Acknowledgment

On successful completion of our project, we would like to place on record our sincere thanks and gratitude to concerned people, whose suggestions and words of encouragement has been valuable.

We express our heartfelt gratitude to ***BNM Institute of Technology***, for giving us the opportunity to pursue Degree of Telecommunication Engineering and helping us to shape our career.

We take this opportunity to thank our beloved Principal ***Dr. K. Udaya Kumar***, Dean ***Dr. K. Ranga*** and Director, ***Prof T. J. Ramamurthy*** for their support and encouragement in doing the project.

We would also like to thank ***Prof. Rajashree Narendra***, Head of the Department of Telecommunication**,** who has been the source of inspiration throughout our project work. We are highly thankful to our internal guide ***Smt. Ashwini S. Savanth***, Asst. Professor for providing us with useful information at every stage of our project and for the knowledge we gained under her able guidance.

We would also like to thank our project coordinator ***Sri. P. Venkat Rao***, Asst. Professor, ***Prof. R Narendra,*** Associate Professor, ***Sri. Ravishankar Holla*** Asst. Professor, Dept of Telecommunication and ***Sri.D. Madhusudan Rao,*** Director of UG Consultants who were kind enough to extend their help for our project whenever the need aroused.

Finally we are thankful to all the teaching and non teaching staff of Department of Telecommunication, for their help in successful completion of our project. Last but not the least we would like to extend our sincere gratitude to our parents and all our friends who gave us a constant source of inspiration.

Abstract

A hands-free telephone or full-duplex intercom system has a feedback or echo problem because the output from the loudspeaker feeds into the microphone.

Echo is a delayed and distorted version of an original sound or electrical signal which is reflected back to the source. If a reflected wave arrives after a very short time of direct sound, it is considered as a spectral distortion or reverberation. However, when the reflected wave arrives a few tens of milliseconds after the direct sound, it is heard as a distinct echo. In data communication, the echo can incur a big data transmit error. In applications like hands-free telecommunications, the echo, in extreme conditions, can make the conversation impossible. The echo has been a big issue in communication networks. Hence there is a need for the development of an effective way to control the acoustic echo in hands-free communications.

Table of Contents

List of Figuers

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Basics of Echo

Echo is a phenomenon where a delayed and distorted version of an original sound or electrical signal is reflected back to the source. With rare exceptions, conversations take place in the presence of echoes. Echoes of our speech are heard as they are reflected from the floor, walls and other neighboring objects. If a reflected wave arrives after a very short time of direct sound, it is considered as a spectral distortion or reverberation. However, when the leading edge of the reflected wave arrives a few tens of milliseconds after the direct sound, it is heard as a distinct echo. Since the advent of telephony echoes have been a problem in communication networks. In particular, echoes can be generated electrically due to impedance mismatches at various points along the transmission medium. The most important factor in echoes is called end-to-end delay, which is also known as latency. Latency is the time between the generation of the sound at one end of the call and its reception at the other end. Round trip delay, which is the time taken to reflect an echo, is approximately twice the end-to-end delay.

Echoes become annoying when the round trip delay exceeds 30 ms. Such an echo is typically heard as a hollow sound. Echoes must be loud enough to be heard. Those less than 30 decibels (dB) are unlikely to be noticed. However, when round trip delay exceeds 30 ms and echo strength exceeds 30 dB, echoes become steadily more disruptive. However, not all echoes reduce voice quality. In order for telephone conversations to sound natural, callers must be able to hear themselves speaking. For this reason, a short instantaneous echo, termed side tone, is deliberately inserted. The side tone is coupled with the caller's speech from the telephone mouthpiece to the earpiece so that the line sounds connected.

1.2 Literature Survey

Several algorithms are introduced to solve the problem of acoustic echo cancellation such as LMS, NLMS and VSS NLMS. The use of the VSS-NLMS algorithm will eliminate much of the trade-off between residual error and speed of convergence existing with the fixed step-size NLMS algorithm and therefore resulting in an improved performance. Because of its simplicity, the Least Mean Square (LMS) algorithm is the most popular adaptive algorithm. However, the LMS algorithm suffers from slow and data-dependent convergence behavior. The NLMS algorithm an equally simple, but more robust variant of the LMS algorithm, exhibits a better balance between simplicity and performance than the LMS algorithm, and has been given more attention in real time applications. A very serious problem, however, encountered in both the LMS and the NLMS algorithms, is the choice of the step size parameter that is a trade-off between the steady-state excess error and the speed of convergence.

1.3 Types of Echo

In telecommunications networks there are two types of echo. One source for an echo is electrical and the other echo source is acoustic

1.3.1 Hybrid/Electrical Echo

Hybrid echoes have been inherent within the telecommunications networks since the advent of the telephone. This echo is the result of impedance mismatches in the analog local loop. For example, this happens when mixed gauges of wires are used, or where there are unused taps and loading coils. In the Public Switched Telephone Network, (PSTN), by far the main source of electrical echo is the hybrid. This hybrid is a transformer located at a juncture that connects the two-wire local loop coming from a subscriber’s premise to the four-wire trunk at the local telephone exchange. The four-wire trunks connect the local exchange to the long distance exchange. This situation is illustrated in Figure 1.1

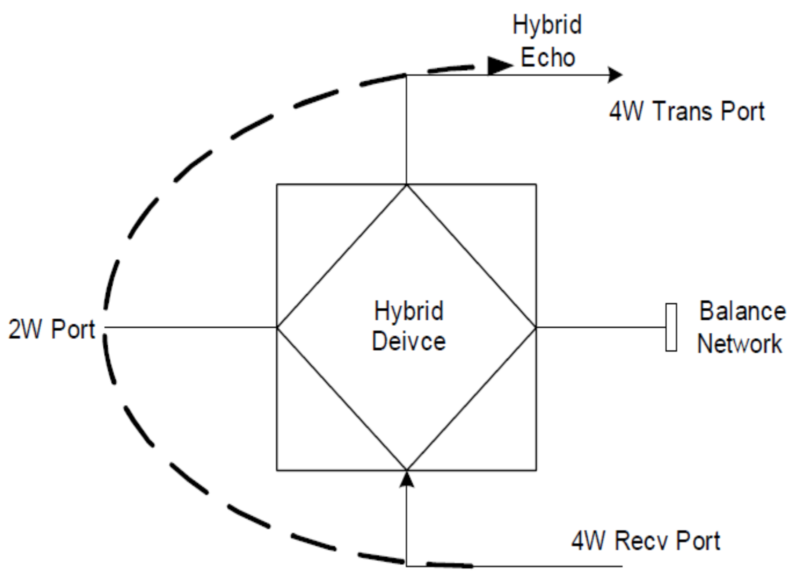


Figure 1.1 Hybrid Echo

The hybrid splits the two-wire local loop into two separate pairs of wires. One pair is used for the transmission path and the other for the receiver path. The hybrid passes on most of the signal. However, the impedance mismatch between the two-wire loop and the four-wire facility causes a small part of the received signal to “leak” back onto the transmission path.

The speaker hears an echo because the far-end receives the signal and sends part of it back again. Electrical echo is definitely not a problem on local calls since the relatively short distances do not produce significant delays. However, the electrical echo must be controlled on long distance calls. In the early years, when the public network was entirely circuit switched, the hybrid echo was the only significant source of echo. Since the locations of hybrids and most other causes of impedance differences in circuit switched networks were known, adequate echo control could be planned and provisioned. However, in today’s digital networks the points where two wires split into four wires is typically also the point where analog to digital conversion takes place. Regardless of whether the hybrid and analog to digital conversion is implemented in the same device or in two devices, the tow four wire conversions constitute an impedance mismatch and echoes are produced.

1.3.2 Acoustic Echo

The acoustic echo, which is also known as a “multipath echo”, is produced by poor voice coupling between the earpiece and microphone in handsets and hands-free devices. Further voice degradation is caused as voice-compressing and encoding/decoding devices process the voice paths within the handsets and in wireless networks. This results in returned echo signals with highly variable properties. When compounded with inherent digital transmission delays, call quality is greatly diminished for the wire line caller. Acoustic coupling is due to the reflection of the loudspeaker’s sound waves from walls, door, ceiling, windows and other objects back to the microphone. The result of the reflections is the creation of a multipath echo and multiple harmonics of echoes, which are transmitted back to the far-end and are heard by the talker as an echo unless eliminated. Adaptive cancellation of such acoustic echoes has become very important in hands-free communication systems such as teleconference or videoconference systems. The multipath echo phenomenon is illustrated in Figure 1.2.

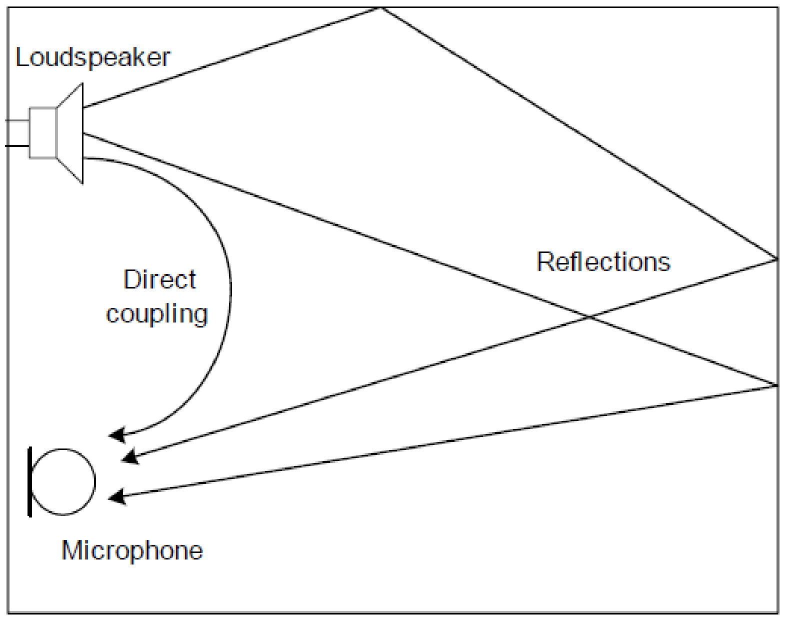


Figure 1.2 Sources of Acoustic Echo in a Room

1.4 Acoustic echo cancellation

Wireless phones are regarded as essential communications tools in the present global communications, and Wireless phones have a direct impact on people's day-to-day personal and business communications. As new network infrastructures are implemented and competition between wireless carriers increases, digital wireless subscribers are becoming ever more critical of the service and voice quality they receive from network providers. Subscriber demand for enhanced voice quality over wireless networks has driven a new and key technology termed echo cancellation, which can provide near wire line voice quality across a wireless network.

Subscribers use speech quality as a standard for assessing the overall quality of a network. Regardless of whether or not the subscribers' opinion is subjective, it is the key to maintaining subscriber loyalty. For this reason, the effective removal of hybrid and acoustic echoes, which are inherent within the telecommunications network infrastructure, is the key to maintaining and improving the perceived voice quality of a call. Ultimately, the search for improved voice quality has led to intensive research into the area of echo cancellation. Such research is conducted with the aim of providing solutions that can reduce background noise and remove hybrid and acoustic echoes before any transcoder processing occurs. By employing echo cancellation technology, the quality of speech can be improved significantly.

1.5 The Process of Echo cancellation

An echo canceller is basically a device that detects and removes the echo of the signal from the far end after it has echoed on the local end's equipment. In the case of circuit switched long distance networks, echo cancellers reside in the metropolitan Central Offices that connect to the long distance network. These echo cancellers remove electrical echoes made noticeable by delay in the long distance network. An echo canceller consists of three main functional components:

- Adaptive filter
- Doubletalk detector

- Non-linear processor

A brief overview of these components is presented in this chapter. However, a detailed sketch that involves mathematical illustrations is provided in chapter

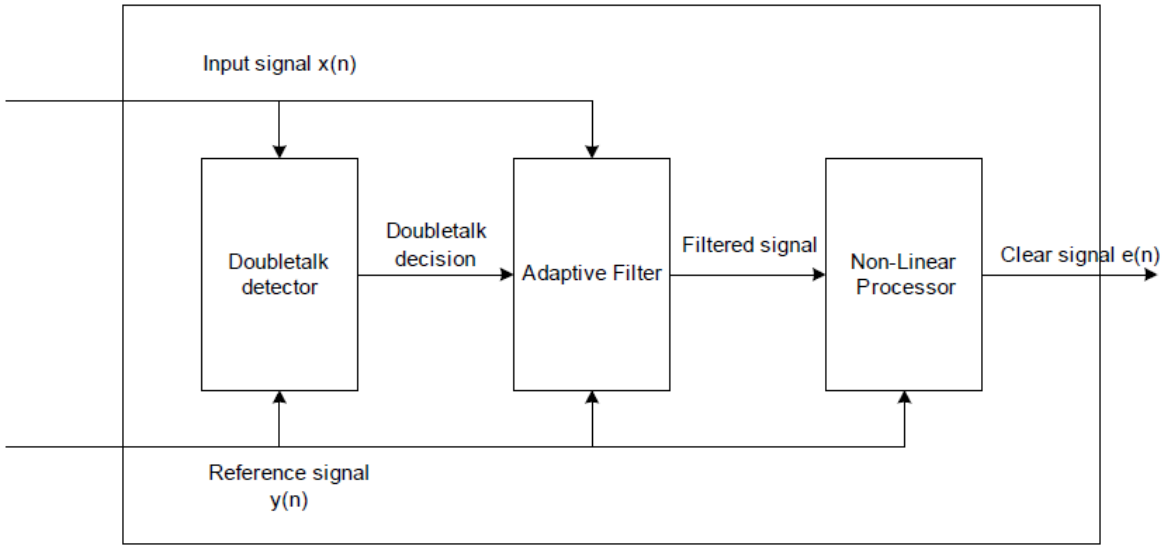


Figure 1.3 Block Diagram of a Generic Echo Canceller

1.5.1 Adaptive Filter

The adaptive filter is made up of an echo estimator and a subtractor. The echo estimator monitors the received path and dynamically builds a mathematical model of the line that creates the returning echo. The model of the line is convolved with the voice stream on the receive path. This yields an estimate of the echo, which is applied to the subtractor. The subtractor eliminates the linear part of the echo from the line in the send path. The echo canceller is said to converge on the echo as an estimate of the line is built through the adaptive filter.

1.5.2 Doubletalk Detector

A doubletalk detector is used with an echo canceller to sense when far-end speech is corrupted by near-end speech. The role of this important function is to freeze adaptation of the model filter when near-end speech is present. This action prevents divergence of the adaptive algorithm.

1.5.3 Nonlinear Processor

The non-linear processor evaluates the residual echo, which is nothing but the Amount of left over echo after the signal has passed through the adaptive filter. The Nonlinear processor removes all signals below a certain threshold and replaces them with Simulated background noise which sounds like the original background noise without the echo.

1.6 Echo Cancellation Challenges

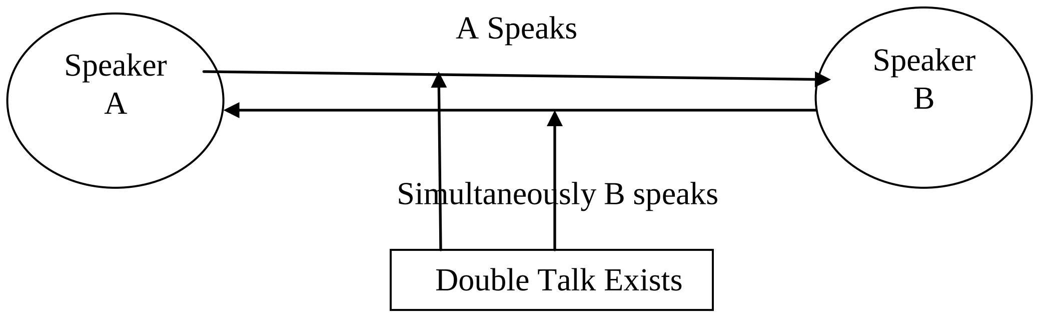
An echo canceller has to deal with a number of challenges in order to perform robust echo cancellation. They are

1.6.1 Avoiding Divergence

Divergence is an adaptive filter problem that arises when suitable solution for the line model is not found through the use of a mathematical algorithm. Under specific conditions (when parameters are not tuned accordingly) certain algorithms are bound to diverge and corrupt the signal or even add echo to the line. Good echo cancellers are tuned to avoid divergence situations in all conditions.

1.6.2 Handling Double Talk

In an active conversation, both end users often speak at the same time or interrupt each other. Those situations are called Double talk. Double talk presents a special processing challenge to echo cancellers.



A good echo canceller must detect and distinguish the double talk from the background noise. After detecting the double talk it should not update the filter model to avoid incurring of divergence which could result.

A smooth transition between double talk detection and its processing is necessary so that the end users feel the normal conversation.

1.6.3 Preventing Clipping

When a part of the speech is erroneously removed during conversation clipping is said to be occurred. This is due to the lack of precision in the NLP, as NLP fails to start and stop at the right time. NLP typically is not rapid enough to respond when speech is introduced at the near end/local end. Due to this, parts of speech signal (words) are replaced with background noise. The above may happen when NLP confuses with the voice level fading with that of residual echo.

CHAPTER 2

ADAPTIVE FILTER

2.1 Introduction

The below given figure shows the block diagram for the adaptive filter method

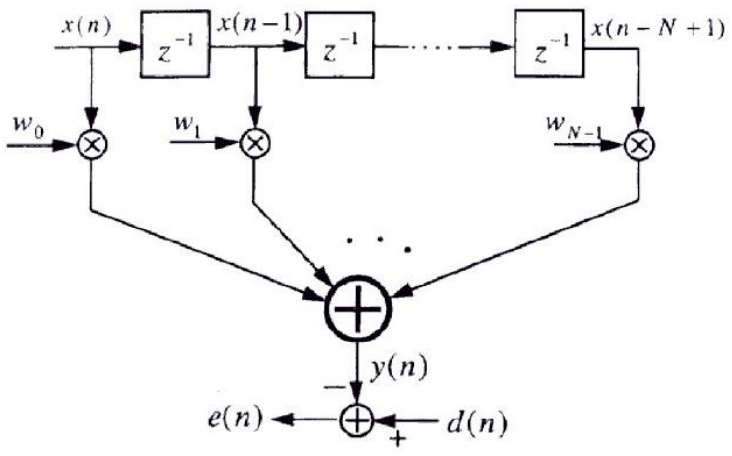


Figure 2.1 Block diagram for the adaptive filter method

Here w represents the coefficients of the FIR filter tap weight vector, x(n) is the input vector samples, z^{-1} is a delay of one sample period, y(n) is the adaptive filter output, d(n) is the desired echoed signal and e(n) is the estimation error at time n.

The aim of an adaptive filter is to calculate the difference between the desired signal and the adaptive filter output, e(n). This error signal is fed back into the adaptive filter and its coefficients are changed algorithmically in order to minimize a function of this difference, known as the cost function. In the case of acoustic echo cancellation, the optimal output of the adaptive filter is equal in value to the unwanted echoed signal.

When the adaptive filter output is equal to desired signal the error signal goes to zero. In this situation the echoed signal would be completely cancelled and the far user would not hear any of their original speech returned to them.

The various methods can be divided into two groups based on their cost functions. The first class are known as Mean Square Error (MSE) adaptive filters, they aim to minimize a cost function equal to the expectation of the square of the difference between the desired signal d(n), and the actual output of the adaptive filter y(n).

$$\xi(n) = E\left[e^2(n)\right] = E\left[(d(n) - y(n))^2\right] \tag{2.1}$$

The second class are known as Recursive Least Squares (RLS) adaptive filters and they aim to minimize a cost function equal to the weighted sum of the squares of the difference between the desired and the actual output of the adaptive filter for different time instances. The cost function is recursive in the sense that unlike the MSE cost function, weighted previous values of the estimation error are also considered.

Wiener filters are a special class of transversal FIR filters which build upon the mean square error cost function to arrive at an optimal filter tap weight vector which reduces the MSE signal to a minimum. The optimal wiener solution is the set of filter tap weights which reduce the cost function to zero. This vector can be found as the product of the inverse of the input vector autocorrelation matrix and the cross correlation vector between the desired signal and the input vector.

2.2 Adaptive filter Algorithms

Several algorithms are implemented to achieve the best outcome for cancellation of acoustic echo. The algorithms include

2.2.1 Least Mean Square (LMS)

LMS is a search algorithm that is widely used in various applications of adaptive filtering. Due to low computational complexity and proof of convergence in stationary environment and stable behavior made this algorithm attractive.

2.2.2 Normalized Least Mean Square (NLMS)

Many algorithms are derived from conventional LMS algorithm for adaptive filters. The main purpose behind alternative LMS is to reduce computational complexity and convergence time. The NLMS uses variable convergence factor that reduces instantaneous error and convergence time but leads to maladjustment.

The updating algorithm for NLMS algorithm with variable convergence factor µk is given by

$$w(k+1) = w(k) + (e(k)x(k))/(xT(k)x(k) \qquad (2.2)$$

Where µk is given by µk = 1/(2x^(k)*x(k))

The Least Mean Square algorithm of adaptive filtering attempts to find the optimal wiener solution using estimations based on instantaneous values. The LMS algorithm is a type of adaptive filter known as stochastic gradient-based algorithms as it utilizes the gradient vector of the filter tap weights to converge on the optimal wiener solution. With every iteration of the LMS algorithm, the filter tap weights of the adaptive filter are updated.

One of the primary disadvantages of the LMS algorithm is having a fixed step size parameter for every iteration. This requires an understanding of the statistics of the input signal prior to commencing the adaptive filtering operation. In practice this is rarely achievable. Even if we assume the only signal to be input to the adaptive echo cancellation system is speech, there are still many factors such as signal input power and amplitude which will affect its performance. The normalized least mean square algorithm (NLMS) is an extension of the LMS algorithm which bypasses this issue by selecting a different step size value, µ(n), for each iteration of the algorithm. This step size is proportional to the inverse of the total expected energy of the instantaneous values of the coefficients of the input vector **x**(n). This sum of the expected energies of the input samples is also equivalent to the dot product of the input vector with itself, and the trace of input vectors auto-correlation matrix, **R.**

$$tr[\mathbf{R}] = \sum_{i=0}^{N-1} E\left[x^2(n-i)\right] = E\left[\sum_{i=0}^{N-1} x^2(n-i)\right], \tag{2.3}$$

The recursion formula for the NLMS algorithm.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\mathbf{x}^T(n)\mathbf{x}(n)} e(n)\mathbf{x}(n) \tag{2.4}$$

2.2.2.1 Derivation of the NLMS algorithm

This derivation of the normalized least mean square algorithm is based on LMS concept. To derive the NLMS algorithm we consider the standard LMS recursion, for which we select a variable step size parameter, μ(n). This parameter is selected so that the error value, e+ (n), will be minimized using the updated filter tap weights, w (n+1), and the current input vector, **x**(n).

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n) \\ e^+(n) &= d(n) - \mathbf{w}^T(n+1)\mathbf{x}(n) \\ &= (1 - 2\mu(n)\mathbf{x}^T(n)\mathbf{x}(n))e(n) \end{aligned} \tag{2.5}$$

Next we minimize $(e^+(n))^2$, with respect to μ(n). Using this we can then find a value for μ(n) which forces e+(n) to zero.

$$\mu(n) = \frac{1}{2\mathbf{x}^T(n)\mathbf{x}(n)} \tag{2.6}$$

This μ(n) is then substituted into the standard LMS recursion replacing μ, resulting in the following.

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

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\mathbf{x}^T(n)\mathbf{x}(n)}e(n)\mathbf{x}(n) \qquad (2.7)$$

Often the NLMS algorithm is expressed as the equation given below, this is a slight modification of the standard NLMS algorithm detailed above. Here the value of ψ is a small positive constant in order to avoid division by zero when the values of the input vector are zero. The parameter μ is a constant step size value used to alter the convergence rate of the NLMS algorithm, it is within the range of $0<\mu<2$, usually being equal to 1.

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

$$\text{where } \mu(n) = \frac{\mu}{\mathbf{x}^T\mathbf{x} + \psi} \qquad (2.8)$$

2.2.2.2 Implementation of the NLMS algorithm

As the NLMS is an extension of the standard LMS algorithm, the NLMS algorithms practical implementation is very similar to that of the LMS algorithm. Each iteration of the NLMS algorithm requires these steps in the following order.

1. The output of the adaptive filter is calculated.

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

2. An error signal is calculated as the difference between the desired signal and the filter output.

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

3. The step size value for the input vector is calculated.

$$\mu(n) = \frac{1}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)}$$

4. The filter tap weights are updated in preparation for the next iteration.

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

Each iteration of the NLMS algorithm requires 3N+1 multiplications, this is only N more than the standard LMS algorithm and this is an acceptable increase considering the gains in stability and echo attenuation achieved.

Initial Condition	$0 < \mu_n \leq 2$ $x(0) = w(0) = [0,\cdots,0]^T$ γ = a small constant
For each instant of time, k = 1, 2, ⋯, compute	
Filter output:	$y(k) = x(k)^T w(k)$
Estimation Error:	$e(k) = d(k) - \hat{y}(k)$
Tap-Weight Adaptation:	$w(k+1) = w(k) + \frac{2\mu_n}{\gamma + x^T(k)x(k)} e(k)x(k)$

Table 2.1: **Implementation of the NLMS algorithm.**

2.2.3 VSS-NLMS based Echo Canceller Implementation

2.2.3.1 Method 1:

Given the input vector $\mathbf{X}k$, the Euclidean norm of the input vector $I\mathbf{X}_k 1^2$, the NLMS algorithm with fixed step size, μ, for adjusting the adaptive echo canceller's coefficients at time instant k is defined as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu e_k \frac{\mathbf{x}_k}{\|\mathbf{x}_k\|^2} , \tag{2.9}$$

where the error ek is defined as $ek = dk + nk - \mathbf{X}^T{}_k\, \mathbf{w}k$, dk is the desired value and nk is the additive noise. In this work, the fixed step size μ in (1) is made variable and is up-dated according to the following recursion:

$$\mu_k = \mu_{k-1} - \frac{\rho}{2} \frac{\partial e_k^2}{\partial \mu_{k-1}}, \tag{2.10}$$

This can be transformed, after substituting Equation (1), to the form

$$\mu_k = \mu_{k-1} + \rho e_k e_{k-1} \frac{\mathbf{x}_k^T \mathbf{x}_{k-1}}{\|\mathbf{x}_{k-1}\|^2} , \tag{2.11}$$

Where the parameter ρ is a small positive constant that controls the adaptive behavior of the step-size sequence μk and T denotes transpose operation. Accordingly, the coefficients of the VSS-NLMS echo canceller will be updated according to a variable step-size NLMS (VSSNLMS) algorithm given by:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu_k e_k \frac{\mathbf{x}_k}{\|\mathbf{x}_k\|^2} , \tag{2.12}$$

Where the variable step size parameter is confined to the following limits :

$$\mu_k = \begin{cases} \mu_{max} & \text{if } \mu_k > \mu_{max} \\ \mu_{min} & \text{if } \mu_k < \mu_{min} \\ \mu_k & \text{otherwise,} \end{cases} \quad , \qquad (2.13)$$

And μmin, μmax are chosen to satisfy the convergence requirements of the NLMS algorithm with fixed step size, that is $0 < \mu min < \mu max < 2$.

2.2.3.2 Method 2:

The purpose of a variable step-size normalized LMS filter is to solve the dilemma of fast convergence rate and low excess MSE. In the past two decades, many VSS-NLMS algorithms have been presented and have claimed that they have good convergence and tracking properties.

Adaptive filtering algorithms have been widely employed in many signal processing applications. Among them, the normalized least mean square (NLMS) adaptive filter is most popular due to its simplicity. The stability of the basic NLMS is controlled by a fixed step-size constant *ì*, which also governs the rate of convergence, speed of tracking ability and the amount of steady-state excess mean-square error (MSE). In practice, the NLMS is usually implemented by adding the squared norm of input vector with a small positive number *å*, commonly called the regularization parameter. For the basic *å*-NLMS algorithm, the role of *å* is to prevent the associated denominator from getting too close to zero, so as to keep the filter from divergence. Since the performance of *å*-NLMS is affected by the overall step-size parameter, the regularization parameter has an effect on convergence properties and the excess MSE as well, i.e., a too big *å* may slow down the adaptation of the filter in certain applications.

There are conflicting objectives between fast convergence and low excess MSE for NLMS with fixed regularization parameter. In the past two decades, many variable step-size NLMS (VSS-NMS) algorithms have been proposed to solve this dilemma of the conventional NLMS [1-11]. For example, Kwong used the power of instantaneous error to introduce a variable step-size LMS (VSSLMS) filter [6]. This VSSLMS has a larger step size when the error is large, and has a smaller step size when the error is small. Later Aboulnasr pointed out that

VSSLMS algorithm is fairly sensitive to the accompanying noise, and presented a modified VSSLMS.

Another type of VSS algorithms has time-varying regularization parameter that is fixed in the conventional å-NLMS filters. By making the regularization parameter gradient-adaptively, Mandic presented a generalized normalized gradient descent (GNGD) algorithm. Mandic claimed that the GNGD adapts its learning rate according to the dynamics of the input signals, and its performance is bounded from below by the performance of the NLMS. Very recently, Mandic introduced another scheme with hybrid filters structure to further improve the steady-state maladjustment of the GNGD. Choi, Shin, and Song then proposed a robust regularized NLMS (RR-NLMS) filter, which uses a normalized gradient to update the regularization parameter. While most variable step-size algorithms need to tune several parameters for better performance, we presented an almost tuning-free generalized square-error-regularized NLMS algorithm (GSER) recently. Our GSER exhibits very good performance with fast convergence, quick tracking and low steady-state MSE.

2.2.4 Generalized Normalized Gradient Descent Algorithm (GNGD)

The GNGD belongs to the family of time-varying regularized VSS algorithm. Mandic presented a generalized normalized gradient descent (GNGD) algorithm. Mandic claimed that the GNGD adapts its learning rate according to the dynamics of the input signals, and its performance is bounded from below by the performance of the NLMS. Very recently, Mandic introduced another scheme with hybrid filters structure to further improve the steady-state maladjustment of the GNGD. The filter coefficient vector is updated as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu_c}{\|\mathbf{x}(n)\|^2 + \varepsilon(n)} e(n)\mathbf{x}(n) , \tag{2.14}$$

Where μ_c is a fixed step size and the regularization parameter, $\varepsilon(n)$ is recursively calculated as

$$\varepsilon(n) = \varepsilon(n-1) - \rho\mu_c \frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\|\mathbf{x}(n-1)\|^2 + \varepsilon(n-1))^2} , \tag{2.15}$$

Where ρ an adaptation parameter is needs tuning, and the initial value $\varepsilon(0)$ has to be set as well.

2.2.5 Robust Regularization NLMS Algorithm (RR-NLMS)

Choi, Shin, and Song then proposed a robust regularized NLMS (RR-NLMS) filter, which uses a normalized gradient to update the regularization parameter. While most variable step-size algorithms need to tune several parameters for better performance. Choi's RR-NLMS algorithm is a modified version of GNGD. The regularization parameter is updated as

$$\varepsilon(n) = \varepsilon(n-1) - \rho\mu_c \frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\|\mathbf{x}(n-1)\|^2 + \varepsilon(n-1))^2}, \tag{2.16}$$

Where, sgn(x) represents the sign function, and $\varepsilon_{\min}$ is a parameter needs tuning.

2.2.6 Generalized Square Error Regularized (GSER) NLMS Algorithm

An almost tuning-free generalized square-error-regularized NLMS algorithm (GSER) which exhibits very good performance with fast convergence, quick tracking and low steady-state MSE.

The GSER updates **w**(n) as follows

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\hat{\sigma}_e^2(n)\mu_c}{\hat{\sigma}_e^2(n)\|\mathbf{x}(n)\|^2 + \theta} e(n)\mathbf{x}(n), \tag{2.17}$$

Where θ is a positive parameter that makes the filter more general and the power of the error signal is estimated as.

$$\hat{\sigma}_e^2(n) = \beta\hat{\sigma}_e^2(n-1) + (1-\beta)e^2(n). \tag{2.18}$$

CHAPTER 3

DOUBLE TALK DETECTOR

3.1 Introduction

An important characteristic of a good echo canceller is its performance during double talk. The condition where both ends, the near-end and the far-end, are speaking is referred to as double talk. If the echo canceller does not detect a double talk condition properly the near end speech will cause the adaptive filter to diverge. Therefore, it is important to have a reliable double-talk detector. A DTD is used with an echo canceller to sense when the far-end speech is corrupted by the near-end speech. The role of this important function is to freeze adaptation of the model filter, hˆ, when the near-end speech, *v*, is present in order to avoid divergence of the adaptive algorithm. The far-end talker signal, *x*, is filtered with the impulse response, *h*, and the resulting signal. The echo is added to the near-end speech signal, *v*, in order to obtain the corrupted signal

$$d(n) = H^T x(n) + v(n) \tag{3.1}$$

Where

$$H = [H_0, H_1, \cdots, H_{L-1}]^T \tag{3.2}$$

and

$$x(n) = [x(n), x(n-1), \cdots, x(n-L+1)]^T. \tag{3.3}$$

L is the length of the echo path. The error signal at time *n* is defined by

$$e(n) = d(n) - \hat{H}^T x(n). \tag{3.4}$$

This error signal is used in the adaptive algorithm to adjust the L taps of the filter ,hˆ. For simplicity it is assumed that the length of the signal vector, *x*, is the same as the effective length of the echo path, *h*. When *v* is not present, with any adaptive algorithm, hˆ will quickly converge to an estimate of *h*, which is the best way to cancel the echo. When *x* is not present, or very small, adaptation is halted by the nature of the adaptive algorithm. When both *x* and *v* are present the near-end talker signal could disrupt the adaptation of hˆ and cause divergence. Therefore, the goal of a double talk detection algorithm is to stop the adaptation of hˆ when the level of *v* becomes significant in relation to the level of *x* and to keep the adaptation going when the level of *v* is negligible.

The basic double talk detection process starts with computing a detection statistic and comparing it with a preset threshold. Different methods have been proposed to form the detection statistic. The Geigel algorithm has proven successful in line echo cancellers.

3.2 The Geigel Algorithm

One simple algorithm for the double talk detection is the Geigel algorithm. Giegel declares the presence of near-end speech whenever

$$\xi = \frac{\max\{|x(k)|,\cdots,|x(k-N+1)|\}}{|d(k)|} < T \tag{3.5}$$

Where N and T are suitably chosen constants. This detection scheme is based on a waveform level comparison between the microphone signal, *d*, and the far-end speech, *x*, assuming the near-end speech, *v*, in the microphone signal will be stronger than the echo. The maximum, or norm, of the N most recent samples of *x* is chosen for the comparison due to uncertain delay in the echo path. The threshold, T, is used to compensate for the energy level of the echo path response, *h*, and is often set to ½ for line echo cancellers since the hybrid loss is typically approximately 6dB. However, for an AEC, it is not easy to set a universal threshold that will work reliably in all the various situations since the loss through the acoustic echo path can vary greatly depending on many factors. For N, one easy choice is to set it equal to the adaptive filter length L.

3.3 DTD Implementation

The proposed DTD algorithm implemented in the present project are shown in steps.

Step 1: Form an error vector with its initial values equal to zero.

Step 2: Keep the new error sample calculated as the difference between the original echo and output of the adaptive FIR filter of AEC, on the top of the error vector like what we do for the signal vector.

Step 3: Find the norm of the error vector i.e. calculate the error energy or variance.

Step 4: If the error norm is less than a minimum threshold value (say EPSILON = 0.000001), then make the error norm equal to that value to not to let the norm of the error vector equal to zero or a too low value.

Step 5: If the norm of the error vector is less than 0.1 times the norm of the signal vector, which happens when near end speaker is silent, then the effective step size is made equal to step size μ times inverse of the norm of the signal vector as what is done in the NLMS based AEC.

Step 6: If the norm of the error vector is more than 0.1 times the norm of the signal vector, which happens when near end speaker is speaking, then the step size is made equal to the product of step size μ and the inverse of the four times of the sum of the input vector and the error vector norms.

Step 7: Filter coefficient update is done using the update step-size modified in accordance with step 5 or step 6 depending upon whether single talk or double talk scenario.

Step 8: After filter update, the NLP is implemented if the double talk is not detected. This algorithm is simple, robust and is very quick in determining the presence of double talk or not.

In the simulation, initially single talk situation exists and after a specified duration, near speech signal which is smaller in length to the far end signal is added to the echo thus error signal contains a mixture of residual echo and the near end speech signal. After the near end speech signal is over, again the far end signal alone remains and the error signal containing only the residual echo is observed.

The output signal is recorded and is observed using the WAVOSAUR software. It can be observed that there exists no divergence of the residual echo during the double talk period and also the convergence time of the residual echo, after the near end speech signal ends, is also very less.

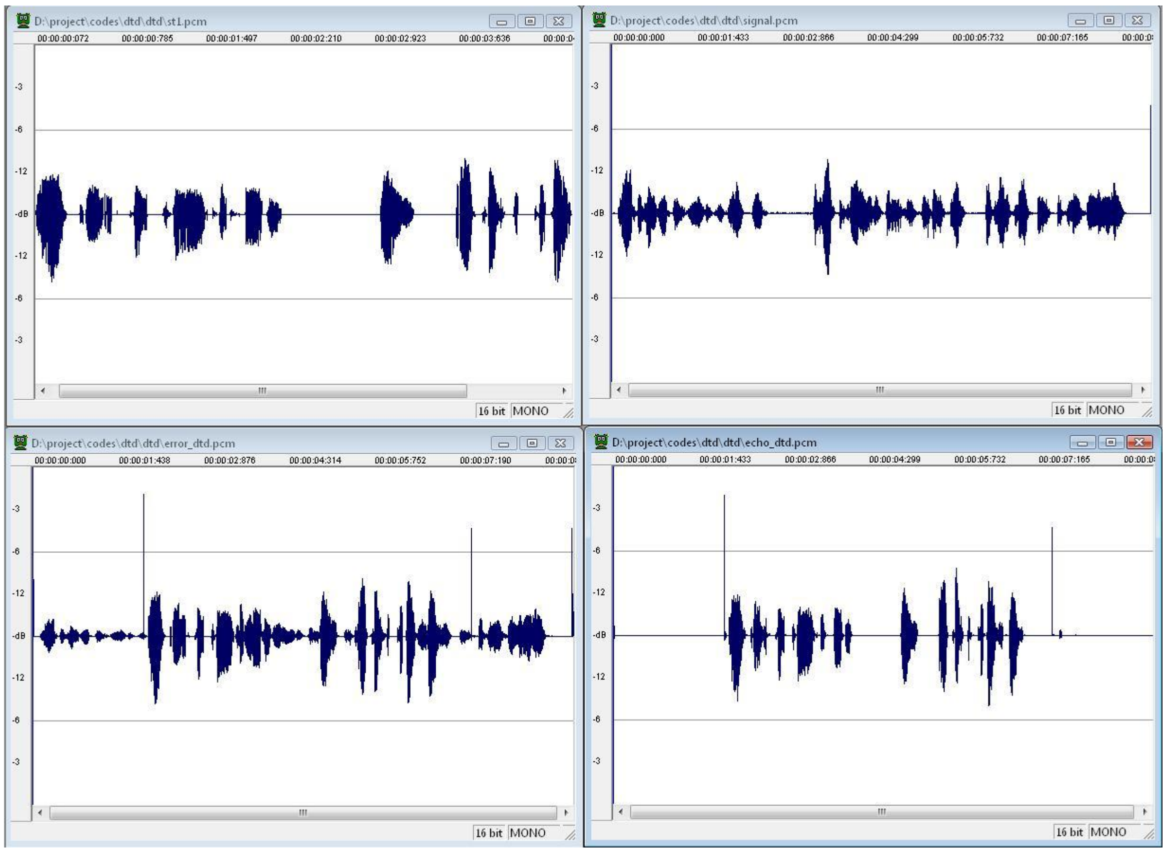

D:\project\codes\dtd\dtd\st1.pcm
D:\project\codes\dtd\dtd\signal.pcm
D:\project\codes\dtd\dtd\error_dtd.pcm
D:\project\codes\dtd\dtd\echo_dtd.pcm
16 bit
MONO

Figure 3.1Near End Speech Signal input and Near End Speech Signal Output during single talk period initially and double talk period and then again single talk situation.

CHAPTER 4

HARDWARE AND SOFTWARE REQUIREMENTS

This chapter gives the hardware and software required for the implementation of project. And also it gives the system requirements, MATLAB version, CODE COMPOSITER STUDIO, VIUSIAL STUDIO and TMS320C6713DSK used.

4.1 System requirements

Desktop personnel computer with minimum of the following configuration.

Processor	: Intel centrino
Platform	: windows XP
RAM	: 120 MB

The project implements on **TI's Software Development Tool: Code Composer Studio** and **Spectrum Digital's TMS320C6713 DSK Board** which is mainly used for audio interfacing.

4.2 Hardware

4.2.1 Introduction to TMS320C6x DSP

The number and variety of products that include some form of digital signal processing has grown dramatically over the past. DSP has become a key component in many consumers, communications, medical, and industrial products. These products use a variety of hardware approaches to implement DSP. Programmable "DSP processors," a class of microprocessors optimized for DSP have an advantage in terms of speed, cost, and energy efficiency.

From the outset, DSP processor architectures have been molded by DSP algorithms.

For nearly every feature found in a DSP processor, there are associated DSP algorithms whose computation is in some way eased by inclusion of this feature.

Therefore, perhaps the best way to understand the evolution of DSP architectures is to examine typical DSP algorithms and identify how their computational requirements have influenced the architectures of DSP processors.

4.2.2 Features

The C6713DSK has the following features

- The 6713 DSK is a low-cost standalone development platform that enables customers to evaluate and develop applications for the TI C67XX DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market.
- The DSK uses the 32-bit EMIF for the SDRAM (CE0) and daughter card expansion interface (CE2 and CE3). The Flash is attached to CE1 of the EMIF in 8-bit mode.
- An on-board AIC23 codec allows the DSP to transmit and receive analog signals. McBSP0 is used for the codec control interface and McBSP1 is used for data. Analog audio I/O is done through four 3.5mm audio jacks that correspond to microphone input, line input, line output and headphone output. The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP1 can be rerouted to the expansion connectors in software.
- A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD has a register based user interface that lets the user configure the board by reading and writing to the CPLD registers. The registers reside at the midpoint of CE1.
- The DSK includes 4 LEDs and 4 DIP switches as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers.

- An included 5V external power supply is used to power the board. On-board voltage regulators provide the 1.26V DSP core voltage, 3.3V digital and 3.3V analog voltages. A voltage supervisor monitors the internally generated voltage, and will hold the board in reset until the supplies is within operating specifications and the reset button is released. If desired, JP1 and JP2 can be used as power test points for the core and I/O power supplies.
- Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

4.2.3 Functional Overview

The DSP on the 6713 DSK interfaces to on-board peripherals through a 32-bit wide EMIF (External Memory Interface). The SDRAM, Flash and CPLD are all connected to the bus. EMIF signals are also connected daughter card expansion connectors which are used for third party add- in boards. The DSP interfaces to analog audio signals through an on-board AIC23 codec and four 3.5 mm audio jacks (microphone input, line input, line output, and headphone output). The codec can select the microphone or the line input as the active input. The analogg output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP0 is used to send commands to the codec control interface while McBSP1 is used for digital audio data. McBSP0and McBSP1 can be re- routed to the expansion connectors in software. A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The
CPLD has a register based user interface that lets the user configure the board by reading and writing to its registers.

4.2.4 Diagrams of TMS320C6713 DSK

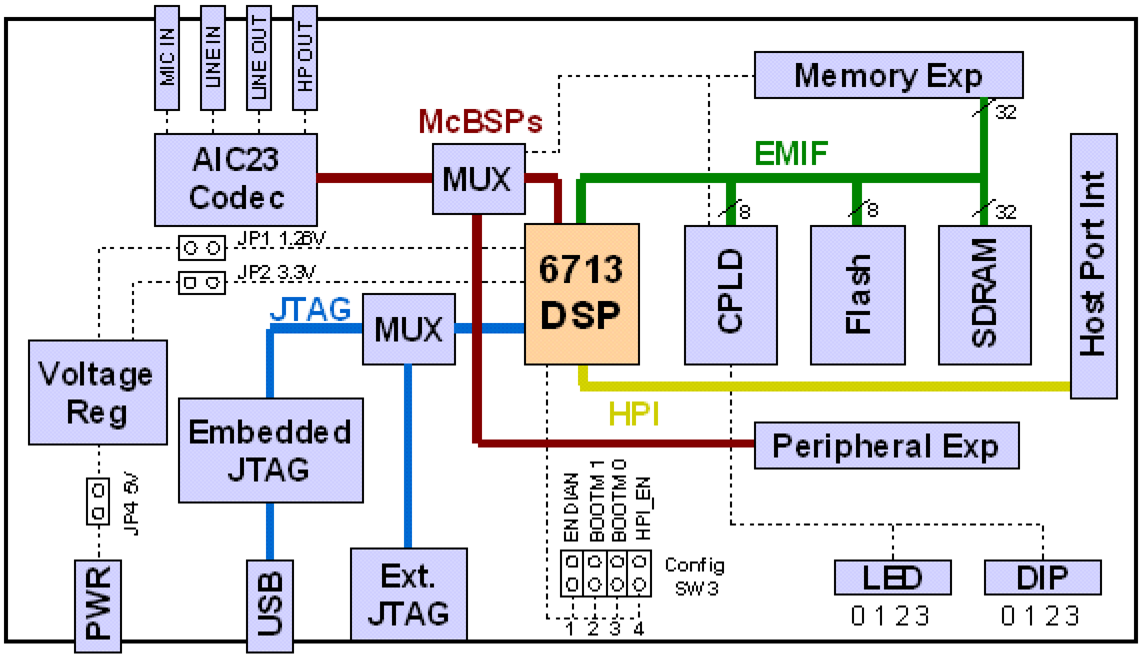


Figure 4.1 Block Diagram

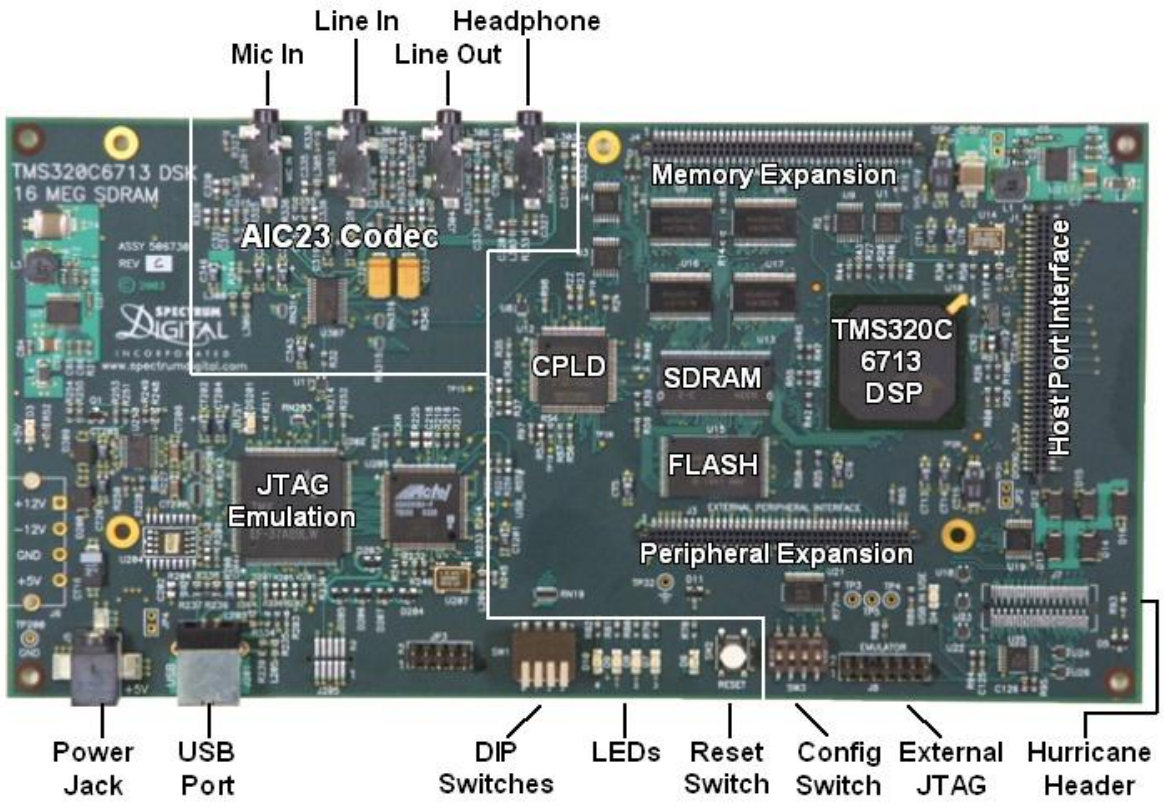


Figure 4.2 Board diagram of TMS320C6713 DSK

4.3 Software

4.3.1 Introduction to Code Composer studio

The Code Composer Studio development environment is used to build and debug embedded real-time software applications Code Composer is the DSP industry's first fully integrated development environment (IDE) with DSP-specific functionality. With a familiar environment liked MS-based C++TM, Code Composer lets you edit, build, debug, profile and manage projects from a single unified environment. Other unique features include graphical signal analysis, injection/extraction of data signals via file I/O, multi-processor debugging, automated testing and customization via a C-interpretive scripting language and much more.

4.3.1.1 Features

- IDE
- Debug IDE
- Advanced watch windows
- Integrated editor
- File I/O, Probe Points, and graphical algorithm scope probes
- Advanced graphical signal analysis
- Interactive profiling
- Automated testing and customization via scripting
- Visual project management system
- Compile in the background while editing and debugging
- Multi-processor debugging
- Help on the target DSP

4.3.2 Introduction to MATLAB

MATLAB is a programming environment for algorithm development, data analysis, visualization, and numerical computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN.

MATLAB can be used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology.

MATLAB provides a number of features for documenting and sharing work. MATLAB code can be integrated with other languages and applications, and distribute the MATLAB algorithms and applications.

4.3.2.1 Features

- High-level language for technical computing
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics function for visualizing data
- Tools for building custom graphical user interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C,C++, Fortran, Java, COM, and Microsoft Excel

4.3.2.2 Keywords

- **ECHO**:-Echo commands in M-files. ECHO ON turns on echoing of commands inside Script-files. ECHO OFF turns off echoing.
- **END: -** Terminate scope of FOR, WHILE, SWITCH, TRY, and IF statements. Without END's, FOR, WHILE, SWITCH, TRY, and IF wait for further input.
- **INPUT: -** Prompt for user input.

- **CFTOOL: -** Curve Fitting Tool. CFTOOL displays a window for fitting curves to data. You can create a data set using data in your workspace and you can create graphs of fitted curves superimposed on a scatter plot of the data.
- **IF**:-Conditionally execute statements.
- **FOR: -** Repeat statements a specific number of times.
- **Float: -** Create structure describing a floating point data type
- **SUBPLOT**:-Create axes in tiled positions.
- **PLOT: -** Linear plot. PLOT(X, Y) plots vector Y versus vector X. If X or Y is a matrix, then the vector is plotted versus the rows or columns of the matrix whichever line up.
- **FOPEN**: - Open file. FID = FOPEN (FILENAME) opens the file FILENAME for read access.
- **FCLOSE: -** Close file. ST = FCLOSE (FID) closes the file associated with file identifier FID

4.3.3 Introduction to VISUAL STUDIOS

Visual Studio is an integrated development environment (IDE). It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE,.NET Framework, .NET Compact Framework and Microsoft Silver light.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle.

Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages

include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#). It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

4.3.3.1 Features

4.3.3.1.1 **Code editor**

Visual Studio, like any other IDE, includes a code editor that supports syntax highlighting and code completion using IntelliSense for not only variables, functions and methods but also language constructs like loops and queries. IntelliSense is supported for the included languages, as well as for XML and for Cascading Style Sheets and JavaScript when developing web sites and web applications.

The Visual Studio code editor also supports setting bookmarks in code for quick navigation. The code editor also includes a multi-item clipboard and a task list. The code editor supports code snippets, which are saved templates for repetitive code and can be inserted into code and customized for the project being worked on. The Visual Studio code editor also supports code refactoring including parameter reordering, variable and method renaming, interface extraction and encapsulation of class members inside properties, among others.

4.3.3.1.2 **Debugger**

Visual Studio includes a debugger that works both as a source-level debugger and as a machine-level debugger. It works with both managed code as well as native code and can be used for debugging applications written in any language supported by Visual Studio. In addition, it can also attach to running processes and monitor and debug those processes. The Visual Studio debugger can also create memory dumps as well as load them later for debugging. Multi-threaded programs are also supported. The debugger can be configured to be launched when an application running outside the Visual Studio environment crashes.

The debugger allows setting breakpoints and watches (which monitor the values of variables as the execution progresses). The debugger supports *Edit and Continue*, i.e., it allows code to be edited as it is being debugged. When debugging, if the mouse pointer hovers over any variable, its current value is displayed in a tooltip ("data tooltips"), where it can also be modified if desired. During coding, the Visual Studio debugger lets certain functions be invoked manually from the immediate tool window.

4.3.3.1.3 **IntelliSense**

IntelliSense is the trademark feature of Visual Studio. IntelliSense simply helps you while programming by showing you the available classes and the methods and properties available on those classes.

4.3.3.1.4 **Designers**

Visual Studio includes visual WSYIWYG designers for Windows applications, ASP.NET applications, and Windows Mobile applications. These designers make it much easier to get the application looking just right.

4.3.3.1.5 **Organization**

Visual Studio is built for developing applications, so it provides intuitive methods for organizing various code files into projects.

4.3.3.2 Keywords

- **Else**: - Introduces a group of statements to be run or compiled if no other conditional group of statements has been run or compiled.
- **Double**: - The Double data type provides the largest and smallest possible magnitudes for a number. The default value of Double is 0. Holds signed IEEE 64-bit (8-byte) double-precision floating-point numbers that range in value from -1.79769313486231570E+308 through -4.94065645841246544E-324 for negative values and from 4.94065645841246544E-324 through 1.79769313486231570E+308

for positive values. Double-precision numbers store an approximation of a real number.

- **For**:-introduces a loop that is iterated with different values of a loop variable.
- **While**: - specifies a condition that enables the execution of a **Do** loop to continue
- **Char**: - Holds unsigned 16-bit (2-byte) code points ranging in value from 0 through 65535.Use the Char data type when you need to hold only a single character and do not need the overhead of String. The default value of Char is the character with a code point of 0.
- **Short:** - Holds signed 16-bit (2-byte) integers that range in value from -32,768 through 32,767. Use the Short data type to contain integer values that do not require the full data width of Integer. The default value of Short is 0.
- **Return:** - Returns control to the code that called a Function, Sub, Get, Set, or Operator procedure. In a Sub or Set procedure, the Return statement is equivalent to an Exit Sub or Exit Property statement, and *expression* must not be supplied.

CHAPTER 5

SOFTWARE DESIGN

5.1 Design of FIR Filters

A filter is something that transforms data to extract useful information from a noisy environment. In digital filtering there are two primary types; infinite impulse response (IIR) and finite impulse response FIR). IIR filters can normally achieve the same filter characteristics as a FIR filter using less memory and calculations with the cost of possibly becoming unstable. As the filters become more complex though, IIR filters needs more parameters and the advantages are reduced. Because of the high complexity of the many strong and sharp peaks in a room impulse response the filters that are being used in acoustic echo cancellation are usually of the FIR type.

Design of FIR filters using Windowing Method and implementation using MATLAB:

- Rectangular Window
- Hamming Window
- Barlett Window
- Kaiser Windom
- Blackman Window

5.2 Real Time Implementation of a FIR filter using TMS320C6713 DSK board

The Real-Time implementation of FIR system generates single tone signal of a specific frequency and is inputted to a FIR System having a given specific cut-off frequency and order.

This Real Time implementation is developed using TMS320C6713DSK Board. The output was observed for different single tone frequencies, for different FIR systems with different cut-off frequencies and different order. Real Time implementation of this application uses (Spectrum Digital's) **TMS320C5416 DSK** Board Specific CODEC API functions for the audio interfacing.

5.3 Real Time Implementation of Echo Generation

Real Time Implementation of Echo Generation for a speech signal using Delay-Buffer Logic with different Delay is done.

The output of this implementation consists of the main Speech Signal along with its Echo Signal.

- Single delay-Delayed version of the original signal. Signal is time delayed only once
- Multi delay-Signal is delayed in time more than once and is played along with the original signal.

5.4 Real Time Implementation of AEC on DSK

Real Time Implementation of **AEC** on DSK done by

- Configuring the input and out ports.
- Configuring DIP switches of the DSP as,
    - 1111 = Original signal.
    - 1100 = Echo of the original signal.
    - 0000 = Residual Echo.

Note: Any other switch configuration then the above mentioned will result in Residual echo only.

CHAPTER 6

SIMULATION RESULTS

The far-end signal is provided to the LINE-IN and the residual echo is obtained from the LINE-OUT of the TMS320C6713DSK. The standard speech vectors are used for far-signal inputs.

The far-end input signal ‘signal.pcm’, the corresponding echo signal ‘echo.pcm’ and the obtained residual echo ‘error.pcm’ are shown below.

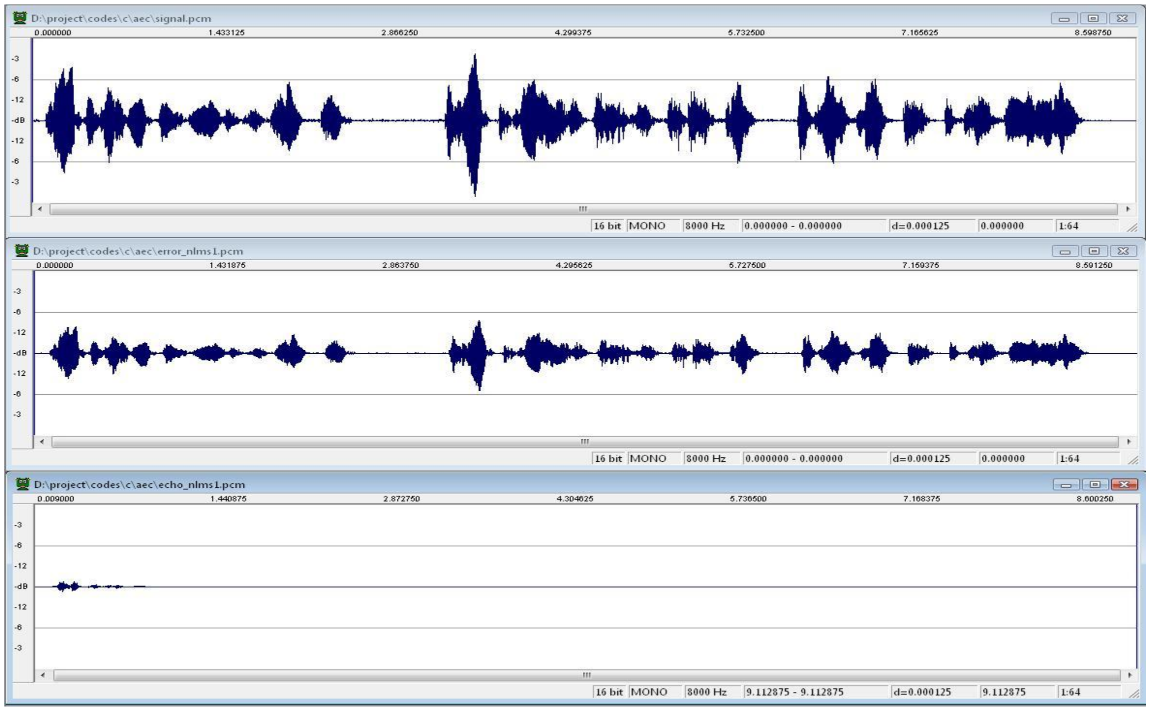

D:\project\codes\c\aec\signal.pcm
0.000000 1.433125 2.866250 4.299375 5.732500 7.165625 8.598750
-3 -6 -12 -dB -12 -6 -3
16 bit MONO 8000 Hz 0.000000 - 0.000000 d=0.000125 0.000000 1:64
D:\project\codes\c\aec\error_nlms1.pcm
0.000000 1.431875 2.863750 4.295625 5.727500 7.159375 8.591250
-3 -6 -12 -dB -12 -6 -3
16 bit MONO 8000 Hz 0.000000 - 0.000000 d=0.000125 0.000000 1:64
D:\project\codes\c\aec\echo_nlms1.pcm
0.009000 1.440875 2.872750 4.304625 5.736500 7.168375 8.600250
-3 -6 -12 -dB -12 -6 -3
16 bit MONO 8000 Hz 9.112875 - 9.112875 d=0.000125 9.112875 1:64

Figure 6.1 The far-end signal, echo signal and the residual echo.

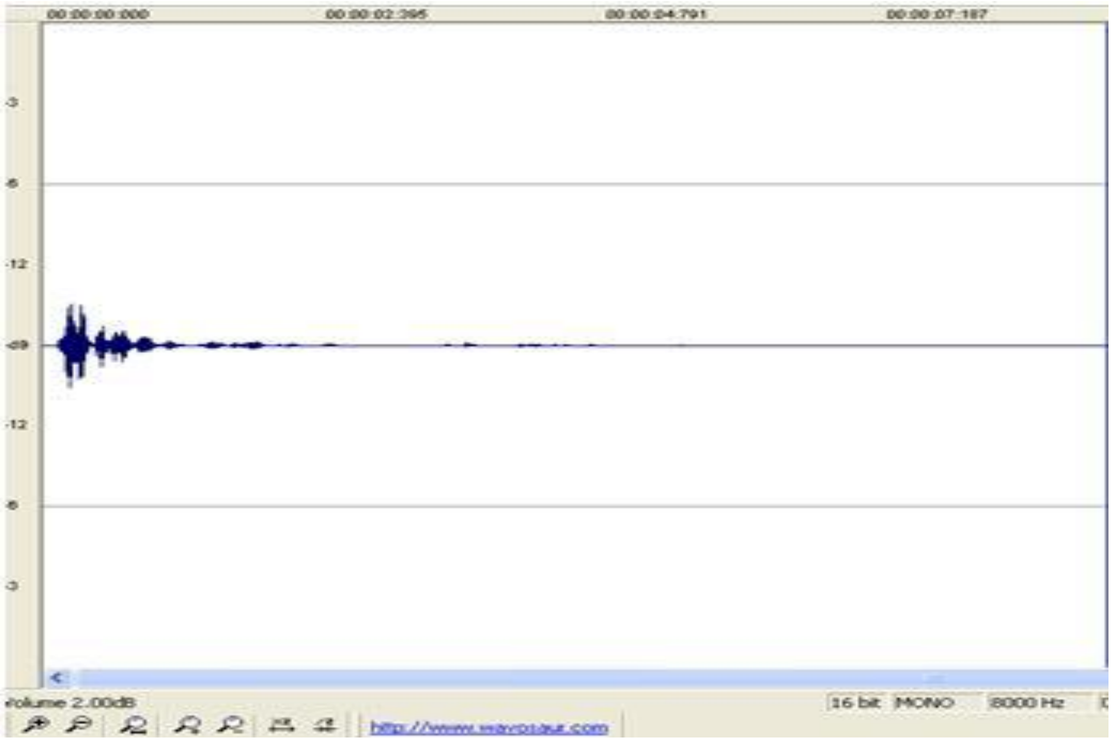


Figure 6.2 Residual Echo Signal for the NLMS method based AEC System

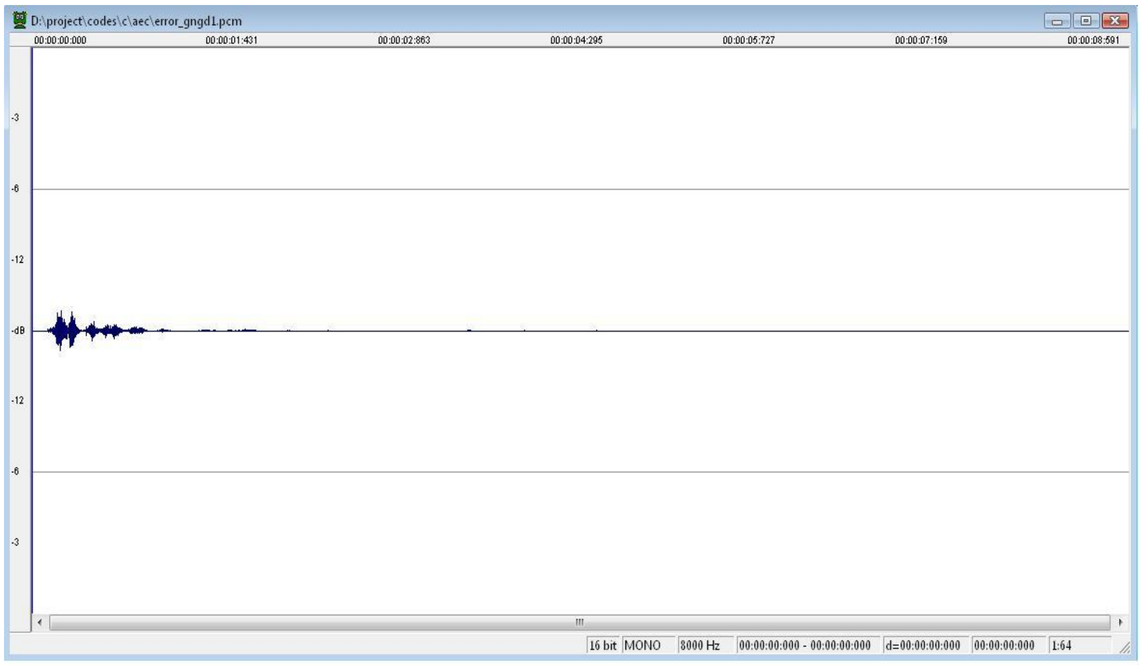


Figure 6.3 Residual Echo Signal for GNGD method based AEC system

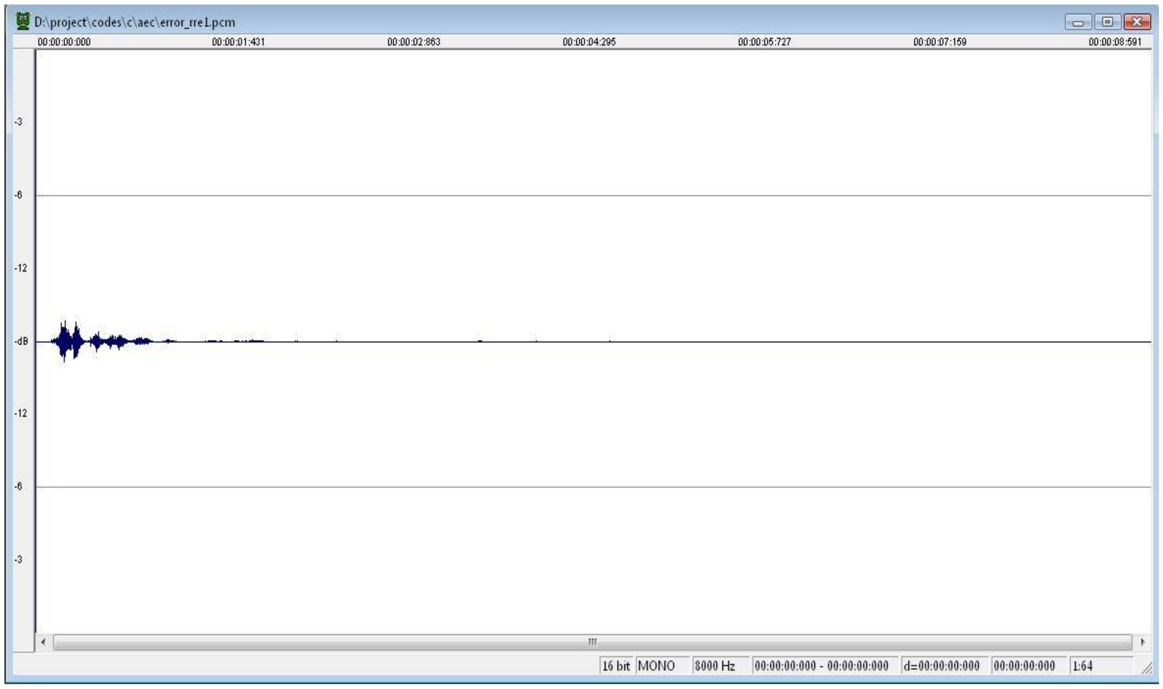


Figure 6.4 Residual echo signal for RR method based AEC system

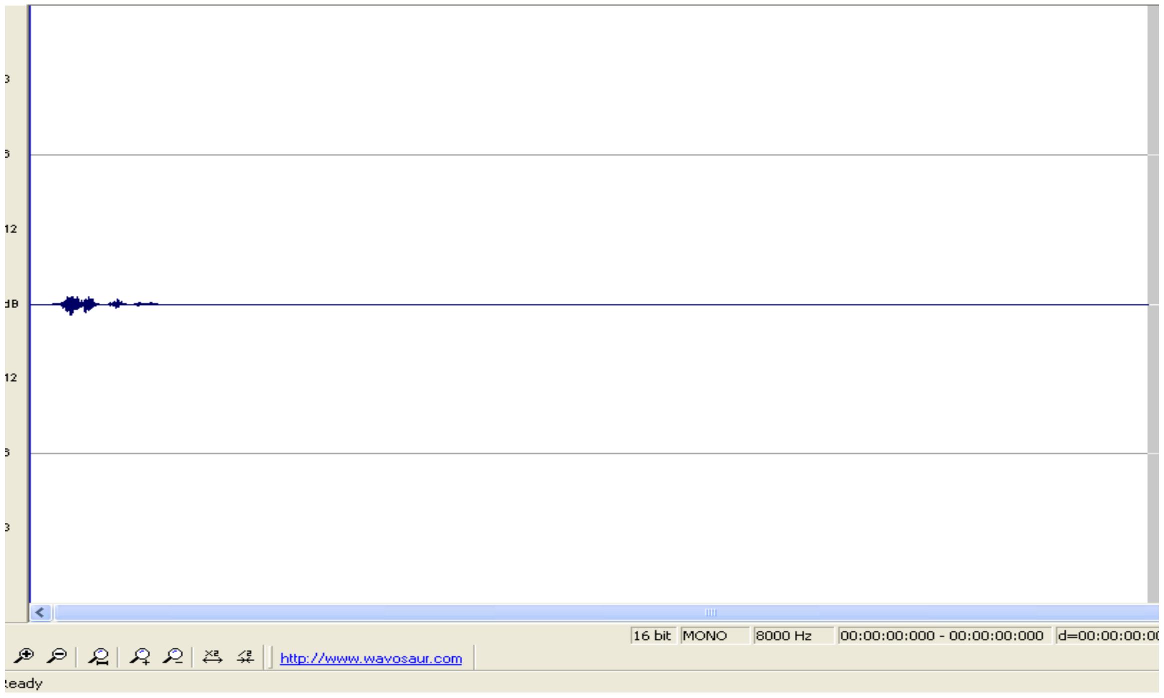


Figure 6.5 Residual Echo Signal for the GESR method based Acoustic Echo Cancellation System

6.1 Echo and Echo^ plots respectively with random signal as input

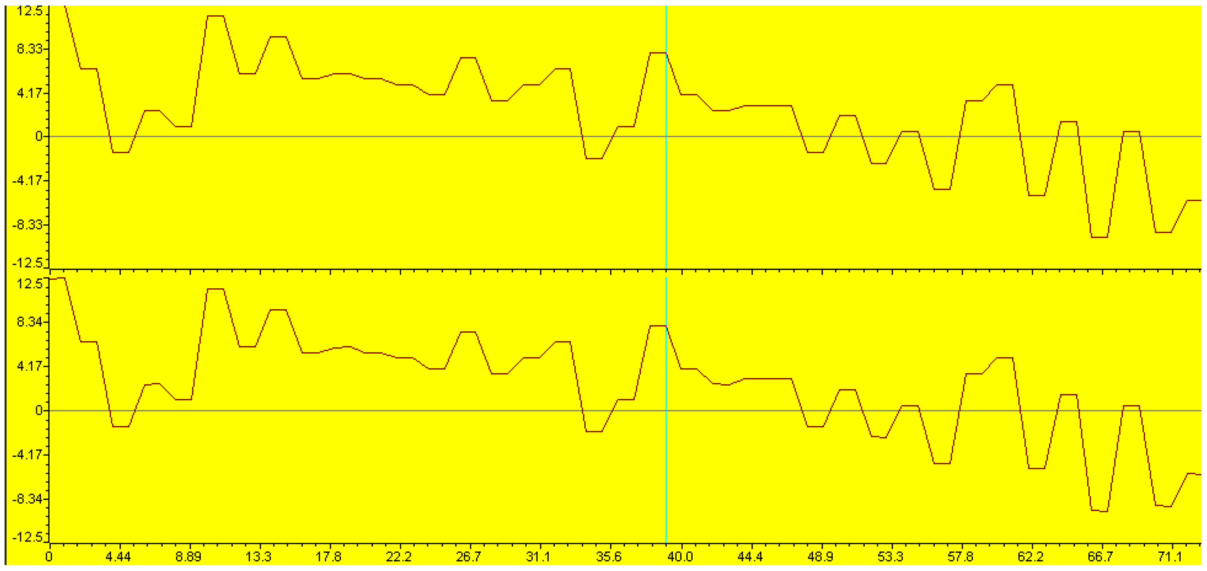


Figure 6.6 Simulation result for Real time signal using NLMS algorithm

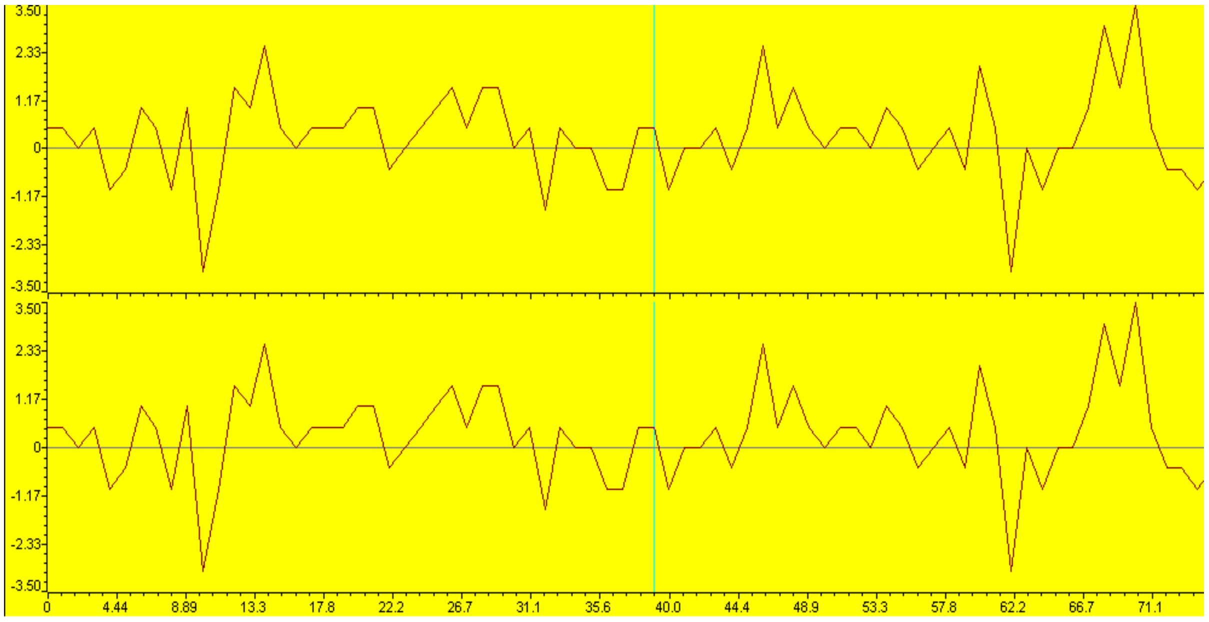


Figure 6.7 Simulation result for Real time signal using GNGD algorithm

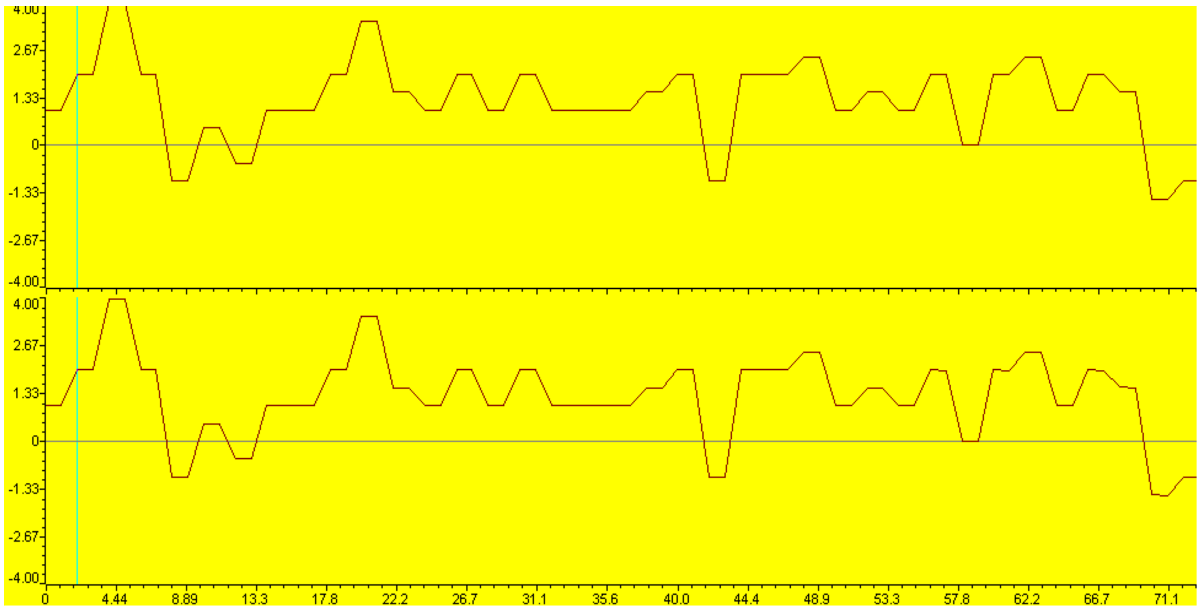


Figure 6.8 Simulation result for Real time signal using RR algorithm

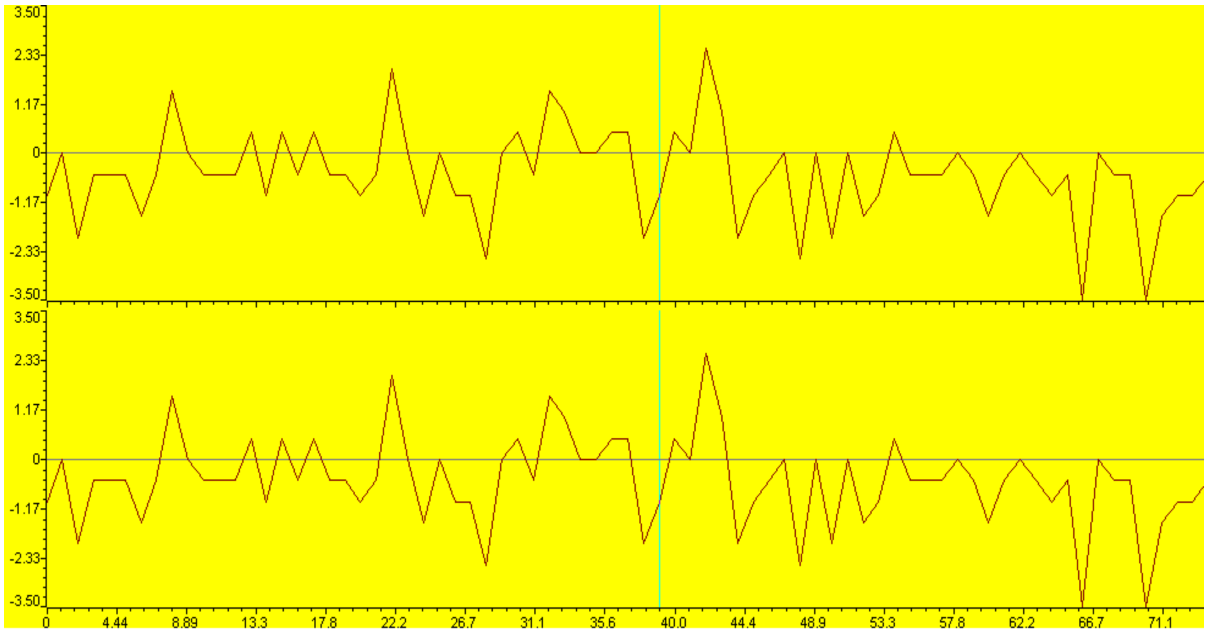


Figure 6.9 Simulation result for Real time signal using GSER algorithm

6.2 Echo Return Loss Enhancement (ERLE)

The measure of performance used is the Echo Return Loss Enhancement (ERLE) defined as

$$ERLE = 10log_{10}\frac{E[y_k^{*2}]}{E[\{y_k^* - y_k\}^2]}\ dB\ ,$$

Where $y_k^* = \mathbf{x}_k^T\mathbf{w}^*$ is the true echo, $\mathbf{w}^*$ is the impulse response of the echo path, and $y_k = \mathbf{x}_k^T\mathbf{w}$ is the simulated one.

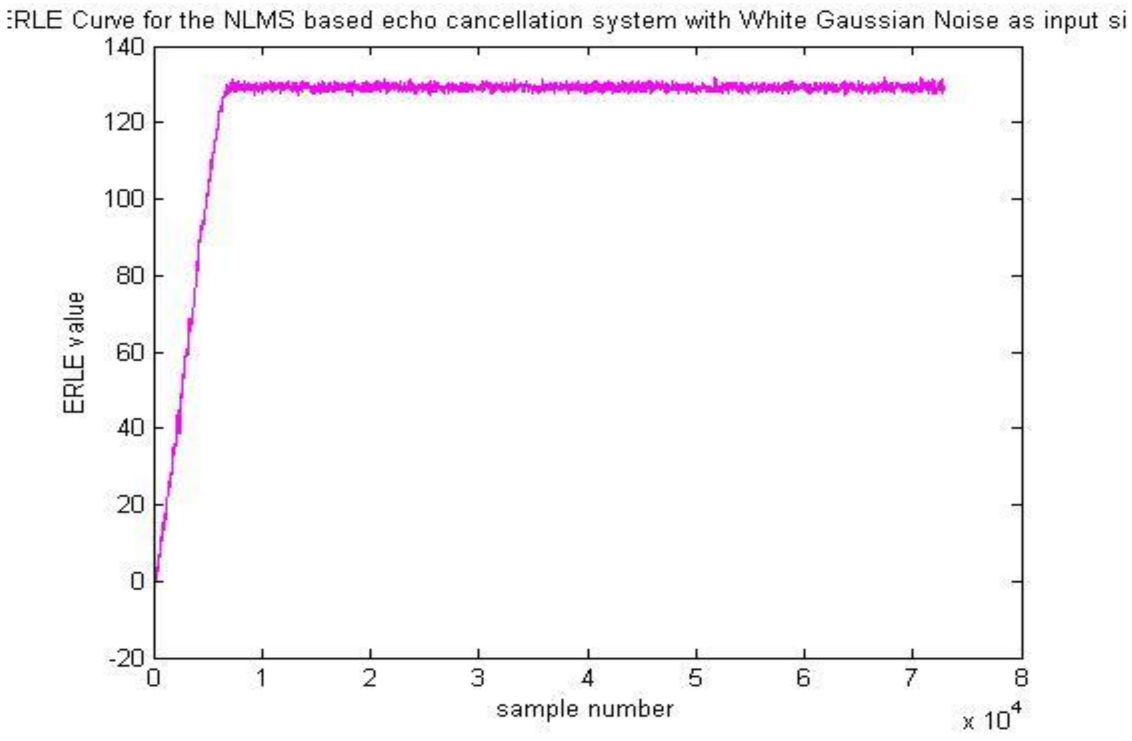


Figure 6.10 ERLE curve for NLMS based AEC with White Gaussian Noise samples as the far-end input

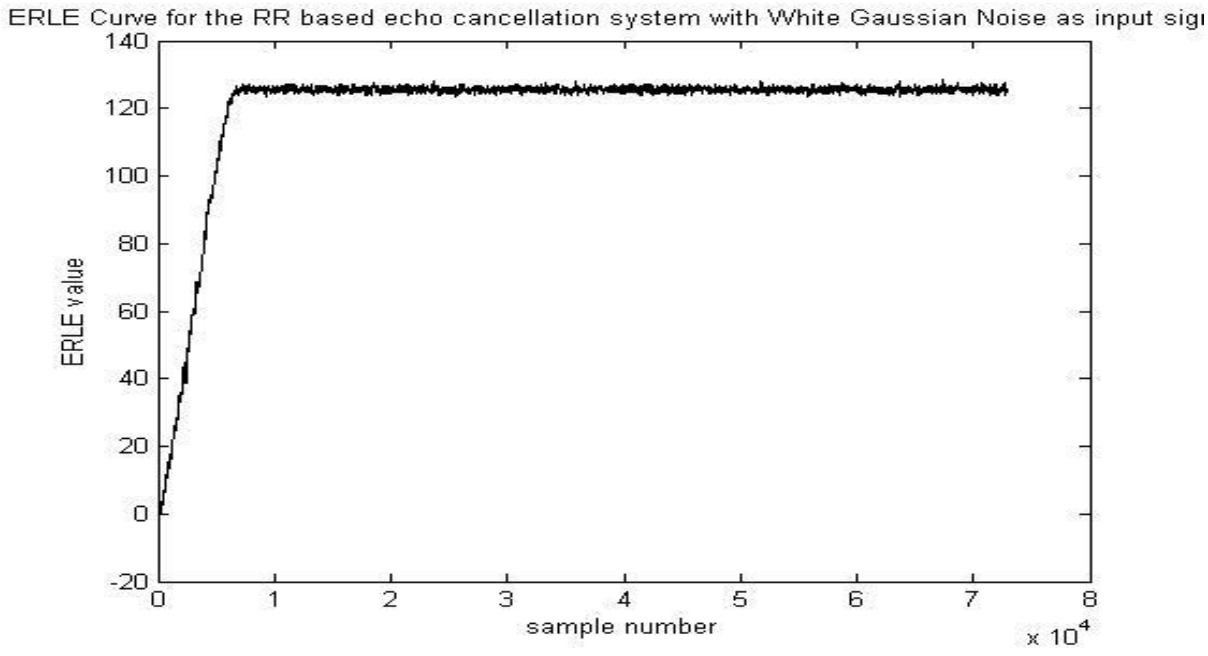


Figure 6.11 ERLE curve for RR based AEC with white Gaussian noise samples as the far-end input

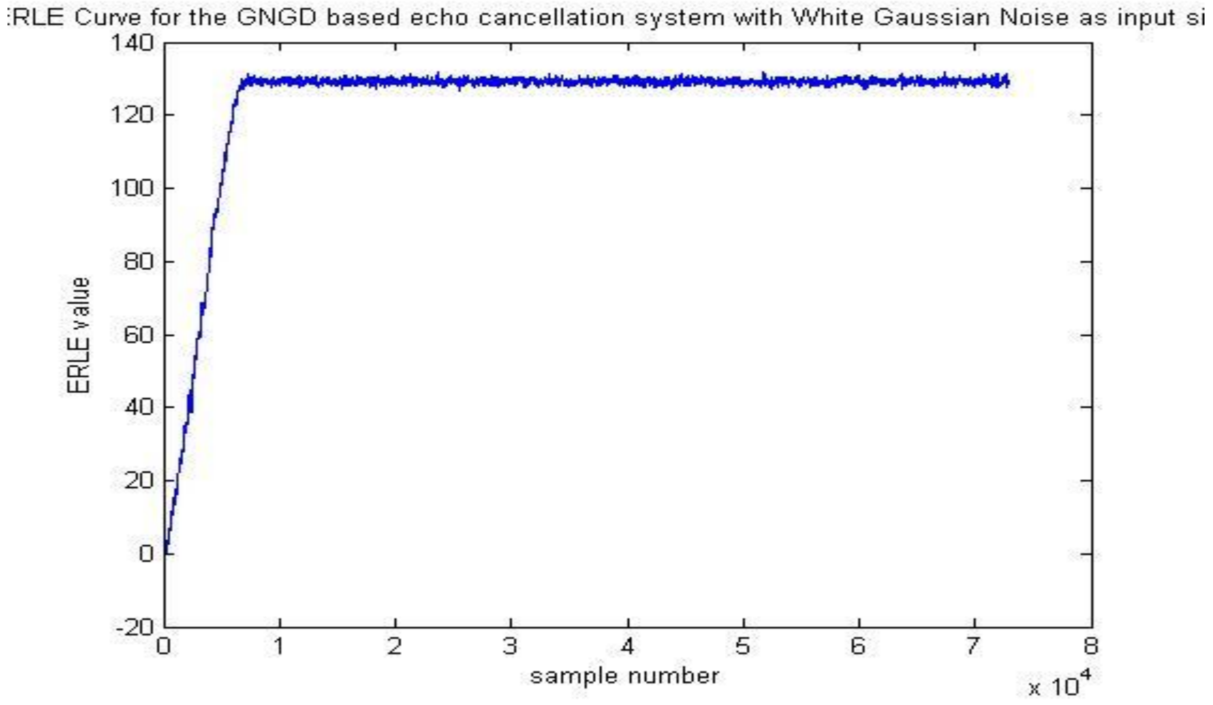


Figure 6.12 ERLE curve for GNGD based AEC with White Gaussian Noise samples as the far-end input

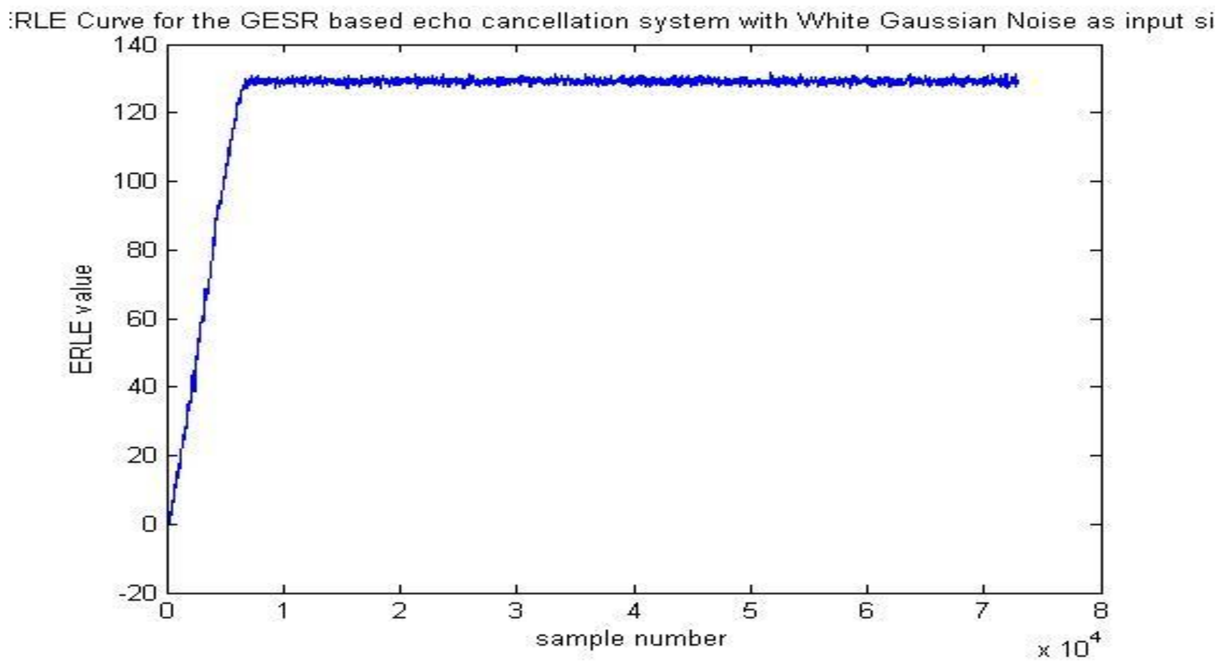


Figure 6.13 ERLE curve for GESR based AEC with White Gaussian Noise samples as the far-end input

6.3 Normalized squared coefficient error (NSCE)

The normalized squared coefficient error (NSCE) is also used to evaluate the performance of the algorithms. The NSCE is defined as

$$NSCE(n) = 10\log_{10}\frac{\left\|\mathbf{h}_o(n)-\mathbf{w}(n)\right\|^2}{\left\|\mathbf{h}_o(n)\right\|^2}$$

The adaptive filter $\boldsymbol{w}$ is used to identify a 256-tap acoustic echo system $\mathbf{h}_o(n)$.

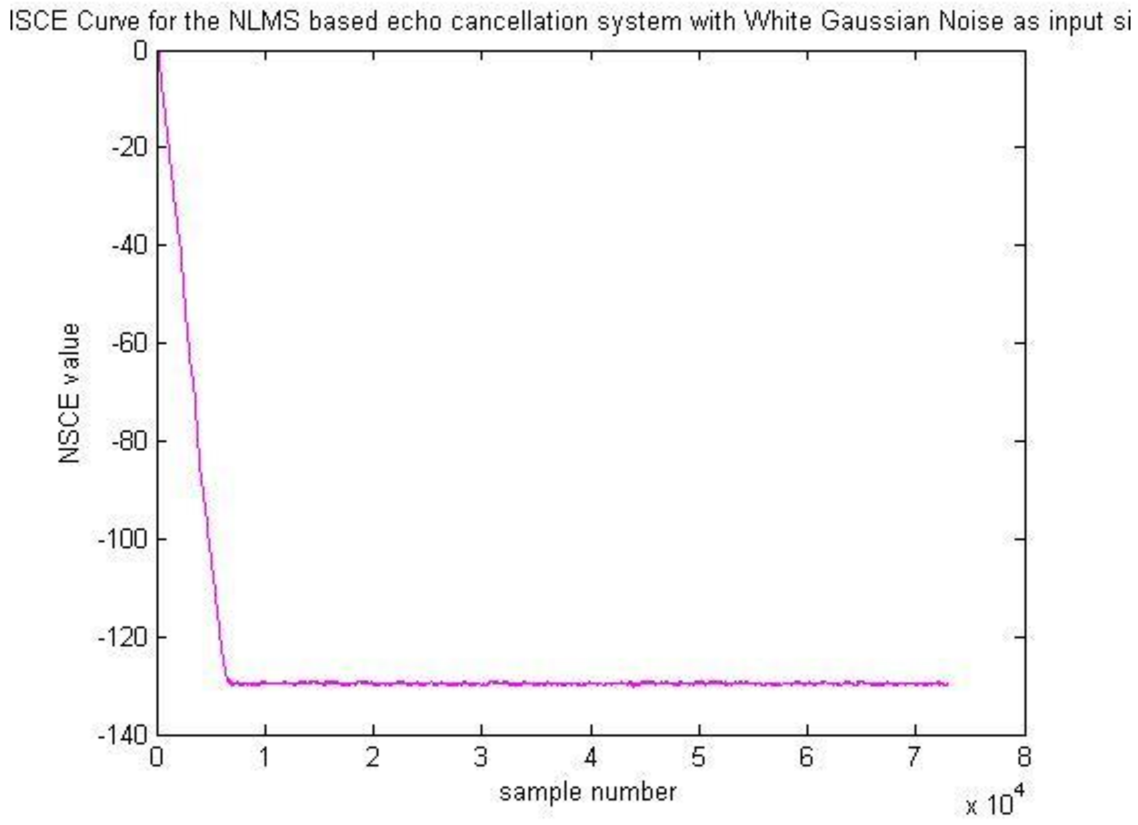


Figure 6.14 NSCE curve for NLMS based AEC with White Gaussian Noise samples as the far-end input

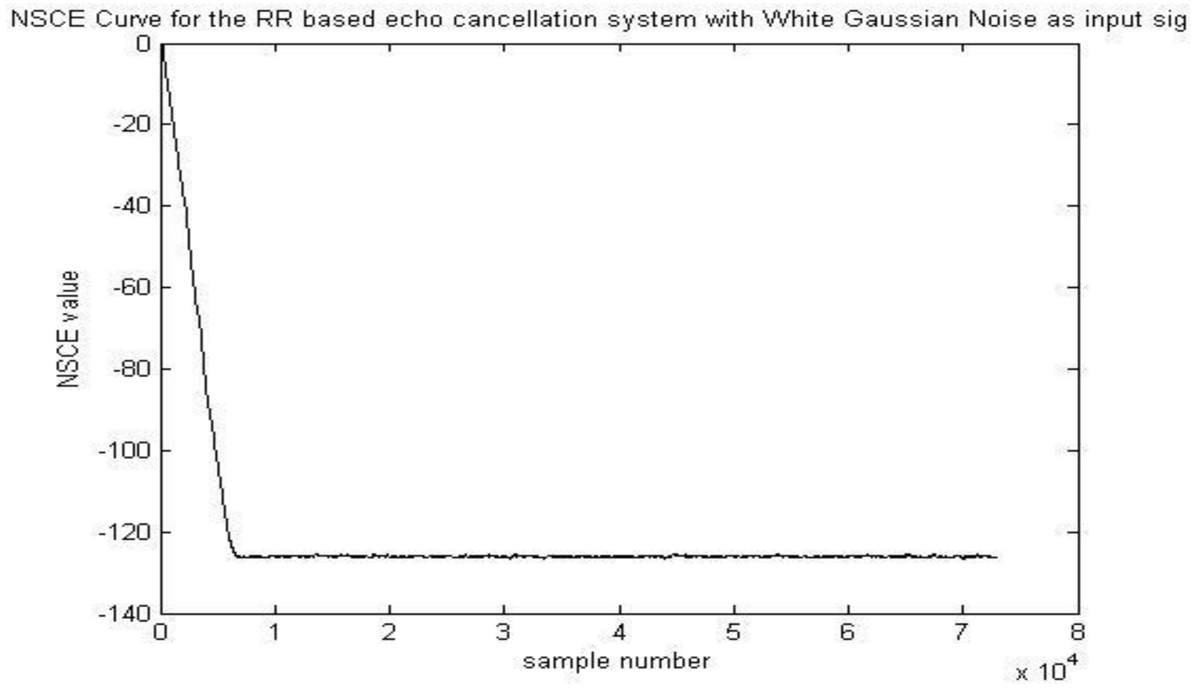


Figure 6.15 NSCE curve for RR based AEC with White Gaussian Noise samples as the far-end input

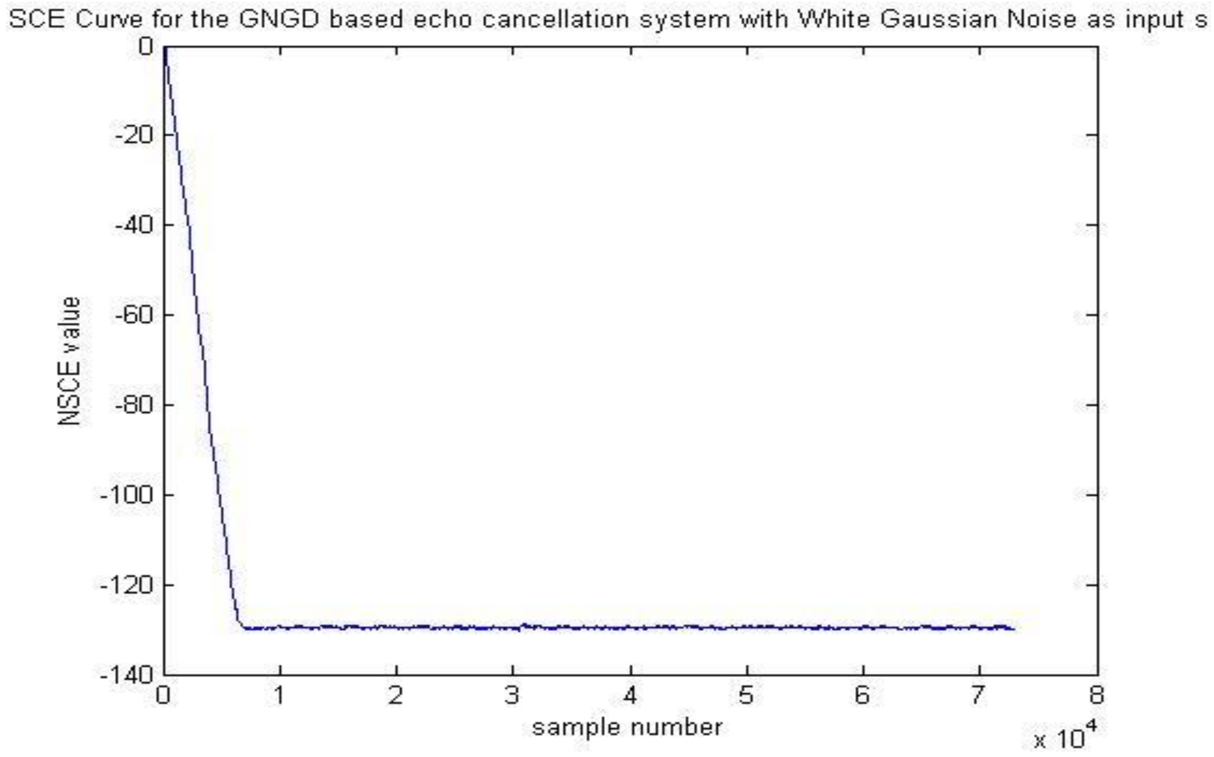


Figure 6.16: NSCE curve for GNGD based AEC with White Gaussian Noise samples as the far-end input

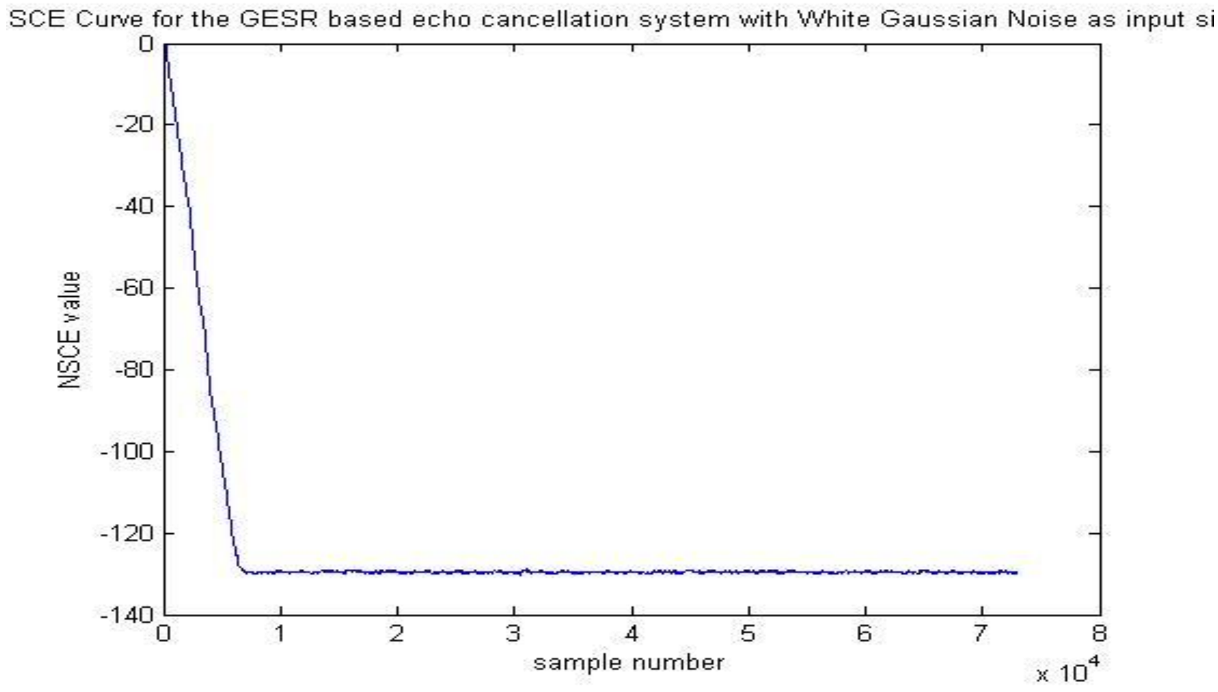


Figure 6.17NSCE curve for GESR based AEC with White Gaussian Noise samples as the far-end input

AEC Method	ERLE (for white Gaussian signal as the input) in dB	Convergence Time in msec
NLMS	129	32.35
VS-NLMS	168	68
RR	128	29
GNGD	132	30
GESR	130	28
GSPAP	160	22

Table 6.1 Provides the maximum ERLE and the corresponding convergence time for the White Gaussian process samples as the far-end input signal.

CHAPTER 7

Conclusion and Future work

7.1 Conclusion

The real-time implementation of Acoustic Echo Canceller Systems based on NLMS and Variable Step Sized NLMS algorithms such as VS-NLMS, RR, GNGD and GESR are implemented using TMS320C6713DSK. Among these algorithms the general VS-NLMS has high ERLE value but poor convergence time. The GESR method based VS-NLMS provides a reasonable ERLE and a minimum convergence time as given in the table 6.1.

7.2 Future work

As a proof of concept, NLMS, Generalized Square Error Regularized (GSER) NLMS Algorithm, RR and GNGD algorithms based acoustic echo cancellers (AEC) are implemented and the performances are evaluated in this project using Floating Point DSP processor based hardware board. The future scope of this project is to implement these algorithms on high performing Fixed-Point DSP processors.

Acoustic Echo Cancellers based on sub-band filtering approach is considered to be most coveted research topics nowadays. The simulation and consequently the hardware implementation of these frequency domain based algorithms can be considered as a future scope of this project.

BIBLIOGRAPHY

1. Srinivasaprasath Raghavendran “Implementation of an Acoustic Echo Canceller Using Matlab”
2. Lu Lu “Implementation of Acoustic Echo Cancellation for PC Applications”
3. ITU-T’s G.167 “Acoustic Echo Cancellers”
4. Simon Haykin, “Adaptive filter theory,” 4th Edition
5. Junghsi Lee, Jia-Wei Chen, and Hsu-Chang Huang “Performance Comparison of Variable Step-Size NLMS Algorithms”
6. Y. S. Choi, H. C. Shin, and W. J. Song, “Robust regularization for normalized LMS algorithms,” IEEE Transactions on Circuits and Systems II, Express Briefs, Vol. 53, No. 8, pp. 627–631, Aug. 2006.
7. J. Lee, H. C. Huang, and Y. N. Yang, “The generalized square-error-regularized LMS algorithm,” Proceedings of WCECS 2008, pp. 157 – 160, Oct. 2008.
8. D. P. Mandic, “A generalized normalized gradient descent algorithm,” IEEE Signal Processing Letters, Vol. 11, No. 2, pp. 115–118, Feb. 2004.
9. M.Medvecky, S.Herrera-Garcia “Implementation of modified NLMS algorithm for acoustic echo cancellation” Mar 2003.
10. TI’s Code Composer Studio (CCS) User’s Guide.
11. H. Ding, “A stable fast affine projection adaptation algorithm suitable for low-cost processors,” in *Proc. IEEE ICASSP*, vol. 1, pp. 360-363, June 2000.
12. Jean-Marc Valin and Iain B. Collings, “Interference-Normalized Least Mean Square Algorithm” IEEE Signal Processing Letters, . VOL. 14, NO. 12, DECEMBER 2007
13. Ahmed I. Sulyman & Azzedine Zerguine “Echo Cancellation Using a Variable Step-Size NLMS Algorithm”

14. F. Albu**,** A. Fagan, " The Gauss-Seidel pseudo affine projection algorithm and its application for echo cancellation " Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on Volume 2, 9-12 Nov. 2003 Page(s): 1303 - 1306 Vol.2, California, U.S.A.

15. K. Ozeki, T. Umeda, 'An adaptive Filtering Algorithm Using an Orthogonal Projection to an Affine Subspace and its Properties,' Electronics and Communications in Japan, Vol. 67-A, No.5, 1984

16. S.L. Gay, and S. Tavathia, "The fast affine projection algorithm", Proceedings of ICASSP 1995, Detroit (MI), May 1995, vol. 5, pp. 3023-302617.

17. F. Albu, J. Kadlec, N. Coleman, A. Fagan, "The Gauss-Seidel Fast Affine Projection Algorithm", IEEE Workshop on Signal Processing Systems (SIPS'02), pp. 109-114, San Diego, U.S.A, October 2002

18. F. Bouteille, P. Scalart, M. Corazza, "Pseudo Affine Projection Algorithm New Solution for Adaptive Identification", Eurospeech, 1999

Appendix A

The Audio Interface Onboard the TMS320C6713 DSK

The TMS320C6713 DSK uses a Texas Instruments AIC23 codec. In the default configuration, the codec is connected to the two serial ports, McBSP0 and McBSP1. McBSP0 is used as a unidirectional channel to control the codec's internal configuration registers.

It should be programmed to send a 16-bit control word to the AIC23 in SPI format. The top 7 bits of the control word specify the register to be modified and the lower 9 bits contain the register value.

Once the codec is configured, the control channel is normally idle while audio data is being transmitted. McBSP1 is used as the bi-directional data channel for ADC input and DAC output samples. The codec supports a variety of sample formats. For the experiments in this course, the codec should be configured to use 16-bit samples in two's complement signed format.

The codec should be set to operate in master mode so it supplies the frame sync and bit clocks at the correct sample rate to McBSP1. The codec has a 12 MHz system clock which is the same as the frequency used in many USB systems. The AIC23 can divide down the 12 MHz clock frequency to provide sampling rates of 8000, 16000, 24000, 32000, 44100, 48000, and 96000 Hz. McBSP0 and McBSP1 can be individually disconnected through software from the AIC23 codec and routed to the Peripheral Expansion Connector.

This allows commercially available and individually designed daughter cards to be plugged into the expansion sockets on the DSK.

There are daughter cards for a significant number of codecs that are more capable than the AIC23. A special Board Support Library (BSL) is supplied with the TMS320C6713 DSK. The BSL provides C-language functions for configuring and controlling all the on-board devices. The library includes modules for general board initialization, access to the AIC23 codec, reading the DIP switches, controlling the LED's, and programming and erasing the Flash memory.

The source code for this library is also included. The version of Code Composer supplied with the DSK is set up to automatically use the BSL. The function for configuring the codec in the BSL sets McBSP1 to transmit and receive 16-bit words. The codec sends 16-bit left and right channel input samples to McBSP1 alternately and a program reading these samples from McBSP1's Data Receive Register (DRR1) would have to be programmed to figure out which is the right and which is the left channel sample.

The first word transmitted by the AIC23 codec is the left channel 16-bit sample and the right channel 16-bit sample is transmitted immediately after the left channel sample. The AIC23 generates single frame sync at the beginning of the left channel sample. Therefore, a 32-bit word received by McBSP1 contains the left sample in the upper 16 bits and the right sample in the lower 16 bits.

This solves the channel ambiguity problem. The reverse process takes place when sending samples from the DSP to the codec. The user's program should pack the left channel 16-bit sample in the upper 16 bits of an integer and the right channel 16-bit sample in the lower 16 bits and then write this word to the Data Transmit Register (DXR1) of McBSP1.

Appendix B

LEAST MEAN SQUARE (LMS) ALGORITHM

The Least Mean Square (LMS) algorithm was first developed by Widrow and Hoff in 1959 through their studies of pattern recognition. From there it has become one of the most widely used algorithms in adaptive filtering. The LMS algorithm is a type of adaptive filter known as stochastic gradient-based algorithms as it utilizes the gradient vector of the filter tap weights to converge on the optimal wiener solution.

It is well known and widely used due to its computational simplicity. It is this simplicity that has made it the benchmark against which all other adaptive filtering algorithms are judged.

With each iteration of the LMS algorithm, the filter tap weights of the adaptive filter are updated according to the following formula.

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

Here **x**(n) is the input vector of time delayed input values, **x**(n) = [x(n) x(n-1) x(n-2) .. x(n-N+1)]T. The vector **w**(n) = [w0(n) w1(n) w2(n) .. wN-1(n)] T represents the coefficients of the adaptive FIR filter tap weight vector at time n. The parameter μ is known as the step size parameter and is a small positive constant. This step size parameter controls the influence of the updating factor. Selection of a suitable value for μ is imperative to the performance of the LMS algorithm, if the value is too small the time the adaptive filter takes to converge on the optimal solution will be too long; if μ is too large the adaptive filter becomes unstable and its output diverges.

Derivation of the LMS algorithm.

The derivation of the LMS algorithm builds upon the theory of the wiener solution for the optimal filter tap weights, **w**o.. It also depends on the steepest descent algorithm, this is a formula which updates the filter coefficients using the current tap weight vector and the cur-rrent gradient of the cost function with respect to the filter tap weight coefficient vector, $\nabla\xi(n)$.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla\xi(n)$$
$$\text{where } \xi(n) = E\left[e^2(n)\right]$$

As the negative gradient vector points in the direction of steepest descent for the N-dimensional quadratic cost function, each recursion shifts the value of the filter coefficients closer toward their optimum value, which corresponds to the minimum achievable value of the cost function, ξ(n).

The LMS algorithm is a random process implementation of the steepest descent algo-rithm. Here the expectation for the error signal is not known so the instantaneous value is used as an estimate. The steepest descent algorithm then becomes the following equation.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla\xi(n)$$
$$\text{where } \xi(n) = e^2(n)$$

The gradient of the cost function, ∇ξ(n), can alternatively be expressed in the following form.

$$
\begin{aligned}
\nabla\xi(n) &= \nabla(e^2(n)) \\
&= \frac{\partial e^2(n)}{\partial \mathbf{w}} \\
&= 2e(n)\frac{\partial e(n)}{\partial \mathbf{w}} \\
&= 2e(n)\frac{\partial (d(n)-y(n))}{\partial \mathbf{w}} \\
&= -2e(n)\frac{\partial e\mathbf{w}^T(n)\mathbf{x}(n)}{\partial \mathbf{w}} \\
&= -2e(n)\mathbf{x}(n)
\end{aligned}
$$

Substituting this into the steepest descent algorithm, we arrive at the recursion for the LMS adaptive algorithm.

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