based on this technique. These coders use an autoregressive model to describe the speech signal. The reconstructed speech is synthesised by feeding a suitable excitation through the autoregressive synthesis filter. Coders differ mainly in the way the excitation is selected and coded. Perhaps the main reason for the success of Linear Prediction is that the autoregressive model combines good prediction capabilities in the time domain with an accurate description of the speech spectral envelope. These properties can be used for efficient coding of the model and its excitations and for techniques such as interpolation, error weighting and postfiltering. This thesis focused

on application of autoregressive models for speech coding, such as estimation, interpolation and quantisation.

The Autocorrelation method of Linear Prediction is treated. This estimation method is frequently used in speech coding. It is known that it does not perform satisfactorily without a tapering data window Many Linear Prediction based coders process the speech on a frame by frame basis. It is important for the subjective quality that the model changes sufficiently smoothly from one frame to another. Therefore, interpolation of the autoregressive spectral models of consecutive frames is used. Different transformations of the parameters of the autoregressive model, called representations, can be used for interpolation. Then the autoregresive model is quantized and send to a channel.

5.3 Future Work

Future work in this realm could include a more complex encoding scheme than the ones implemented in this project. It could be worthwhile to take the residual in the RELP scheme and code it using a wavelet encoder in order to reduce the overall bit-rate.Another scheme known as CELP, code excited linear prediction, is a very useful compromise between the LPC and RELP schemes. CELP is similar to the LPC system except that it has a look-up table of driving signals.

This allows much less information to be sent than in the RELP system, yet due to the many driving signals in the table, the reconstruction is comparable to the RELP. The additional driving signals allow a better approximation of the true residual than LPC which only uses white noise or an impulse train to model the residual. The CELP system is one that is commonly implemented in many digital cellular phones. Finally, it could be useful to explore non-linear models instead of the AR model implemented in the LPC schemes.

Non-linear models have the potential to improve system performance, since they can match a broader class of signals.

BIBLIOGRAPHY

1. B.S. Atal and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *Journ. Acoust. Soc. America*, Vol. 50, pp. 637-655, 1971.

2. B.S. Atal, "Predictive Coding of Speech at Low Bit Rates," *IEEE Trans. Communications, vol. 30(4)*, pp.600-614, 1982.

3. J.S. Erkelens and P.M.T. Broersen, "Equivalent Distortion Measures for Quantisation of LPC Model," *Electronics Letters Vol. 31 No. 17*, pp. 1410-1412, 1995d.

4. L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Ed. A.V. Oppenheim, Prentice-Hall, 1978.

5. R. Veldhuis and M. Breeuwer, *An Introduction to Source Coding*, Prentice Hall, 1993.

6. A.M. Kondoz, *Digital Speech*, John Wiley & Sons, 1994.

7. Athanasios Papoulis. Probability, Random Variables, and Stochastic Processes. McGraw-Hill, New York, NY, 1984.

8. Thomas Parsons. Voice and Speech Processing. McGraw-Hill, New York, 1987.

9. A.V. Oppenheim, R.W. Schafer, and J. Buck. Discrete-Time Signal Processing. Prentice-Hall, New York, 2nd edition, 1999.

10. J.D. Markel and A.H. Gray, *Linear Prediction of Speech*, Springer-Verlag, New York, 1976.

11. MATLAB Identification and Signal Processing Toolbox User's Guide, The Mathworks, 1990.

12. A.H. Gray and D.Y. Wong, "The Burg Algorithm for LPC Speech Analysis/Synthesis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 28(6), pp. 609-615, 1980.

13. A. El-Jaroudi and J. Makhoul. Discrete all-pole modelling. *IEEE Trans. Signal Processing*, vol. 39, pp. 411{423, Feb. 1991.

14. A. S. Spanias, "Speech coding: A tutorial review," Proceedings of the IEEE, vol. 82, pp. 1539–1582, Oct. 1994.

15. F. A. Westall, R. D. Johnston, and A. V. Lewis, "Speech Technology for Telecommunications", London, BT Technol J, Vol 14, No 1, pp. 9-27, January 1996.

16. W T K Wong, R M Mack, B M G Cheetham, and X Q Sun, "Low rate speech coding for telecommunications", London, BT Technol J, Vol 14, No 1, pp. 28-44, January 1996.

17. Wesley Pereira, "Modifying LPC Parameter Dynamics to Improve Speech Coder Efficiency" , Master's thesis, McGill University, Montreal, Canada, September 2001.

18. Jacek Stachurski, "A Pitch Pulse Evolution Model for Linear Predictive Coding of Speech" , Doctora's thesis, McGill University, Montreal, Canada, February 1998.

19. Athanasios Papoulis. Probability, Random Variables, and Stochastic Processes. McGraw-Hill, New York, NY, 1984.

20. K.Sam Shanmugan and A.M.Breipohl, Random Signals:Detection,Estimation and Data Analysis,John Wiley&Sons, New York, 1988.

21.Chanyan Li, Allen Gersho and Vladimir Cuperman, "Analysis by Synthesis Low–Rate Multimode Harmonic Speech Coding" , University of California, Santa Barbara, 1999.

22. J.S. Erkelens and P.M.T. Broersen, "Bias Propagation in the Autocorrelation Method of Linear Prediction" *IEEE Trans. On Speech and Audio Processing,* Vol. 5, No. 2, pp. 116-119, March 1997.

23. J.S. Erkelens and P.M.T. Broersen, "Reconstruction error measure for quantisation of LPC models," *Electronics Letters Vol. 32 No. 15,* pp. 1347-1349, 1996j.

24. J.S. Erkelens and P.M.T. Broersen, "LPC Interpolation by Approximation of the Sample Autocorrelation Function"*IEEE Trans. On Speech and Audio Processing,* Vol. 6, No. 6, p.p 569-573, November 1998.

25. Mark Hasegawa-Johnson, " Lecture Notes in Speech Production, Speech Coding, and Speech Recognition," University of Illinois at Urbana-Champaig, February 17, 2000.

26. Panos E. Papamichalis, *Practical Approaches to Speech Coding*, Prentice Hall.

27. Tomas P. Burnwell III, Kumbiz Nayebi and Craig H. Richardson, *Speech Coding: A Computer Labarotory Textbook*, John Wiley&Sons.

28. Chris Rowden, *Speech Processing*, University of Essex, McGRAW-HILL,1992..

29. North Atlantic Treaty Organisation Defence Research Group, "Potentials of Speech and Language Technology Systems For Military Use: An Application and Technology - Oriented Survey," *Technical Report, AC/243 (Panel3) TR/2* ,14August 1996.

30. Shawn Hunt, Norman A. Lopez, "RELP Coding for Voice Communication," University of Puerto Rico.

31. T. Dutoit, "An Introductory Course on Speech Processing," TCTS Lab, Faculté Polytechnique de Mons, Belgium, 2000.

32. Earl E. Swartzlander, Jr. "From Concept to Production in Secure Voice Communications," University of Texas at Austin.

33. Andreas Spanias, "Digital Signal Processing," *Lecture Notes,* Arizona State University, Spring 2000.

34. Luis Miguel Teixeira de Jesus and Gavin C. Cawley, "Speech Coding and Synthesis Using Parametric Curves," University of East Anglia, Norwich, U.K.

35. Philip J.B. Jackson, "Fricative and Aspiration Components in Speech Production," , Doctora's thesis, University of Southampton, May 2000.

36. The speech samples. Retrieved October 23, 2002 from the World Wide Web: http://www.ctr.columbia.edu/~dpwe/sounds/".

APPENDIX A: VOCAL TRACT

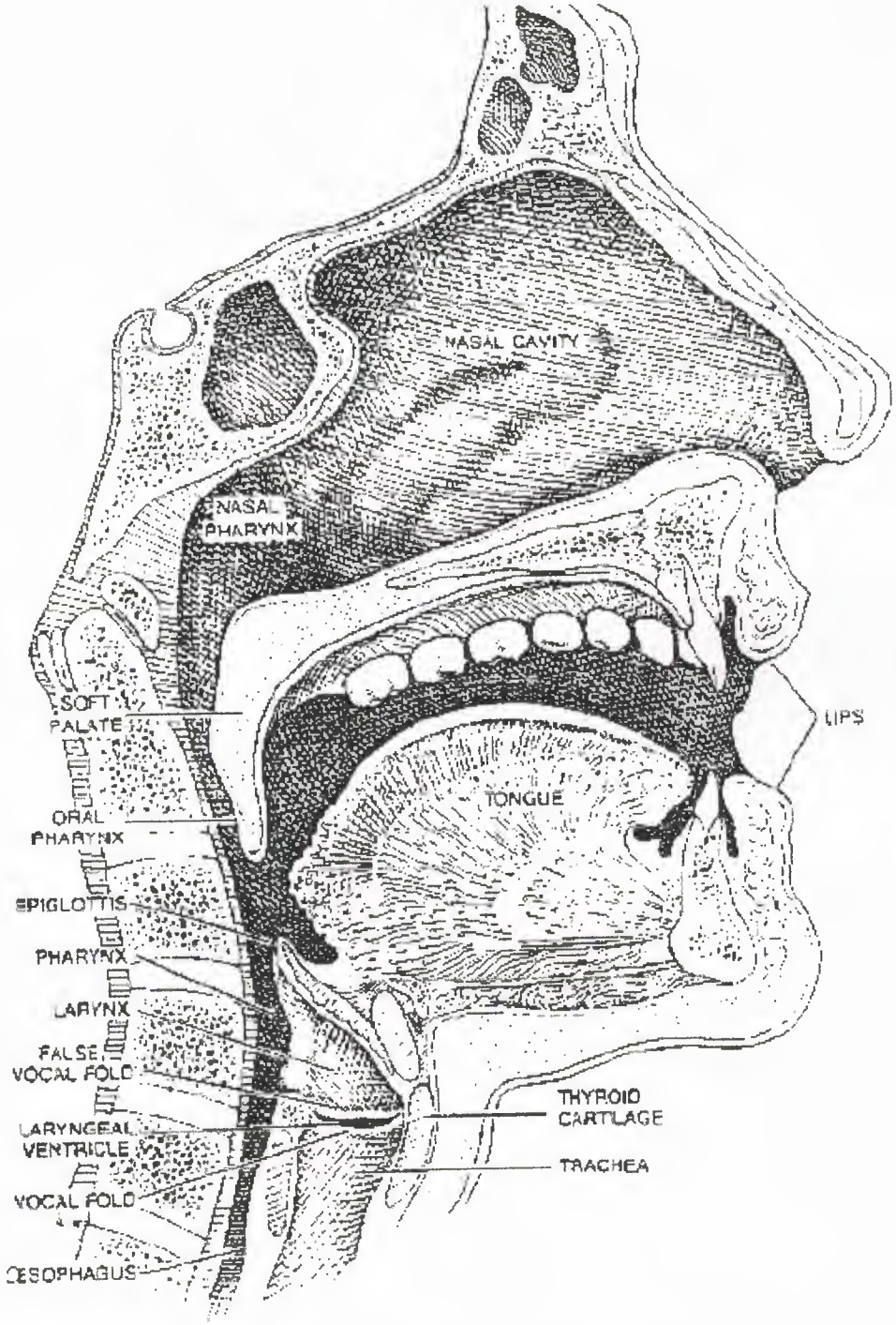


Figure : Sagittal, or longitudinal, section of the human vocal apparatus, reprinted from Sundberg(1977).

Figure is a sketch taken from sundenberg(1977), which shows the airways that are involved in sound production during speech, except the lungs which are connected via the trachea: the larynx, the pharynx, and the oral and nasal cavities, which together constitute the vocal tract. The shape of these passage ways is modified by the tongue, the lips, the jaw and the velum,which hangs down from the soft palate. The epiglottis covers the larynx during swallowing to prevent any unwanted food stuffs from entering the trachea, but is normally held open during speech and lies close to the back of the tongue. These parts of anatomy are often referred to as the articulators since the adjustment of the geometry along the vocal tract allows for the full range of sounds that make up our phonetic repertory to be produced.

APPENDIX B: CALCULATION AND ESTIMATION

B.1 PARAMETER ESTIMATION

Given N samples of speech, we would like to compute estimates to α_i that result in the best fit. One reasonable way to define ``best fit" is in terms of mean squared error. These can also be regarded as ``most probable" parameters if it is assumed the distribution of errors is Gaussian and a priori there were no restrictions on the values of a_i. The error at any time, e_n, is:

$$e_n = s_n - \hat{s}_n$$

$$= s_n - \sum_{i=1}^{p} a_i s_{n-i}$$

Hence the summed squared error, E, over a finite window of length N is:

$$E = \sum_{n=0}^{N-1} e_n^2 = \sum_{n=0}^{N-1} \left(s_n - \sum_{k=1}^{p} a_k s_{n-k} \right)^2$$

The minimum of E occurs when the derivative is zero with respect to each of the parameters, a_k. As can be seen from equation the value of E is quadratic in each of the a_k therefore there is a single solution. Very large positive or negative values of a_k must lead to poor prediction and hence the solution to $\partial E / \partial a_k = 0$ must be a minimum.

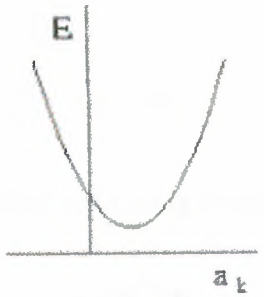


Figure B1: Schematic showing single minimum of a quadratic.

Hence differentiating equation with respect to a_j and setting equal to zero gives the set of p equations:

$$\frac{\partial E}{\partial a_j} = 0 = -\sum_{n=0}^{N-1}\left(2\left(s_n - \sum_{k=1}^{p} a_k s_{n-k}\right)s_{n-j}\right)$$

$$= -2\sum_{n=0}^{N-1} s_n s_{n-j} + 2\sum_{n=0}^{N-1}\sum_{k=1}^{p} a_k s_{n-k} s_{n-j}$$

rearranging equation gives:

$$\sum_{n=0}^{N-1} s_n s_{n-j} = \sum_{k=1}^{p} a_k \sum_{n=0}^{N-1} s_{n-k} s_{n-j}$$

Define the covariance matrix, Φ with elements $\Phi_{i,k}$:

Now we can write equation as:

$$\Phi_{i,0} = \sum_{k=1}^{p} \Phi_{i,k} a_k$$

or in matrix form:

$$\begin{pmatrix} \Phi_{1,0} \\ \Phi_{2,0} \\ \Phi_{3,0} \\ \ldots\ldots \\ \Phi_{p,0} \end{pmatrix} = \begin{pmatrix} \Phi_{1,1}\Phi_{1,2}\Phi_{1,3}\ldots\Phi_{1,p} \\ \Phi_{2,1}\Phi_{2,2}\Phi_{2,3}\ldots\Phi_{2,p} \\ \Phi_{3,1}\Phi_{3,2}\Phi_{3,3}\ldots\Phi_{3,p} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \Phi_{p,1}\Phi_{p,2}\Phi_{p,3}\ldots\Phi_{p,p} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \ldots \\ a_p \end{pmatrix}$$

or simply:

$$\Phi_0 = \Phi_a$$

Hence the *Covariance method* solution is obtained by matrix inverse:

$$a = \Phi^{-1}\Phi_0$$

Note that Φ is symmetric, i.e. $\Phi_{i,k} = \Phi_{k,i}$, and that this symmetry can be expoited in inverting Φ

B.2 THE AUTOCORRELATION METHOD

When dealing with windowed speech we need to take into account the boundary effects in order to avoid large prediction errors at the edges. Can refine the area over which we perform least squares minimisation in equation above and make use of the fact that samples are zero outside of the window to rewrite $\Phi_{i,j}$ as:

$$\Phi_{i,j} = \sum_{n=0}^{N-1-(i-j)} s_n s_{n+(i-j)}$$

Now $\Phi_{i,j}$ is only dependent on the difference, *i-j*, and may be written in terms of the autocorrelation function, $\Phi_{i,j} = r_{i,j}$:

$$r_k = \sum_{n=0}^{N-1-k} s_n s_{n+k}$$

Now Φ is Toeplitz:

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \dots \\ r_p \end{pmatrix} = \begin{pmatrix} r_0 r_1 r_2 \dots\dots r_{p-1} \\ r_1 r_0 r_1 \dots\dots r_{p-2} \\ r_2 r_1 r_0 \dots\dots r_{p-3} \\ \dots\dots\dots\dots \\ r_{p-1} r_{p-2} \dots r_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ a_p \end{pmatrix}$$

Efficient methods exist to invert such matrices, one of which is Durbin's algorithm. Denoting the values of the LP parameters at iteration *i* by $a_k^{(i)}$ and the residual energy by $E(i)$, $E^{(0)} = r_0$ for i = 1, 2, ...

$$k_i = \left(r_i - \sum_{j=1}^{i-1} a_j^{i-1} r_{i-j} \right) / E^{(i-1)}$$

$$a_i^{(i)} = k_i$$

$$a_i^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}$$

$$E^{(i)} = \left(1 - k_i^2\right) E^{(i-1)}$$

For example, for the signal let be:

$$r_0 = 2.4470 * 10^8$$

$$r_1 = 2.2466 * 10^8$$

$$r_2 = 1.7823 * 10^8$$

Therefore on the first iteration:

$$k_1 = r_1 / E^{(0)} = 0.9181$$

$$a_1^{(1)} = k_1 = 0.9181$$

$$E^{(1)} = \left(1 - k_1^2\right) E^{(0)} = \left(1 - 0.9181 * 0.9181\right) 2.4470 * 10^8 = 0.38441 * 10^8$$

And on the second iteration:

$$k_2 = \left(r_2 - a_1^{(1)} r_1\right) / E^{(1)} = \left(1.7823 * 10^8 - 0.9181 * 2.2466 * 10^8\right) / 0.38442 * 10^8 = -0.72915$$

$$a_2^{(2)} = k_2 = -0.72915$$

$$a_1^{(2)} = a_1^{(1)} - k_2 a_1^{(1)} = 1.58753$$

$$E^{(2)} = \left(1 - k_2^2\right) E^{(1)} = 0.18 * 10^8$$

The parameters k_i are known as the reflection parameters.

APPENDIX C: MATLAB PROGRAM

C.1 Linear Predictive Encoding and Decoding

```
% elpc.m -  linear predictive encoding
%[Data1]=wavread('sf1_cln.wav') ;
% [Data]=decimate(Data1,2);
  [Data]=wavread('buraka.wav') ;
%specgram(Data, 512, 8000);
%colorbar
% Initial value %%%%%%%%%%%%%%%%%%%%
global flt_speech;
burak_speech = Data-mean(Data);        % speech signal
S=length(Data);
N=240;                                 % length of frame window
overlap = 60;                          % amount of overlap between adjacent windows
frame = N-overlap;                     % frame rate
[B,A] = butter(10,pi*9/40);            % lowpass filter used in v/uv decision
% to sure that data length evenly divides frame length
rm = rem(length(burak_speech)-overlap,frame);
burak_speech = burak_speech(1:length(burak_speech)-rm);  % Cut-off end
number_of_blocks = (length(burak_speech)-overlap)/frame;  % number of blocks in speech
vuv = zeros(1,number_of_blocks);                  % voiced/unvoiced for each block
pperiod = zeros(1,number_of_blocks);              % pitch period for each block
encoder = zeros(number_of_blocks,12);             % model the coefficient matrix
% Voiced/unvoiced decisions (each block) %%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:number_of_blocks,
  indices = (k-1)*frame;
  blkind = (indices+1):(indices+N);               % block indices
  origblk = burak_speech(blkind);                 % block from original data
  [vuv(k) pperiod(k),error,fthrd] = vunv1(origblk,B,A);  %the v/unv decision and the pitch P
end;

% AUTOGRESSIVE(AR) modeling and encoding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

flt_speech = filter([1 -0.9375],1,burak_speech); % premphasis filtering

for k = 2:(number_of_blocks-1),         % model the inner blocks
  if (vuv(k-1) == vuv(k+1)),          % if adjacent blocks are similar, ensure that the
    vuv(k) = vuv(k-1);              % current block is of similar type (voiced or unvoiced)
    if( vuv(k)==1)
      pperiod(k) = floor((pperiod(k-1)+pperiod(k+1))/2);  % Changeto voiced - pperiod=average        else
        pperiod(k) = 0;                       % Changeto unvoiced - pperiod=0
    end
  end

 fltrblk = make_block((k-1)*frame+1,N);                % block from filtered data
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  if (vuv(k) == 1)
    [model,refl] = ar(fltrblk,10,'burg');            % Apply an AR 10 for voiced

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Refl_coef = hut(:,1);                  % The PARCOR coeffcients
     %Aparm = [1 a(10,:)]; % The A-par of Prediction Error Filter A(z);
     %error_f= ef';
     %error_b=eb;
     %Gain=MSEP(10);
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     % LAR=log10((1-refl)./(1+refl));

     refl_coeffs = refl(1,2:11);           % Quasi-4 bit quantization
     refl_coeffs = round(refl_coeffs*8)/8;
     energy = model(1,1);

    if (energy > 1e-5)
        energy = 1e-5;
     end
     energy=quant(energy,1e-7);             % 100 values => 7 bits
     qperiod = 2*round(pperiod(k)/2);         % 40 vals (100-20)/2 => 6 bits
     encoder(k,:) = [energy  qperiod refl_coeffs];
     encoderburak(k,:) = [refl_coeffs];

   else
    [model,refl] = ar(fltrblk,4,'burg');       % Apply an AR 4 for unvoiced

    % [h,a,ef,eb,MSEP]=myburg(fltrblk,4);

     refl_coeffs = refl(1,2:5);
     refl_coeffs = round(refl_coeffs*8)/8;    % Quasi-4 bit quantization
     energy = model(1,1);

      if (energy > 1e-5)
         energy = 1e-5;
      end
      energy=quant(energy,1e-7);             % 100 values => 7bits
      qperiod = 2*round(pperiod(k)/2);         % 40 vals (100-20)/2 => 6 bits
      encoder(k,:) = [energy  qperiod refl_coeffs zeros(1,6)];
      encoderburak(k,:) = [refl_coeffs zeros(1,6)];

   end;
end;

function chunk=make_blk(m,N)

% chunk=blk(m). This function first obtains a block of length N=240
% starting from index m, Hamming windows this block and returns it.

global flt_speech;

chunk = flt_speech(m:(m-1)+N);
chunk = chunk .* hamming(length(chunk));
```

```
if (length(chunk) ~= N)
  disp('Error');
end;


function [vu,pperiod,error,fthrd] = vunv(x,B,A)
% This function determines whether x is voiced or unvoiced. It also
% determines the pitch period

K = 0.6;                          % clipping factor

[model,refl] = ar(x,10,'burg');          % obtain an AR model for voiced block
%[h,a,ef,eb,MSEP]=myburg(x,10);

[num,den] = th2tf(model);
error = filter([1 den],num,x);
error1 = filter([1 den],num,x);

error = error .* hamming(length(error));     % windowing error data

fltx = filter(B,A,error);                % filtering data

fthrd = 1:round(length(fltx)/3);          % first third indices
fmax1 = max(abs(fltx(fthrd)));

sthrd = (length(fltx)-length(fthrd)):length(fltx); %last third indices
fmax2 = max(abs(fltx(sthrd)));

C = K*min([fmax1 fmax2]);                    % clipping value

fltx = (fltx - C*sign(fltx)).*(fltx > C | fltx < -C);    % clipping

cor = xcorr(fltx);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Draw the ACF of both BlkData and error_f;
% Rxx=xcorr(BlkData);
% Rxx=xcorr(Data);
% Ree=xcorr(error);
% figure('Name','The Autocorrelation of BlkData and error_f');
% subplot(2,1,1);plot(cor);
% title('The ACF of BlkData');
% subplot(2,1,2);plot(Ree);
% title('The ACF of error_f');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
[amax,indices] = max(cor);          % maximum correlation value at center

rcor = cor(indices+20:indices+100);   %indices 20-100 of right half original correlation

start = find(rcor < 0.3*amax);
start = start(1);

[cmax,indices] = max(rcor(start:length(rcor)));

if (cmax > 0.3*amax)
  pperiod = indices + 20 + start - 1;
    vu = 1;
else
```

```
  pperiod = 0;
  vu = 0;
end;

%%%%%%%%%%%%%%%%%%%%%%Cepstrum%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%if (length(x)<3)
 %  pperiod=0;
%else

%xfft=abs(fft(x));
%logfft=log(xfft);
%ceps=ifft(logfft);
%[cmax,indicesburak]=max(ceps);
%pperiod=indicesburak;
%end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%  decode lpc encoding
%
% Input - encodem (# of frames)x12 matrix of AR parameters
%


% Glottal Excitation sequence to mimic actual voice excitations
glttl = [ 249 -262 363 -362 100 367 79 78 10 -277 82 376 288 -65 -20 138 -62 -315 -247 -78 -82 -123 -39
65 64 19 16 32 18 -15 -29 -21 -18 -27 -31 -22 -12 -10 -10 -4];


N=240;                          % length of window
overlap = 60;                    % amount of overlap between adjacent windows
frame = N-overlap;                 % frame length

numblks = length(encoder);
outsig = [];
finalsignal = zeros(1,numblks*frame+overlap);


int_win = [(1:overlap)/overlap ones(1,N-2*overlap) (overlap:-1:1)/overlap];

for k = 2:(numblks-1),
   if (encoder(k,2) ~= 0)              % check if frame was voiced
     insig = zeros(1,N+length(glttl));
     pp = encoder(k,2);               % pitch period of frame
     for l = 1:floor(N/pp)
       st = (l-1)*pp;
       insig((st+1):(st+length(glttl)))=glttl;  % periodic glottal train
     end
     insig = insig(1:N);
     insig = insig/std(insig);              % normalize variance
   else                               % Unvoiced excitation
     insig = randn(1,N);                    % form random noise input signal
   end;

   insig = sqrt(encoder(k,1))*insig;          % assign excitation energy
   den = rc2poly(encoder(k,3:12));             % turn reflect coeffs into AR
   outsig = int_win.*filter(1,den,insig);
```

```
  st = (k-1)*frame;

  % Add data from each overlapping frame into final signal
  finalsignal((st+1):(st+N)) = finalsignal((st+1):(st+N)) + outsig;

end;

finalsignal = filter(1,[1 -0.9375],finalsignal);  % de-emphasis filtering of speech
wavwrite(finalsig,'burakb.wav');


% rate.m
% compression rate for our lpc encoder


frame_number = length(encoder)-2;        % Omit first and last frames
                                         % since we don't encode them


% Original bits = bits per sample * # of frame * sample per frame
original_bits = 8*frame_number*180

our_bits = (6+7)*frame_number+ 10*4*sum(encoder(:,2)>0)+4*4*sum(encoder(:,2)==0)

compression_ratio = original_bits/our_bits

bits_sec= 64000/compression_ratio
```

C.2 Levinson and Burg Function

```
function [k,a,er_f,er_b,MSE]=myburg(s,P)
% the P-order LPC coefcients
% Lattice-burg algorithm;
% Burak Alacam

N=length(s);
a=zeros(P,P)
k=zeros(P,1);

MSE=zeros(P+1,1);

MSE(1)=s'*s/N;

complexity=1;                    % computing complexity;

ef=zeros(P+1,N); eb=zeros(P+1,N);

% (1).Initialize the ef(1,:)=eb(1,:)=s(1:N);
ef(1,1:N)=s(1:N)';
eb(1,1:N)=s(1:N)';

% k(1)=
nef=sum(ef(1,2:N).*ef(1,2:N));neb=sum(eb(1,1:N-1).*eb(1,1:N-1));
norm=nef+neb;
k(1)=-2*sum(ef(1,2:N).*eb(1,1:N-1))/norm;       %complexity=N;
a(1,1)=k(1);

MSE(2)=(1-k(1)*k(1))*MSE(1);
% compute the ef(2,n)=ef(2,n)+k(1)*eb(1,n-1)---n:1-N;
%                    eb(2,n)=eb(1,n-1)+k(1)*ef(1,n);
ef(2,1)=ef(1,1);
ef(2,2:N)=ef(1,2:N)+k(1)*eb(1,1:N-1);

eb(2,1)=k(1)*ef(1,1);
eb(2,2:N)=eb(1,1:N-1)+conj(k(1))*ef(1,2:N);

% Iterate to compute the p=2:Pth orders LPC pars;
for p=2:P
  nef=sum(ef(p,p+1:N).*ef(p,p+1:N));        neb=sum(eb(p,p:N-1).*eb(p,p:N-1));
  norm=(nef+neb)/2;
  k(p)=-sum(ef(p,p+1:N).*eb(p,p:N-1))/norm;          %complexity=complexity+N;

  a(p,p)=k(p);

  % compute a(p,2:p)
  a(p,1:p-1)=a(p-1,1:p-1)+k(p)*a(p-1,p-1:-1:1);       %complexity=complexity+1;

  % compute ef(p+1,n)=ef(p,n)+k(p)*eb(p,n-1);
  %                   eb(p+1,n)=eb(p,n-1)+k(p)*ef(p,n);
  ef(p+1,1)=ef(p,1);
  ef(p+1,2:N)=ef(p,2:N)+k(p)*eb(p,1:N-1);

  eb(p+1,1)=-k(p)*ef(p,1);
  eb(p+1,2:N)=eb(p,1:N-1)+k(p)*ef(p,2:N);
  MSE(p+1)=(1-k(p)*k(p))*MSE(p);
```

```
end

er_f=ef(P+1,:);
er_b=eb(P+1,:);


function [A, MSEP, K] = myLevinson(x,P)

N=length(x);

R=xcorr(x);
R=R(N:2*N-1);   % shift it to make it R(0)-R(N-1);

% To compute the p-order prediction, we only need R(0)-R(p);


Apar=zeros(P,P); % To store the A parameters in each recursion;
MSEp=zeros(P);     % To store the MSE(Rou(f,p)) in each recursion;
Kpar=zeros(P);          % To store the PARCOR coef in each recursion;

%compute the initializations
Apar(1,1)=-R(2)/R(1);
MSE(1)=R(1)-R(2)^2/R(1);
Kpar(1)=Apar(1,1);
%loop to compute the parameters in 2--> P orders;
for p=1:P-1
  Deltap=R(p+2);
  for m=1:p
    Deltap=Deltap+Apar(p,m)*R(p+2-m);
  end
  Kpar(p+1)=-Deltap/MSE(p);
  for m=1:p
    Apar(p+1,m)=Apar(p,m)+Kpar(p+1)*Apar(p,p+1-m);
  end
  Apar(p+1,p+1)=Kpar(p+1);
  MSE(p+1)=MSE(p)*(1-Kpar(p+1)^2);
end

A=Apar;       % A P*P matrix;
MSEP=MSE;   % 1*P vector;
K=Kpar;     % 1*P vector;

return;
```

C.3 LPC ANALYSIS

```
% LPC ANALYSİS

function [Refl_coef,Aparm,error_f,error_b,Gain]=LPCanalysis

[Data]=wavread('rain8k.wav');
Start=1;
WL=1000;
WindowType=[
          '1.hamming ';
       '2.hanning ';
       '3.bartlett';
       '4.triang  ';
       '5.blackman';
       '6:Rectrang'
     ];

disp('Type of Window:');
disp(WindowType);

WT =[];
in=[];
while( ~strcmp(in,'1')&~strcmp(in,'2')&~strcmp(in,'3')&~strcmp(in,'4')&~strcmp(in,'5')&~strcmp(in,'6'))
  in=input('please input the type of Window:','s');
  switch in
  case '1'
    WT='hamming';
  case '2'
    WT='hanning';
  case '3'
    WT='bartlett';
  case '4'
    WT='triang';
  case '5'
    WT='blackman';
  case '6'
    WT='Rectrang';
  end
end
BlkData=Data(Start:Start+WL-1);
  if strcmp(WT,'Rectrang')
    BlkData=BlkData;
    w=ones(WL,1);
  else
    winfun=strcat(WT,'(');
    winfun=strcat(winfun,'WL');
    winfun=strcat(winfun,')');
    w=eval(winfun);          % Get the window data;
    BlkData=BlkData.*w; % This is for pure Analysis Case.
  end

BlkData=Data;

FS=8000;

 %w = hamming(length(Data));      % windowing error data
```

```
N=length(Data);

POrder=10;

FFTpoint = 1024;

% [A, MSEP, K] = myLevinson(BlkData,POrder);
 %  Refl_coef = K(:,1)
  % Aparm=[1 A(POrder,:)]
  %error_f = filter(Aparm,1,BlkData)
  % error_b=[]                    % for Levinson algorithm, no backward_error;
  % Gain=MSEP(POrder);


   [k,a,ef,eb,MSEP]=myburg(BlkData,POrder);
   Refl_coef = k(:,1);            % The PARCOR coeffcients
   Aparm = [1 a(POrder,:)];       % The A-par of Prediction Error Filter A(z);
   error_f= ef';
   error_b=eb;
   Gain=MSEP(POrder);



% plot the Coeffecients;

   % In LPC analysis, we will plot the following graphs:
   % Fig1.
   figure('Name','The signals in Time Domain!');

   subplot(4,1,1);plot((1:N)/FS,Data);hold on;
   title('The original Signal and windowed signal!');
   xlblstr=strcat('Time in Sec, FS=',num2str(FS));
   xlblstr=strcat(xlblstr,' Data Length=');
   xlblstr=strcat(xlblstr,num2str(N));
   xlblstr=strcat(xlblstr,' W0=');xlblstr=strcat(xlblstr,num2str(Start));
   xlblstr=strcat(xlblstr,' WL=');xlblstr=strcat(xlblstr,num2str(WL));
   xlabel(xlblstr);ylabel('Value of signal');
   Wdtmp=zeros(N,1);
   Wdtmp(Start:Start+WL-1)=w(1:WL)*max(BlkData);
   % plot the Window ;
   plot((1:N)/FS,Wdtmp,'g-.');hold on;
   Wdtmp(Start:Start+WL-1)=BlkData(1:WL);    % plot the windowed data
   plot((1:N)/FS,Wdtmp,'r--');hold on;

   % plot the zoomed out data;
   subplot(4,1,2);plot((1:WL)/FS+(Start)/FS,Data(Start:Start+WL-1));hold on;
   % title('The zoom out windowed data!');
   Wdtmp=zeros(WL,1);
   Wdtmp(1:WL)=w(1:WL)*max(BlkData);  % plot the Window;
   plot((Start+(1:WL))/FS,Wdtmp,'g-.');hold on;
   Wdtmp=BlkData;                % plot the windowed data
   plot((Start+(1:WL))/FS,BlkData,'r--');hold on;

   subplot(4,1,3);plot(error_f);
   % title('The forward prediction error');

   RecData=filter(1,Aparm,error_f);
   subplot(4,1,4);plot(RecData);
   % title('The reconstructed data with error_f as excitation!');
```

```
figure('Name','The Spectra of these signals');

PSDData=20*log10(abs(fft(Data)));          % For the whole data
subplot(3,1,1);plot(PSDData(1:N/2));
title('The long-term PSD of data');

PSDerrf=20*log10(abs(fft(error_f,FFTpoint)));
subplot(3,1,2);plot(PSDerrf(1:FFTpoint/2));
title('The PSD of error_f');

AF=abs(fft(Aparm,FFTpoint)).^2;
%AModelPSDdb=10*log10(AF);
InvModPSD=Gain./AF;
InvModelPSDdb=10*log10(InvModPSD);
PSDWData=abs(fft(BlkData,FFTpoint)).^2;
PSDWDatadb=10*log10(PSDWData);
subplot(3,1,3);plot(PSDWDatadb(1:FFTpoint/2));hold on;
%subplot(3,1,3);plot(Aparm,'r--');
%subplot(3,1,3);plot(AModelPSDdb(1:FFTpoint/2),'r--');
subplot(3,1,3);plot(InvModelPSDdb(1:FFTpoint/2),'r-.');
figure('Name','Put together');

plot(PSDWDatadb(1:FFTpoint/2));hold on;
plot(InvModelPSDdb(1:FFTpoint/2),'r-');hold on;
plot(PSDerrf(1:FFTpoint/2),'g-.');hold on;

 grid on;legend('DataPSD','Model PSD','errorPSD');

figure('Name','Analysis of The linear prediction parameters');
subplot(4,1,1);plot(Refl_coef);
LAR=log10((1-Refl_coef)./(1+Refl_coef));
subplot(4,1,2);plot(LAR);
title('LAR ');
subplot(4,1,3);plot(MSEP);
[Ha,Wa]=freqz(1,Aparm);
subplot(4,1,4);z=roots(Aparm);zplane(1,z); % plot the poles


Hord=figure('Name','The order selection of the Linear predictor');
for p=1:POrder
  AIC(p)= WL*log(MSEP(p))+2*p;
     MDL(p)=WL*log(MSEP(p))+p*log(WL);
   end
plot(AIC);hold on;
plot(MDL,'-.');hold on;
legend('AIC value','MDL value');
grid on;

% Draw the ACF of both BlkData and error_f;
Rxx=xcorr(BlkData);
Ree=xcorr(error_f);
figure('Name','The Autocorrelation of BlkData and error_f');
subplot(2,1,1);plot(Rxx);
title('The ACF of BlkData');
subplot(2,1,2);plot(Ree);
title('The ACF of error_f');
```