ADAPTIVE NOISE CANCELLER USING VARIABLE STEP-SIZE ADAPTIVE IIR NOTCH FILTER

MANOLOM CHANTHAVONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN INFORMATION ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2009

KMITL–2009-EN-M-230-159

หัวข้อวิทยานิพนธ์	ตัวขจัดสัญญาณรบกวนปรับตัวได้โดยใช้ตัวกรองอะแดปทีฟไอไออาร์แบบนอตช์แบบปรับเสต็ปไซส์
นักศึกษา	นางสาว มโนรม จันทวงค์
รหัสประจำตัว	50061051
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมสารสนเทศ
พ.ศ.	2552
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ. ดร. ชวลิต เบญจางคประเสริฐ

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้นำเสนอ ตัวขจัดสัญญาณรบกวนปรับตัวได้โดยใช้ตัวกรองอะแดปทีฟไอไออาร์แบบนอตช์แบบปรับเสต็ปไซส์ อัลกอริทึมที่นำเสนอนี้มีสามแบบคือ อัลกอริทึมแรก (VSS 1) จะใช้อัตสหสัมพันธ์ของสัญญาณเอาต์พุต ค่าอัตสหสัมพันธ์ของค่าความคลาดเคลื่อน และค่า MSE ในการปรับตัวของอัลกอริทึม อัลกอริทึมแบบที่สอง (VSS 2) โดยใช้อัตสหสัมพันธ์ของสัญญาณเอาต์พุตใหม่ ในการปรับเปลี่ยนค่าเสต็ปไซส์ของอัลกอริทึมของทั้งสองวิธีที่กล่าวมานั้นกำหนดแบนด์วิดท์ของตัวกรองแบบคงที่และอัลกอริทึมแบบสุดท้าย (VNB) ใช้วิธีการปรับเปลี่ยนแบนด์วิดท์แบบแปรค่าตามเวลา ทำการทดสอบสมรรถนะของอัลกอริทึมโดยการประมาณค่าสัญญาณซายน์ที่ปนมากับสัญญาณรบกวนแบบเกาส์เซียน โดยการจำลองการทำงานด้วยโปรแกรมคอมพิวเตอร์พบว่าอัลกอริทึมที่นำเสนอสามารถประมาณค่าสัญญาณซายน์ได้ดี มีการลู่เข้าสู่ค่าสัมประสิทธิ์ที่แท้จริงของตัวกรองได้อย่างรวดเร็ว และมีความผิดพลาดต่ำ

Thesis Title	Adaptive Noise Canceller Using Variable Step-Size Adaptive IIR Notch Filter
Student	Ms. Manolom Chanthavong
Student ID.	50061051
Degree	Master of Engineering
Program	Information Engineering
Year	2009
Thesis Adviser	Assoc. Prof. Dr. Chawalit Benjangkaprasert

Abstract

In this thesis presents adaptive noise canceller using variable step-size adaptive IIR notch filter. The proposed adaptive algorithms consist of three types of variable step-size algorithms. The first type is the variable step-size algorithm 1 (VSS 1) using the autocorrelation of output signal and the autocorrelation of the error signal and mean square error (MSE) to control the adaptation process. The second type is the variable step-size algorithm 2 (VSS 2) using the new autocorrelation of the output signal to control the step-size of the algorithm. In both cases, the first type and second type algorithms are used a fixed bandwidth parameter of the filter. Finally, the third type is variable notch bandwidth (VNB) algorithm using the variable notch bandwidth technique to control the adaptation process of the algorithm. To evaluate the performance of the proposed algorithms are used to estimate a single sinusoidal in Gaussian noise. The simulation results show that the proposed algorithms provide fast convergence speed, the estimated parameters of the filter such as variance, bias and mean square error (MSE) with high accuracy in Gaussian noise environment.

Acknowledgements

This thesis was possible because of the help and support of numerous individuals. I would like to take this occasion to express my appreciation to all of them.

Firstly, I would like to express my sincere thanks to my advisor, Assoc. Prof. Dr. Chawalit Benjangkaprasert, who give me not only many helpful suggestions and useful advice. I also appreciate their attention and stimulating the progress of this research.

I would like to thank Assoc. Prof. Dr. Kanok Janchitrapongvej, Assoc. Prof. Nopphin Anantrasisrichai, Assoc. Prof. Ornlarp Sangaroon and Ms Vanvisa Chutehavong for their assist and suggestion me during my research in KMITL, University of Thailand.

I would like to appreciate Prof. Dr. Takenobu MATSUURA, Department of Electrical and Electronic Engineering, School of Engineering, Tokai University, Japan for helpful suggestion during his short-term expert at KMITL.

This research was enables by the sponsorship of AUN/SEED-Net. I wish to express my sincere thanks to the AUN/SEED-NET for their financial support. I also thank international school of Engineering, Graduate School of KMITL University and department of information engineering of KMITL University for their many helps. I also thank all friends in Thailand, Bangkok.

I, particularly, thank Department of Electronic, Faculty of Engineering, National University of Laos for giving the best opportunities to do this research through the AUN/SEED-Net and JICA project.

Finally, I shall always be grateful to my family, for providing me with the best educational opportunities. They have been a source of incessant and encouragement, which have been vital to all of my accomplishments.

Manolom Chanthavong

Table of Contents

Table of Contents (Cont)

Table of Contents (Cont)

List of Tables

List of Figures

List of Figures (Cont)

Chapter 1

Introduction

This chapter describes the concept of adaptive noise canceller of Widrow [1], an alternative method of estimating signals corrupted by additive noise or interference which is a mixture of the signal source and the noise source. The principle of interference cancellation is to obtain an estimate of the interfering signal and subtract that from the corrupted signal. The feasibility of this idea relies on the availability of a reference source from which the interfering signal originates.

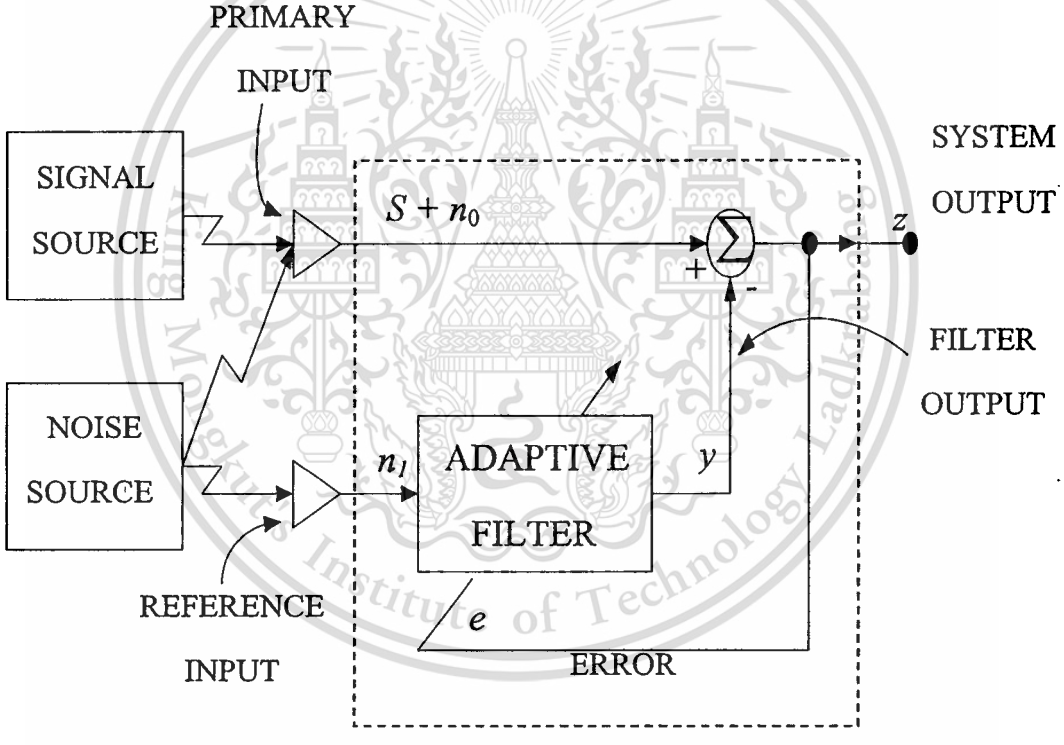


Figure 1.1 Adaptive noise canceller.

In the Figure 1.1 depicts the concept of adaptive noise canceller. A signal s is transmitted over a channel to a sensor that it has also a noise to be received, as represented by n_0, then n_0 correlated with the signal s. Which means the combination of the signal and noise, as represented by $s + n_0$, are occurred from the primary input then go through the canceller. A

second sensor receives a noise n_1 uncorrelated with the signal but correlated in some unknown way with the noise n_0. This sensor provides the reference input to the canceller. The noise n_1 is filtered to produce an output y that is as close a replica as possible of n_0. This output is subtracted from the primary input $s+n_0$ to produce the system output $z=s+n_0-y$. Subtracting noise from a received signal would seem to be a dangerous procedure. If done improperly it could result in an increase in output noise power. However, filtering and subtraction are controlled by an appropriate adaptive process, noise reduction can be accomplished with little risk of distorting the signal or increasing the output noise level.

In the system shown in Figure 1.1 the reference input is processed by an adaptive filter. An adaptive filter differs from a fixed filter in that it automatically adjusts its own impulse response. Adjustment is accomplished through an algorithm that responds to an error signal dependent, among other things, on the filter's output. Thus with the proper algorithm, the filter can operate under changing conditions and can readjust itself continuously to minimize the error signal.

Adaptive digital filter are very useful in many signal processing applications such as communication, sonar, radar and biomedical engineering [1-2]. One of the interest applications of adaptive digital filters is the detection of corrupted by white Gaussian noise. There are two methods of adaptive noise canceller. First method is a canceling method used a reference input [1]. Second method is using an IIR (Infinite Impulse Response) notch filter. This thesis treats the method using an IIR notch filter.

1.1 Objective of the Study

In this thesis presents adaptive noise canceller by using variable step-size adaptive IIR notch filter. It provides three types of variable step-size algorithms.

The first type is variable step-size algorithm 1 [10] that using the autocorrelation of output signal, the autocorrelation of the error signal and mean square error (MSE) to control the adaptation process of the algorithm.

The second type is variable step-size algorithm 2 that using the new autocorrelation of the output signal and the power of output signal to control the adaptation process of the algorithm.

Finally, the third type is variable notch bandwidth algorithm that using the variable notch bandwidth technique to control the adaptation process of the algorithm.

The objective of the proposed algorithms is to improve the performance of the adaptive IIR notch filter such as fast convergence speed, low variance, and low means-square error (MSE).

1.2 Scopes of Research

The aim of this thesis is to develope the new algorithms for adaptive noise canceller using adaptive IIR notch filter. To evaluate and verify the performance of the proposed algorithms are used to estimate sinusoidal signal corrupted by white Gaussian noise.

1.3 The Outline of the Thesis

This thesis consists of 6 chapters as follows:

The chapter 1 is an introduction of adaptive noise canceller. The objectives of this thesis are described.

In chapter 2, presents theory and design of digital filters. There are two fundamental types of digital filters: first type is finite impulse response (FIR) filter and another type is infinite impulse response (IIR) filter, inwhich an IIR filter and structure are described.

In chapter 3, is basic theory of discrete time random processes. Beginning with a review of random process, it discusses classification of random processes, characterization of random processes including the autocorrelation function of a random process, autocorrelation function and crosscovariance function, stationary random processes.

In chapter 4 provides theory and structure of a second order adaptive IIR notch filter. The useful adaptive algorithm such as least mean square (LMS) algorithm, variable step-size LMS (VSLMS) algorithm are described. Finally, the proposed algorithms (VSS 1, VSS 2 and VNB) for the adaptive noise canceller are presented.

In chapter 5 describes the numerical results of the performance of the adaptive noise canceller. The computer simulation results are presented to demonstrate the convergence

performance, variance, bias and steady-state mean square error of the previous algorithm and the proposed algorithms.

Finally, chapter 6 is conclusion.

Chapter 2

Digital Filters

2.1 Introduction

In electronics, computer science and mathematics, a digital filter is a system that performs mathematical operations on a sampled, discrete-time signal to reduce or enhance certain aspects of that signal. This is in contrast to the other major type of electronic filter, the analog filter, which is an electronic circuit operating on continuous-time analog signals. An analog signal may be processed by a digital filter by first being digitized and represented as a sequence of numbers, then manipulated mathematically, and then reconstructed as a new analog signal. In an analog filter, the input signal is "directly" manipulated by the circuit

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range.

Figure 2.1 Basic idea of filter.

There are two main kinds of filter, analog and digital. They are quite different in their physical makeup and in how they work.

An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and op-amps to produce the required filtering effect. Such filter circuits are widely used in such applications as noise reduction, video signal enhancement, graphic equalisers in hi-fi systems, and many other areas.

There are well- established standard techniques for designing an analog filter circuit for a given requirement.

At all stages, the signal being filtered is an electrical voltage or current which is the direct analogue of the physical quantity (e.g. a sound or video signal or transducer output) involved.

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialised DSP (Digital Signal Processor) chip.

The analog input signal must first sampled and digitised using an ADC (analog to digital converter). The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them. These calculations typically involve multiplying the input values by constants and adding the products together. If necessary, the results of these calculations, which now represent sampled values of the filtered signal, are output through a DAC (digital to analog converter) to convert the signal back to analog form.

Note that in a digital filter, the signal is represented by a sequence of numbers, rather than a voltage or current.

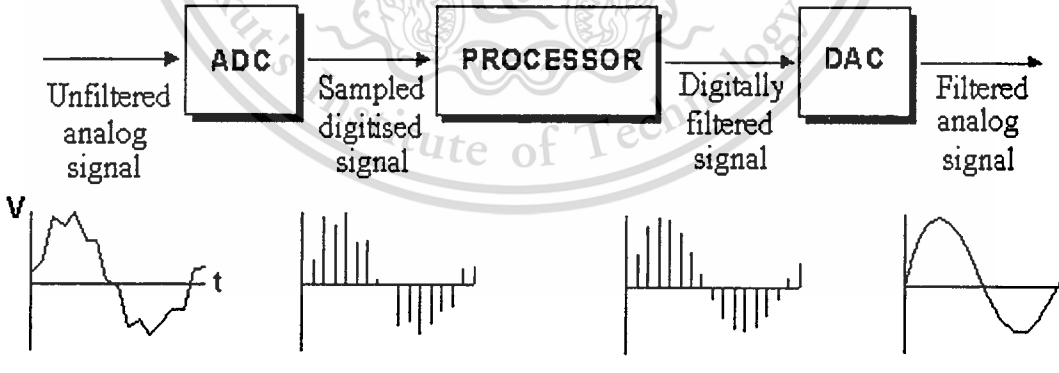


Figure 2.2 Block diagram of signal processing.

- **Advantages of Using Digital Filters**

The following list gives some of the main advantages of digital over analog filters.

1. A digital filter is programmable, i.e. its operation is determined by a program stored in the processor's memory. This means the digital filter can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.

2. Digital filters are easily designed, tested and implemented on a general-purpose computer or workstation.

3. The characteristics of analog filter circuits (particularly those containing active components) are subject to drift and are dependent on temperature. Digital filters do not suffer from these problems, and so are extremely stable with respect both to time and temperature.

4. Unlike their analog counterparts, digital filters can handle low frequency signals accurately. As the speed of DSP technology continues to increase, digital filters are being applied to high frequency signals in the RF (radio frequency) domain, which in the past was the exclusive preserve of analog technology.

5. Digital filters are very much more versatile in their ability to process signals in a variety of ways; this includes the ability of some types of digital filter to adapt to changes in the characteristics of the signal.

6. Fast DSP processors can handle complex combinations of filters in parallel or cascade (series), making the hardware requirements relatively simple and compact in comparison with the equivalent analog circuitry.

In the term "filter" [2] is often used to describe a device in the form of a piece of physical hardware or software that is applied to a set of noisy data in order to extract information about a prescribed quantity of interest. The noise may arise from a variety of sources. For example, the data may have been derived by means of noisy sensors or may represent a useful signal component that has been corrupted by transmission through a communication channel. In any event, we may use a filter to perform three basic information processing tasks.

- **Filtering** means the extraction of information about a quantity of interest at time t by using data measured up to and including time t.

- **Smoothing** differs from filtering in that information about the quantity of interest need not be available at time t, and data measured later than time t can be used in obtaining this information.

This means that in the case of smoothing there is a delay in producing the result of interest. Since in the smoothing process we are able to use data obtained not only up to time t, but also data obtained after time t, we would expect smoothing to be more accurate in some sense than filtering.

- **Prediction** is the forecasting side of information processing. The aim here is to derive information about what the quantity of interest will be like at some time $t + \tau$ in the future, for some $\tau > 0$, by using data measured up to and including time t.

We may classify filters into linear and nonlinear. A filter is said to be linear if the filtered, smoothed, or predicted quantity at the output of the device is a linear function of the observations applied to the filter input. Otherwise, the filter is nonlinear.

There are two fundamental types of digital filters: finite impulse response (FIR filter) and infinite impulse response (IIR filter). As the terminology suggests, these classifications refer to the filter's impulse response. By varying the weight of the coefficients and the number of filter taps, virtually any frequency response characteristic can be realized with an FIR filter. However, high performance FIR filters generally require a large number of multiply-accumulates and therefore require fast and efficient DSP. On the other hand, IIR filters tend to mimic the performance of traditional analog filters and make use off feedback. Therefore their impulse response extends over an infinite period of time. Because of feedback, IIR filters can be implemented with fewer coefficients than for an FIR filter. Some advantages and disadvantages of FIR filters compared to their IIR counterparts are as follows.

	IIR filters	FIR filter
Transfer function	Zeros and poles	Only zeros and no poles
Phase	Difficult to control, no particular.	Linear phase always possible
Order	Less	More
Stability	Can be unstable	Always stable
others	Use feedback (recursion)	Not use feedback (non-recursion)
	Analog equivalent	No analog equivalent
		Easy to understand and design

Table 2.1 Comparison between FIR and IIR filters.

In this chapter we have interest on the techniques used to design IIR or recursive filters only.

2.2 Infinite Impulse Response Filters

IIR (Infinite Impulse Response) filters are important components in a variety of systems for discrete time processing. They offer a number of advantages over FIR filters in certain applications because of their excellent magnitude response characteristics, particularly when the width of a pass band or stop band must be very narrow or when very narrow transition bands or high attenuations are required.

2.2.1 Transfer Function Derivation

IIR filters are can be implemented efficiently using sets of recursive difference equations of the output of finite order linear time invariant system at time n can be expressed as a linear combination of the input and outputs,

$$y(n) = \sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

or

$$y(n) - \sum_{k=1}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k) \qquad (2.1)$$

where a_k and b_k are constants with $a_0 \neq 0$ and $M \leq N$

The current output $y(n)$ is equal to the sum of past outputs, from $y(n-1)$ to $y(n-N)$, which are scaled by the delay-dependent feedback coefficient a_k, plus the sum of future, present and past inputs, which are scaled by the delay-dependent feed forward coefficient b_k. $x(n)$ is input sequence and $y(n)$ is the output sequence. When N consecutive sample of $y(n)$ and $x(n)$ (known as initial conditions) are known, in Equation (2.1) can be used to calculate all off the successive output samples. This type of realization, in which previously computed output sample are used to compute future ones, is called recursive. The terms IIR filter and recursive filter are often used synonymously because of Equation (2.1) can realize both, but technically, they are different. The adjective "IIR" refers to the form of the impulse response of the filter while the adjective "recursive" tells how it is implemented. There are recursive implementations of FIR filters. And there are IIR filters that can not be implemented recursively. Nonetheless, we will use these terms interchangeably in this chapter without ambiguity, since we will always be discussing IIR filters described by difference equations of the form given in Equation (2.1).

Recursive filters are conveniently defined by their system functions. (recall that the system function is the z - transform of the impulse response of the filter). These are rational function in the variable z^{-1}. The system specified by Equation (2.1) has the system function.

Taking the z-transform of the output sequence $y(n)$ given in Equation (2.1), we get

$$Y(z) = \sum_{n=-\infty}^{\infty} y(n) z^{-n}$$

$$= \sum_{n=-\infty}^{\infty} \left(\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=1}^{M} b_k x(n-k) \right) z^{-n}$$

changing the order of summation

$$Y(z) = \sum_{k=1}^{N} a_k \left\{ \sum_{n=-\infty}^{\infty} y(n-k) z^{-n} \right\} + \sum_{k=0}^{M} b_k \left\{ \sum_{n=-\infty}^{\infty} x(n-k) z^{-n} \right\}$$

$$= \sum_{k=1}^{N} a_k z^{-k} Y(z) + \sum_{k=0}^{M} b_k z^{-k} X(z)$$

$$Y(z)\left\{1-\sum_{k=1}^{N}a_k z^{-k}\right\}=\sum_{k=0}^{M}b_k z^{-k}X(z)$$

Therefore,

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{k=0}^{M}b_k z^{-k}}{1-\sum_{k=1}^{N}a_k z^{-k}} \qquad (2.2)$$

If it is at initial rest (Example: if it has zero initial conditions). This can also be written in factored form as

$$H(z)=G\frac{\prod_{m=0}^{M}(1-\beta_k z^{-1})}{\prod_{n=1}^{N}(1-\alpha_k z^{-1})} \qquad (2.3)$$

The roots of the numerator polynomial, β_k, is called the *zeros* of the filter, and the roots of the denominator polynomial, α_k, are called the poles. The final parameter G is a constant gain. In general, the number of poles and zeros is directly related to the effort required to implement the filter; it is also related to how well the frequency response of the filter can approximate an ideal response. The filter order, N, for an IIR filter is the number of poles that it contains in the finite z-plane.

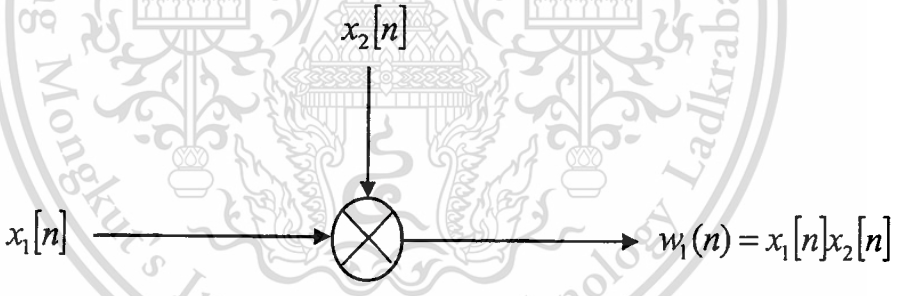
A linear, time invariant filter is causal if its impulse response is zero for $n<0$. The current output sample of a causal filter depends only on the current and previous samples of the input. When we limit the inputs to the filter to those for which $x(n)=0$, for $n<0$ and initialize Equation (2.1) with the initial conditions $y(-1)=y(-2)=\ldots=y(-N)=0$, the recursive implementation in Equation (2.1) results in a causal filter. Causality is an important constraint for filters that must operate in a real-time environment, where at each tick of a cock one input sample arrives and one output sample must be produced. In certain non-real-time applications, where the entire input signal has been received before any processing begins, noncausal filters can be used.

An even more important constraint on an IIR filter is that it be stable. If a filter is unstable, its output sequence can grow without bounds even when the input signal is well

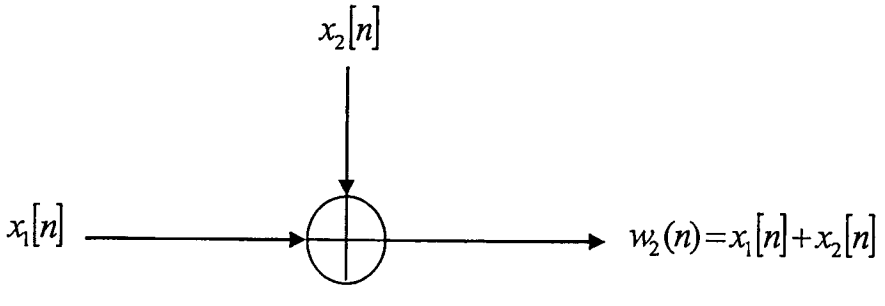
behaved. Whether or not a filter is stable depends on the locations of its poles; a causal LTI IIR filter is stable if and only if each pole satisfies the condition $|\alpha_k| < 1$. This means that all of its poles must lie inside the unit circle in the z - plane.

2.2.2 Description of Block Diagram [5].

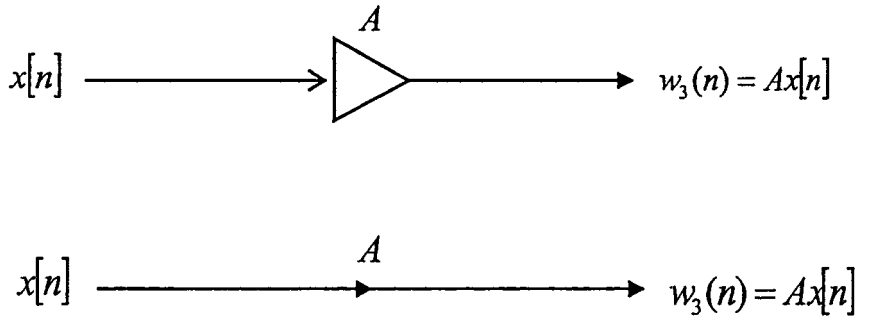
The implementation of the linear time invariant discrete-time system by iteratively evaluating a recurrence formula obtained from a difference equation requires that delay values of the output, input and intermediate sequence be available. The delay of sequence values implies the need for storage of past sequence values. Also, we must provide means of multiplication of the delayed sequence values by the coefficient as well as means for adding the resulting products. Therefore, the basic elements required for implementation of a linear time-invariant discrete-time system are adders, modulator, multipliers and memory for storing delayed sequence values. The interconnection of these basic elements is conveniently depicted by block diagrams composed of the basic pictorial symbols show in Figure 2.3.



(a) Modulator

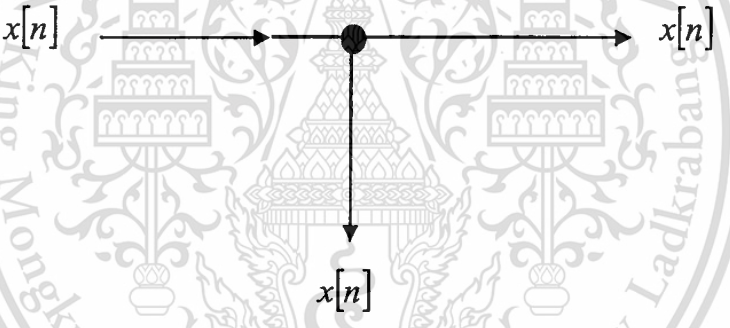


(b) Adder



(c) Multiplier

(d) Unit delay



(e) Pick-off node

Figure 2.3 Schematic representation of basic operations on sequences:
(a) Modulator, (b) Adder, (c) Multiplier, (d) Unit delay and (e) Pick-off node.

Basic Operations

Let $x_1[n]$ and $x_2[n]$ be two known sequences. By forming the product of the sample values of these two sequences at each instant, we form a new sequence $w_1(n) = x_1[n]x_2[n]$, in some applications, the product operation is also known as modulation. The device implementing the modulation operation is called a modulator and its schematic representation is shown in Figure 2.3 (a)

The second basic operation is the addition by which a new sequence $w_2(n)$ is obtained by adding the sample values of two sequences $x_1[n]$ and $x_2[n]$: $w_2[n] = x_1[n] + x_2[n]$, the device implementing the addition operation is called an adder and its schematic representation is shown in Figure 2.3 (b)

The third basic operation is the scalar multiplication, where by a new sequence generated by multiplying each sample of $x[n]$ by a scalar A: $w_3[n] = Ax[n]$, the device implementing the multiplication operation is called a multiplier and its schematic representation is shown in Figure 2.3 (c)

The time-shifting operation illustrated in Figure 2.3 (c) shows the relation between $x[n]$ and its time-shifted version $w_4[n] = x[n-N]$, where N is integer. If $N > 0$, it is a delaying operation and if $N < 0$,it is advancing operation. The device implementing the delay operation by one sample is called a unit delay and its schematic representation is shown in Figure 2.3 (d)

In Figure 2.3 (e) we also show a pick-off node which is used to provide multiple copies of a sequence.

Advantages of representing the digital system in block diagram form

- Just by inspection, the computation algorithm can be easily written
- The hardware requirements can be easily determined
- A variety of equivalent block diagram representations can be easily developed from the transfer function
- The relationship between the output and the input can be determined.

2.3 Basic of Structures for IIR Systems [3]

Causal IIR systems are characterized by the constant coefficient difference equation of Equation (2.1) or equivalently, by the real rational transfer function of Equation (2.2). From these equations, it can be seen that the realization of infinite duration impulse response (IIR) systems involves a recursive computational algorithm. In this section, the most important filter structures direct forms I and II, lattice form realizations for IIR systems are discussed.

2.3.1 Direct Form Realization of IIR System

Equation (2.2) is the standard form of the system transfer function. By inspection of this equation, the block diagram representation can be drawn directly for the direct form realization. The multipliers in the feedforward paths are the numerator coefficients and the multipliers in the feedback paths are the negatives of the denominator coefficients. Since the multiplier coefficients in the structures are exactly the coefficients of the transfer function, they are call direct form structures.

2.3.1.1 Direct Form I

The digital system structure determined directly from either Equation (2.1) or Equation (2.2) is called the direct form I. In the case, the system function is divided into two parts connected in cascade, the first part containing only the zeros, followed by the part containing only the poles. An intermediate sequence $u(n)$ is introduced. A possible IIR system direct form I realization is shown in Figure (2.4), in which $u(n)$ represents the output of the first part and input to the second.

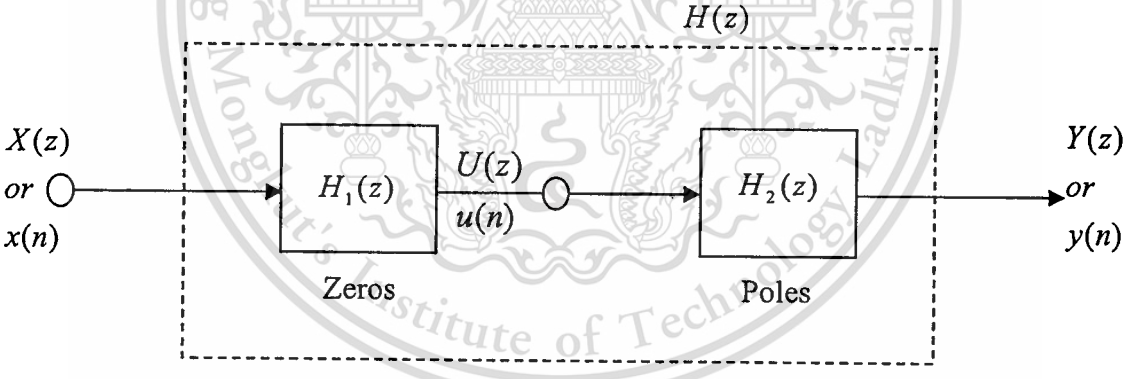


Figure 2.4 A possible IIR system direct-form realization.

$$u(n) = \sum_{k=0}^{M} b_k x(n-k)$$

or

$$u(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \ldots + b_M x(n-M) \qquad (2.4)$$

and

$$y(n) = \sum_{k=1}^{N} a_k y(n-k) + u(n)$$

or

$$y(n) = -a_1 y(n-1) + a_2 y(n-2) + a_3 y(n-3) + \ldots + a_N y(n-N) + u(n) \quad (2.5)$$

Taking z-transform, we get

$$U(z) = \sum_{k=0}^{M} b_k z^{-k} \quad (2.6)$$

and

$$Y(z) = Y(z)\sum_{k=1}^{N} a_k z^{-k} + U(z) \quad (2.7)$$

Therefore,

$$Y(z) = \frac{U(z)}{1 - \sum_{k=1}^{N} a_k z^{-k}} \quad (2.8)$$

The direct form I realization is shown in Figure 2.5. The direct form I realization requires $M + N$ storage elements. For out convenience, we have assumed that $M = N$, for drawing the network.

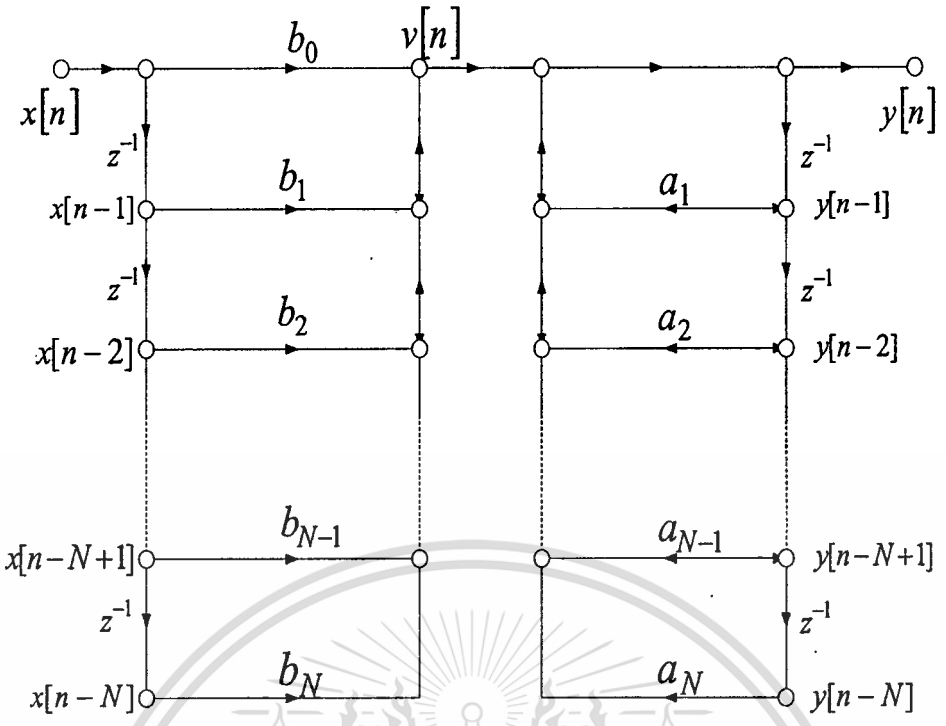


Figure 2.5 Direct form I realization of an N order system.

2.3.1.2 Direct Form II

Since we are dealing with linear systems, the order of these parts can be interchanged. This property yields a second direct form realization. In direct form II, the poles of $H(z)$ are realized first and the zeros second. Here, the transfer function $H(z)$ is broken into a product of two transfer functions $H_1(z)$ and $H_2(z)$, where $H_1(z)$ has only poles and $H_2(z)$ contains only the zeros as given below:

$$H(z)=H_1(z).H_2(z)$$

where $$H_1(z)=\frac{1}{\left(1-\sum_{k=1}^{N}a_k z^{-k}\right)}$$ and $$H_2(z)=\sum_{k=0}^{M}b_k z^{-k}$$

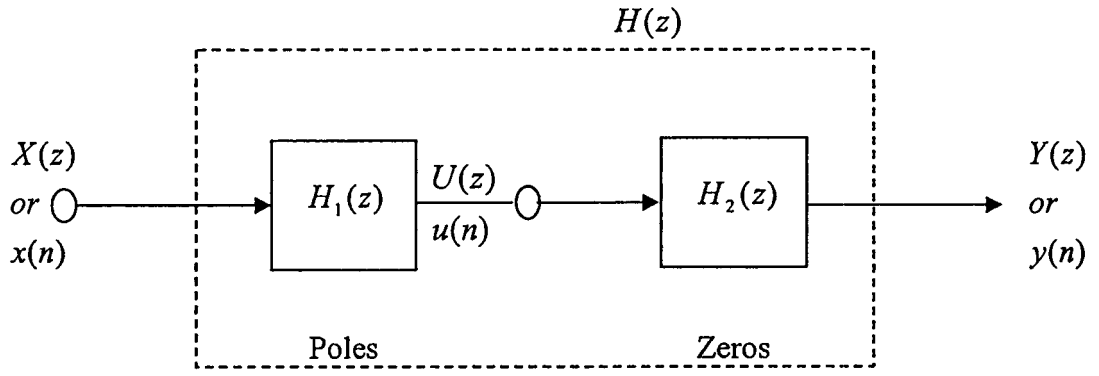


Figure 2.6 The decomposition for direct form II realization.

The decomposition for direct form II realization is shown in Figure 2.6 an intermediate sequence $u(n)$ is introduced to obtain the output of the filter.

$$u(n) = \sum_{k=1}^{N} a_k u(n-k) + x(n)$$

or

$$u(n) = x(n) - a_1 u(n-1) - a_2 u(n-2) - a_3 u(n-3) - \ldots - a_N u(n-N) \quad (2.9)$$

and

$$y(n) = \sum_{k=0}^{M} b_k u(n-k)$$

or

$$y(n) = b_0 u(n) + b_1 u(n-1) + b_2 u(n-2) + \ldots + b_M u(n-M) \quad (2.10)$$

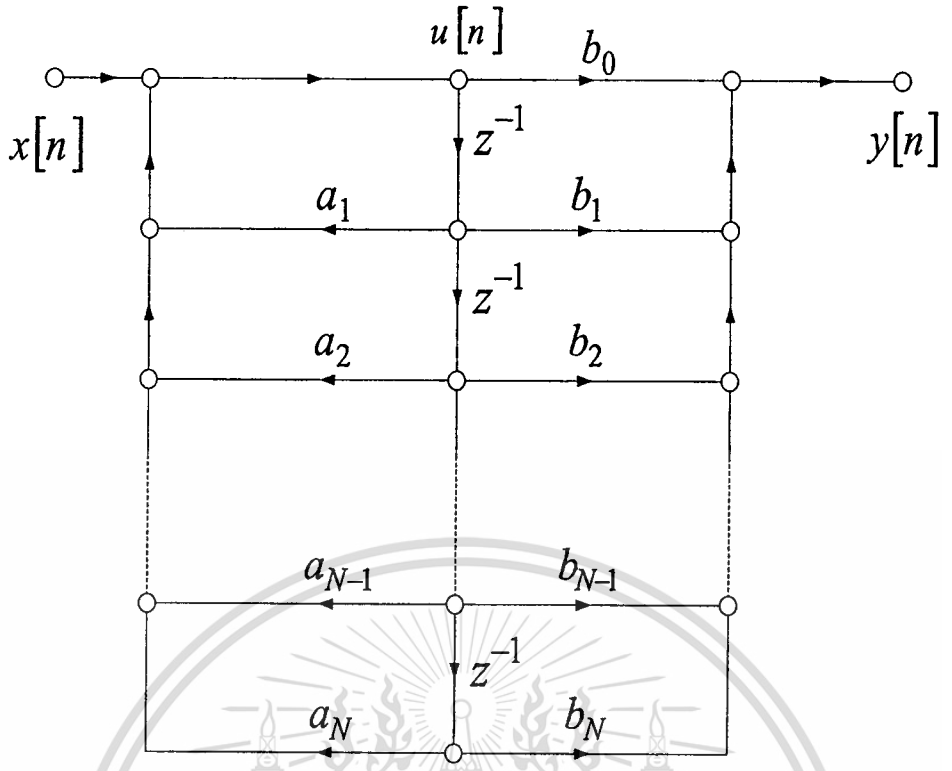


Figure 2.7 Direct form II realization of an N order system.

Taking z-transform of the above equations, we get

$$U(z) = \frac{X(z)}{1 - \sum_{k=1}^{M} a_k z^{-k}} \tag{2.11}$$

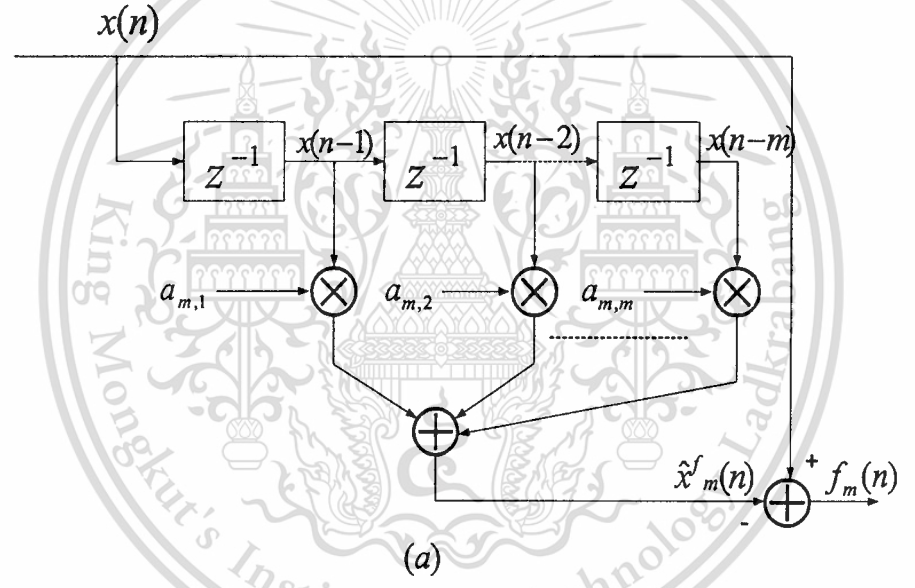
and

$$Y(z) = U(z) \sum_{k=0}^{M} b_k z^{-k} \tag{2.12}$$

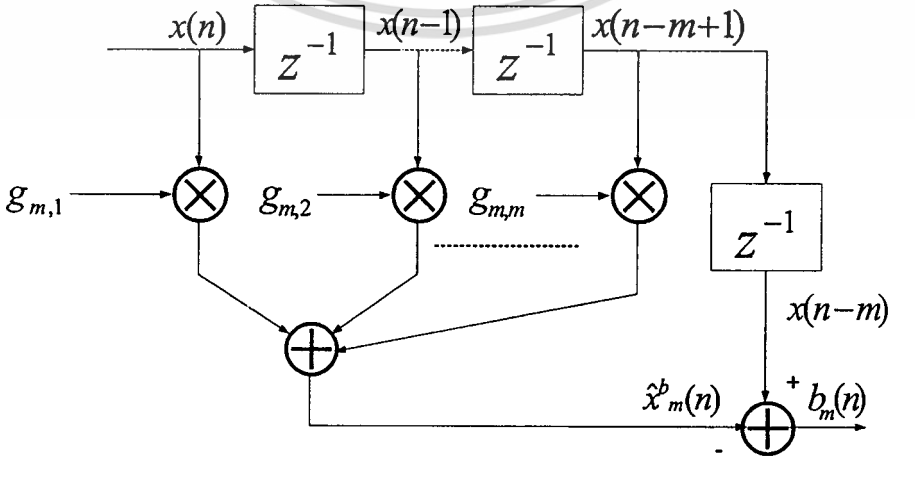
The direct form II realization requires only the larger of M or N storage elements. When compared to direct form I realization, the direct form II uses the minimum number of storage elements and hence said to be a canonic structure. However, when the addition is performed sequentially, the direct form II needs two adders instead of one adder required for the direct form I.

2.3.2 The Lattice Form Structure [6]

The prediction-error filter (PEF) is defined as a structure which combines successive samples of a signal multiplied by coefficients, so that the output (prediction-error) power of the filter is minimized. There are two kinds of PEF, depending on the form of prediction error utilized. Based on a given sequence of input samples, a forward PEF is designed to minimize the mean-square value of the forward prediction error, defined as the difference between the predicted value of the input one step into the future and its actual value. On the other hand, a backward PEF is designed to minimize the mean square value of the backward prediction error, defined as the difference between the predicted value of the input one step into the past and its actual value.



(a)



(b)

Figure 2.8 Lattice-structure prediction-error filter.

(a) Forward linear predictor

(b) Backward linear predictor

Figure 2.8 (a) forward linear prediction: depicts the direct implementation of an m order forward linear predictor. A transversal filter with tab-input vector to $x_m(n-1)=[x(n-1)\ x(n-2)\cdots x(n-m)]^T$ and tap-weight vector $a_m=[a_{m,1}a_{m,2}....a_{m,m}]^T$ is used to obtain an estimate of the input simple $x(n)$. We use the subscript m in the vectors a_m and $x_m(n)$ and the elements of the a_m to emphasize that the predictor order is m. The implementation structure of the type shown in Figure 2.8 is called the transversal or tapped delay line predictor.

We assume that the input sequence, $x_m(n)$ is the realization of stationary stochastic process. Furthermore, we assume that the predictor tap weights are optimized in the mean-square sense according to the Wiener filter theory. Thus, the optimum value of the predictor tap weights $a_{m,1}, a_{m,2}, \ldots\ldots, a_{m,m}$ is obtained by minimizing the function.

$$P_m^f = E[f_m^2(n)] \tag{2.13}$$

where

$$f_m(n) = x(n) - \hat{x}_m^f(n) \tag{2.14}$$

is the forward prediction error and

$$\hat{x}_m^f(n) = \sum_{i=0}^{m} a_{m,i}x(n-i) = a_m^T x_m(n-1) \tag{2.15}$$

is the m^{th} order forward prediction of the input sample $x(n)$.

Figure 2.8 (b) backward linear prediction: depicts an m order backward linear predictor. A transversal filter with tab-input vector $x_m(n)=[x(n)x(n-1)x(n-2)\cdots x(n-m+1)(n-m+1)]^T$ and tap-weight vector $g_m=[g_{m,1}g_{m,2}\cdots g_{m,m}]^T$ is used to obtain an estimate of the input simple $x(n-m)$. As the forward prediction case, we assume that the backward predictor tap weights are optimized in the mean-square sense according to the Wiener filter theory. the

optimum value of the predictor tap weights $g_{m,1}, g_{m,2}, \ldots\ldots, g_{m,m}$ are then obtained by minimizing the function.

$$P_m^b = E[b_m^2(n)] \tag{2.16}$$

where

$$b_m(n) = x(n-m) - \hat{x}_m^b(n) \tag{2.17}$$

is the backward prediction error and

$$\hat{x}_m^b(n) = \sum_{i=0}^{m} g_{m,i}\, x(n-i+1) = g_m^T x_m(n) \tag{2.18}$$

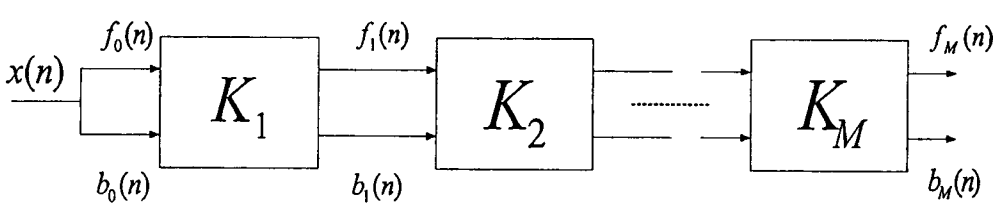
is the m^{th} order backward prediction of the input sample $x(n-m)$.

The optimum tap weights of the m^{th} order forward predictor of the wide sense stationary process $x(n)$ are the same as the optimum tap weights of the corresponding backward predictor, but in reverse order. Thus, we may write

$$f_n(n) = x(n) - \sum_{i=1}^{m} a_{m,i}\, x(n-i) \tag{2.19}$$

and

$$b_m(n) = x(n-m) - \sum_{i=1}^{m} a_{m,m+1-i}\, x(n-i+1) \tag{2.20}$$



(a)

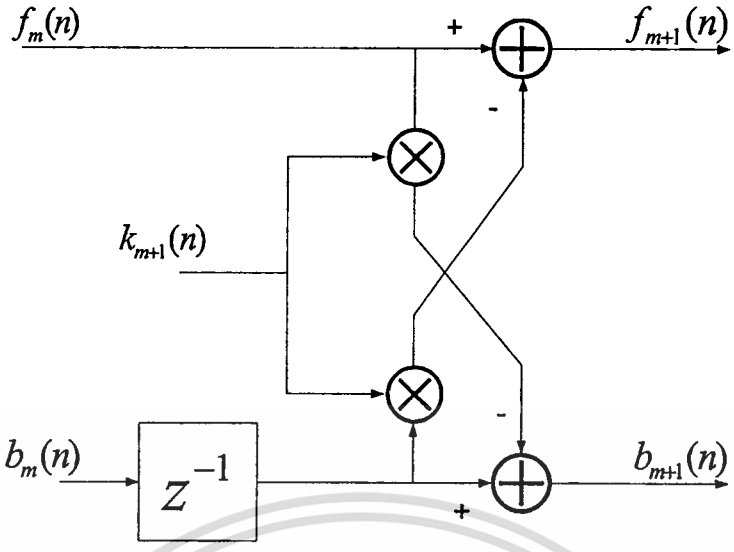


Figure 2.9 Lattice predictor:

(a) Overall structure

(b) Details of stage m.

Figure 2.9 (a) shows that the lattice structure of and M -stage forward and backward predictor. Each stage has two inputs. These are the forward and backward prediction errors from the previous stage. The output of each stage are the forward and backward prediction errors of one order higher. These are calculated according to the order-update Equations (2.21) and (2.22). The two inputs to stage 1 are common and are equal to the predictor input $x(n)$.

Figure 2.9 (b) depicts the details of the m^{th} stage of the lattice predictor. It follows from the order-update Equations (2.21) and (2.22).

An IIR lattice filter is a cascade of two-port networks as shown in Figure 2.9 (b). Each two-port network is defined by the value of its reflection coefficient, k_m. The two inputs, $f_m(n)$ and $b_m(n)$ are related to the outputs $f_{m+1}(n)$ and $b_{m+1}(n)$ by a pair of coupled difference equations

$$f_{m+1} = f_m(n) - k_{m+1} b_m(n-1) \tag{2.21}$$

$$b_{m+1} = b_m(n) - k_{m+1} f_m(n) \tag{2.22}$$

A special feature of the lattice predictor is that to obtain the M th order prediction errors, all prediction errors (forward and backward) of lower orders are also calculated.

In other words, the M th order lattice predictor is a structure with a single input $x(n)$ and $2M+2$ outputs $f_0(n), b_0(n), f_1(n), b_1(n), \ldots\ldots, f_M(n)$ and $b_M(n)$. How many of these outputs will be used is application dependent. For example, in and M th order forward predictor where the final goal is $f_M(n)$, the rest of the prediction errors (with the exception of $b_M(n)$ which, in this case, can be dropped from the structure) are required only as intermediate signal sequences.

A useful relationship which we shall establish before ending this section is an order update equation for the mean square value of the prediction errors. We not that

$$\begin{aligned} P_{m+1} &= E[f_{m+1}^2(n)] \\ &= E[f_{m+1}(n)x(n)] - \sum_{i=1}^{m+1} a_{m+1,i} E[f_{m+1}(n)x(n-i)] \\ &= E[f_{m+1}(n)(x(n) - \sum_{i=1}^{m+1} a_{m+1,i} x(n-i))] \end{aligned} \tag{2.23}$$

But from property 2 of the prediction errors we know that all the expectations under the summation on the right-hand side of Equation (2.23) are zero. Thus we obtain

$$P_{m+1} = E[f_{m+1}(n)x(n)] \tag{2.24}$$

Substituting for $f_{m+1}(n)$ and $x(n)$ in Equation (2.24) from Equation (2.21) and Equation (2.19) respectively, we get

$$P_{m+1} = E[(f_m(n) - k_{m+1} b_m(n-1))(f_m(n) + \sum_{i=1}^{m} a_{m,i} x(n-1))]$$

$$= E[f_m^2(n)] - k_{m+1} E[f_m(n) b_m(n-1)] \tag{2.25}$$

when $m=0,1,2,3,....$ and the input to the first section being $f_0(n)=b_0(n)=x(n)$ and reflection coefficients k_{m+1} which is

$$k_{m+1}=\frac{e[f_m(n)b_m(n-1)]}{P_m} \tag{2.26}$$

$$P_{m+1}=(1-k_{m+1}^2)P_m \tag{2.27}$$

There are a number of important advantages to using the lattice structure PEF. One of the most important of these is the fact that for each stage the backward prediction error at the output is orthogonal to both prediction errors at the input.

2.4 Stability Test [5]

The transfer function in Equation (2.3) that the BIBO stability of causal rational transfer function requires that all its poles be inside the unit circle. For very high order transfer functions, it is difficult to determine the pole locations analytically, and the use of some type of root finding computer program is necessary.

For the second-order transfer function, the stability can be checked easily by examining its denominator coefficients. Let

$$D(z)=1+d_1z^{-1}+d_2z^{-2} \tag{2.28}$$

denote the denominator of the transfer function. In terms of its poles, $D(z)$ can be expressed as

$$D(z)=(1-\lambda_1z^{-1})(1-\lambda_2z^{-1})=1-(\lambda_1+\lambda_2)z^{-1}+\lambda_1\lambda_2z^{-2} \tag{2.29}$$

comparing Equations (2.26) and (2.27), we obtain

$$d_1=-(\lambda_1+\lambda_2) \text{ and } d_2=\lambda_1\lambda_2 \tag{2.30}$$

Now for stability of the transfer function, its poles must be inside the unit circle; that is,

$$|\lambda_1| < 1 \quad \text{and} \quad |\lambda_2| < 1 \tag{2.31}$$

Since from Equation (2.28), the coefficient d_2 is given by the product of the poles, we must have

$$|d_2| < 1 \tag{2.32}$$

Now the roots of the polynomial $D(z)$ are given by

$$\lambda_1 = -\frac{d_1 + \sqrt{d_1^2 - 4d_2}}{2}$$

$$\lambda_2 = -\frac{d_1 - \sqrt{d_1^2 - 4d_2}}{2} \tag{2.33}$$

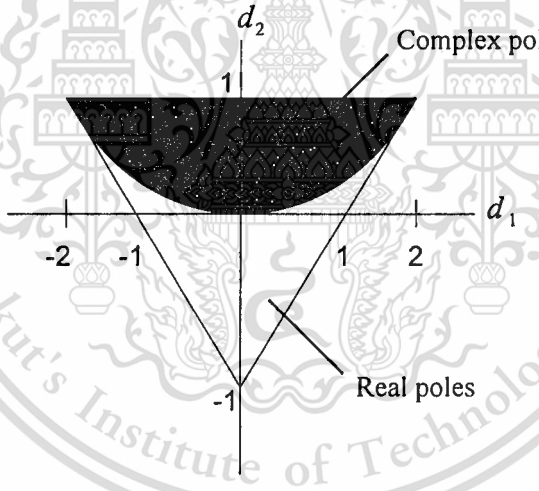


Figure 2.10 Stability triangle for a second-order IIR digital transfer function.

Subtraction Equation (2.33) in Equation (2.31) we get

$$|\lambda_2| < 1$$

$$-\frac{d_1 - \sqrt{d_1^2 - 4d_2}}{2} < 1$$

$$\sqrt{d_1^2 - 4d_2} < d_1 + 2$$

$$d_1^2 - 4d_2 < d_1^2 + 4d_1 + 4$$

So we get $$|d_1| < 1 + d_2 \tag{2.34}$$

The region in the (d_1, d_2) plane where the two coefficient conditions of Equations (2.32) and (2.34) are satisfied is a triangle, as sketched in Figure 2.10, and is known as the stability triangle for a second-order digital transfer function.

Chapter 3

Discrete Time Random Processing

3.1 Random Process [7]

This section introduces the concepts of random and stochastic processes and describes a method for generation of random numbers.

3.1.1 Information-Bearing Random Signals Vs Deterministic Signals

Signals, in terms of one of their most fundamental characteristics, i.e. their ability to convey information, can be classified into two broad categories:

- Deterministic signals such as sine waves that on their own convey no information but can act as carriers of information when modulated by a random information-bearing signal.
- Random signals such as speech and image that contain information.

In each class, a signal may be continuous or discrete in time, may have continuous-valued or discrete valued amplitudes and may be one-dimensional or multi-dimensional.

A deterministic signal has a predetermined trajectory in time and/or space. The exact fluctuations of a deterministic signal can be described in terms of a function of time, and its exact value at any time is predictable from the functional description and the past history of the signal. For example, a sine wave $x(t)$ can be modelled, and accurately predicted either by a second-order linear predictive model or by the more familiar Equation $x(t) = A\sin(2\pi ft + \phi)$.

Note that a deterministic signal carries no information other than its own shape and characteristic parameters.

Random signals have unpredictable fluctuations; hence it is not possible to formulate an equation that can predict the exact future value of a random signal. Most signals of interest such as speech, music and noise are at least in part random. The concept of randomness is closely associated with the concepts of information, bandwidth and noise.

For a signal to have a capacity to convey information, it must have a degree of randomness: a predictable signal conveys no information. Therefore the random part of a signal is either the information content of the signal, or noise, or a mixture of information and noise. In telecommunication it is wasteful of resources, such as time, power and bandwidth, to retransmit the predictable part of a signal. Hence signals are randomised (de-correlated) before transmission.

Although a random signal is not predictable, it often exhibits a set of well-defined statistical values such as maximum, minimum, mean, median, variance, correlation, power spectrum and higher order statistics. A random process is described in terms of its statistics, and most completely in terms of a probability model from which its statistics can be calculated.

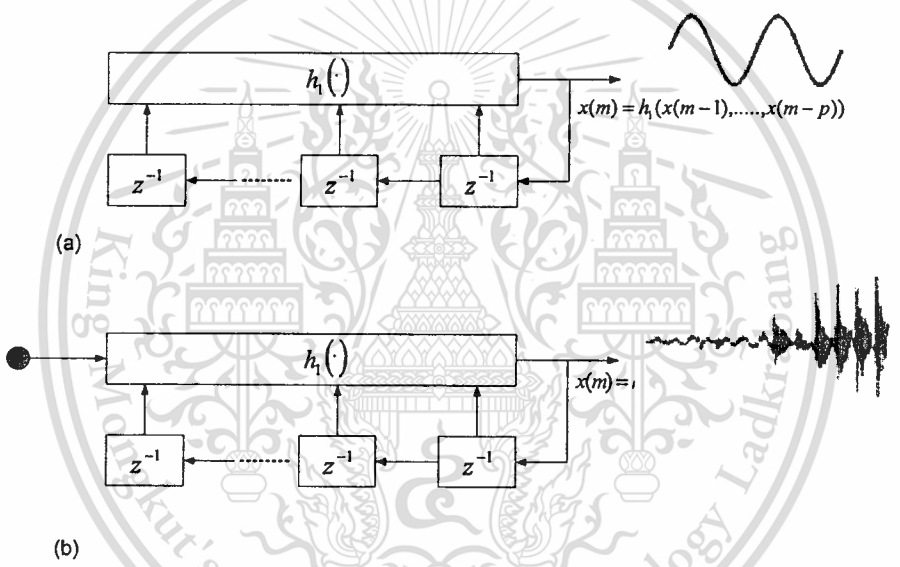


Figure 3.1 Illustration of deterministic and random signal models:
(a) adeterministic signal model, (b) a random signal model.

Figure 3.3 (a) shows a model of a deterministic discrete-time signal. The model generates an output signal $x(m)$ from the P past output samples as

$$x(m) = h_1(x(m-1); x(m-2),..., x(m-P)) + \delta(m) \qquad (3.1)$$

where the function $h1$ may be a linear or a non-linear model and $\delta(m)$ is a delta function that acts as an initial 'kick-start' impulse input. Note that there is no sustained input. A functional

description of the model h_1 together with the P initial sample values are all that is required to generate, or predict the future values of the signal $x(m)$.

For example, for a discrete-time sinusoidal signal generator (i.e. a digital oscillator) Equation (3.1) becomes

$$x(m) = a_1 x(m-1) - a_2 x(m-2) + \delta(m) \tag{3.2}$$

where the choice of the parameter $a_1 = 2\cos(2\pi F_0 / F_S)$ determines the oscillation frequency F_0 of the sinusoid, at a sampling frequency of F_S.

Figure 3.1 (b) illustrates a model for a random signal given by

$$x(m) = h_2(x(m-1); x(m-2), \ldots, x(m-P)) + e(m) \tag{3.3}$$

where the random input $e(m)$ models the unpredictable part of the signal $x(m)$ that is unpredictable, and the function h_2 models the part of the signal that is correlated with and predictable form the past samples. For example, a narrowband, second-order autoregressive process can be modelled as

$$x(m) = a_1 x(m-1) - a_2 x(m-2) + e(m) \tag{3.4}$$

where the choice of the parameters $a1$ and $a2$ will determine the centre frequency and the bandwidth of the process.

3.1.2 Stochastic and Random Processes

A random process is any process or function that generates random signals. The term 'stochastic process' is broadly used to describe a random process that generates sequential random signals such as a sequence of speech samples, a video sequence, a sequence of noise samples, a sequence of stock market data fluctuations or a DNA sequence.

In signal processing terminology, a random or stochastic process is also a probability model of a class of random signals, e.g. Gaussian process, Markov process, Poisson process, binomial process, multinomial process, etc.

In this chapter, we are mainly concerned with discrete-time random processes that may occur naturally or may be obtained by sampling a continuous-time band-limited random process. The term 'discrete-time stochastic process refers to a class of discrete-time random signals, $x(m)$ characterised by a probabilistic model. Each realisation of a discrete-time stochastic process $x(m)$ may be indexed in time and space as $x(m,s)$, where m is the discrete time index, and s is an integer variable that designates a space index to each realisation of the process.

3.1.3 The Space of Variations of a Random Process

The collection of all realisations of a random process is known as the ensemble, or the space, of the process. For an illustration, consider a random noise process over a telecommunication network as shown in Figure 3.2.

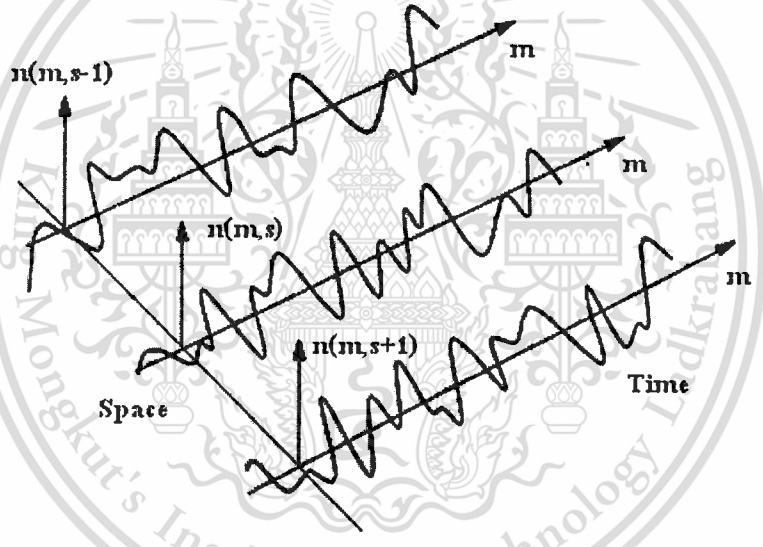


Figure 3.2 Illustration of three different realisations in the space of a random noise process $N(m)$.

The noise on each telephone line fluctuates randomly with time, and may be denoted as $n(m,s)$, where m is the discrete time index and s denotes the line index. The collection of noise on different lines form the ensemble (or the space) of the noise process denoted by $N(m) = \{n(m,s)\}$, where $n(m,s)$ denotes a realisation of the noise process $N(m)$ on the line s. The "true" statistics of a random process are obtained from the averages taken over the ensemble of many different realisations of the process. However, in many practical cases, only one or a finite number of realisation of a process are available. In Section 3.3, we consider the ergodic

random processes in which time-averaged statistics, from a single realisation of a process, may be used instead of the ensemble-averaged statistics.

Notation The following notation is used in this chapter: $X(m)$ denotes a random process, the signal $x(m,s)$ is a particular realisation of the process $X(m)$, the random signal $x(m)$ is any realisation of $X(m)$, and the collection of all realisations of $X(m)$, denoted by $x(m,s)$ forms the ensemble or the space of the random process $X(m)$.

3.2 Probability Models of Random Signals

Probability models, devised initially to calculate the odds for the different outcomes in a game of chance, provide a complete mathematical description of the distribution of the likelihood of the different outcomes of a random process. In its simplest form a probability model provides a numerical value, between 0 and 1, for the likelihood of a discrete-valued random variable assuming a particular state or value. The probability of an outcome of a variable should reflect the fraction of times that the outcome is observed to occur.

3.2.1 Random Variables and Random Processes

At this point it is useful to define the difference between a random variable and a random process. A random variable is a variable that assumes random values such as the outcomes of a chance game or the values of a speech sample or an image pixel or the outcome of a sport match. A random process, such as a Markov process, generates random variables usually as functions of time and space. Also a time or space series, such as a sequence of speech or an image is often called a random process.

Consider a random process that generates a time-sequence of numbers $x(m)$. Let $x(m,s)$ denote a collection of different time-sequences generated by the same process where m denotes time and s is the sequence index for example as illustrated in Figure 3.2. For a given time instant m, the sample realisation of a random process $x(m,s)$ is a random variable that takes on various values across the space s of the process. The main difference between a random variable and a random process is that the latter generates random time/space series. Therefore the probability models used for random variables are also applied to random processes. We continue this section with the definitions of the probability functions for a random variable.

3.2.2 Probability Mass Function (pmf)

For a discrete random variable X that can only assume discrete values from a finite set of N numbers $\{x_1, x_2, ..., x_N\}$, each outcome x_i may be considered as an event and assigned a probability of occurrence. The probability that a discrete-valued random variable X takes on a value of x_i, $P(X = x_i)$, is called the probability mass function (*pmf*). For two such random variables X and Y, the probability of an outcome in which X takes on a value of x_i and Y takes on a value of y_i, $P(X = x_i, Y = y_i)$, is called the joint probability mass function. The joint *pmf* can be described in terms of the conditional and the marginal probability mass functions as

$$P_{X,Y}(x_i, y_i) = P_{Y|X}(y_i/x_i)P_X(x_i)$$

$$= P_{X|Y}(x_i/y_i)P_Y(y_i) \tag{3.5}$$

where $P_{Y|X}(y_i/x_i)$ is the probability of the random variable Y taking on a value of y_i conditioned on X having taken a value of x_i, and the so-called marginal pmf of X is obtained as

$$P_X(x_i) = \sum_{j=1}^{M} P_{X,Y}(x_i/y_j)$$

$$= \sum_{j=1}^{M} P_{X|Y}(x_i/y_j)p_Y(y_i) \tag{3.6}$$

where M is the number of values or outcomes, in the space of the discrete random variable *Y*. From Equations (3.5) and (3.6), we have Bayes' rule for the conditional probability mass function, given by

$$P_{X,Y}(x_i/y_i) = \frac{1}{P_Y(y_j)} P_{Y|X}(y_i/x_i)P_X(x_i)$$

$$= \frac{P_{Y|X}(y_i/x_i)P_X(x_i)}{\sum_{i=1}^{M} P_{Y|X}(y_i/x_i)P_X(x_i)} \tag{3.7}$$

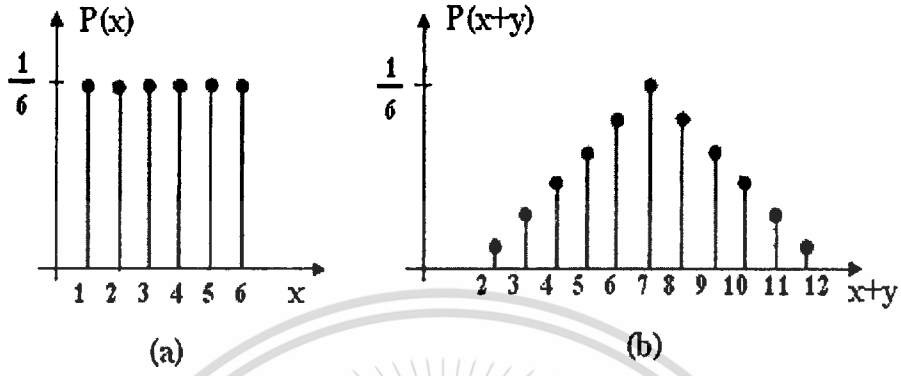


Figure 3.3 The probability mass function (*pmf*) of
(a) a die, and (b) the sum of a pair of dice.

Figure 3.3 (a) shows the *pmf* of a die. Now, let the variables (x, y) represent the outcomes of throwing a pair of dice. The probability that the sum of the outcomes of throwing two dice is equal to A, is given by

$$P(x+y=A) = \sum_{i=1}^{6} P(x=i)P(y=A-i) \tag{3.8}$$

The *pmf* of the sum of two dice is plotted in Figure 3.3 (a). Note from Equation (3.8) that the probability of the sum of two random variables is the convolution sum of the probability functions of the individual variables. Note that from the central limit theorem, the sum of many independent random variables will have a normal distribution.

3.2.3 Probability Density Function (*pdf*)

Now consider a continuous-valued random variable. A continuous-valued variable can assume an infinite number of values, and hence, the probability that it takes on a given value vanishes to zero. For a continuous-valued random variable X the cumulative distribution function (*cdf*) is defined as the probability that the outcome is less than x as:

$$F_X(x) = \Pr ob(X \le x) \tag{3.9}$$

where $\Pr ob(\bullet)$ denotes probability. The probability that a random variable X takes on a value within a band of Δ centred on x can be expressed as

$$\frac{1}{\Delta}\Pr ob(x - \Delta/2 \le X \le x + \Delta/2) = \frac{1}{\Delta}\left[\Pr ob(X \le x + \Delta/2) - \Pr ob(X \le x - \Delta/2)\right]$$

$$= \frac{1}{\Delta}\left[F_X(X \le x + \Delta/2) - F_X(X \le x - \Delta/2)\right] \tag{3.10}$$

As Δ tends to zero we obtain the probability density function (pdf) as

$$f_X(x) = \lim_{\Delta \to 0}\frac{1}{\Delta}\left[F_X(x + \Delta/2) - F_X(x - \Delta/2)\right]$$

$$= \frac{\partial F_X(x)}{\partial x} \tag{3.11}$$

Since $F_X(x)$ increases with x, the pdf of x, which is the rate of change of $F_X(x)$ with x, is a non-negative-valued function; i.e. $f_X(x) \ge 0$. The integral of the pdf of a random variable X in the range $\pm\infty$ is unity:

$$\int_{-\infty}^{+\infty} f_X(x)dx = 1 \tag{3.12}$$

The conditional and marginal probability functions and the Bayes' rule, of Equations (3.5) – (3.7), also apply to probability density functions of continuous-valued variables. Figure 3.4 shows the cumulative density function (cdf) and probability density function of a Gaussian variable.

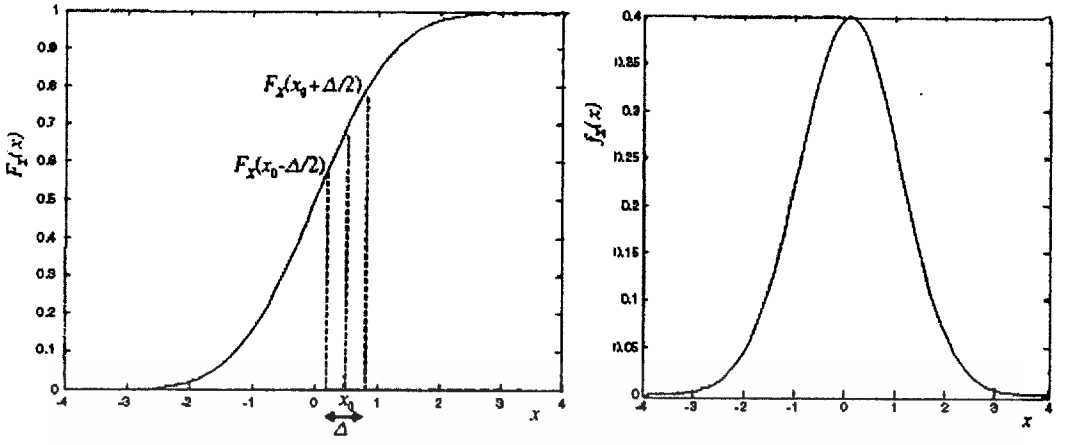


Figure 3.4 The cumulative density function (cdf) $F_X(x)$ and probability density function $f_X(x)$ of a Gaussian variable.

3.3 Stationary and Non-Stationary Random Processes

Although the amplitude of a signal $x(m)$ fluctuates with time m, the characteristics of the process that generates the signal may be time-invariant (stationary) or time-varying (non-stationary). An example of a nonstationary process is speech, whose loudness and spectral composition changes continuously as the speaker generates various sounds.

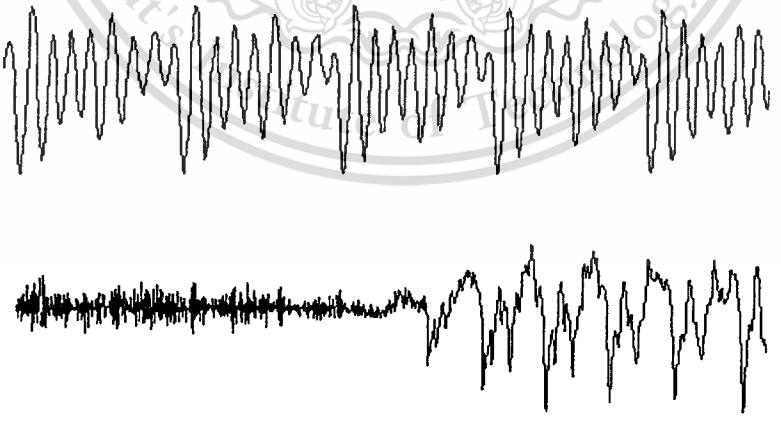


Figure 3.5 Examples of a quasi-stationary and a non-stationary speech segment.

A process is stationary if the parameters of the probability model of the process are time invariant; otherwise it is non-stationary (Figure 3.5). The stationarity property implies that all the parameters, such as the mean, the variance, the power spectral composition and the higher-order moments of the process, are time-invariant. In practice, there are various degrees of stationarity: it may be that one set of the statistics of a process is stationary, whereas another set is time-varying. For example, a random process may have a time-invariant mean, but a time-varying power.

Although many signals are non-stationary, the concept of a stationary process has played an important role in the development of signal processing methods. Furthermore, even non-stationary signals such as speech can often be considered as approximately stationary for a short period of time. In signal processing theory, two classes of stationary processes are defined: (a) strict-sense stationary processes and (b) wide-sense stationary processes, which is a less strict form of stationarity, in that it only requires that the first-order and second-order statistics of the process should be time-invariant.

3.3.1 Strict-Sense Stationary Processes

A random process $X(m)$ is stationary in a strict sense if all its distributions and statistical parameters are time-invariant. Strict-sense stationarity implies that the nth order distribution is translation-invariant for all $n = 1,2,3,....$:

$$\Pr ob[x(m_1) \le x_1, x(m_2) \le x_2, \ldots, x(m_n) \le x_n]$$

$$= \Pr ob[x(m_1 + \tau) \le x_1, x(m_2 + \tau) \le x_2, \ldots, x(m_n + \tau) \le x_n] \quad (3.13)$$

From Equation (3.13) the statistics of a strict-sense stationary process including the mean, the correlation and the power spectrum, are time invariant; therefore we have

$$E[x(m)] = \mu_x \quad (3.14)$$

$$E[x(m)x(m+k)] = r_{xx}(k) \quad (3.15)$$

and

$$E[|x(f,m)|^2] = E[|x(f)|^2] = P_{xx}(f) \quad (3.16)$$

where $\mu_x, r_{xx}(m)$ and $P_{xx}(f)$ are the mean value, the autocorrelation and the power spectrum of the signal $x(m)$ respectively, and $X(f,m)$ denotes the frequency–time spectrum of $x(m)$.

3.3.2 Wide-Sense Stationary Processes

The strict-sense stationarity condition requires that all statistics of the process should be time-invariant. A less restrictive form of a stationary process is so-called wide-sense stationarity. A process is said to be wide-sense stationary if the mean and the autocorrelation functions of the process are time invariant:

$$E[x(m)] = \mu_x \tag{3.17}$$

$$E[x(m)x(m+k)] = r_{xx}(k) \tag{3.18}$$

From the definitions of strict-sense and wide-sense stationary processes, it is clear that a strict-sense stationary process is also wide-sense stationary, whereas the reverse is not necessarily true.

3.3.3 Non-Stationary Random Processes

A random process is non-stationary if its distributions or statistics vary with time. Most stochastic processes such as video signals, audio signals, financial data, meteorological data, biomedical signals, etc., are nonstationary, because they are generated by systems whose environments and parameters vary over time. For example, speech is a non-stationary process generated by a time-varying articulatory system. The loudness and the frequency composition of speech changes over time, and sometimes the change can be quite abrupt. Time-varying processes may be modelled by a combination of stationary random models as illustrated in Figure 3.6. In Figure 3.6 (a) a non-stationary process is modelled as the output of a time-varying system whose parameters are controlled by a stationary process. In Figure 3.6 (b) a time-varying process is modelled by a chain of time-invariant states, with each state having a different set of statistics or probability distributions.

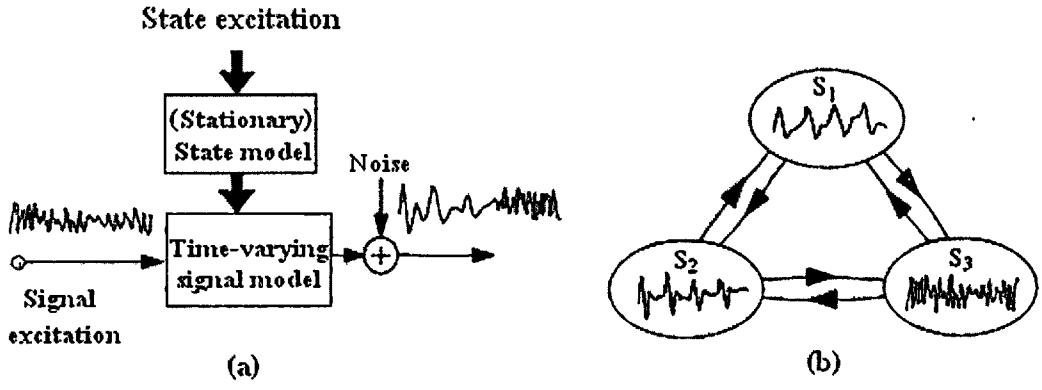


Figure 3.6 Two models for non-stationary processes:

(a) a stationary process drives the parameters of a continuously time varying model

(b) a finite-state model with each state having a different set of statistics.

3.4 Expected Values of a Random Process

The expected values of a random process, also known as its statistics or moments, are the mean, variance, correlation, power spectrum and the higher-order moments of the process.

Expected values play an indispensable role in signal processing. Furthermore, the probability models of a random process are usually expressed as functions of the expected values. For example, a Gaussian pdf is defined as an exponential function centred about the mean and with its volume and orientation determined by the covariance of the process, and a Poisson pdf is defined in terms of the mean of the process.

In signal processing applications, we often use our prior experience or the available data or perhaps sour intuitive feeling to select a suitable statistical model for a process, e.g. a Gaussian or Poisson pdf. To complete the model we need to specify the parameters of the model which are usually the expected values of the process such as its mean and covariance. Furthermore, for many algorithms, such as noise reduction filters or linear prediction, what we need essentially is an estimate of the mean or the correlation function of the process.

The expected value of a function $h(X)$ of random process X, $h(X(m1),X(m2), \ldots, X(mM))$, is defined as

$$E[h(X(m_1),...,X(m_M))] = \int_{-\infty}^{\infty}...\int_{-\infty}^{\infty} h(x_1,...,x_M) f_{X(m_1)...X(m_M)}(x_1,...,x_M)dx_1...dx_M \tag{3.19}$$

The most important, and widely used, expected values are the mean value, the correlation, the covariance, and the power spectrum.

3.4.1 The Mean Value

The mean value of a process plays an important part in signal processing and parameter estimation from noisy observations. In this chapter it is shown that the optimal linear estimate of a signal from a noisy observation, is an interpolation between the mean value and the observed value of the noisy signal. The mean value of a random vector $[X(m_1),...,X(m_M)]$ is its average value across the ensemble of the process defined as

$$E[X(m_1),...,X(m_M)] = \int_{-\infty}^{\infty}...\int_{-\infty}^{\infty}(x_1,...,x_M) f_{X(m_1)...X(m_M)}(x_1,...,x_M)dx_1...dx_M \tag{3.20}$$

For a segment of N samples of a signal $x(m)$, an estimate of the mean value of the segment is obtained as

$$\hat{\mu}_x = \frac{1}{N}\sum_{m=0}^{N-1} x(m) \tag{3.21}$$

Note that the estimate of the mean $\hat{\mu}_x$ in Equation (3.21), from a finite number of N samples, is itself a random variable with its own mean value, variance and probability distribution.

3.4.2 Autocorrelation

The correlation function and its Fourier transform, the power spectral density, are used in modeling and identification of patterns and structures in a signal process. Correlators play a central role in signal processing and telecommunication systems, including predictive coders, equalizers, digital decoders, delay estimators, classifiers and signal restoration systems. The autocorrelation function of a random process $X(m)$, denoted by $r_{xx}(m_1, m_2)$, is defined as

$$r_{xx}(m_1, m_2) = E[x(m_1)x(m_2)]$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(m_1)x(m_2) f_{X(m_1),X(m_1)}(x(m_1), x(m_2))dx(m_1)dx(m_2) \quad (3.22)$$

The autocorrelation function $r_{xx}(m_1, m_2)$ is a measure of the similarity, or the mutual relation, of the outcomes of the process X at time instants m_1 and m_2. If the outcome of a random process at time m_1 bears no relation to that at time m_2 then $X(m_1)$ and $X(m_2)$ are said to be independent or uncorrelated and $r_{xx}(m_1, m_2) = 0$. For a wide-sense stationary process, the autocorrelation function is time-invariant and depends on the time difference $m = m_1 - m_2$:

$$r_{xx}(m_1 + \tau, m_2 + \tau) = r_{xx}(m_1, m_2) = r_{xx}(m_1 - m_2) = r_{xx}(m) \quad (3.23)$$

The autocorrelation function of a real-valued wide-sense stationary process is a symmetric function with the following properties:

$$r_{xx}(-m_1) = r_{xx}(m) \quad (3.24)$$

$$r_{xx}(m_1) \le r_{xx}(0) \quad (3.25)$$

Note that for a zero-mean signal, $r_{xx}(0)$ is the signal power.

3.4.3 Autocovariance

The autocovariance function $c_{xx}(m_1, m_2)$ of a random process $X(m)$ is measure of the scatter, or the dispersion, of the random process about the mean value, and is defined as

$$c_{xx}(m_1, m_2) = E[(x(m_1) - \mu_x(m_1))(x(m_2) - \mu_x(m_2))]$$

$$= r_{xx}(m_1, m_2) - \mu_x(m_1)\mu_x(m_2) \quad (3.26)$$

where $\mu_x(m)$ is the mean of $X(m)$. Note that for a zero-mean process the autocorrelation and the autocovariance functions are identical. Note also that $c_{xx}(m_1, m_2)$ is the variance of the process. For a stationary process the autocovariance function of Equation (3.26) becomes

$$c_{xx}(m_1, m_2) = c_{xx}(m_1 - m_2) = r_{xx}(m_1 - m_2) - \mu_x^2 \tag{3.27}$$

3.4.4 Cross-Correlation and Cross-Covariance

The cross-correlation of two random process $x(m)$ and $y(m)$ is defined as

$$r_{xy}(m_1, m_2) = E[x(m_1)y(m_2)]$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(m_1)y(m_2) f_{X(m_1),Y(m_2)}(x(m_1), y(m_2))dx(m_1)dy(m_2) \tag{3.28}$$

For wide-sense stationary processes, the cross-correlation function $r_{xy}(m_1, m_2)$ depends only on the time difference $m = m_1 - m_2$:

$$r_{xy}(m_1 + \tau, m_2 + \tau) = r_{xy}(m_1, m_2) = r_{xy}(m_1 - m_2) = r_{xy}(m) \tag{3.29}$$

The cross-covariance function is defined as

$$c_{xy}(m_1, m_2) = E[(x(m_1) - \mu_x(m_1))(y(m_2) - \mu_y(m_2))]$$

$$= r_{xy}(m_1, m_2) - \mu_x(m_1)\mu_y(m_2) \tag{3.30}$$

Note that for zero-mean processes, the cross-correlation and the crosscovariance functions are identical. For a wide-sense stationary process the cross-covariance function of Equation (3.30) becomes

$$c_{xy}(m_1, m_2) = c_{xy}(m_1 - m_2) = r_{xy}m_1 - m_2) - \mu_x\mu_y \tag{3.31}$$

For example: Time-delay estimation. Consider two signals $y_1(m)$ and $y_2(m)$, each composed of an information bearing signal $x(m)$ and an additive noise, given by

$$y_1(m) = x(m) + n_1(m) \tag{3.32}$$

$$y_2(m) = Ax(m-D) + n_2(m) \tag{3.33}$$

where A is an amplitude factor and D is a time delay variable. The crosscorrelation of the signals $y_1(m)$ and $y_2(m)$ yields

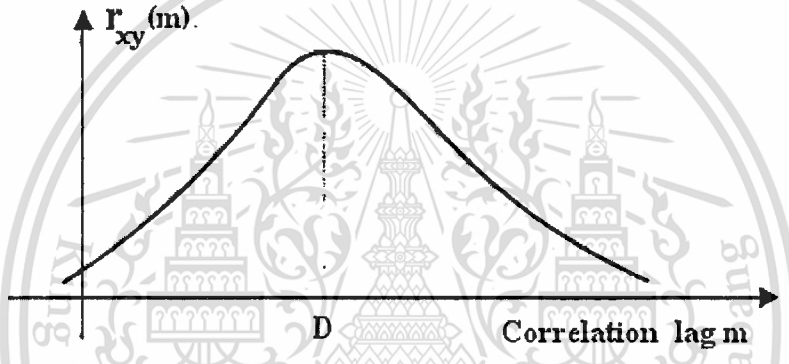


Figure 3.7 The peak of the cross-correlation of two delayed signals can be used to estimate the time delay D.

$$r_{xy}(k) = E[y_1(m)y_2(m+k)]$$

$$= E\{[x(m)+n_1(m)][Ax(m-D+k)+n_2(m+k)]\}$$

$$= Ar_{xx}(k-D) + r_{xn_2}(k) + Ar_{xn_1}(k-D) + r_{n_1n_2}(k) \tag{3.34}$$

Assuming that the signal and noise are uncorrelated, we have $r_{y_1y_2}(k) = Ar_{xx}(k-D)$ As shown in Figure 3.7, the cross-correlation function has its maximum at the lag D.

3.5 Some Useful Classes of Random Processes

In this section, we consider some important classes of random processes extensively used in signal processing applications for the modelling of signals and noise.

3.5.1 Gaussian Process (Normal)

The Gaussian process, also called the normal process, is perhaps the most widely applied of all probability models. Some advantages of Gaussian probability models are the following:

(a) Gaussian pdfs can model the distribution of many processes including some important classes of signals and noise.

(b) Non-Gaussian processes can be approximated by a weighted combination (i.e. a mixture) of a number of Gaussian pdfs of appropriate means and variances.

(c) Optimal estimation methods based on Gaussian models often result in linear and mathematically tractable solutions.

(d) The sum of many independent random processes has a Gaussian distribution. This is known as the central limit theorem

A scalar Gaussian random variable is described by the following probability density function:

$$F_X(x) = \frac{1}{\sigma_x\sqrt{2\pi}} \exp\left[-\frac{(x-\mu_x)^2}{2\sigma_x^2}\right] \tag{3.35}$$

where μ_x and σ_x^2 are the mean and the variance of the random variable x, and exp denotes the exponential function. The Gaussian process of Equation (3.35) is also denoted by $N(x, \mu_x, \sigma_x^2)$ The maximum of a Gaussian pdf occurs at the mean μ_x, and is given by

$$f_X(\mu_x) = \frac{1}{\sigma_x\sqrt{2\pi}} \tag{3.36}$$

where σ_x is the standard deviation (a measure of the "width" of the bell), For a mean of 0 and a standard deviation of 1, this formula simplifies to

$$F_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{3.37}$$

The probability density function has notable properties including:

- The graph of the associated probability density function is bell-shaped, with a peak at the mean, and is known as the Gaussian function or bell curve
- symmetry about its mean μ_x
- they are defined by two parameter: mean (μ_x) and standard deviation (σ_x)
- the inflection points of the curve occur one standard deviation away from the mean, i.e. at $\mu_x - \sigma_x$ and $\mu_x + \sigma_x$

From Equation (3.35), the Gaussian *pdf* of x decreases exponentially with the increasing distance of x from the mean value. The distribution function F(x) is given by

$$F_X(x) = \frac{1}{\sigma_x \sqrt{2\pi}} \int_{-\infty}^{x} \exp\left[-\frac{(x-\mu_x)^2}{2\sigma_x^2} \right] dx \tag{3.38}$$

The standard normal *cdf* is just the general *cdf* evaluated with mean $(\mu_x = 0)$ and standard deviation $(\sigma_x = 1)$

$$F_X(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{x^2}{2} \right) dx \tag{3.39}$$

The standard normal *cdf* can be expressed in terms of a special function called the error function, as

$$F_X(x) = \frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}} \right) \right] \tag{3.40}$$

and the *cdf* itself can hence be expressed as

$$F_X(x) = \frac{1}{2}\left[1 + erf\left(\frac{x-\mu_x}{\sigma_x \sqrt{2}} \right) \right] \tag{3.41}$$

Figure 3.8 shows the bell-shaped *pdf* and the *cdf* of a Gaussian model. The most probable values of a Gaussian process happen around the mean value and the probability of a value decreases exponentially with the increasing distance from the mean value. The total area under the *pdf* curve is one. Note that the area under the *pdf* curve one standard deviation on each side of the mean value $(\mu \pm \sigma)$ is 0.682, the area two standard deviations on each side of the mean value $(\mu \pm 2\sigma)$ is 0.955 and the area three standard deviations on each side of the mean value $(\mu \pm 3\sigma)$ is 0.997.

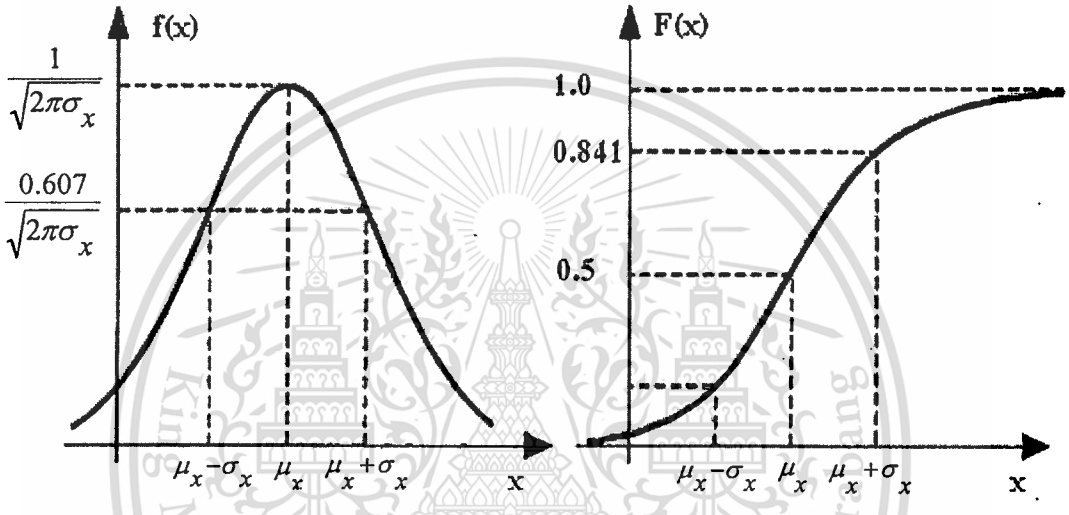


Figure 3.8 Gaussian probability density and cumulative density functions.

3.5.2 Binary-State Gaussian Process

Consider a random process $x(m)$ with two statistical states: such that in the state s_0 the process has a Gaussian pdf with mean $\mu_{x,0}$ and variance $\sigma^2_{x,0}$ and in the state s_1 the process is also Gaussian with mean $\mu_{x,1}$ and variance $\sigma^2_{x,1}$. The state-dependent pdf of $x(m)$ can be expressed as

$$f_{X/S}(x(m)/S_i) = \frac{1}{\sigma_{x,i}\sqrt{2\pi}} \exp\left\{-\frac{1}{2\mu^2_{x,i}}\left[x(m)-\mu_{x,i}\right]^2\right\}, \quad i = 0,1 \qquad (3.42)$$

The joint probability distribution of the binary-valued state s_i and the continuous-valued signal $x(m)$ can be expressed as

$$f_{X/S}(x(m)/s_i) = f_{X/S}(x(m)/s_i)P_S(s_i)$$

$$= \frac{1}{\sigma_{x,i}\sqrt{2\pi}} \exp\left\{-\frac{1}{2\mu_{x,i}^2}\left[x(m)-\mu_{x,i}\right]^2\right\} P_S(s_i) \qquad (3.43)$$

where $P_S(s_i)$ is the state probability. For a multistate process we have the following probabilistic relations between the joint and marginal probabilities:

$$\sum_S f_{X,S}(x(m),s_i) = f_X(x(m)) \qquad (3.44)$$

$$\int_X f_{X,S}(x(m),s_i)dx = P_S(s_i) \qquad (3.45)$$

and

$$\sum_S \int_X f_{X,S}(x(m),s_i)dx = 1 \qquad (3.46)$$

Note that in a multistate model, the statistical parameters of the process switch between a number of different states, whereas in a single-state mixture *pdf*, a weighted combination of a number of pdfs models the process.

Chapter 4

Adaptive IIR Notch Filter

The design of a Wiener filter [4] requires a priori information about the statistics of the data to be processed. The filter is optimum only when the statistical characteristics of the input data match the a priori information on which the design of the filter is based. When this information is not known completely, however, it may not be possible to design the Wiener filter or else the design may no longer be optimum. A straightforward approach that we may use in such situations is the "estimate and plug" procedure. This is a two-stage process whereby the filter first "estimates" the statistical parameters of the relevant signals and then "plugs" the results so obtained into a nonrecursive formula for computing the filter parameters. For a real-time operation, this procedure has the disadvantage of requiring excessively elaborate and costly hardware. A more efficient method is to use an adaptive filter. By such a device we mean one that is self-designing in that the adaptive filter relies on a recursive algorithm for its operation, which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available. The algorithm starts from some predetermined set of initial conditions, representing whatever we know about the environment. Yet, in a stationary environment, we find that after successive iterations of the algorithm it converges to the optimum Wiener solution in some statistical sense. In a nonstationary environment, the algorithm offers a tracking capability, in that it can track time variations in the statistics of the input data, provided that the variations are sufficiently slow.

As a direct consequence of the application of a recursive algorithm whereby the parameters of an adaptive filter are updated from one iteration to the next, the parameters become data dependent. This, therefore, means that an adaptive filter is in reality a nonlinear device, in the sense that it does not obey the principle of superposition. Notwithstanding this property, adaptive filters are commonly classified as linear or nonlinear. An adaptive filter is said to be linear if the estimate of quantity of interest is computed adaptively (at the output of the filter) as a linear combination of the available set of observations applied to the filter input. Otherwise, the adaptive filter is said to be nonlinear.

A wide variety of recursive algorithms have been developed in the literature of the operation of linear adaptive filters. In the final analysis, the choice of one algorithm over another is determined by one or more of the following factors [4]:

- **Rate of Convergence**

This is defined as the number of iterations required for the algorithm, in response to stationary inputs, to converge "close enough" to the optimum Wiener solution in the mean-square sense. A fast rate of convergence allows the algorithm to adapt rapidly to a stationary environment of unknown statistics.

- **Misadjustment**

For an algorithm of interest, this parameter provides a quantitative measure of the amount by which the final value of the mean-squared error, averaged over an ensemble of adaptive filters, deviates from the minimum mean-squared error that is produced by the Wiener filter.

- **Tracking**

When an adaptive filtering algorithm operates in a nonstationary environment, the algorithm is required to track statistical variations in the environment. The tracking performance of the algorithm, however, is influenced by two contradictory features: (1) the rate of convergence and (2) the steady-state fluctuation due to algorithm noise.

- **Robustness**

For an adaptive filter to be robust, small disturbances (i.e., disturbances with small energy) can only result in small estimation errors. The disturbances may arise from a variety of factors internal or external to the filter.

- **Computational Requirements**

Here, the issues of concern include (1) the number of operations (i.e., multiplications, divisions, and additions/subtractions) required to make one complete iteration of the algorithm, (2) the size of memory locations required to store the data and the program, and (3) the investment required to program the algorithm on a computer.

- **Structure**

This refers to the structure of information flow in the algorithm, determining the manner in which it is implemented in hardware form. For example, an algorithm whose structure exhibits

high modularity, parallelism, or concurrency is well suited for implementation using very large-scale integration (VLSI).

- **Numerical Properties**

When an algorithm is implemented numerically, inaccuracies are produced due to quantization errors. The quantization errors are due to analog-to-digital conversion of the input data and digital representation of internal calculations. Ordinarily, it is the latter source of quantization errors that poses a serious design problem. In particular, there are two basic issues of concern: numerical stability and numerical accuracy. Numerical stability is an inherent characteristic of an adaptive filtering algorithm. Numerical accuracy, on the other hand, is determined by the number of bits (i.e., binary digits used in the numerical representation of data samples and filter coefficients). An adaptive filtering algorithm is said to be numerically robust when it is insensitive to variations in the word length used in its digital implementation.

These factors, in their own ways, also enter in to the design of nonlinear adaptive filters, except for the fact that we now no longer have a well-defined frame of reference in the form of a Wiener filter. Rather, we speak of a nonlinear filtering algorithm that may converge to a local minimum or, hopefully, a global minimum on the error performance surface.

There are many types of algorithm for adaptive IIR notch filter developed in the literature [4], [8]. For adaptive algorithm analysis, the best adaptive algorithm requires fast convergence rate, low variance, unbiased and low steady-state mean square error (MSE).

4.1 Adaptive IIR Notch Filter

Adaptive IIR notch filters (ANF) are very useful in various signal processing applications such as communications, active noise control, radar systems, sonar systems, biomedical engineering and others [1], [9]-[20]. The common feature of these applications which brings them under the same basic formulation of adaptive filtering is that they all involve a process of filtering some input signal to match a desired response. The main advantage of using this filter is that it has less computation complexities than their finite impulse response (FIR) counterparts as only one coefficient is adapted.

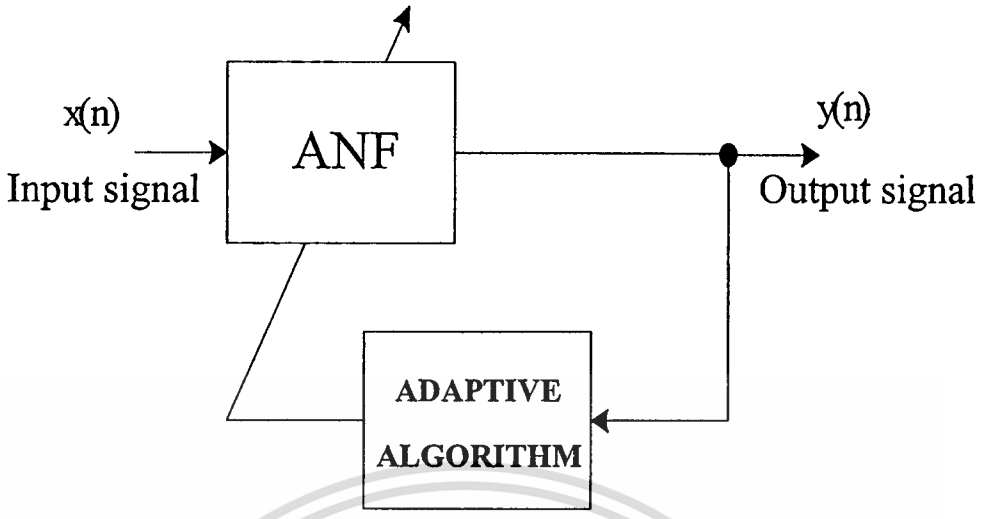


Figure 4.1 A block diagram of system identification using adaptive filtering.

In Figure 4.1 shows a block diagram of system identification using adaptive filtering. The certain situations a primary input is available consisting of a signal component with an additive undesired sinusoidal interference. The conventional method of eliminating such interference is through the use of a notch filter.

- The objective is to change (adapt) the coefficients of an IIR filter to match as closely as possible the response of an unknown system.
- The adaptive filter will change its coefficients in an attempt to reduce the error.

The transfer function of the second-order adaptive IIR notch filter with direct form structure [14] from input to notch output is expressed as:

$$H(z) = \frac{1 + a(n)z^{-1} + z^{-2}}{1 + \rho a(n)z^{-1} + \rho^2 z^{-2}} \tag{4.1}$$

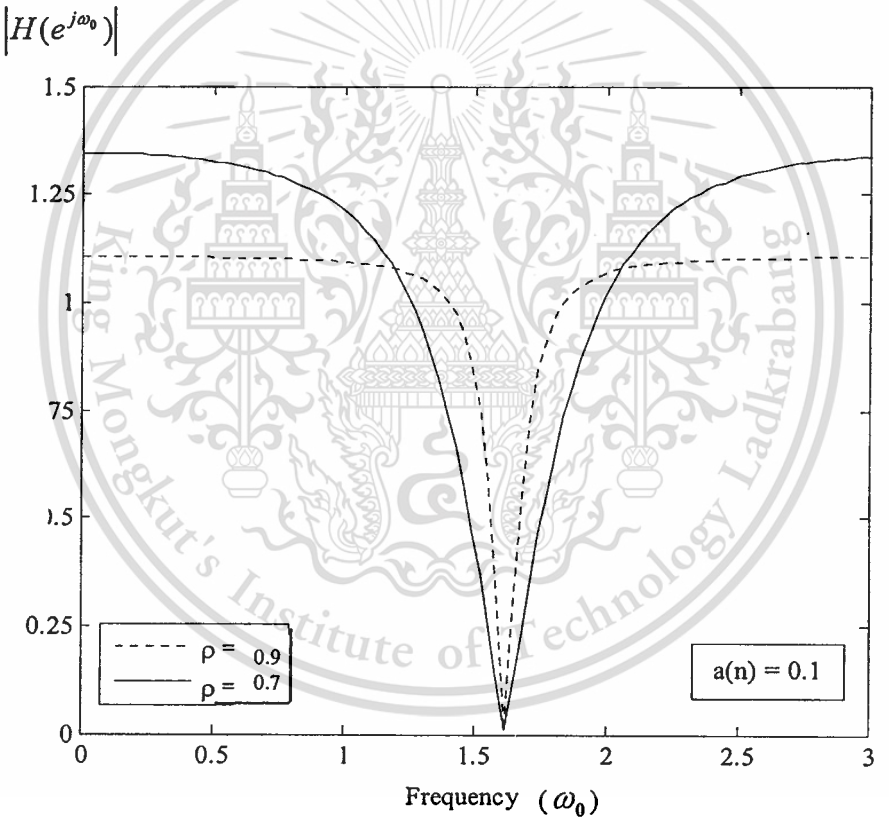
where ρ is the pole contraction factor whose value is interval $(0 < \rho < 1)$, which uses to control the notch bandwidth of the filter. The larger parameter ρ, the narrower notch bandwidth is obtained, and $a(n)$ is the filter coefficient will be estimated by an adaptive algorithm is

between $(-2 < a(n) < 2)$, which should converge to $-2\cos(\omega)$ to reject an unknown frequency of signal sinusoid signal whose frequency ω_0, and ω_0 can be defined as Equation (4.2)

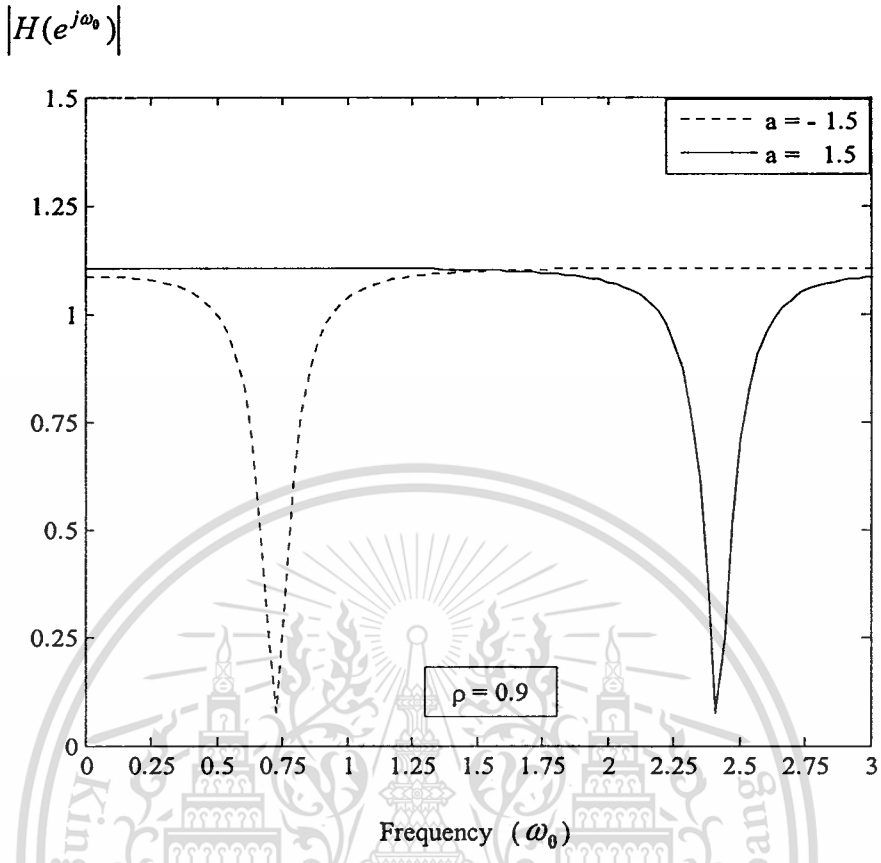
$$\omega_0 = \cos^{-1}(-a/2) \tag{4.2}$$

The magnitude response $\left|H(e^{j\omega_0})\right|$ can be given as

$$\left|H(e^{j\omega_0})\right| = \sqrt{\frac{(a(n)+2\cos(\omega_0))^2}{(\rho a(n)+(1+\rho^2)\cos(\omega_0))^2+((1-\rho^2)\sin(\omega_0))^2}} \tag{4.3}$$



(a)



(b)

Figure 4.2 Frequency response.

Figure 4.2 shows the frequency response of the second-order IIR notch filter with direct form structure by Equation (4.1), where the parameters employed in Figure 4.2 (a) and Figure 4.2 (b) are $a(n) = 0.1$, $\rho = 0.7$, $\rho = 0.9$ and $\rho = 0.9$, $a(n) = -1.5$, $a(n) = 1.5$, respectively.

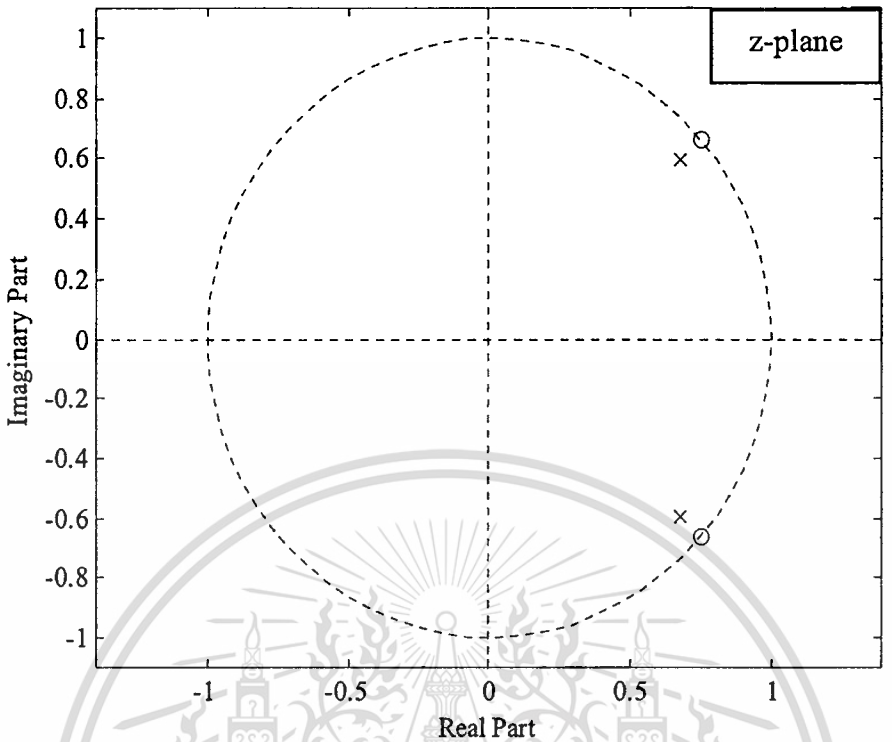


Figure 4.3 Pole–zero plots.

Figure 4.3 shows the pole–zero plots in z–plane of the transfer function $H(z)$ in Equation (4.1) when $a = -1.5$ and $\rho = 0.9$.

The output of the second-order adaptive IIR notch filter can be expressed as

$$y(n) = x(n) + a(n)x(n-1) + x(n-2) - a(n)\rho y(n-1) - \rho^2 y(n-2) \qquad (4.4)$$

and $g(n)$ is the gradient signal can be expressed as

$$g(n) = \frac{\partial y(n)}{\partial a(n)} = x(n-1) - \rho y(n-1) \qquad (4.5)$$

4.2 Method of Steepest Descent [8]

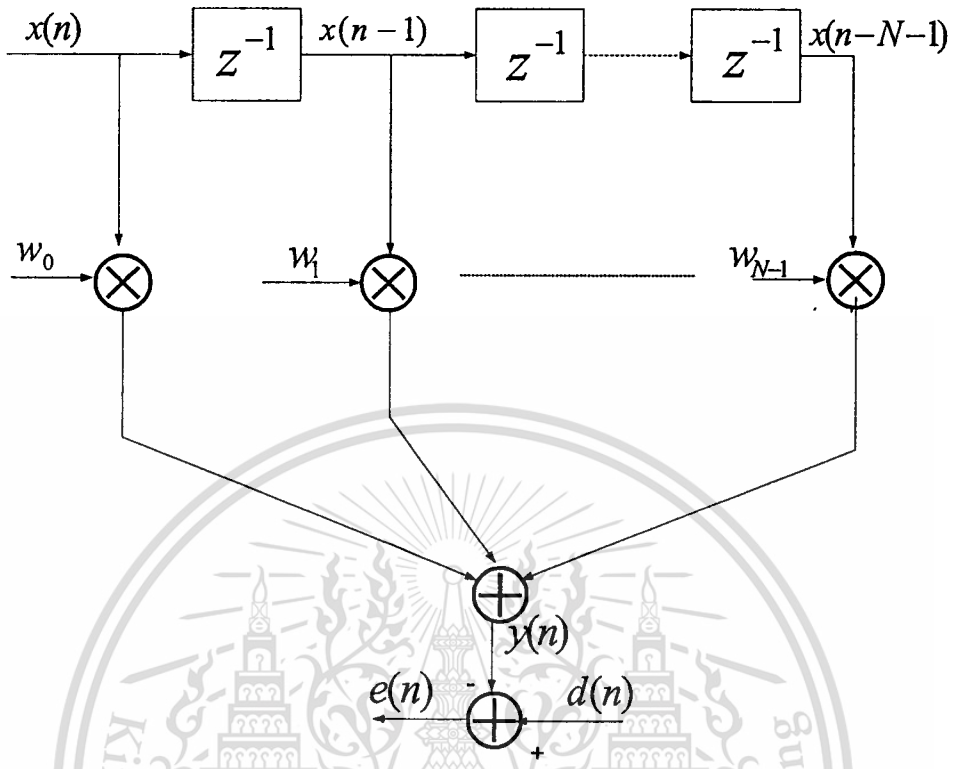


Figure 4.4 A transversal filter.

Consider a transversal Wiener filter in Figure 4.4. The filter input, $x(n)$, and its desired output $d(n)$, are assumed to be real-values stationary processes. The filter tap weights, $w_0, w_1, w_3, ..., w_{N-1}$, are also assumed to be real-value. The filter input and tep-weight vectors are defined, respectively, as the column vectors

$$\mathbf{w} = [w_0, w_1, w_3, ..., w_{N-1}]^T \tag{4.6}$$

and

$$\mathbf{x}(n) = [x(n) \quad x(n-1) \ \ldots \ x(n-N+1)]^T \tag{4.7}$$

where the superscript T stands for transpose.

The filter output is

$$y(n) = \sum_{i=0}^{N-1} w_i x(n-i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n) \tag{4.8}$$

which can be also be written as

$$y(n) = \mathbf{x}^{\mathrm{T}}(n)\mathbf{w} \tag{4.9}$$

Since $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n)$ is a scalar and thus it is equal to its transpose, Example: $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n) = (\mathbf{w}^{\mathrm{T}}\mathbf{x}(n))^{\mathrm{T}}$ $= \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}$. Thus, we may write

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= d(n) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(n) \\ &= d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w} \end{aligned} \tag{4.10}$$

The objective of function is

$$\xi = E[e^2(n)] \tag{4.11}$$

Using Equation (4.10) in Equation (4.11) we get

$$\xi = E[e^2(n)] = E[(d(n) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(n))(d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w})] \tag{4.12}$$

Expanding the right hand side of Equation (4.12) and nothing that $\mathbf{w}$ can be shifted out of the expectation operator, $E[\bullet]$, since it is not a statistical variable, we obtain

$$\xi = E[d^2(n)] - \mathbf{w}^{\mathrm{T}}E[\mathbf{x}(n)d(n)] - E[d(n)\mathbf{x}^{\mathrm{T}}(n)]\mathbf{w} + \mathbf{w}^{\mathrm{T}}E[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\mathbf{w} \tag{4.13}$$

Next, if we define the $Nx1$ cross-correlation vector

$$\mathbf{p} = E[\mathbf{x}(n)d(n)] = [p_0, p_1, p_3, ..., p_{N-1}]^T \tag{4.14}$$

and the NxN autocorrelation matrix

$$\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)] = \begin{bmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,N-1} \\ r_{1,0} & r_{1,1} & \cdots & r_{N-1,1} \\ \vdots & \vdots & \vdots & \vdots \\ r_{N-1,0} & r_{N-1,1} & \cdots & r_{N-1,N-1} \end{bmatrix} \tag{4.15}$$

and note that $E[d(n)\mathbf{x}^{\mathrm{T}}(n)] = \mathbf{p}^T$, and also $\mathbf{w}^{\mathrm{T}}\mathbf{p} = \mathbf{p}^{\mathrm{T}}\mathbf{w}$, we obtain

$$\xi = E[e^2(n)] = E[d^2(n)] - 2\mathbf{w}^{\mathrm{T}}\mathbf{p} + \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} \tag{4.16}$$

where ∇ is the gradient operator defined as the column vector

$$\nabla = \left[\frac{\partial}{\partial w_0} \quad \frac{\partial}{\partial w_1} \quad \cdots \quad \frac{\partial}{\partial w_{N-1}} \right]^T \tag{4.17}$$

so we get $\nabla\xi = \left[\frac{\partial E[e^2(n)]}{\partial w_0} \quad \frac{\partial E[e^2(n)]}{\partial w_1} \quad \cdots \quad \frac{\partial E[e^2(n)]}{\partial w_{N-1}} \right]^T$

$$\nabla\xi = 2\mathbf{R}\mathbf{w} - 2\mathbf{p} \tag{4.18}$$

where $\nabla\xi = 0$ gives the following equation from which the optimum set of Wiener filter weights can be obtained:

$$\mathbf{R}\mathbf{w}_0 = \mathbf{p} \quad \text{or} \quad \mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p} \tag{4.19}$$

The method of steepest descent can be find update value of tap-weight vector by

$$\mathbf{w}(n-1) = \mathbf{w}(n) - \mu\nabla_n\xi \tag{4.20}$$

where $\mathbf{w}(n)$ is the tap-weight vector at the n iteration

μ is the step-size parameter

$\nabla_n\xi$ denotes the gradient vector $\nabla\xi$ evaluated at the point $\mathbf{w} = \mathbf{w}(n)$

4.3 Least-Mean-Square Algorithm [8]

Tapped-delay line or transversal filter as the structural basis for implementing the linear adaptive filter. For the case of stationary inputs, the cost function, also referred to as the index of performance, is defined as the mean-squared error (i.e., the mean-square value of the difference between the desired response and the transversal filter output). This cost function is precisely a second-order function of the tap weights in the transversal filter. The dependence of the mean-squared error on the unknown tap weights may be viewed to be in the form of a multidimensional paraboloid (i.e., punch bowl) with a uniquely defined bottom or minimum point. As mentioned previously, we refer to this paraboloid as the error-performance surface; the tap weights corresponding to the minimum point of the surface define the optimum Wiener solution.

To develop a recursive algorithm for updating the tap weights of the adaptive transversal filter, we proceed in two stages. We first modify the system of Wiener-Hopf equations (i.e., the matrix equation defining the optimum Wiener solution) through the use of the method of steepest descent, a well-known technique in optimization theory. This modification requires the use of a gradient vector, the value of which depends on two parameters: the correlation matrix of the tap inputs in the transversal filter and the cross-correlation vector between the desired response and the same tap inputs. Next, we use instantaneous values for these correlations so as to derive an estimate for the gradient vector, making it assume a stochastic character in general. The resulting algorithm is widely known as the least-mean-square (LMS) algorithm, the essence of which may be described in words as follows for the case of a transversal filter operating on real-valued data:

$$\begin{pmatrix} updated\ value \\ of\ tap-weight \\ vactor\ \ vector \end{pmatrix} = \begin{pmatrix} old\ \ \ value \\ of\ tap-weight \\ vector \end{pmatrix} + \begin{pmatrix} learning \\ rate \\ parameter \end{pmatrix} \begin{pmatrix} tap \\ input \\ vector \end{pmatrix} \begin{pmatrix} error \\ signal \end{pmatrix}$$

where the error signal is defined as the difference between some desired response and the actual response of the transversal filter produced by the tap-input vector.

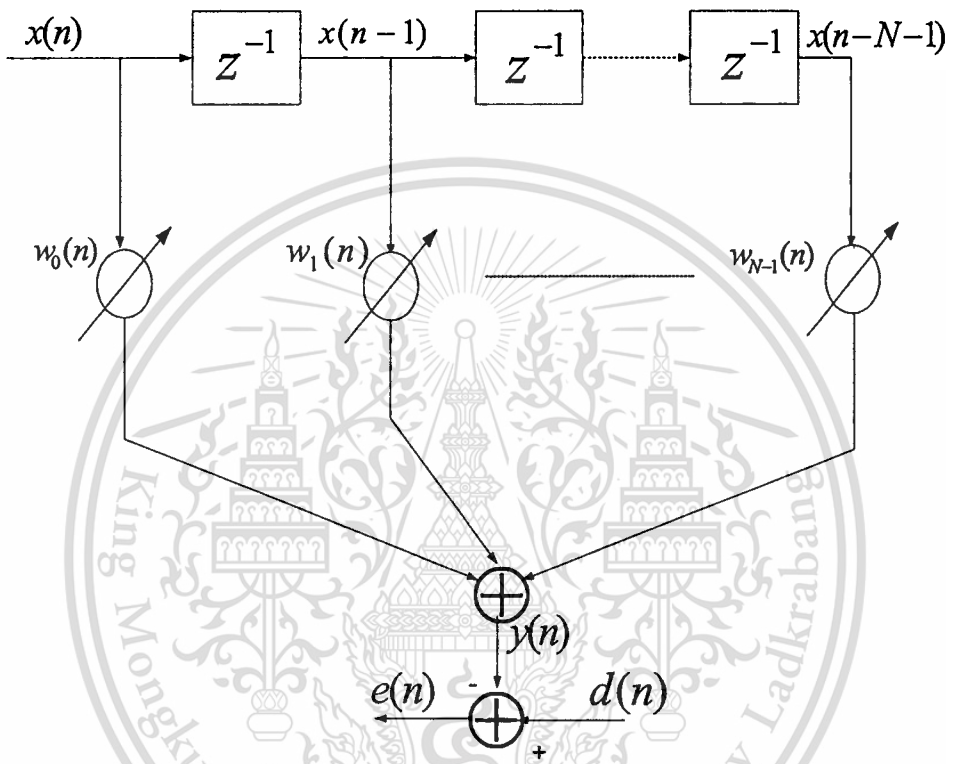


Figure 4.5 An N-tap transversal adaptive filter.

Figure 4.5 depicts an N-tap transversal adaptive filter. The filter input, $x(n)$, desired output, $d(n)$, and the filter output,

$$y(n) = \sum_{i=0}^{N-1} w_i(n)x(n-i) \tag{4.21}$$

The error signal $e(n)$ of the difference between the desired response $d(n)$ and the transversal filter output $y(n)$

$$e(n) = d(n) - y(n) \tag{4.22}$$

Is minimized in some sense. It may be noted that filter tap-weights are explicitly indicated to be functions of the time index n. This signifies that in an adaptive filter, in general, tap-weight are time varying, since they are continuously being adapted so that any variations in signal's statistics could be tracked. The LMS algorithm changes (adapts) the filter tap-weight so that $e(n)$ is minimized in the mean square sense, thus the name least mean square. When the process $x(n)$ and $d(n)$ are jointly stationary, this algorithm converges to a set of tap-weights which, on average, are equal to the Wiener Hopf. It is a sequential algorithm which can be used to adapt the tap-weights of a filter by continuous observation of its input, $x(n)$, and desired output, $d(n)$.

The conventional LMS algorithm is a stochastic implementation of the steepest-descent algorithm. It simply replaces the cost function $\xi = E[e^2(n)]$ by its instantaneous coarse estimate $\xi(n) = e^2(n)$. Substituting $\xi(n) = e^2(n)$ for ξ in the steepest-descent recursion (4.20), and the iteration index n, we obtain

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu\nabla e^2(n) \tag{4.23}$$

where $\mathbf{w}(n) = \left[w_0(n) \quad w_1(n) \quad \ldots \quad w_{N-1}(n)\right]^T$, μ is the algorithm step-size parameter and ∇ is the gradient operator defined as the column vector

$$\nabla = \left[\frac{\partial}{\partial w_0} \quad \frac{\partial}{\partial w_0} \quad \ldots \quad \frac{\partial}{\partial w_{N-1}}\right]^T \tag{4.24}$$

We note that the i element of the gradient vector $\nabla e^2(n)$ is

$$\frac{\partial e^2(n)}{\partial w_i} = 2e(n)\frac{\partial e(n)}{\partial w_i} \tag{4.25}$$

Substituting Equation (4.22) in the last factor on the right hand side of Equation (4.25) and noting that $d(n)$ is independent of w_i, we obtain

$$\frac{\partial e^2(n)}{\partial w_i} = -2e(n)\frac{\partial y(n)}{\partial w_i} \tag{4.26}$$

Substituting for $y(n)$ from Equation (4.21) we get

$$\frac{\partial e^2(n)}{\partial w_i} = -2e(n)x(n-i) \tag{4.27}$$

Using Equation (4.24) and Equation (4.27) we obtain

$$\nabla e^2(n) = -2e(n)\mathbf{x}(n) \tag{4.28}$$

where $x(n) = [x(n) \quad x(n-1) \quad \ldots \quad x(n-N+1)]^T$. Substituting this result in Equation (4.23) we get

$$\mathbf{w}(n-1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \tag{4.29}$$

The LMS algorithm, summarized in Table 4.1, is simple and yet capable of achieving satisfactory performance under the right conditions. Its major limitations are a relatively slow rate of convergence and a sensitivity to variations in the condition number of the correlation matrix of the tap inputs, the condition number of a Hermitian matrix is defined as the ratio of its largest eigenvalue to its smallest eigenvalue. Nevertheless, the LMS algorithm is highly popular and widely used in a variety of applications.

LMS algorithm	
Input	Tap-weight vector, $\mathbf{w}(n)$ Input vector, $\mathbf{x}(n)$ and desired output, $d(n)$.
Output	Filter output, $y(n)$ Tap-weight vector update, $\mathbf{w}(n+1)$
1. Filtering: $y(n) = \mathbf{w}^{\mathbf{T}}(n)\mathbf{x}(n)$ 2. Error estimation: $e(n) = d(n) - y(n)$ 3. Tap-weight vector adaptation: $\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$	

Table 4.1 Summary of the LMS algorithm.

4.4 Variable Step-Size LMS Algorithm [8]

The speed of convergence of the LMS algorithm changes in proportion to its step-size parameter may be required to minimize the transient time of the LMS algorithm. On the other hand, to achieve a small misadjustment a small step-size parameter has to be used. These are conflicting requirements and, thus, a compromise solution has to be adopted.

The variable step-size LMS (VSLMS) algorithm works on the basic of the sample heuristic that comes from the mechanism of the LMS algorithm. Each tap of the adaptive filter is given a separate time varying step-size parameter and the LMS recursion is written as

$$w_i(n+1) = w_i(n) + 2\mu_i(n)e(n)x(n-i), \quad \text{for } i = 0,1,\ldots,N-1 \tag{4.30}$$

where $w_i(n)$ is the i^{th} element of the tap-weight vector $\mathbf{w}(n)$ and $\mu_i(n)$ is its associated step-size parameter at iteration n. The adjustment of the step-size parameter $\mu_i(n)$ is done as follows. The corresponding stochastic gradient term $g_i(n) = e(n)x(n-i)$ is monitored over successive iterations of the algorithm and $\mu_i(n)$ is increased if the latter term consistently shows a positive or negative direction. This happens when the adaptive filter has not yet converged. As the adaptive filter tap weights converge to some vicinity of their optimum values, the averages of the stochastic gradient terms approach zero and hence they change signs more frequently. This is detected by the algorithm and the corresponding step-size parameters are gradually reduced to some minimum values. If the situation changes and the algorithm begins to hunt for a new optimum point, then the gradient terms will indicate consistent (positive or negative) directions, resulting in an increase in the corresponding step-size parameters. To ensure that the step-size parameters do not become too large (which may result in system instability) or too small (which may result in a slow reaction of the system to sudden changes), upper and lower limits should be specified for each step-size parameter.

Following the above argument, the VSLMS algorithm step-size parameters, the $\mu_i(n)$, may be adjusted using the following recursion:

$$\mu_i(n) = \mu_i(n-1) + \rho sign[g_i(n)]sign[g_i(n-1)] \tag{4.31}$$

where ρ is a small positive step-size parameter. The 'sign' function maybe dropped from Equation (4.31). This results in the following alternative step-size parameter update equation:

$$\mu_i(n) = \mu_i(n-1) + \rho g_i(n) g_i(n-1) \tag{4.32}$$

Both update Equations (4.31) and (4.32) work well in practice. Which of the two choices works better is application dependent. The choice of one over the other may also be decided on the basis of the available hardware or software platform on which the algorithm is to be implemented. For instance, if a digital signal processor is being used, then Equation (4.32) maybe much easier to implement. On the other hand, if a custom chip is to be designed, then the update Equation (4.31) may be preferred.

The set update Equation (4.30) may be written in vector form as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n) \tag{4.33}$$

VSLMS algorithm	
Input	Tap-weight vector, $\mathbf{w}(n)$ Input vector, $\mathbf{x}(n)$ Gradient terms $g_0(n-1), g_1(n-1), \ldots, g_{N-1}(n-1)$; Step-size parameters, $\mu_0(n-1), \mu_1(n-1), \ldots, \mu_{N-1}(n-1)$; and desired output, $d(n)$.
Output	Filter output, $y(n)$ Tap-weight vector update, $\mathbf{w}(n+1)$ Gradient terms $g_0(n), g_1(n), \ldots, g_{N-1}(n)$; and updated step-size parameters, $\mu_0(n), \mu_1(n), \ldots, \mu_{N-1}(n)$;
1. Filtering: $y(n) = \mathbf{w}^T(n)\mathbf{x}(n)$ 2. Error estimation: $e(n) = d(n) - y(n)$	

3. Tap-weight and step-size parameters adaptation:

For $i = 0,1,\ldots,N-1$

$g_i(n) = e(n)x(n-i)$

$\mu_i(n) = \mu_i(n-1) + \rho sign[g_i(n)]sign[g_i(n-1)]$

if $\mu_i(n) > \mu_{max}, \mu_i(n) = \mu_{max}$

if $\mu_i(n) < \mu_{min}, \mu_i(n) = \mu_{min}$

$w_i(n+1) = w_i(n) + 2\mu_i(n)g_i(n)$

end

Table 4.2 Summary of an implementation of variable step-size LMS algorithm.

4.5 Interesting Algorithms

4.5.1 The Previous Algorithm [10]

The adaptive algorithm to control the update filter coefficient is described as follows [10]:

$$a(n+1) = a(n) - \mu(n)y(n)\operatorname{sgn}\left(\frac{g(n)}{1+|g(n)|}\right) \tag{4.34}$$

where $\mu(n)$ is the variable step-size parameter.

$y(n)$ is the output signal.

and $g(n)$ is gradient signal.

The variable step-size update equation used to control notch filter coefficient is expressed as follows:

$$\mu(n+1) = \alpha(n)\mu(n) + \gamma p^2(n) \tag{4.35}$$

where $\alpha(n)$ is the estimate of the error control signal and $p(n)$ is the estimate of the output control signal, respectively.

The estimate of the error control signal is given by

$$\alpha(n) = \varepsilon\alpha(n-1) + (1-\varepsilon)e(n)e(n-1) \tag{4.36}$$

$$e(n) = x(n) - y(n) \tag{4.37}$$

and the estimate of the output control signal is given by

$$p(n) = \sigma p(n-1) + (1-\sigma)y^2(n) \tag{4.38}$$

where γ, ε and σ are positive constants. Its value is interval $[0,1]$

4.5.2 The Variable Step-Size Algorithm 1

In this section, the new variable step-size adaptive algorithm 1 (VSS 1) for adaptive noise canceller is proposed. The goal is to improve the performance of the system.

The adaptive algorithm for a second order adaptive IIR notch filter is given by

$$a(n+1) = a(n) - \mu(n)y(n)\operatorname{sgn}\left(\frac{g(n)}{1+|g(n)|}\right) \tag{4.39}$$

where $\mu(n)$ is the variable step-size parameter.

Herein, the new variable step-size parameter $\mu(n)$ is given as

$$\mu(n+1) = \alpha(n)\mu(n) + \gamma p^2(n)y^2(n) \tag{4.40}$$

where $\alpha(n)$ is the new estimate of the autocorrelation of the error control signal, $p(n)$ is the estimate of the autocorrelation of the output signal, and $y(n)$ is the output signal.

The Equation (4.40) is a modified version of Equation (4.35).

There are two objectives of the new variable step-size algorithm. First, we use the time-averaged estimate of the autocorrelation of output signals to control the adaptation process of the step function $\mu(n)$ for fast convergence speed, which can be expressed as

$$p(n) = \sigma p(n-1) + (1-\sigma)y(n)y(n-1) \tag{4.42}$$

At the beginning, the value of $p^2(n)$ has high value making $\mu(n)$ has high and fast convergence rate. After the adaptive algorithm converges, the value of $p^2(n)$ is lowest value made $\mu(n)$ has the lowest value too.

Second, the goal is to reject the effect of uncorrelated noise sequence on the step-size update to ensuring low misadjustment. We use the new autocorrelation of the error signal $e(n)e(n-1)$ and the mean square error (MSE) $|e^2(n)|$ of the error correlation to accommodate an effective adjusting of $\mu(n)$ when the solution is far from the optimum. The step-size update $\mu(n)$ is decreasing as the system approaches the optimum even in the presence of noise. Therefore, the new time varying estimate of autocorrelation of the error signal is given by

$$\alpha(n) = \varepsilon\alpha(n-1) + (1-\varepsilon)\{e(n)[e(n-1) + e(n)]\} \tag{4.43}$$

or

$$\alpha(n) = \varepsilon\alpha(n-1) + (1-\varepsilon)\{e(n)e(n-1) + e^2(n)\} \tag{4.44}$$

The parameters $y(n)$, $g(n)$ and $e(n)$ are using the Equations (4.4), (4.5), and (4.37), respectively.

4.5.3 The Variable Step-Size Algorithm 2

The new variable step-size algorithm 2 (VSS 2) for adaptive noise canceller is proposed. The VSS 2 is using the autocorrelation of the output signal and the power of the output signal to control the adaptation process of the step–size function $\mu(n)$ for fast convergence speed, which can be expressed as

$$\mu(n+1) = \alpha(n)\mu(n) + \gamma p^2(n)m^2(n) \tag{4.45}$$

$$p(n) = \sigma p(n-1) + (1-\sigma)y(n)y(n-1) \tag{4.46}$$

$$m(n) = \beta m(n-1) + (1-\beta)\{y(n)[y(n-1) + y(n)]\} \tag{4.47}$$

where $\alpha(n)$ is the estimate of the autocorrelation of the error control signal, $p(n)$ is the estimate of the output control signal, and $m(n)$ is the autocorrelation of the output signal, respectively.

The parameters $y(n)$, $g(n)$, $e(n)$, $a(n+1)$, and $\alpha(n)$ are using the Equations (4.4), (4.5), (4.37), (4.39), and (4.43), respectively.

4.5.4 The Variable Notch Bandwidth Algorithm

In this section, the new variable notch bandwidth algorithm (VNB) for the adaptive noise canceller is proposed. We use the variable notch bandwidth technique can further improve the performance of the ANF system. Then, the time-varying notch bandwidth is adjusted to be wide when the algorithm is far from the optimum. As the algorithm approaches the optimum, the notch bandwidth is automatically adjusted to be narrow. The transfer function of Equation (4.29) is rewritten as

$$H(z) = \frac{1 + a(n)z^{-1} + z^{-2}}{1 + \rho(n)a(n)z^{-1} + \rho^2(n)z^{-2}} \tag{4.48}$$

where $a(n)$ is the filter coefficient.

$\rho(n)$ is the time-varying notch bandwidth.

The output signal of variable notch bandwidth algorithm can be expressed in time domain as follows:

$$y(n) = x(n) + a(n)x(n-1) + x(n-2) - a(n)\rho(n)y(n-1) - \rho^2(n)y(n-2) \tag{4.49}$$

The update equation for the time-varying notch bandwidth is expressed as

$$\rho(n+1) = \rho(n) - \mu(n)\,\mathrm{sgn}[e(n)g_\rho(n)] \tag{4.50}$$

and the gradient signal $g_\rho(n)$ is given by

$$g_\rho(n) = \frac{\partial y(n)}{\partial \rho(n)} = -a(n)y(n-1) - 2\rho(n)y(n-2) \tag{4.51}$$

The parameters $g(n)$, $e(n)$, $a(n+1)$, $\alpha(n)$, $\mu(n+1)$, $p(n)$, and $m(n)$ are using the Equations (4.5), (4.37), (4.39), (4.43), (4.45), (4.46) and (4.47), respectively.

❖ The Summary Equation

1. The Previous Algorithm [10]

$$y(n) = x(n) + a(n)x(n-1) + x(n-2) - a(n)\rho y(n-1) - \rho^2 y(n-2)$$

$$g(n) = \frac{\partial y(n)}{\partial a(n)} = x(n-1) - \rho y(n-1)$$

$$a(n+1) = a(n) - \mu(n)y(n)\operatorname{sgn}\left(\frac{g(n)}{1+|g(n)|}\right)$$

$$\mu(n+1) = \alpha(n)\mu(n) + \gamma p^2(n)$$

$$p(n) = \sigma p(n-1) + (1-\sigma)y^2(n)$$

$$\alpha(n) = \varepsilon\alpha(n-1) + (1-\varepsilon)e(n)e(n-1)$$

$$e(n) = x(n) - y(n)$$

2. The Variable Step-Size Algorithm 1

$$y(n) = x(n) + a(n)x(n-1) + x(n-2) - a(n)\rho y(n-1) - \rho^2 y(n-2)$$

$$g(n) = \frac{\partial y(n)}{\partial a(n)} = x(n-1) - \rho y(n-1)$$

$$a(n+1) = a(n) - \mu(n)y(n)\operatorname{sgn}\left(\frac{g(n)}{1+|g(n)|}\right)$$

- $\mu(n+1) = \alpha(n)\mu(n) + \gamma p^2(n)y^2(n)$

- $p(n) = \sigma p(n-1) + (1-\sigma)y(n)y(n-1)$

- $\alpha(n) = \varepsilon\alpha(n-1) + (1-\varepsilon)\{e(n)[e(n-1)+e(n)]\}$

  $e(n) = x(n) - y(n)$

3. The Variable Step-Size Algorithm 2

$$y(n) = x(n) + a(n)x(n-1) + x(n-2) - a(n)\rho y(n-1) - \rho^2 y(n-2)$$

$$g(n) = \frac{\partial y(n)}{\partial a(n)} = x(n-1) - \rho y(n-1)$$

$$a(n+1) = a(n) - \mu(n)y(n)\operatorname{sgn}\left(\frac{g(n)}{1+|g(n)|}\right)$$

- $\mu(n+1) = \alpha(n)\mu(n) + \gamma p^2(n)m^2(n)$
- $p(n) = \sigma p(n-1) + (1-\sigma)y(n)y(n-1)$
  - $m(n) = \beta m(n-1) + (1-\beta)\{y(n)[y(n-1)+y(n)]\}$
- $\alpha(n) = \varepsilon\alpha(n-1) + (1-\varepsilon)\{e(n)[e(n-1)+e(n)]\}$

  $e(n) = x(n) - y(n)$

4. The Variable Notch Bandwidth Algorithm

$$y(n) = x(n) + a(n)x(n-1) + x(n-2) - a(n)\rho(n)y(n-1) - \rho^2(n)y(n-2)$$

$$g(n) = \frac{\partial y(n)}{\partial a(n)} = x(n-1) - \rho y(n-1)$$

$$a(n+1) = a(n) - \mu(n)y(n)\operatorname{sgn}\left(\frac{g(n)}{1+|g(n)|}\right)$$

- $g_\rho(n) = \frac{\partial y(n)}{\partial \rho(n)} = -a(n)y(n-1) - 2\rho(n)y(n-2)$
- $\rho(n+1) = \rho(n) - \mu(n)\operatorname{sgn}[e(n)g_\rho(n)]$

- $\mu(n+1) = \alpha(n)\mu(n) + \gamma p^2(n)m^2(n)$

- $p(n) = \sigma p(n-1) + (1-\sigma)y(n)y(n-1)$

  - $m(n) = \beta m(n-1) + (1-\beta)\{y(n)[y(n-1) + y(n)]\}$

- $\alpha(n) = \varepsilon\alpha(n-1) + (1-\varepsilon)\{e(n)[e(n-1) + e(n)]\}$

  $e(n) = x(n) - y(n)$

Note:

- The equation has changed
  - Additional equation

4.6 Computational Complexity

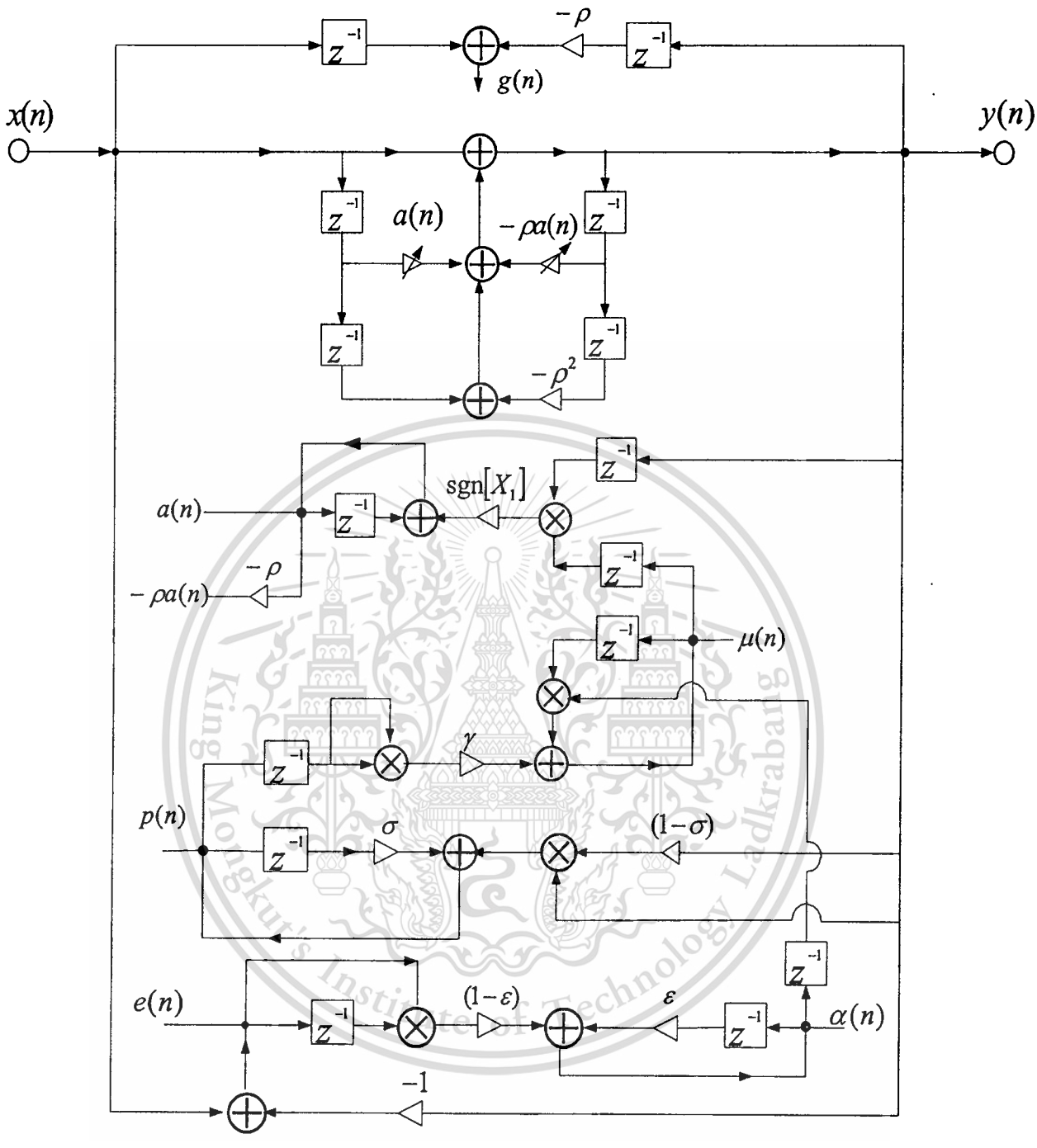
In this section, the computational complexity of the adaptive noise cancellers are compared.

Table 4.3 shows the comparison of computational complexity of previous algorithm [10] and proposed algorithms (VSS 1, VSS 2 and VNB). From Table 4.3, it is seen that the proposed algorithms provide higher computational complexity.

Algorithms	Summation	Multiplication	Total
The previous algorithm [10]	10	16	26
The VSS 1 algorithm	11	18	29
The VSS 2 algorithm	13	20	33
The VNB algorithm	15	23	38

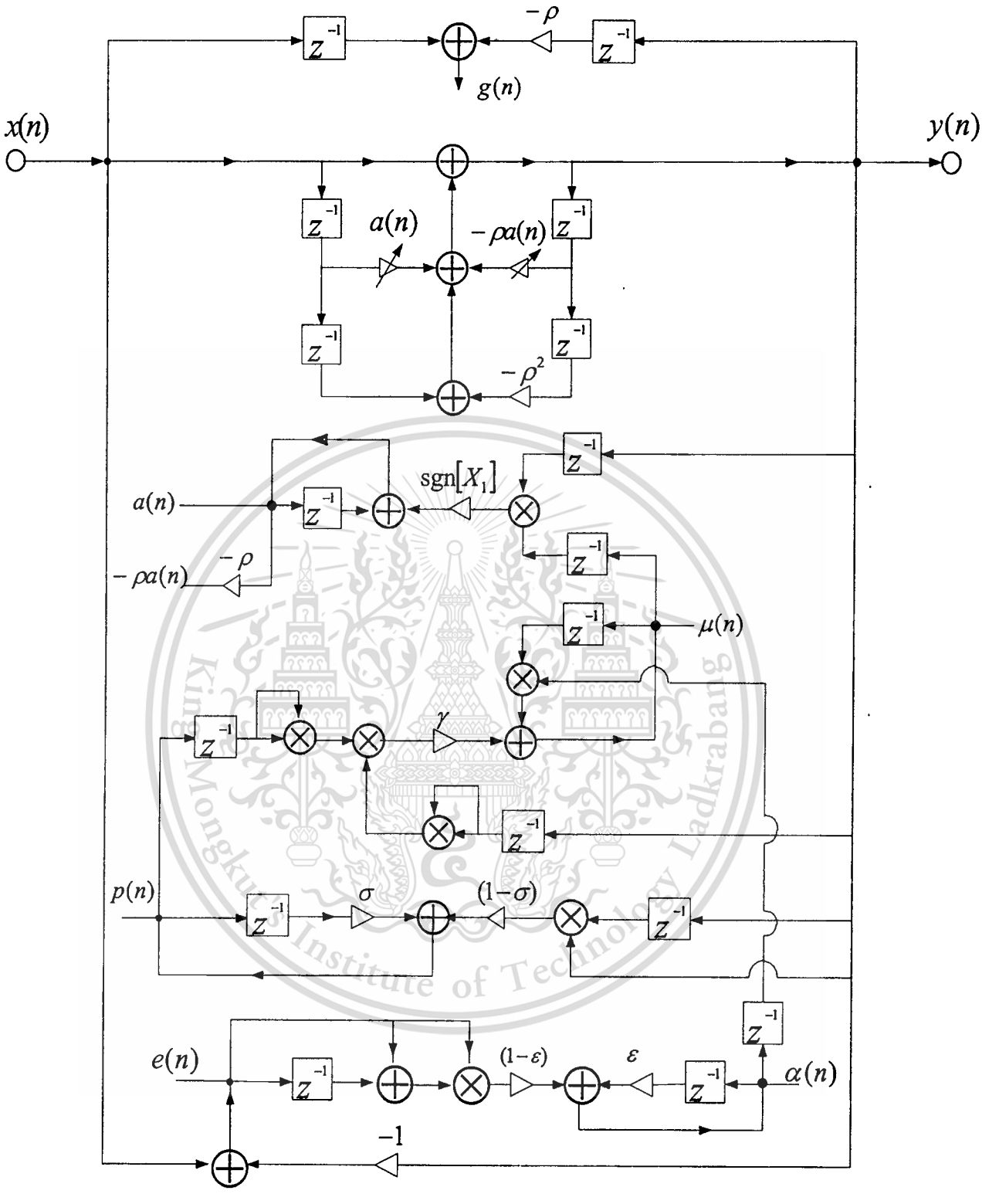
Table 4.3 Comparison of computational complexity of algorithms.

Figures 4.6-4.9 show a structure of adaptive noise canceller using the previous algorithm [10], the VSS 1 algorithm, the VSS 2 algorithm and the VNB algorithm, respectively.



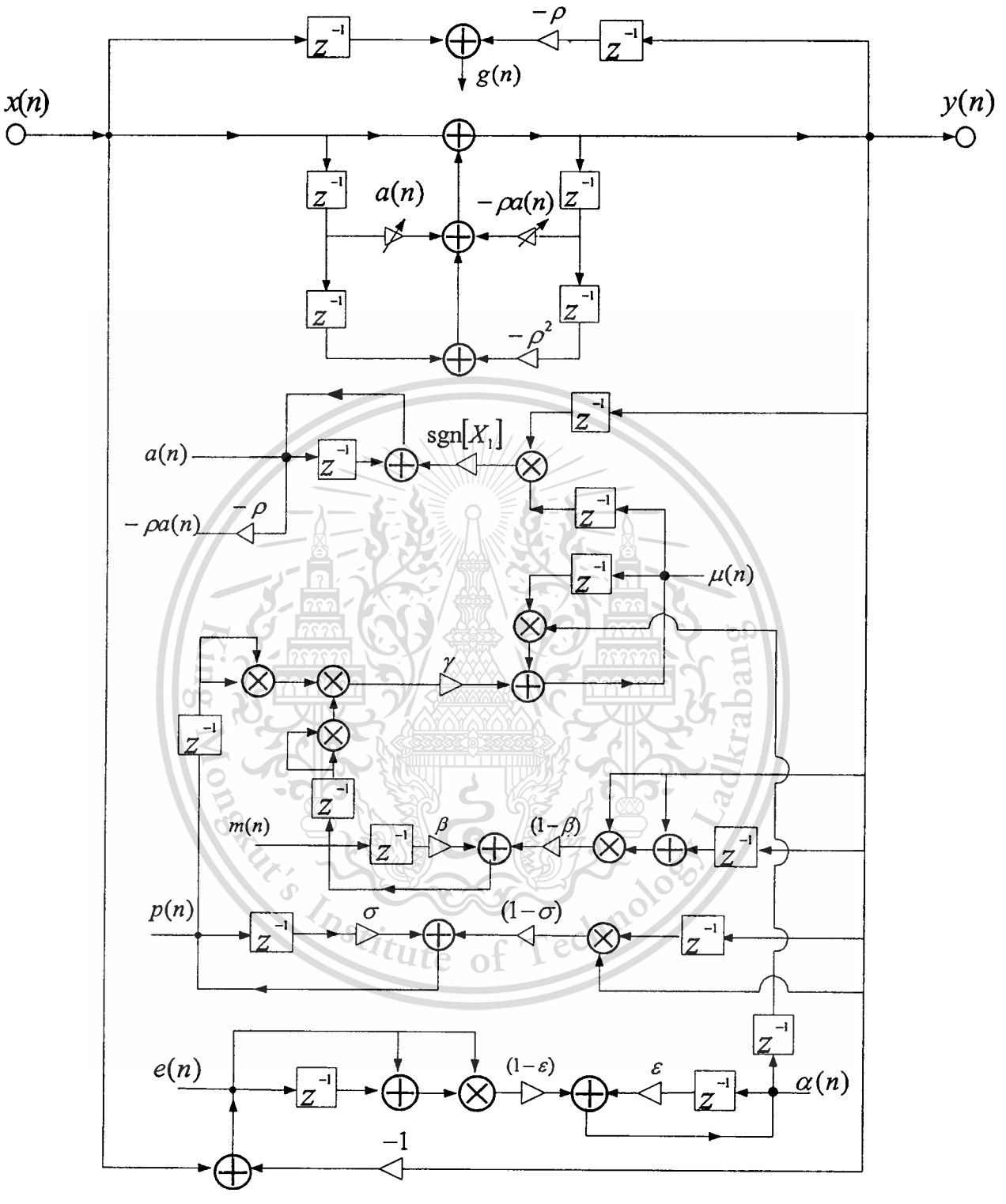
$$\text{where } X_1 = \frac{g(n)}{1-|g(n)|}$$

Figure 4.6 Adaptive noise canceller using the previous algorithm [10].



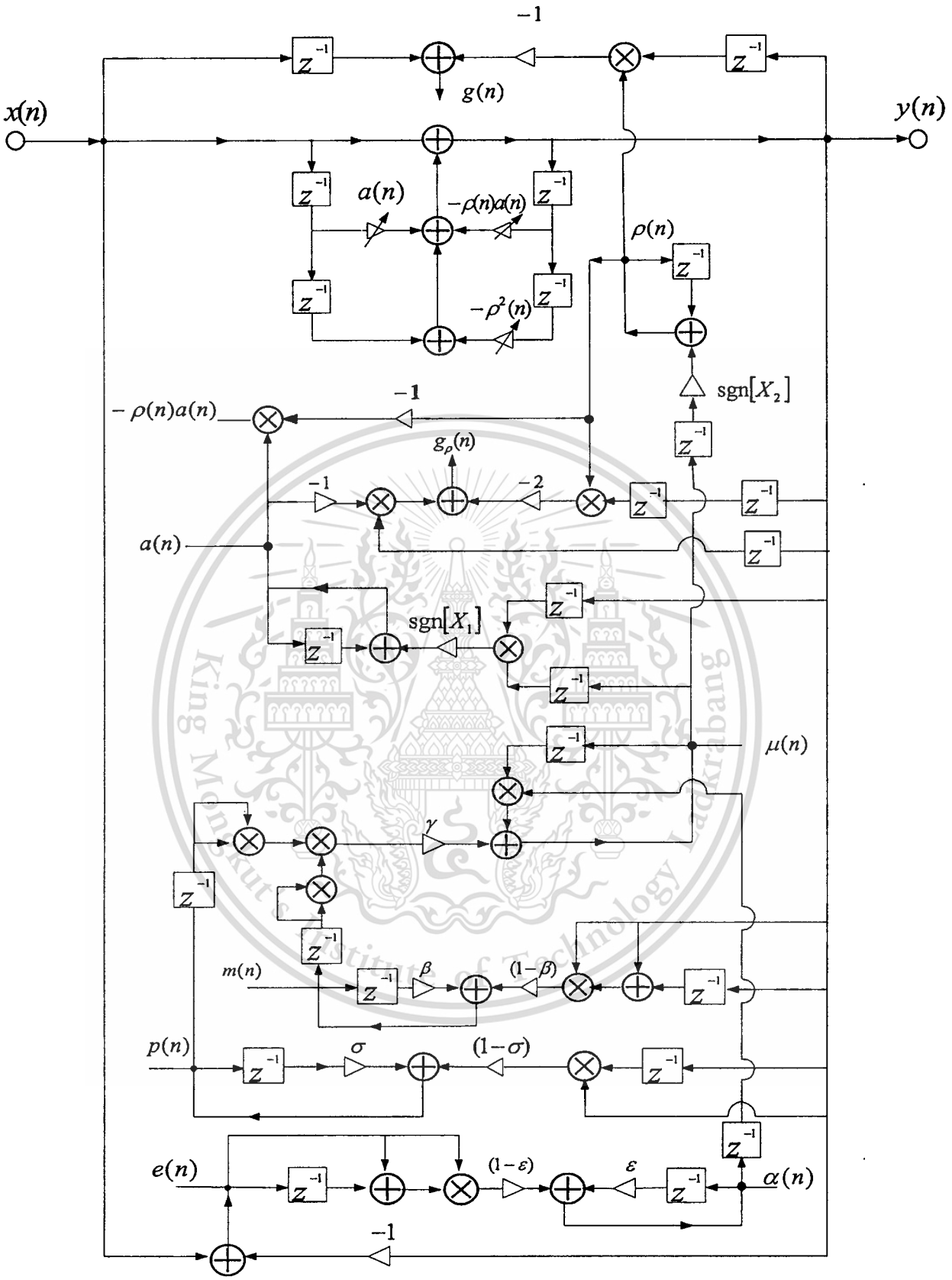
$$\text{where } X_1 = \frac{g(n)}{1-|g(n)|}$$

Figure 4.7 Adaptive noise canceller using the VSS 1 algorithm.



$$\text{where } X_1 = \frac{g(n)}{1-|g(n)|}$$

Figure 4.8 Adaptive noise canceller using the VSS 2 algorithm.



where $X_1 = \dfrac{g(n)}{1-|g(n)|}$ and $X_2 = e(n)g_\rho(n)$

Figure 4.9 Adaptive noise canceller using the VNB algorithm.

Chapter 5

Simulation Results

In this chapter, to verity the performance of the proposed adaptive noise canceller is used to estimate a single sinusoid signal in Gaussian noise environment. The computer simulation results of this application using the proposed algorithms (VSS 1, VSS 2 and VNB) are compared with the previous algorithm [10].

5.1 Sinusoidal Estimation in Gaussian Noise

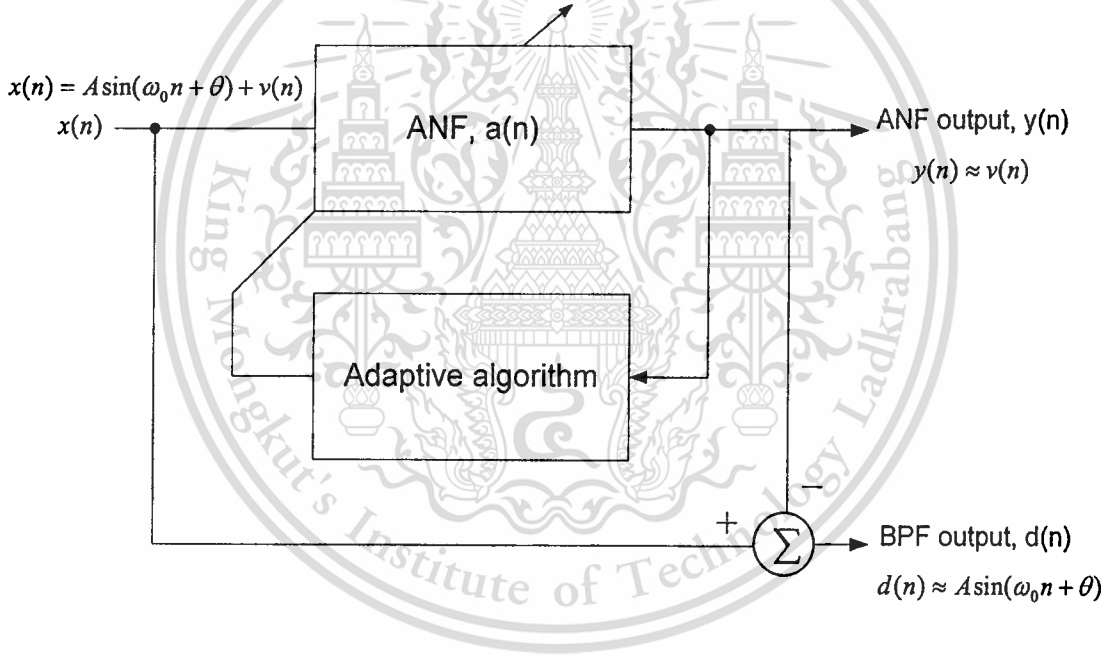


Figure 5.1 The block diagram for sinusoidal estimation by using ANF.

In Figure 5.1 shows the block diagram for sinusoidal estimation by using ANF. The input signal $x(n)$ is a single sinusoid disturbed by Gaussian noise $v(n)$ with zero mean and variance σ_v^2 that can be defined as follows:

$$x(n) = A\sin(\omega_0 n + \theta) + v(n) \tag{5.1}$$

where A is amplitude of sinusoid.

ω_0 is unknown frequency of digital signal.

n is time index.

θ is the initial phase which is uniformly distributed between 0 and 2π.

$v(n)$ is Gaussian noise.

$y(n)$ is output signal of adaptive notch filter (ANF).

and $d(n)$ is output signal of band-pass filter (BPF).

Figure 5.2 shows an example of adaptive noise cancelling using adaptive IIR band pass filter.

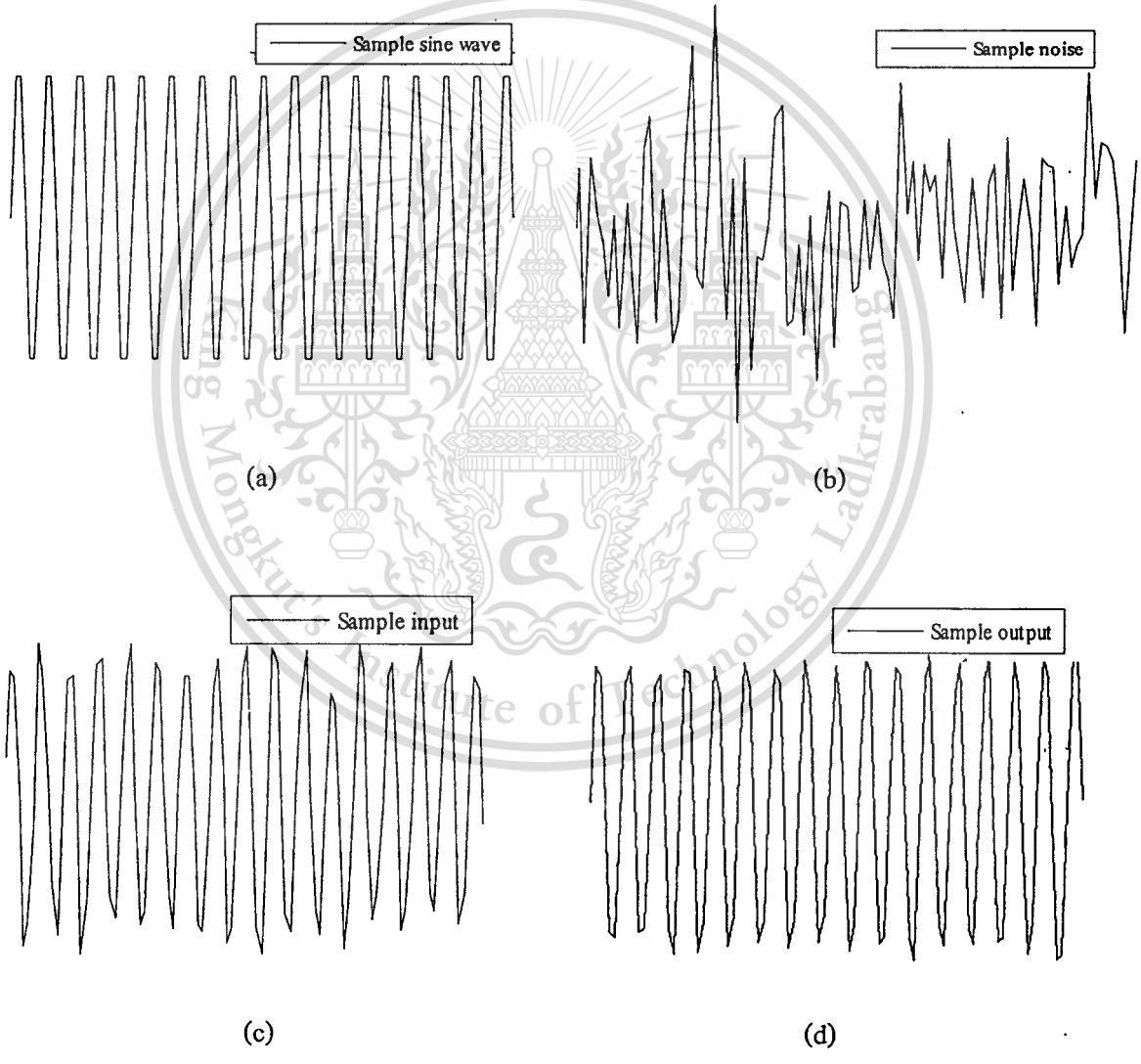


Figure 5.2 (a) Sinusoidal, (b) Gaussian noise, (c) Sinusoidal with noise (Input) and (d) Estimated of sinusoidal (Output)

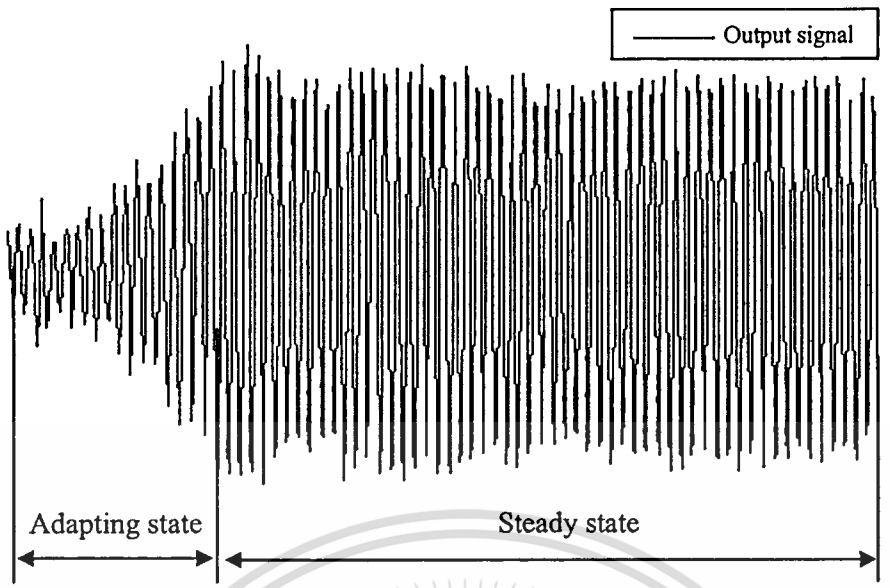


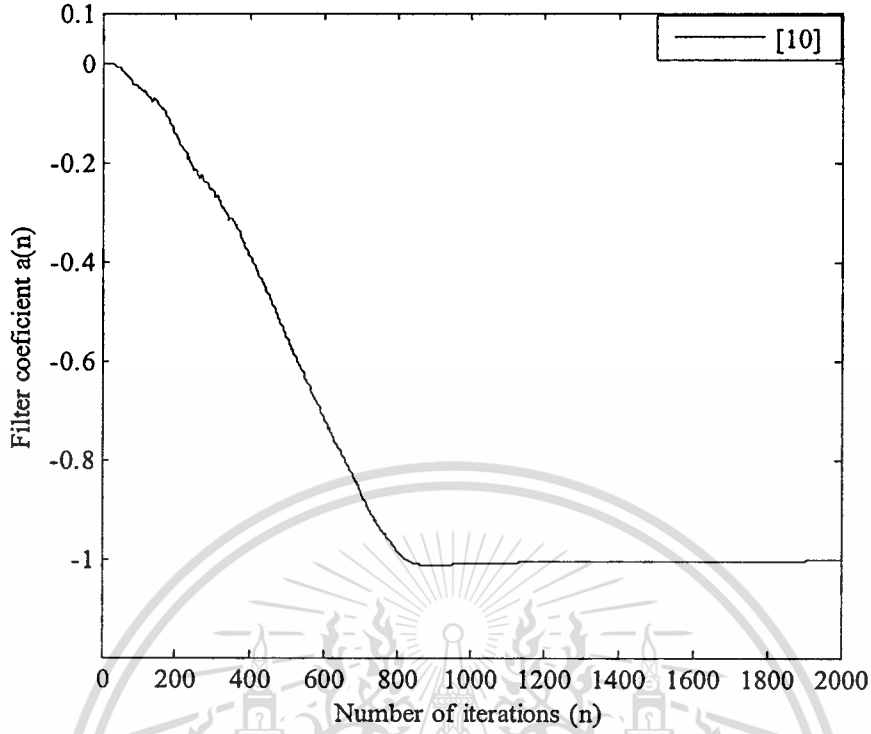
Figure 5.3 The output signal of the adaptive filter.

In Figure 5.3 shows the output signal of the adaptive filter. It is obviously shown an operation of system which have two important states

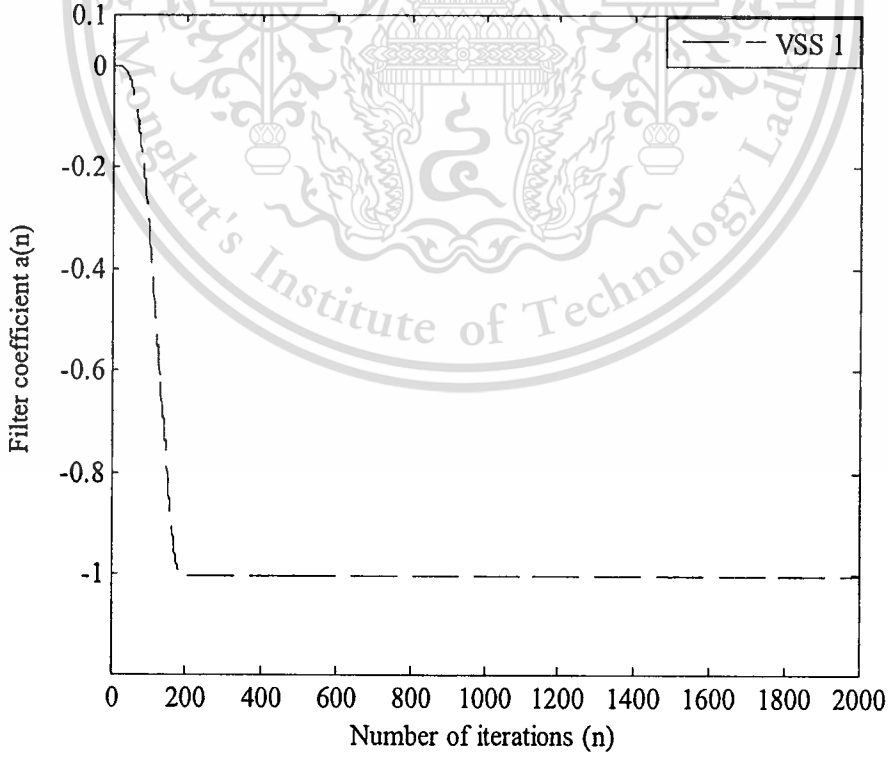
- Adapting-state: it is a first state of the signal which is mixed to noise signal and an algorithm will adapt the filter coefficient to be closed with the real value, and the coefficient of this state should adapt fastest for getting closed the real value of the coefficient and this is the estimate algorithm to be closed real coefficient value.

- Steady-state: this state when the adaptation of the algorithm is slow, this case shown that the coefficient has closely reached the real value, the changing of the coefficient of the algorithm should be minimize in terms of reducing of the variation coefficient.

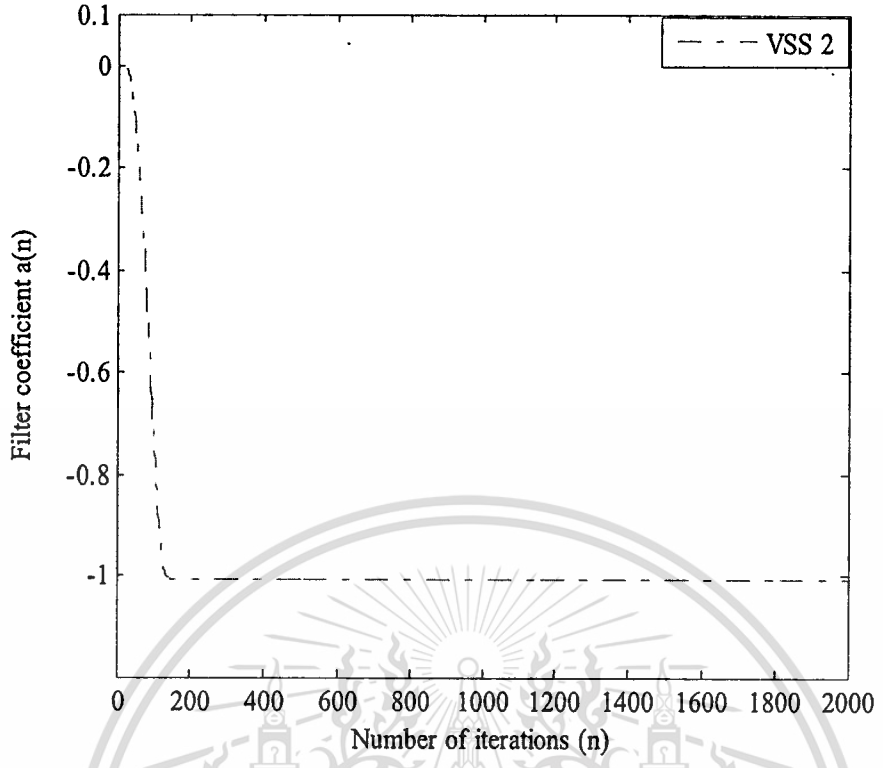
In the simulations, the parameters chosen are as follows: frequency of sinusoidal $\omega_0 = \pi/3$ radial, $\omega_0 = \pi/4$ radial, magnitude of sinusoidal $A = 1$, and disturbed by Gaussian noise, it has signal to noise ratio (SNR) = 15 dB. $a(n)$ is the filter coefficient which is converted to $-2\cos(\omega_0)$, and the parameters for the algorithms are $\rho = 0.95$, $\sigma = 0.98$, $\gamma = 0.01$, $\beta = 0.98$, $\varepsilon = 0.0001$, $\rho(0) = 0.85$, $p(0) = 0$, $\alpha(0) = 0$ and with the same step-size parameter value $\mu(0) = 0.001$.



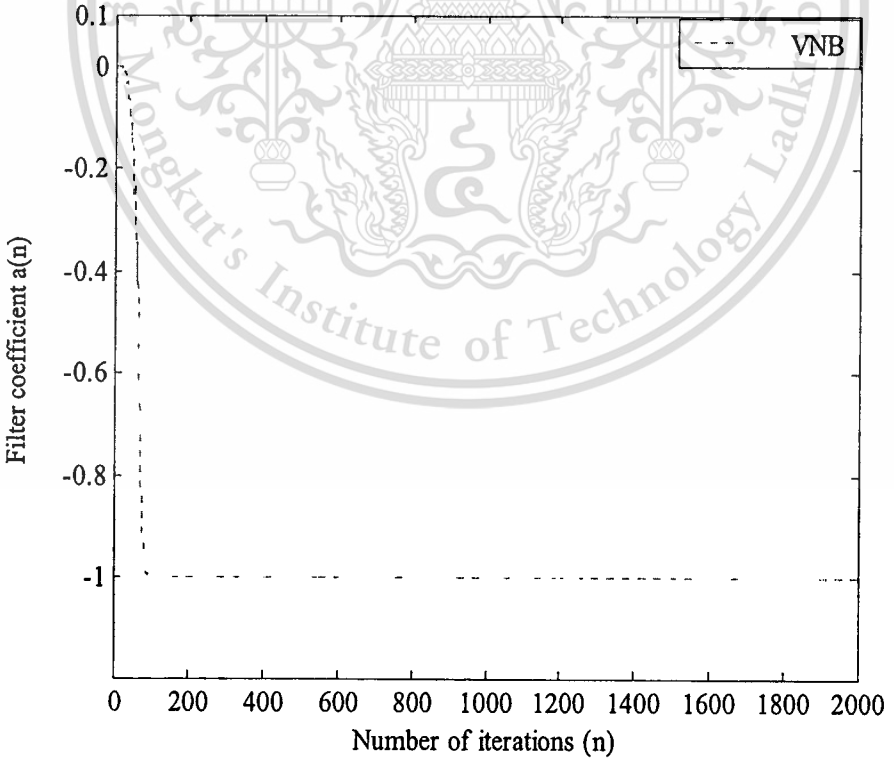
(a)



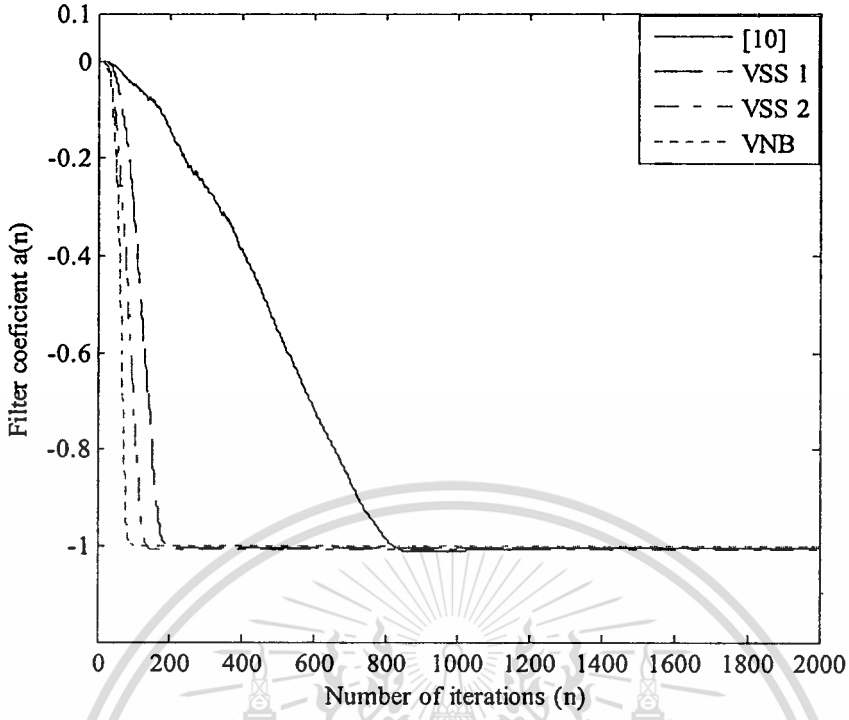
(b)



(c)

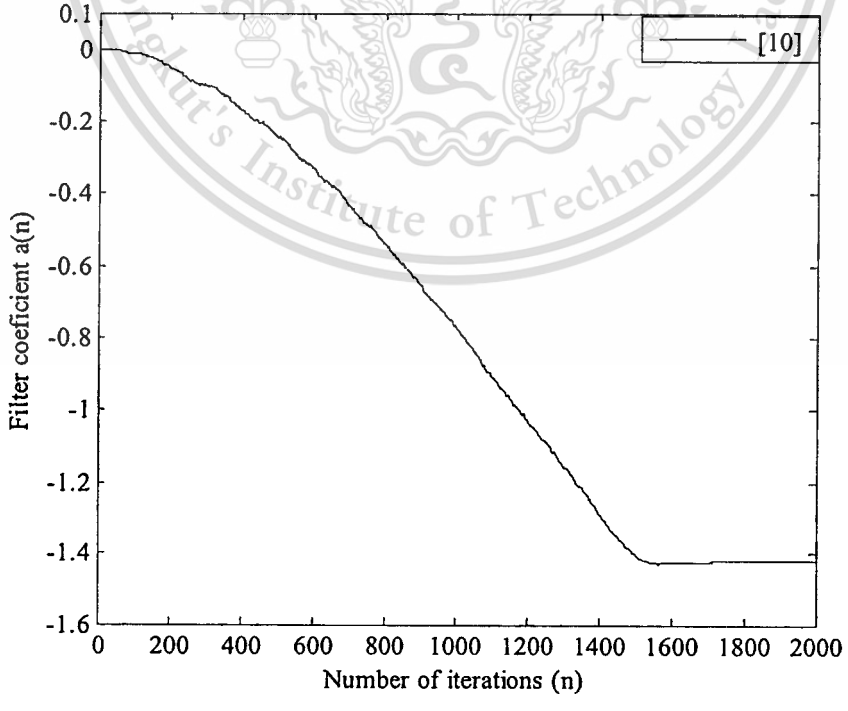


(d)

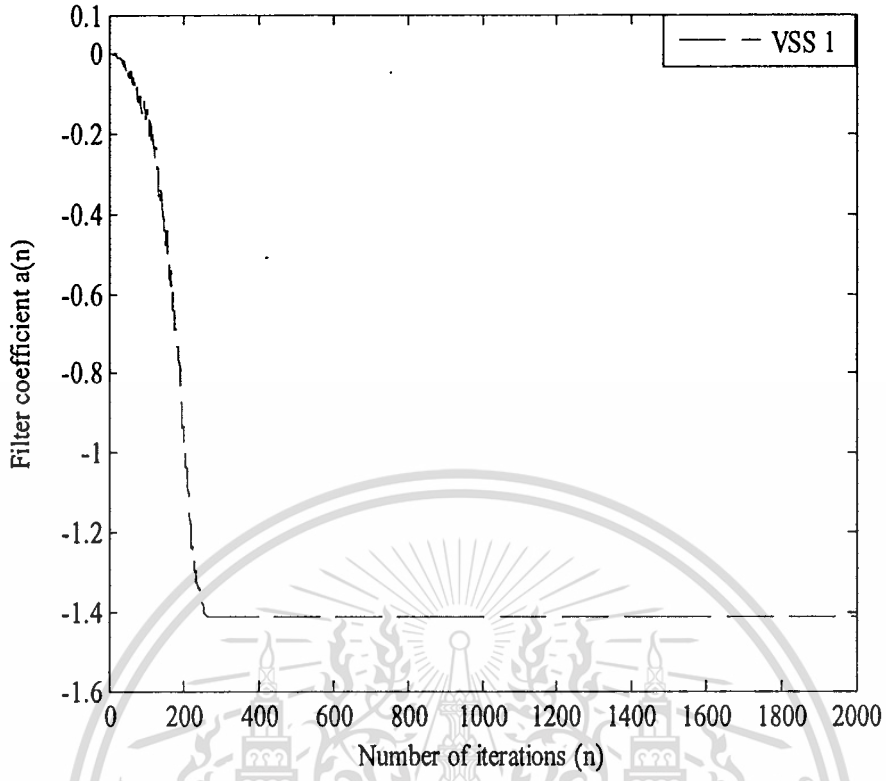


(e)

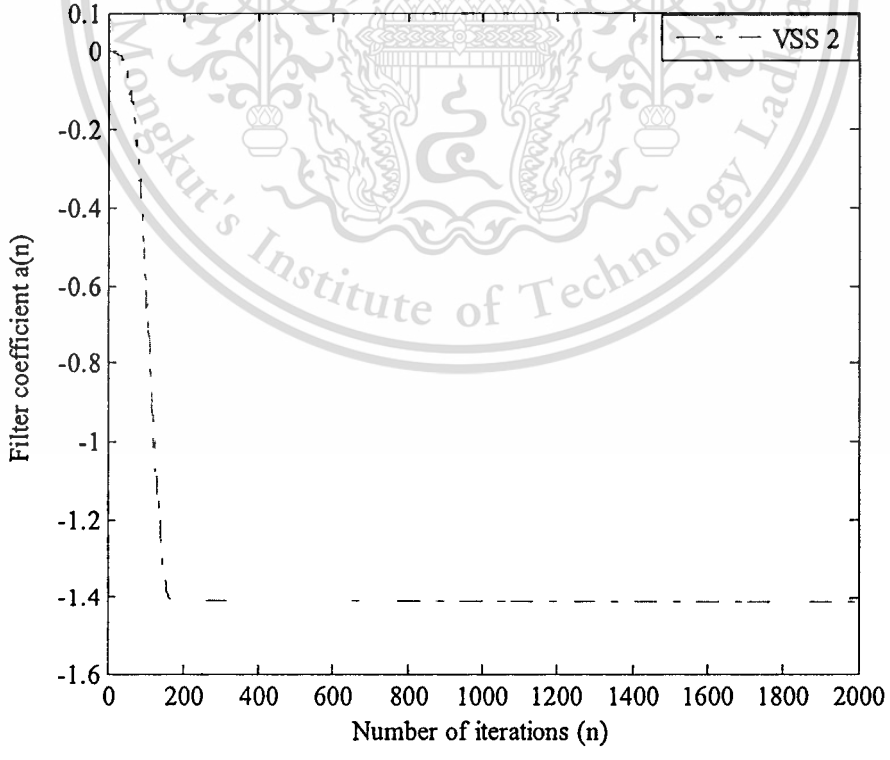
Figure 5.4 The evaluations of filter coefficient with $\omega_0 = \pi/3$.



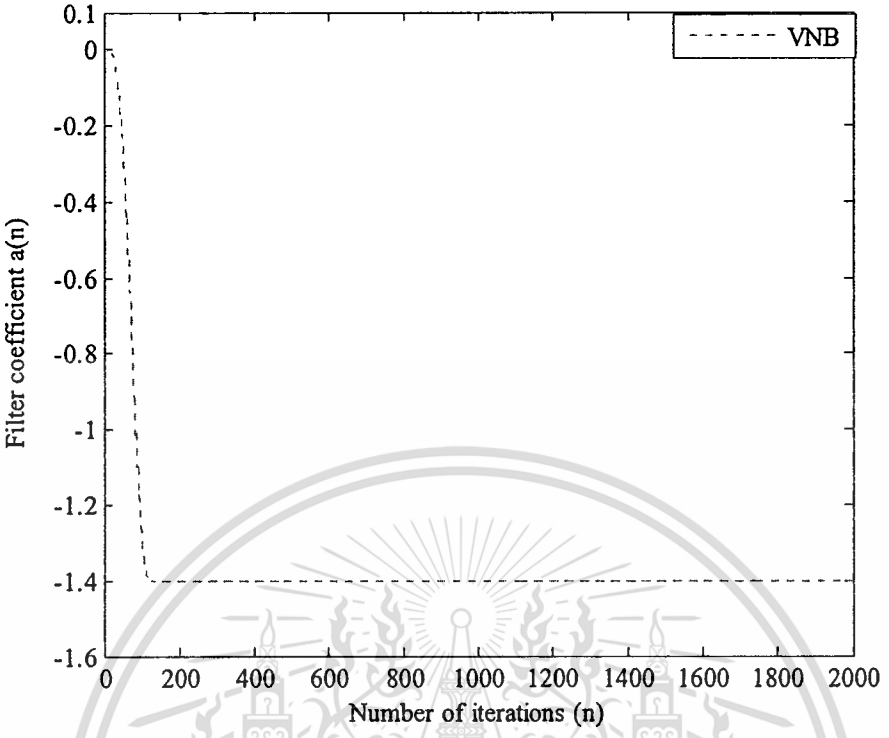
(a)



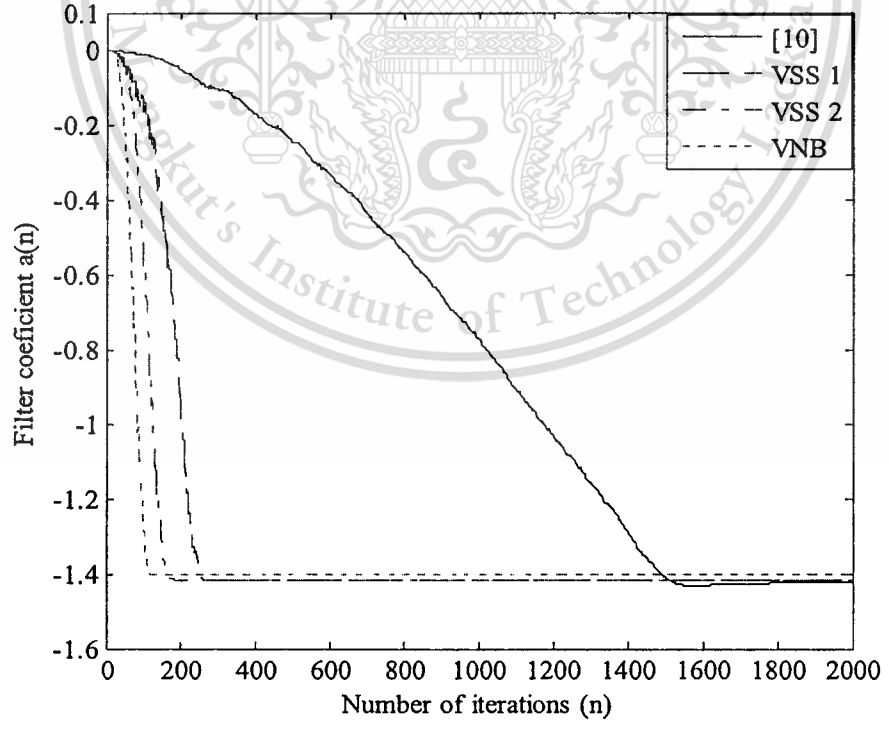
(b)



(c)



(d)



(e)

Figure 5.5 The evaluations of filter with $\omega_0 = \pi/4$.

Figures 5.4 and 5.5 illustrate the estimated filter coefficient $a(n)$ results derived from the algorithms in Gaussian noise environment with frequency of sinusoidal signal by $\omega_0 = \pi/3$ and $\omega_0 = \pi/4$, respectively. It can be seen that in the same environment SNR = 15 dB, in both cases, the convergence speed of the proposed algorithms (VSS 1, VSS 2 and VNB) are faster than the previous algorithm [10].

5.2 Misadjustment

An efficiency property of the adaptive filters is the Misadjustment which can be identified by Mean-square error (MSE). By using an statistical average of adaptive filters.

The mean, variance, bias and mean square error of estimated filters coefficient equation are provided as below:

$$E\{a\} = \frac{1}{M}\sum_{i=1}^{M} a_i(n) \qquad ;n = 0,1,2,\ldots,N \qquad (5.2)$$

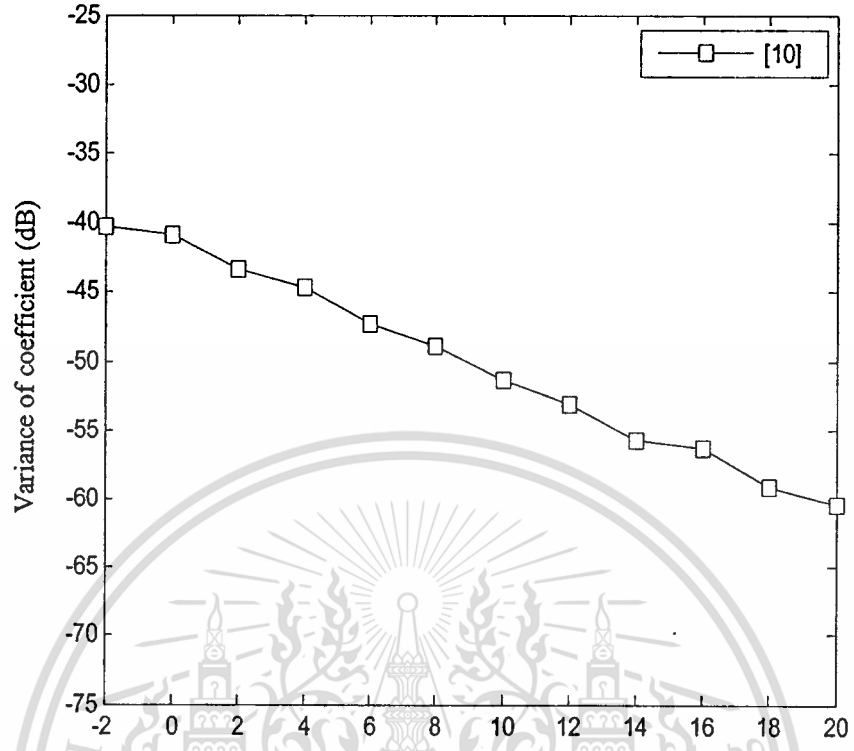
$$Var(a) = \frac{1}{M}\sum_{i=1}^{M} (a_i(n) - E(a))^2 \qquad ;n = 0,1,2,\ldots,N \qquad (5.3)$$

$$bias(a) = a_0 - E\{a\} \qquad (5.4)$$

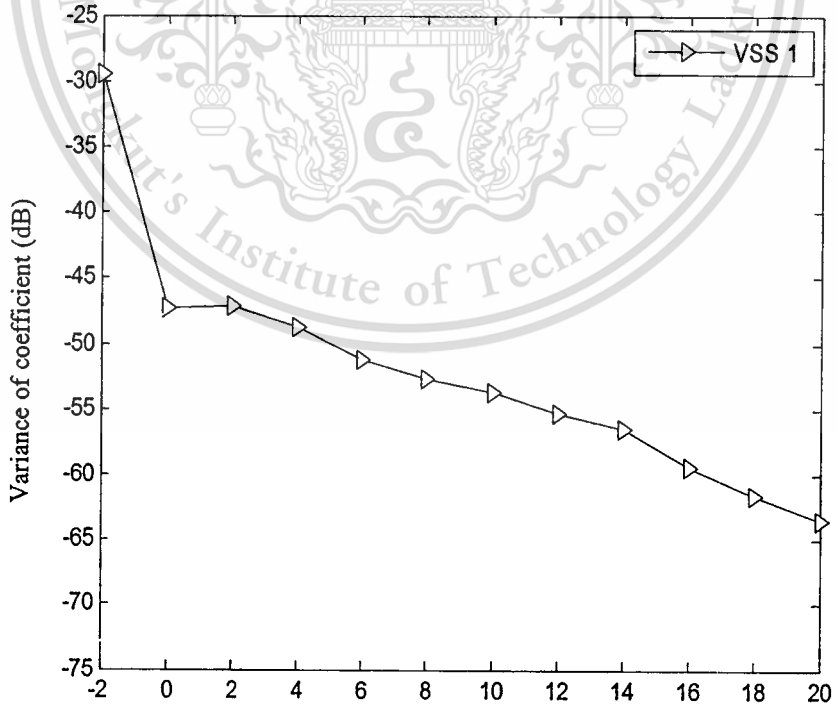
$$MSE = \text{var}(a) + bias^2(a) \qquad (5.5)$$

In the simulations, the parameters chosen are as follows: frequency of sinusoidal $\omega_0 = \pi/3$ radial, magnitude of sinusoidal $A = 1$, and disturbed by Gaussian noise, it has signal to noise ratio (SNR) in the range of 0 - 26 dB, respectively. $a(n)$ is the filter coefficient which is converted to $-2\cos(\omega_0)$, and the parameters for the algorithms are $\rho = 0.95$, $\sigma = 0.98$, $\gamma = 0.01$, $\beta = 0.98$, $\varepsilon = 0.0001$, $\rho(0) = 0.85$, $p(0) = 0$, $\alpha(0) = 0$, data of length N = 2000, independent computer runs M = 100 and with the same step-size parameter value $\mu(0) = 0.001$.

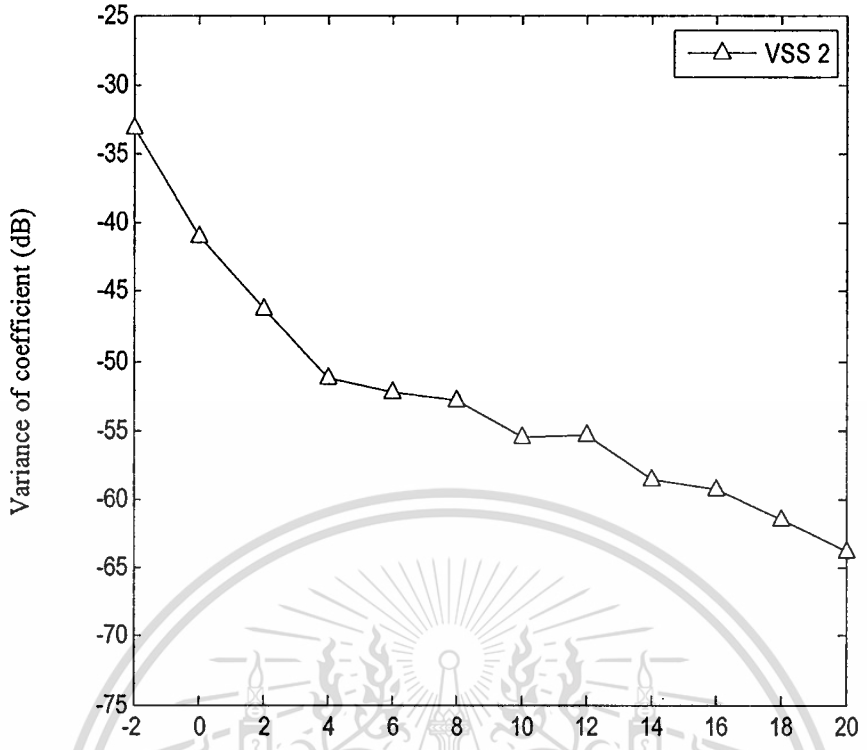
5.2.1 Variance of Filter Coefficients



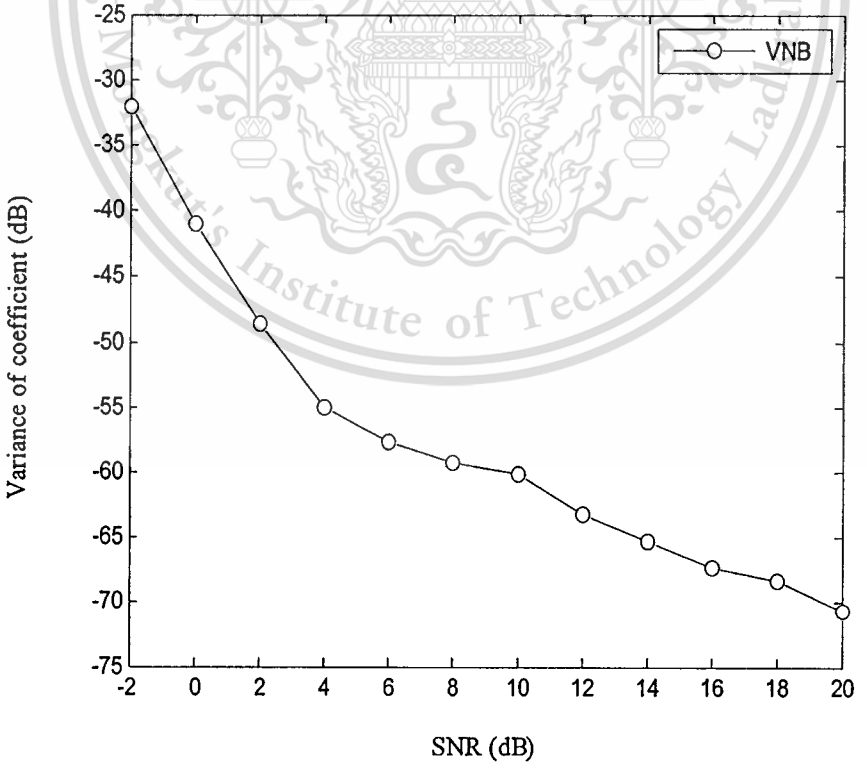
(a)



(b)



(c)



(d)

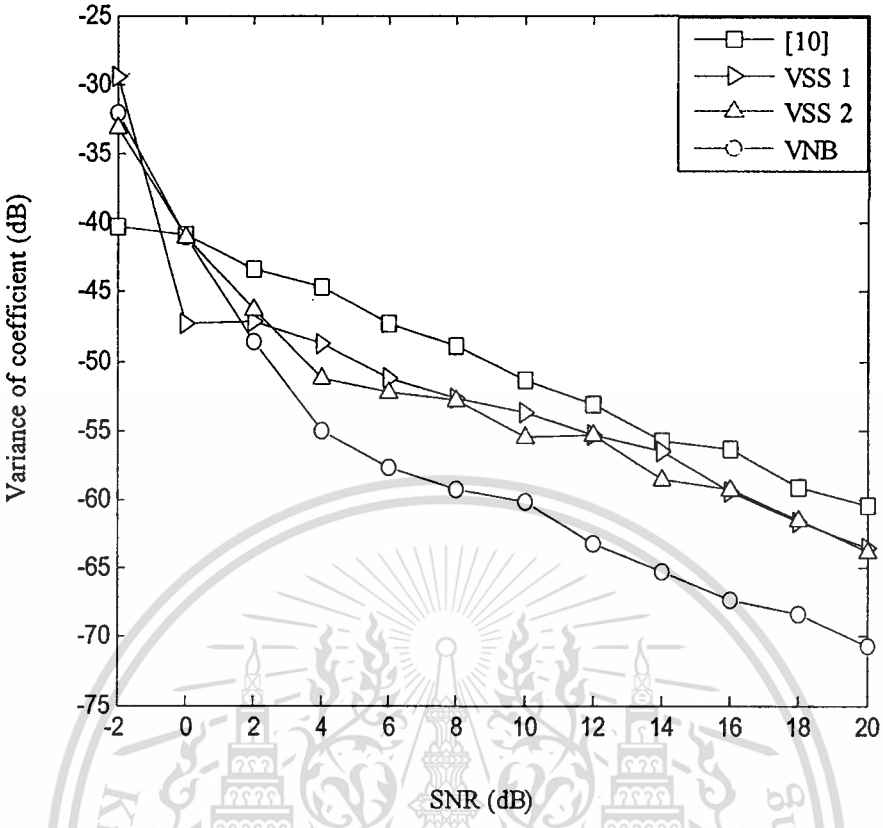
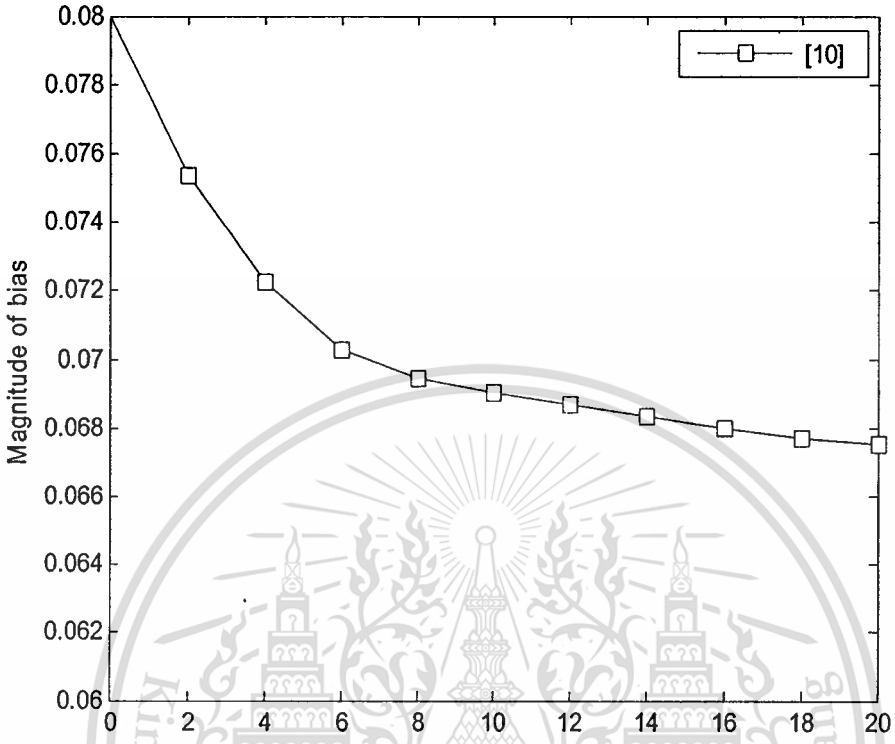


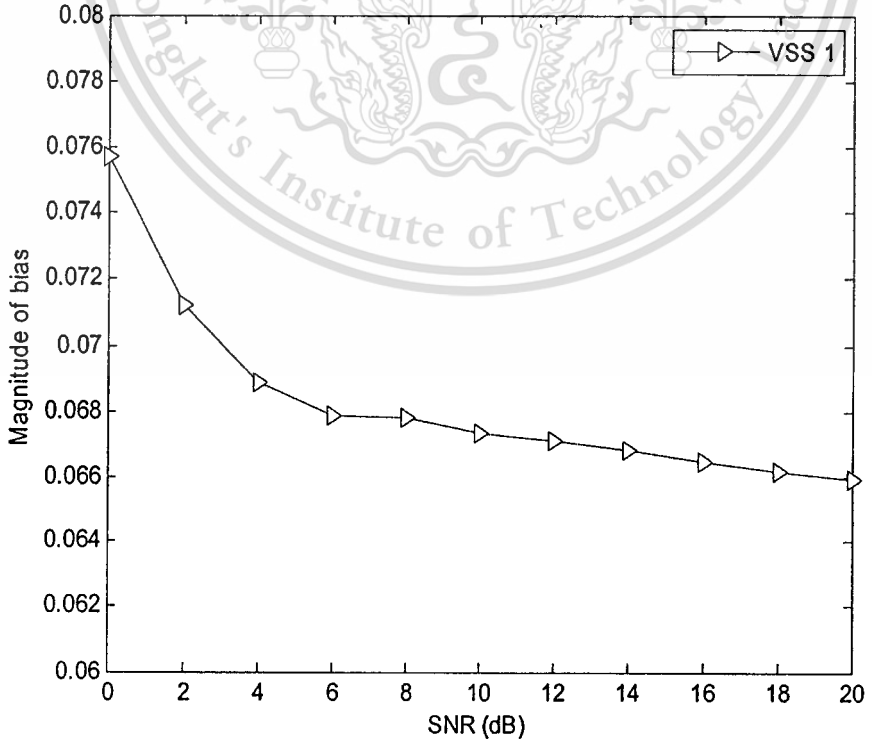
Figure 5.6 Estimation of variance versus signal to noise ratio.

In Figure 5.6 shows a comparison of an estimated variance values of the filter coefficient in various signal to noise ratio (SNR). From Figure 5.6 (e), it can be seen that the variance value of filter coefficient decreased regularly from SNR = -2 dB to 20 dB, and the proposed algorithms (VSS 1, VSS 2 and VNB) provide lower variance of the filter coefficient which gains higher performance than the previous algorithm [10].

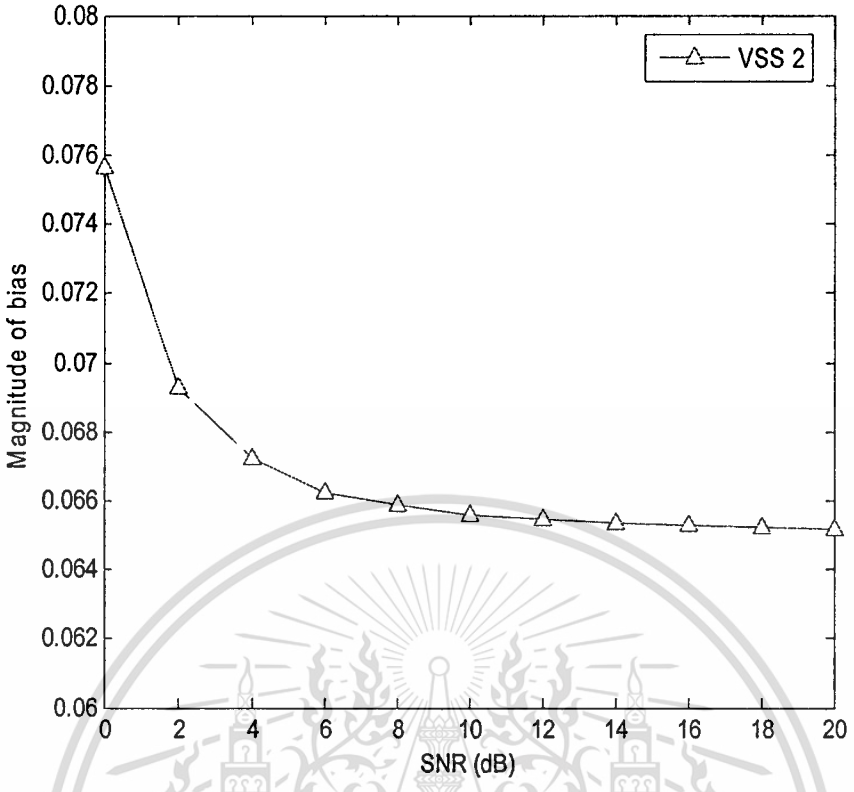
5.2.2 The Magnitude of Bias



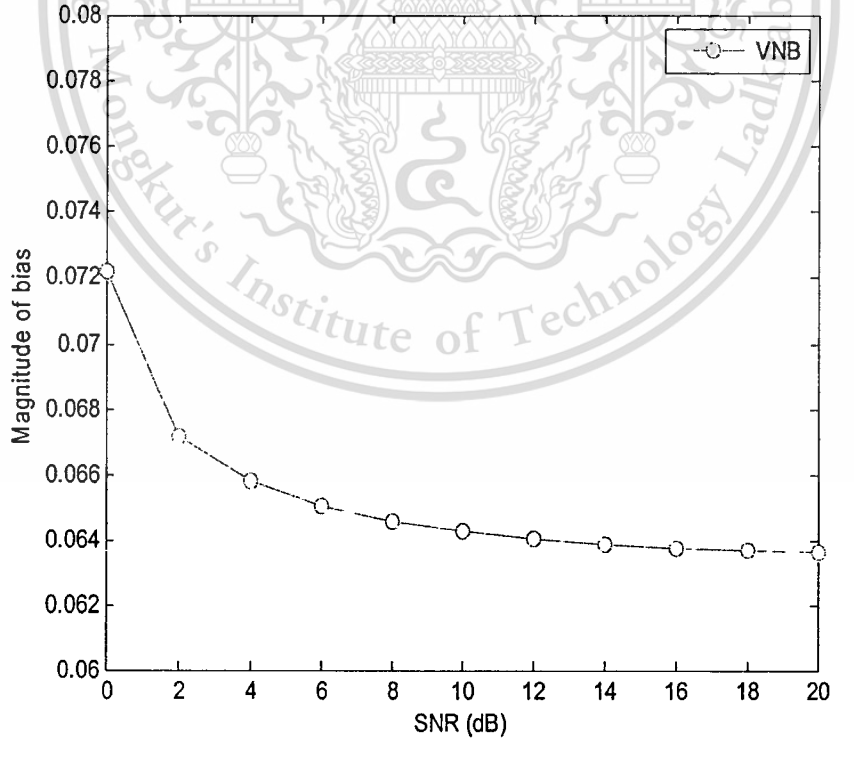
(a)



(b)



(c)



(d)

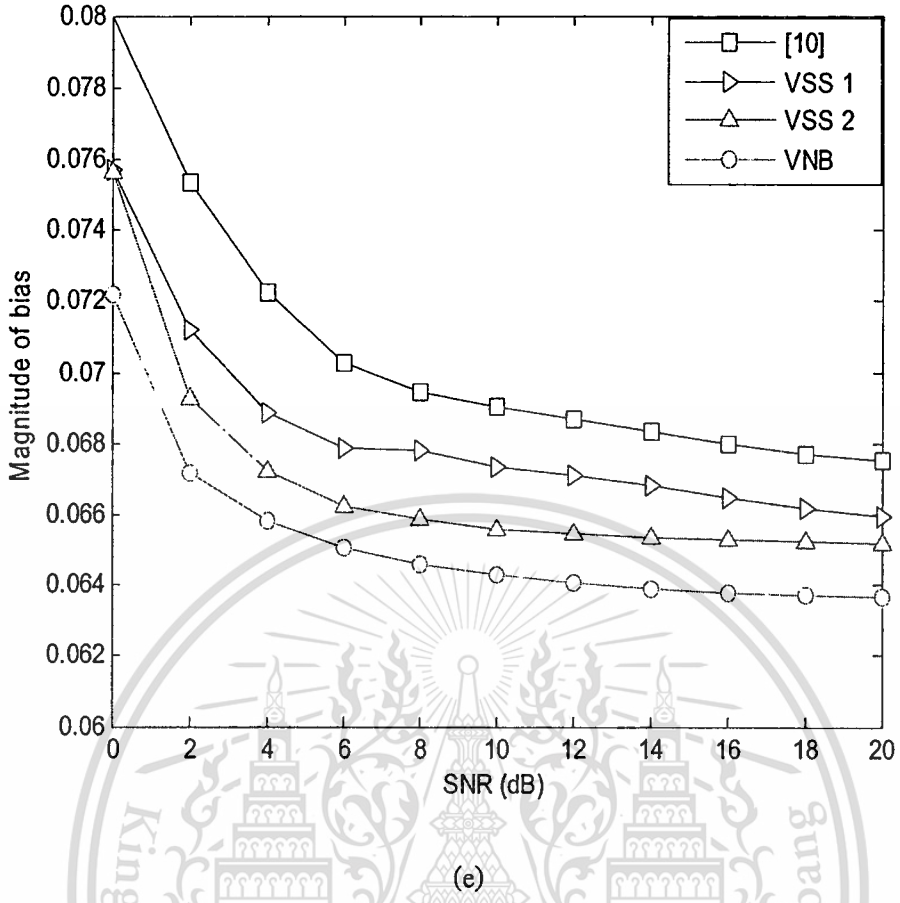
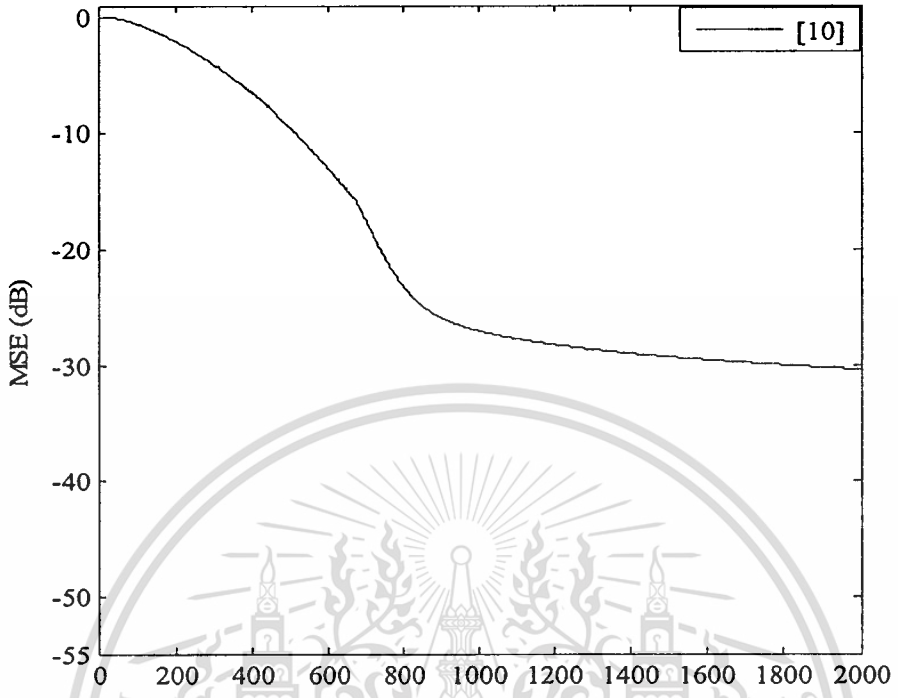


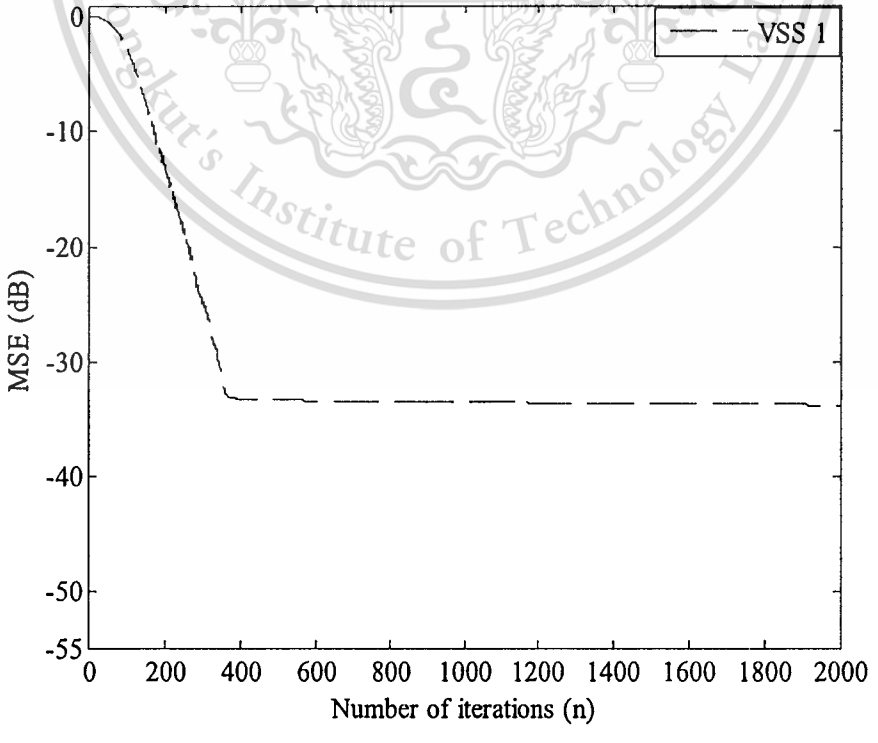
Figure 5.7 Estimation of bias versus signal to noise ratio.

In Figure 5.7 shows estimated bias value of the filter coefficient in various signal to noise ratio, SNR in the range of 0 dB to 20 dB. From Figure 5.7 (e) it is evident that the variable step–size algorithm (VSS 1) and variable step–size algorithm (VSS 2) has lower bias value of filter coefficient than the previous algorithm [10]. Also variable notch bandwidth algorithm (VNB) give higher performance and robustness than the variable step–size algorithm (VSS 1), variable step–size algorithm (VSS 2) and the previous algorithm [10].

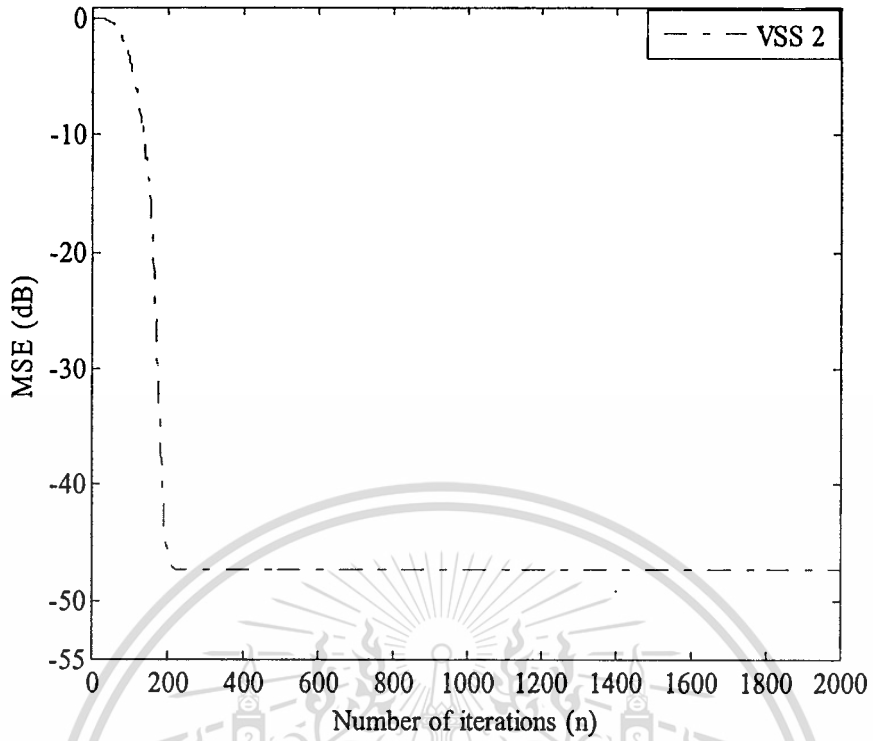
5.2.3 Mean-Square-Error



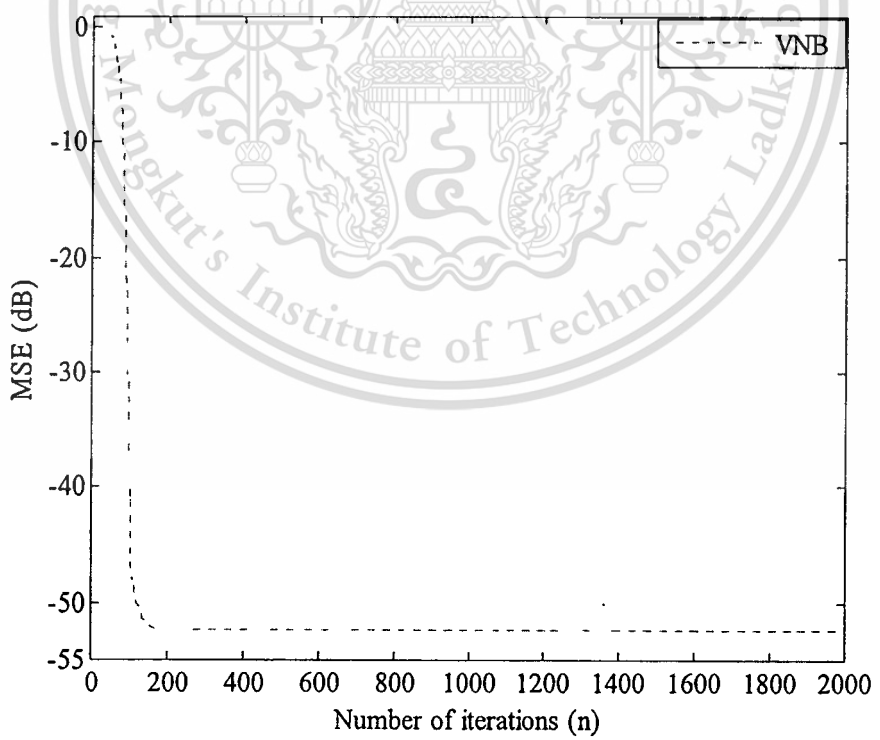
(a)



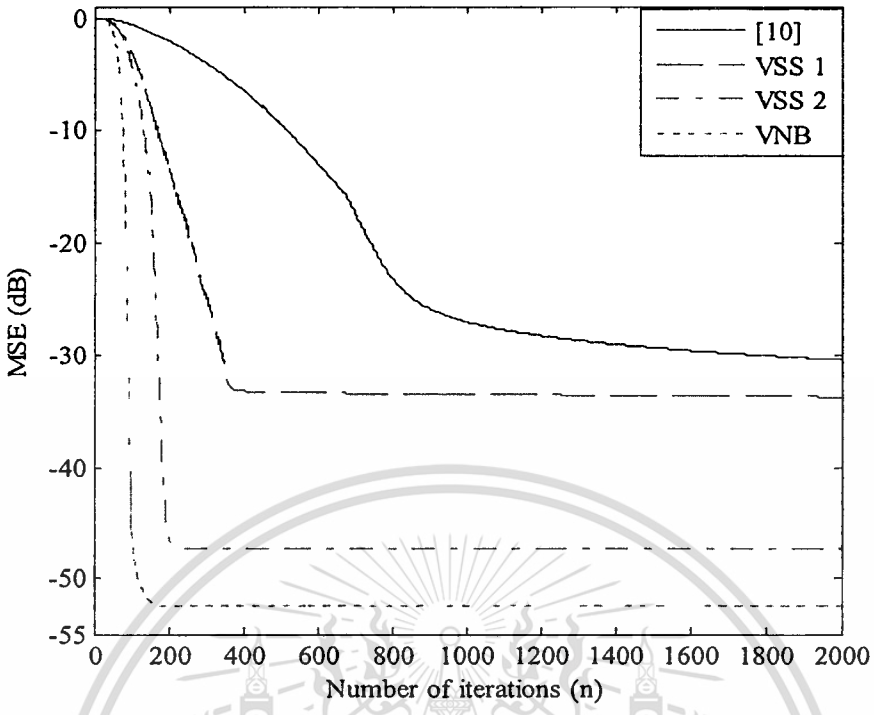
(b)



(c)



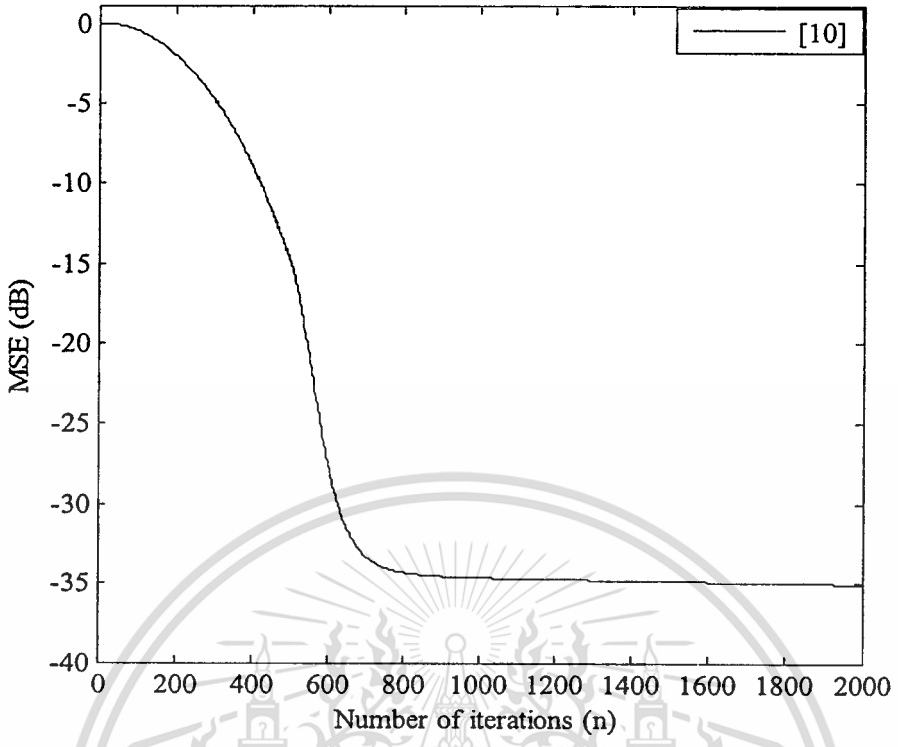
(d)



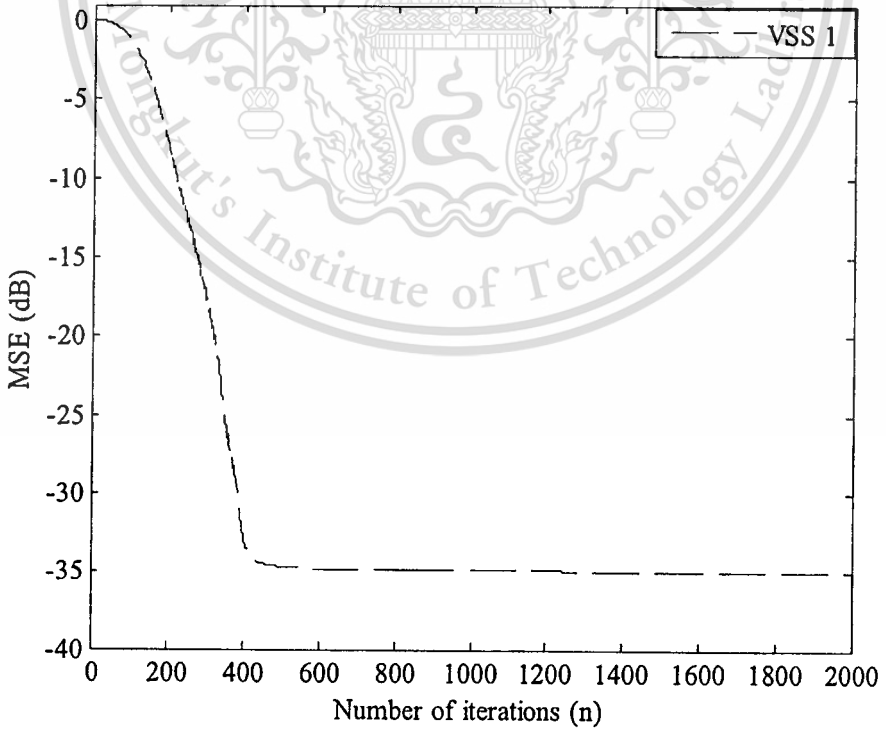
(e)

Figure 5.8 Comparison of an estimated steady-state MSE.

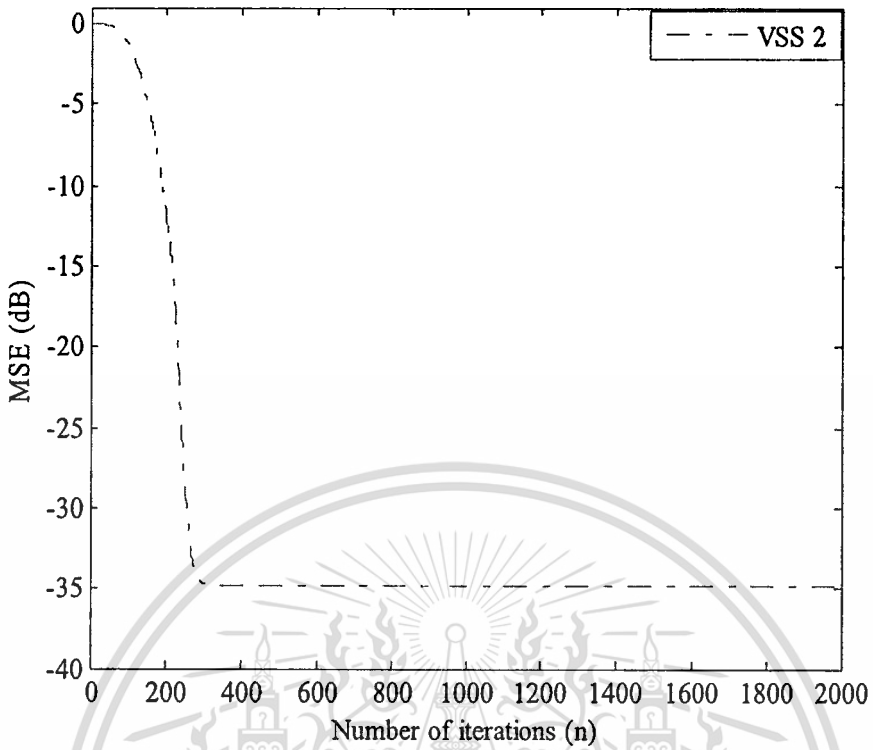
Figure 5.8 shows the comparison of an estimated steady-state MSE of the previous algorithm [10] and the proposed algorithms (VSS 1, VSS 2 and VNB), it is seen that the proposed algorithms provide faster convergence speed and produces less steady-state MSE when compared to the previous algorithm [10].



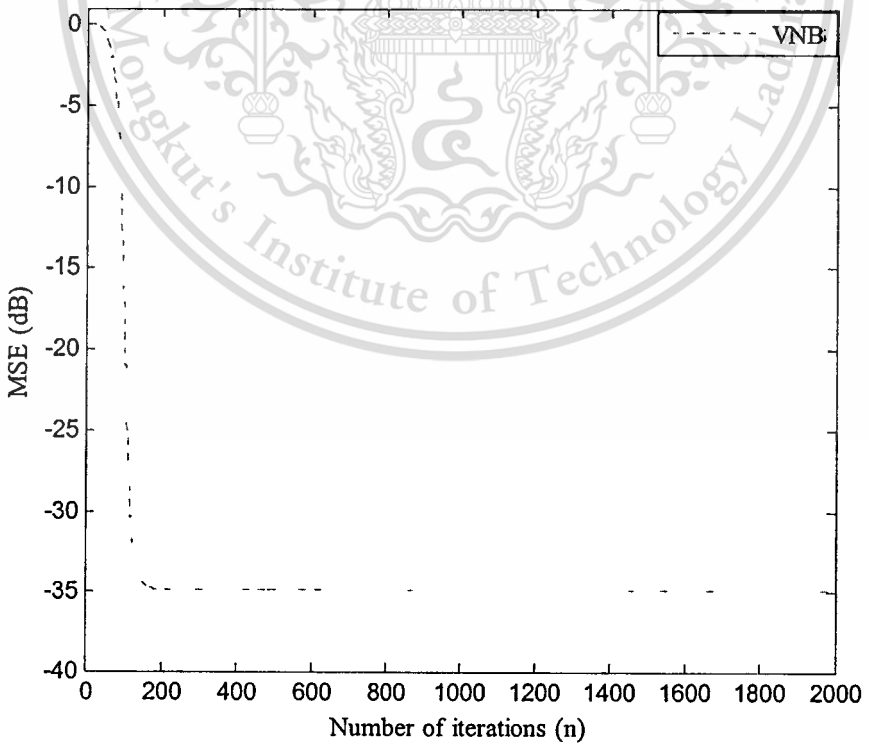
(a)



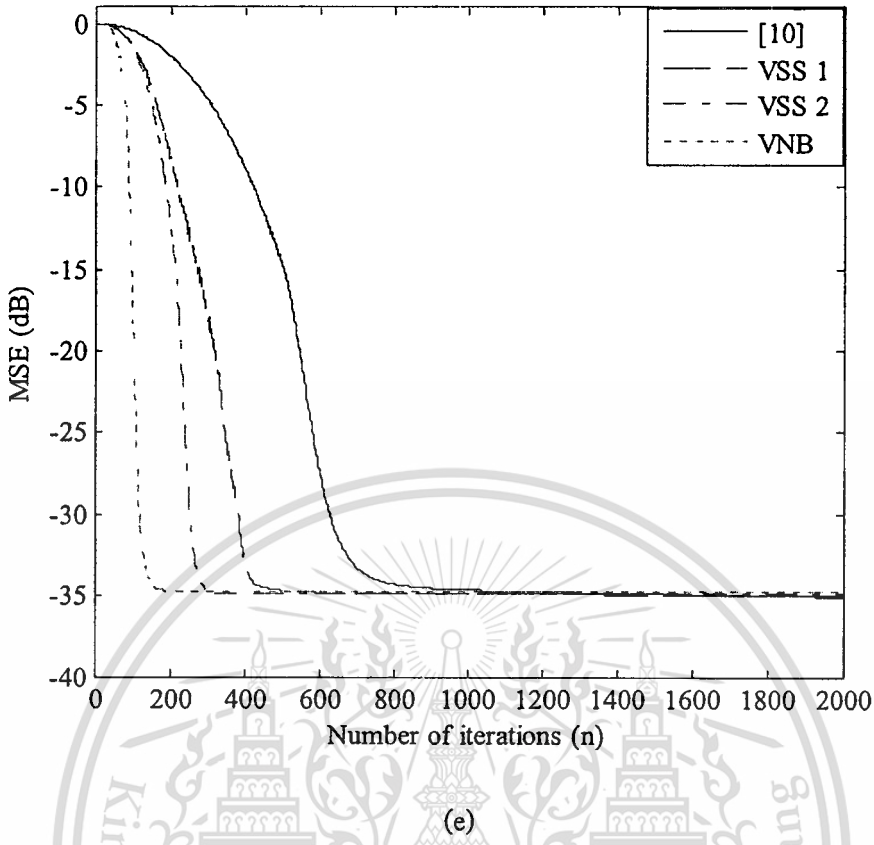
(b)



(c)



(d)



(e)

Figure 5.9 Comparison of convergence rate of the algorithms at the same MSE.

Figure 5.9 shows the comparison of convergence rate of the previous algorithm and proposed algorithms (VSS 1, VSS 2 and VNB) at the same MSE (-35 dB). It is found that the VNB algorithm performs the fastest convergence speed and followed by the VSS 2 algorithm and the VSS 1 algorithm, respectively.

Chapter 6

Conclusion

In this thesis, presents the adaptive noise canceller by using variable step-size adaptive IIR notch filter. The proposed adaptive algorithm for adaptive noise canceller consists of three types.

The first type is the new variable step-size adaptive algorithm 1 (VSS 1) using the power of output signal, the estimate of autocorrelation of the output signal to control the adaptation process of the step function $\mu(n)$ for fast convergence speed, and the new autocorrelation of the error signal and the mean square error (MSE) of the error correlation to accommodate an effective adjusting of $\mu(n)$ for reject the effect of uncorrelated noise sequence on the step-size update to ensuring low misadjustment.

The second type is the new variable step-size algorithm 2 (VSS 2) using the new autocorrelation of the output signal to control the step-size of the algorithm.

Finally, the third type is the new variable notch bandwidth (VNB) algorithm using the variable notch bandwidth technique to control the adaptation process of the algorithm.

To evaluate the performance of the proposed algorithms are used to estimate a single sinusoidal in Gaussian noise environments. The results of the computer simulation have shown superiority of the performance of the proposed algorithms (VSS 1, VSS 2 and VNB) over the previous algorithm [10]. The proposed algorithms provide fast convergence speed, the estimated parameters of the filter such as variance, bias and mean square error (MSE) with high accuracy in Gaussian noise environment. The proposed algorithms are the interested algorithms, but also has more computational complexity and more parameters to appropriately adjust.

References

[1] B. Widrow et al, "Adaptive noise canceling: Principles and applications," Processing of IEEE, vol. 63, pp. 1692-1716, Dec, 1975.

[2] S. Stergiopoulos, "Advanced signal processing handbook: theory and implementation for radar, sonar, and medical imaging real-time systems," Electrical engineering and signal processing series, 2001.

[3] S. Salivahanan, A. Vallavaraj, C. Gnanapriya, "Digital signal processing," Ata McGraw-Hill Publishing Company, International Edition, 2001.

[4] S. Haykin, "Adaptive filter theory," 2nd New jersey: Prentice-Hall, 2002.

[5] S. K. Mitra, "Digital signal processing a computer-based approach," 3rd Edition, McGraw-Hill international Edition, 2006.

[6] C. J. Gibson and S. Haykin, "Learning characteristics of adaptive lattice filtering algorithm," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-28, No. 6, Dec. 1980.

[7] S V. Vaseghi, "Advanced digital signal processing and noise reduction," 4nd Edition, John Wiley & Sons, Ltd, 2008.

[8] B. Farhang - Boroujeny, "Adaptive filters: Theory and applications," John Wiley, 1998.

[9] J. F. Chicharo and T. S. Ng, "Gradient-based adaptive IIR notch filtering for frequency estimation," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-38, pp. 769-777, May 1990.

[10] C. Benjangkaprasert, S. Sukhumalwong, S. Teerasakworakun, and K. Janchitrapongvej, "The new variable step-size algorithm adaptive lattice structure for echo cancellation," Proceeding of ICCAS2003, pp. 2090-2092, Oct. 2003.

[11] Y. Xiao, Y. Tadakoro, and Y. Kobayashi, "A new memoryless nonlinear gradient algorithm for a second-order adaptive IIR notch filter and its performance analysis," IEEE Trans. on Circuit and Systems, Vol. 45, No. 4, pp. 462-472, April 1998.

[12] Y. Xiao and K. Shida, "New gradient-based algorithms for adaptive IIR notch filters," Processing of IEEE ICSP' 98, Vol. 1, pp. 453-456, Oct. 1998.

[13] R. H. Kwong and E. W. Johnson, "A variable step-size LMS algorithm," IEEE Trans. on Signal Processing, Vol.40, pp.4633-1642, July 1992.

[14] N. I. Cho and S.U Lee, "On the adaptive lattice IIR notch filter for the detection of sinusoids," IEEE Trans. Circuits Syst., vol. 40, no. 7, pp. 405-416, July 1993.

[15] T. Aboulnasr and K. Mayyas, "A robust variable step-size LMS- type algorithm analysis and simulation," IEEE Trans. Signal Processing, vol. 45, no. 3, pp. 631-639, March 1997.

[16] P. Tupchai, C. Benjangkaprasert, O. Sangaroon, and K. Janchitrapongvej, "A new algorithm of adaptive IIR notch filter for the detection of sinusoids," Proc. in IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 2002), pp. 505–508, Oct. 2002.

[17] S. Sukhumalwong, C. Benjangkaprasert, and K. Janchitrapongvej, "Adaptive lattice structure filters using variable step-size algorithm for echo cancellation," Proceeding of IEEE TENCON, pp 14-17, 2006.

[18] N. I. Cho and S. U. Lee, "Adaptive line enhancement by using an IIR lattice notch filter," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 37, No. 4, pp.585-589, April 1989.

[19] P. Wattanaluk, C. Benjangkaprasert, O. Sangaroon, and K. Janchitrapongvej, "New variable step-Size algorithm for lattice form structure adaptive IIR filter," Processing of Electronics, computer, telecommunications, and information technology (ECTI-CON), pp. 14-17, May 2006.

[20] C. Benjangkaprasert, M. Chanthavong, O. sangaroon, K. Janchitrapongvej "Adaptive noise canceller using adaptive IIR notch filter based on lattice form structure," International Symposium on Communications and Information Technologies (ISCIT), pp. 18-21, October 21-23, 2008.

Appendix

Appendix A

Research has been published

[1] C. Benjangkaprasert, M. Chanthavong, O. sangaroon, K. Janchitrapongvej "Adaptive noise canceller using adaptive IIR notch filter based on lattice form structure," International Symposium on Communications and Information Technologies (ISCIT), pp. 18-21, October 21-23, 2008.

Appendix B

Abbreviations

ADC	Analog Digital Converter
ANF	Adaptive Notch Filter
BIBO	Bound Input Bound Output
BLUE	Best Linear Unbiased Estimator
Cdf	Cumulative density Function
CRLB	Cramer Rao Lower Bound
DAC	Digital Analog Converter
dB	decibel
DSP	Digital Signal Processing
Exp	Exponential function
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LTI	Linear Time Invariant
LMS	Least Mean Square
LMMSEE	Linear Minimum Mean Squared Error Estimator
LSE	Least Squares Estimator
MAPE	Maximum A Posteriori Estimator
MLE	Maximum Likelihood Estimator
MMSEE	Minimum Mean Squared Error Estimator
MNG	Memoryless Nonlinear Gradient
MSE	Mean Square Error
MVUE	Minimum Variance Unbiased Estimator
PC	Personal Computer
Pdf	Probability density function
PEF	Prediction Error Filter
Pmf	Probability mass function
RF	Radio Frequency
SNR	Signal to Noise Ratio

Appendix B

Abbreviations (Cont)

VNB	Variable Notch Bandwidth
VLSI	Very Large Scale Integration
VSLMS	Variable Step-Size Least Mean Square
VSS	Variable Step-Size
VSS 1	Variable Step-Size 1
VSS 2	Variable Step-Size 2

Biography

Manolom Chanthavong was born in March 11, 1982 at Vientiane, Municipality, Lao P.D.R. In 2004, she received B. Eng in Electronics Engineering, Faculty of Engineering from National University of Laos. She has worked for the Computer Department, National university of Laos as a lecturer since 2004. She is a postgraduate student at KMITL, Thailand under the AUN (SEED-Net Program). Her research interests in adaptive filter.