Växjö University

School of Mathematics and Systems Engineering

Reports from MSI - Rapporter från MSI

Implementation of the LMS and NLMS algorithms for Acoustic Echo Cancellation in teleconference system using MATLAB

Hung Ngoc Nguyen
Majid Dowlatnia
Azhar Sarfraz

December
2009

MSI
Växjö University
SE-351 95 VÄXJÖ

Report 09087
ISSN 1650-2647
ISRN VXU/MSI/ED/E/--09087/--SE

Abstract

In hands-free telephony and in teleconference systems, the main aim is to provide a good free voice quality when two or more people communicate from different places. The problem often arises during the conversation is the creation of acoustic echo. This problem will cause the bad quality of voice signal and thus talkers could not hear clearly the content of the conversation, even thought lost the important information. This acoustic echo is actually the noise which is created by the reflection of sound waves by the wall of the room and the other things exist in the room. The main objective for engineers is the cancellation of this acoustic echo and provides an echo free environment for speakers during conversation. For this purpose, scientists design different adaptive filter algorithms. Our thesis is also to study and simulate the acoustics echo cancellation by using different adaptive algorithms.

Acknowledgements

We would like to express our sincere gratitude to our supervisor: Professor Sven Nordebo, for being a constant source of help and inspiration throughout our work. His timely advice and guidelines have assisted us to get through a lot of difficult situations. Finally, we want to say thanks to our families and friends for their encouragements to support us to accomplish this Master's thesis.

Table of contents

List of tables

List of figures

CHAPTER I : INTRODUCTION

1.1. OVERVIEW

In hands-free telephony and in teleconference systems, the main aim is to provide a good free voice quality when two or more people communicate from different places. The problem often arises during the conversation is the creation of acoustic echo. This problem will cause the bad quality of voice signal and thus talkers could not hear clearly the content of the conversation, even thought lost the important information. This acoustic echo is actually the noise which is created by the reflection of sound waves by the wall of the room and the other things exist in the room. The main objective for engineers is the cancellation of this acoustic echo and provides an echo free environment for speakers during conversation. For this purpose, scientists design different adaptive filter algorithms. Our thesis is also to study and simulate the acoustics echo cancellation by using different adaptive filter algorithms.

1.1.1. Echo

In principle, *"Echo is the phenomenon in which delayed and distorted version of an original sound or electrical signal is reflected back to the source"* [4]. There are two types of echo :

1. Electrical echo: caused by the impedance mismatch at the hybrids transformer which the subscriber two-wire lines are connected to telephone exchange four-wire lines in the telecommunication systems.

2. Acoustic echo: caused by the reflection of sound waves and acoustics coupling between the loudspeaker and the microphone.

In teleconference system (figure I-1), the speech signal from far-end generated from loud speaker after directing and reflecting from the wall, floor and other objects inside the room is receipt by microphone of near-end, as the result, this makes the echo that is sent back to the far-end. The acoustic echo problem will disturb the conversation of the people and reduce the quality of system. This is a common problem of the communication networks.

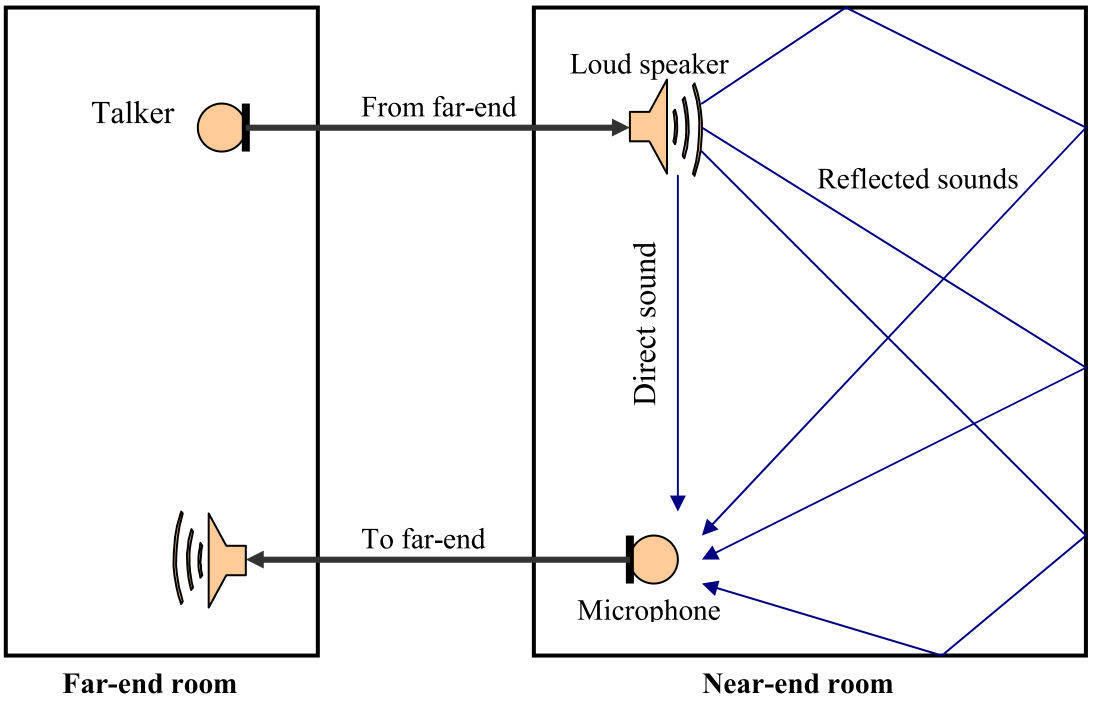


Figure I-1: *A teleconference system with echo paths of room*

Two main characteristics of echo are reverberation and latency. Reverberation is the persistence of sound after stopping the original sound. This sound will slowly decay because of the absorption by the materials constructing the environment. Latency or delay is the different time of the signal between the transmitter and receiver. In the case of teleconference system, the sound is generated from loud speaker and received by microphone, the delay can compute base on the distance between them (i.e., the length of the direct sound).

Delay = distance/speed of sound

1.1.2. Acoustic Echo Cancellation (AEC)

To handle with the acoustic echo problem above in teleconference systems, one can use voice switches and directional microphones but these methods have placed physical restriction on the speaker. The common and more perfective method is implementing the Acoustic Echo Cancellation (AEC) to remove the echo. AEC enhances greatly the quality of the audio signal of the hands-free communication system. Due to their assistance, the conferences will work more smoothly and naturally, keep the participants more comfortable.

Some echo cancellation algorithms are used for this purpose. All of them process the signals follow the basic steps below:

1. Estimate the characteristics of echo path of the room.
2. Create a replica of the echo signal.
3. Echo is then subtracted from microphone signal (includes near-end and echo signals) to obtain the desired signal.

Adaptive filter is a good supplement to achieve a good replica because of the echo path is usually unknown and time-varying. The figure below illustrates about three step of the AEC using adaptive filter.

In the Figure (I-2), by using adaptive filter for AEC follows three basic steps above:

1. Estimate the characteristics of echo path $h(n)$ of the room: $\hat{h}(n)$
2. Create a replica of the echo signal: $\hat{y}(n)$
3. Echo is then subtracted from microphone signal (includes near-end and echo signals) to obtain the desired signal: $clear\ signal = d(n) - \hat{y}(n)$

In the modern digital communication system such as: Public Switched Telephone Network (PSTN), Voice over IP (VoIP), Voice over Packet (VoP) and cell phone networks; the application of AEC is very important and necessary because it brings the better quality of service and obtains the main purpose of the communication service providers.

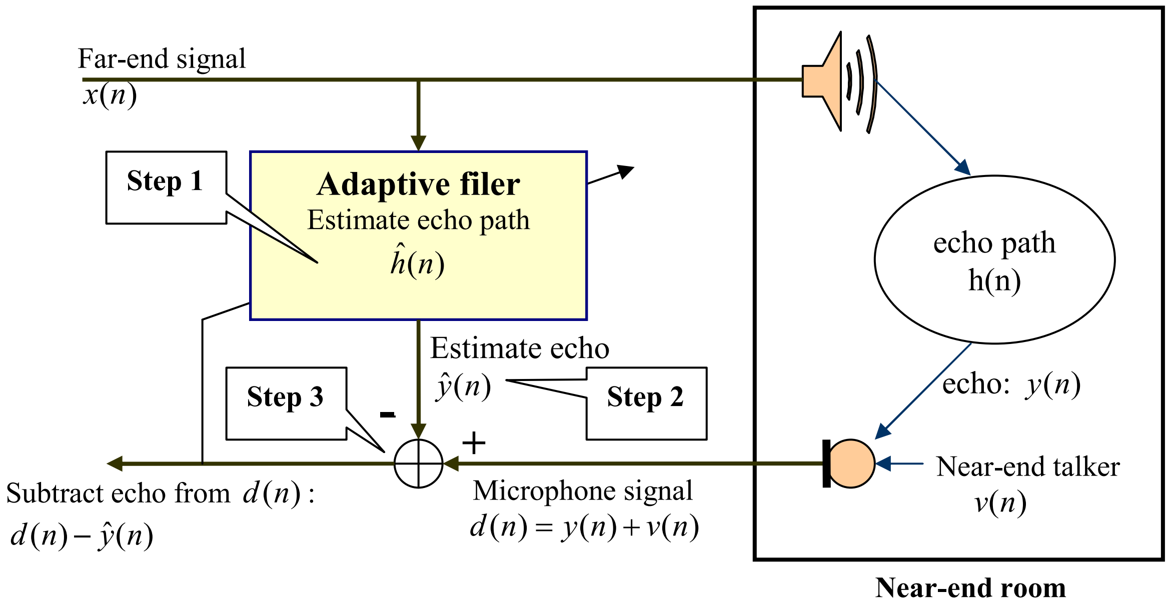


Figure I-2: *Implement Acoustic Echo Canceller using adaptive filter*

1.2. THESIS ORGANIZATION

In this thesis, we will perform the works related to the Acoustic Echo cancellation. It contains 4 chapters that focuses on two main parts are theory and simulation. All of them try to express and discuss about two main issues of acoustic echo cancellation, namely the adaptation algorithms and the control of adaptation in double-talk situation.

Chapter 1: give the general information and introduction of the problems and solutions related to the thesis' topic. And mention the brief descriptions of echo theory and acoustic echo problem in teleconference system and other telecommunication systems.

Chapter 2: presents all the theory backgrounds. The adaptive filter which is used to model the acoustic echo path is the central part of the AEC. Hence much effort and researches have been devoted to it. Least Mean Square (LMS) algorithm is an old, simple and proven algorithm which has turned out to work well in comparison with newer more advanced algorithms. In this project, we use the normalized LMS (NLMS) for the main filter in AEC, since NLMS is so far the most popular algorithm in practice for its computational simplicity. After that, the generic double talk detection scheme is outlined and then several well-known double talk detectors are discussed. The Geigel algorithm is simple and works well when the far-end signal is sufficiently smaller than the near-end speech, namely it has assumption of the echo path, so in practice not widely applied to the echo cancellation algorithms. The Normalized Cross-correlation method uses the correlation value between the error signal and the microphone signal which would bring more promising results compared to the Geigel algorithm.

Chapter 3: is devoted to the evaluation of all the algorithms discussed above. Through a bunch of recordings and simulations in MATLAB, we try to find out which adaptive filtering and double talk detection algorithms suit better for the PC application.

Chapter 4: the conclusion is drawn and also the possible future work is presented.

CHAPTER II : THEORY OF ACOUSTIC ECHO CANCELLATION

2.1. SYSTEM OVERVIEW

Acoustic echo cancellation is required in different fields of communication for removing the echo of the coupling between the loudspeaker and the microphone. In case of not doing this, then this coupling results in an undesired acoustic echo which degrades the quality of sound.

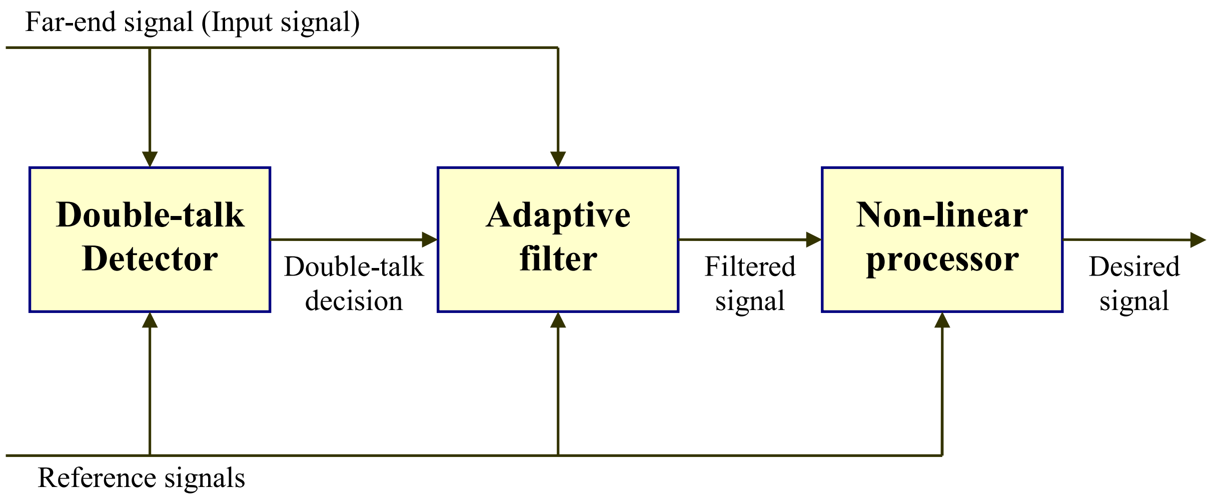


Figure II-1: *Block diagram of AEC*

We describe a block diagram of an AEC system as in Figure (II-1). This system consists of following three components:

1. Adaptive filter.
2. Doubletalk detector.
3. Nonlinear processor.

2.1.1. Adaptive Filter

Adaptive filter is the most important component of acoustic echo canceller and it plays a key role in acoustic echo cancellation. It performs the work of estimating the echo path of the room for getting a replica of echo signal. It requires an adaptive update to adapt to the environmental change. Another important thing is the convergence rate of the adaptive filter which measures that how fast the filter converges for best estimation of the room acoustic path.

2.1.2. Double-talk detector (DTD)

It is rather difficult to predict when the adaptation of the filter should stop or slow down and it is also important to know that the near-end speech signal exists or not in the presence of far-end signal. In the situation when both ends talk (near-end and far-end), this is known as double-talk. In case of double-talk, the error signal will contain both echo estimation error and near-end speech signal. When we use this signal for updating the filter coefficient then it diverges. As the result, the adaptive filter will work incorrectly and finally the bad sound signal was issued. So to overcome this problem, one uses Double-talk Detector.

2.1.3. Nonlinear Processor (NLP)

The nonlinear processor (NLP) is required for completely or partly cancels the residual signal in the absence of near-end speech signal. By removing the residual signal will cancel any occurring acoustic echo. The NLP will gradually cancel the signal and insert a form of comfort noise to give the impression to far-end. The NLP as well as the adaptive filter need an accurate estimation from the DTD to operate efficiently.

2.2. ADAPTIVE FILTER ALGORITHMS

Adaptive filtering is the process which is required for echo canceling in different applications. Adaptive filter is such type of filter whose characteristics can be changed for achieving optimal desired output. An adaptive filter can change its parameters to minimize the error signal by using adaptive algorithms. The error is the difference between the desired signal and the output signal of the filter. The figure below shows the basic model of adaptive filter used in AEC.

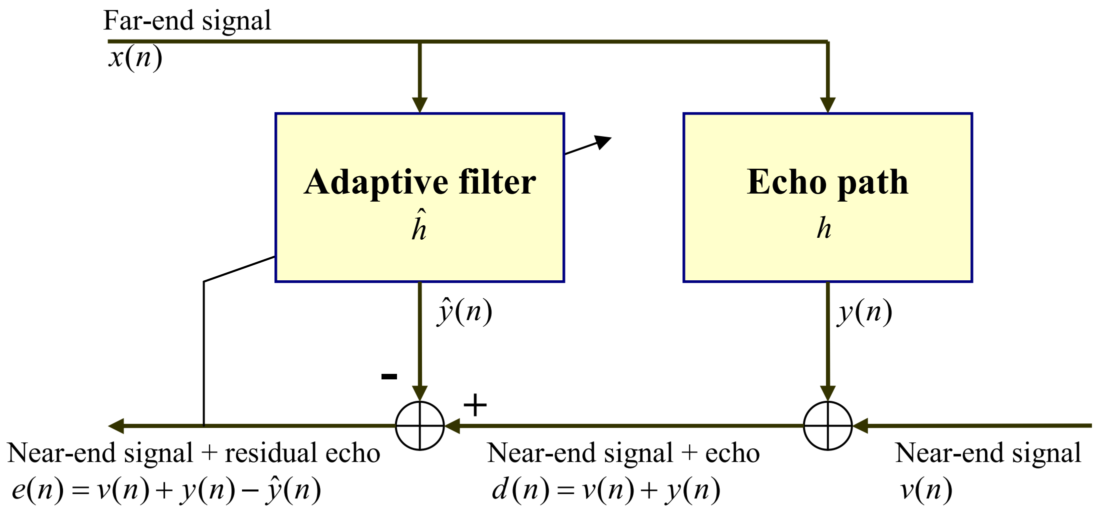


Figure II-2: *The basic model of AEC*

The notations are used in the figure above and during this thesis in turn are:

- Far-end signal: $x(n)$
- Near-end signal: $v(n)$
- The true echo path (room impulse response): h
- Echo signal: $y(n)$
- Microphone signal: $d(n)=v(n)+y(n)$
- Estimated echo path: $\hat{h}$
- Estimate echo signal: $\hat{y}(n)$
- Error signal: $e(n)=v(n)+y(n)-\hat{y}(n)$

The echo path h of the room normally variable depends on the room structure and the moving object inside. The estimated echo $\hat{y}(n)$ is calculated from the reference input signal $x(n)$ and the adaptive filter $\hat{h}$. The near-end signal $v(n)$ and background noise are added into echo signal $y(n)$ to create the desired signal $d(n)$,

$$d(n) = v(n) + y(n) \tag{2.1}$$

The signal $x(n)$ and $y(n)$ are correlated. We get the error signal as,

$$error(n) = d(n) - \hat{y}(n) = v(n) + y(n) - \hat{y}(n) \tag{2.2}$$

The adaptive filter works to minimize the echo ($y(n) - \hat{y}(n)$) to be zero to obtain only near-end signal $v(n)$ in the perfect case.

In Acoustic Echo Cancellation (AEC), the adaptive filter plays the main role to adapt the filter tap weight in order to overcome the echo problem. There are different types of algorithms are used for this purpose such as Least Mean Square (LMS), Normalized Least Mean Square (NLMS), Recursive Least Square (RLS) and Affine Projection Algorithm (APA) and etc. The LMS is widely used algorithm for adaptive application such as channel equalization and echo cancellation. This algorithm is the most simple if we compare it with NLMS and RLS algorithm. The normalized least mean square (NLMS) is also famous algorithm due to its computational simplicity.

2.2.1. Wiener Filter

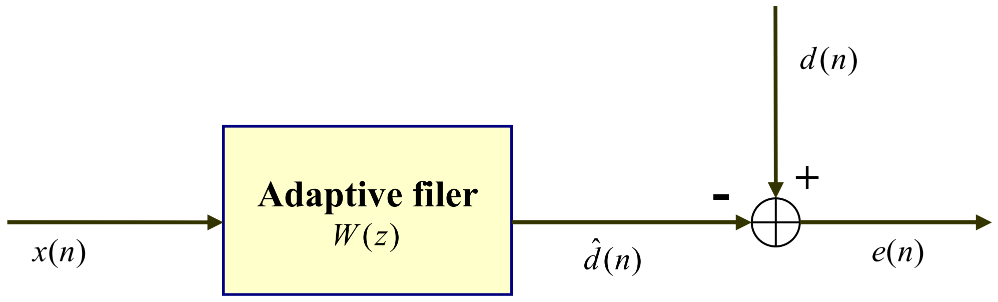


Figure II-3: *General Wiener filter problem.*

Wiener filters play a central role in a wide range of applications such as linear prediction, echo cancellation, signal restoration, channel equalization and system identification. [5]

The FIR Wiener Filter is the signal processing to produces the minimum mean-square estimate, $\hat{d}(n)$ of $d(n)$. Two signals $x(n)$ and $d(n)$ are assumed to be wide-sense stationary with known autocorrelations $r_x(k)$, $r_d(k)$ and cross-correlation $r_{dx}(k)$. $w(n)$ is the unit sample response of the wiener filter:

$$W(z)=\sum_{n=0}^{p-1} w(n)z^{-n} \tag{2.3}$$

The output signal $\hat{d}(n)$ of the Wiener filter is the convolution of $w(n)$ and $x(n)$,

$$\hat{d}(n)=\sum_{l=0}^{p-1} w(l)x(n-l) \tag{2.4}$$

The requirement of the filter is to find filter coefficients $w(k)$ that minimize the mean-square error,

$$\xi = E\left\{|e(n)|^2\right\} = E\left\{\left|d(n)-\hat{d}(n)\right|^2\right\} \tag{2.5}$$

Now taking the derivative to both sides with respect to $w^*(k)$,

$$\frac{\partial \xi}{\partial w^*(k)} = \frac{\partial}{\partial w^*(k)} E\left\{e(n)e^*(n)\right\} = E\left\{e(n)\frac{\partial e^*(n)}{\partial w^*(k)}\right\} \tag{2.6}$$

For $k=0,1,...,p-1$,

This derivative must be equal to zero to minimize ξ for a set of filter coefficients,

$$E\left\{e(n)\frac{\partial e^*(n)}{\partial w^*(k)}\right\}=0 \tag{2.7}$$

Where,

- The error signal: $e(n)=d(n)-\hat{d}(n)=d(n)-\sum_{l=0}^{p-1} w(l)x(n-l)$ (2.8)

it follows that,

$$\frac{\partial e^*(n)}{\partial w^*(k)} = -x^*(n-k)$$

Now the above Equation (2.7) becomes,

$$E\left\{e(n)x^*(n-k)\right\}=0; \qquad k=0,1,2,...,p-1 \tag{2.9}$$

This equation is known as *orthogonally principle or the projection theorem.*

By substituting $e(n)$ in Equation (2.8) into Equation (2.9) , we have

$$E\left\{d(n)x^*(n-k)\right\}-\sum_{l=0}^{p-1}w(l)E\left\{x(n-l)x^*(n-k)\right\}=0 \qquad (2.10)$$

Already assumed that $x(n)$ and $d(n)$ are jointly wide-sense stationary, then

$$E\left\{x(n-l)x^*(n-k)\right\}=r_x(k-l) \qquad (2.11)$$

$$E\left\{d(n)x^*(n-k)\right\}=r_{dx}(k) \qquad (2.12)$$

so the above equation becomes,

$$\sum_{l=0}^{p-1}w(l)r_x(k-l)=r_{dx}(k); \qquad k=0,1,2,\ldots,p-1 \qquad (2.13)$$

This equation is known as Wiener-Hopf equation and we can write this equation in generalized form as,

$$\mathrm{R}_x\,\mathrm{w}=\mathrm{r}_{dx} \qquad (2.14)$$

Where:

- R_x is $p\times p$ Hermitian Toeplitz matrix of auto correlation
- w is vector of filter coefficients
- r_{dx} is vector of cross-correlation between $d(n)$ and $x(n)$

By taking the Equation (2.5), we try to find the minimum mean square error,

$$\xi=E\left\{|e(n)|^2\right\}=E\left\{e(n)\left[d(n)-\sum_{l=0}^{p-1}w(l)x(n-l)\right]^*\right\}$$

$$=E\left\{e(n)d^*(n)\right\}-\sum_{l=0}^{p-1}w^*(l)E\left\{e(n)x^*(n-l)\right\} \qquad (2.15)$$

By following the Equation (2.9), the second term of above Equation (2.15) is equal to zero. So we attain,

$$\xi_{\min}=E\left\{e(n)d^*(n)\right\} \qquad (2.16)$$

Also, we have,

$$e(n)=d(n)-\sum_{l=0}^{p-1}w(l)x(n-l)$$

So Equation (2.16) becomes,

$$\xi_{\min} = E\left\{e(n)d^*(n)\right\} = E\left\{\left[d(n) - \sum_{l=0}^{p-1} w(l)x(n-l)\right]d^*(n)\right\} \quad (2.17)$$

Finally, by taking the expected values, we have

$$\xi_{\min} = r_d(0) - \sum_{l=0}^{p-1} w(l) r_{dx}^*(l) \quad (2.18)$$

In vector form, we have from Equation (2.14) and Equation (2.18)

$$\xi_{\min} = r_d(0) - \mathrm{r}_{dx}^H \mathrm{w} \quad (2.19)$$

Or
$$\xi_{\min} = r_d(0) - \mathrm{r}_{dx}^H \mathrm{R}_x^{-1} \mathrm{r}_{dx} \quad (2.20)$$

2.2.2. The Steepest Decent Method

The method of steepest descent is an iterative procedure that has been used to find extreme of nonlinear functions since before the time of Newton. [5]

In the steepest decent or gradient algorithm, the mean square error surface (respect to an FIR filter coefficients) is a quadratic bowl-shaped curve as shown in Figure (II-4) below.

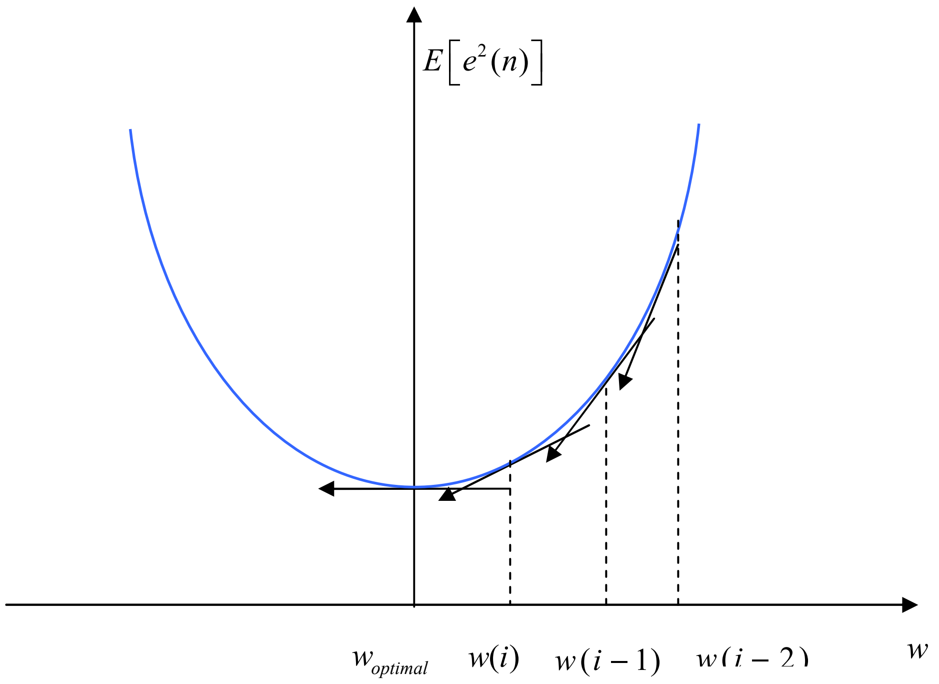


Figure II-4: *Illustration of gradient search of the mean square error surface for the minimum error point*

This figure explains the mean square error curve for a single coefficient filter and the steepest decent search for the coefficient of minimum mean square error. This steepest decent search is to find a value by taking successive downward step in the direction of negative gradient of the error surface. By taking start with different initial values and the coefficients of the filter are updated while moving in the downward direction towards the negative gradient and until a point comes where the gradient shows zero value. This steepest decent adaptation method can be written as,

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla\xi(n) \qquad (2.22)$$

where μ is the step-size parameter and $\xi(n)$ is the mean square error at time n.

Now we assume that we have,

$$\hat{d}(n) = \mathbf{w}^T\mathbf{x}(n) \qquad (2.23)$$

$$\mathbf{R}_x = E\left\{\mathbf{x}(n)\mathbf{x}^T(n)\right\} \qquad (2.24)$$

$$\mathbf{r}_{dx} = E\left\{d(n)\mathbf{x}(n)\right\} \qquad (2.25)$$

The gradient of the mean square error function is,

$$\nabla\xi(n) = \nabla E\left\{\left|e(n)\right|^2\right\} = E\left\{\nabla\left|e(n)\right|^2\right\} = E\left\{e(n)\nabla e^*(n)\right\} \qquad (2.26)$$

And we know that,

$$\nabla e^*(n) = -\mathbf{x}^*(n)$$

Thus it yields,

$$\nabla\xi(n) = -E\left\{e(n)\mathbf{x}^*(n)\right\} \qquad (2.27)$$

In the case of stationary processes, if $x(n)$ and $d(n)$ are jointly WSS (*wide-sense stationary*) then,

$$E\left\{e(n)\mathbf{x}^*(n)\right\} = E\left\{d(n)\mathbf{x}^*(n)\right\} - E\left\{\mathbf{w}_n^T\mathbf{x}(n)\mathbf{x}^*(n)\right\} \qquad (2.28)$$

$$= \mathbf{r}_{dx} - \mathbf{R}_x\mathbf{w}(n)$$

Therefore,

$$\nabla\xi(n) = -\mathbf{r}_{dx} + \mathbf{R}_x\mathbf{w}(n) \qquad (2.29)$$

By considering the above two Equations (2.22) and (2.29), we have

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left[\mathbf{r}_{dx} - \mathbf{R}_x\mathbf{w}(n)\right] \qquad (2.30)$$

Now we can define a filter coefficients error vector as,

$$\tilde{\mathbf{w}}(n) = \mathbf{w}(n) - \mathbf{w}_0 \qquad (2.31)$$

where ,

- w_0 is the optimal least square error filter coefficient vector.

By Wiener filter, w_0 is given by,

$$\mathbf{w}_0 = \mathbf{R}_x^{-1}\mathbf{r}_{dx} \qquad (2.32)$$

After few mathematical arrangements in last three equations, (2.30) becomes,

$$\tilde{\mathbf{w}}(n+1) = [\mathbf{I} - \mu\mathbf{R}_x]\tilde{\mathbf{w}}(n) \qquad (2.33)$$

The step-size parameter μ controls the stability and rate of convergence of the adaptive filter. The filter shows instability if the value of μ is too large and low convergence rate if μ too small. The stability of the filter depends on the selection of step-size adaptive parameter μ and the autocorrelation matrix. The correlation matrix can be expressed in term of the matrices of eigenvectors and eigenvalues as,

$$\mathbf{R}_x = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \qquad (2.34)$$

Where,

- Q is orthonormal matrix of the eigenvectors of $\mathbf{R}_x$
- $\mathbf{\Lambda}$ is a diagonal matrix having diagonal elements corresponding to the eigenvalues of $\mathbf{R}_x$

By putting the value of $\mathbf{R}_x$ in Equation (2.34) into Equation (2.33) we obtain,

$$\tilde{\mathbf{w}}(n+1) = [\mathbf{I} - \mathbf{Q}\mathbf{\Lambda}Q^T]\tilde{\mathbf{w}}(n) \qquad (2.35)$$

Multiplying $\mathbf{Q}^T$ to both sides of Equation (2.35) and the using relations $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$ yields,

$$\mathbf{Q}^T\tilde{\mathbf{w}}(n+1) = [\mathbf{I} - \mu\mathbf{\Lambda}]\mathbf{Q}^T\tilde{\mathbf{w}}(n) \qquad (2.36)$$

Let $\mathbf{v}(n) = \mathbf{Q}^T\tilde{\mathbf{w}}(n)$,

So the Equation (2.36) becomes,

$$\mathbf{v}(n+1) = \left[\mathbf{I} - \mu\mathbf{\Lambda}\right]\mathbf{v}(n) \qquad (2.37)$$

Here, $\mathbf{I}$ and $\mathbf{\Lambda}$ are diagonal matrices. So the above equation can be written in term of individual elements of the error vector $\mathbf{v}(n)$ as,

$$v_k(n+1) = [1-\mu\lambda_k]v_k(n) \qquad (2.38)$$

where λ_k is the k^{th} eigenvalue of the autocorrelation of the filter input $x(n)$

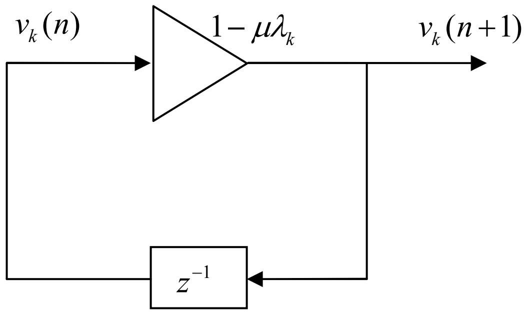


Figure II-5: *Feedback model of the variation of coefficient error with time*

By considering the Equation (2.38), we make a condition for stability for the process of adaptation and the coefficient error vector decay is,

$$-1 < 1-\mu\lambda_k < 1 \qquad (2.39)$$

Let's denote $\lambda_{\max}$ the maximum eigenvalue of the autocorrelation matrix, the limits of μ for stable adaptation is,

$$0 < \mu < \frac{2}{\lambda_{\max}} \qquad (2.40)$$

2.2.2. Least Mean Square (LMS) Algorithm

In 1959, Widow and Hoff [3] derived an algorithm whose name was Least Mean Square (LMS) algorithm and till now it is one of the best adaptive filtering algorithms. This algorithm is used widely for different application such as channel equalization and echo cancellation. This algorithm adjusts the coefficients of $w(n)$ of a filter in order to reduce the mean square error between the desired signal and output of the filter. This algorithm is basically the type of adaptive filter known as stochastic gradient-based algorithms. Why it's called stochastic gradient algorithm? Because in order to converge on the optimal Wiener solution, this algorithm use the gradient vector of the filter tap weights. This algorithm is also used due to its computational simplicity.

The equation below is LMS algorithm for updating the tap weights of the adaptive filter for each iteration.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}^*(n) \tag{2.41}$$

Where,

- $\mathbf{x}(n)$: input vector of time delayed input values.
- $\mathbf{w}(n)$: weight vector at time n.

μ is a step-size parameter and it controls the immediate change of the updating factor. It shows a great impact on the performance of the LMS algorithm in order to change its value. If the value of μ is so small then the adaptive filter takes long time to converge on the optimal solution and in case of large value the adaptive filter will be diverge and become unstable.

Derivation of the LMS algorithm:

The derivation of LMS algorithm is the development of the steepest decent method and also takes help from the theory of Wiener solution (optimal filter tap weights). This algorithm is basically using the formulas which updates the filter coefficients by using the tap weight vectors $\mathbf{w}$ and also update the gradient of the cost function accordingly to the filter tap weight coefficient vector $\nabla\xi(n)$. From Equation (2.22) in the steepest decent algorithm,

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla\xi(n)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu E\left\{e(n)\mathbf{x}^*(n)\right\} \tag{2.42}$$

In practice, the value of the expectation $E\left\{e(n)\mathbf{x}^*(n)\right\}$ is normally unknown, therefore we need to introduces the approximation or estimated as the sample mean,

$$\hat{E}\left\{e(n)\mathbf{x}^*(n)\right\} = \frac{1}{L}\sum_{l=0}^{L-1} e(n-l)\mathbf{x}^*(n-l) \tag{2.43}$$

With this estimate we obtain the updating weight vector as,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu}{L}\sum_{l=0}^{L-1} e(n-l)\mathbf{x}^*(n-l) \tag{2.44}$$

If we using one point sample mean (L=1) then,

$$\hat{E}\left\{e(n)\mathbf{x}^*(n)\right\} = e(n)\mathbf{x}^*(n) \tag{2.45}$$

And finally, the weight vector update equation become the simple form,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}^*(n) \tag{2.46}$$

2.2.3. Normalized Least Mean Square (NLMS) Algorithm

By using this normalized step-size parameter in Least Mean Square algorithm, this algorithm is known as Normalized Least Mean Square (NLMS) algorithm [5]. The step-size for computing the update weight vector is,

$$\mu(n) = \frac{\beta}{c + \left\|\mathbf{x}(n)\right\|^2} \tag{2.47}$$

Where,

- $\mu(n)$ is step-size parameter at sample n
- β is normalized step-size ($0 < \beta < 2$)
- c is safety factor (small positive constant)

Derivation of the NLMS algorithm:

Normalized Least Mean Square (NLMS) is actually derived from Least Mean Square (LMS) algorithm. The need to derive this NLMS algorithm is that the input signal power changes in time and due to this change the step-size between two adjacent coefficients of the filter will also change and also affect the convergence rate. Due to small signals this convergence rate will slow down and due to loud signals this convergence rate will increase and give an error. So to overcome this problem, try to adjust the step-size parameter with respect to the input signal power. Therefore the step-size parameter is said to be normalized.

When design the LMS adaptive filter, one difficulty we meet is the selection of the step-size parameter μ. For stationary processes, this algorithm converts in the limits:

$$0 < \mu < \frac{2}{\lambda_{\max}} \text{ And } 0 < \mu < \frac{2}{trace(\mathbf{R}_x)}$$

However, the auto-correlation $\mathbf{R}_x$ generally is unknown, for this reason, the maximum lambda $\lambda_{\max}$ and $\mathbf{R}_x$ are estimated in order to use the bounds. To solve this problem, one introduces new estimate of $trace(\mathbf{R}_x)$ as,

$$trace(\mathbf{R}_x) = (p+1)E\left\{|x(n)|^2\right\} \tag{2.48}$$

Where,

- $p = 0,1,2,...$
- $E\left\{|x(n)|^2\right\}$ is the power of input signal. It can be estimated by estimator:

$$\hat{E}\left\{|x(n)|^2\right\} = \frac{1}{p+1}\sum_{k=0}^{p}|x(n-k)|^2 \tag{2.49}$$

Therefore, the limits of step-size parameter will become,

$$0 < \mu < \frac{2}{(p+1)E\left\{\lfloor x(n) \rfloor^2\right\}} \tag{2.50}$$

Substitutes Equation (2.49) into Equation (2.50), one get the step-size parameter as,

$$0 < \mu < \frac{2}{\mathbf{x}^H(n)\mathbf{x}(n)} \tag{2.51}$$

For time-varying processes, one computes the step-size parameter in time (sample n),

$$\mu(n) = \frac{\beta}{\mathbf{x}^H(n)\mathbf{x}(n)} = \frac{\beta}{\|\mathbf{x}(n)\|^2} \tag{2.52}$$

Where,

- β is normalized step-size ($0 < \beta < 2$)

By replaced μ by $\mu(n)$ into the Equation (2.46) for updating the weight vector in LMS algorithm, we achieve a new algorithm was known as Normalized Least Mean Square (NLMS). The weight vector update now is,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)e(n)\mathbf{x}^*(n)$$

Or

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\|\mathbf{x}(n)\|^2}e(n)\mathbf{x}^*(n) \tag{2.53}$$

In the LMS algorithm, because the weight vector $\mathbf{w}(n)$ changes depending on the input signal $\mathbf{x}(n)$. Thus it will get the problem is called as *gradient noise amplification* [5] when $\mathbf{x}(n)$ is too large. However, by using NLMS algorithm we can avoid this problem. Take a look the Equation (2.53), when $\mathbf{x}(n)$ is very small the calculation of weight vector updating equation will be the big problem. For this reason, one implements the safety factor as,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{c + \|\mathbf{x}(n)\|^2} e(n)\mathbf{x}^*(n) \tag{2.54}$$

Where

- c is safety factor (small positive constant)

Finally, the Equation (2.54) is the weight vector updating equation for NLMS algorithm.

2.2.4. Recursive Least Square (RLS)

The RLS filter is a simple adaptive and time update version of wiener filter [12]. For non-stationary signals, this filter tracks the time variations but in case of stationary signals, the convergence behavior of this filter is the same as Wiener filter that it converges to the same optimal coefficients. This filter has fast convergence rate and it is widely used in the application such as echo cancellation, channel equalization, speech enhancement and radar where the filter should do fast changes in signal process. This adaptive algorithm is used due to following factors:

- Computational complexity
- Speed of convergence
- Minimum error at convergence
- Numerical stability
- Robustness

For RLS algorithm, we consider the following:

- $\mathbf{x}(n)$ is the discrete time array $M \times 1$ array input vector.
- $y(n) = \mathbf{w}^H \mathbf{x}(n)$ is the output signal.
- $d(n)$ is the desired signal.
- And w is the $M \times 1$ complex weight vector

The cost function $f_{n\partial}(w)$ at time instant n is given by,

$$f_n(\mathbf{w}) = \sum_{k=1}^{n} \lambda^{n-k} \left|\mathbf{w}^H \mathbf{x}(k) - d(k)\right|^2 + (\mathbf{w} - \mathbf{w}_0)^H \lambda^n \mathbf{R}_0 (\mathbf{w} - \mathbf{w}_0) \quad (2.55)$$

$$n = 1, 2, 3, ...$$

where,

- $\mathbf{w}_0$ and $\mathbf{R}_0$ are the initial chosen parameters
- λ is the real-positive constant $(0 < \lambda < 1)$

And we define the new function,

$$f_0(\mathbf{w}) = (\mathbf{w} - \mathbf{w}_0)^H \mathbf{R}_0 (\mathbf{w} - \mathbf{w}_0) \quad (2.56)$$

$$\mathbf{p}_0 = \mathbf{R}_0 \mathbf{w}_0$$

Now consider two column matrices,

$$\mathbf{A} = \begin{bmatrix} \lambda^{\frac{n-1}{2}} \mathbf{x}^H(1) \\ \vdots \\ \mathbf{x}^H(n) \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \lambda^{\frac{n-1}{2}} d^*(1) \\ \vdots \\ d^*(n) \end{bmatrix}$$

Now the cost function $f_n(w)$ could rewrite as,

$$f_n(\mathbf{w}) = (\mathbf{Aw} - \mathbf{b})^H (\mathbf{Aw} - \mathbf{b}) + (\mathbf{w} - \mathbf{w}_0)^H \lambda^n \mathbf{R}_0 (\mathbf{w} - \mathbf{w}_0) \quad (2.57)$$

When n is large ($n>M$), then $\lambda^n \to 0$, hence the second term of the Equation (2.57) will disappear and the least square error now is,

$$f_n(\mathbf{w}) = (\mathbf{Aw} - \mathbf{b})^H (\mathbf{Aw} - \mathbf{b}) \quad (2.58)$$

$\mathbf{w}_n$ will be a solution of over-determined linear system of equation $\mathbf{Aw} = \mathbf{b}$.

For otherwise, by differentiating the Equation (2.57), we have $\mathbf{w}_n$ as

$$\mathbf{w}_n = \mathbf{R}_n^{-1} \mathbf{p}_n \quad (2.59)$$

Where,

- $\mathbf{R}_n = \mathbf{A}^H \mathbf{A} + \lambda^n \mathbf{R}_0 = \sum_{k=1}^{n} \lambda^{n-k} \mathbf{x}(k) \mathbf{x}^H(k) + \lambda^n \mathbf{R}_0 \quad (2.60)$

- $\mathbf{p}_n = \mathbf{A}^H\mathbf{b} + \lambda^n\mathbf{p}_0 = \sum_{k=1}^{n}\lambda^{n-k}\mathbf{x}(k)d^*(k) + \lambda^n\mathbf{p}_0$ (2.61)

Then we obtain the recursive relations for $\mathbf{R}_n$ and $\mathbf{p}_n$ as,

$$\mathbf{R}_n = \sum_{k=1}^{n-1}\lambda^{n-k}\mathbf{x}(k)\mathbf{x}^H(k) + \mathbf{x}(n)\mathbf{x}^H(n) + \lambda^n\mathbf{R}_0$$

$$= \lambda\left(\sum_{k=1}^{n-1}\lambda^{n-1-k}\mathbf{x}(k)\mathbf{x}^H(k) + \lambda^{n-1}\mathbf{R}_0\right) + \mathbf{x}(n)\mathbf{x}^H(n)$$

$$= \lambda\mathbf{R}_{n-1} + \mathbf{x}(n)\mathbf{x}^H(n) \qquad (for\ n \geq 1) \tag{2.62}$$

$$\mathbf{p}_n = \sum_{k=1}^{n-1}\lambda^{n-k}\mathbf{x}(k)d^*(k) + \mathbf{x}(n)d^*(n) + \lambda^n\mathbf{p}_0$$

$$= \lambda\left(\sum_{k=1}^{n-1}\lambda^{n-1-k}\mathbf{x}(k)d^*(k) + \lambda^{n-1}\mathbf{p}_0\right) + \mathbf{x}(n)d^*(n)$$

$$= \lambda\mathbf{p}_{n-1} + \mathbf{x}(n)d^*(n) \qquad (for\ n \geq 1) \tag{2.63}$$

Rewrite the Equation (2.62) for recursive relation of $\mathbf{R}_n$. We have,

$$\lambda^{-1}\mathbf{R}_n = \mathbf{R}_{n-1} + \mathbf{x}(n)\lambda^{-1}\mathbf{x}^H(n) \tag{2.64}$$

Using matrix inversion, suppose that A and B are two positive-definite matrices related by,

$$\mathbf{B}^{-1} = \mathbf{A}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H \tag{2.65}$$

So the relations of these matrices are:

$\mathbf{B}^{-1} = \lambda^{-1}\mathbf{R}_n$, $\mathbf{A}^{-1} = \mathbf{R}_{n-1}$, $\mathbf{C} = \mathbf{x}(n)$ and $\mathbf{D}^{-1} = \lambda^{-1}$

Now by taking the inverse of $\lambda^{-1}\mathbf{R}_n$ we have,

$$\lambda\mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - \mathbf{R}_{n-1}^{-1}\mathbf{x}(n)\frac{1}{\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{x}(n) + \lambda}\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1} \tag{2.66}$$

Therefore,

$$\mathbf{R}_n^{-1} = \lambda^{-1}\mathbf{R}_{n-1}^{-1} - \frac{\lambda^{-1}\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}}{\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{x}(n) + \lambda} \tag{2.67}$$

Multiplying both sides with $x(n)$, the useful relation $\mathbf{R}_n^{-1}\mathbf{x}(n)$ is,

$$\mathbf{R}_n^{-1}\mathbf{x}(n) = \lambda^{-1}\mathbf{R}_{n-1}^{-1}\mathbf{x}(n) - \frac{\lambda^{-1}\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)}{\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)+\lambda}$$

$$= \frac{\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)}{\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)+\lambda} \qquad (2.68)$$

Now we will use the above equations to attain the recursive relation for least square solution $\mathbf{w}_n$ as,

$$\mathbf{w}_n = \mathbf{R}_n^{-1}\mathbf{p}_\circ = \mathbf{R}_n^{-1}(\lambda\mathbf{p}_{n-1} + \mathbf{x}(n)d^*(n))$$

$$= \mathbf{R}_n^{-1}\lambda\mathbf{p}_{n-1} + \mathbf{R}_n^{-1}\mathbf{x}(n)d^*(n)$$

$$= \left(\lambda^{-1}\mathbf{R}_{n-1}^{-1} - \frac{\lambda^{-1}\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}}{\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)+\lambda}\right)\lambda\mathbf{p}_{n-1} + \mathbf{R}_n^{-1}\mathbf{x}(n)d^*(n)$$

$$= \mathbf{R}_{n-1}^{-1}\mathbf{p}_{n-1} - \frac{\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{p}_{n-1}}{\mathbf{x}^H(n)\mathbf{R}_{n-1}^{-1}\mathbf{x}(n)+\lambda} + \mathbf{R}_n^{-1}\mathbf{x}(n)d^*(n)$$

$$= \mathbf{w}_{n-1} - \mathbf{R}_n^{-1}\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_{n-1} + \mathbf{R}_n^{-1}\mathbf{x}(n)d^*(n)$$

$$= \mathbf{w}_{n-1} + \mathbf{R}_n^{-1}\mathbf{x}(n)(d^*(n) - \mathbf{x}^H(n)\mathbf{w}_{n-1}) \qquad (2.69)$$

Finally, we know that the error signal is $\varepsilon(n) = d(n) - \mathbf{w}_{n-1}^H\mathbf{x}(n)$, substitute this term into Equation (2.69), we achieve the weight vector update equation of RLS algorithm as following,

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{R}_n^{-1}\mathbf{x}(n)\varepsilon^*(n) \qquad (2.70)$$

2.3. DOUBLE-TALK DETECTOR (DTD)

In Acoustic Echo Cancellation, the most difficult problem is to handle with the situation of Double-talk presence. Double-talk occurs when far-end and near-end talk at the same time, as a result, the far-end speech signal is corrupted by near-end signal. To solve this problem, one introduces the Double-talk Detector. The task of DTD is freezes the adaptation step during filtering algorithm in case of near-end speech present to avoid the divergence of adaptive algorithm. Without DTD, when the near-end talking would make the system estimation process fail and produce extremely erroneous results.

Now we see the Figure (II-6), when near-end speech is not present ($v(n) = 0$) then the adaptive algorithm will quickly converse to an estimate echo path. This is the best case of canceling echo. But when near-end speech present ($v(n) \neq 0$) then this signal could influence to the adaptation of the filter and cause the divergence. The process of adaptive algorithm will be incorrect and the echo can not be removed.

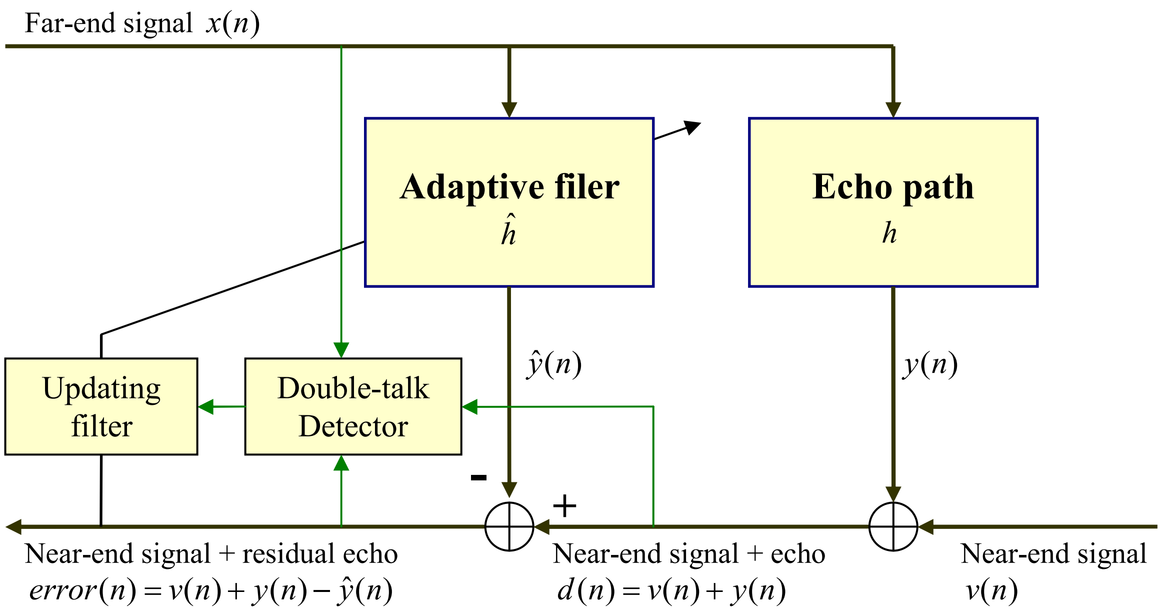


Figure II-6: *Double-talk detector with AEC*

By implementing the DTD and Updating filter blocks in the figure above, the DTD will estimate the statistic decision may depend on far-end speech, near-end speech and error signal. After that, it will compare to the threshold to make the DTD decision to control the Updating filter (freeze the adaptation or not). The "Updating filter" block here has the meaning as a switch (on or off) which permits to update weight vector or not.

There are several methods of DTD, one can use the basic algorithm as Geigel, one bases on the cross correlation calculations (Benesty and Normalized Cross-Correlation algorithms) and another method is Variance Impulse Response.

2.3.1. The Geigel algorithm

One simple algorithm is introduced by A.A. Geigel [9]. His approach is that first measure the power of the received signal (microphone signal) and then compares this power to the power of the far-end signal. Due to damping the signal by room acoustic filter, as a result the power of the received signal containing only the echo will be lesser than the signal consisting of echo and a near-end speaker. This is known as *Geigel* Double-talk detector. The decision variable for this algorithm is,

$$\xi_G(t) = \frac{\max\left\{|x(t)|,\ldots,|x(t-L+1)|\right\}}{|d(t)|} \tag{2.71}$$

Where,

- L is length of adaptive filter

Make the comparison this value to the threshold T_G. If $\xi_G(t)$ is greater than the preset threshold, it is supposed that doubletalk is present and otherwise is not. That is mean:

$$Decision = \begin{Bmatrix} \xi_G(t) < T_G & doubletalk \\ \xi_G(t) > T_G & no-doubletalk \end{Bmatrix}$$

The selection of T_G requires to be chosen carefully because it strongly affect the performance of the detector. The Geigel detector has the benefit of being computationally simple and requiring very little memory. This detection approach is based on a waveform level comparison between microphone signal $d(n)$ and the far-end signal $x(n)$. And also assume that the near-end speech signal $v(n)$ in the microphone signal will be stronger than the echo $y(n) = \mathbf{h}^{\mathrm{T}}\mathbf{x}(n)$. For AEC, it is difficult to set threshold which works in any situation because the loss through the acoustic echo path depends on different factors. In general, this detector has quite poor performance.

2.3.2. The Cross-correlation (Benesty) algorithm

Ye and Wu [8] firstly introduced the idea by using cross-correlation vector between the far-end signal $x(n)$ and the error signal $e(n)$ for doubletalk detection which is given as,

$$r_{ex} = E\left\{e(n)\mathbf{x}(n)^T\right\} \tag{2.72}$$

Where,

- r_{ex} : is the cross-correlation vector between far-end and error signal.

But Benesty [8] worked on this with different approach and he claimed that the above approach does not work well for doubletalk detection. He mentioned that both near-end speech $v(n)$ and the far-end speech signal $x(n)$ are independent and assume that all the signals are zero mean.

According to him, the cross-correlation r_{xd} between far-end signal and microphone signal will be used to calculate the decision statistic.

$$\begin{aligned} r_{xd} &= E\left\{\mathbf{x}(n)d(n)^T\right\} \\ &= E\left\{\mathbf{x}(n)\left(y(n)+v(n)\right)^T\right\} \\ &= E\left\{\mathbf{x}(n)\left(\mathbf{h}^T\mathbf{x}(n)\right)^T\right\} \\ &= \mathbf{R}_x\mathbf{h} \end{aligned} \tag{2.73}$$

Where,

- $\mathbf{R}_x = E\left\{\mathbf{x}(n)\mathbf{x}(n)^T\right\}$ is the autocorrelation vector of far-end signal.

Benesty's decision statistic for double-talk detection is,

$$\xi_{CC} = \sqrt{r_{xd}^T(\sigma_d^2\mathbf{R}_x)^{-1}r_{xd}} \tag{2.74}$$

In this equation, the variance of the microphone signal σ_d^2 is,

$$\begin{aligned} \sigma_d^2 &= E\left\{d(n)d(n)^T\right\} \\ &= E\left\{\left(y(n)+\mathbf{v}(n)\right)\left(y(n)+\mathbf{v}(n)\right)^T\right\} \\ &= E\left\{y(n)y(n)^T\right\} + E\left\{\mathbf{v}(n)\mathbf{v}(n)^T\right\} \\ &= E\left\{\mathbf{h}^T\mathbf{x}(n)\left(\mathbf{h}^T\mathbf{x}(n)\right)^T\right\} + \sigma_v^2 \\ &= \mathbf{h}^T\mathbf{R}_x\mathbf{h} + \sigma_v^2 \end{aligned} \tag{2.75}$$

Where,

- σ_v^2 is variance of the near-end speech

Finally, the Equation (2.74) of the decision statistic becomes,

$$\xi_{Benesty} = \xi_{CC}^2 = r_{xd}^T(\sigma_d^2\mathbf{R}_x)^{-1}r_{xd}$$

$$= \frac{\mathbf{h}^T\mathbf{R}_x\mathbf{R}_x\mathbf{h}}{(\mathbf{h}^T\mathbf{R}_x\mathbf{h}+\sigma_v^2)\mathbf{R}_x}$$

$$= \frac{\mathbf{h}^T\mathbf{R}_x\mathbf{h}}{\mathbf{h}^T\mathbf{R}_x\mathbf{h}+\sigma_v^2} \tag{2.76}$$

Therefore, observe the above equation, easily to see that,

- If near-end speech is present ($v(n)=0$), then $\xi_{Benesty} \approx 1$
- If near-end speech is not present ($v(n)\neq 0$), then $\xi_{Benesty} < 1$

Thus, finally we get the double-talk decisions as,

$$Decision = \begin{Bmatrix} \xi_{Benesty}(t) < T & doubletalk \\ \xi_{Benesty}(t) > T & no-doubletalk \end{Bmatrix}$$

Where,

- T is a threshold with the chosen value approximately is 1.

2.2.3. Normalized cross-correlation (NCC) algorithm

Another method here we will discuss for doubletalk detection is the Normalized Cross-Correlation algorithm [8]. The NCC algorithm computes the decision statistic depending on the relations of microphone signal and error signal. It can be approached by considering the values of variance of near-end signal and cross-correlation between error signal and microphone signal.

The cross-correlation r_{ed} between the error signal $e(n)$ and microphone signal $d(n)$ which is given as,

$$r_{ed} = E\{e(n)d(n)\}$$

$$= E\left[\left(y(n)+\mathbf{v}(n)-\mathbf{h}^T\mathbf{x}(n)\right)(y(n)+v(n))^T\right]$$

$$= E\left[\left(\mathbf{h}^T\mathbf{x}(n)-\hat{\mathbf{h}}^T\mathbf{x}(n)+v(n)\right)(\mathbf{h}^T\mathbf{x}(n)+v(n))^T\right]$$

$$= E\left[\left(\mathbf{h}^\mathbf{T}\mathbf{x}(n)-\hat{\mathbf{h}}^T\mathbf{x}(n)\right)\mathbf{x}(n)^T\mathbf{h}+v(n)v(n)^T\right]$$

$$=\left(\mathbf{h}^{\mathrm{T}}-\hat{\mathbf{h}}^{T}\right)\mathbf{R}_x\mathbf{h}+\sigma_v^2 \tag{2.77}$$

Now one introduces the normalized decision statistic as,

$$\xi_{NCC}=1-\frac{r_{ed}}{\sigma_d^2} \tag{2.78}$$

By substituting the values of r_{em} and σ_m^2 from above relations into Equation (2.78), we have,

$$\xi_{NCC}=1-\frac{\left(\mathbf{h}^{\mathrm{T}}-\hat{\mathbf{h}}^{T}\right)\mathbf{R}_x\mathbf{h}+\sigma_v^2}{\hat{\mathbf{h}}^{T}\mathbf{R}_x\mathbf{h}+\sigma_v^2}$$

$$=\frac{\hat{\mathbf{h}}^{T}\mathbf{R}_x\mathbf{h}}{\mathbf{h}^{\mathrm{T}}\mathbf{R}_x\mathbf{h}+\sigma_v^2} \tag{2.79}$$

Look at the Equation (2.79), when the adaptive filter works well to converge to an estimate echo path $\hat{\mathbf{h}}$ that approximately equal to the true echo path $\mathbf{h}$. Therefore, easily to obtain bellow conclusion,

- If near-end speech is present ($v(n)=0$), then $\xi_{MECC}\approx 1$
- If near-end speech is not present ($v(n)\neq 0$), then $\xi_{MECC}<1$

Thus, finally we get the double-talk decisions as,

$$Decision=\begin{Bmatrix}\xi_{NCC}(t)<T & doubletalk \\ \xi_{NCC}(t)>T & no-doubletalk\end{Bmatrix}$$

Where, T is a threshold with the chosen value approximately is 1.

The values of r_{ed} and σ_v^2 are not available in practice, so we define the new estimated decision statistic as,

$$\xi_{NCC}=1-\frac{\hat{r}_{ed}}{\hat{\sigma}_d^2} \tag{2.80}$$

Where,

- $\hat{r}_{ed}$ is the estimate of r_{ed}
- $\hat{\sigma}_d^2$ is the estimate of σ_d^2

We can found these estimates by using the exponential recursive weighting algorithm.

$$\hat{r}_{ed}(n) = \lambda \hat{r}_{ed}(n-1) + (1-\lambda)e(n)d^T(n) \qquad (2.81)$$

$$\hat{\sigma}_d^2(n) = \lambda \hat{\sigma}_d^2(n-1) + (1-\lambda)d(n)d^T(n) \qquad (2.82)$$

Where,

- $e(n)$ is the captured cancellation error sample at time n
- $d(n)$ is the captured microphone signal sample at time n
- λ is the exponential weighting factor ($\lambda < 1$ and $\lambda \approx 1$)

2.3. FREQUENCY-DOMAIN ACOUSTIC ECHO CANCELLATION

Above all algorithms that we described in this thesis are time domain algorithms. They deals with low frequencies and we can get good result in case of acoustic echo cancellation for low frequency signals. But when we deal with high frequencies then we get good result by implementing frequency domain adaptive algorithm. The main advantages of frequency domain adaptive algorithm is the fast convergence rate especially when we are dealing with speech signals and second is the low computational complexity due to the efficiency of block processing in connection with discrete Fourier transform (DFT). The frequency domain adaptive filter has two basic types [10]:

1. Gradient constrained frequency domain adaptive filter.
2. Unconstrained frequency domain adaptive filter.

2.2.1. The generic frequency domain echo canceller

In this thesis, we focus on unconstrained frequency domain adaptive filter [10]. This filter has low computational complexity and it can converge to the Wiener solution when the length of the unknown system is less than half of the block size of DFT. This algorithm is based on overlap-save sectioning with the DFT and its robustness is based on a nonlinear function that provides scaling of the reference and error signal levels. Due to this algorithm, we get good results without time-variant threshold estimators. This algorithm is useful for the applications of echo cancellation.

In the below figure, $x(n)$ is the far-end speech signal with discrete time index n and after passing through the room echo path this signal is picked up by microphone. The room impulse response h is given by,

$$h = [h_1, h_2, \ldots, h_L]^T \tag{2.83}$$

where L is the length of the adaptive filter. The microphone signal or the output signal $y(n)$ is given as,

$$y(n) = \mathbf{h}^T \mathbf{R}\mathbf{x}(n) + v(n) + w(n) \tag{2.84}$$

Where,

- $v(n)$ is the near-end speech signal.
- $w(n)$ is the ambient noise.
- $\mathbf{x}(n) = [x(n-L+1), \ldots, x(n)]^T$.
- $\mathbf{R}$ is the matrix that reverses the order of the elements of $\mathbf{x}(n)$.

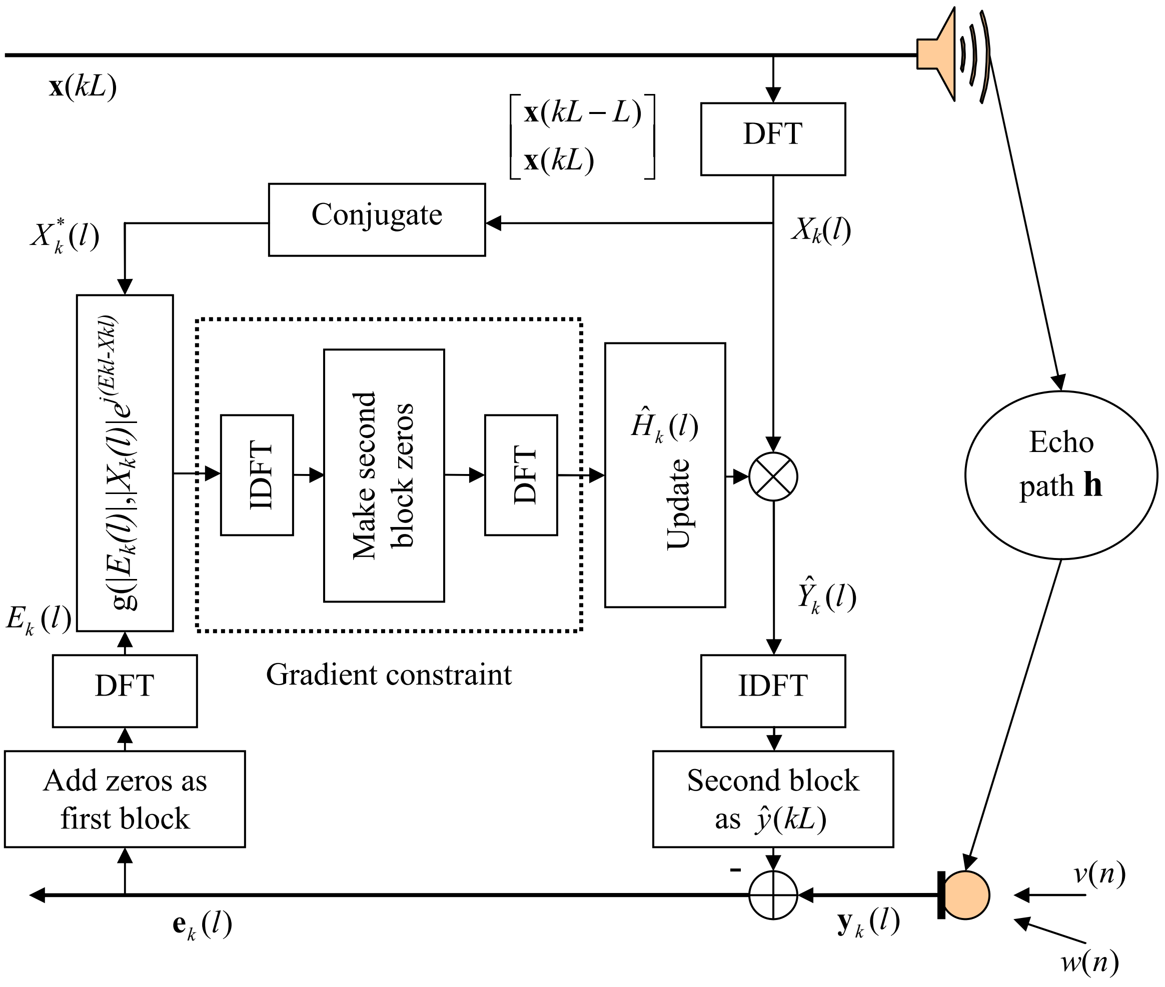


Figure II-7: *Frequency domain echo canceller*

The impulse response $\mathbf{h}$ is assumed to be fixed or vary slowly with the convergence rate of adaptive filter. The transformed far-end signal in l-th frequency bin at k-th step is $X_k(l)$ and it is an element of DFT of $\left[x^T(kL-L), x^T(kL)\right]^T$. By increasing the value of L, the index k is incremented after every time n and $l = 0,...,2L-1$.

The coefficient of the filter for k and l is $\hat{H}_k(l)$, so the output signal is given as

$$\hat{Y}_k(l) = \hat{H}_k(l) X_k(l) \tag{2.85}$$

The echo replica $\hat{y}(kl)$ corresponds to the last L elements of the inverse DFT (IDFT) of $\left[\hat{Y}_k(0), \hat{Y}_k(1), ..., \hat{Y}_k(2L-1)\right]^T$. And the error signal becomes,

$$e(kL) = y(kL) - \hat{y}(kL) \tag{2.86}$$

Where,

- $y(kL) = y(kL-L+1), ..., y(kL)$

In time-domain the error signal and the filter output are scalars whereas in frequency domain these are vectors.

The transformed error signal $E_k(l)$ is an element of DFT of $\left[z^T, e^T(kL)\right]^T$, where z is an $L \times 1$ zero vector.

Here we are focusing on unconstrained case. So by neglecting the gradient constrained in figure and the updating equation for $\hat{H}_k(l)$ is,

$$\hat{H}_{k+1}(l) = \hat{H}_k(l) + \mu . g(|E_k(l)|, |X_k(l)|) e^{j(\theta_{Ekl} - \theta_{Xkl})} \tag{2.87}$$

Where,

- θ_{Ekl} and θ_{Xkl} are the phases of $E_k(l)$ and $X_k(l)$.
- $g\left(|E_k(l)|, |X_k(l)|\right)$ is an arbitrary function of $|E_k(l)|$ and $|X_k(l)|$.
- μ is the step-size parameter depend on $g\left(|E_k(l)|, |X_k(l)|\right)$.

2.2.2. Sub-band adaptive filter

The basic idea of a sub-band [15] decomposition approach is its increase in convergence speed in comparison to a full-band solution, especially when extremely long FIR filters are being adapted. This is due to a reduced spectral magnitude range, i.e. sub-band filtering has a de-correlating effect because colored input signals are decomposed into sub-bands with ''whiter'' sub-spectra.

Figure (II-8) depicts the sub-band adaptive filtering. Using analysis filter banks P(Z) the original signal from far-end signal and near-end signal, microphone signal are decomposed by subdividing their spectra into smaller intervals (x_0(n),x_1(n),…). Adaptive filtering is then performed in these sub-bands by a set of independent filters $(\vec{h_0}(n), \vec{h_1}(n), ...)$. The outputs of these filters are subsequently combined using a synthesis filter bank Q(z) to reconstruct the full-band output.

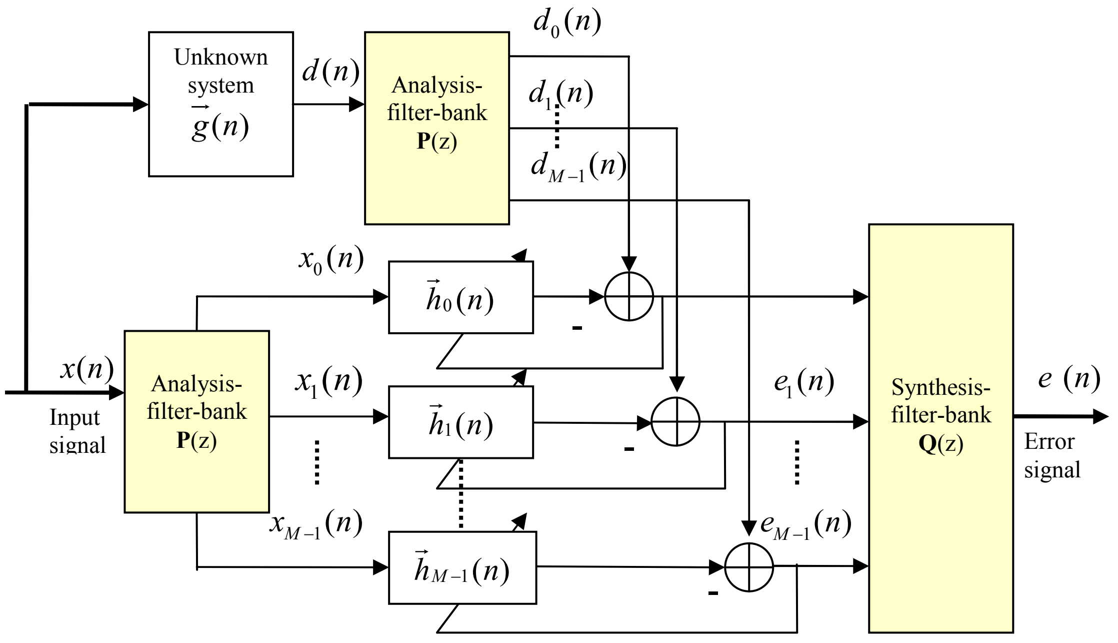


Figure II-8: Sub-band adaptive filtering (SAF) for M sub-bands

The width of each sub-band is reduced because the sampling frequency for each filter can be lowered. Consequently the sub-band adaptive filters need fewer taps in comparison to full-band solutions to cover the same time interval and are updated at a lower rate. This leads to a significant reduction of computational complexity.

Because linear group delays are required for sub-band adaptive filtering, only non-recursive filters are allowed for the filter banks.

Filter Bank Structure:

Because the quality of sub-band separation is highly significant for the obtained decimation rate and for the convergence behavior of the adaptive filters in sub-band, the design of analysis and synthesis filter banks [15] is the determining factor for the quality and efficiency of the overall system.

The following figures depict the analysis filter bank and synthesis filter bank. To ease the processing, down-sampling $(L\downarrow)$ and up-sampling $(L\uparrow)$ can be inserted between the analysis and synthesis filter banks.

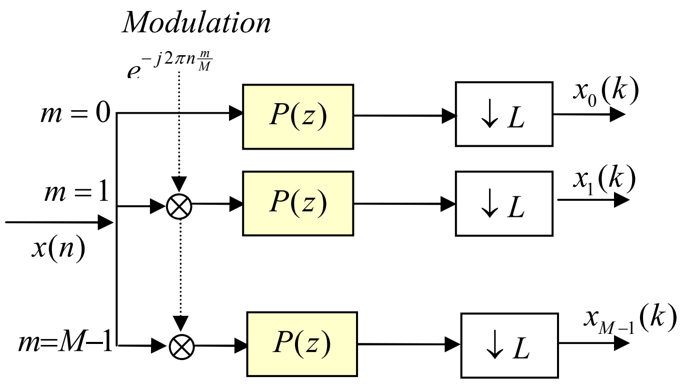


Figure II-9: *Analysis filter bank*

For sub-band separation and recombination we use DFT filter banks. To subdivide the sequence x(n) (apart from the low-frequency part) any part of the spectrum centered around the frequencies $\omega=\omega_m$ (for m=0,1,…,M-1) are shifted into the base-band by multiplying x(n) with the complex sinusoid $e^{-j\omega_m n}$ (with $\omega_m = 2\pi\frac{m}{M}$).

The ideal filters have unit magnitude and zero-phase in the pass-band while zero for the stop-band magnitude. The choice is to use FIR filters that have linear phase, but not ideal magnitude requirements.

The synthesis filter bank design reduces also to the design of a signal synthesis prototype filter $Q(z)$. Since we always use FIR sub-band filters and sub-band models, residual errors are unavoidable. This implies that in the design of a sub-band identification system, there is a tradeoff between asymptotic residual error and computational cost.

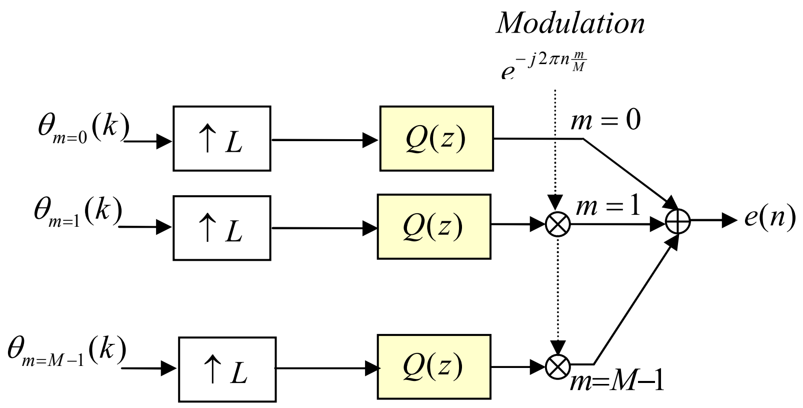


Figure II-10: *Synthesis filter bank*

CHAPTER III : SIMULATION

In the previous chapters above provided us the detail theory about the Acoustic Echo Cancellation including Algorithms of Adaptive filter, Double-talk Detection and other issues.

This chapter will perform these ideas to simulate the topic's problems by using the software environment (MATLAB).

3.1. GENERAL SETUP OF SIMULATION

3.1.1. Setup of the Simulation

1. MATLAB

MATLAB is a numerical computing environment that especially effective to calculate and simulate the technical problems. This programming language is very powerful allows matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with other programming languages (C, C++, Fortran and Java).

One of the most beneficial features is graphical visualization which helps us have confidence in results by monitoring and analyzing resultant plots.

In addition, MATLAB implement Simulink, the software package models, simulates, and analyzes dynamic systems. It enables us to pose a question about a system, model the system, and see what happens.

For our simulation purpose, MATLAB actually is necessary and effective software to attain the convincible results because of some reasons as following:

- Easy to record audio signals of the far-end and near-end speeches. These data are indispensable of the simulation.
- Matrix calculation is very important since data was processed as the matrix formats.
- Easy to monitor the results by plotting desired graphs. Especially, we need to hear the resultant sounds – By MATLAB, it is simple to achieve.
- The structure of the commands is suitable to compute with Signal Processing.

2. Requirements during the simulation

- This simulation tries to perform the tasks of the acoustic echo canceller and double-talk detector at the near-end conference room. We assume that both far-end and near-end rooms are the same characteristics (size, acoustic features). In the case of the perfectively performance of the far-end echo canceller, we only try to do the task of near-end room.

- The speech signals (including far-end and near-end signals) were recorded by MATLAB software at the sampling rate of 8 kHz. The speech signal is the audio signal contains the frequencies between 300Hz-3400Hz. Because of the sampling theorem (Nyquist–Shannon sampling theorem), the analog signal will reconstruct perfectly from the sequence of samples if the sampling rate exceeds 2B (B is highest frequency of the analog signal). Thus by using sampling rate fs of 8000Hz, we will satisfy to the sampling theorem (fs=8000Hz>2B=3400x2=6800Hz).

- For our simulation, the duration of the signals is 20 seconds (160.000 samples) which can express 4 cases (5 seconds for each case, respective to 40.000 samples) of the communication between far-end and near-end in the teleconference system. These signals are plotted as the figures below. 4 cases in teleconference are:

1. Far-end talks only
2. Double talk
3. Near-end talks only
4. Both of them are silent

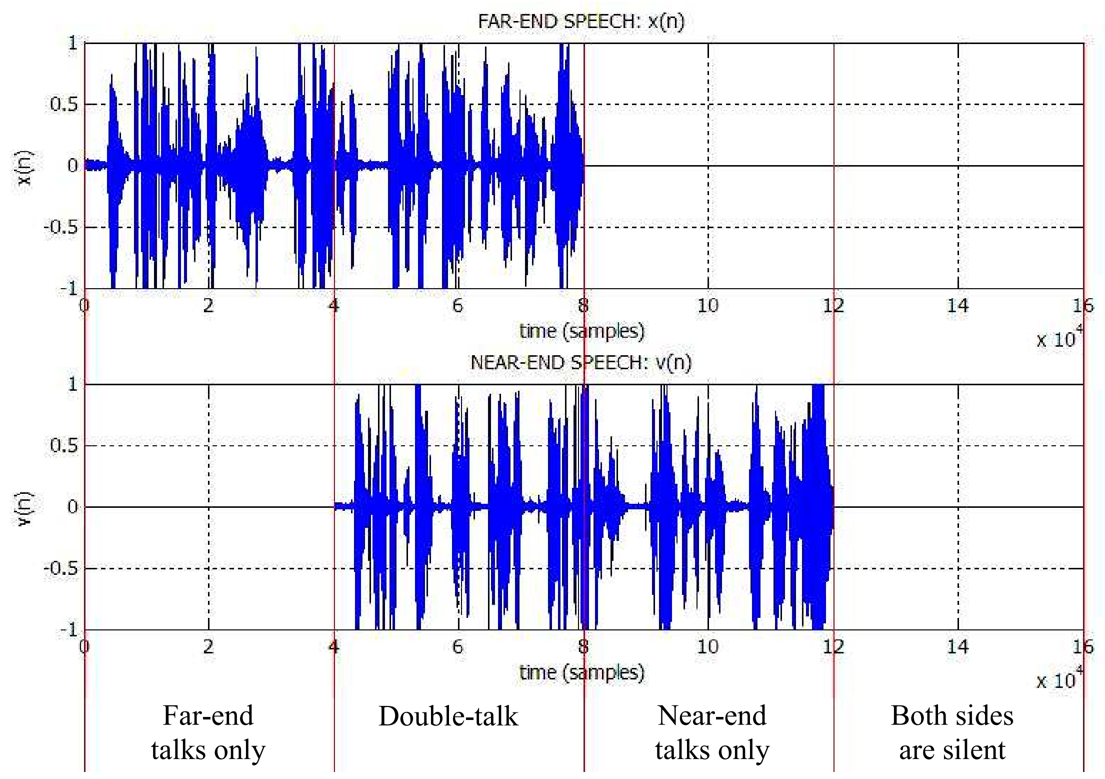


Figure III-1: *Far-end and near-end speeches*

- The background noise and the ambient noise were generated by MATLAB as a white noise that has a zero mean. The suitable noise we used here compares to the echo so that the Signal to Noise Ration (SNR) is approximately 45dB.

- The process of the simulation will be performed in off-line mode, i.e. the performance of the acoustic echo canceller and double-talk detector will work in MATLAB with the recorded speech and measured room impulse response.

3.1.2. Flowchart of the AEC algorithm

The flowchart of the AEC algorithm is very important for us to orient all steps we need to do in the simulation. This is shown in the figure next page.

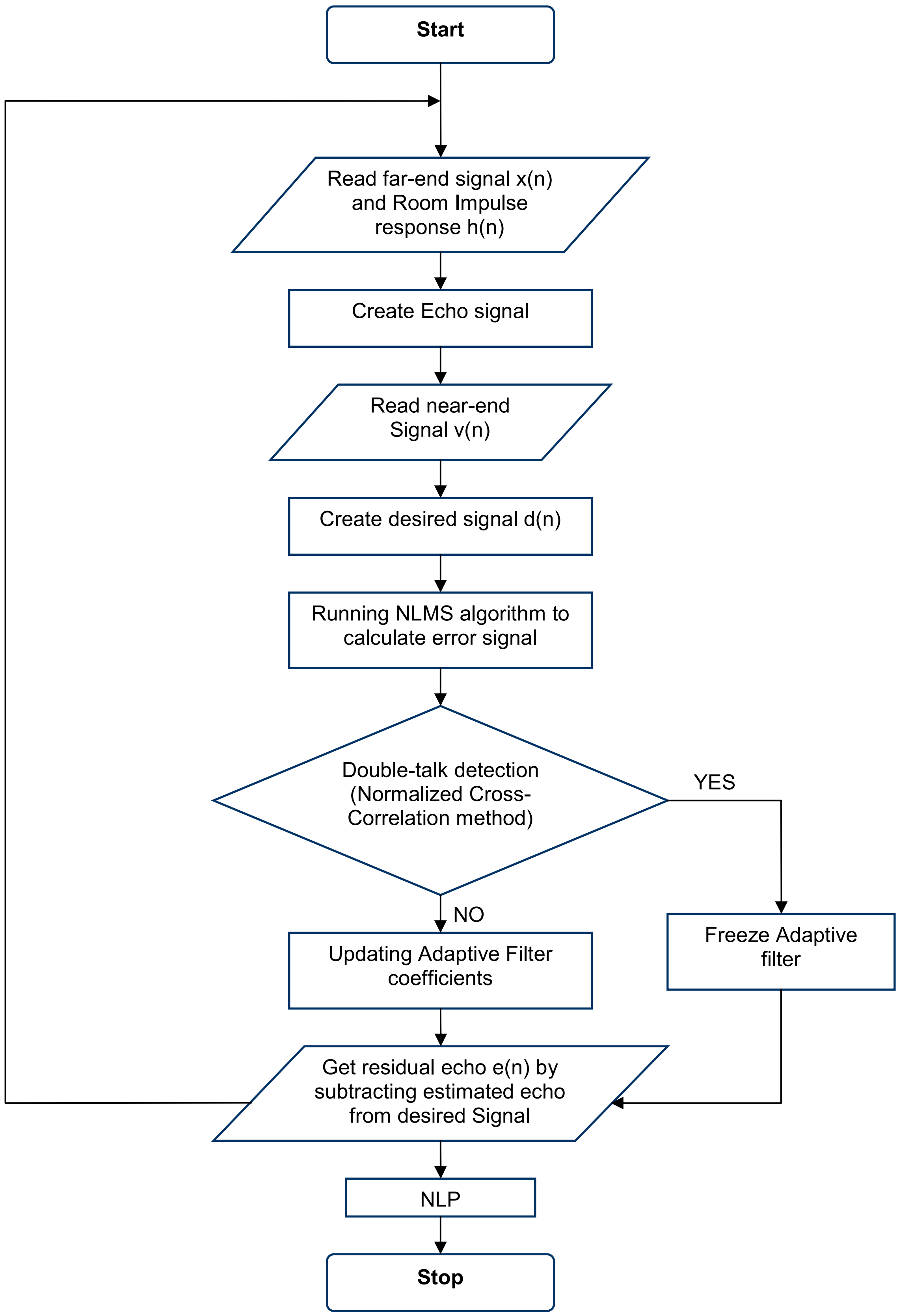

Start
Read far-end signal x(n) and Room Impulse response h(n)
Create Echo signal
Read near-end Signal v(n)
Create desired signal d(n)
Running NLMS algorithm to calculate error signal
Double-talk detection (Normalized Cross-Correlation method)
YES
NO
Updating Adaptive Filter coefficients
Freeze Adaptive filter
Get residual echo e(n) by subtracting estimated echo from desired Signal
NLP
Stop

Figure III-2: *Flowchart of acoustic echo cancellation algorithm*

3.2. MEASURE ROOM IMPULSE RESPONSE (RIR)

3.2.1. Why we must to measure RIR

In real time performance, the Adaptive Filter works to estimate the true value of impulse response $h(n)$ of the specific room. Therefore, if we have the exact Room Impulse Response, we can compare our result $\hat{h}(n)$ to this to make sure it is correct or not. In addition, by comparing to the true impulse response we could adjust the value of the factors to get the correct convergence of adaptive filter.

The acoustic characteristic of different rooms is different. They are frequency response, cumulative spectral decay, energy decay and reverberation characteristics, they depend on the three main factors:

1. Size of the room.
2. Constructing materials of the room (hard wood, concrete, ceramics…).
3. Objects inside the room (tables, chairs, people…).

3.2.2. Method of measuring RIR

There are several methods to measure RIR [13], we can use various excitation signals such as: white noise, pink noise, Dirac pulse, swept-sine and so on.

To achieve the room impulse response, we need to record the response (microphone signal) of the excitation signals (loud speaker signal). Impulse response may be obtained by direct de-convolution or by spectral division between the spectrum of the response and the spectrum of the excitation.

The chosen method in this simulation is measuring the room impulse response by using white noise signal as excitation. The reason is this signal contains equal amounts of energy for all frequencies, thus it will be good for demonstrating the frequencies of the speech that are used during the simulation. Moreover, this method only requires us with the simple equipments: one computer is connected to microphone and loud speaker. And after that, all calculations of the impulse response are performed easily by the MATLAB software.

3.2.3. Result

Our experiment was done under the setup as follow:

- The study room at the Vaxjo University's library is acoustically isolated with dimensions of 4m x 5m x 3m.
- The microphone was located on a 1m high table and far from the loud speaker the distant of 1m.
- The white noise was generated at the sampling frequency of 8000Hz and 16bits resolution.
- The recorded sounds were processed off-line using MATLAB.

Because the results are different for different times we measure. For this reason, to obtain the exact approximation of the room impulse response we took several times of measurements and got the average of them. In our experiment, we took 15 times of measurements and calculated the mean of them by using MATLAB.

To obtain the real room impulse response, we must take approximated 1 second (8000 taps) since it is depend on the reverberation factor. The graph below will illustrate this,

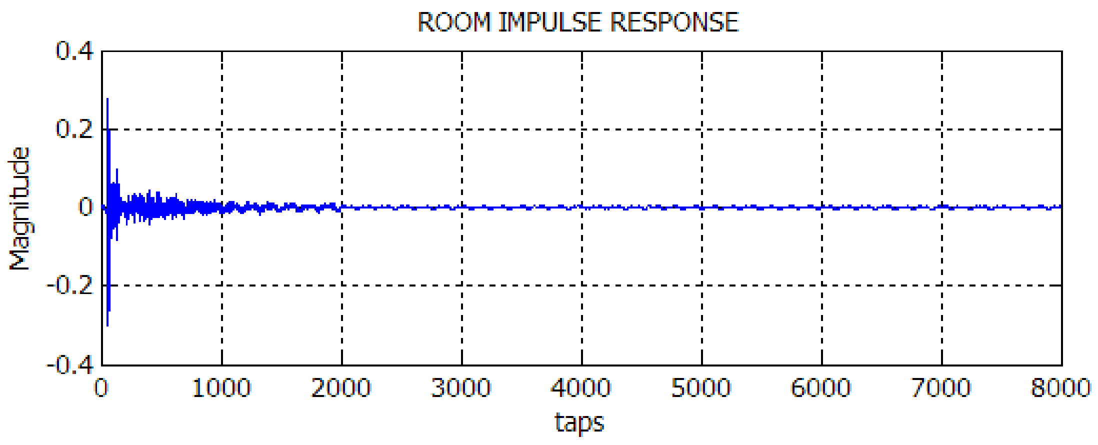


Figure III-3: *Room impulse response is measured with 8000 taps length.*

For this simulation, to make the simulation in MATLAB simpler, thus we measured the room impulse response with 128 taps length (16ms).

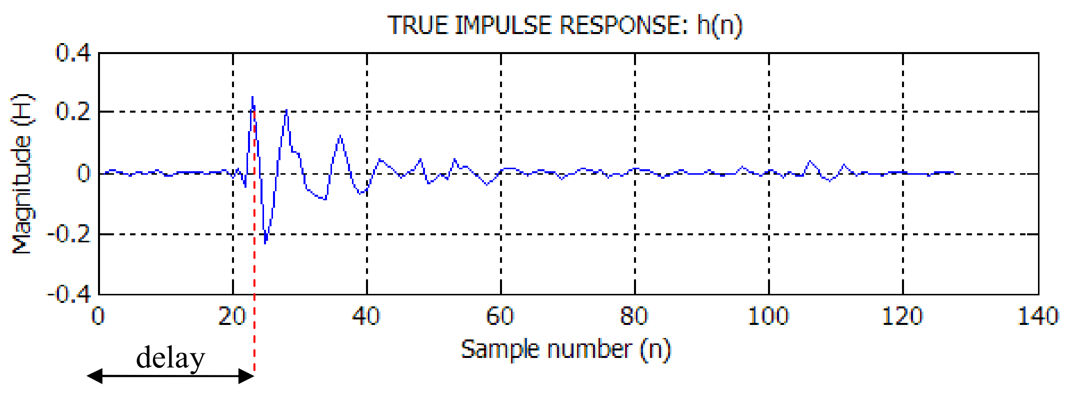


Figure III-4: *The room impulse response (128 taps length) is used in this simulation.*

As the figure's illustration, the length of the filter is approximately 16ms (respective to 128 taps length) and the delay is 3ms (24 taps from beginning to the first maximum magnitude)

The delay of 4ms is the approximate value we can compute from the distance (d=1m) between loud speaker and microphone.

$$\text{delay}=\text{distance/speed of sound}=1/343\approx 3\ [\text{ms}]$$

3.3. EXPLANATION OF MATLAB CODE

3.3.1. Adaptive filter algorithms

In these code segments, the purpose we need to obtain is processing and performing the LMS and NLMS algorithms.

The comparison below between the theory and the simulation parts (MATLAB code) may bring the better understanding about not only the algorithm but the programming work.

The summary of the algorithms (LMS, NLMS) and the code segments are demonstrated in the tables below.

1. Least Mean Square (LMS) algorithm

LMS algorithm	
Initial Conditions:	$0 < \mu < 1$ Length of adaptive filter: L Input vector: $x_{L,1} = [0,0,...,0]^T$ Weight vector: $w_{L,1} = [0,0,...,0]^T$
For each instant of time, n = 1, 2,…, compute:	
Output signal:	$y(n) = w^T(n)x(n)$
Estimation Error:	$e(n) = d(n) - y(n)$
Tap-Weight Adaptation:	$w(n+1) = w(n) + 2\mu x(n)e(n)$

Table III-1: *Summary of LMS algorithm*

MATLAB code	

Initial Conditions:	`mu=0.014;`
	`L=length(h);     %the same length of RIR`
	`w=zeros(L,1);    %Initial weight vector`
	`xin=zeros(L,1);  %Initial input signal`

For each instant of time, k = 1, 2,…, compute:	
Output signal:	`y(i)=w'*xin;`
Estimation Error:	`error= d(i)-y(i);`
Tap-Weight Adaptation:	`wtemp = w + 2*mu*error*xin;`

Table III-2: *MATLAB code of LMS algorithm*

2. Normalized Least Mean Square (NLMS) algorithm

NLMS algorithm	
Initial Conditions:	$0 < \alpha < 1$ and c: a small constant Length of adaptive filter: L Input vector: $x_{L,1} = [0,0,...,0]^T$ Weight vector: $w_{L,1} = [0,0,...,0]^T$
For each instant of time, n = 1, 2,…, compute:	
Output signal:	$y(n) = w^T(n)x(n)$
Estimation Error:	$e(n) = d(n) - y(n)$
Tap-Weight Adaptation:	$w(n+1) = w(n) + \frac{2\alpha}{c + x^T(n)x(n)} x(n)e(n)$

Table III-3: *Summary NLMS algorithm*

MATLAB code	

Initial Conditions:	`alfa=0.42;       %Alfa`
	`c=0.01;          %A small constant`
	`L=length(h);     %the same length of RIR`
	`w=zeros(L,1);    %Initial weight vector`
	`xin=zeros(L,1);  %Initial input signal`

For each instant of time, k = 1, 2,…, compute:	
Output signal:	`y(i)=w'*xin;`
Estimation Error:	`error= d(i)-y(i);`

Tap-Weight Adaptation:	`mu=alfa/(c+xin'*xin);`
	`wtemp = w + 2*mu*error*xin;`

Table III-4: *MATLAB code of NLMS algorithm*

3.3.2. Double-talk Detector algorithm

In this simulation, we used the Normalized Cross-Correlation method to detect the existence of the double-talk. We calculate the estimates using the exponential recursive weighting algorithm to obtain the values of the cross-correlation (r_{em}) between error signal and microphone signal and the variance (σ_m^2) of the microphone signal. And achieve the decision statistic (ξ_{DTD}) from these values. Finally, we compare this value to the threshold (*T*) to make the decision of Double-talk Detector.

Because of the convergence time of the adaptive filter, we must setup the first time (DTDbegin) when the Double-talk Detector start working.

The summary and MATLAB code below will illustrate the theory and simulation the problem of Double-talk detection:

NCC algorithm summary	
Initial Conditions:	$0 < \lambda < 1$ and $\lambda \approx 1$ Threshold ≈ 1
For each instant of time, n= 1, 2,…, compute:	
Cross-correlation:	$r_{ed}(n) = \lambda r_{ed}(n-1) + (1-\lambda)e(n)d^T(n)$
Variance of near-end speech:	$\hat{\sigma}_d^2(n) = \lambda \hat{\sigma}_d^2(n-1) + (1-\lambda)d(n)d^T(n)$
Decision statistic:	$\xi_{NCC} = 1 - \frac{\hat{r}_{ed}}{\hat{\sigma}_d^2}$
Making the DTD decision:	If ξ_{DTD}<*T*, freeze adaptive filter If ξ_{DTD}>*T*, updating adaptive filter coefficients

Table III-5: *Summary of NCC double-talk detection algorithm*

In the MALAB code, one introduces the new variable wtemp (temporary value) for updating the adapter filter coefficients.

1. If $\xi_{DTD} < T$, then freeze the adaptive filter, i.e., stop updating the adaptive filter coefficients. Double-talk mode or both near-end and far-end are silent.

```
wtemp = w;        %Freeze the adaptive filter coefficients
```

2. If $\xi_{DTD} > T$, then continue running the loop of the adaptive filter algorithm, i.e., the updating the adaptive filter coefficients will be continuous. This case for no near-end speech is detected: far-end talks only.

```
w=wtemp;         %Update filter coefficients
```

MATLAB code of DTD using NCC algorithm	

Initial Conditions:	`T=0.92;          %Threshold`
	`Lambda_DTD=0.95; %Constant`
	`DTDbegin=50*L;   %The time to activate DTD`

For each instant of time, n= 1, 2,…, compute:	
Cross-correlation:	`r_em(i)=lambda_DTD*(r_em(i-1))+(1-lambda_DTD)*e(i)*d(i)';`
Variance of near-end speech:	`varMIC(i)=sqrt(lambda_DTD*(varMIC(i-1)^2)+(1-lambda_DTD)*d(i)*d(i)');`
Decision statistic:	`decision_statistic(i)=1-(r_em(i)/varMIC(i)^2);`

Making the DTD decision:	`if (decision_statistic(i)>threshold(i))`
	`    w=wtemp;    %Update filter coefficient`
	`end`

Table III-6: *MATLAB code of NCC double-talk detection algorithm*

3.3.3. Calculate other issues

1. Mean Square Error (MSE)

The purpose of the adaptive filter is minimizing the Mean Square Error MSE:

$$\xi = E\left\{|e(n)|^2\right\} = E\left\{\left|d(n) - \hat{d}(n)\right|^2\right\}$$

Therefore, the values and graph of this quantity will be essential to evaluate the performance of the adaptive filter. If the adaptive algorithm works well, after convergence time, the value of MSE should be reduced gradually to zero (for the case of no near-end signal). The segment of MATLAB code below to calculate this parameter.

```
mse_iteration(i)=error^2;  %Square Error
for i=1:N-L
    mse(i)=mean(mse_iteration(i:i+L)); %MSE - Mean Square Error
end
```

2. Echo Return Loss Enhancement (ERLE)

Echo Return Loss Enhancement ERLE is one of the most important parameters is commonly used to evaluate the performance of the echo cancellation algorithm. This quantity measures how much echo attenuation the echo canceller removed from the microphone signal.

ERLE, measures in dB, is defined as the ratio of the microphone signal's power (*d[n]*) and the residual error signal's power (*e[n]*).

$$ERLE = 10\log\frac{P_d(n)}{P_e(n)} = \frac{E[d^2(n)]}{E[e^2(n)]}$$

ERLE depends on the algorithm we use for the adaptive filter, two quantities are considered with ERLE are the convergence time and near-end attenuation will be different relative to different algorithms. In our simulation, we made the MATLAB code to fulfill the computation of ERLE as follow,

```
powerD(i) = abs(d(i))^2;     %Power of Microphone signal
powerE(i)=abs(e(i))^2;       %power of Error signal
for i=1:N-L
    ERLE(i)=10*log10(mean(powerD(i:i+L))/mean(powerE(i:i+L));
end
```

3.4. RESULTS

The resulting plots below demonstrate the performance of acoustics echo canceller with different algorithms. Fist, we show the graph of near-end speech which we need to compare with the error signal (it is should be approximately equal). Second, the plots in turn are: microphone signal $d(n)$, output signal $y(n)$ of the adaptive filter and error signal $e(n)$. The error signal in real teleconference system is transmitted from near-end user to the far-end user. If no near-end speech, it should be a silent signal and if near-end talks, then the error signal contains only near-end speech.

Other graphs will express the evaluations of the estimated impulse response, Double-talk detector and performance of the adaptive filter (MSE, ERLE).

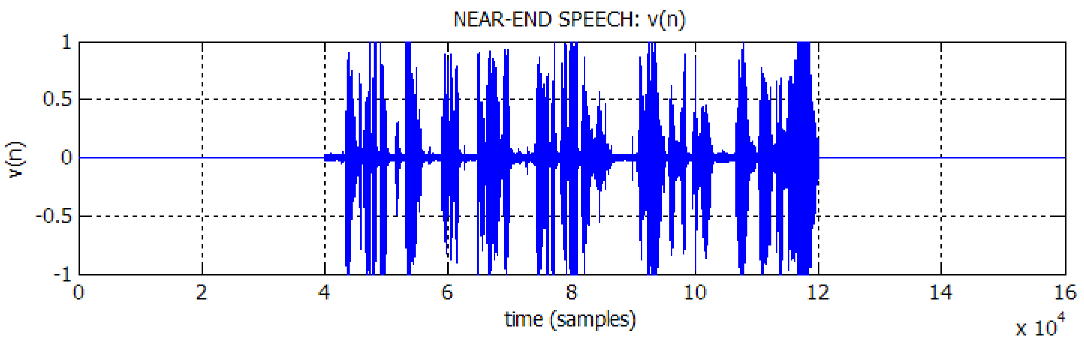


Figure III-5: *Plot near-end speech, it is necessary to compare with the error signal that the echo canceller produced.*

3.4.1. LMS algorithm

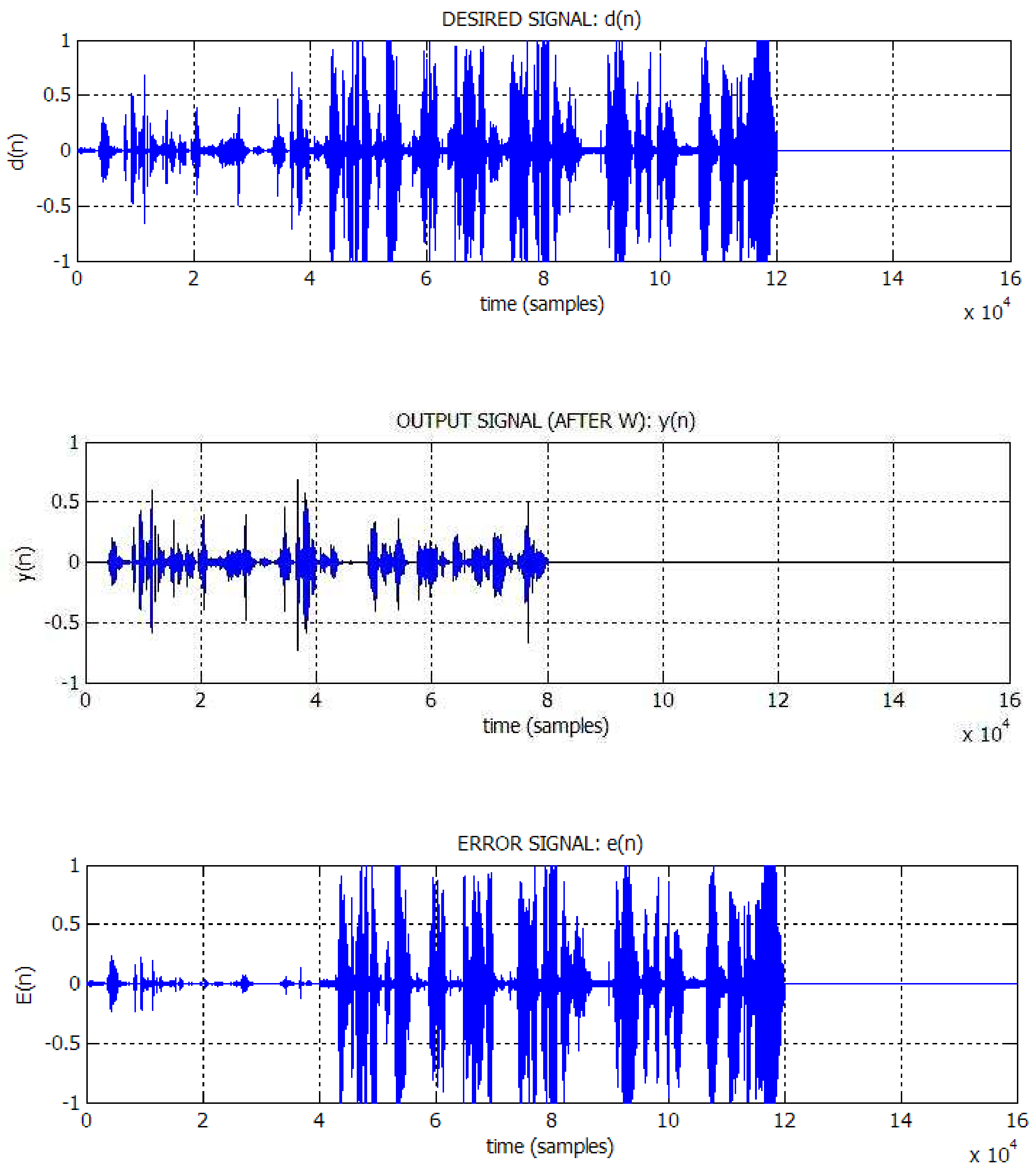


Figure III-6: *Plot the needed signals (LMS algorithm) in turn are: desired signal, output signal and error signal*

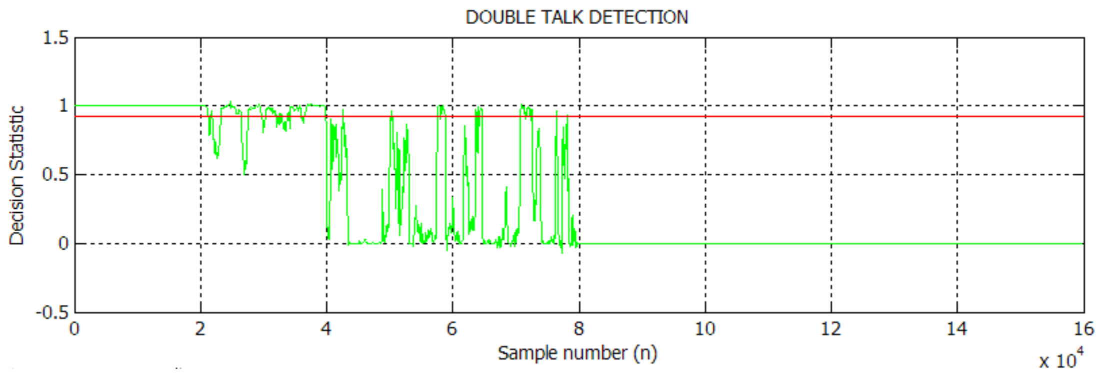


***Figure III-7:** Double-talk detection of LMS algorithm where the decision statistic $\xi_{NCC}(n)$ (green line) compare to the threshold T (red line). If $\xi_{NCC}(n) > T$, far-end talks only and if $\xi_{NCC}(n) < T$, double-talk or only near-end talks or both side are silent*

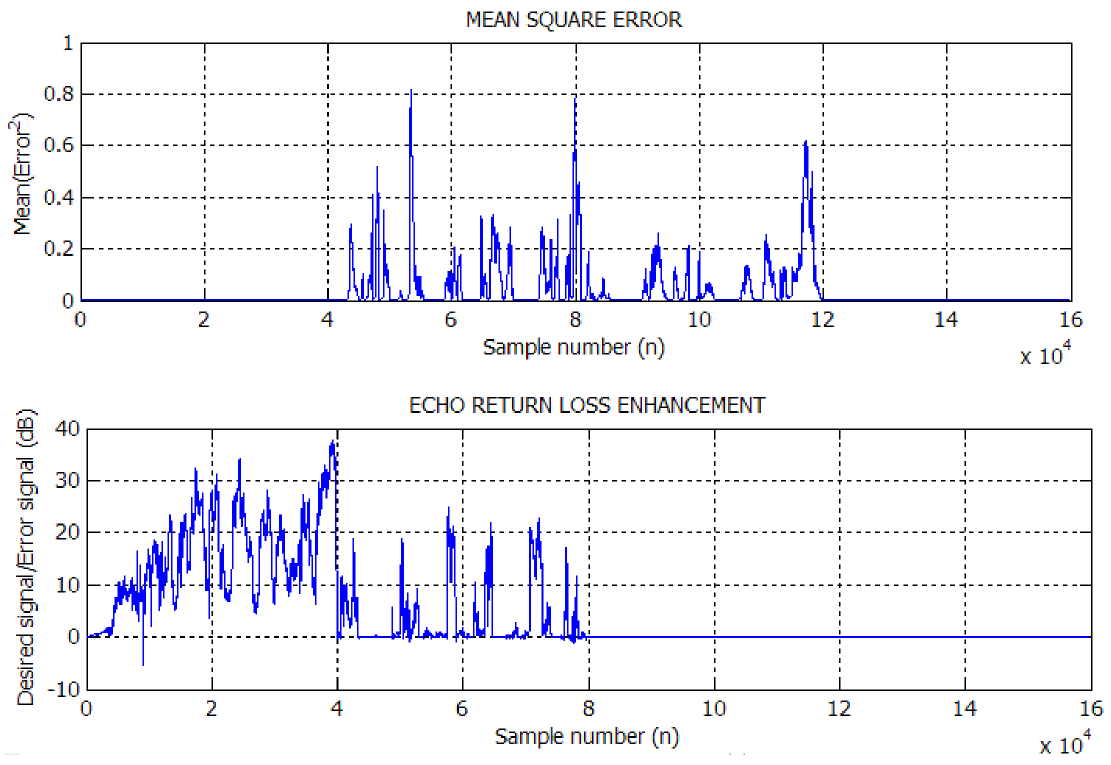


***Figure III-8:** Evaluation of LMS algorithm, Mean Square Error (measure how much the algorithm minimize the echo) and Echo Return Loss Enhancement (measure the echo attenuation the echo canceller removed)*

3.4.1. NLMS algorithm

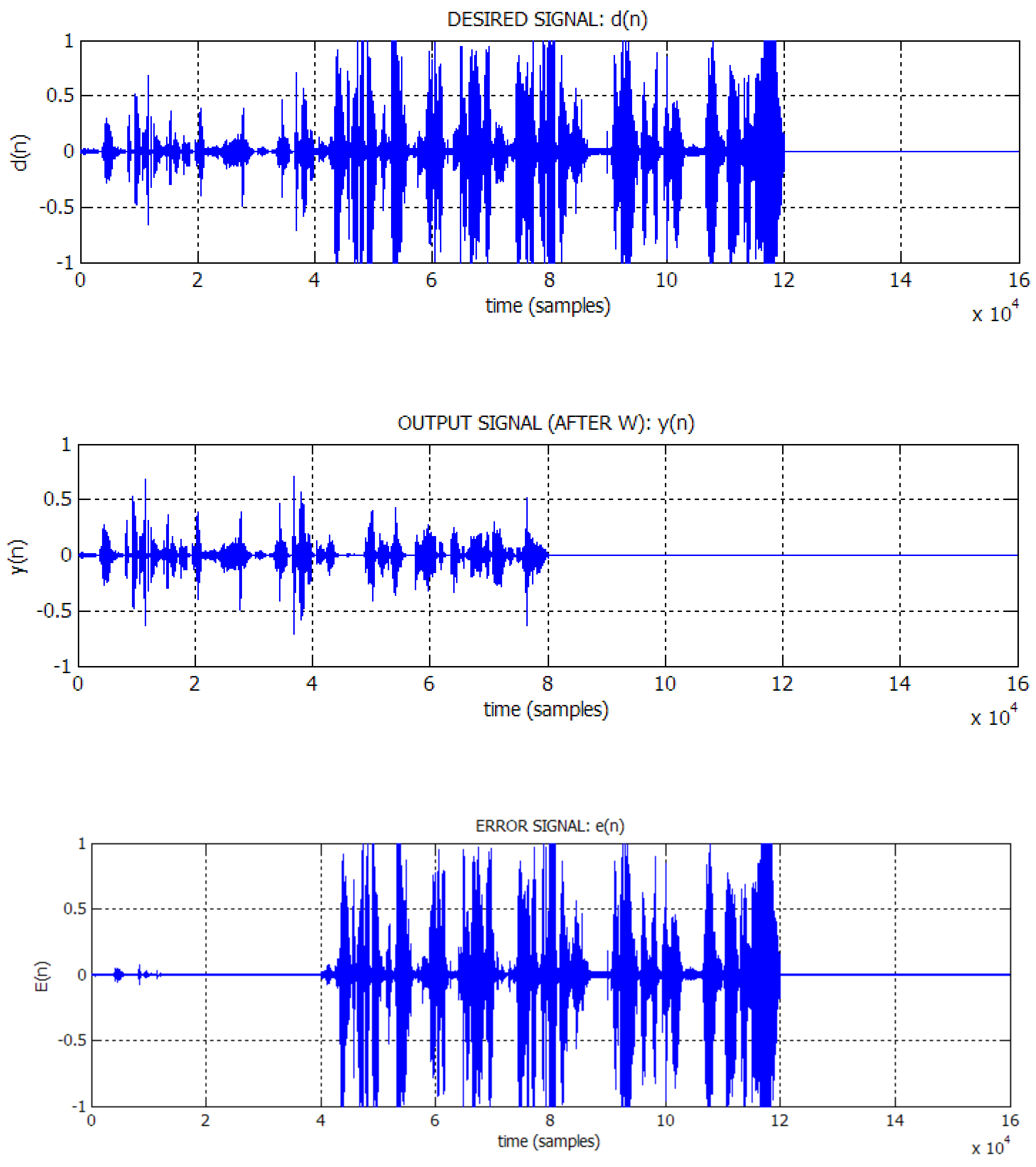


Figure III-9: *Plot the needed signals (NLMS algorithm) in turn are: desired signal, output signal and error signal*

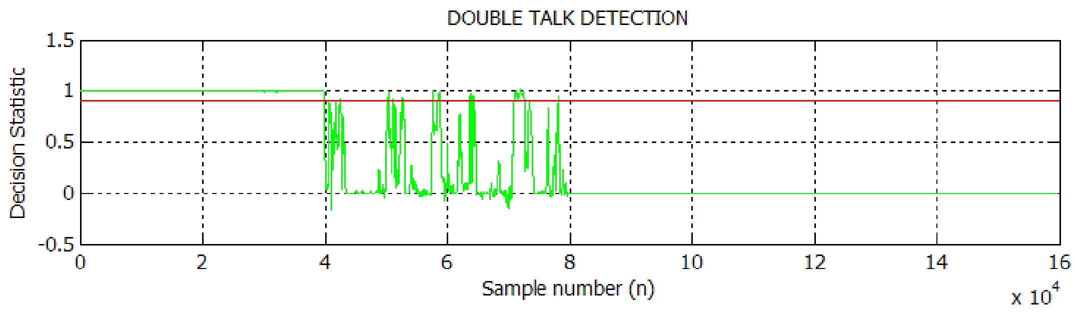
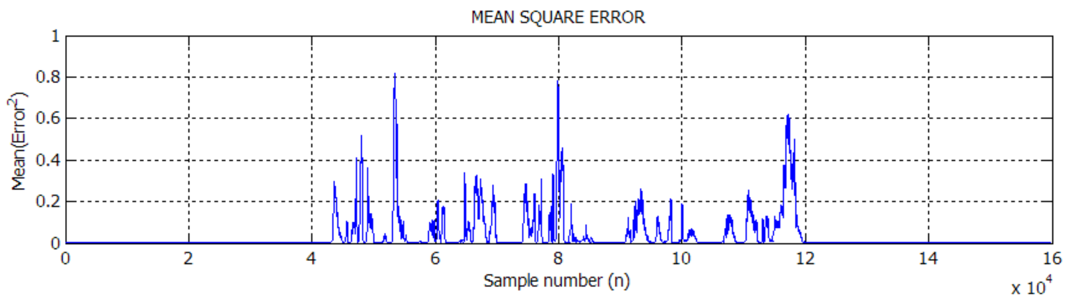


Figure III-10: *Double-talk detection of NLMS algorithm where the decision statistic $\xi_{NCC}(n)$ (green line) compare to the threshold T (red line). If $\xi_{NCC}(n) > T$, far-end talks only and if $\xi_{NCC}(n) < T$, double-talk or only near-end talks or both side are silent*



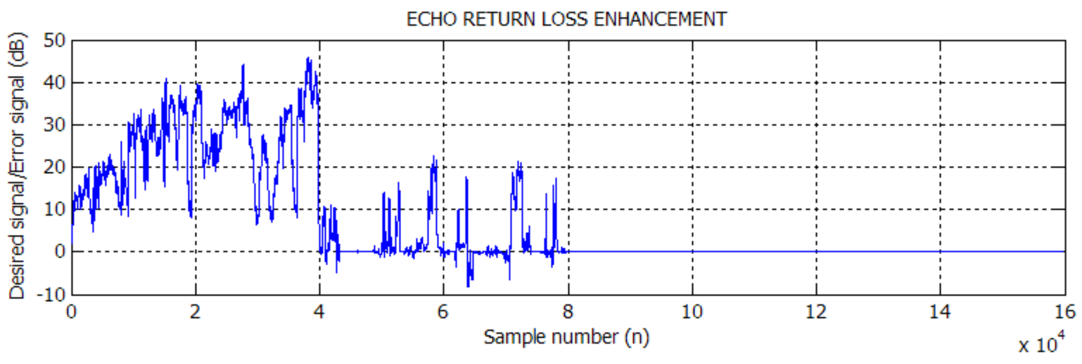


Figure III-11: *Evaluation of NLMS algorithm, Mean Square Error (measure how much the algorithm minimize the echo) and Echo Return Loss Enhancement (measure the echo attenuation the echo canceller removed)*

3.5. EVALUATION

From the experiment works and the resultant graphs, we can evaluate the echo cancellation algorithm in order to have a deeper understanding and the conclusion of the thesis' topic. As the results above of two echo cancellation algorithms are LMS and NLMS, we have some evaluations as follow.

1. Comparison between LMS and NLMS

Both of them could converge approximately the estimated impulse response $\hat{\mathbf{h}}$ of the true room impulse response h, thus the estimated echo $\hat{y}(n)$ signal look like similar the true echo signal $y(n)$. As a result, the error signals almost obtain our desired results. Considering the Figure (III-6) and (III-9), it is easily to see that the resultant error signals of NLMS is more convincing than the ones of LMS. The figure (III-7) and (III-10), the double-talk decision of NLMS also is better than LMS (more exactly).

From these comments above, we can conclude that the performance of NLMS algorithm is better and exacter than LMS algorithm.

2. Convergence test

Convergence is the most important factor to observe when running the echo cancellation algorithm. If the filter coefficients used in the adaptive filter algorithm did not converge, the code could get problem. In this simulation, we used the standard signals as white noise (as the input signals), the low pass filter (model the impulse response) to check the operation of the algorithm. If the problem still exist, then we verify the convergence factor μ_n. By varying this factor, we can control and adjust the convergence of the adaptive filter algorithm.

3. Echo Return Loss Enhancement (ERLE)

This parameter is used in order to evaluate the quality of the echo cancellation algorithm. If the echo cancellation algorithm perform well, then the values of ERLE should be in the range of (45dB, -40dB) (for Signal to Noise ratio SNR of 45dB).

Both LMS and NLMS gave the plots of ERLE within this range, thus the ERLE for these algorithms achieved the required value or in other words, the algorithms work well.

4. Check the resultant audio signals

With MATLAB, we can check the resultant signals by playing them. The microphone signal and the error signal after echo canceller are play by loud speaker gave us the good result we want. We realized that there was litter echo still existed in the beginning of the error signal, this because of the convergence time of the adaptive filtering algorithm. In general, the sounds we heard bring us with the convincing results.

CHAPTER IV : CONCLUSION AND FURTHER WORK

4.1. CONCLUSION

In this thesis we studied how to cancel acoustic echo by AEC. One of the major problems in a telecommunication application over a telephone system is echo. The Echo cancellation algorithm presented in this thesis successfully attempted to find a software solution for the problem of echoes in the telecommunications environment. AEC is the conventional method for solving the acoustic echo problem. Under ideal conditions AEC can achieve perfect echo cancellation, because it estimates both the phase and amplitude of the echo signal. The proposed algorithm was completely a software approach without utilizing any Digital Signal Processing (DSP) hardware components.

Speech has most of its energy concentrated to lower frequencies. Therefore it is most important to achieve an optimal echo cancellation at these frequencies. At higher frequencies the ear is not sensitive to phase information.

The algorithm was capable of running in any PC with MATLAB software installed. In addition, the results obtained were convincing. The audio of the output speech signals were highly satisfactory and validated the goals of this research.

4.2. FURTHER WORKS

The algorithm proposed in this thesis presents a solution for single channel acoustic echoes. However, most often in real life situations, multi-channel sound is the norm for telecommunication. For example, when there is a group of people in a teleconference environment and everybody is busy talking, laughing or just communicating with each other multi-channel sound abounds. Since there is just a single microphone the other end will hear just a highly incoherent monographic sound. In order to handle such situations in a better way the echo cancellation algorithm developed during this research should be extended for the multi-channel case.

Another thing we need to implement in the acoustic echo canceller is that make it working in real time (in practice) performance of teleconference system. Our work only simulate the thesis' topic in offline mode, therefore, it is necessary to implement in real communication between far-end and near-end room of the teleconference network.

Reference

[1] Jacob Benesty, Tomas Gansler, Denis R. Morgan, M. Mohan Sondhi and Steven L. Gay, *Advances in Network and acoustic echo cancellation*, ISBN: 3-540-41721-4, Springer, 2001

[2] Farhang-Boroujeny, *Adaptive Filters, Theory and Applications*, John Wiley and Sons, New York, 1999

[3] Haykin, Simon, *Adaptive Filter Theory, 2nd Edition*. Prentice-Hall Inc., New Jersey, 1991

[4] Sadaoki Furui and M. Mohan Sondhi, "Advances in Speech Signal Processing", Marcel Dekker, Inc, 1992

[5] Monson H. Hayes, *Statistical Digital Signal Processing And Modeling,* John Wiley & Sons, Inc., 1996

[6] Haykin, Simon, *Modern Filters*, Macmillan Publishing Company New York, ISBN 0-02-35275 0-1, 1989

[7] S. Haykin, Englewood, *Adaptive Filter Theory (4th edition)*, N.J.: Prentice Hall, Inc., 2002

[8] Mohammad Asif Iqbal, Jack W. Stokes and Steven L. Grant, *Normalized Double-talk Detection based on microphone and AEC error, cross-correlation*

[9] Steven L. Gay and Jacob Benesty, *Acoustic Signal Processing for Telecommunication,* Kluwer Academic Publishers, 2001, pp. 81-97

[10] Suehiro Shimauchi, Yoichi Haneda and Akitoshi Kataoka, *Frequency domain adaptive algorithm with nonlinear function of error-to-reference ratio for double-talk robust echo cancellation,* NTT Cyber Space Laboratories, NTT Corporation, July, 2004

[11] Sven Nordebo, *Signal Processing Antennas I,* September 18, 2004
http://w3.msi.vxu.se/~sno/ED4024/ED4024.pdf

[12] Sven Nordebo, *Signal Processing Antennas II,* May 11, 2004.
http://w3.msi.vxu.se/~sno/ED4034/ED4034.pdf

[13] Ivo Mateljan, Kosta Ugrinovic, *The comparison of room impulse response measuring systems*, University of Split, 21000 Split, Croatia

[14] Samuel D. Stearns, *Digital Signal Processing with Example in MATLAB*, ISBN: 0-8493-1091-1, CRC Press LLC, 2003

[15] Irina Dornean, Marina Topa, Botond Sandor Kirei and Marius Neag, *Sub-Band Adaptive Filtering for Acoustic Echo Cancellation*, IEEE Xplore, 2009

Appendix A: MATLAB code of LMS algorithm

```
clear all
%-----------------------------------------------------------------------
%Load Data
[x, Fs, nbits] = wavread('c:/audiofiles/fe1');    %Far-end signal
[v, Fs, nbits] = wavread('c:/audiofiles/ne1');    %Near-end signal
[h, Fs, nbits] = wavread('c:/audiofiles/room_impulse_response_128taps');
%Room impulse response

%Declare the needed variables
L=length(h);        %Length of adaptive filter (same length of RIR)
N=length(x);        %Number of iterations
T=0.92;             %Threshold for Double talk detection
lambda_DTD=0.95;    %Constant for calculating decision statistic of DTD
DTDbegin=21000;     %The time to activate DTD

%Intial value 0
w=zeros(L,1);       %Initial weight vector of AF Lx1
xin=zeros(L,1);     %Initial input signal of AF Lx1
varMIC=zeros(N,1);  %Initial variance of microphone signal of AF Nx1
r_em=zeros(N,1);    %Initial Cross correlation between error and microphone
signals

%Ambient noise
WhiteNoise = wgn(N,1,-65);     %With make SNR of 45dB
%Microphone signal
EchoSignal=filter(h,1,x);      %Echo signal after filter H
d=EchoSignal+WhiteNoise+v;     %Desired signal (Microphone Signal)

%Make column vectors
x=x(:);             %Far end signal Nx1
d=d(:);             %Desired signal Nx1

%The values for calculate Step-Size of Adaptive Filter
mu=0.014;

%Calculate the average SNR (desired signal/noise)
powerMic = sum(abs(d).^2)/N;          %Power of Microphone signal
powerN = sum(abs(WhiteNoise).^2)/N;   %Power of White Noise
SNR=10*log10(powerMic/powerN);        %Calculate the SNR

%------------------------------------------------------------------------
%-------------LMS algorithm for Adaptive Filter-----------------------
for i=1:N
for j=L:-1:2
    xin(j)=xin(j-1);
end
    xin(1)=x(i);               %Insert new sample at beginning of input

    y(i)=w'*xin;               %Output signal after adaptive filter
    error=d(i)-y(i);           %ERROR
    e(i)=error;                %Store estimation error
    wtemp = w + 2*mu*error*xin;%Update filter

% -----------NORMALIZED CROSS-CORELATION ALGORITHM DTD--------------
threshold(i)=T;     %Threshold for plotting DTD
if (i<=DTDbegin)    %The beginning time of the DTD
    w=wtemp;        %Update filter coefficients
```

```
end
if (i>DTDbegin)
    %Cross correlation between error and microphone signal
    r_em(i)=lambda_DTD*(r_em(i-1))+(1-lambda_DTD)*e(i)*d(i)';
    %Variance of microphone signal
    varMIC(i)=sqrt(lambda_DTD*(varMIC(i-1)^2)+(1-lambda_DTD)*d(i)*d(i)');
    decision_statistic(i)=1-(r_em(i)/varMIC(i)^2);  %Decision statistic
%Making the double-talk decision
if (decision_statistic(i)>threshold(i))
    w=wtemp;         %Update filter coefficients
end

end
%-------------ERLE-------------------------------------
powerD(i) = abs(d(i))^2;    %Power of Microphone signal
powerE(i)=abs(e(i))^2;      %power of Error signal
%--------------MSE-------------------------------------
mse_iteration(i)=error^2;  %Square Error
end

for i=1:N-L
    %MSE - Mean Square Error
    mse(i)=mean(mse_iteration(i:i+L));
    %Echo Return Loss Enhancement
    ERLE(i)=10*log10(mean(powerD(i:i+L))/mean(powerE(i:i+L)));

    %Plotting Double-talk Detection
    if (i>DTDbegin)
       ds(i)=mean(decision_statistic(i:i+L));
    else
       ds(i)=1;
    end
end

%----------------------------------------------------------------------
%PlOTTING THE NECESSARY SIGNALS
%----------------------------------------------------------------------
figure(1)
%-------echo signal-------------------------
subplot(4,1,1)
plot(EchoSignal)
xlabel('time (samples)');
ylabel('echo(n)');
title('ECHO SIGNAL: echo(n)')
grid on
axis([0 N -1 1]);

%-------Desired signal-----------------------
subplot(4,1,2)
plot(d)
xlabel('time (samples)');
ylabel('d(n)');
title('DESIRED SIGNAL: d(n)')
grid on
axis([0 N -1 1]);

%-------Output signal x(n)-------------------
subplot(4,1,3)
plot(y)
xlabel('time (samples)');
```

```
ylabel('y(n)');
title('OUTPUT SIGNAL (AFTER W): y(n)')
grid on
axis([0 N -1 1]);

%-------Error signal x(n)--------------------
subplot(4,1,4)
plot(e,'red')
xlabel('time (samples)');
ylabel('E(n)');
title('ERROR SIGNAL: e(n)')
axis([0 N -1 1]);
grid on

%-------Estimation system w-----------------
figure(2)
subplot(2,1,1)
plot(w,'red')
xlabel('Tap');
ylabel('Magnitude (W)');
title('ESTIMATE SYSTEM: W(N)')
grid on

%-------True system h----------------------
subplot(2,1,2)
plot(h)
xlabel('Sample number (n)');
ylabel('Magnitude (H)');
title('TRUE IMPULSE RESPONSE: h(n)')
grid on

%-------Estimator for DTD------------------
figure(3)

%-------Decision Statistic-----------------
subplot(311)
plot(ds,'green')
hold all
plot(threshold,'red')
hold off
xlabel('Sample number (n)');
ylabel('Decision Statistic');
title('DOUBLE TALK DETECTION')
grid on

%-------Mean square error-------------------
subplot(312)
plot(mse)
xlabel('Sample number (n)');
ylabel('Mean(Error^2)');
title('MEAN SQUARE ERROR')
grid on

%-------Echo return loss enhancement---------
subplot(313)
plot(ERLE)
xlabel('Sample number (n)');
ylabel('Desired signal/Error signal (dB)');
title('ECHO RETURN LOSS ENHANCEMENT')
grid on
```

Appendix B: MATLAB code of NLMS algorithm

```
clear all
%-----------------------------------------------------------------
%Load Data
[x, Fs, nbits] = wavread('c:/audiofiles/fe1');    %Far-end signal
[v, Fs, nbits] = wavread('c:/audiofiles/ne1');    %Near-end signal
[h, Fs, nbits] = wavread('c:/audiofiles/room_impulse_response_128taps');
%Room impulse response

%Declare the needed variables
L=length(h);         %Length of adaptive filter (same length of RIR)
N=length(x);         %Number of iterations
T=0.92;              %Threshold for Double talk detection
lambda_DTD=0.95;     %Constant for calculating decision statistic of DTD
DTDbegin=21000;      %The time to activate DTD

%Intial value 0
w=zeros(L,1);        %Initial weight vector of AF Lx1
xin=zeros(L,1);      %Initial input signal of AF Lx1
varMIC=zeros(N,1);   %Initial variance of microphone signal of AF Nx1
r_em=zeros(N,1);     %Initial Cross correlation between error and microphone
signals

%Ambient noise
WhiteNoise = wgn(N,1,-65);     %With make SNR of 45dB
%Microphone signal
EchoSignal=filter(h,1,x);      %Echo signal after filter H
d=EchoSignal+WhiteNoise+v;     %Desired signal (Microphone Signal)

%Make column vectors
x=x(:);              %Far end signal Nx1
d=d(:);              %Desired signal Nx1

%The values for calculate Step-Size of Adaptive Filter
alfa=0.42;      %Alfa
c=0.01;         %A small constant

%Calculate the average SNR (desired signal/noise)
powerMic = sum(abs(d).^2)/N;            %Power of Microphone signal
powerN = sum(abs(WhiteNoise).^2)/N;     %Power of White Noise
SNR=10*log10(powerMic/powerN);          %Calculate the SNR

%------------------------------------------------------------------
%-------------NLMS algorithm for Adaptive Filter-----------------------
for i=1:N
for j=L:-1:2
    xin(j)=xin(j-1);
end
    xin(1)=x(i);               %Insert new sample at beginning of input

    y(i)=w'*xin;               %Output signal after adaptive filter
    error=d(i)-y(i);           %ERROR
    e(i)=error;                %Store estimation error
    mu=alfa/(c+xin'*xin);      %Calculate Step-size
    wtemp = w + 2*mu*error*xin;%Update filter

% -----------NORMALIZED CROSS-CORELATION ALGORITHM DTD--------------
threshold(i)=T;      %Threshold for ploting DTD
```

```
if (i<=DTDbegin)    %The beginning time of the DTD
    w=wtemp;        %Update filter coefficients
end
if (i>DTDbegin)
    %Cross correlation between error and microphone signal
    r_em(i)=lambda_DTD*(r_em(i-1))+(1-lambda_DTD)*e(i)*d(i)';
    %Variance of microphone signal
    varMIC(i)=sqrt(lambda_DTD*(varMIC(i-1)^2)+(1-lambda_DTD)*d(i)*d(i)');
    decision_statistic(i)=1-(r_em(i)/varMIC(i)^2);  %Decision statistic
%Making the double-talk decision
if (decision_statistic(i)>threshold(i))
    w=wtemp;        %Update filter coefficients
end

end
%-------------ERLE--------------------
powerD(i) = abs(d(i))^2;    %Power of Microphone signal
powerE(i)=abs(e(i))^2;      %power of Error signal
%--------------MSE---------------------
mse_iteration(i)=error^2;  %Square Error
end
for i=1:N-L
    %MSE - Mean Square Error
    mse(i)=mean(mse_iteration(i:i+L));
    %Echo Return Loss Enhancement
    ERLE(i)=10*log10(mean(powerD(i:i+L))/mean(powerE(i:i+L)));
    %Plotting Double-talk Detection
    if (i>DTDbegin)
       ds(i)=mean(decision_statistic(i:i+L));
    else
       ds(i)=1;
    end
end

%-----------------------------------------------------------------------
%PlOTTING THE NECESSARY SIGNALS
%-----------------------------------------------------------------------
figure(1)
%-------echo signal------------------------
subplot(4,1,1)
plot(EchoSignal)
xlabel('time (samples)');
ylabel('echo(n)');
title('ECHO SIGNAL: echo(n)')
grid on
axis([0 N -1 1]);

%-------Desired signal--------------------
subplot(4,1,2)
plot(d)
xlabel('time (samples)');
ylabel('d(n)');
title('DESIRED SIGNAL: d(n)')
grid on
axis([0 N -1 1]);

%-------Output signal x(n)----------------
subplot(4,1,3)
plot(y)
xlabel('time (samples)');
ylabel('y(n)');
```

```
title('OUTPUT SIGNAL (AFTER W): y(n)')
grid on
axis([0 N -1 1]);

%-------Error signal x(n)-----------------
subplot(4,1,4)
plot(e,'red')
xlabel('time (samples)');
ylabel('E(n)');
title('ERROR SIGNAL: e(n)')
axis([0 N -1 1]);
grid on

%-------Estimation system w---------------
figure(2)
subplot(2,1,1)
plot(w,'red')
xlabel('Tap');
ylabel('Magnitude (W)');
title('ESTIMATE SYSTEM: W(N)')
grid on

%-------True system h---------------------
subplot(2,1,2)
plot(h)
xlabel('Sample number (n)');
ylabel('Magnitude (H)');
title('TRUE IMPULSE RESPONSE: h(n)')
grid on

%-------Estimator for DTD-----------------
figure(3)

%-------Decision Statistic----------------
subplot(311)
plot(ds,'green')
hold all
plot(threshold,'red')
hold off
xlabel('Sample number (n)');
ylabel('Decision Statistic');
title('DOUBLE TALK DETECTION')
grid on

%-------Mean square error-----------------
subplot(312)
plot(mse)
xlabel('Sample number (n)');
ylabel('Mean(Error^2)');
title('MEAN SQUARE ERROR')
grid on

%-------Echo return loss enhancement------
subplot(313)
plot(ERLE)
xlabel('Sample number (n)');
ylabel('Desired signal/Error signal (dB)');
title('ECHO RETURN LOSS ENHANCEMENT')
grid on
```

Växjö University

Matematiska och systemtekniska institutionen

SE-351 95 Växjö

Tel. +46 (0)470 70 80 00, fax +46 (0)470 840 04

http://www.vxu.se/msi/