FILTERED-X AFFINE PROJECTION ALGORITHM IN ACTIVE NOISE CONTROL USING VOLTERRA FILTER

Thesis submitted in partial fulfillment of the requirement for

the award of the degree of

MASTER OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

MANI KANT KUMAR

Roll No. 800861024

Under the guidance of

Dr. AMIT KUMAR KOHLI

Assistant Professor , ECED

THAPAR UNIVERSITY, Patiala

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT,

THAPAR UNIVERSITY, PATIALA-147004, PUNJAB, INDIA

JUNE -2010

CERTIFICATE

I, Mani Kant Kumar, hereby declare that this thesis entitled **"FILTERED-X AFFINE PROJECTION ALGORITHM IN ACTIVE NOISE CONTROL USING VOLTERRA FILTER"** is an authentic record of my own work carried out towards the partial fulfillment for the award of degree of Master of Engineering at Thapar University, Patiala under the guidance of Dr. Amit Kumar Kohli, Assistant Professor, ECED.

The matter presented in this report has not been submitted in any other University or Institute for the award of any degree.

Mani Kant Kumar

Roll No. 800861024

Dated: 02/7/2010

This is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Dr. Amit Kumar Kohli
Assistant Professor, ECED
Thapar University, Patiala
Dated: 2/7/2010

Dr. A.K. Chatterjee
Professor & head
ECED, Thapar University, Patiala
Dated: 2.7.10

Dr. R.K. Sharma
Dean of Academic Affairs
Thapar University, Patiala
Dated: 5/7/10

CONTENTS

ABSTRACT

The affine projection algorithms are a family of adaptive filters that can produce a good tradeoff between convergence speed and computational complexity. Their properties have been exploited for many years in the field of acoustic echo cancellation, and recently it was recognized that the filtered- AP algorithms can be very helpful also in the field of active noise control both for singlechannel and multichannel solutions.

Active noise controllers are based on the destructive interference in a given location of the noise produced by some primary sources with an interfering signal generated by some secondary sources driven by an adaptive controller. A commonly used strategy is based on the feedforward and feedback methods, where some reference signals measured in the proximity of the noise source are available. These signals are used together with the error signals captured in the proximity of the zone to be silenced in order to adapt the controller. Single-channel and multichannel schemes have been proposed in the literature according to the number of reference sensors, error sensors, and secondary sources used. These systems work on the principle of destructive interference between an original primary signal and secondary signal.
From the simulation results, Feedforward system has given better performance than feedback methods for LMS Algorithm, Filtered-X LMS Algorithm and filtered-AP Algorithm.

In thesis, we analyzed LMS Algorithm, Filtered-X LMS Algorithm and filtered-AP Algorithm for active noise control. In Filtered-X AP Algorithm, we used volterra filter. From the simulations results, AP Algorithm has given better performance and less complexity than LMS Algorithm and Filtered-XLMS algorithm. Filtered-X AP Algorithm used for high frequency and multichannel, decreased sensitivity to errors in the error-path model. Filtered-X AP Algorithm has taken less number of iterations for mean square error and mean attenuation for single and multichannel.

ACKNOWLEDGMENT

With the biggest contribution to this thesis I would like to thank my guide **Dr. Amit Kumar Kohli**, Assistant Professor, Electronics & communication Engineering Department, Thapar University, Patiala has given me his full support in guiding me with stimulating suggestions and encouragement to go ahead in all the time of thesis work.

I am also thankful to **Dr. A. K. Chatterjee**, Head, Electronics and Communication Engineering department, for the providing us with adequate infrastructure in carrying the work.

I am also thankful to **Ms. Alpana Agarwal**, P.G. Coordinator, Electronics and Communication Engineering department, for the motivation and inspiration that triggered me for the thesis work.

I am also thankful to entire faculty and staff member of Electronics and Communication Engineering Department for their unyielding encouragement.

At last but not the least my gratitude towards my parents, I would also like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Mani Kant Kumar

Roll No.800861024

Roll No.800861024

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANC	Active Noise Control
DSP	Digital Signal Processing
A/D	Analog to Digital
D/A	Digital to Analog
VS	Volterra Series
FM	Frequency Modulation
LT	Lapalace Transform
FT	Fourier Transform
LMS	Least Mean Square
RLS	Recursive Least Square
FIR	Finite Impulse Response
MSE	Mean Square Error
MMSE	Minimum Mean Square Error
NLMS	Normalised Least Mean Square
FXLMS	Filtered-x Least Mean Square Error
LS	Least Square
APA	Affine Projection Algorithm

CHAPTER 1
INTRODUCTION

The first active noise control (ANC) patent was filed by P. Lueg in 1936. Lueg's experimental work in noise control was largely unsuccessful due to problems he faced with instability[1]. ANC experiments were further explored in the 1950's. Olson and May implemented their so-called electric sound absorber as a "spot sound reducer". Many early attempts to suppress noise relied on the superposition of waves inside of a tube or duct and were realized by analog feedback or feed-forward control [2].

Advancements in the performance and stability of ANC systems did not surface until more recent years with the development of digital signal processing (DSP) hardware. Digital electronics allow for stable implementations of very complex control systems that are unrealizable with analog hardware. Digital signal processing has led to the development of new algorithms for control with increased capabilities.

Improvements have also been made in the physical implementation of ANC systems. Hardware arrangements that incorporate the effects of strong source coupling in addition to wave superposition provide a more effective and global control of noise that does not rely on the noise source being constrained by a duct. ANC systems with multiple control channels can improve the amount of source coupling and therefore achieve greater overall noise reduction. Assuming that the strength of a source is proportional to the size of its radiating surface, the size of secondary source needed to produce the desired output for a given primary source decreases as additional secondary sources are introduced. Decreased secondary source size allows for closer proximity between the primary and secondary sources. This provides stronger source coupling at a given frequency.

Every system has time-varying or certain parameters of the system. This parameter can be linear or nonlinear. We adapt the filtering technique to track the dynamics of the system or learn the unknown parameters. Nonlinear effect has been reduced by certain Algorithm.

1.1 Channel used in ANC

The signal picked up by the microphone will in realistic situations often also contain disturbance components for which no reference signal is available. Also for this case, multiple approaches to noise cancellation exist.

Single channel techniques

A microphone picks up a signal of interest, together with noise. Single microphone approaches to noise cancellation will try to estimate the spectral content of the noise (during periods where the signal of interest is absent), and assuming that the noise signal is stationary compensate for this spectrum in the spectrum of the microphone input signal whenever the signal of interest is present. The technique is commonly called 'spectral subtraction' [3]. Single channel approaches are known to perform poorly when the noise source is non–stationary, and when the spectral content of the noise source and the signal of interest are similar.

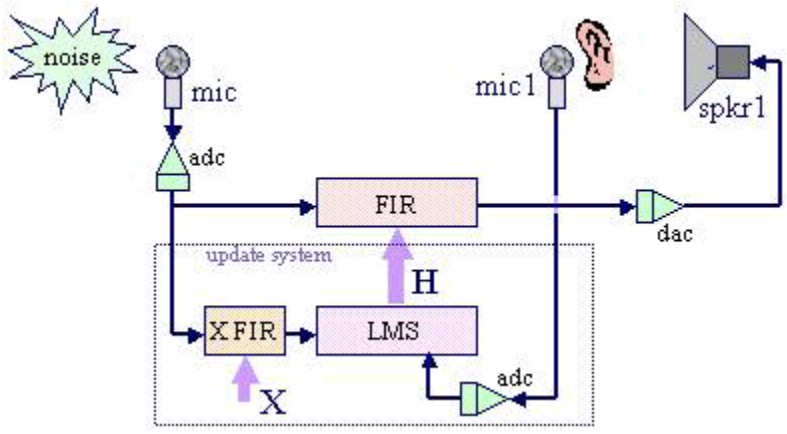

Figure 1.1 single channel ANC system [4]

A typical single-channel active noise cancellation system consists of:

- A microphone reference sensor to sample the disturbance to be cancelled
- An electronic control system to process the reference signal and generate the control signal
- A loudspeaker driven by the control signal to generate the cancelling disturbance
- An error microphone to provide the controller with information so that it can adjust itself to minimize the resulting sound field.

The active noise cancellation system just described is known as an "adaptive" system as it can adapt itself to changing characteristics of the noise to be cancelled and changing environmental conditions that affect the acoustic field.

Multi–channel techniques

In multi–channel acoustic noise cancellation, a microphone array is used instead of a single microphone to pick up the signal. Apart from the spectral information also the spatial information can be taken into account. Different techniques that exploit this spatial information exist.

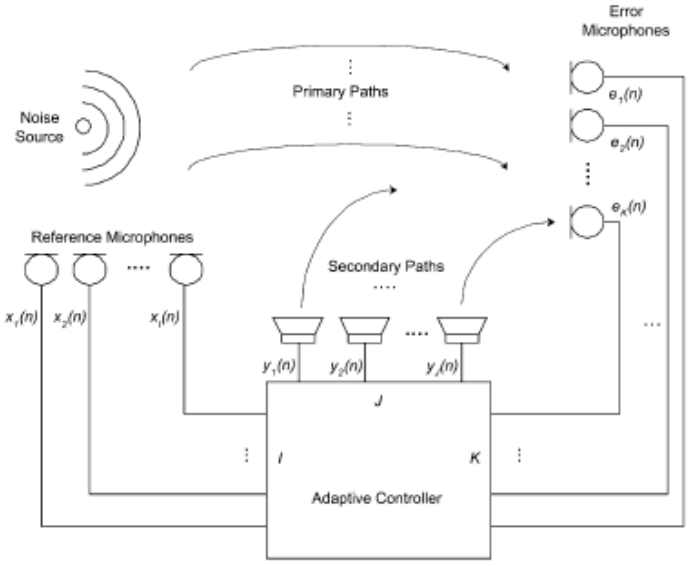

Figure 1.2 Multi-channel active noise control [14]

The Conventional approach for reducing noise or interference is based on linear signal processing techniques. Noise or interference in the past has been regarded as random process. Usually these random are assumed to be white Gaussian. Linear techniques are then applied to average the white noise out, predict the signal of interest or use filtering techniques in the frequency domain. Filtering techniques are most straightforward and employ band pass or low pass to capture the band limited signal of interest.

During the last three decade it has emerged that nonlinear deterministic system can generate time series which have many properties of classical noise. This is not really surprising as most process in nature is nonlinear. However, if the noise or interference is nonlinear and deterministic then it should be modeled as a nonlinear deterministic or even chaotic process rather than a stochastic noise but are not able to exploit the coherence of nonlinear deterministic noise .Nonlinear models on the other hand ,such as Volterra series, radial basis function, have been shown to be rather successful in modeling and predicting nonlinear deterministic time series.

Nonlinear models are usually far more complex and computational expansive than linear model. However, digital signal processing (DSP) are becoming much more faster and specialized, So that in the future the implementation of nonlinear model will not cause major problem. For these reason, it is beneficial to investigate the capabilities of nonlinear models to model or to predict nonlinear determination noise. It is very likely that nonlinear models will overcome a variety of problem which are encounters in linear filtering techniques.

Linear System

A system/filter is said to be linear if the output is a linear function of the input. The system property of linearity is based on two principles:

(1) Superposition

(2) Homogeneity

A system obeys the principle of superposition if the outputs from different inputs are additive: for example, if the output $y_1(t)$ corresponds to input $x_1(t)$ and the output $y_2(t)$ corresponds to input $x_2(t)$. Now if the system is subjected to an additive input

$x(t) = (x_1(t) + x_2(t))$

and the corresponding output is

$y(t) = (y_1(t) + y_2(t))$

then the system obeys the superposition principle. See Figure 1-1.

A system obeys the principle of homogeneity if the output corresponding to a scaled version of an input is also scaled by the same scaling factor: for example, if the output y (t) corresponds to input x(t). Now if we apply an input ax(t) and we get an output ay(t), then the system obeys the principle of homogeneity. See Figure 1-1

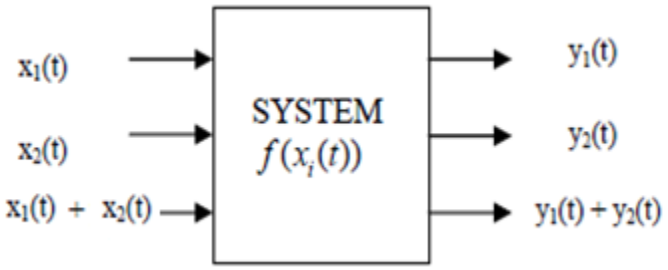

Figure 1.3 Superposition principle of linear system [4]

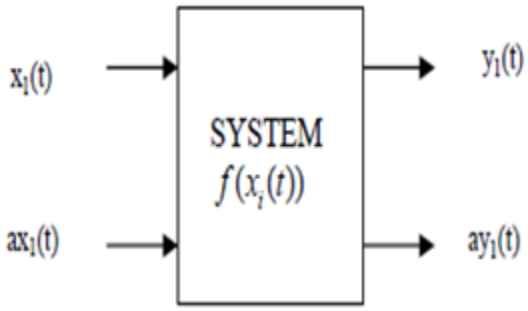

Figure 1.4 Homogeneity principle of linear system [4]

Nonlinear Systems

Nonlinear systems are systems whose outputs are a nonlinear function of their inputs. Many naturally-occurring systems are nonlinear. Nonlinear systems are still quite "mysterious". A polynomial nonlinear system represented by the infinite Volterra series. It can be shown to be time-invariant for every order except for the zeroth order.

Corresponding to the real-valued function of *n* variables h_n (t_1 , t_2 , …… t_n) *d*efined for t_i=-∞ to +∞ ,i=1,2,3…n and such that h_n (t_1 , t_2 ,…… ,t_n) = 0 if any t_i<0, (which implies causality), consider the input-output relation

$$\mathbf{y(t)} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \boldsymbol{h(\tau_1, \tau_2, \ldots \tau_n)\, x(t-\tau_1)(x-\tau_2) \ldots x(t-\tau_n) d\tau_1 d\tau_2 \ldots d\tau_n} \quad 1.1$$

This is the so-called degree-n homogeneous system [5] and is very similar to the convolution relationship defined earlier for linear, time invariant systems. However this system is not linear. An infinite sum of homogeneous terms of this form is called a polynomial Volterra series. A finite sum is called a truncated polynomial Volterra series.

First-order nonlinear system

$$y(n)=h_0 + \sum_{k_1=0}^{M-1} h_1\,(k_1)\, x(n-k_1) \quad 1.2$$

For a second-order nonlinear system

$$y(n) = h_0 + \sum_{k_i=0}^{M-1} h_1\,(k_1)\, x(n-k_1) + \sum_{k_i=0}^{M-1} \sum_{k_i=0}^{M-1} h_2\ (k_1, k_2)\, x(n-k_1)\ x(n-k_2) \quad 1.3$$

For example, See Figure 1-1 and Figure 1-2 below for Wiener and Hammerstein models of nonlinear systems which are interconnections of such subsystems. In the Weiner model, the first subsystem is an LTI system in cascade with a pure nonlinear polynomial (memory-less) subsystem.

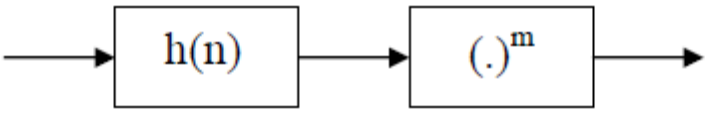

Figure 1.5 Weiner nonlinear model [6]

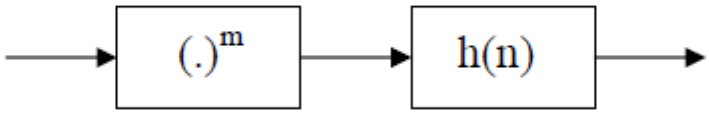

Figure 1.6. Hammerstein nonlinear model [6]

In the Hammerstein model, the first subsystem is a pure nonlinear polynomial (memory-less) in cascade with an LTI subsystem. In the general model (Figure 1-3), the first subsystem is an LTI subsystem, followed by a pure nonlinear polynomial (memory-less) then in cascade with another LTI subsystem.

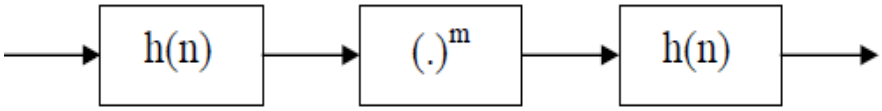

Figure 1.7. General nonlinear model [6]

Structure of ANC

Active noise control systems are ideally suited for use in the low frequency range, below approximately 500 Hz. Although higher frequency active control systems have been built, a number of technical difficulties, both structural/acoustic (for example, more complex vibration and radiated sound fields) and electronic (where higher sampling rates are required) limit their efficiency, so they are restricted to very special applications.

Also, at higher frequencies, passive systems generally become more cost effective. A "complete" active noise control system would usually consist of active control for low frequencies and passive control for higher frequencies.

An important property of many modern active sound control systems (particularly feed-forward systems) is that they are self-tuning (adaptive) so that they can adapt to small changes in the system being controlled. These changes are a result of such things as a changing acoustic environment and transducer wear. Changes only need to be small to cause a non-adaptive feed-forward control system to become ineffective. Non-adaptive controllers are generally confined to the feedback type in cases where slight changes in the environmental conditions will not be reflected in significant degradation in controller performance. One example of an effective non-adaptive feedback control application in active ear protection and headsets where analog feedback control systems have been used successfully for some time. An interesting aside is that an adaptive feed-forward controller is effectively a closed loop implementation of a non-adaptive feed-forward controller [4]. Two major types of active noise control system will be considered

- **Adaptive Feed-forward ANC**
- **Feedback ANC**

Both time domain and frequency domain versions of the feed-forward adaptive filtering form and the time domain version of the feedback form of controller will be considered

Adaptive Feed-forward Control

Referring to Figure 1.8 (feed-forward control), A reference sensor (usually a microphone) samples the incoming signal, which is filtered by the electronic controller to produce the output signal to drive the control source (loudspeaker in this case). The controller effectiveness is measured by the error sensor, which provides a signal for the control algorithm to use in adjusting the controller output so that the sound pressure at the error microphone is minimized. The signal processing time of the controller must be less than the time for the acoustic signal to propagate from the reference sensor to the control source for broadband noise control, but for tonal noise control, the maximum permitted

processing time can be much larger (as the signal is repetitive), but it is limited by the rate at which the amplitude and frequency of the tones change.

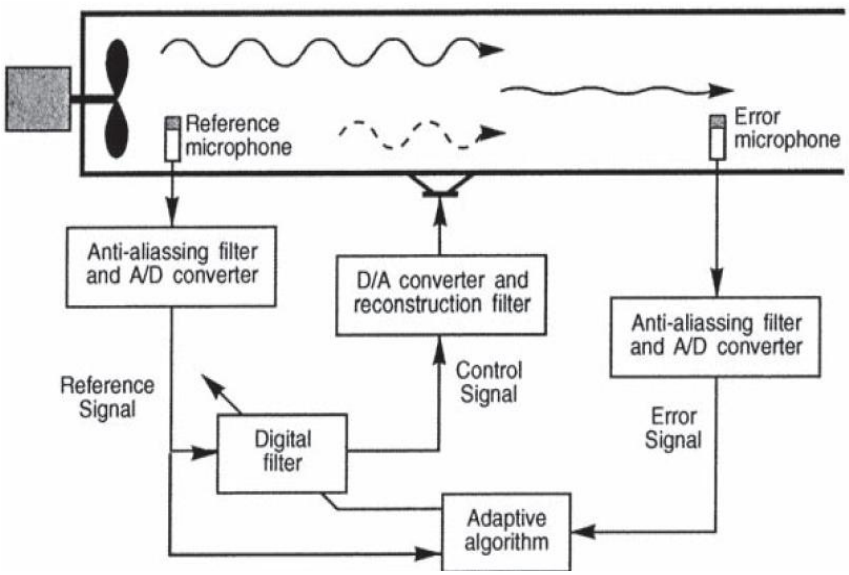

Figure 1.8 Feed-forward active noise control system with adaptive algorithm [4]

A feed-forward control system must be adaptive (self-tuning); continually tuning itself to provide the optimal result. Thus, in practice, a feed-forward controller must be implemented using digital electronics. To facilitate self-tuning, the signal from the error microphone is used together with an adaptive algorithm to continually update the characteristics of the control filter (shown as Figure 1.8). Analog to digital (A/D) converters, anti-aliasing filters, digital to analog (D/A) converters and reconstruction filters are needed for practical operation of the system. Reconstruction filters are just low pass filters that "smooth out" the edges of the digital signal, thus preventing the passage of the high frequency components that are not desired in the control signal.

Feedback ANC

Feedback control systems differ from feed-forward systems in the manner in which the control signal is derived. Whereas feed-forward systems rely on some predictive measure of the incoming disturbance to generate an appropriate "cancelling" disturbance,

feedback systems aim to attenuate the residual effects of the disturbance after it has passed. Feedback systems are thus better at reducing the transient response of systems, while feed-forward systems are better at reducing the steady state response.

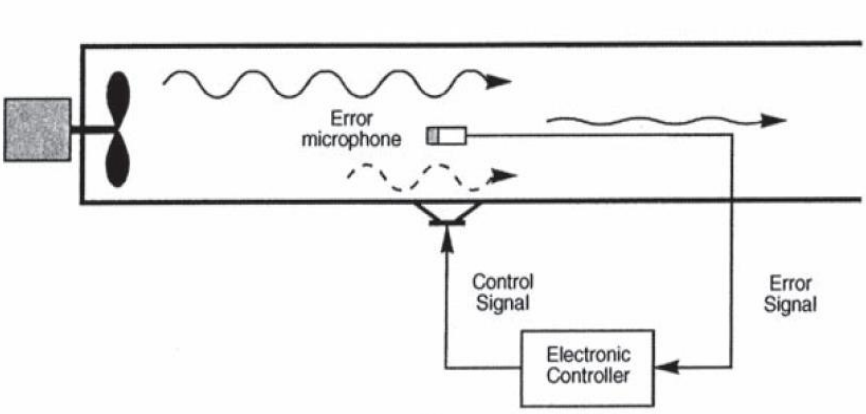

Figure 1.9 Feedback ANC [6]

Unlike feed-forward systems for which the physical system and controller can be optimised separately, feedback systems must be designed by considering the physical system and controller as a single coupled system. A feedback controller derives a control signal by processing an error signal, not by processing a reference signal as is done by a feed-forward controller. For a feedback controller, the error signal is processed to derive a suitable control signal for the control source such that the error signal is minimised, whereas in feed-forward systems, the error signal is used to help the controller to optimise its performance in terms of minimising the error signal, and it is the reference signal that is processed to generate the control signal.

Because of their inherent stability, feed-forward controllers are generally preferred over feedback controllers when a reference signal, which is correlated with the error signal, is available. This arrangement minimizes cost and maximizes performance. One disadvantage of feed-forward controllers that is not shared by feedback controllers is the often encountered problem of feedback of the control source output to the reference sensor via an acoustic path.

1.2 Algorithm used in ANC

Affine projection algorithms have been proposed for adaptive system applications as an efficient alternative to the slow convergence speed of least mean square (LMS) type algorithms, showing, then, good performance, robustness and stability. These algorithms update the weights based, instead of on the current input vector (as the NLMS algorithm), on N previous input vectors, being N the projection order. For that reason, the affine projection algorithm can be consider as a NLMS extension. Whereas much attention has been focused on the development of efficient versions of affine projection algorithms for echo cancellation applications, the similar adaptive problem presented by active noise control (ANC) systems has not been studied so deeply. This is focused on the necessity to reduce even more the computational complexity of affine projection algorithms for real-time ANC applications.

Furthermore, while in the ANC context the commonly used affine projection algorithm is based on the modified filtered-x structure, an efficient affine projection algorithm based on the (non-modified) conventional filtered-x structure, as well as efficient methods to reduce its computational burden, are discussed and analyzed throughout this thesis. Although the modified filtered-x scheme exhibits better convergence speed than the conventional filtered-x structure and allows recovery of all the signals needed in the affine projection algorithm for ANC, the conventional filtered-x scheme provides a significant computational saving, avoiding the additional filtering needed by the modified filtered-x structure. In this work, it is shown that the proposed efficient versions of affine projection algorithms based on the conventional filtered-x structure show good performance, comparable to the performance exhibited by the efficient approaches of modified filtered-x affine projection algorithms, and also achieve meaningful computational savings.

1.3 Overview of Thesis

This thesis is organized in four chapter.

Chapter 1 Summarize the introduction of ANC , type of channel , and algorithm used for ANC.

Chapter 2 We review about nonlinear model. In nonlinear model we study about order of wiener and volterra filter.

Chapter 3 We study about different type of Algorithm in ANC .

Chapter 4 Simulation results for Comparison between Feedforward and Feedback System, Analysis of Filtered-x AP Algorithm Using Volterra Filter, Multichannel Analysis for ANC .All simulations are done with using MATLAB.

Finally we conclude our work for ANC using different Algorithm.

CHAPTER 2

NONLINEAR MODELS

As shown in Introduction chapter, there are many approaches to nonlinear system identification. These approaches rely on a variety of different models for the nonlinear system and also depend on the type of nonlinear system.

In this report we have focused primarily on two polynomial models, the ones based on truncated Volterra series (VS) [5]. These models are the Volterra and Wiener models. In this chapter the polynomial modeling of nonlinear systems by these two models is discussed in detail. Relationships that exist between the two models are explained and the limitations of potentially applying each model to system identification applications are explained.

2.1 Volterra series

The Spanish mathematician Vito Volterra first introduced the notion of what is now known as a Volterra series in his "Theory of Functionals". The first major application of Volterra's work to nonlinear circuit analysis was done by the mathematician Norbert Wiener at M.I.T., who used them in a general way to analyze a number of problems including the spectrum of an FM system with a Gaussian noise input. Since then, Volterra series have found a great deal of use in calculating small, but nevertheless troublesome, distortion terms in transistor amplifiers and other systems.

Volterra Series Representation

A linear, causal system with memory can be described by the convolution representation

$$Y(t) = \int_{-\infty}^{\infty} h(\sigma)x(t-\sigma)d\sigma \qquad 2.1$$

Where x(t) is the input, y(t)the output, and h(t) the impulse response of the system. A Nonlinear system without memory can be described with a Taylor series

$$Y(t) = \sum_{n=1}^{\infty} a_n [x(t)]^n \quad 2.2$$

Where x(t) is the input , y(t)the output and a_n are the tayler series coefficient.

A Volterra series combines the above two representations to describe a nonlinear system with memory [6]

$$Y(t) = \sum_{n=1}^{\infty} \frac{1}{n!} \int_{-\infty}^{\infty} du_{1\ldots\ldots} \int_{-\infty}^{\infty} du_n \, g_n(u_1, \ldots\ldots u_n) \prod_{r=1}^{n} x(t-u_r) \quad 2.3$$

$$= \frac{1}{1!} \int_{-\infty}^{\infty} du_1 \, g_1(u_1) x(t-u_1) \quad 2.4$$

$$+ \frac{1}{2!} \int_{-\infty}^{\infty} du_1 \int_{-\infty}^{\infty} du_2 \, g_2(u_1, u_2) x(t-u_1) x(t-u_2) \quad 2.5$$

$$+ \frac{1}{3!} \int_{-\infty}^{\infty} du_1 \int_{-\infty}^{\infty} du_2 \int_{-\infty}^{\infty} du_3 \, g_3(u_1, u_2 u_3) x(t-u_1) x(t-u_2) x(t-u_3) \quad 2.6$$

$$+ \ldots\ldots\ldots$$

X(t) is the input, y(t) is the output, and the $g_n(u_1, \ldots\ldots\ldots u_n)$ are called the *Volterra kernels* of the system or simply the *kernel*. The u_i are time variables and are labeled u_i instead of t_i to distinguish them better from t . For n=1, $g_1(u_1)$ will be recognized as the familiar impulse response h(t) in equation (2.1)); thus, g_n for n > 1 are rather like "higher-order impulse responses." These serve to characterize the various orders of nonlinearity. The first few terms of (2.3) have been explicitly written out; (2.4) is the familiar convolution integral (2.3), and (2.5) and (2.6) may be thought of as two-fold and three-fold convolution. (2.3) is an infinite sum of n-fold convolution integrals.

Just as (2.3) is analogous to n-fold convolution, there exist n-fold analogies to Laplace and Fourier transforms. These are defined by

$$G_n(f_1, \ldots\ldots\ldots, f_n) = \int_{-\infty}^{\infty} du_1 \ldots\ldots \int_{-\infty}^{\infty} du_n \, g_n(u_1, \ldots\ldots\ldots, u_n) e^{-s_1 u_{1\ldots\ldots\ldots}} e^{-s_n u_n} \quad 2.7$$

and

$$G_n(f_1, \ldots\ldots\ldots, f_n) = \int_{-\infty}^{\infty} du_1 \ldots\ldots \int_{-\infty}^{\infty} du_n \, g_n(u_1, \ldots\ldots\ldots, u_n) e^{-jw_1 u_{1\ldots\ldots\ldots}} e^{-jw_n u_n} \quad 2.8$$

where $w_i = 2\pi f_i$. It is clear that $G_1(s_1)$ is the familiar linear transfer function, and thus G_n in general is referred to as the n^{th}*-order transfer function*. These frequency domain representations G_n are useful because often they are much simpler to calculate than the sometimes prohibitively-complex time-domain representations g_n and because radio problems are often approached in the frequency domain. From the definitions (3.7) and (3.8), it follows that if $g_n(u_1, \ldots\ldots\ldots u_n)$ is a symmetric function of the u_{iu} then $G_n(f_1, \ldots\ldots f_n)$ is a symmetric function of the f_i The traditional frequency-domain input-output representation now becomes

$$Y(f) = \frac{1}{1!}G_1(f)X(f) + \frac{1}{2!}\int_{-\infty}^{\infty} df_1\, G_2(f_1, f - f_2)X(f - f_1) + \frac{1}{3!}\int_{-\infty}^{\infty} df_1 \int_{-\infty}^{\infty} df_2\, G_3(f_1, f_2, f - f_2)X(f_1)X(f_2)X(f - f_1 - f_2) + \ldots\ldots\ldots\ldots\ldots\ldots \quad 2.9$$

The term "kernel" will sometimes be used to describe either g_n or G_n.

Determination of the Kernels

When an explicit equation relating the input x(t) to the system output y(t) is known, the techniques below may be used to determine the Volterra kernels g_n or G_n.

The Harmonic Input Method

This method is for determining the kernels in the frequency domain. When the input is

$$X(t) = \exp(jw_1 t) + \ldots\ldots\ldots\ldots + \exp(jw_n t) \quad 2.10$$

Where $w_i = 2\pi f_i$, i=1,2,……,n and the w_i are incommensurable ,then

$G_n(f_1, \ldots\ldots\ldots f_n)$ = {coefficient of $\exp\{j(w_1 + \ldots\ldots\ldots + w_n)t\}$ in 2.3

Thus, if we assume

$$x(t) = \exp(jw1t)$$

$$y(t) = \sum_{k=1}^{\infty} c_n \exp(jkw_1 t)$$

then $g_1(f_1)$ is equal to c_1.similarly.if we assume

$$X(t) = \exp(jw_1 t) + \exp(jw_2 t) \qquad 2.11$$

$$Y(t) = \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} c_{kl} \, exp(j(kw_1 + lw_2)t)$$

Then, $G_2(f_1,f_2) = c_{11}$. Also ,we have that $c_{00} = 0, c_{10} = G1(f1)$, and $c_{01} = G1(f2)$.

Similar relations hold true when x(t)) is composed of more than two terms.

A nice example of the harmonic input method is to apply it to a system where

$$Y(t) = x(t) + \in [\dot{x(t)}]^2 \ddot{x(t)} \qquad 2.12$$

This equation arises in some forms of the quasi-static approximation to filtered FM.

Let us find the Volterra kernels for this system. First, let x (t) = exp (jw_1t) and substitute this into (2.12).

$$y(t) = \exp(jw1t) + \in [(jw_1)^2 exp(2\dot{j}w1t)](jw_1)^2 \exp(jw_1 t)$$

$$= \exp(jw_1 t) + \in w_1{}^4 \exp(3jw_1 t) \qquad 2.13$$

Therefore from 2.11

$G_1(f_1)$ = { Coef. of exp (jw_1t) in 2.13 }

= 1

If X(t) = exp (jw_1t) + exp (jw_2t) in 2.12 , Now

$$Y(t) = \exp(jw_1 t) + \exp(jw_2 t) + \in [jw_1 \exp(jw_1 t) + jw_2 \exp(jw_2 t)]^2$$
$$\times [(jw_1)^2 \exp(jw_1 t) + (jw_2)^2 \exp(jw_2 t)]$$

$$= |[w_1]| + |[w_2]| + \epsilon \{w_1^2 |[2w_1]| + w_1 w_2 |[w_1 + w_2]| + w_2^2 |[2w_2]|\} \times \{w_1^2 |[w_1]| + w_2^2 |[w_2]|\}$$

$$= |[w_1]| + |[w_2]| + \epsilon \{w_1^4 |[3w_1]| + w_1^3 w_2 |[2w_1 + w_2]| + w_1^2 w_2^2 |[w_1 + 2w_2]| \{w_1^2 w_2^2 |[2w_1 + w_2]| + w_1 w_2^3 |[w_1 + 2w_2]| + w_2^4 |[3w_2]|\} \quad 2.14$$

Where |[x]| is an abbreviation for exp(jxt). There are no |[w1+w2]| terms in 2.14 which means

$G_2(f_1 + f_2) = 0$

Lastly, letting $x = |[w_1]| + |[w_2]| + |[w_3]|$ and substituting into 2.12 gives

$$Y(t) = |[w_1]| + |[w_2]| + |[w_3]| + \epsilon \{jw_1 |[w_1]| + j w_2 |[w_2]| + jw_3 |[w_3]|\}^2 \times \{-w_1^2 |[w_1]| - w_2^2 |[w_2]| - w_3^2 |[w_3]|\} \quad 2.15$$

We can see there will be three $x = |[w_1]| + |[w_2]| + |[w_3]|$ terms in (2.15) , Which makes

$G_2(f_1 + f_2 + f_3) = 2 \epsilon w_1 w_2 w_3 (w_1 + w_2 + w_3)$

For $n>3$, $G_n(f_1, \ldots\ldots f_n)$ will turn to be zero.

It is clear that the complexity of the harmonic input method increases rapidly as n Increases.

The Direct Expansion Method

This method is for determining the kernels g_n in the time domain. Here, the system equations are manipulated until they are brought into the form of a Volterra series (2.3), and the g_n are simply "read off" the representation. The circuit in Figure 2.1, the multiplicative connection of three linear subsystems, is amenable to analysis using the direct expansion method.

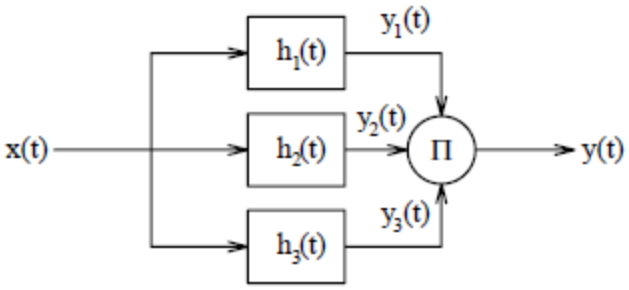

Figure 2.1 System for direct Expansion method [6]

For each subsystem, the defining equation can be written

$y_i(t) = \int_{-\infty}^{\infty} du_i \, h_i(u_i)x(t-u_i), i = 1,2,3$

and so the overall transfer function is

$y(t) = y_1(t)\, y_2(t)\, y_3(t)$

$= \int_{-\infty}^{\infty} du_1 \, h_1(u_1)x(t-u_1) \int_{-\infty}^{\infty} du_2 \, h_2(u_2)x(t-u_2) \int_{-\infty}^{\infty} du_3 \, h_3(u_3)x(t-u_3)$

$= \int_{-\infty}^{\infty} du_1 \int_{-\infty}^{\infty} du_2 \int_{-\infty}^{\infty} du_3 \; h_1(u_1)h_2(u_2)h_3(u_3) \prod_{r=1}^{3} x(t-u_r)$

Comparing (2.16) to 2.3), we get that $g_u(u_1,\ldots\ldots,u_n) = 0$ for all n except n= 3. At n = 3 We must multiply (2.16) by $\frac{1}{3!}$ in front and multiply by 3! inside the integral

$Y(t) = \frac{1}{3!} \int_{-\infty}^{\infty} du_1 \int_{-\infty}^{\infty} du_2 \int_{-\infty}^{\infty} du_3 \; 3!\, h_1(u_1)h_2(u_2)h_3(u_3) \prod_{r=1}^{3} x(t-u_r)$

$= \frac{1}{3!} \int_{-\infty}^{\infty} du_1 \int_{-\infty}^{\infty} du_2 \int_{-\infty}^{\infty} du_3 \; \gamma_3(u_1,u_2,u_3) \prod_{r=1}^{3} x(t-u_r)$ 2.16

Where

$\gamma_3(u_1,u_2,u_3) = 3!\, h_1(u_1)h_2(u_2)h_3(u_3)$

This kernel γ_3 is, in general, unsymmetrical since, For example $h_1(u_1)h_2(u_2)$ will not equal $h_1(u_2)h_2(u_1)$, and thus $\gamma_3(u_1,u_2,u_3) \neq \gamma_3(u_2,u_1,u_3)$ .To find a symmetric kernel g_3, we must symmetric γ_3 by permuting its arguments in 3! ways, adding the result and dividing by 3! . Thus, for the system of Figure 2.1

$g_3(u_1, u_1, u_2\ u_3) = \frac{1}{3!}\sum_{3!}\ \gamma_3(u_1, u_2, u_3)$

$$= \frac{1}{3!}\sum_{3!} 3!\, h_1(u_1)h_2(u_2)h_3(u_3)$$

$$= \sum_{3!} h_1(u_1)h_2(u_2)h_3(u_3)$$

Experience shows that the harmonic input method is generally easier when n is small; the direct expansion method seems to work best when the general formulae for high n are needed.

2.2 Order of volterra model

Zeroth and First-Order Volterra Model

The zeroth-order Volterra model is just a constant defined as

$Y_o[x(n)]=h_0$ 2.17

where x(n) is the input signal and h_0 is a constant.

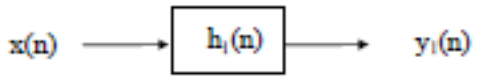

Figure 2.2 Isolated First order linear system block diagram [6]

The first-order Volterra system is basically the same as the linear system. In other words, the linear system is a subclass of the Volterra system. Consider a general isolated linear system as shown in figure 2.1[7].

Where the $h_1(n)$ represents the linear filter coefficients. The output $y_1(n)$ can be expressed by input x(n) as:

$y_1(n) = x(n)*h_1(n) = \sum_{i=0}^{\infty} h_1(k)\ x(n-k)$ 2.18

Where the * means linear convolution. If all the components in $h_1(n)$ can be represented by some linear combination of orthonormal basis $b_m(n)$. This means that the first-order Volterra kernel $h_1(k)$ in equation 2.18 can be represented by:

$$h_i(k) = \sum_{m=0}^{\infty} a_1(m) b_m(k) \qquad 2.19$$

Where $a_l(m)$ are some proper constants. Note that $\{ b_m(n), 0 \leq m \leq \infty \}$ is the set of orthonormal basis, which means

$$\langle bi(n), bm(n) \rangle = \delta(l - m) \qquad 2.20$$

where $\langle , \rangle$ denotes the inner product and $\delta(l - m)$ is the Dirac delta function. Substituting equation 2.19 in equation 2.18, we can define the first order Volterra functional as:

$$y_1(n) = \sum_{k=0}^{\infty} \sum_{m=0}^{\infty} a_1(m) b_m(k) x(n-k) = \sum_{m=0}^{\infty} a_1(m) [b_m(k) * x(n)] \qquad 2.21$$

Note that equation 3.5 is the first-order homogeneous functional, which means that $Y_1[cx(n)] = cY_1[x(n)]$, where c is a constant. Equation 2.21 can be expressed by the block diagram shown in figure 2-3:

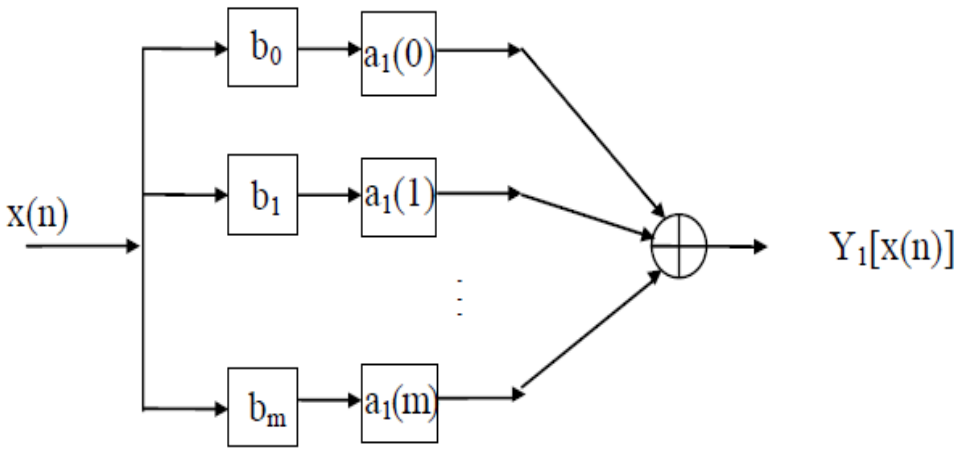

Figure.2.3 First Order Volterra model [6]

For the most general form of first-order Volterra system, we should include the DC term in equation 2.18, which can be expressed in terms of the Volterra functional $Y_0[x(n)]$ and $Y_1[x(n)]$ as:

$$Y(n) = h_0 + x(n)*h_1(n) = Y_0[x(n)] + y_1[x(n)] \quad 2.22$$

From equation 2.22, we conclude that a general first-order Volterra system with DC term is one for which the response to a linear combination of inputs is the same as the linear combination of the response of each individual input.

Second-Order Volterra Model

The linear combination concept described above can be extended to the second-order case, which is one for which the response to a second-order Volterra system is a linear combination of the individual input signals. Consider the isolated second-order extension version of figure 2.1 shown in figure 2.3:

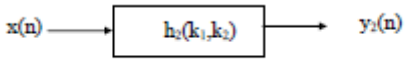

Figure 2.4 Isolated second order Volterra model block diagram [6]

$$y_i(n) = \sum_{i=0}^{\infty} h_2(k_1, k_2)\ x(n - k_1)x(n - k_2) \quad 2.23$$

where $h_2(k_1, k_2)$ is defined as the second-order Volterra kernel. As in the literature , for simplicity and without loss of generality, we assume the kernel to be symmetric, which implies that $h_2(k_1, k_2) = h_2(k_2, k_1)$. If $h_2(k_1, k_2)$ can be expressed by the linear combination of orthonormal basis set $b_k(n)$, then $h_2(k_1, k_2)$ can be written as

$$h_2(k_1,k_2) = \sum_{m1=0}^{\infty} \sum_{m2=0}^{\infty} a_2(m_1, m_2) b_{m1}(k_1) b_{m2}(k_2) \quad 2.24$$

Substituting equation 2.24 in equation 2.23, we can obtain the output as:

$$\begin{aligned} Y_2(n) &= \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} a_2(0,0) b_0(k_1) b_0(k_2) x(n-k_1)x(n-k_2) \\ &+ \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} a_2(1,1) b_0(k_1) b_1(k_2) x(n-k_1)x(n-k_2) \\ &+ \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} a_2(1,0) b_1(k_1) b_0(k_2) x(n-k_1)x(n-k_2) \\ &+ \dots \\ &= a_2(0,0)[b_0(n)*x(n)]^2 + a_2(1,1)[b_1(n)*b_2(n)]^2 + \dots\dots \end{aligned}$$

$$+ [a2(0,1)]+ a_2(1,0)][b_0(n)*x(n)][b_1(n)*x(n)]+\ldots\ldots$$

This equation, which is defined as the second-order Volterra functional

$$y_2(n) = Y_2[x(n)] \qquad 2.25a$$

In equation 2.23, we recognize that all the operations are two-dimensional convolutions, therefore equation 2.25a is a second-order homogeneous functional; i.e.

$$y_2[cx(n)] = c_2 Y_2[x(n)]$$

Consider a special case such that the linear orthonormal set contains two orthonormal bases $\{b_m(n), 0 \leq m \leq 1\}$. From equation 2.24, the second-order Volterra functional $Y_2(n)$ can be expressed as

$$y_2(n) = a_2(0,0)\,[b_0(n)*x(n)]^2 + a_2(1,1)\,[b_1(n)*x(n)]^2$$

$$+ [a_2(0,1)+ a_2(1,0)]\,[b_0(n)*x(n)]\;\;[b_1(n)*x(n)] \qquad 2.25b$$

The block diagram of equation 2.25b is shown in figure 2.4.

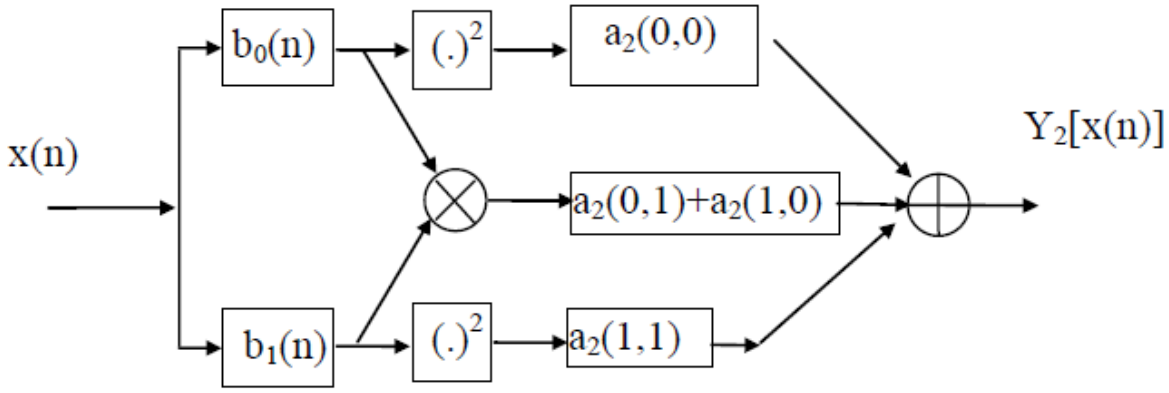

Figure 2.5 Second Order Volterra system with orthonormal basis set $\{b_0, b_1\}$ [6]

Based on the description above, the general second-order Volterra system can be represented in terms of $Y_0[x(n)]$, $Y_1[x(n)]$ and $Y_2[x(n)]$ which is

$$y(n) = h_0 + \sum_{ki=0}^{\infty} h_1(k)\, x(n-k) + \sum_{k1=0}^{\infty}\sum_{k2=0}^{\infty} h_2(k_1,k_2)\; x(n-k_1)x(n-k_2)$$

$$= Y_0[x(n)] + Y_1[x(n)] + Y_2[x(n)] \quad 2.26$$

To make the VS filter adaptive is straight forward, because adaptive least square algorithm from linear adaptive models can be used. Two well known adaptive algorithms are the LMS and the RLS algorithm. The LMS algorithm has low computational complexity and slow convergence in least square sense and the RLS algorithm has high computational complexity and fast convergence [8].

The LMS and RLS algorithm for a second order VS filter may be found in [7]. It is pointed out that the linear expansion in the input vector X_N will cause the eigen value spread to increase, even with a white input signal. Therefore, It is important to use algorithm whose convergence speed is on dependent or less dependent on the statistics of the input signal. It is well known that the LMS algorithm suffers in convergence speed when the Eigen value spread of the autocorrelation matrix is large. One approach to circulate this dilemma is to use the RLS algorithm at the expense of high computational complexity.

The VS filter will fail if it has to control discontinuities, for instance a saturation type of nonlinearity. This is because the VS expansion is taylor series with memory which only fits the data well when the function are smooth ,in the sense that they are at least once differentiable [9].

2.3 Wiener Model

Because of these two difficulties in Volterra model representation, with proper rearrangement, a Volterra system can be described in an alternative form which is called the nonlinear Wiener model [8].

To extend this result to the j^{th} Order, the general Volterra system is considered. The general Volterra system may have equivalent non-homogeneous G-functional representation as:

$y(n) = G_0[k_0; x(n)] + G_1[k_1; x(n)] + \ldots\ldots\ldots + G_j[k_j; x(n)]$

$$= \sum_{i=0}^{\infty} G_i [k; x(n)] \quad 2.27$$

From the previous section, the general relation between the non-homogeneous G-functional and the homogeneous Wiener kernel can be deduced as:

For m = even

$$G_m[k_m;x(n)] = g_m[k_m , k_{m-2(m)} , k_{m-(4m)},..., k_{0(m)} ; x(n)] \quad 2.28a$$

For m = odd

$$G_m[km;x(n)] = g_m[k_m,k_{m-2(m)},k_{m-(4m)},...,k_{1(m)};x(n)] \quad 2.28b$$

Note that, for m = even, all the homogeneous functional kernels with odd index numbers are equal to zero. For m = odd, all the homogeneous functional kernels with even index numbers are equal to zero. The more general expression of equation 2.28a and equation 2.28b can be expressed as a function of the Wiener kernel as:

$$G_m[k_m;x(n)] = \sum_{r=0}^{\left[\frac{m}{2}\right]} \sum_{i_1}^{\infty} \dots\dots \sum_{i_{m-2t}=0}^{\infty} k_{m-2r(m)} (i_1, i_2, \dots\dots, i_{m-2r}) \; x(n - i_1) \dots . x(n - i_{m-2r}) \quad 2.29a$$

where [m/2] means the largest integer less than or equal to m/2 and

$$K_{m-2r(m)}(i_1,i_2,\dots..i_{m-2r}) = \frac{(-1)^r \, m!(\sigma_x^2)^r}{(m-2r)!r!2^r} \sum_{j_1=0}^{\infty} \dots . \sum_{j_r=0}^{\infty} k_m \, (j_1, j_2, \dots\dots\dots, j_r, i_1, i_2, \dots\dots i_{m-2r}) \quad 2.29b$$

We note that all the homogeneous functional kernels $k_{m-2r(m)}$ can be expressed in terms of the leading kernel k_m. Furthermore, the Wiener G-functional series is an orthogonal series for white Gaussian inputs, that is:

$$E\{G_l[kl ; x(n)]G_m[k_m ; x(n)]\} = C_m \delta \, (l - m) \qquad l, m = 0, 1, 2, ..., p \quad 2.30$$

Where

$$C_m = m! \, (\sigma_x^2)^m \sum_{i_1}^{\infty} \sum_{i_2}^{\infty} \dots . \sum_{i_m}^{\infty} k_m \, (i_1, i_2, \dots . , i_2) k_m \, (i_1, i_2, \dots\dots\dots . , i_m,)$$

Equation 2.30 can be obtained by direct inspection from Table 2-1. The derivation procedure is exactly the same as in previous sections. For the general Volterra system, which is described by the Volterra series, we now have two equivalent representations: the Volterra model and the Wiener model. This means that:

$$y(n) = \sum_{i=0}^{\infty} G_i\,[k\,; x(n)] = \sum_{i=0}^{\infty} Y_i\,[x(n)] \qquad 2.31$$

To explore the relationship of Volterra kernels and Wiener kernels, we need to expand equation 2.31 and compare the kernels for both the Volterra model and the Wiener model. This relationship is summarized in Table 2.1.

Table 2.1. The kernel relation between Volterra and Wiener model of m^{th}-order for m = even Number.

	h_0	h_1	h_2	h_3	..... h_{m-2}	h_{m-1}	h_m
	$k_{0(m)}$						k_m
G_m			$k_{2(m)}$		$k_{m-2(m)}$		
G_{m-1}							
		$K_{1(m-2)}$		$k_{3(m-1)}$		k_{m-1}	
G_{m-2}							
	$K_{0(m-2)}$		$k_{2(m-2)}$		k_{m-2}		
…				.....			
G_2	$K_{0(2)}$		k_2				
G_1		k_1					
G_0	k_0						

From Table 3-1, we can show that the relationship between Volterra kernels and Wiener kernels is:

$h_m=k_m$ 2.32

$h_{m-1} =k_{m-1}$

$h_{m-2} =k_{m-2} +k_{m-2(m)}$

…….

$h_1 =k_1 +k_{1(3)} +\ldots\ldots\ldots\ldots+k_{1(m-3)} +k_{1(m-1)}$

$h0 = k_0 +k_{0(2)} + \ldots.. + k_{0(m-2)} + k_{0(m)}$

In equation 2.32, We obtain one unique Volterra kernel from Wiener kernels. In other words, the Volterra series is a subset of the Wiener G-functional representation. This means that any system that can be described by a Volterra series also can have the Wiener G-functional representation [8].

Zeroth- and First-Order Nonlinear Wiener Model

The main problem in the Volterra model is that even when the input signal is Gaussian white noise, the Volterra functionals Y_j [x(n)] lack orthogonality. To overcome this difficulty, instead of using for example Gram-Schmidt orthogonalization procedure, we can rearrange equation 2.22.as:

$Y(n) =g_0 [k_0: (n)] + g_1 [k_{01} :x(n)]$ 2.33

where g_0 [k_0; x(n)] and g_1[k_1, $k_{0(1)}$; x(n)] are called zero and first-order non-mogeneous functional respectively. Define g_0 [k_0: (n)] as

$g_0 [k_0: (n)] = k_0$ 2.34

where k_0 is a constant which is called the zeroth-order kernel .Define g_1 [k_{01} :x(n)] as

$g_1 [k_1,k_{01} : x(n)] = k_1 [x(n)] + k_{01} [x(n)]$ 2.35

$K_{0(1)}[x(n)]$ and $K_1[x(n)]$ are zeroth and First-Order homogeneous functional with $k_0(1)$ and k_1 kernels respectively. In the homogeneous K-functional, the subscript with parenthesis indicates that it is a homogeneous component of the non-homogeneous g-functional. To develop the orthogonal property, $g_1[k_1, k_{0(1)} ; x(n)$ is required to be orthogonal to the homogeneous Volterra functional $Y_0[x(n)]$.

That is:

$$E \{Y_0 [x(n)] g_1 [k_1,k_{01} : x(n)]\} = 0 \quad 2.36$$

To evaluate k_0, we need to substitute equation 2.35 in equation 2.36 which is:

$$E \{Y_0 [x(n)] g_1 [k_1,k_{01} : x(n)]\} = h_0 k_{01} + h_0 E \{ \sum_{j=0}^{\infty} k_1(j)\, x(n-j)\} \quad 2.37$$

To satisfy equation 2.37, we obtain $k_{01} =0$ for any constant h_0 .
We define the zero Order and first Order G- Fuctional as

$$G_0[k_0;x(n)] = k_0 \quad 2.38$$

$$G_1[k_1 ; x(n)] = g_1[k_1;x(n)] \quad 2.39$$

From equations 2.38 and 2.39, we can say that G-functional is the cano nical form of g-functional. Note that, like the g-functional, the G-functional is also a non-homogeneous functional. For general second-order Volterra systems, therefore, we can have two equivalent Volterra and Wiener model representations which are:

$$y(n) = Y_0[x(n)] + Y_1[x(n)] = G_0[k_0; x(n)] + G_1[k_1; x(n)] \quad 2.40$$

CHAPTER 3

Introduction to Adaptive Algorithms

Adaptive systems refer to systems that are able to effectively change their own parameters and thereby adapt" to the changes of the environment in which they operate. Hence, the basic nature of the problem is that some properties of the environment are changing in an unknown manner, and therefore must be tracked. Most of these problems have been formulated as linear Filtering problems. An adaptive filter is said to be linear if the estimated quantity of interest, computed adaptively, is as a linear combination of the observable data. In this thesis we concentrate mainly on linear adaptive algorithms due to their inherent simplicity and efficiency [8].

Applications

An adaptive filter is useful whenever the statistics of the input signals to the filter are unknown or time-varying. Since its parameters are changing according to the conditions of an ambient environment, little or no a-priori information is required about the input signals. Hence, it is also of interest to note that adaptive systems are capable of working with non-stationary signals, provided they can be considered stationary at least in a short interval. Numerous applications of adaptive filters have been proposed in literature. Some of the most noteworthy include:

1. Channel equalization and interference suppression
2. Teleconferencing and videoconferencing
3. Hands-free telephony
4. Voice control

3 .1 The basic Adaptive System

The adaptive system continually compares an output signal of an adaptive filter to a desired output signal. By observing the error between the output of the adaptive filter and the desired output, an adaptation algorithm can update its coefficients with the aim to

minimize an error function [8]. Fig. 3.1 shows the basic schematic diagram of an adaptive filter, where $x(n)$, $y(n)$, $d(n)$, and $e(n)$ are the input, output, desired output and error signals, respectively. For the convenience of presentation, the adaptive system considered in this chapter is assumed to be Finite Impulse Response (FIR) filter with N coefficients such that the output of the adaptive filter can be expressed as

$$Y(n)=W^{T}(n)x(n) \quad 3.1$$

Where

$$W(n)=[w_1(n),w_2(n)w_3(n),\ldots\ldots..w_N(n)]^{T} \quad 3.2$$

It is a vector containing the coefficient of the FIR filter

$$X(n)=[x(n),x(n-1),\ldots\ldots,x(n-N+1)]^{T} \quad 3.3$$

It is vector containing the input sample. The error signal, which has been mentioned in the previous paragraph can be expressed as

$$E(n)=d(n)-y(n) \quad 3.4$$

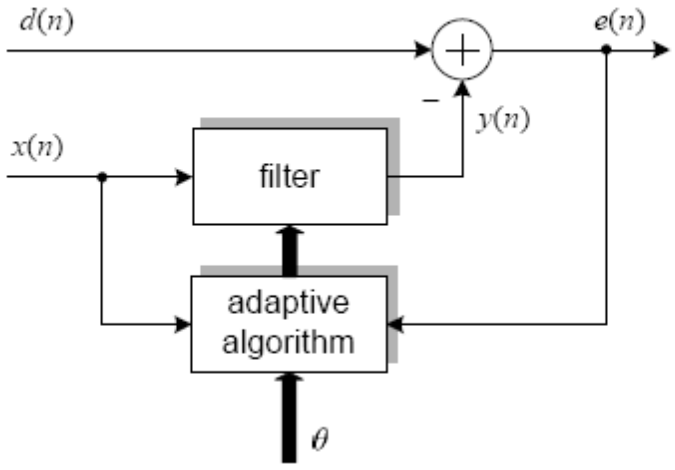

Figure 3.1 Schematic diagram of an adaptive system [8]

The structure of the adaptive system is shown in Fig. 3.2. This structure is also known as the tapped-delay line, since the input is tapped through the delay elements. The coefficients of the FIR filter are sometimes called the tap-weights. We will see in this

thesis that most gradient adaptive algorithms are more or less based on this simple structure.

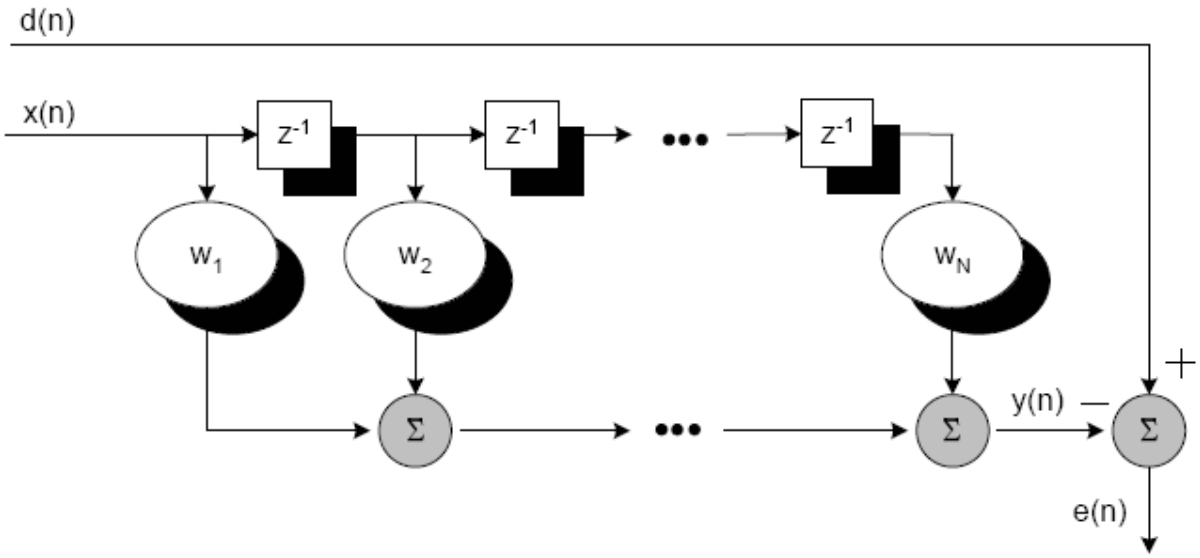

Figure 3.2 Block diagram of Adaptive system [8]

Objective function

An adaptive algorithm tries to minimize an objective function, usually denoted as Jw. The choice of this function has a profound impact on the properties of the whole algorithm [8]. It influences the rate at which the iterative adjustment process converges to its steady state. This state represents a point at which the objective function achieves its minimal value. Moreover, the objective function determines the robustness of the system and its stability. By carefully selecting the objective function we can condition its shape to have a continuous, convex character with a single “easy-to-track" minimum.

The mean-square error (MSE)

$$J_w(n) = E[\, e^2(n) \,] \qquad 3.5$$

The least-square error (LS)

$$J_w = \frac{1}{N}\sum_{i=1}^{N} e^2(i) \quad 3.6$$

The weighted least-square error (WLS)

$$J_w = \sum_{i=1}^{N} \lambda^{(N-i)} e^2(i) \quad 3.7$$

The objective function is sometimes referred to as the error performance surface, since the signal e (n) usually denotes an estimation error. The index w in the definition of the functions above signifies their dependence on the tap-weight vector w. Since adaptive methods are iterative, the values of the tap-weights are continuously updated based on the current value of the objective function. It is therefore one of the most useful quantities to describe the state of the filtering process [15].

3.2 Properties of Adaptive Algorithms

Convergence rate

It is the primary parameter at which we look when comparing different adaptive algorithms together. It determines the number of iterations required to get to the vicinity of a steady-state solution. It is desirable to achieve the *highest* rate possible. Since in many applications the system has to meet stringent deadlines, the convergence must be fast enough to meet the deadlines and not to affect the performance. This is also the case of speech signal processing, since speech is considered stationary only in short frames not longer than 30 ms.

Computational complexity

The computational complexity is the determining factor of algorithm's demand of resources. By resources we usually mean the time of processing and the memory for data storage. Adaptive algorithms are iterative methods that repeat their job all round. Thus, to estimate the computational power needed, we have to count the number of operations in a single iteration. For block-based algorithms this value determines the number of operations needed to process N consecutive samples whereas for sample-based

algorithm's it is only for a single sample. Therefore, when comparing different algorithms together, the strategy is to count the number of operations for a block of N samples.

Robustness

This may be viewed as a combined requirement of maximum immunity against internal errors, such as quantization and round-of

errors and insensitivity to external errors. Sometimes, however, it is better for the algorithm to be sensitive to certain changes of the environment, such as non-stationary of speech signals and noise processes. The trade-off between sufficient sensitivity and relative robustness is often a difficult task to solve.

Structural efficiency

In the era of powerful computational facilities certain questions arise when choosing the filtering structure. Finite-duration filters were, until lately, the preferred structures mainly due to their simplicity and efficiency. What we see now, however, in the field of computer technology is the application of parallel processors and multitasking at the lowest level. Texas Instruments, one of the worldwide leaders in DSP technology, started to integrate multithreading capabilities into DSP development tools a few years ago and thus provided support for the implementation of algorithms with a parallel structure.

3.3 Normalized Least Mean Square (NLMS) Algorithm

NLMS Algorithm conceptually stems from the Least Mean Squares (LMS) algorithm, which has been first developed by Widrow and Hoff [8]. The LMS algorithm is a stochastic gradient method that updates iteratively the coefficients of its own adaptive filter (the tap-weights) in a direction of the gradient vector.

The main drawback of the “pure” LMS Algorithm is that it is sensisitive to the scaling of its input(n).The NLMS is variant of the LMS Algorithm that solve this problem by normalizing with the power of input. This vector is calculated as a partial derivative of the Mean-Square Error (MSE) function with respect to the tap- weight. The LMS algorithm is described by the following equations

$$E(n = d(n)- w^{T} (n)) x(n) \quad 3.7$$

$$W(n+1) = w(n) + \mu_{LMS}X(n)e(n). \quad 3.8$$

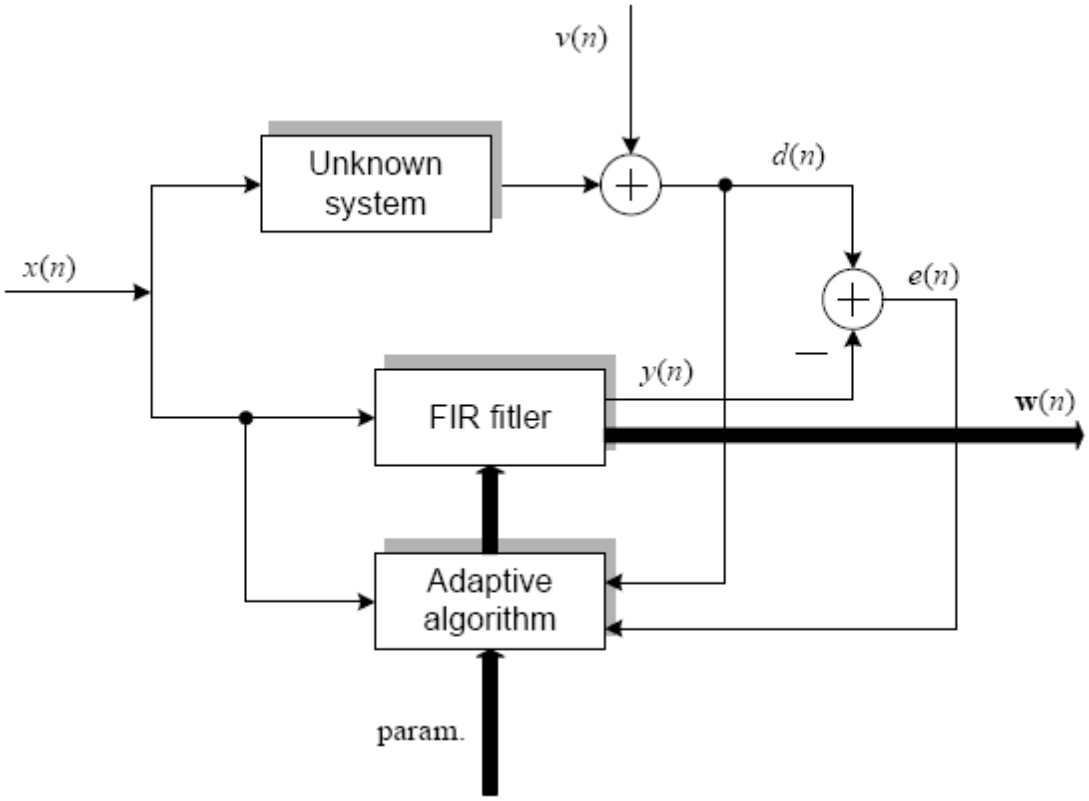

Figure 3.3 Block Diagram of NLMS Algorithm[8]

The term x(n) e(n), which is incorporated in the tap-weight correction term in the second line of (3.8) is an “instantaneous estimate" of the gradient of the MSE function Jw. Specifically, the estimate is given by δJw/δwn = -2e(n)x(n). Thus, the adaptation proceeds along the gradient of the MSE function until a minimum point of the function is reached. The iterative, weight-adjustment process of the NLMS algorithm is described by

$$w(n+1) = w(n) + (\mu/||X(n)||^{2}) x(n)e(n) \quad 3.9$$

where μ is the step size controlling the convergence rate, stability, and maladjustment. It must be chosen in the range $0 < \mu < 2$ to ensure stable operation.

To study the convergence rate of the NLMS algorithm one has to investigate the properties of the objective function Jw and its dependence on the tap-weight vector w.

From the theory of optimal filtering we know that the optimal solution to the problem of MSE estimation is the Wiener solution

$$W_0 = R^{-1}P \qquad 3.10$$

where R = E[x(n) x^T (n)] denotes an N-by-N correlation matrix of the inputs x(n) , x(n-1),……,x(n-N+1) and p = E[x(n) d(n)] is an N-by-1 cross-correlation vector between the inputs and the desired response d(n). Equation 3.10 is referred to as the Wiener-Hopf equations.

$$Jw = E[e^2(n)] \qquad 3.11$$

J(w) is a function of time and for each time instant n it produce a single value of MSE .However jw is also a function of tap-weight vector w

$$J(w) = j_{min} + (w-w_0)^T R(w-w_0) \qquad 3.12$$

*J*min is an irreducible term which corresponds to the minimum MSE function. No solution can achieve a value of *J* which is lower than *J*min.

The NLMS Algorithm update the coefficient of the adaptive filter the product filter by adding the product of the reference signal and the error between the response of unknown system and adaptive filter. The NLMS Algorithm has drawback that in case of correlated reference signal the convergence speed decrease.

3.4 Filtered-X LMS Algorithm

Adaptive filters are useful for situations where the characteristics of the input signals or the system parameters are unknown or time-varying. The design of the filter has to continually adapt to the changing environment. Figure 3.4 illustrates the requirements of an adaptive filter, namely: A filter structure, A filter performance measure and an adaptive update algorithm where the input signal is x(n), the desired response is d(n), the error signal is e(n) and the output signal is y(n) [15].

The output of the adaptive filter y(n) is compared with the desired response d(n). The error signal generated is used to adapt the filter parameters to make y(n) more closely approach d(n) or equivalently to make the error e(n) approach zero.

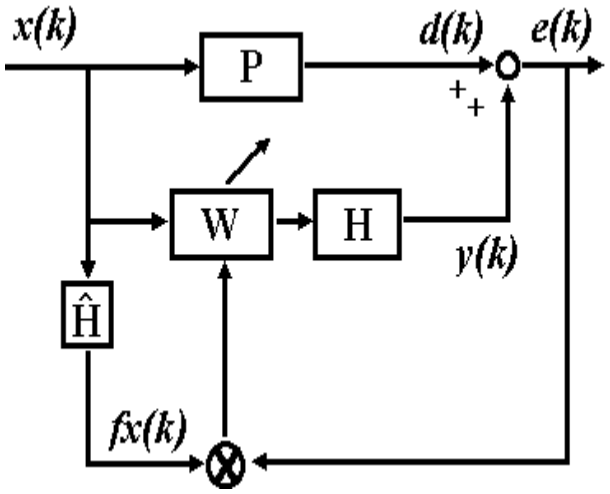

Figure 3.4 Block diagram of Filtered-x LMS algorithm [12]

The minimization of a function of the error signal is used for the design [10]. The most popular is the mean-square-error (MSE) function because it leads to a quadratic error surface

$$\xi(n) = E[e^2_f(n)] \rightarrow \text{Min.} \qquad 3.13$$

We choose FIR, Then we can write

$$Y(n) = WX^t(n) \qquad 3.14$$

$$e(n) = d(n) - y(n) \qquad 3.15$$

We substitute equation 3.14 and 3.15 in 3.13. The steepest gradient descent method of minimization requires that we update the weight vector in the negative direction of the steepest gradient, according to the following formula.

$$W(n+1) = W(n) - \mu \frac{\partial \xi}{\partial W} \qquad 3.16$$

This means we change the weight vector in the direction of the negative gradient at each location on the performance surface at each instance of time, n. The amount of change is proportional to the unit-less constant, μ, also sometimes called the step-size. Sometimes it is time-varying $\mu(n)$

$\widehat{\xi(n)}$ (n) = J(n) = $|e(n)|^2$ = e(n)$*$$e^*$ (n) 3.17

$\frac{\partial J}{\partial W}$=e(n) $\frac{\partial e*(n)}{\partial W} + e*(n)\frac{\partial e(n)}{\partial W}$ 3.18

$\frac{\partial e*(n)}{\partial W}$ = -X^H(n) 3.19a

$\frac{\partial e(n)}{\partial W} = -X(n)$ 3.19b

Put in Eq. 3.16, We get

W (n+1) = W(n) + μ e^*(n) $X(n)$ 3.20

X_f (n) are vector of dimension N

X_f (n) = x(n) $*$ h(n)

y_f (n) = y(n) $*$ h(n)

d_f(n) = d(n) $*$ h(n)

e_f(n) = d_f(n) - y_f(n) + e_0(n)

where e_0(n) is part of error e(n) that is not predictable by adaptive filter . Also It is assumed that e_0(n) is white process.

Weight error vector is equal to

$\varepsilon(n)$ = w (n) – w_0 at time n

The error-path with a transversal filter *h* of order M_h, the mean-square of the filtered error can be calculated

e (n) = X^T(n) $\varepsilon(n)$ + e_0(n) 3.21

$e_f(n) = \sum_{i=0}^{Mh-1} h_i \, (X^T(n-i)\varepsilon(n-i) + e_0(n-i))$ 3.22

The mean-sqaure filtered error can then be calculated

$$E[e_f^2(n)] = \sum_{i=0}^{Mh-1} \sum_{j=0}^{Mh-1} h_i \, h_j \, E.[X^T(n-i)\varepsilon(n-i)\varepsilon^T(n-j)X(n-i)]$$

$$+ E\left[\sum_{i=0}^{Mh-1} \sum_{j=0}^{Mh-1} h_i \, h_j \, e_0(n-i)e_0(n-j)\right] \quad 3.23$$

Using the independence theory and the fact that $e_0(n)$ is white.We finally obtain $J_{min.}$

Filtered-X LMS algorithm has been analysed under the presence of spherically invariant processes. This leads to much tighter but more practical conditions for the step-size in applications where no exact knowledge of the input is available. This will, of course, reduce the convergence rate of the algorithm further and degrade the performance of the whole system.

To correct the situation, a modification of the Filtered-X LMS algorithm has been introduced [12] and that reduces the order of the FXLMS algorithm to that of the LMS algorithm with a filtered input $X_f(n)$. Through this modification, the (normalized) bounds become independent of the input and the convergence rate is increased. The only negative aspect of the modified algorithm is its increased sensitivity to errors in the error-path model, making it only a feasible alternative when small errors in the error-path estimation can be guaranteed.

3.5 Filtered-X Affine Projection Algorithm

In terms of convergence rate and computational complexity, the affine projection algorithm represents an “intermediate stage" between the stochastic gradient algorithm (NLMS) and the least-squares method (LS). However, its inventors, Ozeki and Umeda in 1984 meant it to be a generalization to the well-known NLMS algorithm with a single goal to improve the convergence rate. The major problem that is solved by the APA is the slow convergence for input data, a drawback of all stochastic gradient methods. This is particularly useful in speech enhancement applications, since speech signals are known to be highly correlated [10].

The Affine Projection Algorithm is a generalization of the well-known Normalized least-mean-square (NLMS) adaptive filtering algorithm. Each tap weight vector of NLMS may be viewed as a one-dimensional affine projection [11]. In APA, the projection may be made in multiple dimensions. As the affine projection dimension increase, So does the convergence speed of tap weight vector and unfortunately, the algorithm‘s computational complexity. APA has been used in equalizer for data communication application, active noise control and neural network training algorithm [14].

To remove the problem of slow convergence of the NLMS algorithm, APA expands the number of input vectors to I, where I is called the rank.

The error signal now take different form

$$e(n) = d(n) - U^T(n)w(n) \qquad 3.24$$

where

$d(n) = [d(n)\ d(n-1)\ d(n-2)\ldots\ldots\ldots\ldots\ldots d(n-N+1)]^T$

$X(n) = [x(n)\ x(n-1)\ldots\ldots\ldots\ldots\ldots\ldots x(n-N+1)]^T$

$U(n) = [\ u(\ n\ )\ ,\ u(\ n-1)\ ,\ldots\ldots,\ u\ (\ n-\ L+1)\ ]^T$

These are desired response vector and input sample matrix , respectively. The dimension of the matrix X(n) is M×I is usually much smaller than the number of filter coefficient M , especially in the case of filter. The weight adjustment equation in APA.

$$W(n+1) = w(n) + \mu U(n)g(n) \qquad 3.25$$

Where g(n) is the pre-whitened error signal

$$g(n) = [U^T(n)U(n) + \delta I]^{-1}e(n)$$

APA require the solution to a system of equations involving the implicit inverse of the excitation signal’s covariance matrix. Although with APA, The dimension of the

covariance matrix is the dimension of the projection I, not the length of the joint process estimation, L .This is advantage because usually I is much smaller than L.

Numerical Solution of APA

The APA is an adaptive FIR filter. An adaptive filter attempt to predict the most recent output, [d(n) , d(n-1) ,……., d(n-N+1)] of an unknown system, W_{sys} , from the most recent system input, [u(n) , u(n-1) ,…….., u(n-L+N+1)] and the previous system estimate , w(n-1). U(n) is excitation signal and n is the time index[13].

$$U(n) = [\, u(n) , u(n-1) ,……, u (n- L+1) \,] \quad 3.26$$

L is length excitation or tap-delay line vector.

$$\alpha (n) = [\, u(n) , u(n-1) ,……, u(n-N+1) \,]^T \quad 3.27$$

N is length excitation matrix.

$$U(n) = [\, u(n) , u(n-1) ,……, u(n-L+1) \,]^T = \begin{bmatrix} \alpha(n)^T \\ \vdots \\ \vdots \\ \vdots \\ \alpha(n-L+1)^T \end{bmatrix} \quad 3.28$$

L by N excitation matrix

$$W(n) = [\, w_0(n), w_1(n),……., w_{L-1}(n) \,]^T \quad 3.29$$

L length adaptive coefficient vector where $w_i(n)$ is i^{th} adaptive tap weight or coefficient at time n

$$W_{sys} = [\, w_{0,sys,} w_{1sys},……,w_{L-1,sys}]^T \quad 3.30$$

L length system impulse response vector where $w_{i,sys}$ is i^{th} tap weight or coefficient.

Y(n) is measurement noise signal .

$$Y(n) = [y(n),y(n-1),….., y(n-N+1)]^T \quad 3.31$$

N is length noise signal vector.

TABLE 3.1

Quantities Used for the algorithm definition

Quantity	Dimension	Description
I signal	1	Number of primary source
J signal	1	Number of secondary source
K	1	Number of error sensors
L	1	Affine projection order
N	1	Number of element of vector $x_i(n)$ and $w_{j,I}(n)$
M = N.I.J	1	Number of coefficient of w(n)
$x_i(n)$	N×1	i-th primary source input signal vector
$x(n) = [x_1^T(n), \ldots\ldots x_I^T(n)]^T$	N.I×1	full primary source signal input signal vector
$w_{j,i}(n)$	N×1	coefficient vector of the filter that connect the input I to the output j of the ANC
$w_j(n) = [w_{j,1}^T(n), \ldots\ldots w_{j,I}^T(n)]^T$	N.I×1	aggregate of the coefficient vector related to the output j of ANC
$w(n) = [w_1^T(n), \ldots\ldots w_I^T(n)]^T$	M ×1	full coefficient vector of ANC

Quantity	Dimension	Description
$y_j(n) = w_j^T(n)\, x(n)$	1	j-th secondary source signal
$d_k(n)$	1	output of the k-th primary path
$d_k(n)= [d_k^T(n),\ldots\ldots d_k^T(n-L+1)]^T$	L×1	vector of L past output of k-th primary path
$d(n) = [d_1^T,\ldots\ldots\ldots d_K^T(n)]^T$	L.K ×1	full vector of the L past output of primary path
$s_{K,J(}n)$	1	impulse response of secondary path
$u_{K,J.i}\,(n)= s_{K,J(}n)\odot x_i(n)$	N×1	filtered-x vector obtained by filtering, sample by sample $x_i(n)$ with $s_{K,J(}n)$
$U_{k,j,i}(n)=[\, u_{k,j,i}(n),\, u_{k,j,i}(n-1),\ldots\ldots\ldots\, u_{k,j,i}(n-L+1)]$	N×L	matrix constituted by the last filtered-x vector $u_K(n)$
$U(n)=[U_1(n),\ldots\ldots\ldots U_K(n)]$	M×K.L	full matrix of filtered-x vector
$e_K(n)=d_K(n) +\sum_{j=1}^{J} u_{k,j}^T w_j\,(n)$	1	k-th error sense signal
$e_K(n)=[\, e_K(n),\ldots\ldots., e_K(n-l+1)]^T$	L×1	vector of L past error on k-th primary path
$e(n)=[\, e_1^T(n),\ldots\ldots\ldots., e_K^T(n)] =d(n) +U^T(n).w(n)$	L.K×1	full vector of error

$$d(n) = [d(n),d(n-1),\ldots\ldots..d(n-N+1)]^T = U(n).W_{sys} + Y\ (n) \quad 3.32$$

N is length desired or the system output vector. It's element consist of the echo plus any additional added in the echo path.

$$e(n)=d(n) – u(n)'* w(n) \quad 3.33$$

e(n) is the prior signal or priori error vector.

$$e(n) = [e(n),e(n-1),\ldots.., e(n-N+1)]^T \quad 3.34$$

is the N length a priori error vector.

μ is the relaxation or step size parameter , typically it is positive and slightly less than 1.

δ is the regularization parameter.

Using adaptive filter that minimize the length of its coefficient update vector,

$$R(n) =w(n) – w(n-1) \quad 3.35$$

Under the constraint that the new coefficient yield an N length a priori error vector,

Defined as

$$e_i(n)= d(n) –u(n) \square w(n-1) \quad 3.36$$

That is element by element a factor of 1- μ smaller than the length a priori error vector,

$$e(n)= d(n) –u(n) \square *w(n-1) \quad 3.37$$

By using Eq. 3.23 in 3.24

$$e_i(n)= \mu\ e(n) –u(n)'*r(n) \quad 3.38$$

using a Lagrange multiplier, we may express the cost function

$$C = \delta\ r(n)'\ r(n) + e_i(n)^2 \quad 3.39$$

Where δ is the Lagrange multiplier .We find the r(n) that minimize 3.27 by setting the derivative of C with respect to r(n) to zero and solving for r(n),yielding

$$C = \mu\, u(n)\, [u(n)'u(n)+ \delta I]^{-1}e(n). \qquad 3.40$$

Using Eq. 3.23

$$W(n) =w(n-1) + \mu\, u(n)[u(n)'u(n)+ \delta I]^{-1}e(n). \qquad 3.41$$

CHAPTER 4
Simulation and Results

4.1 Comparison between Feedforward and Feedback System

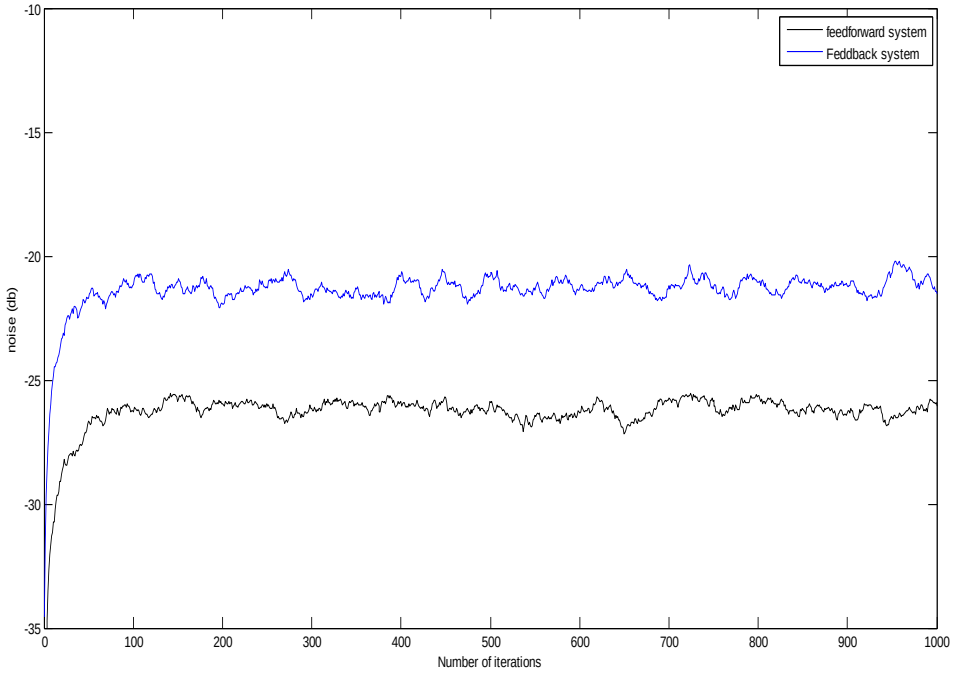

Figure 4.1 Noise level difference between Feedforward and Feedfack system using LMS algorithm

Feedforward system generate some cancelling disturbance while feedback system attenuate the residual effect of disturbance. According to Graph, feedforward system is suppressed more batter than feedback signal. But in LMS Algorithm, it is not stable

performance due phase shift occurred. This Algorithm is not support with high frequency. LMS Algorithm uses gradient descent method.

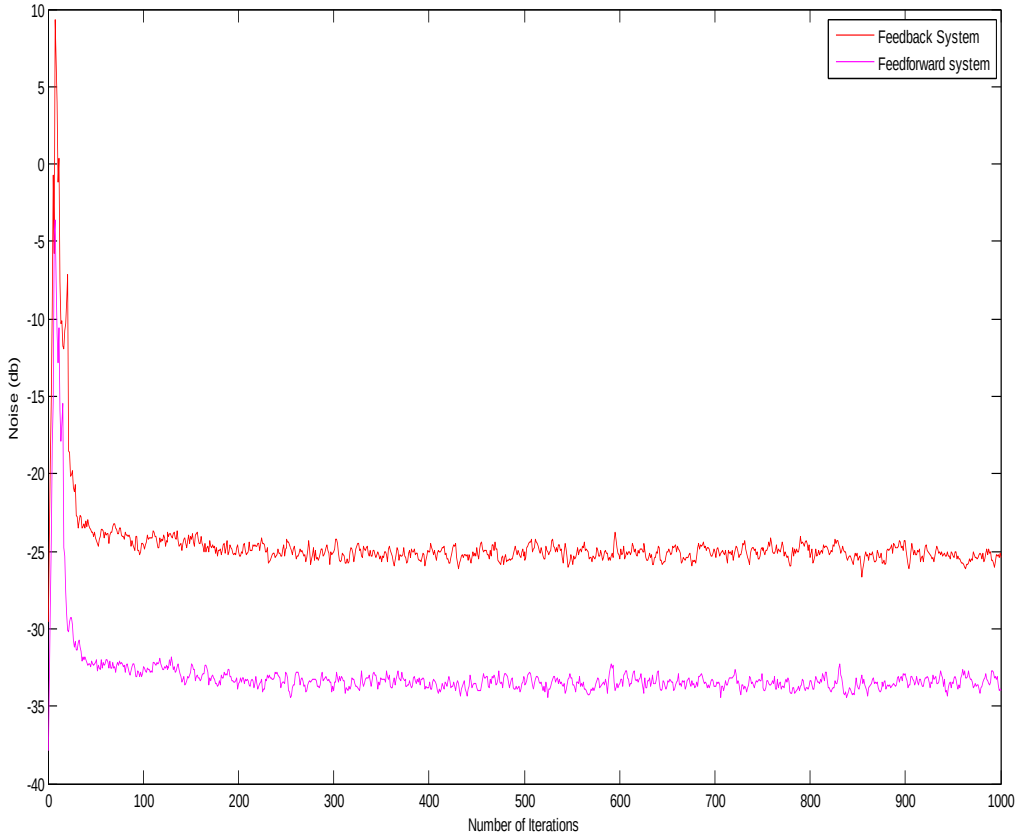

Figure 4.2 Noise level difference between Feedforward and Feedfack system using FX-LMS

Feeddorward system support stady state response while feedback system support transient response. We observe from the graph that feedforward system has given better performance than feedback system. This Algorithm has high convergence coefficient. Convergence coefficient depends upon amplifier gain, cancelation path transfer function amplitude, digital filter length. Convergence coefficient value should be less than 1. The disadvantage of this algorithm is that input signal is correlated with the reference signal.

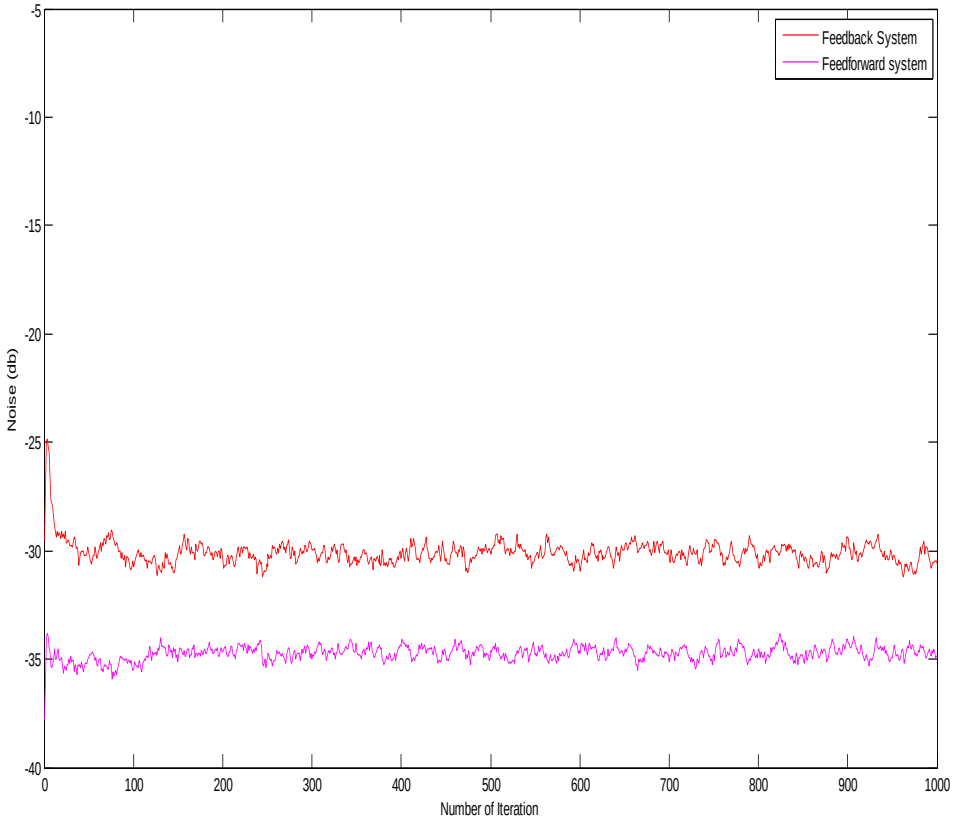

Figure 4.3 Noise level difference between Feedforward and Feedfack system using Filtered-X AP algorithm

This Algorithm has given better comparison than above two Algoritm. It has memory and support multichannel and high frequency.

4.2 Analysis of Filtered-X AP Algorithm Using Volterra Filter

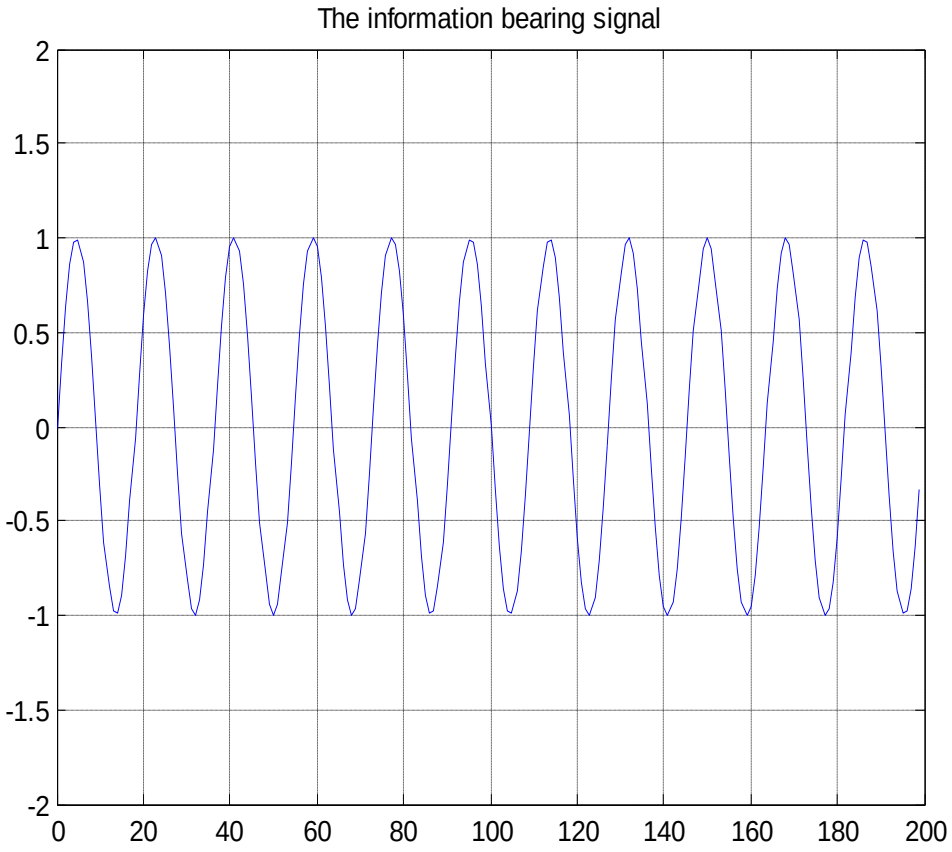

Figure 4.4 Information Signal

Using a loudspeaker as Primary source, Analog Information signal is given to primary source. Graph analyzed between amplitude and time.

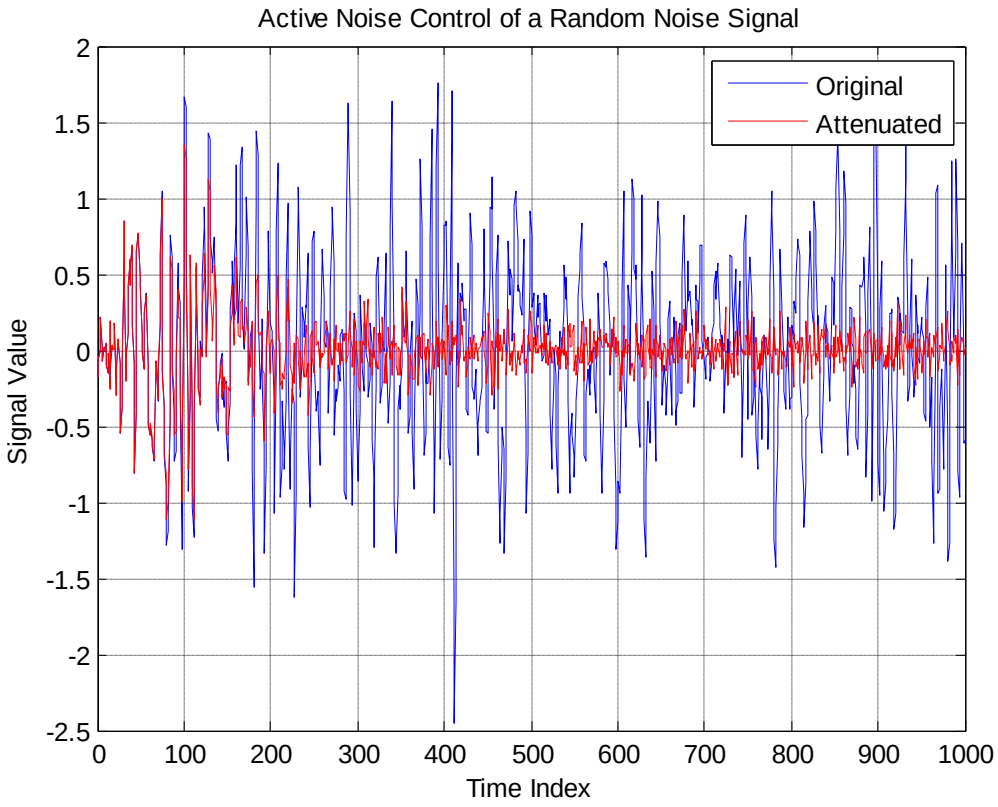

Figure 4.5 Original and noise signal for Filtered–X AP Algorithm

This graph analyzed between original signal and attenuated signal. Attenuated means signal mix with noise. This noise may internal or external. Noise signal may decrease or increase the amplitude of original signal.

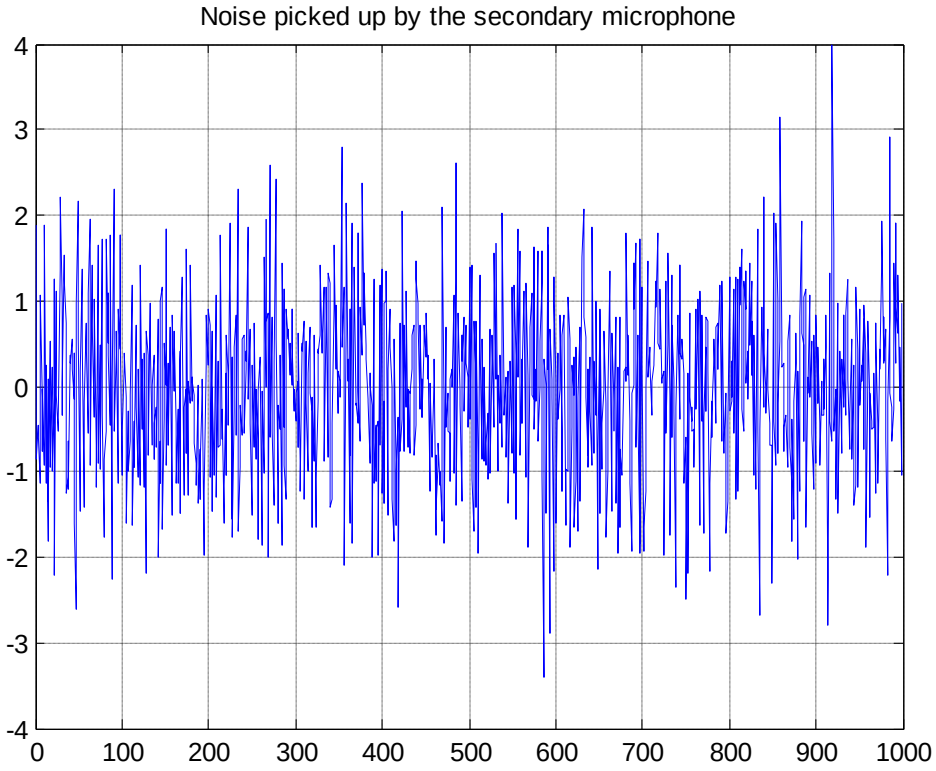

Figure 4.6 Noise detected by secondary path in Filtered-X Algorithm

This graph shows that noise signal is detected by secondary path. Secondary path used for concealing the noise presented in primary signal. Secondary path generate opposite phase of noise that presented in primary signal.

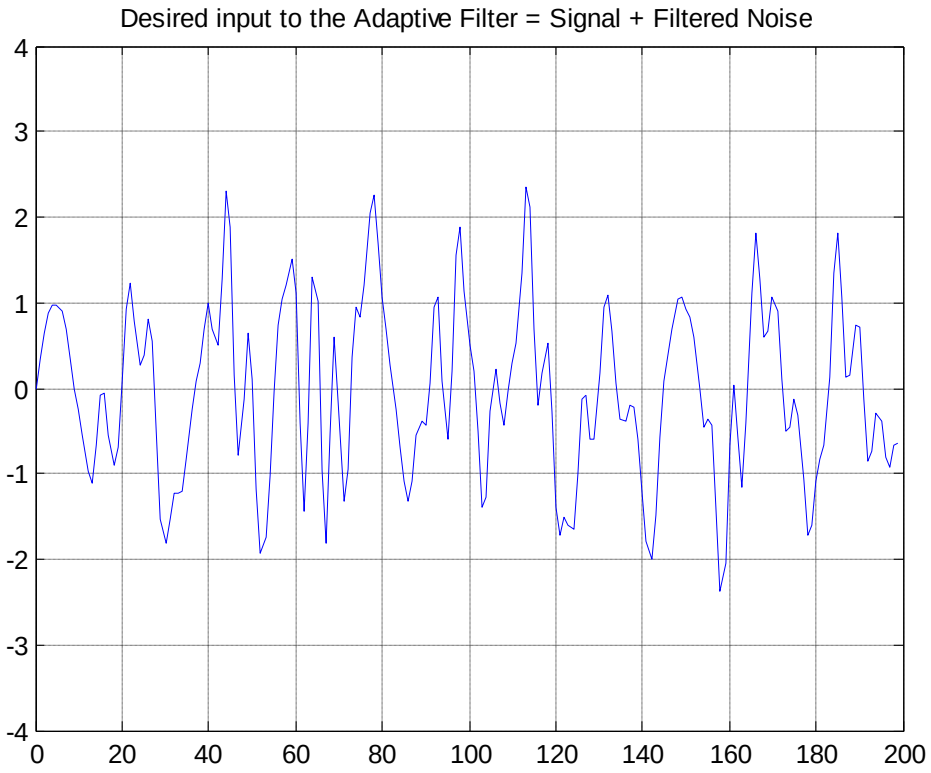

Figure 4.7 Input signal for Filtered-X Algorithm using Volterra Filter

This graph shows that input signal detected by volterra filter. This signal is summation of original and noise signal .This graph is plotted between amplitude and time.

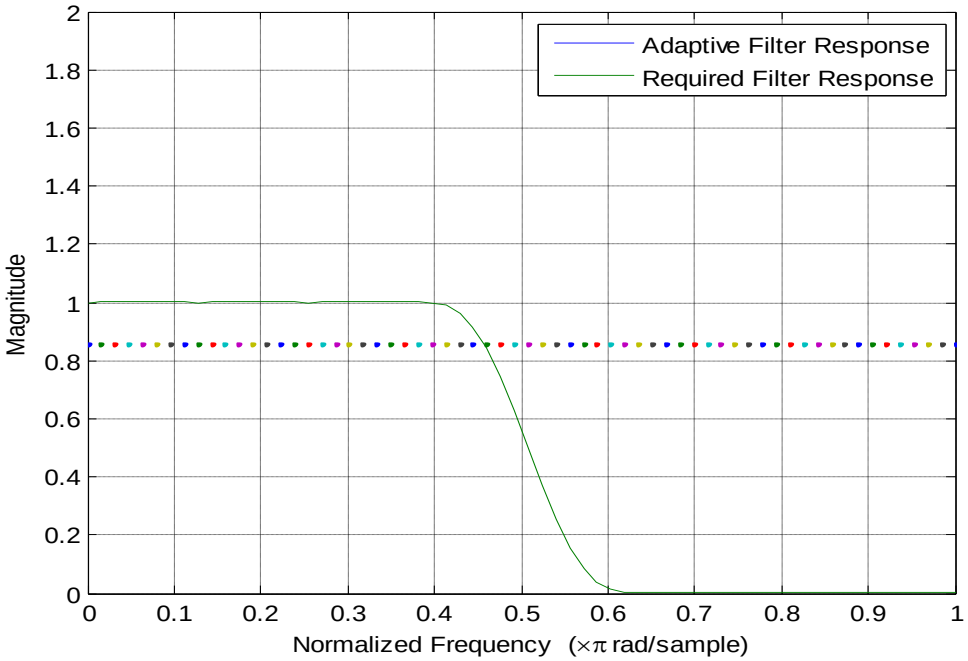

Figure 4.8 Volterra Filter response in Filtered-X AP Algorithm

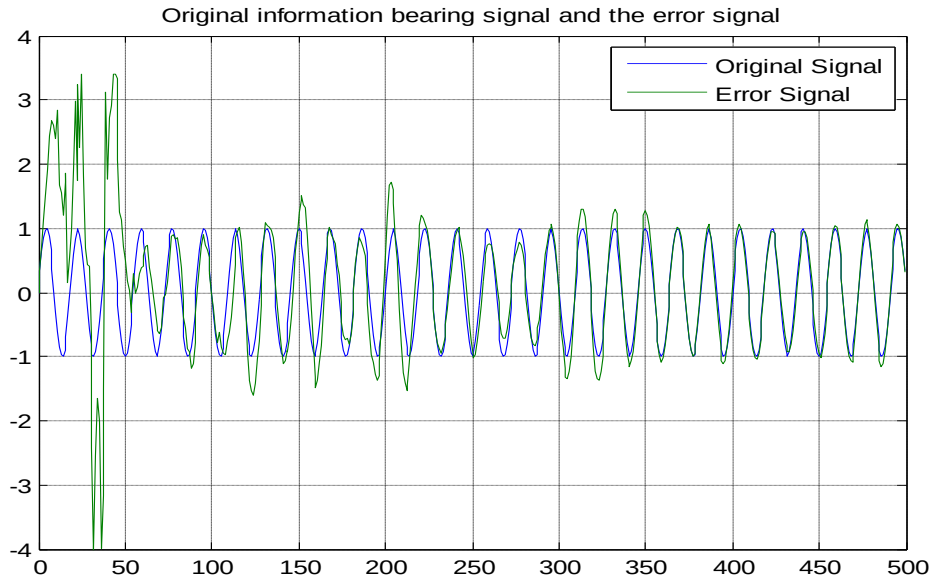

Figure 4.9 Output signal in Filtered-X AP Algorithm using Volterra filter

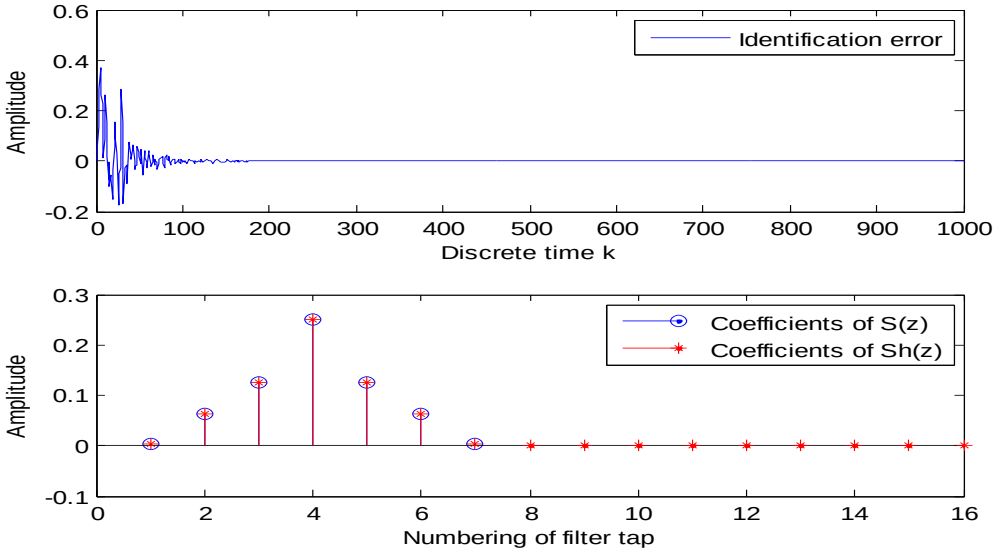

Figure 4.10 Amplitude of secondary path in Filtered-X AP Algorithm using Volterra Filter

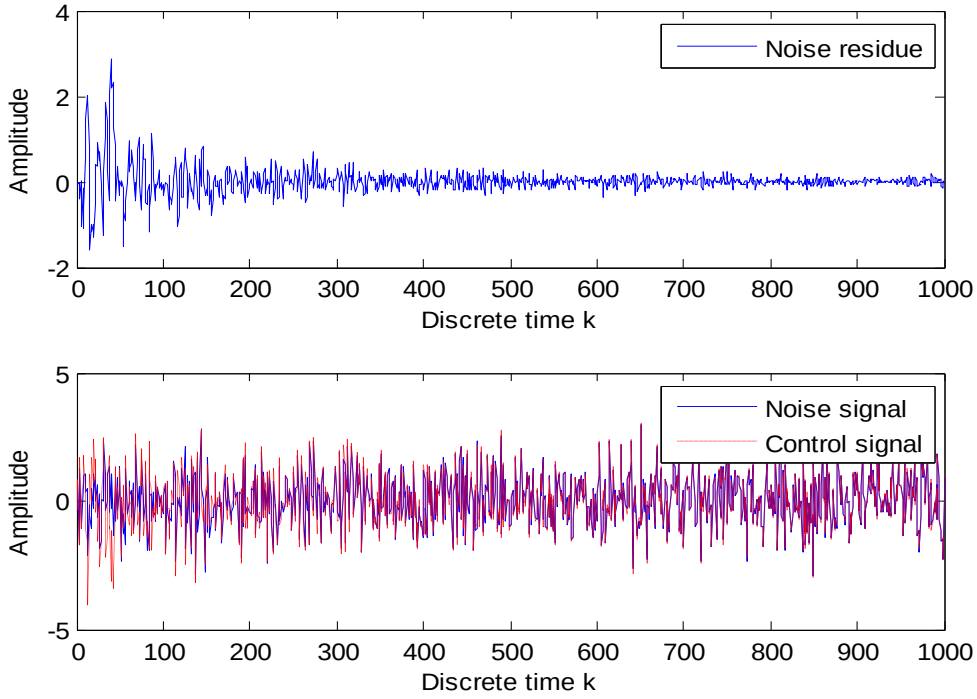

Figure 4.11 Noise Amplitude of secondary path in Filtered-X AP Algorithm using Volterra Filter

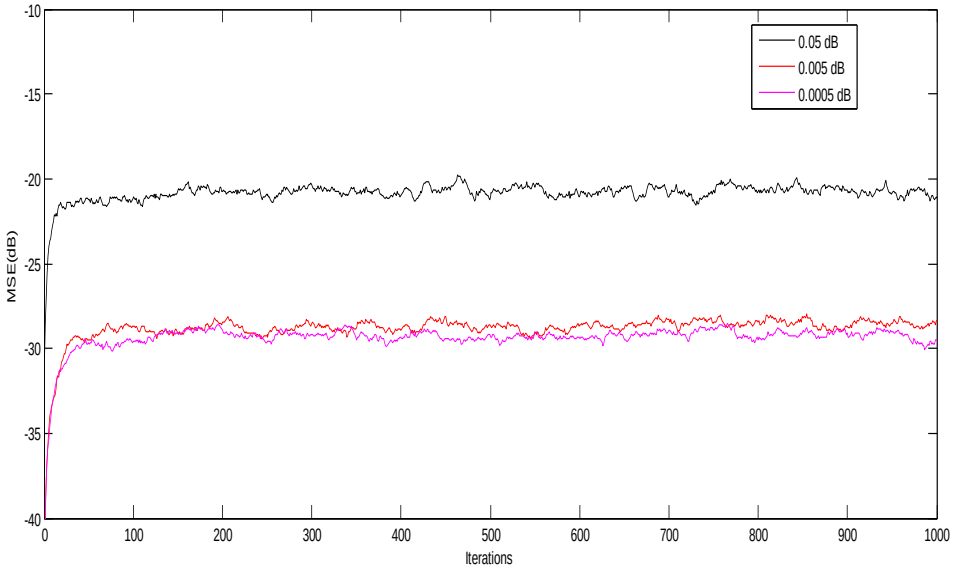

Figure 4.12 Mean square Error of Filtered-X AP Algorithm using Volterra Filter

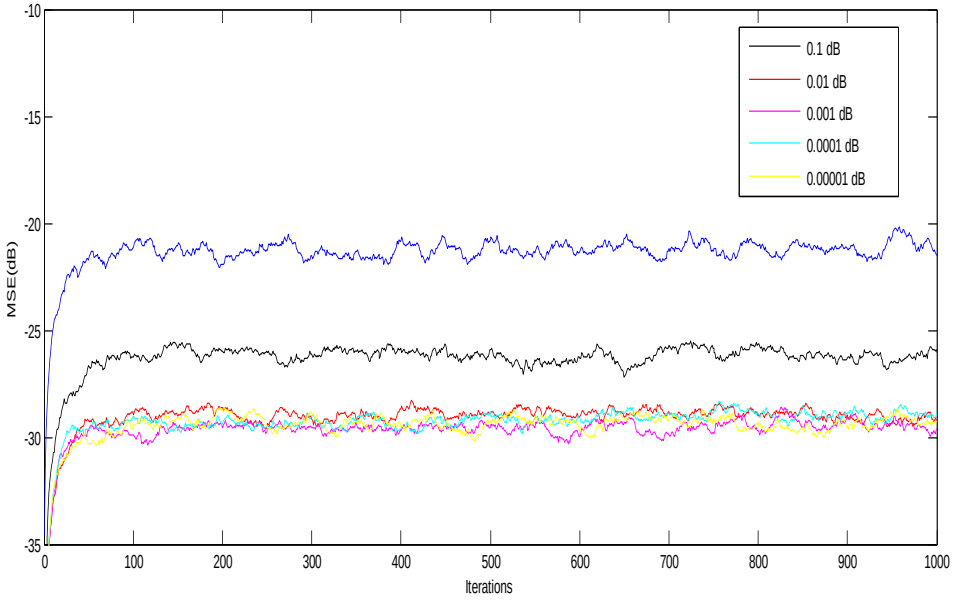

Figure 4.13 Mean square Error of Filtered-X AP Algorithm using Volterra Filter

4.3 Multichannel Analysis

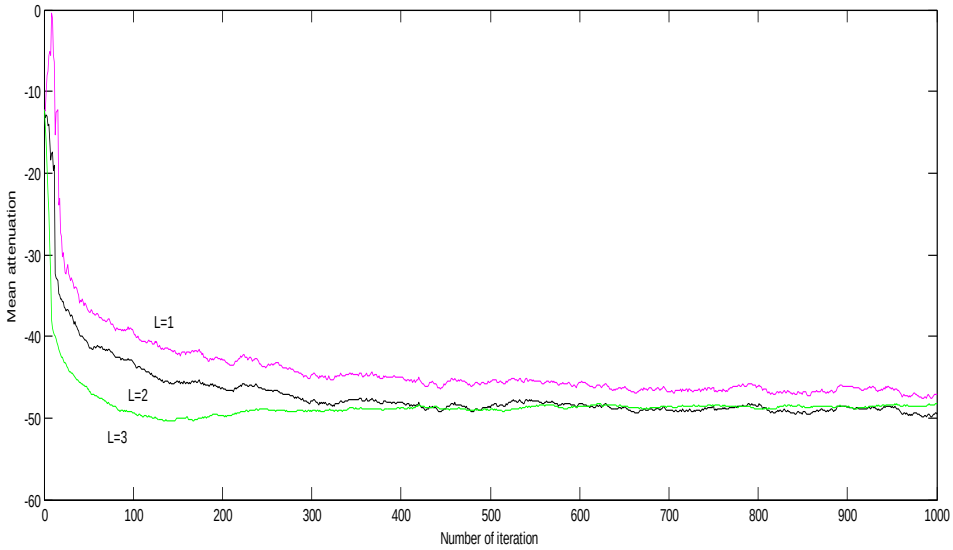

Figure 4.14 Mean Attenuation at error microphone in a multichannel active noise control noise control using AP algorithm using volterra filter for different order L

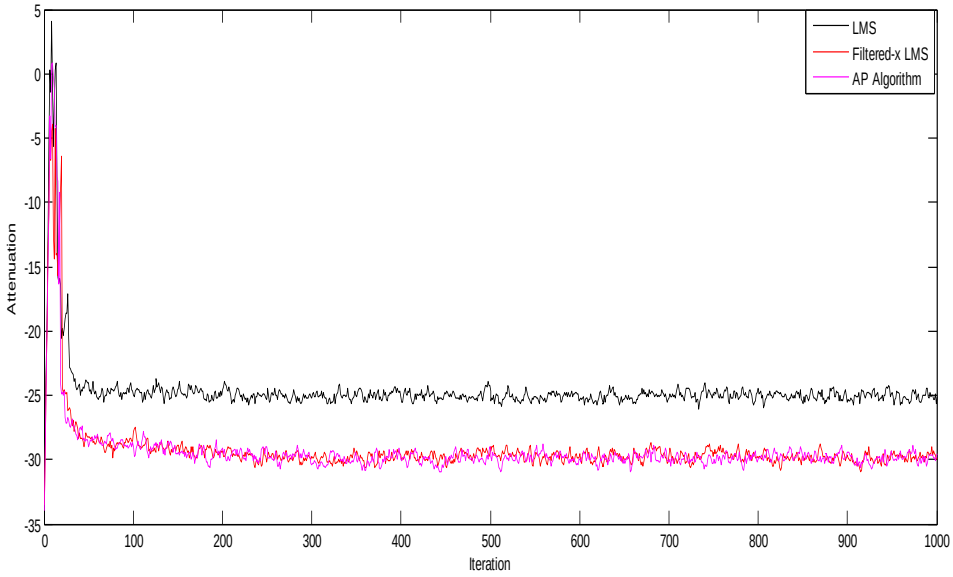

Figure 4.15 Mean Attenuation Using LMS, Filtered-X Algorithm, AP Algorithm using Volterra Filter

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

In this thesis, the convergence and tracking performance of LMS, Filtered-x LMS AP Algorithm are analyzed and compared for active noise control under time varying environment and varying order. From the simulation, following results are observed for the active noise control.

- LMS algorithm is unstable due to phase shift introduce.
- Computational complexity of AP algorithm is decreased.
- AP algorithm has faster convergence due to it has memory.
- Feed-forward control provides 3db or more of reduction in noise.
- Feedback controller provides over 10 db reduction in noise.
- For order L = 1, AP Algorithm is same as Filtered –x LMS algorithm.
- For order L = 2, input signal is speech signal for the faster convergence of filter.
- Second Order Volterra filter offer better performance than FIR filter.
- Number of iterations are decreased in AP algorithm.

.

Algorithms for active noise control deal with nonlinear effects. In such environments, nonlinear controllers based on Volterra filters implemented in the form of multi-channel. According to the multi-channel approach, derivations can be easily extended to a generic Volterra filter. AP technique offers better convergence and tracking capabilities than the classical LMS and NLMS algorithms with a limited increase of the computational complexity.

Another area of possible research work is the development of “fast” algorithms based on the Wiener model and higher order Volterra filter. Wiener model and Volterra filter can be explored further to reduce the computational complexity and number of iterations.

REFERENCES

[1] P. Lueg, "Process of Silencing Sound Oscillations" U.S. Patent No. 2,043, No. 416, 1936 .

[2] H. F. Olsen and E. G. May, "Electronic Sound Absorber," J. Acoust. Soc. Am **25**, Page No.1130-1136, 1953.

[3] P. Dreiseitel, E. Hansler, and H. Puder. Acoustic echo and noise control a long lasting challenge,1998.

[4] Colin H.hansen,"Active Noise Cancellation",Taylor & Francies e-library,2003.

[5] Wilson J. Rugh, *Nonlinear System Theory: The Volterra/ Wiener Approach,* Johns Hopkins University Press, 1981, ISBN 0–8018– 2549–0 (Online Web version published 2002).

[6] Tokunbo Ogunfunmi, Adaptive Nonlinear System Identification: Santa Clara University Santa Clara, CA USA, Springer Science + Business Media, LLC 2007.

[7] Cherry,Distortion Analysis of Weakly Nonlinear Filters Using Volterra Series,1994.

[8] S. Haykin, Adaptive Filter Theory ,4th ed. New Jersey: Prentice Hall 2002.

[9] "Noise cancellation using a modified form of the filtered-XLMS algorithm," in Proc. Eusipco Signal Processing V, Briissel, 1992.

[10] M. Bouchard, "Multichannel affine and fast affine projection algorithms for active noise control and acoustic equalization systems," IEEE Transaction on *speech and audio processin*g ,vol.11,No.1, January 2003.

[11] Felix Albu, Martin Bouchard, and Yuriy Zakharov, Pseudo-Affine projection Algorithms for Multichannel Active Noise Control, IEEE *Trans. Speech Audio Processing*, vol. 15, No 03, March 2007.

[12] Elias Bjamason, Analysis of the Filtere -X LMS Algorithm, *IEEE Trans. Speech Audio Processing*, vol.03, N0. 6, Nov. 1995.

[13] Alberto Carini, and Giovanni L. Sicuranza, Transient and Steady-State Analysis of Filtered- X Affine Projection Algorithms, IEEE Trans. *Speech Audio Processing*, vol. 54, No.02, Feb 2006.

[14] Giovanni L. Sicuranza, "Filtered-X Affine Projection Algorithm for Multichannel Active Noise Control Using Second-Order Volterra Filters", IEEE *Speech signal processing* Vol. 11, No. 11, November 2004.

[15] S. C. Douglas, "Fast implementations of the filtered-X LMS and LMS algorithms for multichannel active noise control," IEEE Trans. *Speech Audio Processing*, vol. 7, pp. 454– 465, July 1999.