EFFECTS OF CHANNEL ERRORS ON CODED SPEECH COMMUNICATION IN SOFTWARE DEFINED RADIO

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
MASTER OF SCIENCE
in Electronics and Communication Engineering

by
Abbas KAGUDDE

December 2021
İZMİR

ACKNOWLEDGEMENT

First and foremost with all the love and adoration, I would like to thank God for giving me good health, understanding and a sound mind during this academic period.

It is always a great pleasure to remember the fine people in the engineering program of İzmir Institute of Technology, especially the Department of Electrical and Electronics for their sincere guidance, I received to uphold my practical as well as theoretical skills in engineering. Firstly, I would wish to thank my thesis advisor and academic parent, Prof. Dr. Barış Atakan of the Department of Electrical and Electronics at İzmir Institute of Technology, for his untimely dedication and selfless guidance. He always showed me positive attitude, allowed me to ask him different questions and giving me prompt replies and advice about this project. May God bless the work of his hands. I would also like to extend my sincere thanks to Prof. Dr. Bilge Karaçalı, the Head of Department of Electrical and Electronics Engineering at İzmir Institute of Technology. My special appreciations also go to both the teaching and non-teaching staff at the Department of Electrical and Electronics especially my course instructors: Assoc. Prof. Dr. Berna Özbek, Dr. Mehmet Zübeyir Ünlü, Prof. Dr. Mustafa Altınkaya, and Assoc.Prof. Dr. Şevket Gümüştekin. It was because of their tireless efforts and professional approach, that I completed this project.

I cannot forget to extend my sincere thanks to Turkey's Presidency for Turks Abroad and Related Communities (YTB), for giving me this wonderful opportunity to facilitate my education in this amazing country and more specifically at İzmir Institute of Technology.

I express my sincere thanks to everyone who has done a lot to see Kagudde Abbas excelling in his studies, more especially my late father Mustafa Kanoonya, my late paternal uncle Ahmed Kavulu and his wife, sisters, brothers and finally my wife Nansubuga Shaluwa. Thank you so much.

ABSTRACT

EFFECTS OF CHANNEL ERRORS ON CODED SPEECH COMMUNICATION IN SOFTWARE DEFINED RADIO

This thesis investigates the performance of software defined radio in reconstruction of a coded speech signal in presence of channel errors, taking into account end to end communication. At the transmitter, the recorded author's voice is encoded using linear predictive coding algorithm, where speech parameters such as linear predicted coefficients, pitch, voicing and gain parameters, are extracted from the speech signal. These parameters are sent to linear predictive decoder to model the speech signal from its parameters. The output from source encoder is sent to channel encoder such that, the digital encoded speech data is protected using linear block codes algorithm to provide error protection to bit stream before transmission to the communication channel. Receiver's blocks or algorithms to curb multipath interference, intersymbol interference, timing offset and carrier offset are based on adaptation. Steepest descent adaptive algorithm is used to design the entire algorithms to run the software receiver. Therefore, steepest descent algorithm is implemented in down conversion, carrier recovery, clock recovery, equalization and correlation. All algorithms running the software receiver are theoretically discussed and implemented in MATLAB software. The results obtained after simulating the whole receiver block in terms of symbol error rate, mean square error and bit error rate are recorded and analysed to investigate how channel errors affect software receiver while reconstructing a coded speech signal.

ÖZET

YAZILIM TANIMLI RADYODA KANAL HATALARININ KODLU KONUŞMA İLETİŞİMİNE ETKİLERİ

Bu tezde uçtan uca iletişim dikkate alınarak kanal hatalarının varlığında kodlanmış bir konuşma sinyalinin yeniden oluşturulmasında yazılım tanımlı radyonun performansı araştırılmaktadır. Vericide kaydedilen ses doğrusal tahmini katsayılar, perde, seslendirme ve kazanç parametreleri gibi konuşma parametrelerinin konuşma sinyalinden çıkarıldığı doğrusal tahmine dayalı kodlama algoritması kullanılarak kodlanmaktadır. Bu parametreler, konuşma sinyalini parametrelerinden modellemek için doğrusal kestirimci kod çözücüye gönderilir. Kaynak kodlayıcıdan gelen çıkış, dijital kodlanmış konuşma verileri, iletişim kanalına iletilmeden önce bit akışında hata koruması sağlamak için doğrusal blok kod algoritması kullanılarak korunacak şekilde kanal kodlayıcıya gönderilir. Alıcının çok yollu karışması, semboller arası karışması, zamanlama ofseti ve taşıyıcı ofsetini frenlemek için adaptasyona dayalı en dik iniş uyarlamalı algoritmalar kullanılmıştır. Yazılım alıcısını çalıştıran tüm algoritmalar teorik olarak tartışılarak MATLAB yazılımında uygulanmaktadır. Sembol hata oranı, ortalama kare hatası ve bit hata oranı açısından tüm alıcı bloğu simüle edildikten sonra elde edilen sonuçlar kaydedilerek kodlanmış bir konuşma sinyalini yeniden oluştururken kanal hatalarının yazılım alıcısını nasıl etkilediği analiz edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES vii

LIST OF TABLES.. ix

CHAPTER 1. INTRODUCTION 1

1.1 Background.. 1

1.2 Significance of the research.. 5

1.3 Objectives of the research... 5

1.4 Thesis Structure 5

CHAPTER 2. LINEAR PREDICTIVE CODING... 7

2.1 Human Speech Production.. 7

2.2 Speech Coding System. 7

2.3 Linear predictive coding algorithm... 9

2.3.1 Filter coefficients 10

2.3.2 Input signal 14

2.3.3 Signal Framing.. 15

2.3.4 Pre-emphasis and De-emphasis Filtering...................... 16

2.3.5 Windowing.. 17

2.3.6 Autocorrelation Analysis. 19

2.3.7 The Levinson–Durbin Algorithm analysis. 19

2.3.8 Pitch estimation... 20

2.3.9 Voicing detector.. 22

2.3.10 Structure of Linear Predictive Coding Algorithm 26

CHAPTER 3. END TO END COMMUNICATION...................................... 30

3.1 Steepest Descent Algorithm... 30

3.2 End to End Communication.. 33

3.3 Channel Coding. 33

3.3.1 Linear Block Codes Algorithm. 34
3.3.2 Hamming Codes Algorithm. 35
3.4 Channel 37
3.5 Pulse Shaping and Receive Filtering 38
3.6 Quadrature Amplitude Modulation (QAM) 40
3.6.1 Demodulation of QAM Signal 41
3.6.2 Carrier Recovery for QAM Signal 43
3.6.3 Correlation. 52

CHAPTER 4. METHODOLOGY 53
4.1 Channel Impairments and Disturbances 53
4.2 Source Coding Block 56
4.3 Channel Coding 58
4.4 Modulation 59
4.5 Channel 59
4.6 Carrier Recovery and Demodulation 60
4.7 Lowpass filtering 60
4.8 Matched filtering and Timing Recovery 61
4.9 Correlation 61
4.10 Equalization Block 62
4.11 Decoding 62

CHAPTER 5. RESULTS AND DISCUSSION 64
5.1 Source Encoder Results Analysis 64
5.2 Carrier synchronization results analysis 66
5.3 Clock Synchronization Results Analysis 68
5.4 Simulation results of software receiver 69

CHAPTER 6. CONCLUSION 74

REFERENCES 75

LIST OF FIGURES

Figure **Page**

Figure 2.1: Speech coded communication system 8
Figure 2.2: Model of speech generation mechanism 9
Figure 2.3: Linear prediction system identification 11
Figure 2.4: Block diagram of filter coefficient implementation 14
Figure 2.5: Time plot of original speech signal 15
Figure 2.6: Frequency spectrum of original speech signal 15
Figure 2.7: Frequency response o f filter at difference filter coefficients 16
Figure 2.8: Frequency spectrum of speech signal after pre-emphasis filtering 17
Figure 2.9: Time plot of original and windowed frame 18
Figure 2.10: Magnitude spectrum of original and windowed frame 18
Figure 2.11: Hamming window time plot 19
Figure 2.12: Voicing estimation based on pitch period 22
Figure 2.13: Classifying voiced and unvoiced frame. 22
Figure 2.14: Prediction error signal 25
Figure 2.15: A graph of prediction gain against prediction order 25
Figure 2.16: LPC encoder block diagram 27
Figure 2.17: LPC encoder block diagram 28
Figure 2.18: Time plot of original and compressed speech signal 29
Figure 3.1: Error surface for objective function 32
Figure 3.2: Error surface for objective function 32
Figure 3.3: Pulse shaping and Receive filtering block 38
Figure 3.4: QAM transmitter-Receiver block. 40
Figure 3.5: Signal demodulation block 42
Figure 3.6: Cosine curve 45
Figure 4.1: Time plot of timing jitter. 54
Figure 4.2: Magnitude spectrum of ISI 56
Figure 4.3: Time plot of Original and Compressed speech signal 57

Figure **Page**

Figure 4.4: Simulator block 58

Figure 4.5: Magnitude spectrum of up converted and original signal 59

Figure 4.6: Magnitude spectrum of demodulated and low passed signal 61

Figure 4.7: Performance evaluation. 63

Figure 5.1: Magnitude spectrum of both original and synthetic speech signal 65

Figure 5.2: Time plot for original and synthetic speech signal 66

Figure 5.3; Phase offset estimation.. 67

Figure 5.4: Timing offset estimation 69

Figure 5.5: Symbol error rate and bit error rate against signal to noise ratio 73

LIST OF TABLES

<u>Table</u> **<u>Page</u>**

Table 3.1: (7,4) Hamming coding system.. 35

Table 3.2: Loop up table for (7,4) Hamming code 37

Table 5.1: Table of results one.. 70

Table 5.2: Table of results two 71

Table 5.3: Table of results three 72

NOMENCLATURE

FPGA	File Programmable gate way Array.
GPP	General Purpose Processor.
RF	Radio Frequency.
IF	Intermediate Frequency.
GPS	Global Positioning System.
ADCs	Analog to Digital Converters.
DRFS	Direct Radio Frequency Sampling.
GNSS	Global Navigation satellite system.
SDR	Software Defined Radio.
LPC	Linear Predictive Coding.
LPCs	Linear Predictive Coefficients.
AR	Auto Regressive.
ISI	Intersymbol Interference.
SRRC	Square Root Raised Cosine.
QAM	Quadrature amplitude modulation.
AM	Amplitude Modulation.
AWGN	Additive White Gaussian Noise.
LMS	Least Mean Square.
SER	Symbol Error rate.
BER	Bit error Rate
MSE	Mean Square Error.
SNR	Signal to Noise Ratio.
MSF	Magnitude Sum Function.

CHAPTER 1

INTRODUCTION

Most software receivers or radios have a programmable processing unit onto which the received analog signal is processed before reconstruction. More specifically, the received signal from antenna is connected to programmable processing unit which processes the received signal. Receiver's operations such as demodulation, carrier recovery, time recovery, sampling, correlation and equalization run on the same hardware component. In turn, software receiver reduces hardware components since only a processing unit like FPGA onto which software can run is needed [1]. Therefore, software receiver should be designed and implemented in a sense to model all the incoming radio frequency signals, and also in position to sample the incoming signal and processes it without degrading its strength.

Since technology is evolving at a high speed for example from 2G,3G, 4G and 5G. All these technological changes call for replacement of some hardware components at the receivers' ends. Introduction of software receiver solutions provides full control of all receiver operations, in order to ease work for designers, engineers and researchers in testing and implement their algorithms, without permanently replacing hardware components in case of any change in technology. Finally, software receivers should be flexible, upgradable and configurable without entirely changing their hardware components for any change in technology or communication standards [1].

1.1 Background

Over the last few years, the existence of new processors with high computation power made it easy for software receivers to be invented whose performance is better than that of fully hardware based in line with flexibility, configurability, upgradability and versatility. In conjunction with software receivers, the main challenge is how a signal in presence of channel errors can be reconstructed perfectly without degrading its strength

[2]. This is because, as a signal propagates from a transmitter to a receiver, it undergoes a lot of channel impairments which may include, flat fading, multipath interferences and additive noise which can be contributed by both narrow and wide band noises.

Receivers based on field-programmable gate arrays (FPGAs) have been developed in the recent years, and such solution provides a good trade-off between the flexibility of the software receivers [2].

In GNSS environment, the design and implementation of a software receiver architecture was studied [3]. Here the incoming analog signals were directly sampled by separate analog to digital converter device. Digital signal processing was implemented on FPGA which was connected to PC board to perform other software requirements. Successfully a simple software-based receiver was designed and implemented to provide an over view on GNSS evolution. It was concluded that, there was a need to design high performance software receivers for specific applications due to rapid technological changes and these software receivers will be playing a vital role in technological advancements.

In software receivers environment, a digital receiver implementation using FPGA was studied [4]. The received signal was directly sampled by a separate ADC with sampling frequency of 60Mbps, the sampled signal was down converted using digital mixer, digital local oscillator and then decimation low pass filter. Then, ADC board was connected to FPGA board for digital signal processing, or all software processes of the received signal were running on the same board. The computer simulations were obtained during each stage and hardware implementation was done using FPGA board. Due to time varying channel and other related interferences a signal encounters as it propagates from transmitter to receiver, there should be algorithms to synchronize the transmitter and the receiver for proper sampling and demodulation of the signal.

The operation, challenges and possible solutions of software defined radio were studied in [5]. Here signal processing, channel selection, modulation and demodulation were all done digitally through software. The received signal was first down converted by analog circuitry of mixers and local oscillators to obtain a lower intermediate frequency signal which was then sampled using analog to digital converter. The digital signal processing software was running on a hardware device, i.e, FPGA. It was also ascertained that, even though FPGA are very computationally powerful and very efficient, but they are not flexible. However due to high level of flexibility and reconfiguration requirements while implementing software receivers, GPP is a better solution as compared to FPGA. This study concluded that, for processing a received signal, there should be a hardware device onto which software is running, and this software should be upgradable in case of any change in technology without prior replacement of a hardware device.

In GPS environment, the applicability and implementation of software receiver was studied in [6]. The GPS signal was received by antenna which then down converted using analog circuitry to obtain a lower version of frequency signal. This work mainly focused on the components that should be applied while implementing the software receiver to process satellite signals. The signal was sampled using analog digital converter and then the samples were processed using Matlab. In conclusion, GPS software receiver should be implemented to allow re-programmability and flexibility such that the receiver can be able to process more navigation signals.

Software receivers are of great importance due to the fact that they are flexible, and their flexibility justifies the fact that they are used in many applications which include military, satellite, radar and also modern cellular technology. Some of the advantages are studied in [5][1].

- Interoperability. A software receiver can be configured to accommodate all wireless communication standards. Software receivers can also be configured to receive signals at different frequencies.
- Numerous functions. The flexibility of software receiver architecture allows multiple communications standards. Therefore, it can be easy to introduce in other applications which can be handled by receivers which include blue tooth, GPS application and other applications.
- Software receivers are easy to be manufactured. Since software receivers can be upgraded by only code updating without necessary replacement of hardware components, it reduces the complexity in designing.
- Software receivers support cognitive radio technology. In case of un-utilized or under-utilized spectrum, cognitive radios sense the presence of that spectrum which can be used by a secondary user for transmission due to the absence of primary users. Therefore, if the primary user is not currently using the spectrum,

software receiver can utilize it until when the primary user needs it again, this improves on the optimality of the available inadequate spectrum.

- High level of upgradability. In case of new technology or any change in standards, software receivers can be configured by coding updating to suit the ongoing technology without permanently replacing the hardware components.
- Since software receivers typically have less analog components, this translates into a reduction in cost of materials, size, power consumption, design time and weight.

However, there are some challenges or limitations to applications of software receiver which include;

- Security problems. A signal undergoes a lot of interferences and security concerns as propagates from transmitter to receiver. Since software is only configured to upgrade receiver, this software may have security threats which can corrupt the whole receiver's architecture.
- Complexity in writing the software for certain systems. This may come up with code writing which may consume a lot of time to finally come with a genuine and secure code.
- High level of sampling or high-speed ADCs. Since direct sampling of incoming RF signal can be fundamental with some software receiver, sampling may be very difficult to implement if very high frequency signals are directly sampled.

In conclusion, it can be conceived from the previous review that tremendous work has been done on software defined radios, but most studies are based on non-real signals with an assumption that these signals are already sampled and readily available at the receiver.

In this thesis therefore, the focus is directed towards investigating the effects of channel errors in reconstruction of a coded speech signal putting into account end to end communication. The attention is put on encoding and decoding human voice using linear predictive coding algorithm, and also designing and simulating the software receiver, where some of its algorithms are derived using steepest descent adaptive algorithm. Hence, the main algorithms that run the software receiver are based on adaption and implemented in Matlab software. The results obtained after simulating transmitter-receiver block are analyzed to assert how channel errors affect the reconstruction of a speech signal from its parameters in software defined radio environment.

1.2 Significance of the research

This research can be a source of literature to the overall research that is being undertaken today, due to the continuous need for new technologies with greater capacity, reduced cost and size as well as improved reliability that is needed to be employed in communication systems and networks like 5G, in order to facilitate reliable communications. This is so because the research gives;

- An in-depth study of different algorithms used software receiver radio.
- Theory, simulation, and implementation of software defined receiver.
- A comprehensive study on how Software defined radio can be affected by channel errors while reconstructing a speech signal from its parameters.
- Nonetheless, it will also widen and enrich the researcher's knowledge as far as end to end software receiver's communication is concerned thereby promoting the feel for innovativeness and creativity.

1.3 Objectives of the research

The main objective of this thesis was to investigate the effects of channel errors on coded speech signal communication in software defined radio. Moreover, the specific objectives of this thesis include the following.

- To study designing approach of software defined radio.
- To determine algorithms in encoding and also in synchronization.
- To model and simulate software defined radio architecture.
- To evaluate effects of errors on performance of SDR in speech reconstruction.

1.4 Thesis Structure

The thesis is structured in six chapters as explained below;

Chapter 1: It gives an introduction, background information, significance of this study as well as research objectives are given in this chapter. Nonetheless, previous research done on software receivers was also reviewed.

Chapter 2: It expounds on linear predictive coding and how different modules of the algorithm can implemented in Matlab software.

Chapter 3: It expounds on end to end communication in software defined radio environment.

Chapter 4: It gives a detailed account on the system simulation of software defined radio using Matlab software.

Chapter 5: It discusses findings and analyzes results

Chapter 6: It gives the concluding remarks about this study and also recommendation for future study

CHAPTER 2

LINEAR PREDICTIVE CODING

In this chapter, linear predictive coding algorithm was discussed, taking into account the theoretical and Matlab implementation of its algorithms. The MATLAB software program was used to practically implement and illustrate the main components of the algorithm.

2.1 Human Speech Production

All over the world people are speaking different languages, interestingly you may find in a single country people are speaking different languages. However the way how sound is produced in humans is the same irrespective of their differences in languages. For sound production in humans, air is pushed out of the lungs, passes through the vocal tract and finally pushed out of mouth to produce sound as explained in [7]. Therefore, lungs is where sound originates from, in other words it is the source of sound.

2.2 Speech Coding System

Speech coding is the way of representing speech (audio) signal with fewer number of bits without degrading its quality [8]. From Figure 2.1, the analog speech signal is filtered by a filter, which is then converted into a discrete signal by a sampler, after uniform quantization, the discrete signal is converted into a digital signal by analog digital converter. The output signal from analog to converter is called a digital speech signal [7].

Most of speech coding algorithms were designed to operate within the frequency range between 300 Hz and 3400 Hz [7]. According to Nyquist sampling theorem, “sampling frequency should be greater than or equal to twice the maximum frequency to avoid aliasing” [7]. This explains why, a sampling rate of 8 kHz is usually chosen for speech signals. For example, let us consider a speech signal being sampled at a rate of

8000 Hz, if each sample has got 8 bits, the bit-rate at the input of source encoder will be equalling to 64 kbps. This bit-rate of 64 kbps is reduced by the source encoder (speech coding algorithm), hence the speech signal at the output of source encoder is represented by a fewer number of bits [10].

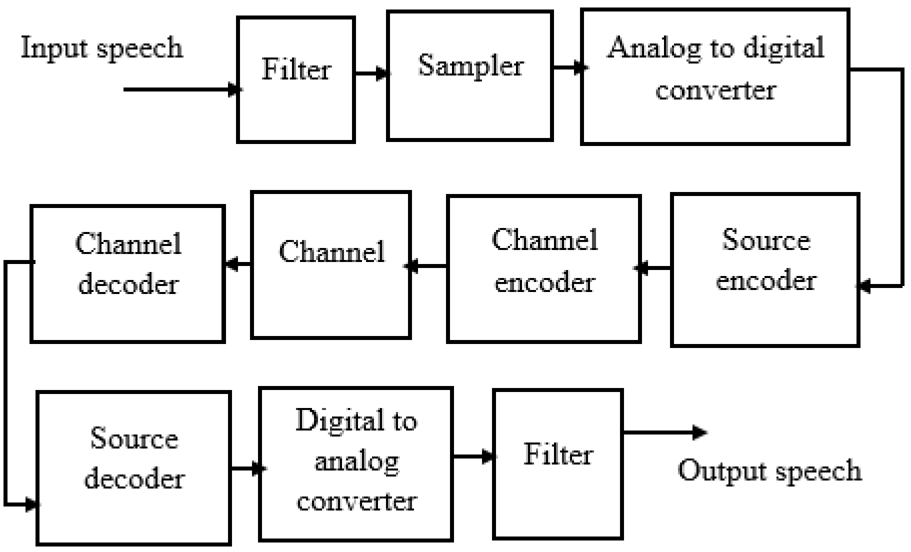


Figure 2.1: Speech coded communication system [7]

Speech coding algorithms effectively compress speech signals into a fewer number of bits at a reasonable quality [7]. There are a number of algorithms upon which speech coding can be done. These algorithms are like computational methods that take in input speech signals at a high bit rate, and produce output signals at a lower bit rate level. Some of speech coding algorithms include [8][13];

- Linear predictive coding
- Waveform coding
- Code excited linear predictive coding.

Speech coding algorithms are classified as lossy and lossless coding algorithms depending on the quality of a produced signal at the output of the decoder. For lossless speech coding algorithms, the processed speech signal at the output of the decoder has the quality which is almost the same as the original input speech signal. However, for lossy coding techniques, the quality of compressed speech signal reduces at the out of the decoder. For this thesis, linear predictive coding algorithm was taken into account.

2.3 Linear predictive coding algorithm

Linear predictive coding is one of the earliest coding algorithms which operates at a very low bit rate and is widely used in audio signal processing. It was first developed to ensure security in communication for military applications by the United States Department of Defence in federal standard 1015, published in 1984 [7][10][11].

Linear predictive coding is used as a compression algorithm at a source encoder, the output of the encoder has a bit rate of 2.4 kbps [7][9]. The output of linear predictive encoder has an output rate of 2.4 kbps which is approximately 27 times less than the input bit rate of 64 kbps with an assumption that the bit rate at the encoder’s input is 64 kbps. The quality of reconstructed speech is reduced at the output decoder hence linear predictive coding is a lossy compression technique. Linear predictive coding algorithm is influenced by a model shown in Figure 2.2, to produce speech signals [7]. This model represents basic properties of speech signals and also represents a mechanism of sound production in humans.

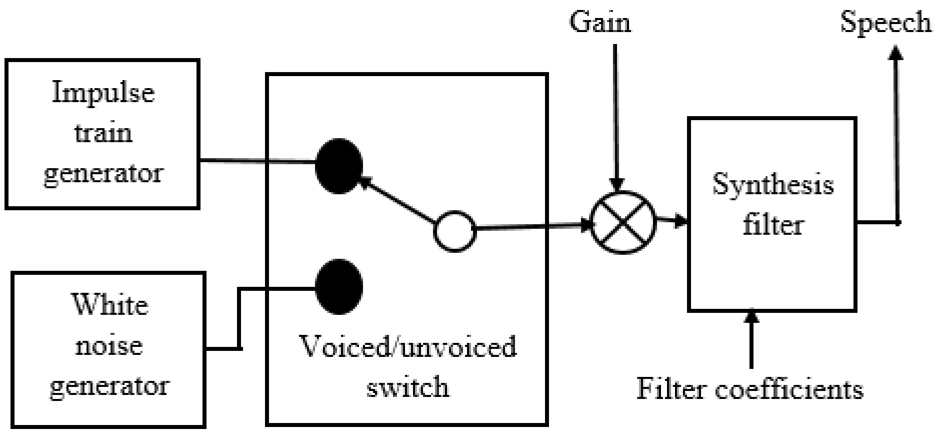


Figure 2.2: Model of speech generation mechanism [7][22].

From Figure 2.2, a synthesis filter represents the role played by the vocal tract, glottal flow and radiation of the lips in human sound production. The input (excitation signal) to this filter can be taken as random noise (white noise) or pulse train. Since a speech signal can be voiced or unvoiced, the type of excitation depends on which type of speech signal needed. For voiced speech signal, periodic sequence of impulses at frequency f can be used as a model for excitation. Note that, the inverse of frequency f is called pitch period. For unvoiced speech signal, white noise or random noise can be used as a model for excitation as explained in [13][22]. Therefore, the purpose of the switch is to select the desired input depending on the voiced and unvoiced nature of the signal. Finally, the goal of the gain parameter is to control the energy level of the output.

Speech signals constantly change over time, hence it is necessary to be divided into frames using a small time duration. If a small time duration is chosen, speech signal parameters during that duration essentially remain constant [7]. In each and every frame, parameters of the model are specified from the speech samples. These LPC parameters extracted from the signal frame(s) consists of, gain, filter coefficients, voicing and pitch period parameters. The gain parameters detail the energy level of the frame, filter coefficients are related with the response of synthesis filter, voicing (voicing detector) indicate if the speech signal is voiced or unvoiced and finally pitch period is related with time length between consecutive periodic sequence of impulses [7]. These parameters are explained as follows;

2.3.1 Filter coefficients

A vocal tract model shown in Figure 2.2, can be modelled as an all pole filter having a difference equation shown in Eq. (2.1) [22].

$$s[n] = \sum_{i=1}^{M} a_i s[n-i] + Gx[n], \tag{2.1}$$

given that $x[n]$ is excitation (white noise or impulse), G is the gain parameter, $\{a_i\}$ are filter coefficients and M is the number of poles of the filter.

Between pitch pulses, $Gx[n]$ is zero [7]. Therefore, from Eq. (2.1), $s[n]$ can be anticipated as linear combination of previous samples hence linear prediction. However, if $Gx[n]$ is included, then $s[n]$ can be predicted approximately [7]. Linear prediction is used to determine filter coefficients $\{a_i\}$ at the encoder. The filter coefficients can also be taken as linear predicted coefficients (LPCs). LPCs determination is also very important since they are used to reconstruct a synthesis filter [12].

2.3.1.1 Linear prediction

In most cases, speech coding algorithms rely on linear prediction while analysing speech signals [13]. The most important point to note about linear prediction is that, a current speech sample can be obtained as a linear combination of the past speech samples, i.e, the incoming sample can be predicted by the previous obtained samples hence the name predictive [14]. Linear prediction can also be seen as a procedure of removing redundancy from the speech signal where repeated information in a signal is removed. By removing redundancy from the information signal, the number of bits to carry the signal is reduced hence compression attained [7].

Linear prediction can be taken as a system identification problem by estimating parameters of Auto Regressive (AR) model from the signal itself which is shown in the Figure 2.3.

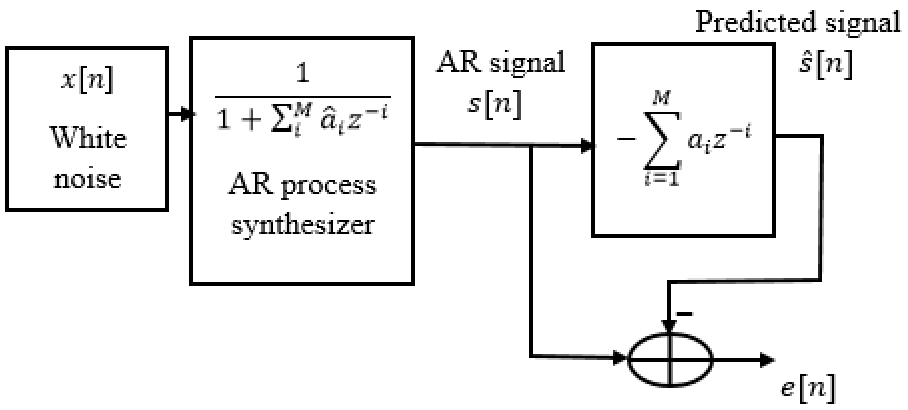


Figure 2.3: Linear prediction system identification [7].

The signal $x[n]$ also known as white noise is filtered by AR process synthesizer to give AR signal $s[n]$. The linear predictor predicts signal $s[n]$ based on past M samples and predicted signal $\hat{s}[n]$ is achieved as shown in Eq. (2.2) [7].

$$\hat{s}[n] = -\sum_{i=1}^{M} a_i s[n-i], \tag{2.2}$$

where M is a constant which is well known as prediction order and a_i are linear predicted coefficients which are estimated by AR parameters. Therefore, a current predicted signal is approximated as a linear combination of the past signal samples hence the linear prediction. The difference between the actual and predicted signal sample gives prediction error $e[n]$ as shown in Eq. (2.3).

$$e[n] = s[n] - \hat{s}[n] \tag{2.3}$$

The mean square error gives a good approximation of AR parameters $\hat{a}_i$ from signal $s[n]$ and the approximated coefficients are known as linear predicted coefficients. Minimizing the mean square error Eq. (2.4) over a short time provides a way of determining LPCs [7].

$$J = E\{e^2[n]\} = E\left\{\left(s[n] + \sum_{i=1}^{M} a_i s[n-i]\right)^2\right\} \tag{2.4}$$

One way of obtaining optimal values of LPCs is to find the derivative of Eq. (2.4), and equate it to zero [15]. Therefore, the derivative of J with respect to a_i is shown in Eq. (2.5), and the end result is equalling to zero.

$$\frac{\partial J}{\partial a_k} = 2E\left\{\left(s[n] + \sum_{i=1}^{M} a_i s[n-i]\right) s[n-k]\right\} = 0, \tag{2.5}$$

for $k = 1,2, \dots \dots . . M$. If Eq. (2.5) is satisfied, then $a_i = \hat{a}_i$, which means that AR parameters $\hat{a}_i$ are equivalent to LPCs a_i.

Eq. (2.5) is re-arranged to give

$$E\{s[n]s[n-k]\} + \sum_{i=1}^{M} a_i E\{s[n-i]s[n-k]\} = 0 \tag{2.6}$$

$$\sum_{i=1}^{M} a_i R_s[i-k] = -R_s[k], \tag{2.7}$$

for $k = 1,2,3 \dots \dots . M$, where,

$$R_s[i-k] = E\{s[n-i]s[n-k]\}, \tag{2.8}$$

$$R_s[k] = E\{s[n]s[n-k]\}. \tag{2.9}$$

The linear predicted coefficients a_i defined in Eq. (2.7) are defined in terms of autocorrelation sequences $\mathrm{R}_s[l]$ of the signal $s[n]$. In addition, Eq. (2.7) can be expressed as,

$$R_s a = -r_s, \tag{2.10}$$

given that,

$$R_s = \begin{bmatrix} R_s[0] & \cdots & R_s[M-1] \\ \vdots & \ddots & \vdots \\ R_s[M-1] & \cdots & R_s[0] \end{bmatrix}, \tag{2.11}$$

$$a = [a_1 \;\; a_2 \;\; a_3 \dots \dots . a_M]^T, \tag{2.12}$$

$$r_s = \left[R_s[1]\;\; R_s[2]\;\; R_s[3] \ldots \ldots R_s[M]\right]^T. \tag{2.13}$$

The linear predictive coefficients in column matrix (a) can be obtained by obtaining the inverse of matrix R_s as shown in Eq. (2.14).

$$a = -R_s^{-1} r_s \tag{2.14}$$

Assuming the autocorrelation values of signal $s[n]$ are given from $l = 0$ to M, then LPCs can be found from Eq. (2.14).

2.3.1.2 The Levinson–Durbin Algorithm

The Levinson Durbin algorithm's importance lies in its computational efficiency to overcome the computational complexity of calculating LPCs using inverse of matrix R_s. However, from Eq. (2.14), LPCs in column matrix a can be obtained by matrix inversion even though it has a high computational complexity. Therefore, autocorrelation sequences are the inputs to Levinson Durbin algorithm, thereby computing LPCs effectively and efficiently [7].

This algorithm defines the solution up to M^{th} prediction order from that of $(M-1)^{th}$ prediction order and it is an iterative method. Therefore, the solution of first prediction is first found which can be used to compute the first order predictor and the iteration continues until the final solution is determined. The autocorrelation sequences $R_s[l]$ are used as input to Levinson Durbin algorithm and output being reflection coefficients (RCs) and linear predictive coefficients (LPCs).

The Levinson Durbin algorithm is described as follows [7].

- Initialize $l = 0$, and set $J_0 = R[0]$, where J is the minimum mean squared prediction error.
- It is a repetitive procedure for $l = 1,2, \ldots \ldots . . M$.

1. The l^{th} reflection coefficients (RCs) are computed as

$$k_l = \frac{1}{J_{l-1}}\left(R[l] + \sum_{i=1}^{l-1} \mathrm{a}_i^{(l-1)}\, R[l-i]\right). \tag{2.15}$$

2. The l^{th} prediction order LPCs a_i are calculated

$$a_l^{(l)} = -k_l, \tag{2.16}$$

$$a_i^{(l)} = a_i^{(l-1)} - k_l a_{l-i}^{(l-1)};\;\; i = 1,2, \ldots . l-1. \tag{2.17}$$

The iteration ends when $l = M$ where M is the prediction order.

3. The minimum mean squared prediction error associated with l^{th} prediction order is computed as

$$J_l = J_{l-1}(1 - k_l^2). \tag{2.18}$$

Set $l \leftarrow l + 1$; then return to 1

Then the final LPCs are

$$a_i = a_i^{(M)}; i = 1,2,3, \dots \dots . M. \tag{2.19}$$

In Matlab software, LPCs (filter coefficients) were extracted from a speech signal following the block diagram shown in Figure 2.4 below.

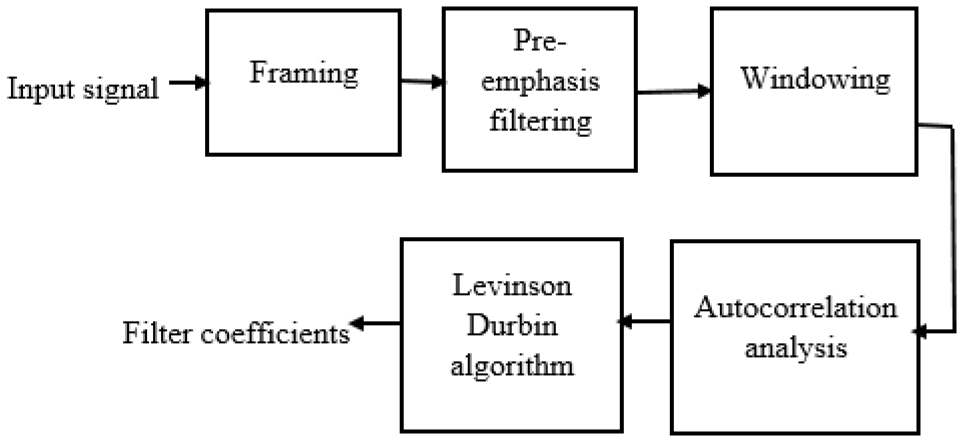


Figure 2.4: Block diagram of filter coefficient implementation

2.3.2 Input signal

The input signal was the author's voice which was recorded by calling Matlab built in function *audiorecoder* and passed three arguments which included sampling rate (frequency), sample size, and number of channels. As explained in [7], speech coding algorithm are mostly designed to operate within the range of 300 Hz and 3400 Hz. Therefore, following Nyquist sampling criteria, a sampling rate of 8000 Hz was chosen to avoid aliasing. The choice for the number of bits per sample depends on the installed hardware, a Matlab default value of 8 bits per sample was used and finally the number of channels chosen was one since mono sound signal was considered for this case. The duration taken for recording was 10 seconds which was implemented in Matlab by calling a Matlab built in function *recordblocking,* in which the created object and the duration of 10 seconds were passed. To have the recorded data in terms of numerical array, a

Matlab built-in function $getaudiodata$ was used which passed the recorded data and the data type ($uint8$). Once the numerical array is obtained, the audio signal was saved for further processing by calling a Matlab built-in function $audiowrite$ which passed the name representing the recorded audio signal, the recorded object and the sampling rate used. Since the time taken to record the voice was 10 seconds at the sampling frequency of 8000 Hz, a total of 80000 samples were obtained each having 8 bits. The time plot and magnitude spectrum of the recorded audio signal is shown in Figure 2.5 and Figure 2.6 respectively.

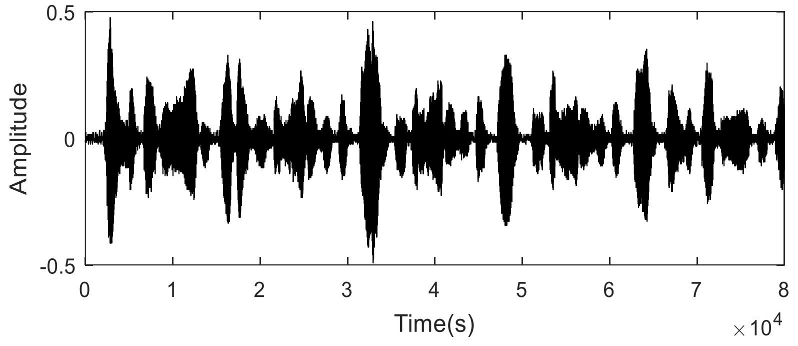


Figure 2.5: Time plot of original speech signal

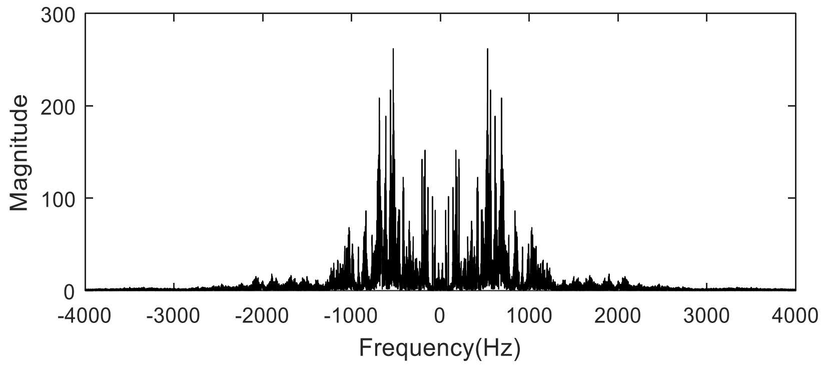


Figure 2.6: Frequency spectrum of original speech signal

2.3.3 Signal Framing

Speech signals constantly change over time, hence it is necessary to be divided into frames using a small time duration. If a small time duration is chosen, speech signal parameters during that duration are assumed to be stationary (constant) [7]. Using 30 ms frame duration, an original speech signal was segmented into frames each having approximately 240 samples.

2.3.4 Pre-emphasis and De-emphasis Filtering

High frequency components at a very low amplitudes impose themselves in speech signals. However, these components contain information which is vital in understanding the speech signal. The pre-emphasis filter passes high frequency components, thereby boosting their amplitudes.

Let us assume a signal being filtered by a filter of impulse response;

$$h[n] = \delta[n] - a\delta[n-1], \tag{2.20}$$

given that its frequency response is

$$H\left(e^{jw}\right) = 1 - ae^{-jw}, \tag{2.21}$$

and its magnitude response is

$$\left|H\left(e^{jw}\right)\right| = \sqrt{1 + a^2 - 2\mathrm{acos}w}, \tag{2.22}$$

given that

$$e^{-jw} = \mathrm{cos}w - j\mathrm{sin}w. \tag{2.23}$$

Eq. (2.22) is then further expressed in decibels (dB) as shown in Eq. (2.24) and the value of filter coefficient a is in the range of $0.9 \le a \le 1$ [7][16].

$$\left|H(e^{jw})\right|_{\mathrm{dB}} = 20\log_{10}\sqrt{1 + a^2 - 2a\mathrm{cos}w}. \tag{2.24}$$

The magnitude response of a pre-emphasis filter was plotted in Matlab software at different values of filter coefficients as shown in Figure 2.7. However, the magnitude response could also be obtained in Matlab software by calling a Matlab built-in command $freqz$.

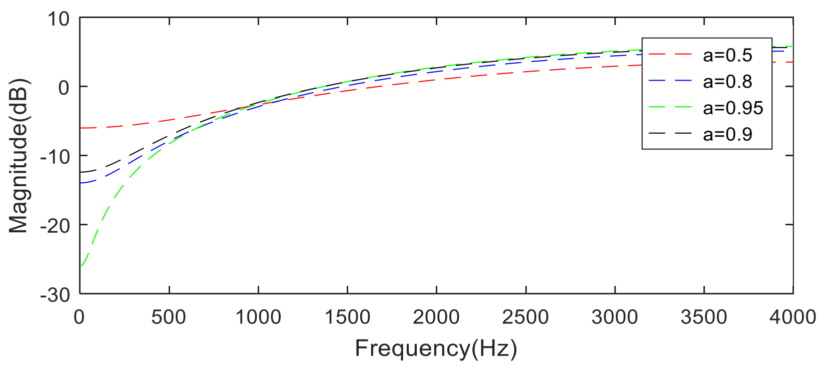


Figure 2.7: Frequency response of filter at difference filter coefficients

The value of 0.97 is taken as the pre-emphasis filter coefficient a. The speech signal is pre-emphasis filtered and a boost in high frequency components is seen as shown in Figure 2.8 below.

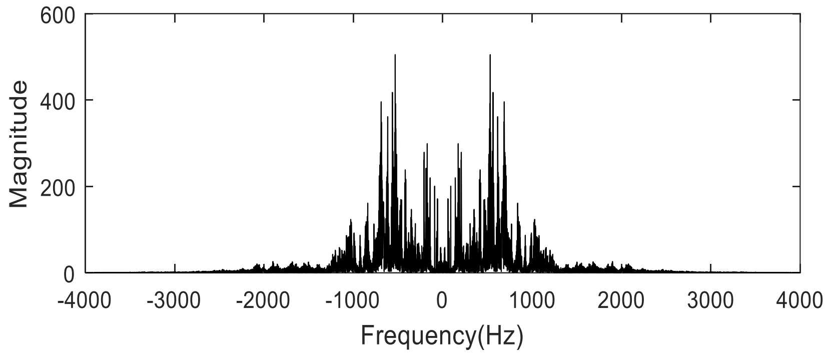


Figure 2.8: Frequency spectrum of speech signal after pre-emphasis filtering

The high frequency components are boosted during pre-emphasis filtering. Therefore, de-emphasis filtering which is the opposite with respect to pre-emphasis is done at the decoder to attenuate these components. De-emphasis is mainly done to maintain the spectral shape of the reconstructed speech as that of original speech [7]. The frequency response $H(e^{jw})$ of de-emphasis filter is expressed in Eq. (2.25). Therefore, the value of the filter coefficient (0.97) used at the encoder should be the same value used when implementing de-emphasis filtering at the decoder [7].

$$H\left(e^{jw}\right) = \frac{1}{1 - \mathrm{a}e^{-jw}} \tag{2.25}$$

2.3.5 Windowing

Dividing the original speech signal into frames, caused the borders of the frames have some discontinuities. Hamming window was applied in time domain to reduce this effect such that, speech signal statistical parameters are not negatively affected [13]. Figure 2.9 shows the time plot of a signal frame before and after applying Hamming window. Hamming windowing is implemented in Matlab software by calling a Matlab built-in function $hamming$. After applying a Hamming window (time plot is shown in Figure 2.11) to the signal frame, the middle portion was the same but amplitudes at the ends shrunk in larger amounts and no sharp changes at the ends of the signal frame, and also the signal frame after windowing was seen to be smooth for further analysis.

However, when Hamming window was applied in frequency domain, there were also changes in spectral domain. Figure 2.10 shows magnitude spectrum of a signal before and after applying a Hamming window. The red plot shows the spectrum after applying the window and the blue plot indicates the spectrum of a signal frame without windowing. The frequency components with higher frequency components goes down due to sharp truncation of hamming window. Therefore it seems like high frequency components are present in the signal frame but in real are not, hence by applying Hamming window, this effect was reduced. The Hamming window is defined in Eq. (2.26) as explained in [7].

$$w[n] = \begin{cases} 0.56 - 0.46\cos\left(\frac{2\pi n}{M-1}\right), & 0 \le n \le M-1 \\ 0\,, & \text{otherwise} \end{cases}, \qquad (2.26)$$

where M is the length of the window.

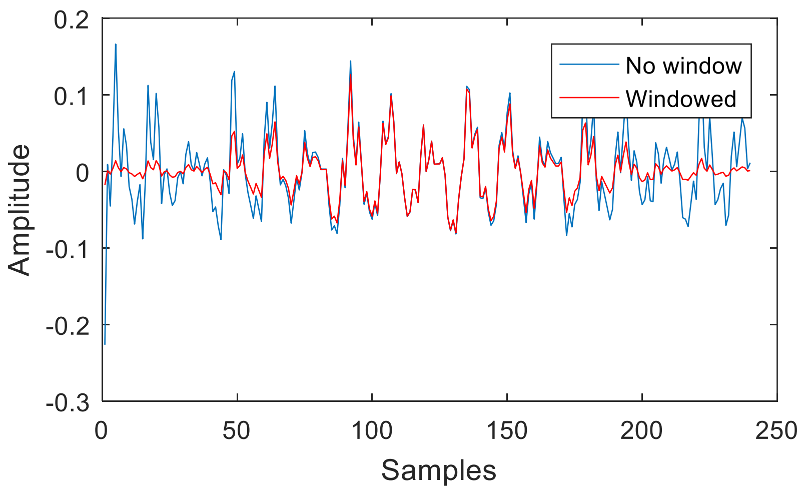


Figure 2.9: Time plot of original and windowed frame

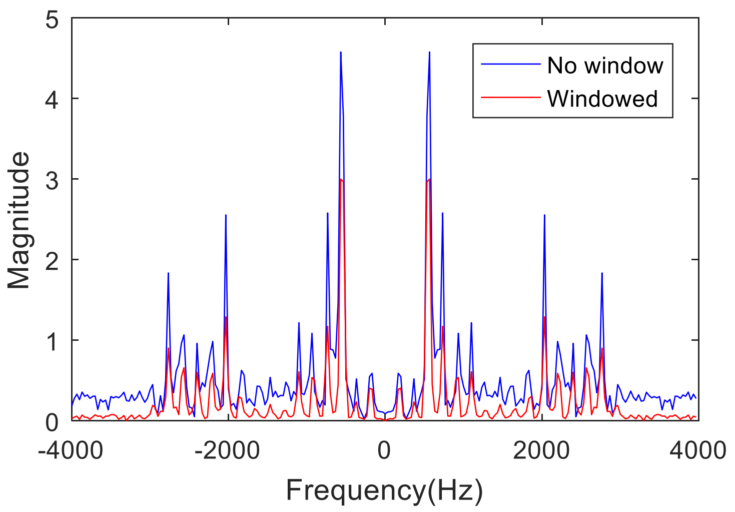


Figure 2.10: Magnitude spectrum of original and windowed frame

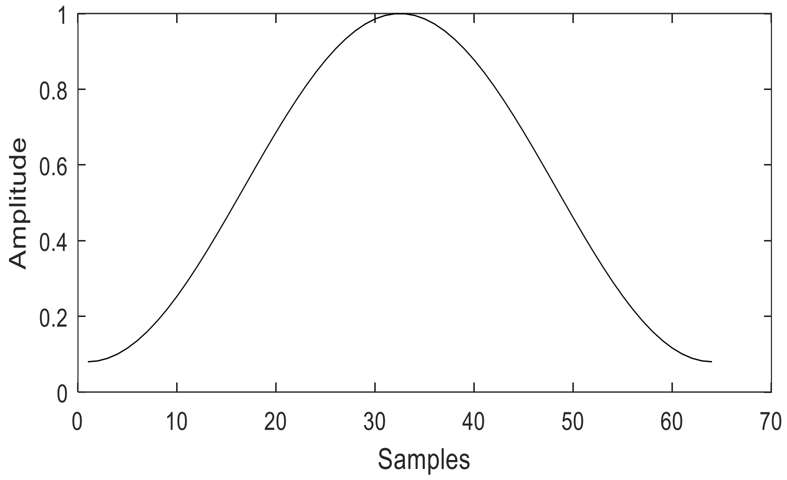


Figure 2.11: Hamming window time plot

2.3.6 Autocorrelation Analysis

Calculation of autocorrelation is a crucial step while estimating linear predictive coefficients (LPCs). For each frame, the prediction order $(M = 10)$ was selected and autocorrelation sequences $R_s[1], \dots \dots \dots., R_s[10]$ were required for each and every frame [7] [17]. A total of 10 autocorrelation sequences were extracted from a single signal frame by calling a Matlab built-in function $xcorr$ which takes in a signal frame as its input argument and returned auto correlation sequences as its output. The values of autocorrelation sequences obtained were compared with those obtained when expression in Eq. (2.7) was implemented in Matlab software. The same values of autocorrelation sequences were obtained in both cases.

2.3.7 The Levinson–Durbin Algorithm analysis

The autocorrelation analysis was done on a signal frame and linear predictive coefficients (a_i) were computed with prediction order $(M = 10)$ using Levinson Durbin recursive algorithm and Matlab built-in commands. Linear predicted coefficients were obtained in Matlab software for a single frame in two ways. The first way was completed by calling a built-in Matlab function lpc which takes in a signal frame and prediction order and finds the coefficients. A total of 10 linear predicted coefficients $(a_1, a_2 \dots \dots \dots. a_{10})$ were extracted to represent a signal frame, and these coefficients were the filter coefficients used to come up with the estimated signal. Finally LPCs were also computed using Levinson Durbin recursive algorithm which takes in autocorrelation sequences as its inputs. Expressions explained from Eq. (2.15) up to Eq. (2.19) are implemented in Matlab software to obtain LPCs. Therefore, instead of transmitting the whole signal samples, the linear predictive coefficients were transmitted to the decoder which act as input to synthesis filter block.

2.3.8 Pitch estimation

Fundamental frequency or pitch period of the voiced speech signal is a very vital parameter in speech analysis and synthesis. Voicing originates from lungs when movements of vocal cords periodically interrupt airflow. Note that, fundamental period or pitch period refers to the time between consecutive vocal cord openings. Theoretically, fundamental frequency for male is the range of 50 and 250 Hz while for that of female is between the range of 120 and 500 Hz [7]. However, the pitch period for men is between 4 to 20 ms and that for women is between 2 to 8 ms. For linear prediction, a current sample is obtained as a linear combination of previous speech samples. Therefore, it can be possible to determine the pitch period in which the signal repeats itself. However, meaningful pitch estimation results are obtained from voiced speech signals otherwise meaningless [7]. There are various methods or algorithms that can be used to estimate pitch period. Autocorrelation technique as a method of estimating pitch period is discussed as follows;

2.3.8.1 Autocorrelation Method of Pitch Estimation

Consider a signal $s[n]$, where n is the time index. Let us also consider a frame that ends at time instant x and the length of the frame taken as N. The autocorrelation value can be obtained as shown in Eq. (2.27) [7].

$$R[l,x] = \sum_{n=x-N+1}^{x} s[n]s[n-l] \tag{2.27}$$

Therefore, the autocorrelation values shows the relationship between the frame $s[n]$, $n = x - N + 1$ to x with respect to a time shifted frame $s[n-l]$, given that l is the time lag. The lag range should be chosen such that it covers a wide range of pitch period values. For example, for l=20 to 147 (2.5 to 18.3 ms) the possible frequency range values are in range between 54.4 to 400 Hz at a sampling rate of 8000 Hz [7]. Note that, while calculating autocorrelation values for entire lag, a lag associated with highest autocorrelation can be found. The highest autocorrelation is an estimate for pitch period. In otherwise, at a point when autocorrelation is maximum, its lag is equivalent to pitch period [7].

In Matlab software [9], pitch period of a signal frame was estimated considering a man's voice recorded at a sampling frequency $f_s = 8000$ Hz. As illustrated in [7], the pitch period for a man's voice frame is between 4 ms to 20 ms. The points in 4 ms and 20 ms were estimated as $(p_{min} = 4 \times 8000)$ and $(p_{max} = 20 \times 8000)$ respectively. A Matlab built-in function $xcorr$ was called and passed a signal frame as its input and returned autocorrelation sequences R. A Matlab function max was called and passed R and returned the maximum peak of R and its midpoint R_{mid}. The pitch period range p_{range} between $(p_{min} + R_{mid})$ to $(p_{max} + R_{mid})$ was chosen to cover a wide range of pitch period values. A Matlab built in function max was called which passed p_{range} and its output was added to p_{min} to end up with pitch period which was sent to the decoder.

However, pitch period estimated values were also used to classify if the signal was voiced or unvoiced depending on the value of threshold taken. A threshold value was computed from the pitch period values estimated and the frame below it was taken to be unvoiced otherwise voiced as shown in Figure 2.12, where voiced frames are of higher magnitudes as compared to unvoiced ones.

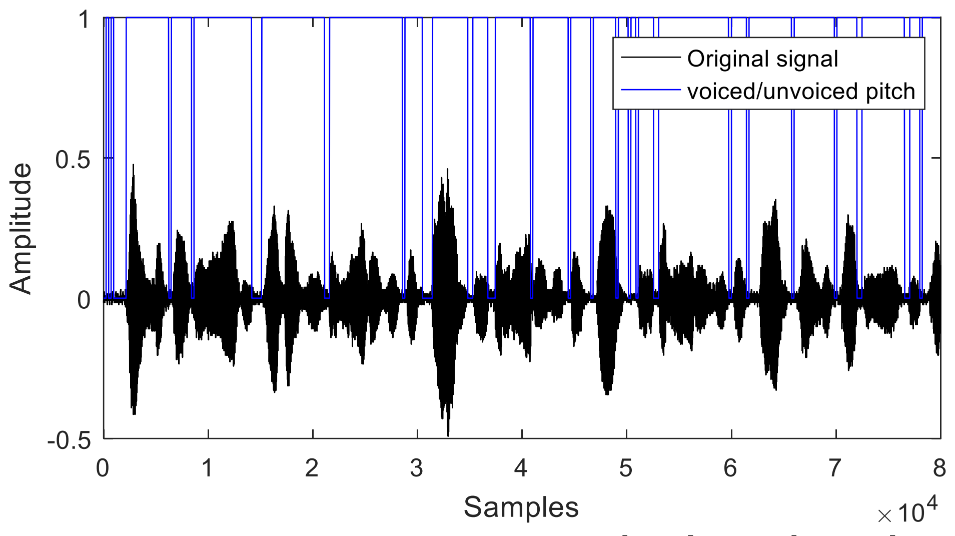


Figure 2.12: Voicing estimation based on pitch period

2.3.9 Voicing detector

Speech frames can either be voiced or unvoiced. This process is completed by a voicing detector. Classifying a speech frame as being voiced or unvoiced is very important in implementing linear predictive coding algorithm. This is because, if the signal is misclassified, the quality of synthetic speech can be negatively affected. The voicing detector depends on some measurements to accomplish this task. These measurements include pitch period, energy (magnitude sum function), zero crossing rate and prediction gain as explained in [7]. In Matlab software [9], magnitude sum function, pitch period and zero crossing rate, were implemented and used as measurements by a voicing detector to classify the frames. As shown below in Figure 2.13, if the fame is equal to one, it is taken as voiced otherwise unvoiced and this voicing is sent to the decoder.

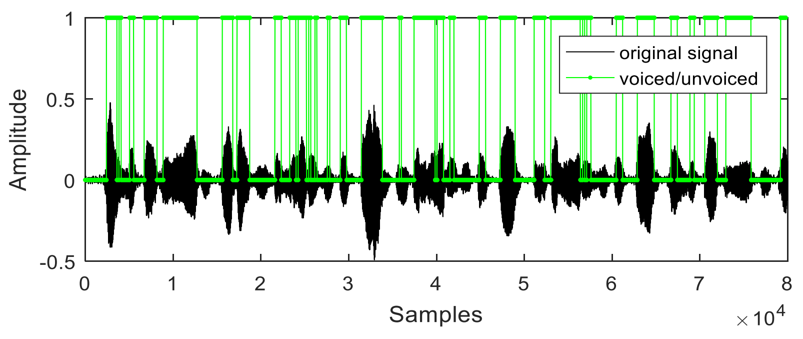


Figure 2.13: Classifying voiced and unvoiced frame

2.3.9.1 Energy

Energy is one of most simple indicator of voiced and unvoiced speech signal. This is because, voiced speech signals are having higher energy than unvoiced ones. Consider a frame of length N ending at time instant x, the energy is given by Eq. (2.28) [7].

$$E[x] = \sum_{n=x-N+1}^{x} s^2[n] \tag{2.28}$$

Precisely, magnitude sum function can be taken as shown in Eq. (2.29) and it serves the same function as Eq. (2.28) [7].

$$MSF[x] = \sum_{n=x-N+1}^{x} |s[n]| \tag{2.29}$$

The voiced speech signal relatively has a low pitch frequency values, hence its energy is higher in low frequency region [7]. Therefore, a voiced signal can be distinguished from unvoiced by lowpass filter a speech signal prior to calculated energy. Thereby taking the energy of low frequency components into consideration. In Matlab implementation, since energy of a voiced signal is concentrated in low frequency components. This was completed by called a Matlab built in function $butter$ which passed a normalised value of cut-off frequency, filter order and filter type (low) and the returned transfer function coefficients of a low pass Butterworth filter. The outputs of $butter$ command and a signal frame were used as input to $filter$ command such that a filtered signal y was obtained. Finally, magnitude sum function was obtained by calling Matlab commands sum and abs which took y as its input. From magnitude sum function obtained, a threshold value was computed in Matlab software and any frame below it was taken to be unvoiced otherwise voiced as shown Figure 2.14, where the voiced frames are of higher magnitude values.

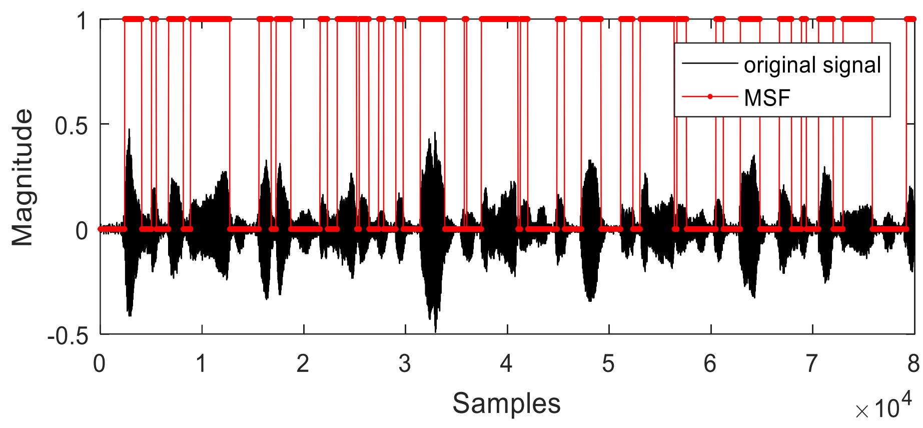


Figure 2.14: Voicing estimation based on MSF

2.3.9.2 Zero Crossing Rate

Consider a frame ending at time instant x, the zeros crossing rate is completed by Eq. (2.30) [7].

$$Z[x] = \frac{1}{2} \sum_{n=x-N+1}^{x} |\mathrm{sgn}(s[n]) - \mathrm{sgn}(s[n-1])|, \tag{2.30}$$

where sgn(.) is a sign function which returns ± 1 depending on the sign of the operand. In situations where consecutive samples are having signs changes, zeros crossing occurs. The presence of pitch frequency component (low frequency nature) in voiced speech makes it to have a lower zero crossing rate in comparison to unvoiced signal. The noise like appearance of unvoiced signal makes it to have a higher zero crossing rate since large portion of its energy is located in high frequency region [7]. In Matlab software [9], Eq. (2.30) was implemented by calling Matlab built-in function $sign$ and abs which passed signal frame and was computed over a range from one up to length of input signal frame. A threshold value is set from computed zero crossing values, and a frame which is below it is classified as voiced otherwise unvoiced.

2.3.9.3 Prediction Gain

The ratio of energy in the signal to energy in prediction error signal is referred to as prediction gain as shown in Eq. (2.31) [7].

$$PG[x] = 10 \log_{10} \left(\frac{\sum_{n=x-N+1}^{x} s^2[n]}{\sum_{n=x-N+1}^{x} e^2[n]} \right) \tag{2.31}$$

Voiced frames attain more prediction gain than the unvoiced due to high correlation between or among samples hence they are more predictable. Due to random nature of unvoiced signal, they are less predictable. However, if the frames are of very low amplitudes, prediction gain is not considered to classify the speech signal as voiced or unvoiced. This is done to avoid numerical errors or problems, thus a speech signal can be classified as unvoiced by considering the energy level [7].

Prediction gain is also a very important tool in determining the optimal value of prediction order which should be used while estimating LPCs for a signal frame. In Matlab software, LPCs were extracted from a speech frame using different values of prediction order M in the range between 1 to 100. The extracted LPCs were used to model prediction error filter which filters the signal frame to end up with estimated frame. The difference between signal frame and estimated frame gives prediction error signal which is illustrated in Figure 2.15. Finally, Eq. (2.31) was implemented in Matlab software and a plot of prediction gain against prediction order was obtained as shown in Figure 2.16 [9].

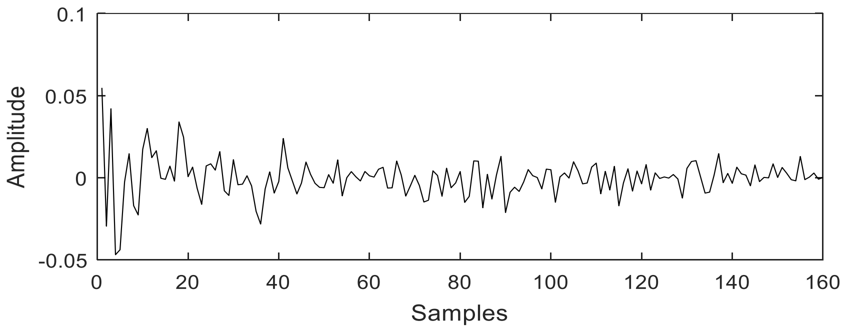


Figure 2.15: Prediction error signal

From Figure 2.16, prediction gain increases from one up to ten, and after ten there is no any further increase. Therefore, the optimal value of prediction order is ten for estimating this signal frame.

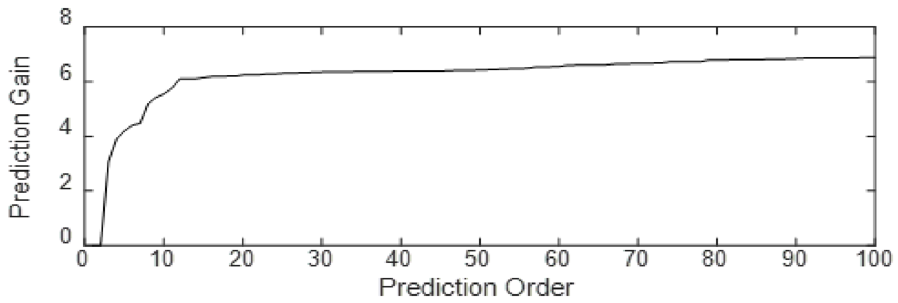


Figure 2.16: A graph of prediction gain against prediction order

2.3.10 Structure of Linear Predictive Coding Algorithm

The linear predictive coding algorithm that was used as an algorithm in speech production is shown below in Figure 2.17 and Figure 2.18 as illustrated in [7]. The role of an encoder was to extract parameters from the speech signal which were then sent to the decoder's input. The decoder used these parameters sent by encoder in speech production model to synthesize speech.

2.3.10.1 LPC Encoder

The LPC encoder is shown in Figure 2.17, where human voice or speech is used at the input. Therefore, instead of transmitting the whole speech samples, speech parameters are extracted from speech signal and then transmitted to the decoder. For linear predictive analysis, speech signals constantly change over time, hence it is necessary to be segmented into frames using a small time duration, i.e, between 20 ms to 30 ms [7]. A segmented frame can also be windowed to smoothen discontinuous changes caused by signal framing [13]. A pre-emphasis filter is used to pass high frequency components thereby adjusting the spectrum of input signal. Extracted LPCs from the speech frame are used to model prediction error filter which filters the pre-emphasized speech resulting into a prediction error signal. If the frame is voiced, prediction error signal is used to estimate pitch period. Therefore, by using a prediction error signal as the input to pitch estimation algorithm (autocorrelation method explained above), it results into a more accurate estimate of pitch period [7]. The LPC index, power index, pitch period index and voicing, are joined or packed together to form one bit stream and then sent to the decoder.

Consider a prediction error signal $e[n]$, given that it is in the range of 0 and $N-1$, where N is the length of the frame. The power of unvoiced frame is given by Eq. (2.32) [7].

$$P = \frac{1}{N}\sum_{n=0}^{N-1} e^2[n]. \tag{2.32}$$

However, the power of voiced frame is obtained from integer number of pitch periods as shown in Eq. (2.33) [7].

$$P = \frac{1}{\lfloor N/T \rfloor T} \sum_{n=0}^{\lfloor N/T \rfloor T - 1} e^2[n], \tag{2.33}$$

where, $\lfloor . \rfloor$ is a floor function and an assumption is made that $N > T$ given that T is pitch period.

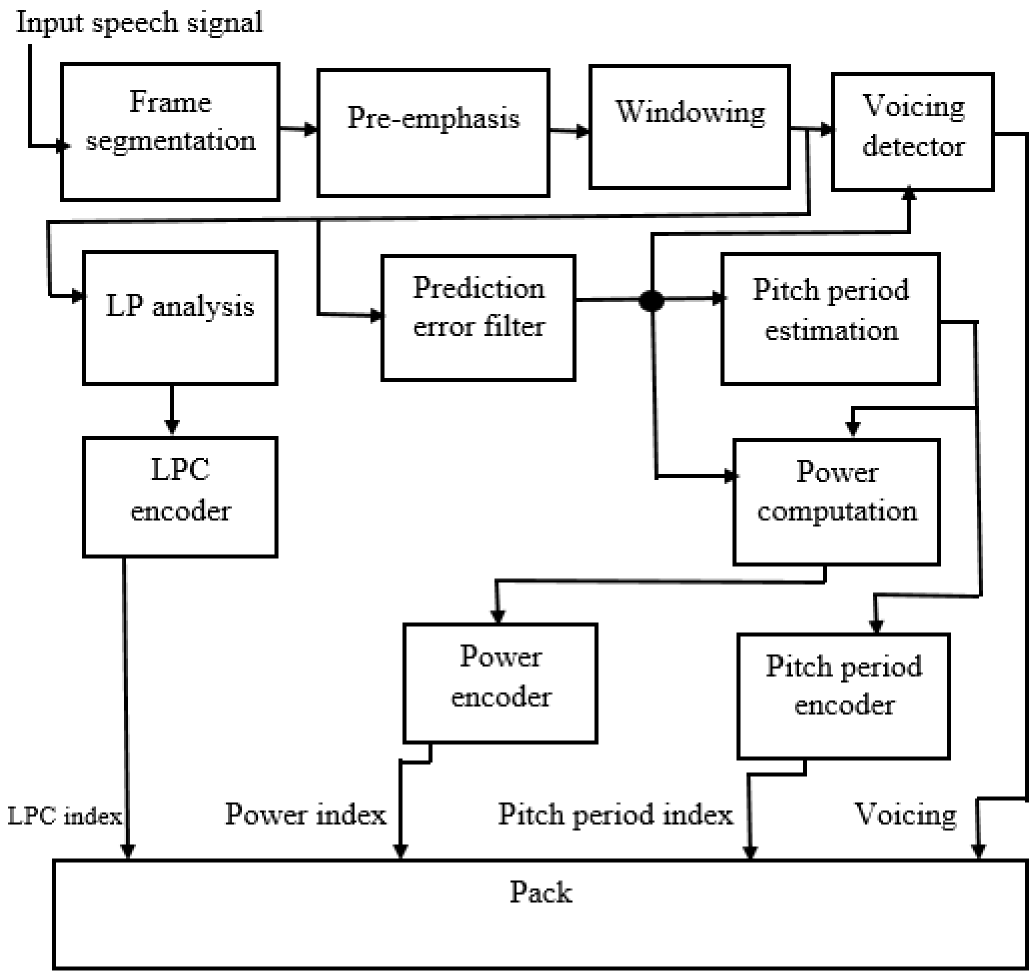


Figure 2.17: LPC encoder block diagram [7][13]

In Matlab software, linear predictive coefficients, pitch period, voicing and gain parameters were extracted from an audio signal recorded and transmitted to the decoder.

2.3.10.2 LPC Decoder

The bit-stream sent from the LPC encoder is used as input to LPC decoder. The LPC decoder is basically a speech production model which uses parameters sent by LPC encoder to produce a synthetic speech. At the decoder, the gain G of unvoiced and voiced frame is computed. The gain of unvoiced and voiced speech is shown in Eq. (2.34) and Eq. (2.35) respectively [7].

$$G = \sqrt{P}\,, \tag{2.34}$$

where P is the computed as shown in Eq. (2.32).

$$G = \sqrt{TP}, \tag{2.35}$$

given that P is the power computed in Eq. (2.33).

However to estimate gain parameters as shown in Eq. (2.34) and Eq. (2.35) for unvoiced and voiced signal respectively, prediction error signal should be determined. In Matlab software [9], the extracted LPCs were used to model prediction error filter which filters the signal frame to end up with estimated frame. The difference between signal frame and estimated frame gave prediction error signal which was used to compute power for the unvoiced and voiced frame as shown in Eq. (2.32) and Eq. (2.33) respectively. However, for voiced frames as shown in Eq. (2.33) and Eq. (2.35), the value T should be known. Therefore, T was considered as pitch period values which had been already computed in Matlab. Finally, the output of synthesis filter (from the speech production model shown in Figure 2.19) was de-emphasis filtered to counteract the effect of pre-emphasis filter hence a synthetic speech was produced. A diagram of LPC decoder is shown in Figure 2.18.

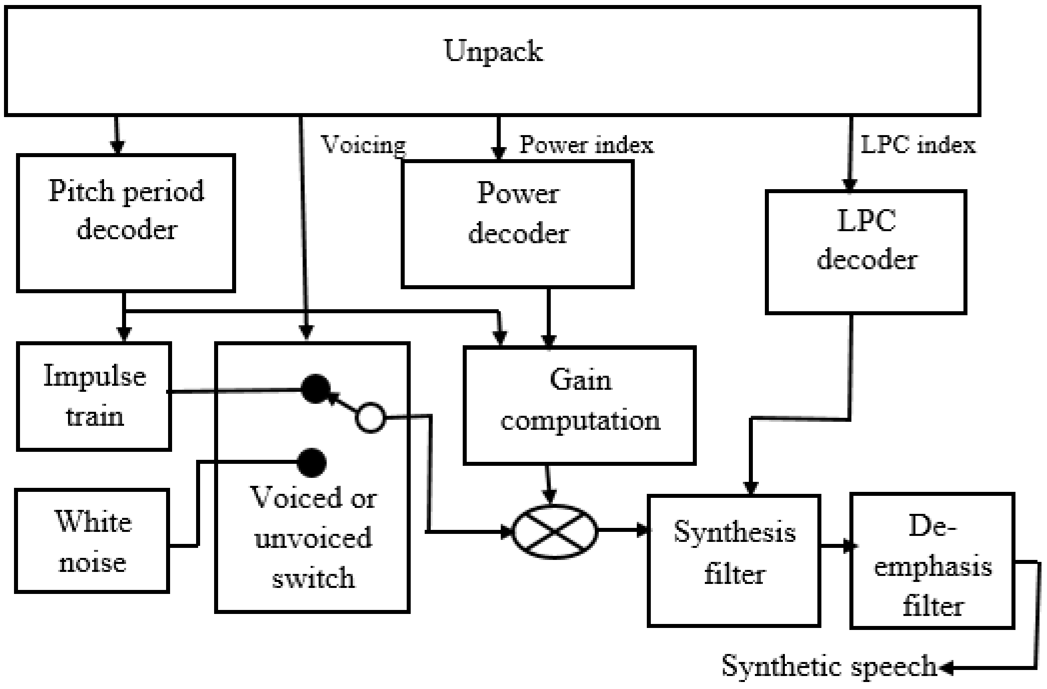


Figure 2.18: LPC decoder block diagram [7]

At the decoder, the excitation signals are generated. For voiced, the excitation is a pulse train which is given by expression shown below as illustrated in [7].

$$\sum_{i=-\infty}^{\infty} \delta[n - iT],$$

with

$$\delta[n] = \begin{Bmatrix} 1, & \text{if } n = 0, \\ 0, & \text{otherwise} \end{Bmatrix},$$

given that T is the pitch period. For unvoiced, the excitation is implemented in Matlab software as random noise implemented by calling a Matlab built-in function $randn$.

At the decoder, LPC parameters in terms filter coefficients, gain, voicing and pitch period parameters were received which were then used to produce a synthetic speech. The output of synthesis was de-emphasis filtered to end up with a synthetic speech [13]. Since de-emphasis filter is inverse of a pre-emphasis filter, the same value of filter coefficient was used. Since linear predictive coding algorithm is a lossy compression technique, the quality in the synthetic speech signal reduced. This can be confirmed by either listening to both the original signal and synthetic speech signal or by looking at their time plots as shown in Figure 2.19.

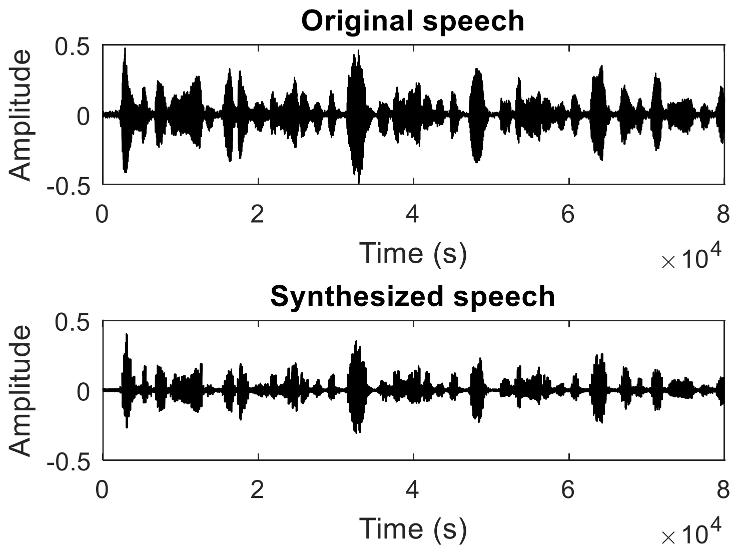


Figure 2.19: Time plot of original and compressed speech signal

CHAPTER 3

END TO END COMMUNICATION

This chapter describes several modules and algorithms in details which are used in end to end communication. These modules or algorithms include channel and channel coding, modulation and demodulation, pulse shaping and receive filtering, carrier recovery, equalization and finally correlation.

Engineering challenges or problems are taken to be optimization problems by most of engineers or researchers as explained in [19]. In order to overcome such challenges, the following aims should be met;

- A goal which consists of selecting a performance or an objective function is set. This can either be to maximize or minimize the set performance function.
- The method to achieve the set goal is selected.
- The chosen method is tested on the objective function to investigate if the intended purpose is obtained.

Setting a goal usually consists of finding a function that can be minimized or maximized. This involves locating a minimum or maximum value, which provides useful information about the problem at hand.

There are many methods or algorithms which can be used to determine a minimum or maximum point of a given function. For instance, if a problem is to find a point at which a polynomial function achieves its minimum value, this can be solved directly by finding the derivative and equate it to zero [15]. However, it is very difficult to get such direct solution when considering a noisy signal. The recursive adaptive algorithm considered in this thesis is steepest descent algorithm and forms a basis of very many adaptive algorithms in communication systems [19].

3.1 Steepest Descent Algorithm

Steepest descent algorithm starts with initializing (guessing) the location of a minimum. Then, assesses which direction from an initial guess is the steepest as going down the hill, and then a new guess is made along the same direction. In the same sense, ascending a hill also starts with an initial estimate or guess of a maximum location, assesses which direction climbs very fast and then a new guess is made along that direction. In most cases, a new guess is better than the old one and so on. The process repeats, ideally getting closer to an optimal location at each step. The most important point to note here is, a direction of a current location is assessed by a gradient. The uphill direction is determined by a positive gradient and a negative gradient is for downhill direction [15][20].

Consider a polynomial shown in Eq. (3.1),

$$f(y) = y^2 + by + c, \tag{3.1}$$

steepest descent algorithm can be implemented to minimize it. Let y be the initial estimate at time k, which can then be represented by $y[k]$. A new estimate of y at time $k+1$ can be obtained by steepest descent algorithm shown in Eq. (3.2) [15][21].

$$y[k+1] = y[k] - \mu \frac{\partial f(y)}{\partial y}, \tag{3.2}$$

where μ is a step size, $f(y)$ is the objective function which is to be optimised at a current estimate $y[k]$.

As long as the step size is reasonably small, the new estimate $y[k+1]$ is very close (closer) to the minimum as compared to the old estimate $y[k]$ [21].

For simplicity, polynomial Eq. (3.1) is modified into;

$$f(y) = y^2 + 10y + 25. \tag{3.3}$$

Steepest descent algorithm is used to minimize the polynomial $f(y)$, such that,

$$y[k+1] = y[k] - \mu(2y[k] + 10),$$

$$y[k+1] = (1 - 2\mu)y[k] - 10\mu. \tag{3.4}$$

By choosing a suitable value of step size μ, an estimate $y[k+1]$ shown in Eq. (3.4) is determined which minimizes the objective function illustrated in Eq. (3.3). The suitable value of step size can be chosen by using a number of iterative methods which include, Golden section search method, Fibonacci method, bisection method and many others as explained in [15].

In order to effectively evaluate how steepest descent algorithm behaves, a graph of objective function against a parameter being optimized is plotted. The graph plotted is called error surface as shown in Figure 3.1 and Figure 3.2.
Figure 3.1 clearly shows a minimum of Eq. (3.3) which occurs at negative five. This can be understood by even some one slopping downhill to the bottom of the valley and finally reaches the minimum point irrespective of his or her starting point. Objective function showed in Eq. (3.3) has got only one minimum. However, there are those with more than one minimum like that one shown in Eq. (3.5) [19].

$$f(y) = e^{-0.2|y|}\sin(y) - 0.2\sin(y)\,sign(y) \tag{3.5}$$

As shown in Figure 3.2 where objective function showed in Eq. (3.5) has got many minima, and this happens in practice where a valley or a hill may have more than one minimum.

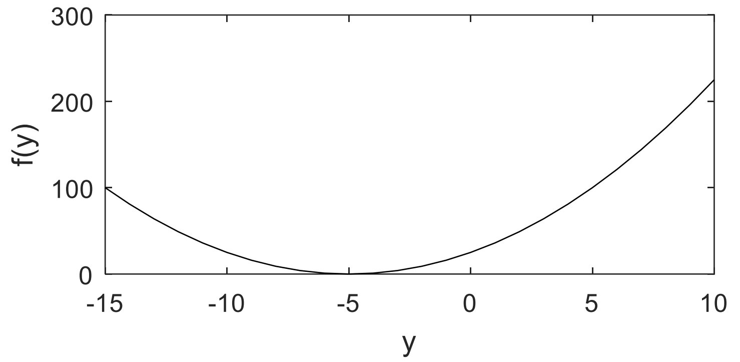


Figure 3.1: Error surface for objective function shown in Eq. (3.3)

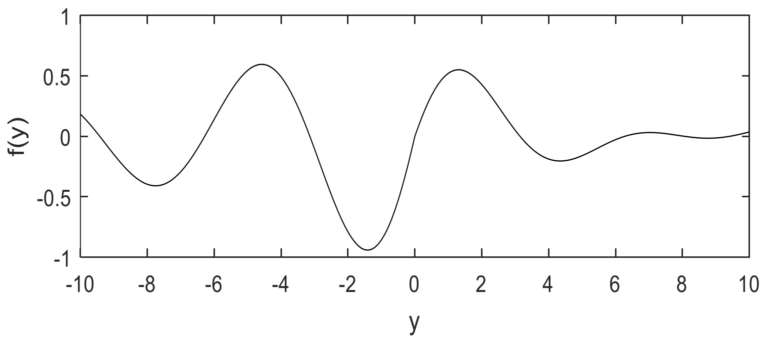


Figure 3.2: Error surface for objective function shown in Eq. (3.5)

However, the true descent algorithm once falls in a minimum even though it's not the lowest one, cannot climb to the peak and then falls to true lowest minimum. Therefore, it is necessary to initialize steepest descent algorithm at the point close to the true minimum [15].

3.2 End to End Communication

The signal propagates from the transmitter to the receiver through the communication channel. From the source encoder (explained in chapter 2), the compressed signal is sent to channel encoder to add redundancy bits, the information bits or symbols are converted into analog signal by a pulse shaping filter. This signal is up converted, propagates through the channel up to the receiver. As the signal propagates through the channel, it undergoes a lot of impairments or interferences before reaching the receiver. To perfectly decode the transmitted signal, the receiver has got blocks or algorithms to complete this task. The main components of transmitter-receiver blocks (algorithms) are explained as follows.

3.3 Channel Coding

The main aim of channel encoding is to reduce the number of errors which may corrupt the signal as it propagates from the transmitter to the receiver. By channel coding, the redundancy is added to reduce the probability of error in the transmission. In communication system, if the signal at the input of the receiver is not the same as that one at the transmitter's output, it means that the errors occurred while the signal was propagating through the channel. At channel encoder, structured redundant bits are added to the transmitted bits in order to;

- detect errors caused by the noisy channel.
- correct errors and reconstruct the original transmitted bits.

Error detection and correction techniques can either be systematic or non-systematic. In systematic coding, the message bits appear unaltered in the code word with the check bits (parity bits). For non-systematic coding scheme, the message bits are mixed with the check bits.

At the transmitter, channel encoder adds some redundancy to the input signal before transmission over a communication channel. At the receiver, channel decoder recovers the transmitted signal and also corrects some errors which might have occurred as the signal propagates over the channel [22]. There are a number of error correcting schemes or algorithms which are being implemented to reduce the percentage of error as the signal propagates over a communication channel, without experiencing a heavy penalty.

3.3.1 Linear Block Codes Algorithm

The (n, k) notation classifies the linear block coding algorithm, where k is the length of information bits and n is the length of code word. For n $> k$, linear block codes operate on k bits to be transmitted and the length of code word n. Generator matrix (G) of dimension $k \times n$ maps information bits (x) of length k to their corresponding code words (C) of length n as shown in Eq. (3.6) [22][23].

$$C = xG \tag{3.6}$$

Consider a generator matrix of the structure,

$$G = [I_k|P], \tag{3.7}$$

where I_k is $k \times k$ identity matrix and P is a $k \times (n-k)$ parity check matrix. If the generator matrix is of the structure shown in Eq. (3.7), then linear block code is considered as systematic code where the first k bits in the code word are called information bits and the $n - k$ are known as parity check bits. And also parity check matrix H can be taken as [22],

$$H = [P^T|I_{n-k}] \tag{3.8}$$

Linear block code algorithm can be implemented following the procedures below [19].

- The message bits to be transmitted are arranged into code vector $x = [x_1, x_2, \dots \dots x_k]$
- Then the code word c of length n is transmitted, i.e, Eq. (3.6).
- The received vector y is multiplied with the parity check matrix, i.e, yH^T
- If $yH^T = 0$, there is no error in the received vector otherwise errors have occurred.
- If $yH^T \neq 0$, the mapping table is generated consisting of syndrome (eH^T) and error column e. To obtain the code word, error vector e corresponding to eH^T are added together to generate a transmitted code word.
- Reconstruct the transmitted information bits x from the code words received.

3.3.2 Hamming Codes Algorithm

Hamming codes are classified as linear block codes of forward error correcting codes which were discovered by Richard Hamming in 1940's. Hamming codes are widely used in computing, telecommunications and other applications as error detection and correcting codes [24]. Table 3.1 shows code words for all the 16 possible messages of the (7,4) hamming coding scheme.

Table 3.1: (7, 4) Hamming coding system

Number	Message	Code words
1	0000	0000000
2	0001	0001011
3	0010	0010110
4	0011	0011101
5	0100	0100101
6	0101	0101110
7	0110	0110011
8	0111	0111000
9	1000	1000111
10	1001	1001100
11	1010	1010001
12	1011	1011010
13	1100	1100010
14	1101	1101001
15	1110	1110100
16	1111	1111111

Hamming distance can be taken as the difference between the number of bit positions in which the code words. For example from the Table 3.1, Hamming distance between code word one and two is 3, and the Hamming distance between code word two and three is 4. In conclusion, the Hamming distance for this coding scheme, is between the range of 3 and 7. Minimum Hamming distance (d_{min}) is the smallest Hamming distance in the set of code words and can be used to determine how many errors a Hamming coding scheme can detect and correct.

The maximum number of errors that can be detected is $d_{min} - 1,$ and the maximum number of error that can be corrected is $\left\{\frac{d_{min}-1}{2}\right\}$ as explained in [25]. In this (7,4) Hamming coding scheme, 4 data bits are transmitted and the length of each code word is 7. The minimum hamming distance is 3, hence the maximum number of errors to be detected are 2 and only one error can be corrected.

For simplicity, consider a $(7,4)$ Hamming code with the generator matrix G [22],

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix},$$

and the parity check matrix H obtained from Eq. (3.8) is given as,

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

$$H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

For decoding, let's multiply a code word say [1 0 1 1 0 1 0] with the parity check matrix H^T, the syndrome [0 0 0] all zero vector is obtained, which shows that the received code word had no error. To determine how this coding scheme can detect and correct errors, let's change the fourth bit from 1 to 0 in the code word [1 0 1 1 0 1 0], and assume it was instead received at the decoder. Multiplying the code word [1 0 1 0 0 1 0] with H^T, the syndrome vector [0 1 1] is obtained, which indicates that the received code word has got error(s). If the syndrome is not all zeros vector, compare the syndrome with the rows of parity check matrix H^T and detect the row number of H^T that matches with the syndrome. The row that matches with the syndrome, denotes the bit position that has an error. Therefore, the syndrome [0 1 1] obtained matches with row 4 in the parity check matrix H^T. This indicates that, the fourth bit in the received code word [1 0 1 0 0 1 0] has got an error. The error in the received code word can be corrected by looking at the look up table shown in Table 3.2. To correct the error, take the error vector $e = [0\ 0\ \ 0\ 1\ 0\ 0\ 0]$ corresponding to the calculated syndrome vector $s = [0\ 1\ 1]$ and then add it to the received code word [1 0 1 0 0 1 0] while considering modulo 2 addition. By doing so, the error in fourth bit position is corrected from 0 to 1.

Table 3.2: Look up table for (7, 4) Hamming code

Syndrome $\boldsymbol{s}$	Error vector $\boldsymbol{e}$
1 1 1	1 0 0 0 0 0 0
1 0 1	0 1 0 0 0 0 0
1 1 0	0 0 1 0 0 0 0
0 1 1	0 0 0 1 0 0 0
1 0 0	0 0 0 0 1 0 0
0 1 0	0 0 0 0 0 1 0
0 0 1	0 0 0 0 0 0 1

3.4 Channel

A signal is transmitted from the transmitter, propagates through a communication channel up to the receiver. For a wireless communication channel, there is no guiding medium between the transmitter and the receiver. Therefore, a signal can propagate in straight line from a transmitter to a receiver, or it can take different paths and then reflected by multiple reflector components arising from objects such as trees and buildings up to the receiver. Hence at the receiver, there are multiple signal components. The straight line components are called line of sight and those arising from scattering actions are called non line of sight components. Since different signal components are received from different paths, hence this is called a multipath propagation channel [22]. Precisely, the signal reaches the receiver as a delayed copy of a transmitted signal. The delay depends on the distance between the transmitting and the receiving antenna, while the strength of reflection is determined by the reflecting (scattering) objects [26][22]. Let us assume that there are l delays represented by $\tau_1, \tau_2, \ldots., \tau_l$, transmitted signal $s(t)$ and the strength reflecting objects as $\alpha_1, \alpha_2, \dots \dots, \alpha_l,$ then the received signal $r(t)$ is given by Eq. (3.9) as illustrated in [19].

$$r(t) = a_1 s(t - \tau_1) + a_2 s(t - \tau_2) + - - - + a_l s(t - \tau_l) + \eta(t), \qquad (3.9)$$

where $\eta(t)$ are additive interferences. The transmission model in Eq. (3.10) has the form of linear filter, and the total length of time over which the impulse response is non zero is called the delay spread of physical medium. Assuming a fixed sampling period $T,$ a transmission channel in Eq. (3.9) can be digitally modelled as Eq. (3.10) [19].

$$r(kT) = a_0 s(kT) + a_1 s\big((k-1)T\big) + - - + \alpha_l s\big((k-l)T\big) + \eta(kT). \qquad (3.10)$$

In order to make the model shown in Eq. (3.9) to be relatively close to model shown in Eq. (3.10), the maximum delay τ_l must be at least less than the total time over which the impulse response is non zero (the time lT) [19].

3.5 Pulse Shaping and Receive Filtering

Pulse shaping does the conversion of digital signals (symbols) to analog signal for transmission over a transmission channel such as space. Therefore, pulse shaping changes each and every digital symbol to its corresponding analog pulse which are retrieved back to digital symbols by receive filter at the receiver. The digital symbols can be taken from finite set of values such as binary values (± 1), or they may be taken from higher four level alphabet $(\pm 1, \pm 3)$ [19]. Pulse shaping and receive filtering is illustrated in Figure 3.3 as explained in [19].

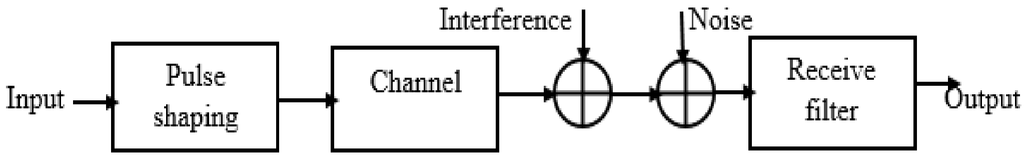


Figure 3.3: Pulse shaping and Receive filtering block [19]

Every symbol is represented by an analog pulse which is a scaled copy of the transmitted signal, the analog pulses propagates over a transmission channel up to the receiver, and if all goes well, the output message is the same as input message [22]. Therefore, these pulses should be chosen and be able to minimize intersymbol interference and maximize signal to noise ratio. These pulses are called Nyquist pulses [22], and dictate the spectrum of the whole signal transmission. A pulse shape which satisfies a Nyquist intersymbol interference (ISI) criteria is called a Nyquist pulse [22]. The Nyquist ISI criteria is given by Eq. (3.11) as explained in [22].

$$h(t = kT) = h_k = \begin{Bmatrix} 1 & k = 0 \\ 0 & k \neq 0 \end{Bmatrix}, \tag{3.11}$$

where $h(t)$ is the pulse.

Raised cosine pulse or raised filter is one of the most used pulses in communication system because [19];

- it has suitable zero crossing at desirable instants.
- its band edges are easy to approximate and they are not severe like with sinc pulse
- it has also characteristic of decaying very fast in time domain and being narrow in frequency domain.

The raised cosine pulse filter frequency response is given by Eq. (3.12) [22].

$$h_{rc}(f) = \begin{Bmatrix} T & 0 \leq |f| \leq \frac{1-\beta}{2T} \\ \frac{T}{2}\left\{1 + \cos\left[\frac{\pi T}{\beta}\left(|f| - \frac{1-\beta}{2T}\right)\right]\right\} & \frac{1-\beta}{2T} \leq |f| \leq \frac{1+\beta}{2T} \\ 0 & |f| > \frac{1+\beta}{2T} \end{Bmatrix}, \tag{3.12}$$

where β is known as a roll off factor in the range $0 \leq \beta \leq 1$ and T is sampling period.

In end to end communication, it should be noted that it is not only the pulse shape which should be Nyquist but the convolution of both the pulse shape and the receive filter. When an ideal channel is considered, the transmitted signal should not have ISI after receive filtering. In this case, the pulse shape and the receive filter are matched together in order to;

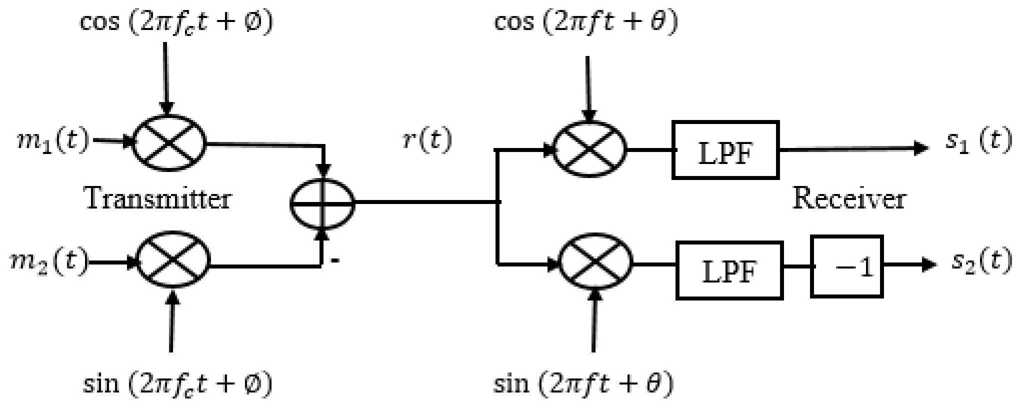
- maximize the signal to noise ratio.
- have zero ISI at the receiver.

Even though raised cosine pulse is a preferred pulse, the convolution with itself does not result into a Nyquist pulse, but it is liked because of its ability to conserve the bandwidth and also its impulse response decays quickly. Taking the square root of raised cosine pulse in frequency domain results into square-root raised cosine filter (SRRC), and its multiplication with itself gives a Nyquist pulse which is highly desired pulse in end to end communication. The time domain $x(t)$ representation of SRRC is shown in Eq. (3.13) as illustrated in [19][22].

$$x(t) = \begin{cases} \dfrac{\left(1-\beta+\dfrac{4\beta}{\pi}\right)}{\sqrt{T}} & t = 0 \\ \dfrac{\beta(\pi+2)}{\sqrt{2T}\pi} sin\left(\dfrac{\pi}{4\beta}\right) + \dfrac{\beta(\pi-2)}{\sqrt{2T}\pi} cos\left(\dfrac{\pi}{4\beta}\right) & t = \pm\dfrac{T}{4\beta} \\ \dfrac{\sin\left(\dfrac{\pi(1-\beta)t}{T}\right) + \left(\dfrac{4\beta t}{T}\right) cos\left(\dfrac{\pi(1+\beta)t}{T}\right)}{\sqrt{T}\left(\dfrac{\pi t}{T}\right)\left(1-\left(\dfrac{4\beta t}{T}\right)^2\right)} & \text{otherwise} \end{cases} \tag{3.13}$$

3.6 Quadrature Amplitude Modulation (QAM)

QAM is quadrature modulation scheme which carries more information than non-quadrature modulation scheme like amplitude modulation (AM), thus enabling efficient utilisation of bandwidth. This saves bandwidth by reducing redundancy in the transmitted message. A block diagram for QAM modulation and demodulation is shown in Figure 3.4.



$\cos(2\pi f_c t + \emptyset)$
$\cos(2\pi f t + \theta)$
$m_1(t)$
$r(t)$
LPF
$s_1(t)$
Transmitter
Receiver
$m_2(t)$
-
LPF
-1
$s_2(t)$
$\sin(2\pi f_c t + \emptyset)$
$\sin(2\pi f t + \theta)$

Figure 3.4: QAM Transmitter-Receiver block [19].

In quadrature modulation, two signals can be sent using cosine and sine orthogonal carriers on the same bandwidth while using the same carrier frequency [22][19]. Consider two message signal $m_1(t)$ and $m_2(t)$ as a function of time t,

$$r(t) = m_1(t)\cos(w_c t + \emptyset) - m_2(t)\sin(w_c t + \emptyset), \tag{3.14}$$

where $\emptyset$ is carrier phase offset and f_c is carrier frequency given that $w_c = 2\pi f_c$.

Eq. (3.14), is real valued but considering a complex valued message being mixed with complex sinusoid $e^{j(2\pi f_c t + \emptyset)}$. Then,

$$m(t) = m_1(t) + jm_2(t). \tag{3.15}$$

From Euler's formula shown below,

$$e^{\pm jx} = \cos(x) \pm j\sin(x),$$

the complex valued message can be obtained as shown Eq. (3.17).

$$m(t)e^{j(w_c t+\emptyset)} = A + jB, \tag{3.16}$$

given that,

$$A = m_1(t)\cos(w_c t + \emptyset) - m_2(t)\sin(w_c t + \emptyset)$$

$$B = m_2(t)\cos(w_c t + \emptyset) + m_1(t)\sin(w_c t + \emptyset)$$

Equation 3.18 takes real part into account as shown in below,

$$r(t) = \mathrm{Re}\left(m(t)e^{j(w_c t+\emptyset)}\right) \tag{3.17}$$

Messages $m_i(t)$ can be represented by,

$$m_i(t) = \sum_k s_i[k]p(t - kT), \tag{3.18}$$

where $s_i[k]$ is the i^{th} message symbols taken from finite alphabet at time k, $p(t)$ is pulse shaping filter and T is adjacent symbol interval. The modulated signal $r(t)$ is shown in Eq. (3.20).

$$r(t) = \sum_k p(t - kT)\,[s_1[k]\cos(2\pi f_c t + \emptyset) - s_2[k]\sin(2\pi f_c t + \emptyset)], \tag{3.19}$$

$$= \mathrm{Re}\left[\sum_k p(t - kT)s[k]e^{j(2\pi f_c t+\emptyset)}\right] \tag{3.20}$$

where, $s[k] = s_1[k] + js_2[k]$.

3.6.1 Demodulation of QAM Signal

Consider a complex valued carrier expression $e^{-jwt+\theta}$ mixed with a modulated signal shown in Eq. (3.20), where θ is the phase at the receiver and f is frequency at the receiver, given that $w = 2\pi f$.

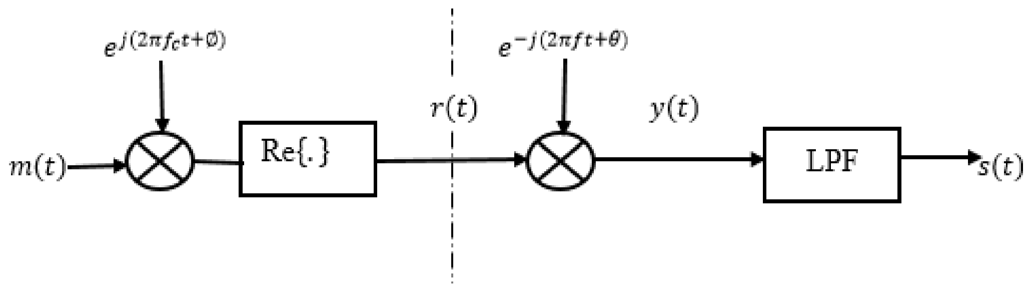


Figure 3.5: Signal demodulation block [19]

From the Figure 3.5, QAM demodulation can be done mathematically [19];

$$y(t) = e^{-j(2\pi f t+\theta)} r(t) \tag{3.21}$$

$$y(t) = e^{-j(2\pi f t+\theta)} \mathrm{Re}\left\{m(t)e^{(j2\pi f_c t+\emptyset)}\right\}$$

$$y(t) = e^{-j(2\pi f t+\theta)}(m_1(t)\cos(2\pi f_c t+\emptyset) - m_2(t)\sin(2\pi f_c t+\emptyset))$$

$$y(t) = \frac{e^{j(\emptyset-\theta)}}{2}\left(m_1(t)e^{-j2\pi(f+f_c)t} + m_1(t)e^{-j2\pi(f-f_c)t}\right)$$
$$+j\frac{e^{j(\emptyset-\theta)}}{2}\left(m_2(t)e^{-j2\pi(f+f_c)t} + m_2(t)e^{-j2\pi(f-f_c)t}\right) \tag{3.22}$$

The signal $y(t)$ is passed through a low pass filter (LPF) to do away with high frequency components. Let us assume $f \approx f_c$ and also that lowpass filter be of cut-off frequency f_l, that is to say $f_c < f_l < 2f_c$. This indicates that all frequency components in signal $y(t)$ lying between f_c and $2f_c$ pass through a low pass filter otherwise eliminated. Then the demodulated signal $s(t)$ is given by,

$$s(t) = \mathrm{LPF}\{y(t)\},$$
$$s(t) = \tfrac{1}{2}m(t)e^{-j2\pi(f-f_c)t+j(\theta-\emptyset)}, \tag{3.23}$$

where $m(t) = m_1(t) + jm_2(t)$.

If the frequency at receiver is the same as the frequency at the transmitter, i.e, $f = f_c$ and also if the phase at the receiver is the same that one at the transmitter, i.e, $\emptyset = \theta$, the demodulated signal $s(t)$ will be attenuated copy of transmitted signal $m(t)$ by a factor of $\frac{1}{2}$ as shown in Eq. (3.24)

$$s(t) = \frac{1}{2}m(t) \tag{3.24}$$

Therefore, for successful demodulation of QAM signal, both frequencies and phases at transmitter and receiver should be aligned.

When modulating frequency is not aligned with or identical to demodulating frequency, i.e, $f \neq f_c$, the constellation points twist at a rate proportional to the difference between receiver and transmitter frequency $(f - f_c)$. Also, when phase shift at receiver is not the same as phase shift at transmitter, the constellation points become rotated hence causing inter symbol interference. Therefore, it is equally important to reverse the rotation of constellation points in order to recover the transmitted symbol sequences [19].

In practical scenarios as signal propagates from a transmitter to receiver, it undergoes a lot of impairments which may include multipath interference, broad and narrow band noise, intersymbol interferences and many others. These interferences cause both frequencies and phases at the receiver and transmitter to diverge from each other. This makes it difficult to demodulate a received signal if frequency and phase at the transmitter is not known at the receiver. However, there exists a number of algorithms which can be used to estimate this unknown parameters. These algorithms include, Costas loop, phase locked loop and decision directed methods. In this study, Costas loop algorithm is considered.

3.6.2 Carrier Recovery for QAM Signal

Estimating the phase and frequency used at the transmitter is important while reconstructing QAM modulated signal at the receiver, hence the name carrier recovery. As shown in Eq. (3.23), for proper demodulation of the signal, phase and frequency at transmitter should be the same as those at receiver in order to recover the transmitted signal. Therefore, there should be a way in which phase θ and frequency f at the receiver are varied to the required values to align with those at the transmitter.

3.6.2.1 Costas Loop Algorithm

In this study, analog front end was considered, where the incoming analog signal is first down converted such that sampling takes place at a lower sampling rate [19]. In order to accurately demodulate a received analog signal as shown in Eq. (3.23), both the phase and the frequency at the transmitter and receiver should be synchronised [19]. From Eq. (3.21), the received signal $r(t)$ is mixed with a complex sinusoid. In ideal setting, the transmitter and receiver are assumed to be synchronised. This means that both the phase and frequency offsets at the transmitter and receiver are the same. However, if the signal propagates over a noisy channel, the phase and frequency at the transmitter and at the receiver differ from each other. Therefore, the phase and frequency offsets can be determined by adaptive Costas loop algorithm which continuously estimates and also eliminates the unknown phase and frequency offsets as explained in [19].

The 4-QAM Costas loop algorithm is derived from that of pulse phase modulation [19]. Let us assume a pulse modulated signal $r(t)$ received at the receiver, which is first demodulated, lowpass filtered and then squared or averaged to come up with Costas loop algorithm invented by J.P Costas shown in Eq. (3.25) as explained in [19][28].

$$J_p(\theta) = \text{avg}\left\{\left(\text{LPF}(r(t)\cos(2\pi f t + \theta))\right)^2\right\}. \tag{3.25}$$

Let us assume that $f = f_c$ and also $r(t) = s(t)\cos(2\pi f_c t + \emptyset)$ then,

$$J_p(\theta) = \text{avg}\left\{\left(\text{LPF}(s(t)\cos(2\pi f t + \emptyset)\cos(2\pi f t + \theta))\right)^2\right\} \tag{3.26}$$

From trigonometry identity $\frac{1}{2}(\cos A + \cos B) = \cos\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right)$ then,

$$J_p(\theta) = \text{avg}\left\{\text{LPF}\left(s(t)\frac{1}{2}(\cos(4\pi f t + \emptyset + \theta) + \cos(\emptyset - \theta))\right)^2\right\}, \tag{3.27}$$

where high frequency components are suppressed by a low pass filter to end with Eq. (3.28) as explained in [19].

$$J_p(\theta) = \frac{1}{4}\text{avg}\{s^2(t)\cos^2(\emptyset - \theta)\} \approx \frac{1}{4}s_{\text{avg}}^2\cos^2(\emptyset - \theta), \tag{3.28}$$

where s_{avg}^2 is the data values which are fixed, therefore, $J_p(\theta)$ is directly proportional to $\cos^2(\emptyset - \theta)$ which forms a basis for the objective function of pulse phase modulation, and $\cos^2(\emptyset - \theta)$ has a maxima at $\theta = \emptyset + \pi x$ at integers of x. For pulse phase modulation, the algorithm is implemented such that the phase θ converges to an offset value of an integer multiple of 180^0. This observation is analysed to determine a performance function for 4-QAM which makes a carrier recovery phase to converge to 90^0 [19].

Firstly, let’s consider an objective function J_p as shown in Eq. (3.29) [19].

$$J_p = \cos^2(2\emptyset - 2\theta). \tag{3.29}$$

From a trigonometry identity shown below,

$$\cos 4\mathrm{A} = \cos^2 2A - \sin^2 2A,$$

$$\cos^2 2A = \frac{1}{2}(1 + \cos 4A),$$

the objective function J_p can be written as

$$J_p = \frac{1}{2}(1 + \cos(4\emptyset - 4\theta)). \quad (3.30)$$

From cosine curve shown in Figure 3.6, expression $\cos(4\emptyset - 4\theta)$ attains its maximum values whenever it is equalling to one, i.e,

$$\cos(4\emptyset - 4\theta) = 1$$

$$4(\emptyset - \theta) = 0, 2\pi, 4\pi, \ldots\ldots\ldots\ldots..$$

$$(\emptyset - \theta) = 0, \frac{\pi}{2}, \pi, \ldots\ldots\ldots\ldots\ldots..$$

In simplicity, expression $\cos(4\emptyset - 4\theta)$ has its maximum values at integer multiples of 90^0. Precisely, $\emptyset = \theta + \left(\frac{\pi}{2}\right)x$ at integer of x.

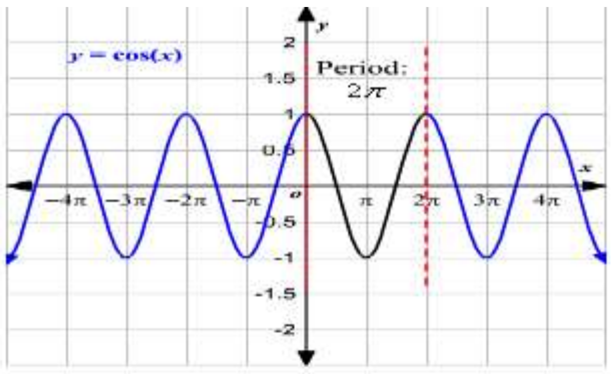


Figure 3.6: Cosine curve

Using steepest descent algorithm shown in Eq. (3.31),

$$\theta[k+1] = \theta[k] + \mu \frac{\partial J_p}{\partial \theta}, \quad (3.31)$$

where $\frac{\partial J_p}{\partial \theta}$ can be defined further as shown in Eq. (3.32). A change of sign in Eq. (3.2) from negative to positive is considered since the goal is to maximise Costas loop objective function J_p [19].

$$\frac{\partial J_p}{\partial \theta} = \frac{\partial\left(\frac{1}{2}(1 + \cos(4\emptyset - 4\theta)\right)}{\partial \theta} = 2\sin(4\emptyset - 4\theta). \quad (3.32)$$

The next task is to determine an expression which is proportional to $\sin(4\emptyset - 4\theta)$. Let's consider a received signal $r(t)$ at the receiver, the signal is first demodulated, lowpass filtered to come up $\mathrm{LPF}\{r(t)\cos(2\pi ft + \theta)\}$ and also $\mathrm{LPF}\{r(t)\sin(2\pi ft + \theta)\}$, their product gives rise to analogous derivative for PAM Costas loop algorithm [19], where LPF is low pass filtering operation. The demodulated signal components are at the same frequency f but 180^0 out of phase. In the same sense, QAM can be solved by low passing the received signal $r(t)$ four times but for this case each 90^0 out of phase from each other and then take product of these four terms [19]. The product of these four terms is taken in order to determine if it can result into signal which is proportional to the derivative obtained in Eq. (3.32).

The received QAM signal is $r(t)$, then the four signals are defined as,

$$x_1(t) = \mathrm{LPF}\{r(t)\cos(2\pi ft + \theta)\},$$

$$x_2(t) = \mathrm{LPF}\left\{r(t)\cos\left(2\pi ft + \theta + \frac{\pi}{4}\right)\right\},$$

$$x_3(t) = \mathrm{LPF}\left\{r(t)\cos\left(2\pi ft + \theta + \frac{\pi}{2}\right)\right\},$$

$$x_4(t) = \mathrm{LPF}\left\{r(t)r(t)\cos\left(2\pi ft + \theta + \frac{3\pi}{4}\right)\right\}$$

where $r(t) = m_1(t)\cos(2\pi f_c t + \emptyset) - m_2(t)\sin(2\pi f_c t + \emptyset)$.

$x_1(t)$ can be expressed as follow assuming $f = f_c$;

$$x_1(t) = \mathrm{LPF}\left\{\frac{1}{2}[m_1(t)(\cos(2\pi f_c t + \emptyset)\cos(2\pi ft + \theta)) - m_2(t)(\cos(2\pi f_c t + \emptyset)\cos(2\pi ft + \theta))]\right\}, \tag{3.33}$$

from cosine-sine product identities shown below,

$$\cos(A) + \cos(B) = 2cos\left(\frac{A+B}{2}\right)cos\left(\frac{A-B}{2}\right),$$

and

$$\sin(A) + \sin(B) = sin\left(\frac{A+B}{2}\right)cos\left(\frac{A-B}{2}\right).$$

$$x_1(t) = \mathrm{LPF}\left\{\frac{1}{2}[m_1(t)(\cos(\emptyset - \theta) + \cos(4\pi f t + \emptyset + \theta)) - m_2(t)(\sin(\emptyset - \theta) + \cos(4\pi f t + \emptyset + \theta))]\right\}, \quad (3.34)$$

then high frequency components in Eq. (3.34) are suppressed by a low pass filter resulting into,

$$x_1(t) = \frac{1}{2}\{m_1(t)\cos(\emptyset - \theta) - m_2(t)\sin(\emptyset - \theta)\}.$$

where, $m_1(t)$ and $m_2(t)$ are transmitted symbols from the transmitter. In the same sense $x_2(t), x_3(t)$ and $x_4(t)$ can be obtained as;

$$x_2(t) = \frac{1}{2}\left\{m_1(t)\cos\left(\emptyset - \theta - \frac{\pi}{4}\right) - m_2(t)\sin\left(\emptyset - \theta - \frac{\pi}{4}\right)\right\},$$

$$x_3(t) = \frac{1}{2}\{m_1(t)\sin(\emptyset - \theta) - m_2(t)\cos(\emptyset - \theta)\},$$

$$x_4(t) = \frac{1}{2}\left\{m_1(t)\cos\left(\emptyset - \theta - \frac{3\pi}{4}\right) - m_2(t)\sin\left(\emptyset - \theta - \frac{3\pi}{4}\right)\right\}.$$

The product of the above expressions reduce to a sine term as explained in [19], i.e,

$$x_1(t)x_2(t)x_3(t)x_4(t) = 4\epsilon^2(t)\sin(4\emptyset - 4\theta), \quad (3.35)$$

where $\epsilon^2(t) = m_1^2(t)m_2^2(t)$ as explained in [19].

The product shown in Eq. (3.35) produces an expression which proportional to the desired derivative in Eq. (3.32), thus adaptive Costas loop algorithm can be obtained as shown Eq. (3.36) [19].

$$\theta[k+1] = \theta[k] + \mu x_1(t)x_2(t)x_3(t)x_4(t), \text{where } t = kT_s, \theta = \theta[k], \quad (3.36)$$

where T_s is the time between consecutive algorithm updates.

Carrier recovery for QAM does not possess a single convergence since its performance function J_p shown in Eq. (3.30) has maxima at $\theta = \emptyset + x\frac{\pi}{2}$ at any given integer x, thus it may lead to phase uncertainties. These phase ambiguities can be rectified by [19];

- the message sequence at the transmitter should be differentially encoded in order for the transmitted message to be carried in successive changes in symbol values instead of being carried in symbol values themselves.

- correlation of down sampled message sequences with a known training signal at the receiver.

3.6.2.2 Timing Recovery Algorithm

A QAM signal is sent from transmitter, propagates through a transmission medium like space and received as an analog signal. In order to extract message sent, analog signal is demodulated and sampled at correct sampling times. To specify the correct instants upon which sampling should take place, time recovery is done.

From Eq. (3.24), two message symbols m_1 and m_2 are received, therefore two differently adaptive timing parameters τ_1 and τ_2 are used to obtain optimal times upon which sampling should take place. However, these two messages are related and need to be sampled at the same time. Therefore, one adaptive parameter τ is opted for to estimate the exact time when to sample s_1 and s_2 at the same time. The main goal is to determine adaptive parameter τ that optimizes the objective function Eq. (3.37). Consider averaging of fourth power of received signals to formulate an objective function as explained in [19].

$$J_T(\tau) = \text{avg}\{s_1^4 + s_2^4\},$$

where $s_1 = m_1(kT+\tau)$ and $s_2 = m_2(kT+\tau)$. Thus,

$$J_T(\tau) = \text{avg}\{m_1^4(kT+\tau) + m_2^4(kT+\tau)\}. \tag{3.37}$$

From steepest decent algorithm,

$$\tau[k+1] = \tau[k] - \mu\frac{\partial J_T(\tau)}{\partial\tau},$$

$$\tau[k+1] = \tau[k] - \bar{\bar{\mu}}\,\frac{\partial\big(m_1^4(kT+\tau) + m_2^4(kT+\tau)\big)}{\partial\tau}, \text{where } \tau = \tau[k]$$

$$= \tau[k] - \bar{\mu}\left[(m_1^3(kT+\tau) + m_2^3(kT+\tau)) \times \frac{\partial(m_1(kT+\tau) + m_2(kT+\tau))}{\partial\tau}\right]$$

The above expression can be numerically expressed as,

$$\begin{aligned}&\tau[k+1] = \tau[k] - \mu\big(m_1^3(kT+\tau[k]) + m_2^3(kT+\tau[k])\big)\\&\times[m_1(kT+\tau+\varepsilon) - m_1(kT+\tau-\varepsilon) + m_2(kT+\tau+\varepsilon)\\&- m_2(kT+\tau-\varepsilon)],\end{aligned}$$

given that,

$\mu = \frac{\bar{\mu}}{\varepsilon}$, where ε a small positive integer. By oversampling m_i, the timing offsets $\tau[k]$ and $\tau[k] \pm \varepsilon$ can be interpolated from the oversampled m_i as explained in [19].

3.6.2.3 Equalization for QAM

As a signal propagates through a transmission medium, it undergoes a lot of impairments which cause symbols to interact with one another. The interaction of symbols leads to ISI which can be caused by nonlinearity of the channel, multipath interferences and also interaction or overlapping of pulse shapes at the receiver. For simplicity, the equalizer mitigates the effect of the channel or can be taken as a filter designed at the receiver to undo the channel effects. The main goal is to determine adaptive parameter of equalizer coefficient g that optimizes the objective function. Consider a known training sequence $p[k]$ at receiver, g_i as equalizer or filter coefficients, $r[k-i]$ as received signals at the receiver and also $d[k]$ is obtained by subtracting a received signal from the training sequence.
Expressing equalizer output as the inner product of equalizer coefficients and with a vector of received signal as shown in Eq. (3.38) [19].

$$y[k] = \sum_{i=1}^{m} g_i r[k-i],$$

$$y[k] = (r[k], r[k-1], r[k-2], \dots\dots\dots, r[k-m]) \begin{pmatrix} g_1 \\ g_2 \\ \cdot \\ \cdot \\ g_m \end{pmatrix} \tag{3.38}$$

Therefore the output of equalizer can be shortened as shown in Eq. (3.39).

$$y[k] = F^T[k]g, \tag{3.39}$$

given that,

$$F^T[k] = \big((r[k], r[k-1], r[k-2], \dots\dots\dots, r[k-m])\big).$$

Eq. (3.40), shows the difference between the output of equalizer $y[k]$ and a known training sequence

$$d[k] = p[k] - y[k]$$
$$d[k] = p[k] - F^T[k]g. \tag{3.40}$$

Least mean square algorithm (LMS) is adaptive filtering method used to optimize the filter coefficient that relate with the difference between desired and actual signal. The filter or equalizer coefficients g are optimized by minimizing the LMS algorithm $J_{LMS}(g)$. The least mean square objective function is shown in Eq. (3.41) [19],

$$J_{LMS}(g) = \frac{1}{2}\text{avg}(d[k]d^*[k]), \tag{3.41}$$

where avg is averaging operation and $d^*[k]$ is complex conjugate.

However these terms are complex valued which can be can be taken real and imaginary parts. This is because QAM is modelled as two signals of real and imaginary parts of a single complex valued signal [19]. Therefore,

$$d[k] = d_R[k] + jd_I[k],$$
$$p[k] = p_R[k] + jp_I[k],$$
$$F[k] = F_R[k] + jF_I[k],$$
$$g[k] = g_R[k] + jg_I[k].$$

Adaptive element g as a filter coefficient can be optimized by minimizing the objective function Eq. (3.41) completed by steepest descent algorithm.

$$g[k+1] = g[k] - \bar{\mu}\left\{\frac{\partial}{g_R}(d[k]d^*[k]) + j\frac{\partial}{g_I}(d[k]d^*[k])\right\}, \tag{3.42}$$

where $g_R = g_R[k],\ g_I = g_I[k]$ and sign $*$ represents a complex conjugate.

From Eq. (3.41), the output of the equalizer $y[k] = F^T[k]g$ can be expanded since $F^T[k] = F_R^T[k] + jF_I^T[k]$ and $g = g_R + jg_I$. Thus,

$$\begin{aligned} F^T[k]g &= (F_R^T[k] + jF_I^T[k])(g_R + jg_I) \\ &= F_R^T[k]g_R + jF_R^T[k]g_I + jF_I^T[k]g_R - F_I^T[k]g_I \\ &= (F_R^T[k]g_R - F_I^T[k]g_I) + j(F_R^T[k]g_I + F_I^T[k]g_R), \end{aligned} \tag{3.43}$$

the conjugate of Eq. (3.43) is shown below in Eq. (3.46),

$$(F^T[k]g)^* = (F^T[k]g_R - F_I^T[k]g_I) - j(F_R^T[k]g_I + F_I^T[k]g_R) \tag{3.44}$$

Note that, $z = x + jy$ and $z^* = x - jy$, then $zz^* = x^2 + y^2$.

Therefore, $d[k]d^*[k] = d_R^2[k] + d_I^2[k]$.

$$\frac{\partial d[k]d^*[k]}{g_R} = \frac{\partial d_R^2[k] + d_I^2[k]}{\partial g_R}$$

$$= \frac{\partial d_R^2[k]}{\partial d_R[k]} \frac{\partial d_R[k]}{\partial g_R} + \frac{\partial d_I^2[k]}{\partial d_I[k]} \frac{\partial d_I[k]}{\partial g_R}$$

$$-2d_R[k]F_R[k] - 2d_I[k]F_I[k],$$

where,

$$\frac{\partial d_R[k]}{g_R} = -F_R[k] \text{ and } \frac{\partial d_I[k]}{g_R} = -F_I[k].$$

This is true due to the fact that, training sequence p does not necessary depend on equalizer coefficient g. Therefore, $\frac{\partial p}{\partial g} = 0$.

In the same sense,

$$\frac{\partial d[k]d^*[k]}{g_I} = 2d_R[k]F_I[k] - 2d_I[k]F_R[k]$$

Therefore, the update in Eq. (3.42) results into Eq. (3.45) as explained in [19].

$$g[k+1] = g[k] + \bar{\mu}\big(d_R[k]F_R[k] + e_I[k]F_I[k] + j(-d_R[k]F_I[k] + d_I[k]F_R[k])\big)$$

$$g[k+1] = g[k] + \mu d[k]F^*[k] \qquad (3.45)$$

Let's consider a scenario where an equalizer has done its job and the eye of channel is opened. This indicates that, all the decisions are perfect but it does not necessary mean that the equalizer parameters are at optimal values. In that situation, the output of equalizer decision device is nothing but an exact copy of delayed source [19]. Consider a sign operator taken as equalizer decision device while taking into account a binary source ± 1. Hence, the recovery error $d[k]$ is computed as shown in Eq. (3.46) [19].

$$d[k] = \text{sign}(y[k]) - y[k], \qquad (3.46)$$

where $y[k]$ is the output of the equalizer and sign$\{y[k]\}$ is delayed source $s[k-\delta]$ as illustrated in [19]. Therefore, adaptive LMS equalization algorithm explained in Eq. (3.46), is replaced by adaptive decision directed LMS equalization whose update is illustrated in Eq. (3.47) [19], which is implemented in Matlab software to complete equalization block.

$$g[k+1] = g[k] + \mu(s\text{ign}(y[k]) - y[k])F^*[k]. \qquad (3.47)$$

3.6.3 Correlation

A correlator is a very important tool in communication systems because it aligns two signals, where it shifts one of the signal in time domain and determines how easily it can match with the other at each and every shift. This task of correlation is completed by a correlator by doing point to point multiplication and then add them together. After adding them together, if their summation is small, it indicates that they are not so much similar, and if their summation is big or large, it indicates that most of their terms are alike. This knowledge of correlation can also be used in finding the appropriate times to sample the transmitted symbols and also to find the start of the message as explained in [19]. Consider two discrete time sequences $x[k]$ and $y[k+i]$, cross correlation is a function of time shift i between those two signals as shown in Eq. (3.48) [19].

$$R_{xy(i)} = \lim_{T\to\infty} \frac{1}{T} \sum_{-\frac{T}{2}}^{\frac{T}{2}} x[k]y[k+i], \tag{3.48}$$

where $\frac{1}{T}$ is a normalization factor which can be ignored as the case with Matlab software that ignores it while computing cross correlation.

CHAPTER 4

METHODOLOGY

There are different modules and algorithms used in software defined radio environment in transmission and reconstruction of a signal. This chapter details Matlab implementation of those algorithms explained in Chapter 3. The real signal of interest to be reconstructed by a receiver, was a human voice speech signal recorded for 10 seconds. The whole speech signal was not transmitted but rather, a compressed version of it (linear predicted parameters) was (were) transmitted as explained in Chapter 2, and the speech signal was reconstructed at the receiver.

4.1 Channel Impairments and Disturbances

The software defined radio to reconstruct the speech signal from its parameters was assumed to be mitigating impairments like; difference in carrier signal frequency and phase at transmitter and receiver, timing offset, intersymbol and multipath interference, timing and phase noise. The disturbances or impairments were implemented in Matlab as follows.

Timing and frequency offset. The frequency and timing offset were assumed to be a decimal number falling in the range between 0 and 1 while carrying out Matlab simulations. In other words, frequency offset was specified as zero and timing offset was specified as 0.25.

Timing noise. Timing noise was modelled as random walk for Matlab simulation. However, it was equally important to have a simple insight about random walk. A random walk can be briefly explained as a process where a current value of the variable x_t consists of a past value of the variable x_{t-1} plus an error ε_t, as illustrated in Eq. (4.1) [29].

$$x_t = x_{t-1} + \varepsilon_t, \tag{4.1}$$

where ε_t is independent and identically distributed variable with zero mean and variance σ^2. Consider a random walk with a stochastic sequence $\{x_n\}$ shown in Eq. (4.2) [30],

$$x_n = \sum_{k=1}^{n} y_k, \tag{4.2}$$

given that $x_0 = 0$.

The random walk is a simple random walk if and only if $y_k = \pm 1$ given that, the probability $p(y_k = -1) = n$ and the probability $p(y_k = 1) = 1 - n$, provided that $p(y_k = -1) + p(y_k = 1) = 1$.

In Matlab software, the error ε_t was implemented in Matlab software as white noise by calling a Matlab software built-in function $randn$ which passed a one and length of the message, and then multiplied it with the square root of specified value of timing jitter noise variance σ_t^2. Then, a Matlab function $filter$ was called and passed the generated white noise, filter numerator coefficient as 1 and filter denominator coefficients as $[1 - 1]$. The output of $filter$ command was taken as timing jitter [19]. A time plot of timing jitter modelled when the value of noise variance was specified as 1×10^{-3}, is shown below in Figure 4.1.

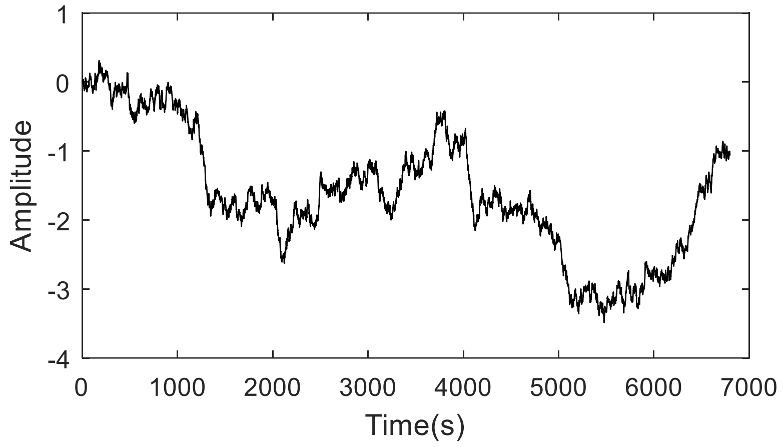


Figure 4.1: Time plot of random walk.

Phase offset. The value of phase offset was specified as $\frac{\pi}{6}$ and it was constant per communication session (for entire time of simulation).

Phase noise. Phase noise was also modelled as random walk for Matlab simulation. Phase noise has to do with random phase fluctuations of periodic signal. An ideal oscillator would produce a pure periodic wave. In frequency domain, this can be characterised by a single a pair of Dirac delta function at oscillator frequency. In other words, all signal power is at single frequency. However, if phase noise is present within the signal, phase noise spread the power of the signal to adjacent frequencies, resulting into noise side bands [31]. For Matlab simulation, phase noise, phase and frequency offset, were added to carrier expression $\exp(j(2\pi f_c(1-f_{off})T_s+(p_{off}+p_{noise})\ )$. Note that, f_{off} is frequency offset and was specified as zero, p_{off} is the phase offset specified as $\frac{\pi}{6}$, p_{noise} is phase noise and was modelled as random walk and f_c is carrier frequency which was specified as 250 Hz [19].

Intersymbol interference (ISI). Intersymbol interference was modelled in Matlab software as additive white Gaussian noise process. Additive white Gaussian noise is a simple noise model which is used in communication systems to imitate the random processes or random noise interferences that occur in practice. However, the model is;

- Additive, since it is added to the transmitted signal. Neglecting all imperfections as the signal propagates from the transmitted to the receiver except noise, the received signal $r(t)$ is equivalent to the transmitted signal $s(t)$ plus the noise $n(t)$ as illustrated below,

$$r(t)=s(t)+n(t).$$

- White, since it has a constant power throughout the whole frequency band.
- Gaussian, since it has a Gaussian (normal) probability distribution function $f(x)$ with zero mean and standard deviation σ as shown below [22],

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}.$$

Additive white Gaussian was implemented in Matlab software by calling a Matlab built-in function $randn$ which passed a one and the length of transmitted signal as its input arguments, and then multiplied with the square root of noise variance σ^2, completed by calling a Matlab built-in function $sqrt$ which computed the square root of noise variance σ^2. The average energy E_s of 4-QAM constellation with a minimum distance of 2 was computed, and then divided by the assumed SNR, i.e, $\left(\frac{E_S}{SNR}\right)$, to come up with the noise variance σ^2. However, additive white Gaussian noise can also be implemented in Matlab software by calling a Matlab built-in function $awgn$ which passes a transmitted signal and specified SNR value per sample in dB as its input arguments. The magnitude spectrum of AWGN implemented in Matlab is shown in Figure 4.2.

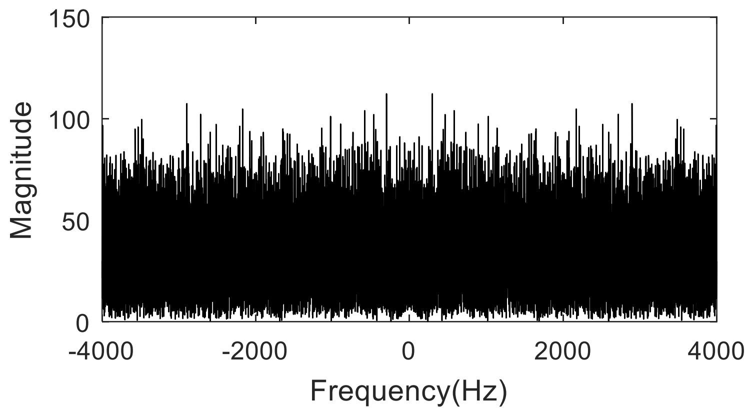


Figure 4.2: Magnitude spectrum of ISI

Multipath interference. The multipath channel was modelled as a fixed multipath channel at a given time. In other words, the multipath channel was assumed to be having the same channel gains and delays for the entire time of simulation. For example, the receiver was tested on different channels such a one tap unity channel $[1]$, a two tap channel or a unit delay channel $[0,1]$ and finally, a three tap channel $[-0.3,1,0.7]$. This was done to assert how each channel affects the receiver in reconstruction of a speech signal.

4.2 Source Coding Block

The real signal of interest to be transmitted was speech signal (recorded human voice), but instead of transmitting the whole speech samples of original signal, the extracted linear predicted parameters from the recorded speech signal, were instead transmitted. Linear predictive coding algorithm (as explained in Chapter 2) accomplished source coding block. For source coding, a recorded human voice was used as input to LPC encoder (as shown in Figure 2.18). The LPC encoder extracted out LPC parameters which include, LPCs (filter coefficients), gain, voicing and finally pitch period parameters. These parameters were then sent to LPC decoder (shown in Figure 2.19) to model a speech from its parameters. Linear predictive coding algorithm is a lossy compression technique, therefore, at the out of LPC decoder, a synthetic speech signal was produced. However, it was at a lower quality as compared to original signal. This was demonstrated by looking at its time plot or by listening to both original and compressed audio sound signal. The time plot for both original and compressed signal is shown in Figure 4.3.

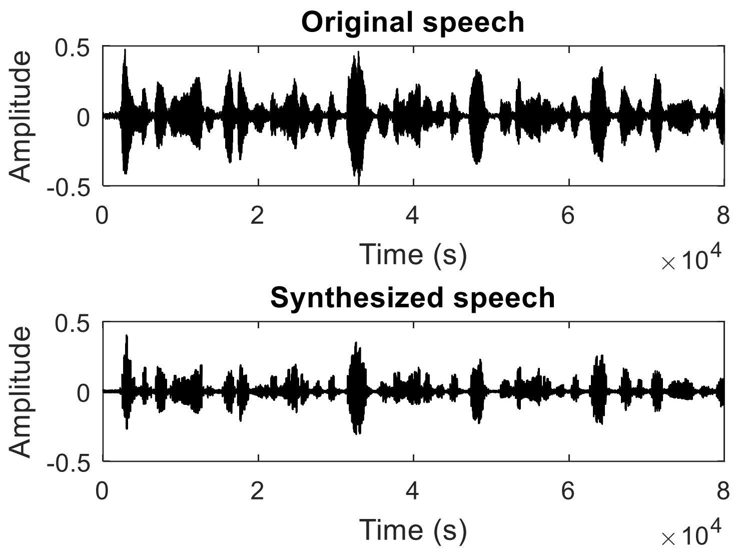


Figure 4.3: Time plot of Original and Compressed speech signal

The output from the source encoder (compressed speech signal) was fed to the channel encoder to protect the bit stream before transmission to the channel. The block diagram shown in Figure4.4, completed end to end communication which was implemented in Matlab software to reconstruct a speech signal at the output of the receiver. The simulator block implemented in this thesis is shown in Figure 4.4.

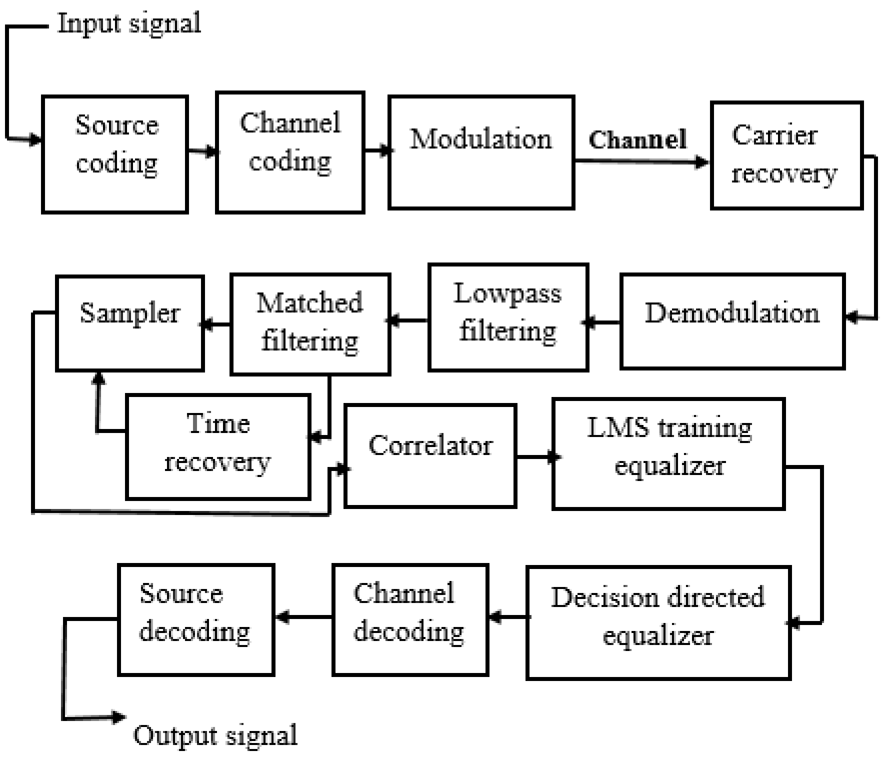


Figure 4.4: Simulator block

4.3 Channel Coding

Channel coding adds redundancy to the transmitted bit stream before transmission to channel. Binary mapping of input signal was done and coded at the transmitter using linear block code algorithm to accomplish channel coding block [19]. The bit stream was then converted into analog signal by using square root raised cosine filter (SRRC) pulse with 0.25 as the roll-off factor. The value of 0.25 was taken as the value of the roll-off factor for SRRC pulse, this is because when implementing adaptive algorithms explained in Chapter 3, it contributed to their optimization such as fourth power adaptive algorithm [19]. However, the value of roll-factor falls in the range between zero and one [22].

A random sequence of length 25 was used to reinstate signal whiteness (in order to make the spectrum flatter by decorrelating the message), since channel coding at the transmitter correlates the intended transmitted message [19]. Scrambling of the coded message was done by “exclusive-or”ing the message with length of 25 random sequence. The receiver had to be synchronized with scrambled message used at the transmitter, for perfect descrambling of the transmitted message, and this was assisted by equalizer training sequence. Descrambling was also completed at the receiver by “exclusive-or”ing the message with the same length of 25 random sequence used at the transmitter [19].

4.4 Modulation

The analog signal was up converted or modulated with a carrier frequency of 250 Hz. From the output of a pulse shaping filter, its real part is multiplied with the expression $\exp(j(2\pi fc(1-f_{off})T_s+(p_{off}+p_{noise}))$ [19], thus accomplishing modulation as illustrated in Eq. (3.20). Note that, f_{off} is equalling to zero, p_{off} is equalling to $\frac{\pi}{6}$, p_{noise} was modelled as random walk whose noise variance was specified as 0.005, T_s is the sampling period was equalling to 400 seconds and fc is equalling to 250 Hz. The transmitted signal spectrum was distorted by a phase noise of variance 0.005 as shown Figure 4.5.

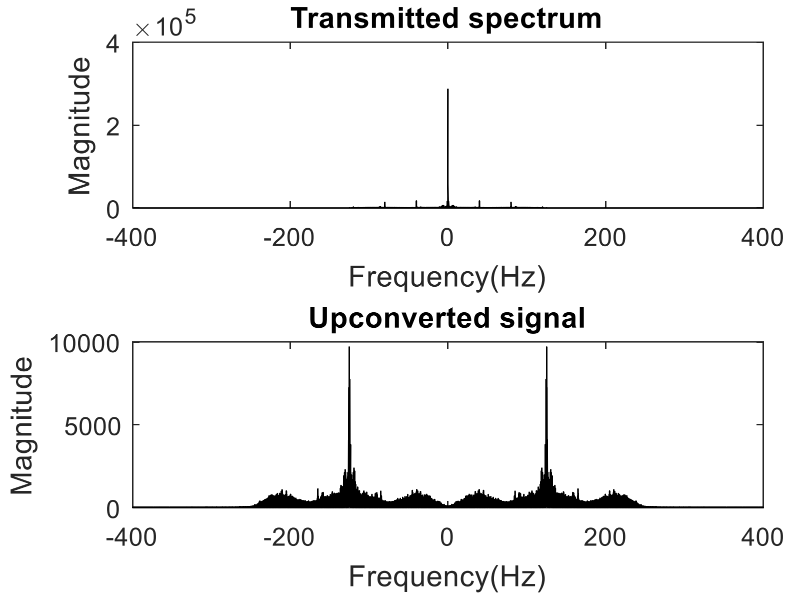


Figure 4.5: Magnitude spectrum of up converted and original signal

4.5 Channel

The modulated signal was passed through different channels, i.e, $[1], [0, 1]$, and $[-0.3, 1, 0.7]$ in addition to other noise impairments such as ISI, phase and timing noise with different assumed variances. This was done to investigate how each channel affects the reconstruction of speech signal from its parameters, since the assumed channels are having different channel gains hence attenuating the transmitted signal differently.

4.6 Carrier Recovery and Demodulation

The up converted signal propagated over the channel and received at the receiver as a noisy signal. For proper demodulation, both phase and frequency used at the transmitter should be known at the receiver. Therefore, this synchronization process was completed by adaptive Costas loop algorithm. The incoming signal entered the carrier recovery block, the phase and frequency estimates were completed by adaptive Costas loop algorithm, and then the signal was down converted by multiplying the transmitted signal with expression $\exp(-j(2\pi f_{est}Ts + p_{est}))$ [19], thereby accomplishing demodulation as illustrated in Eq. (3.21). Note that, f_{est} is the frequency and p_{est} is the phase estimated at the receiver. However, for proper demodulation, one of the replica had to be centred at zero frequency [19][22].

4.7 Lowpass filtering

The signal of interest is centred at zero frequency. After demodulation, one of the replica is centred at zero as shown in Figure 4.6. Therefore to remain with only this replica, a low pass filter was implemented by calling a Matlab function f$ilter$ that passed the normalized cut off frequency values [19]. Since the signal of interest was assumed to be contained within a frequency range of approximately ± 100 Hz. The signal was low passed to remain with only the signal in the band of interest. Note that, to illustrate lowpass filtering operation, the noise variance (phase noise and timing jitter) were varied to zero, to investigate if smooth convergence of adaptive Costas loop algorithm can be achieved, hence proper demodulation of transmitted signal. The magnitude spectrum of low passed signal is shown in Figure 4.6, which is centred at zero frequency, when the channel was specified as a unity channel.

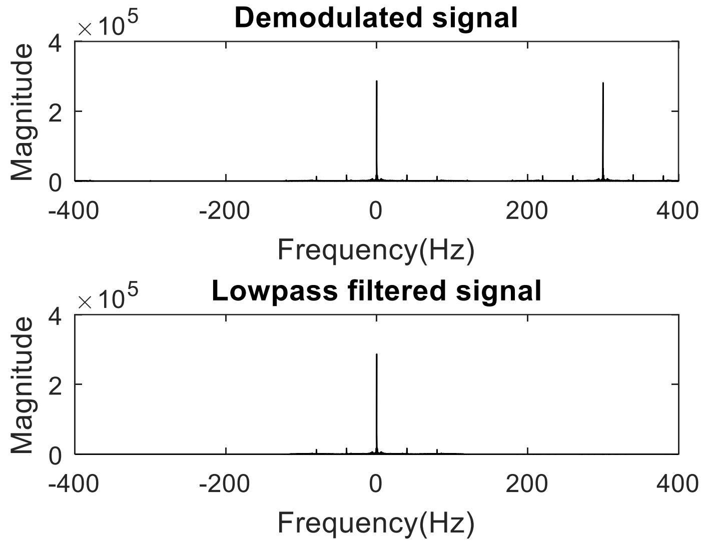


Figure 4.6: Magnitude spectrum of demodulated and low passed signal

4.8 Matched filtering and Timing Recovery

Time recovery block was implemented by fourth-power maximization algorithm as explained in Chapter 3 and implemented in Matlab software [19]. The signal was sent to timing recovery block and interpolated it using SRRC pulse shaping without any timing offset since timing offset was adaptively estimated, then received pulses were matched with transmitted pulse shapes. The over sampled signal was down sampled to reconstruct the transmitted symbols.

4.9 Correlation

Correlation was done to train the equalizer, this was done by finding the training segment which trains the equalizer hence the correlator relates the signal with a pre-defined training sequence. As soon as the training sequence (segment) was found, the starting position of training segment was sent to the equalizer and went along with the training sequence [19].

4.10 Equalization Block

Equalization was accomplished by both trained LMS and decision directed LMS adaptive algorithm, and implemented in Matlab software [19]. The signal to be equalized was first passed through trained adaptive LMS algorithm block and when the training sequence finished, the decision directed adaptive algorithm accomplished equalization to mitigate channel effects.

4.11 Decoding

As soon as descrambling of a message signal from the output of decision directed equalization finished, channel decoding was done by multiplying the received codeword with the parity check matrix to generate the transmitted signal (compressed speech signal). The block diagram shown in Figure 2.18, completed the conversion of LPC coded signal to an original speech signal as explained in Chapter 2.

For comparison, the block diagram shown in Figure 4.7, was used for performance evaluation, using measurement parameters in terms of mean square error (MSE), symbol error rate (SER) and bit error rate (BER). From Figure 4.7, the output of the equalizer (decision directed equalizer) was compared with the original transmitted parameters (at the input of channel encoder) to estimate the mean square error of the equalizer. Therefore, the mean square difference between the outputs of decision directed equalizer and the input of channel encoder, gave the value of MSE. Numerically, mean square error of output data can be expressed as shown in Eq. (4.3) [29].

$$\text{MSE} = \frac{1}{M}\sum_{x=1}^{M}[y(x) - \hat{y}(x)]^2, \tag{4.3}$$

where M is the total number of samples, $y(x)$ and $\hat{y}(x)$ are outputs data at the output of source encoder and output decision directed equalizer respectively.

Furthermore, the pre-decoded symbol message (at the output of equalizer) was compared with the original transmitted coded symbols to estimate the symbol error rate. Bit error rate was estimated by comparing the transmitted data (at output of the source encoder) with the decoded data (at output of channel decoder) at the receiver. To evaluate the effect of channel errors in reconstruction of a coded speech signal, the performance parameters obtained in reconstruction of speech signal such as bit error rate, symbol error rate, and mean square error were obtained for different channel taps and noise variances [4]. Finally, the performance evaluation plots regarding these comparisons were plotted, to investigate how they vary with SNR in the channel [19].

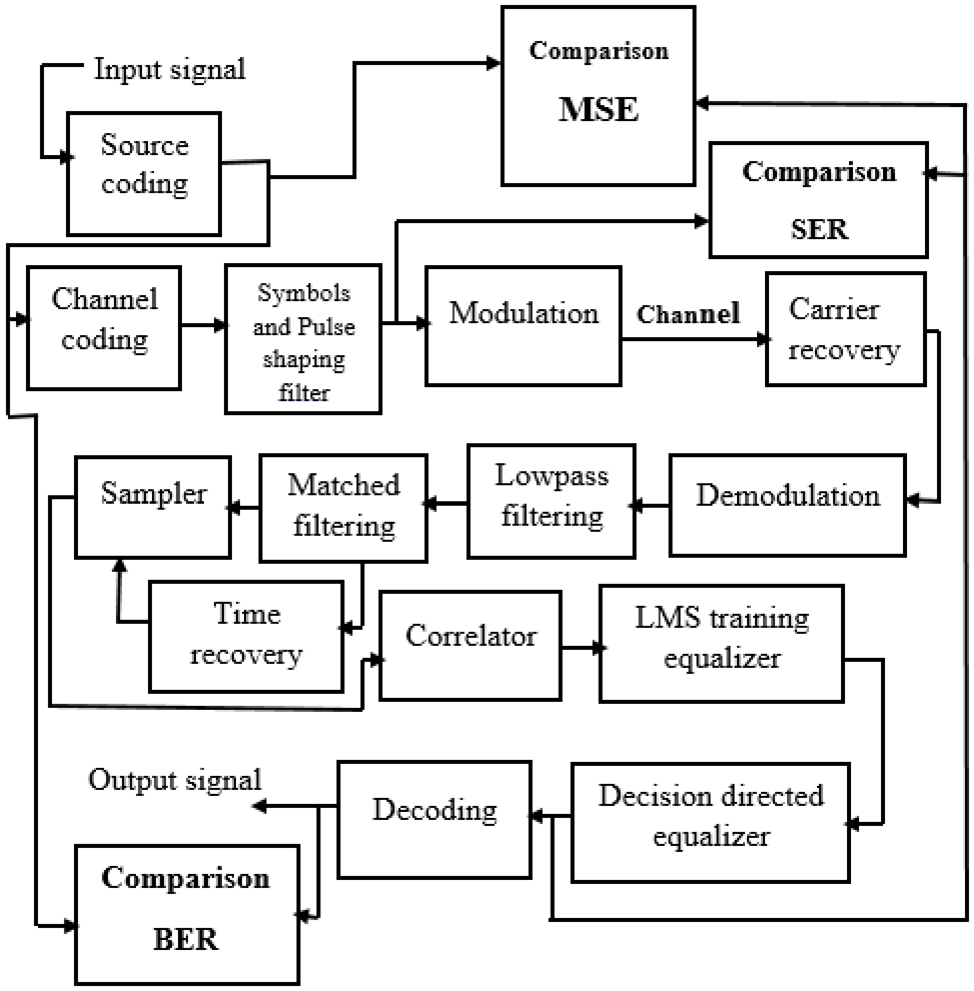


Figure 4.7: Performance evaluation

CHAPTER 5

RESULTS AND DISCUSSION

This chapter discusses the research findings while analysing the results. The results and discussions are following the specific objectives of this thesis as outlined in Chapter 1 and implemented in Matlab software in Chapter 2 and 4, which includes;

- determining algorithms in encoding and also in synchronization.
- modelling and simulating software defined radio architecture.
- evaluating the effects of channel errors on performance of software receiver in reconstruction of a compressed speech signal.

5.1 Source Encoder and Decoding Results Analysis

Source encoding and decoding were accomplished by linear predictive coding algorithm as explained and implemented in Chapter 2. By implementing LPC encoder (shown in Figure 2.17) in Matlab software, LPC parameters were extracted from the speech signal. The purpose of LPC algorithm implemented, was to compress the input speech signal such that it could be represented with less redundancy. For example, an original speech signal had 80000 samples, however, when compressed by LPC algorithm, its samples reduced to 79920 samples. Therefore, a redundancy of 80 samples was removed. However, the only cost paid was the reduction in quality of a compressed speech signal. The extracted LPC parameters from an original signal included, LPCs (filter coefficients), gain, voicing and pitch period parameters. These parameters were used by LPC decoder (shown in Figure 2.18) to synthesis the speech signal (to convert LPC coded signal to original speech signal). At the output LPC decoder, a synthetic speech was produced even though, it was of lower quality in comparison with an original signal. This is explained by the fact that, LPC algorithm is a lossy compression technique, thus it produces a signal which is at a lower quality than its original version [23].

At the LPC encoder, white noise was used as excitation signal when it comes to unvoiced signal frames. Therefore, the output wave forms may seem different from the original signal. This is because, white noise is random even though the same LPC parameters were used to synthesis speech signal at the decoder. However, the most important thing is that, the power spectrum density of original speech signal is very close to that of synthetic speech signal, (as shown in Figure 5.1) due to flat spectrum of white noise. The power spectrum density of original and synthetic speech is shown in Figure 5.1.

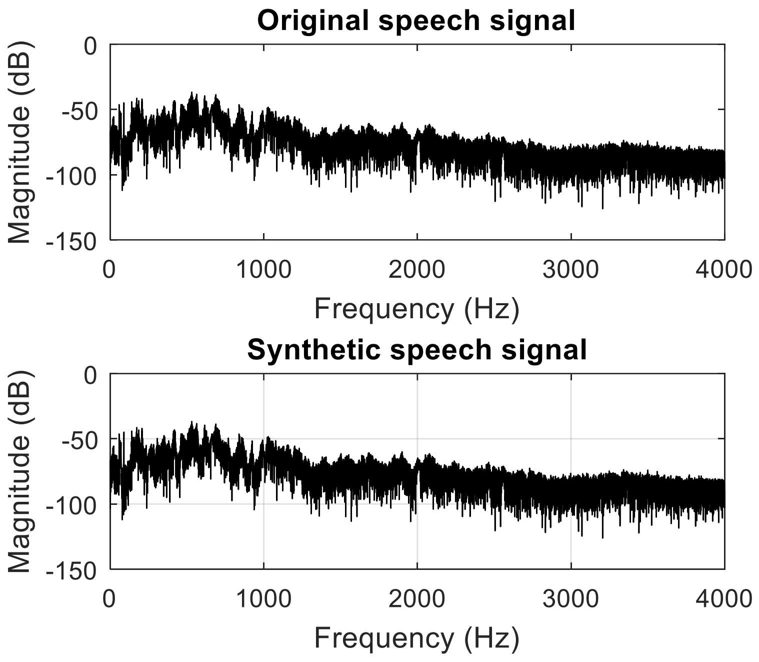


Figure 5.1: Magnitude spectrum of both original and synthetic speech signal

Even though magnitude spectrum of original is very close to that of synthetic speech signal, all phase information of original speech signal is lost, preserving only the magnitude spectrum of the original speech signal [7]. However, the synthetic speech sounded almost the same as original speech signal. This was evidenced by listening to both original and compressed audio sound signal, and the compression can be illustrated by looking at their time plots. The time plot for both original and compressed signal is shown in Figure 5.2.

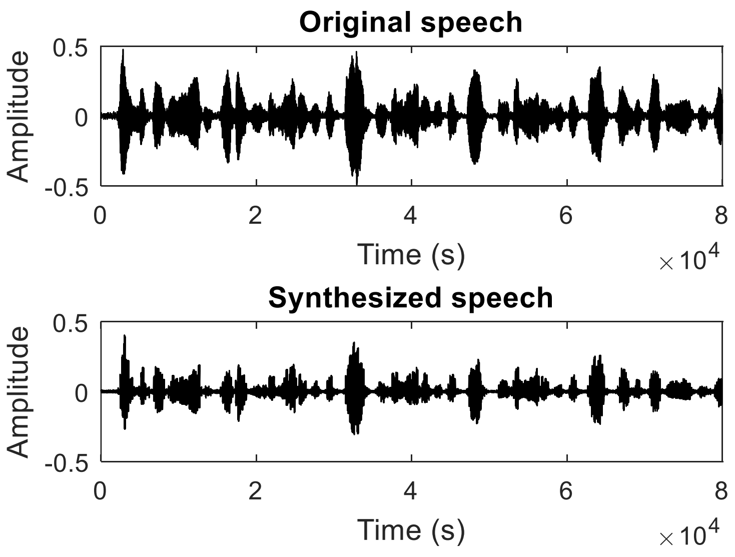


Figure 5.2: Time plot for original and synthetic speech signal

The coded speech signal was sent to channel encoder to add structured redundancy such that a bit stream is protected against errors before transmission over a communication channel. Finally, the end to end communication was completed by a block shown in Figure 4.4.

5.2 Carrier synchronization results analysis

The up converted signal from the transmitter propagated over a channel as analog signal and received at the receiver with high frequency components. This thesis considered a software receiver with analog front end, the incoming analog signal was first down converted (demodulated), such that sampling could take place at a lower sampling rate [22]. In order for accurate demodulation to take place as illustrated in Eq. (3.23), both phase and frequency at the transmitter should be the same as those at the receiver. In ideal situations, the transmitter and receiver are assumed to be synchronized, which means that, both frequency and phase are the same at the transmitter and at the receiver [19]. However, if the signal propagates over a noisy channel like for the case of this thesis, both frequency and phase at the transmitter differ greatly (or may differ slightly) from those at the receiver [22]. Adaptive Costas loop algorithm as illustrated in Eq. (3.36), was implemented in Matlab software [19]. Adaptive Costas loop algorithm continuously estimated and also eliminated the unknown phase and frequency offsets at the transmitter and the receiver, provided that, a good choice of algorithm step size was made [19]. The phase offset value was assumed to be $\frac{\pi}{6}$, and tested on a unity channel [1], assuming that both phase noise and timing jitter variance were equalling to zero. The adaptive Costas loop converged to a correct value as shown below in Figure 5.3. When a channel was modified with a unit delay [0, 1], and with the same step size of 0.5, again the algorithm converged to correct value as shown in Figure 5.3. This is so because, the algorithm continuously or adaptively estimated the offset over time, until it determined the correct value.

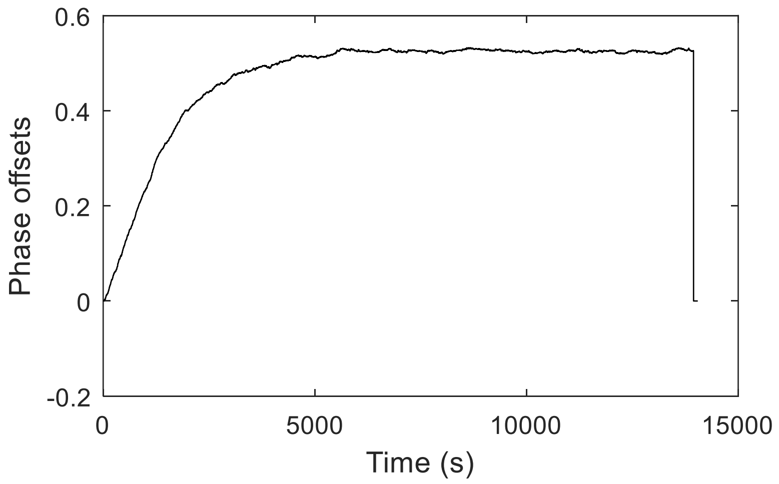


Figure 5.3: Phase offset estimation

Steepest descent algorithm shown in Eq. (3.2), was used as an adaptive algorithm to implement adaptive Costas loop algorithm. For smooth convergence, a small step size is used even though it takes long to achieve the desired convergence, and the long processing time is reduced by increasing the step size. However, for practical implementation like in the case of this thesis which takes into account a noisy channel, large step size leads to divergence. And once the phase and the frequency used at the transmitter diverge from those at the receiver, the transmitted analog signal cannot be properly demodulated, which leads to errors at sampling instants [19].

When a signal was made to pass through a three tap channel $[-0.3, 1, 0.7]$, using the same step size as the case of unit delay and unit channel, the algorithm diverged to a wrong value. This complicated demodulation and also resulted into a symbol error rate of 0. 42, hence the symbol error is approximately 42% which is too big for an acceptable communication system [22]. Due to wrong choice of step size, adaptive Costas loop algorithm diverged and the analog signal was not properly demodulated which also complicated sampling while recovering the transmitted symbols. Due to high symbol error rate, the transmitted symbols are not recovered, resulting into a bit error of 37% and the transmitted speech signal was not reconstructed by the receiver. Reducing the step size to 1×10^{-3} in order to allow smooth convergence, when adaptive Costas loop was again subjected to the same three tap channel, the bit rate reduced to 1.5×10^{-2}. Even though the decoded speech signal is a bit noisier. Therefore, it gives an insight that the choice of step is very important for adaptive Costas loop implementation. In other words, it should not be too be big, neither should it be very small [15].

5.3 Clock Synchronization Results Analysis

A speech signal was sent from transmitter, propagated through a transmission medium, which was modelled as a noisy channel, and received at receiver as an analog signal. In order to extract message sent, analog signal had to be sampled at correct sampling times [19]. To specify the correct instants upon which sampling should take place, time recovery was done. Time recovery was completed by fourth power maximization algorithm (as shown in Eq. (3.8)), and implemented in Matlab software [19]. Fourth power adaptive maximization algorithm was tested on three different channels to assert its effectiveness in estimating the timing offset, in other words the best times to sample.

When the timing offset value was set to be 0.25, and timing offset noise modelled as random walk and specifying its variance as zero. Then, the algorithm was tested on a unity channel, in addition of other noise interferences like phase noise and ISI, fourth power maximization algorithm converged to the correct value of 0.25 as shown in Figure5.4. Modifying the channel as a unity delay $[0, 1]$, the algorithm converged to -0.8. However, when the channel is $[-0.3, 1, 0.7]$, the algorithm converged to -1.5, which deviated from the true value of timing offset.

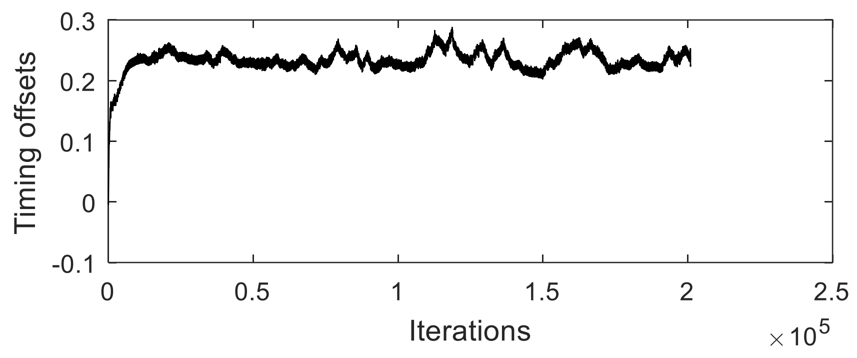


Figure 5.4: Timing offset estimation

The presence of too much noise in the channel, such as ISI, changes the convergent value of the fourth power maximisation algorithm [19]. For example, for a unit delay, the algorithm changes the estimates of the timing offset, to a value that maximises the power output as the algorithm suggests. This power is maximised to account for the added delay [19]. When a channel is more complicated, for example $[-0.3, 1, 0.7]$, the algorithm again moves the estimate to that value that maximises the output power. However, the divergence from the correct value of timing offset also leads to change in channel ISI, which is fixed by equalization block to enable smooth decoding of a transmitted symbols, provided that the noise is not too immense [19]. This explains why, the compressed speech signal was reconstructed at the receiver, even though in two cases the correct value of timing offset was not attained. However, the quality of reconstructed signal was a bit low.

5.4 Simulation results of software receiver

As explained in Chapter 4, the software radio was implemented in Matlab software to assert its effectiveness while reconstructing a compressed speech signal, in presence of channel errors. The parameters for each and every block are specified (assumed if applicable). Finally, they are passed through Matlab functions as arguments. Fixed channels taps that is to say $[1], [0, 1]$ and $[-0.3, 1, 0.7]$, were used for simulation of software receiver and they were constant for entire time of simulation. However, as explained in Chapter 4, all other noise interferences such that timing jitter, phase noise and ISI are added to the signal, and their variances varied to assert how channel errors affect software receiver in reconstruction of a compressed speech signal. The adaptive

LMS agorithm (shown in Eq. (3.45)) was implemented in Matlab softare, and its output is fed to decision directed equalizer (shown in Eq. (3.47)) to complete equalization [19].

The simulation results in terms of BER, MSE and SER (as explained in Figure 4.7) are shown in Table 5.1 below. For the results obtained in Table 5.1, it was assumed that, the variances of phase noise and timing jitter were assumed to be equivalent to zero. However, the variance of ISI was kept constant, specified as 2×10^{-8}.

Table 5.1: Table of results one

Channel	Description	Results
$[1]$	BER	1.2×10^{-9}
	SER	1.9×10^{-5}
	MSE	4.3×10^{-4}
$[0,1]$	BER	1.5×10^{-8}
	SER	1.9×10^{-4}
	MSE	3.4×10^{-3}
$[-0.3, 1, 0.7]$	BER	1.2×10^{-6}
	SER	1.9×10^{-3}
	MSE	1.7×10^{-2}

In the three cases, the compressed speech signal was reconstructed, and its quality when listening to it, was almost the same as the original transmitted signal. This is explained by the fact that, channel errors were not too immense, in additional to adaptive nature of the receiver's algorithms, they iteratively estimated all the offsets and the equalizers (both trained LMS and decision directed) were able to un do the channel effects perfectly, such that, the reconstructed symbols were correctly decoded at the output of the software receiver. The values of variances for phase noise and timing jitter were adjusted to 1×10^{-3}, while still making the variance of ISI constant. The values obtained in terms of BER, SER and MSE are shown in Table 5.2.

Table 5.2: Table of results two

Channel	Description	Results
[1]	BER	3.685×10^{-1}
	SER	7.049×10^{-1}
	MSE	1.601×10^{0}
[0,1]	BER	3.688×10^{-1}
	SER	6.433×10^{-1}
	MSE	1.328×10^{0}
$[-0.3, 1, 0.7]$	BER	5.303×10^{-1}
	SER	9.784×10^{-1}
	MSE	1.691×10^{0}

In the three cases, the compressed speech signal was not perfectly reconstructed by the software receiver. This is illustrated by the values BER, SER and MSE, which are too big for acceptable communication system. This is because, for practical telecommunication systems, the acceptable minimum BER is 10^{-9}, meaning that the value beyond indicates that, the channel is almost ideal which is not the case for practical communication world. However, the maximum value of BER for data transmission is 10^{-3}, hence the value below indicates occurrence of too much errors [22]. Bit error rate is used as a tool in communication systems to show how successfully a receiver has decoded the transmitted data [22]. The receiver was unable to perfectly reconstruct a compressed speech, since the noise was too immense for the adaptive algorithm to estimate all the offset values thereby decoding the symbols perfectly. In other words, when listening to the speech signal at the out of receiver, the quality was very poor and not understandable. The variance values for phase noise and timing jitter were continuously varied, in order, to estimate the range upon which the software receiver could be able to perfectly reconstruct a compressed speech in presence of channel errors. When the values of noise variances adjusted to 1×10^{-4}, the results obtained are shown in Table 5.3.

Table 5.3: Table of results three

Channel	Description	Results
$[1]$	BER	1.12×10^{-8}
	SER	5.99×10^{-4}
	MSE	4.83×10^{-2}
$[0,1]$	BER	1.24×10^{-7}
	SER	3.40×10^{-3}
	MSE	5.76×10^{-2}
$[-0.3, 1,0.7]$	BER	1.43×10^{-1}
	SER	2.63×10^{-1}
	MSE	5.02×10^{-1}

As illustrated from Table 5.3, the receiver perfectly reconstructed a compressed speech signal for a unity channel $[1]$ and a unit delay channel $[0,1]$. Therefore, an insight was given for range of noise variance, which should be chosen for a receiver implemented in Matlab software to reconstruct the speech signal. For the receiver to perfectly reconstruct a compressed (coded) speech signal, the value of noise variance for phase noise and timing jitter, should be in the range between 0 and $y \times 10^{-4}$, where y is an integer, provided that a unity channel and a unit delay channel have been used. However, for a three tap channel $[-0.3,1,0.7]$, the range of noise variance should between 0 and $y \times 10^{-5}$. From Table 5.1, the values of MSE $(4.3 \times 10^{-4}$ and $3.4 \times 10^{-3})$ for a unity channel and unit delay channel are very small. Since mean square error was the square difference between the original signal and the output of an equalizer, this indicated that, equalization was perfectly done in mitigating channel errors. Looking at SER from Table 5.2 for all channels, they indicate that, a noisy channel affects compressed speech signal more and hence care must be taken for proper reconstruction of the compressed speech signal.

The investigation was also made to assert how the signal to noise ratio varies with the symbol error rate and bit error rate. For implementing signal to noise in Matlab software, its value was specified to be constant for entire simulation in the range between 5 and 20. From Figure 5.5, SER and BER have an inverse relationship with SNR. In other words, as SNR increases, the SER or BER reduces.

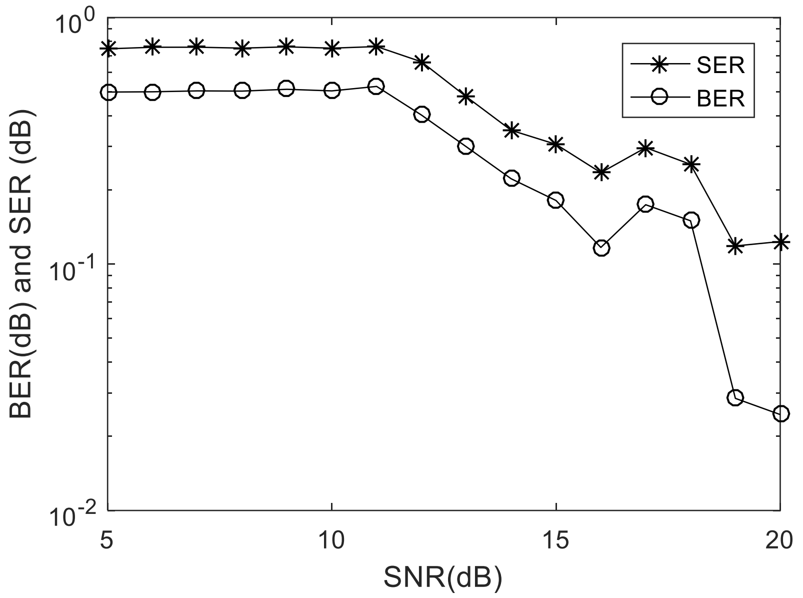


Figure 5.5: Symbol error rate and bit error rate against signal to noise ratio

From the results obtained, they indicate that, a highly compressed speech signal is very much affected by channel errors as it propagates through it. This is evidenced by a very small range of noise variance used, such that a receiver could reconstruct a compressed speech signal. Therefore, if the channel errors are so immense, the compressed speech signal cannot effectively be reconstructed at the receiver, and its quality will be very poor. This explains why the whole of receiver's implementation was based adaption using steepest descent algorithm, to counteract the adverse effects of the channel. If the channel is noisy like for the case of this thesis, adaptive algorithm continuously updates the algorithm until the offset values as a result of channel errors are determined [19]. This in turns lowers or even suppresses the would be errors in the speech signal before reconstruction at the receiver. In some instants where adaptive algorithms delivers than what is expected, it could be as a result of poor choice of the step size value. Steepest descent algorithm is a good adaptive algorithm if the choice of step size has been chosen very well otherwise, it may lead to divergence hence wrong results [15]

CHAPTER 6

CONCLUSION

A comprehensive study was successfully done investigating on how channel errors affect software receivers while reconstructing a highly compressed speech signal. In this work, linear predicted parameters where extracted from the author’s voice, which were transmitted through a noisy channel, and the speech signal was produced at the linear predictive decoder. Additionally, the main components of end to end communication were theoretically discussed and implemented in Matlab software. Investigations were done to assert the effects of channel errors in reconstruction of a speech signal, using measurement parameters such as, symbol error rate, bit error rate and mean square error. From this thesis, the following inferences are drawn:

- The quality of speech signal encoded by linear predictive algorithm reduces, at the output of the decoder.
- Linear predicted parameters are highly affected by channel errors.
- For perfect decoding of a transmitted signal by a receiver, it has to be synchronised with a transmitter.
- The choice of step size affects the adaptive algorithm derived from steepest descent algorithm. In other words, the step should not be too big neither should it be very small.

An experimental study based on Matlab implemented results is proposed to any researcher who would wish to undertake a similar study.

REFERENCES

[1] G. Bacci, F. Principe, F. Giannetti, and M. Luise, “Software-defined radio technologies for GNSS receivers: A tutorial approach to a simple design and implementation,” Int. J. Navig. Obs., vol. 2011, no. iii, 2011, doi: 10.1155/2011/979815.

[2] M. Lee, B. Keum, J. H. Jeong, Y. S. Shim, and H. S. Lee, “A software-based receiver running on a single DSP for terrestrial digital multimedia broadcasting,” IEEE Trans. Consum. Electron., vol. 54, no. 4, pp. 1894–1902, 2008, doi: 10.1109/TCE.2008.4711251.

[3] G. Bacci, F. Principe, F. Giannetti, and M. Luise, “Software-defined radio technologies for GNSS receivers: A tutorial approach to a simple design and implementation,” Int. J. Navig. Obs., no. January 2014, 2011, doi: 10.1155/2011/979815.

[4] A. Geramipour, M. Khazaei, A. Marjaninejad, and M. Khazaei, “Design of FPGA-based Digital PID Controller Using Xilinx SysGen for Regulating Blood Glucose Level of Type-I Diabetic Patient,” vol. 3, no. 7, pp. 56–69, 2013.

[5] D. Sinha, A. K. Verma, and S. Kumar, “Software defined radio: Operation, challenges and possible solutions,” Proc. 10th Int. Conf. Intell. Syst. Control. ISCO 2016, no. November, 2016, doi: 10.1109/ISCO.2016.7727079.

[6] O. V. Korniyenko and M. S. Sharawi, “GPS software receiver implementations,” IEEE Potentials, vol. 26, no. 3, pp. 42–46, 2007, doi: 10.1109/MP.2007.361644.

[7] W. C. Chu, "Algorithms Speech Coding Foundation and Evolution". 2003.

[8] S. K. Jagtap, M. S. Mulye, and M. D. Uplane, “Speech coding techniques,” Procedia Comput. Sci., vol. 49, no. 1, pp. 253–263, 2015, doi: 10.1016/j.procs.2015.04.251.

[9] Hamza Kadir (2021). Speech compression using Linear Predictive Coding (https://www.mathworks.com/matlabcentral/fileexchange/13529-speech-compression-using-linear-predictive-coding), MATLAB Central File Exchange. Retrieved December 10, 2021.

[10] Mr. Mohit Narayanbhai Raja, Miss. Priyanka Richhpal Jangid, Dr. Sanjay M. Gulhane, “Linear Predictive Coding,” vol. 4, no. 4, pp. 373–379, 2015.

[11] K. K. Paliwal and B. S. Atal, “Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame,” IEEE Trans. Speech Audio Process., vol. 1, no. 1, pp. 3–14, 1993, doi: 10.1109/89.221363.

[12] A. Harma, “Linear predictive coding with modified filter structures,” IEEE Trans. Speech Audio Process., vol. 9, no. 8, pp. 769–777, 2001, doi: 10.1109/89.966080.

[13] R. Saputra, "Matlab software for coded excited linear predictive coding", vol. 53, no. 9. 2019.

[14] O. K. Hamid, “Speech Sound Coding Using Linear Predictive Coding (LPC),” vol. 3, no. 1, pp. 13–17, 2017.

[15] Edwin K. P Chong, "An introduction to optimization", Fourth edition. 2012

[16] M. M. Hasan, A. M. Nasr, and S. Sultana, “An approach to voice conversion using feature statistical mapping,” Appl. Acoust., vol. 66, no. 5, pp. 513–532, 2005, doi: 10.1016/j.apacoust.2004.09.005.

[17] W.-H. Steeb, “Linear Prediction Analysis,” Math. Tools Signal Process. with C++ Java Simulations, pp. 149–171, 2005, doi: 10.1142/9789812775047_0010.

[18] W. J. Hess, “Pitch and Voicing Determination of Speech with an Extension Toward Music Signals,” Springer Handbooks, pp. 181–212, 2008, doi: 10.1007/978-3-540-49127-9_10.

[19] C. Richard Johnson, William A. Sethares, Andrew Klein, "Software Receiver Design,” vol.369, 2013.

[20] A. A. B. Ruiz, "Linear and Nonlinear Programming", vol. 3, 2015.

[21] S. T. Alexander, “The Method of Steepest Descent,” Adapt. Signal Process., pp. 46–67, 1986, doi: 10.1007/978-1-4612-4978-8_4.

[22] John G. Proakis and Masoud Salehi, "Fundamentals of communication systems", Second edition, 2005.

[23] Thomas M and Joy A. Thomas, " Elements of Information Theory", 2nd.2006.

[24] R. Rahim, “Bit Error Detection and Correction With Hamming Code Algorithm,” no. January, 2017, doi: 10.31227/osf.io/j3w5z.

[25] W. Rurik and A. Mazumdar, “Hamming codes as error-reducing codes,” 2016 IEEE Inf. Theory Work. ITW 2016, pp. 404–408, 2016, doi: 10.1109/ITW.2016.7606865.

[26] K. Mahender, T. A. Kumar, and K. S. Ramesh, “Analysis of multipath channel fading techniques in wireless communication systems,” AIP Conf. Proc., vol. 1952, no. March 2019, 2018, doi: 10.1063/1.5032012.

[28] R. E. Best, N. V. Kuznetsov, G. A. Leonov, M. V. Yuldashev, and R. V. Yuldashev, “Simulation of analog Costas loop circuits,” Int. J. Autom. Comput., vol. 11, no. 6, pp. 571–579, 2014, doi: 10.1007/s11633-014-0846-x.

[29] Hwei P. Hsu, "Schaum’s Outlines of Probability, Random Variables & Random Processes", 2nd Edition. 2010.

[30] P. Del Moral and S. Penev, “Simple random walks,” Stoch. Process., vol. 2002, no. April 2002, pp. 673–692, 2018, doi: 10.1201/9781315381619-25.

[31] A. Demir, A. Mehrotra, and J. Roychowdhury, “-Phase noise in oscillators: a unifying theory and numerical methods for characterization,” IEEE Trans. Circuits Syst. I Fundam. Theory Appl., vol. 47, no. 5, pp. 655–674, 2000, doi: 10.1109/81.847872.