Fall 2007

Performance improvement of adaptive filters for echo cancellation applications

Deepak Challa

Department:

PERFORMANCE IMPROVEMENT OF ADAPTIVE FILTERS FOR ECHO CANCELLATION APPLICATIONS

by

DEEPAK KUMAR CHALLA

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

2007

Approved by

_____	_____
Dr. Steven L. Grant, Advisor	Dr. Rosa Y. Zheng

_____

Dr. Norman R. Cox

ABSTRACT

This work focuses on performance improvement of adaptive algorithms for both line and acoustic echo cancellation applications. Echo in telephone networks, Line Echo, is observed naturally due to impedance mismatches at the long-distance/local-loop interface. Acoustic echo is due to the acoustic coupling between the microphone and the speaker of a speakerphone. The Affine Projection (APA) and the Fast Affine Projection (FAP) algorithms are two examples of reliable and efficient adaptive filters used for echo cancellation. The FAP Algorithm is considerably less complex than the APA while demonstrating similar convergence properties. Both algorithms use a fixed regularization parameter to mitigate undue noise amplification for small eigenvalues in the excitation covariance matrix. However, this regularization also introduces a bias and limits the convergence property of the adaptive filter. This thesis presents, Variable Regularized Fast Affine Projections (VR-FAP) algorithm, with a varying, optimal regularization value which provides the desirable property of both fast and low misadjustment of the filter.

In a separate section of the thesis, an improved technique for sub-band filter design is discussed. Sub-band implementation of adaptive filters for echo cancellation is popular due to its properties of reduced complexity and fast convergence when compared to the full band processing. However, sub-band structure introduces aliasing and results in degraded performance of the adaptive filter. Therefore, design of the analysis and synthesis filter involves careful tuning of several design parameters. Manual adjustment of these design parameters is a tedious job and could take many days. This thesis introduces an offline technique which uses a Hybrid Particle Swarm Optimizer with a Neural Network to find the optimal parameters effortlessly without manual tuning.

ACKNOWLEDGMENTS

I would like start with thanking Dr. Steven L. Grant for his unparalleled support and encouragement towards my Masters Degree Program at UMR. His courses laid the foundation of my knowledge and have been greatest motivation factors towards my research. I am grateful to him for giving me an opportunity to work with him. Research meetings with him are always a good learning experience and exciting. He has always given me time for discussions and from which I have learned the approach to analyze, understand and solve complicated problems. I am thankful to him for his excellent mentorship in terms of research guidance, academic advising and moral support. I have learned a great deal and I hope to continue working with Dr. Grant and contribute towards the ever growing signal processing world.

I would like to thank my committee members Dr. Rosa Zheng and Dr. Norman Cox for their timely support and invaluable help. I would also like to thank them for reviewing my thesis and offering important feedback and suggestions. I would like to thank Dr. G. K. Venayagamoorthy for his time, support and advice towards my research topics.

I would like to thank my research team members Dr. Asif Mohammad, Mr. Rommel Crasta and the entire research group for sharing their knowledge and for their continuous support towards my research work. I would also like to thank all my friends and especially my roommates at Rolla for their unflinching moral support during my stay here.

I would like to express special thanks to my parents Mr. Ramesh Challa and Mrs. Mohini Challa for their blessings and good wishes, and to my brothers Mr. Dilip Challa and Mr. Dinesh Challa for their undying love and support all the way through.

I would like to thank everyone who helped me accomplish my tasks and achieve my goals towards completing my graduate degree program.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

For decades people have been using the telephone as a means of distant voice communication. As the coverage area and number of subscribers increased, the telephone system has become more and more sophisticated. Long distance connections and the sophisticated systems introduced a lot of challenging problems. One of the major challenges in the telephone system is the "Echo Effect". Echo is an undesired phenomenon experienced by a user over a phone call when he/she hears his/her own voice back after a delay [1]. Echo generation can be characterized as Network/Line Echo and Acoustic Echo [2]. The following sub-sections briefly explain the generation of echo and some promising solutions for the same.

1.1. LINE ECHO

Consider a Public Switched Telephone Network (PSTN) as described in the Figure 1.1.

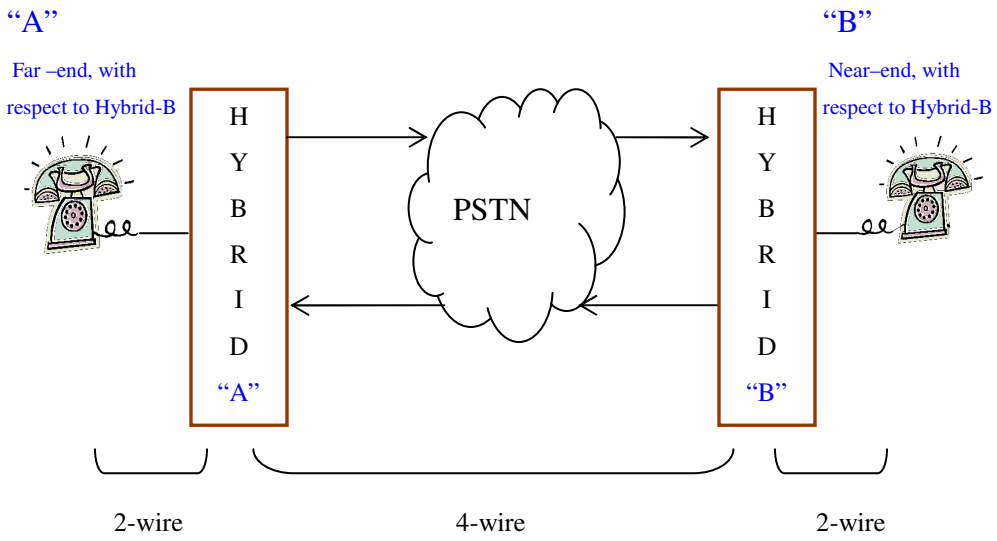

Figure 1.1. A Simplified Telephone Circuit.

User A and user B are connected via 2-wire circuits (local subscriber loop) and 4-wire circuits which used for long distance connections. The 2-wire circuits and 4-wire circuits are interconnected using a hybrid network, as shown in Figure 1.2 [2].

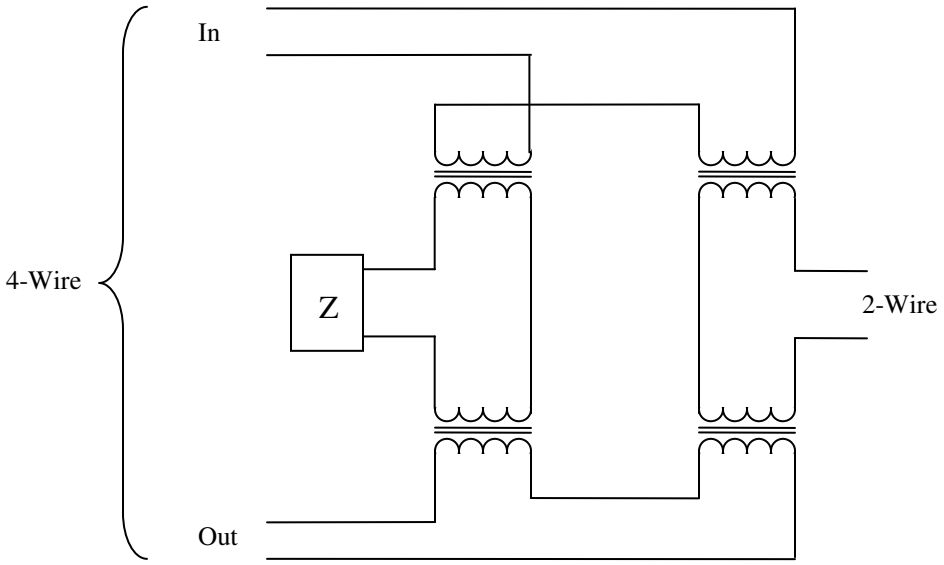

Figure 1.2. Hybrid Network.

When Z is equal to the impedance of the 2-wire circuit, the signal on the In port of the 4-wire side is passed to the 2-wire circuit. The hybrid network consists of passive electrical elements (balanced transformers, a capacitor, and a resistor) and can be connected to different 2-wire circuits of varying length and impedance. An imperfect impedance (Z) match at the hybrid B results in the electrical signal transmitted from A to be sent back to A (or vice versa) and hence producing the network echo. As a result, speaker A hears his own echo after a delay and it can be noticed by the subscriber which is annoying, for example when the round trip delay is greater than 100*ms* and echo signal energy with reference to the excitation signal energy is greater than -30*dB* [2].

Line echo impulse response is typically sparse and the energy of the impulse response is concentrated mostly around a few coefficients of the response. Figure 1.3

illustrates the simulated impulse response at 8000Hz sampling frequency [2]. The length of the impulse response depends on the delay introduced by the telephone network.

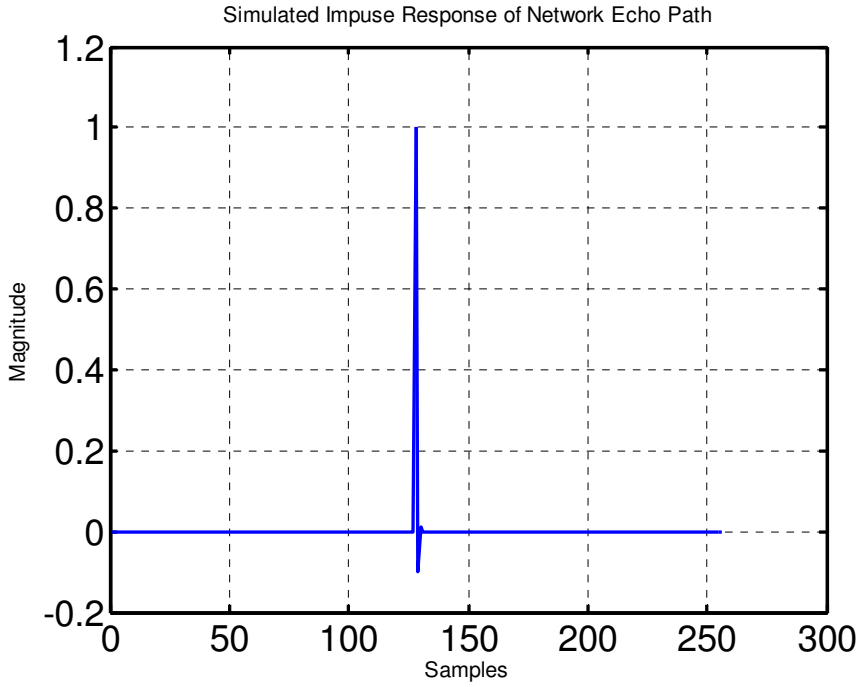

Figure 1.3. Simulated Impulse Response of Line Echo Path-Typical.

1.2. ACOUSTIC ECHO

Acoustic echo is significant for hands-free and teleconferencing applications. Echo generated due to the acoustic path (acoustic coupling) between the loudspeaker and microphone of a communication device is known as acoustic echo. Figure 1.4 illustrates the acoustic echo path [2].

Impulse response of acoustic echo path is dispersive (not sparse) and is characterized by the acoustic environment variables like the size of the room, materials inside the room and the reflection coefficients of the materials. Figure 1.5 shows an Image Model Derived Impulse Response of acoustic echo path in a room at 8000Hz sampling frequency. Acoustic echo path response is tougher to estimate because the echo path is long (typically 150 ms or more). Also the echo path varies rapidly as the room is

not static over time [2]. It varies with temperature, pressure, humidity, movement of the objects and the microphone in the room.

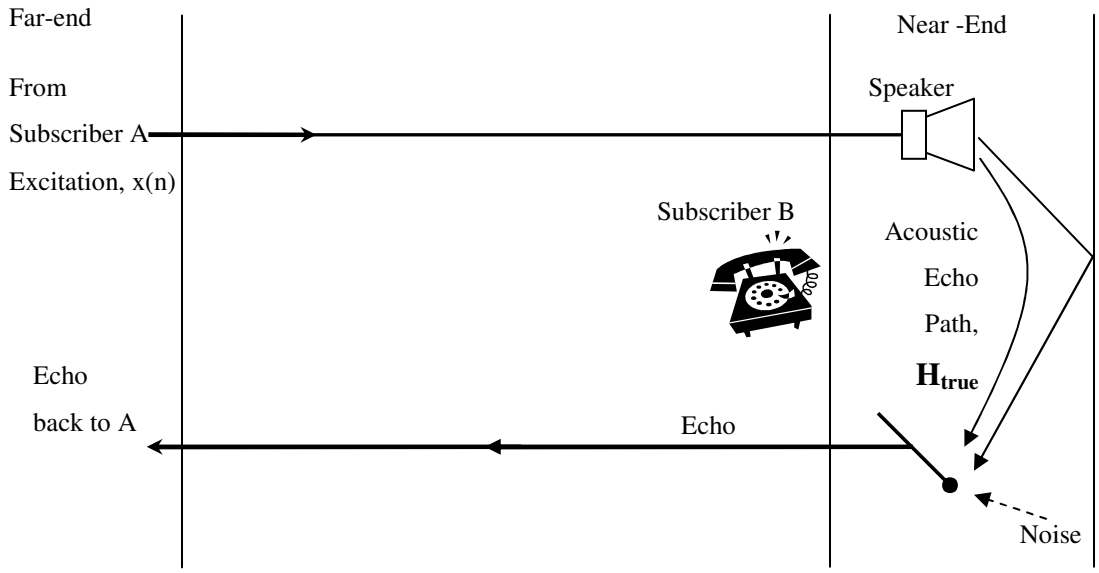

Figure 1.4. Acoustic Echo Path.

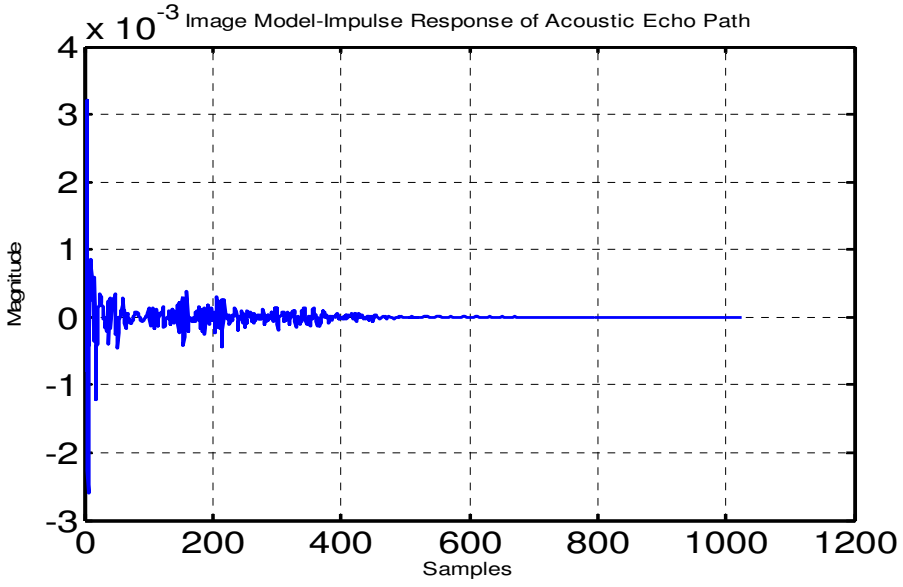

Figure 1.5. Simulated Acoustic Echo Impulse Response – Example.

1.3. ECHO CANCELLER

Telephone companies use a device called an echo canceller (EC) to overcome the echo problem [1]. An echo canceller is a simple adaptive filter with self adjusting coefficients to cancel out the echo. Every echo has an echo path and it is characterized by an impulse response. The echo canceller adapts its filter coefficients to the network echo path such that it cancels out the echo. This process can be visualized with an echo canceller unit in Figure 1.6 and Figure 1.7.

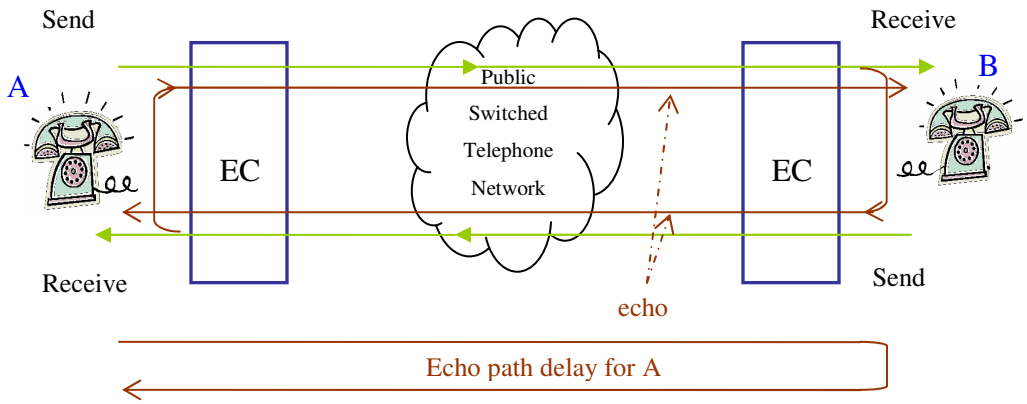

Figure 1.6. Telephone System with Echo Canceller (EC).

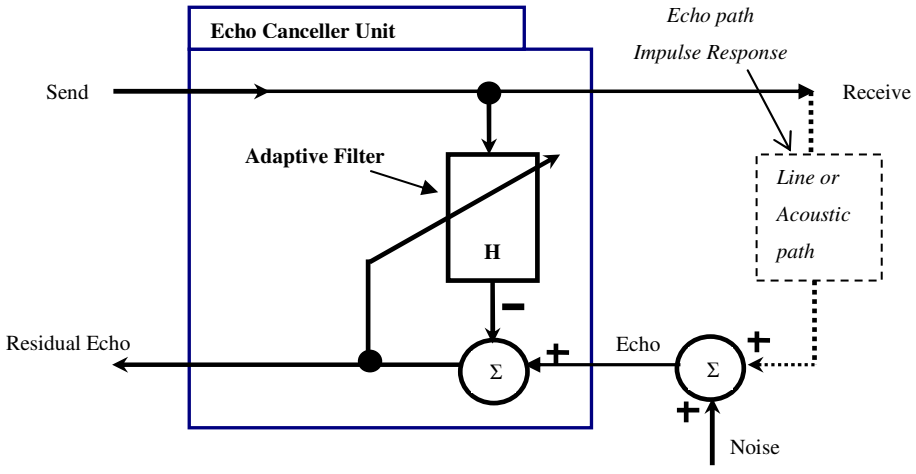

Figure 1.7. Echo Canceller Unit.

1.4. VARIABLE REGULARIZED FAST AFFINE PROJECTIONS

Section 3 introduces a variable regularization method for the fast affine projection algorithm (VR-FAP). It is inspired by a recently introduced technique for variable regularization of the classical affine projection algorithm (VR-APA) [17]. In both the algorithms, the regularization parameter varies as a function of the excitation, measurement noise, and residual error energies. Because of the dependence on the last parameter, VR-APA and VR-FAP demonstrate the desirable property of fast convergence (via a small regularization value) when the convergence is poor and deep convergence (low misadjustment) and immunity to measurement noise (via a large regularization value) when the convergence is good. While the regularization parameter of APA is explicitly available for on-line modification, FAP's regularization is only set at initialization. To overcome this problem, noise-injection with the noise-power proportional to the variable regularization parameter is used. As with their fixed regularization versions, VR-FAP is considerably less complex than VR-APA and simulations verify that they have very similar convergence properties.

1.5. OPTIMIZATION OF SUB-BAND ANALYSIS AND SYNTHESIS FILTERS USING CI TECHNIQUES

In Section 5, an improved technique for sub-band filter design for sub-band adaptive filters is discussed. Sub-band adaptive filters are built on multirate digital filters, which require analysis and synthesis filter banks [3,4]. The analysis filter banks separate a full band signal into sub-band signals. The adaptive filter adapts the filter coefficients in each sub-band. The synthesis filter bank combines the sub-band signals into full-band signals. For the perfect reconstruction of the full band signal, an ideal filter with zero phase has to be used which is impractical. Hence, a Finite Impulse Response (FIR) filter is used for analysis and synthesis filter bank implementation. However, the analysis and synthesis filters should be designed to have a sharp cut-off with low pass band ripple and minimal ripple in the stop band to avoid aliasing effects and enable good reconstruction of the full band signal. The design of these filters involves choosing parameters like pass-band edge, weighting of the synthesis filter versus analysis filter, delay, and the length of

analysis and synthesis filters [4]. These parameters have to be tuned for every combination of the sub-band number and sub-sampling rate. This process is cumbersome, time consuming and tedious. In addition, manual tuning of these parameters does not guarantee optimality and can adversely affect the performance of the sub-band adaptive filter. Therefore, a computational intelligence technique comprised of a neural network (NN) with particle swarm optimization (DEPSO) [5,6,7] with a multi-objective cost function and the offline learning method was used to obtain the parameters for the best performance of the sub-band adaptive filter.

2. ADAPTIVE FILTERS FOR ECHO CANCELLATION

2.1. LEAST MEAN SQUARE ALGORITHMS

In this section we discuss two types of adaptive filters. The first is a stochastic gradient algorithm called least mean squares (LMS) and the second is a derivative of LMS, normalized least mean squares (NLMS).

2.1.1. The LMS Algorithm. LMS is the most widely used adaptive filtering algorithm in the world. It is used in various applications like system identification problems (e.g. echo cancellation), speech coding and channel equalization problems. Although, its speed of convergence is often slower than desired, it is popular because of its robust performance, low cost of implementation and simplicity [8].

LMS is a stochastic gradient algorithm. The derivation of the algorithm is similar to the steepest descent method. However, the steepest descent uses a deterministic gradient to reach the Wiener solution, whereas LMS uses a stochastic gradient for recursive updates which tends to achieve the Wiener solution [8].

LMS's application to echo cancellation can be visualized from figure 2.1. Let us make the following definitions,

$x(n)$ is the *excitation* signal, often called the far-end signal.

$\mathbf{x}(n) = \left[x(n), x(n-1), \ldots, x(n-L+1) \right]^{\mathrm{T}}$ is the *excitation* vector.

$s(n) = \mathbf{x}^{T}(n)\mathbf{h}_{true} + y(n)$ is the *desired* signal, it is the summation of the echo $\mathbf{x}^{T}(n)\mathbf{h}_{true}$, *near-end* background noise/signal, $y(n)$.

$\mathbf{h}_{true}$ is the true echo path impulse response of the system.

$\mathbf{h}(n) = \left[h_0(n), h_1(n), \ldots, h_{L-1}(n) \right]^{\mathrm{T}}$, where $h_i(n)$ is the i$^{\text{th}}$ *tap weight* of the filter at time *n*.

$e(n) = s(n) - \mathbf{x}^{T}(n)\mathbf{h}(n-1)$ is the *a priori error* or *residual echo*.

The performance index (cost function) of LMS is the mean squared error,

$$J(\mathbf{h}(n)) = E\{e^{2}(n)\} = E\left\{ \left[s(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1) \right]^{2} \right\} \quad (1)$$

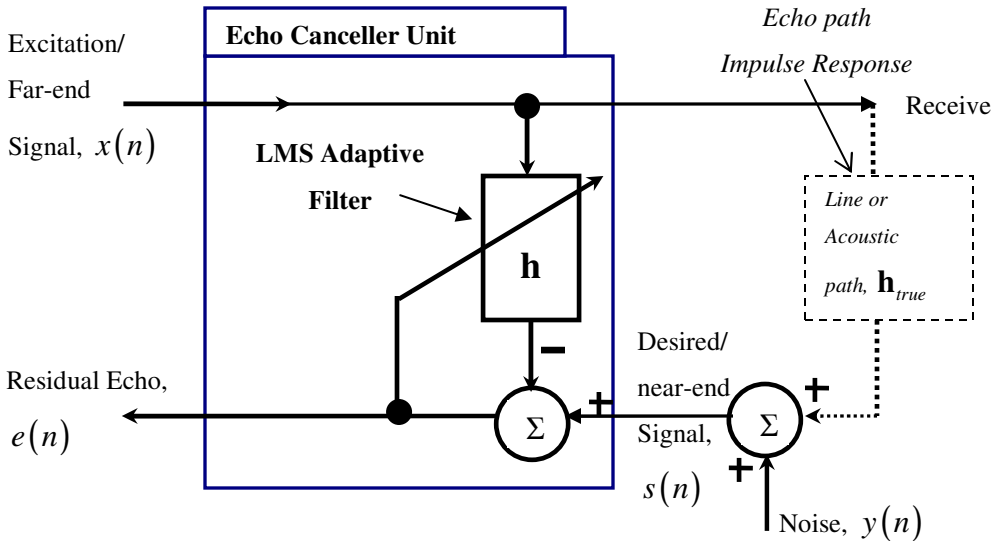

Figure 2.1. Illustration of an Echo Canceller with LMS Algorithm.

Let $J(\mathbf{h}(n))$ be denoted as J and $\mathbf{h}(n)$ be denoted as $\mathbf{h}$ for readability. The first order partial derivative of the cost function, J, with respect to weight vector, $\mathbf{h}$, yields the gradient vector,

$$\nabla J(\mathbf{h}) = -2\mathbf{p} + 2\mathbf{R}\mathbf{h}(n-1) \tag{2}$$

where, $\mathbf{p}=E\{[\mathbf{x}(n)s(n)]\}$ is the cross-correlation vector between the desired response $s(n)$ and excitation vector $\mathbf{x}(n)$. $\mathbf{R}=E\{[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\}$ is the correlation matrix of the excitation vector $\mathbf{x}(n)$.

It is computationally expensive to exactly estimate $\mathbf{R}$ and $\mathbf{p}$ in an unknown environment. Hence, the so-called stochastic approximation [8] is used instead. That is, $\mathbf{R}$ is replaced by $\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)$ and $\mathbf{p}$ is replaced by $\mathbf{x}(n)s(n)$. Therefore, $\nabla J(\mathbf{h})$ becomes,

$$\hat{\nabla} J(\mathbf{h}) = -2\mathbf{x}(n)s^{*}(n) + 2\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{h}}(n-1) \tag{3}$$

where, '*' denotes complex conjugate. According to the steepest descent method, successive adjustments applied to the weight vector, $\mathbf{h}$ are in the direction opposite to the direction of the gradient [2,8], $\nabla J(\mathbf{h})$. Hence, the weight vector update equation is,

$$\mathbf{h}(n) = \mathbf{h}(n-1) + 2\mu_{LMS}\mathbf{x}(n)\left[s^*(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1) \right] \tag{4}$$

Let,

$$e^*(n) = \left[s^*(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1) \right] \tag{5}$$

Then the $\mathbf{h}(n)$ update can be written as,

$$\mathbf{h}(n) = \mathbf{h}(n-1) + 2\mu_{LMS}\mathbf{x}(n)e^*(n) \tag{6}$$

where, μ_{LMS} is a constant, also known as the *step-size* or *relaxation* parameter. For the stable operation of the LMS algorithm, μ_{LMS} is restricted to [8],

$$0 < \mu_{LMS} < \left(\frac{1}{LS_{\max}} \right) \tag{7}$$

where, L is the length of the adaptive filter and $S_{\max}$ is the maximum value of the power spectral density of the excitation vector, $\mathbf{x}(n)$. A high value of μ_{LMS} yields faster, but noisier convergence of the adaptive filter while a low value of μ_{LMS} provides slower but less noisy or *deeper* convergence (or closer to wiener solution) [8].

Considering real valued excitation and desired signals, the LMS algorithm can be summarized as,

$$e(n) = \left[s(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1) \right] \tag{8}$$

$$\mathbf{h}(n) = \mathbf{h}(n-1) + 2\mu_{LMS}\mathbf{x}(n)e(n) \tag{9}$$

The dependence of the stochastic gradient vector of equation (3) on the instantaneous value of the input instead of the ensemble average introduces an error to the gradient,

$\nabla J(\mathbf{h})$. This error is known as gradient noise. Hence, LMS is classified as a stochastic gradient descent algorithm.

The computational complexity of LMS for one sample period is summarized in the Table 2.1. The total memory required for LMS implementation is about 2*L*. Where, *L* is the length of the adaptive filter.

Table 2.1. Complexity of LMS Algorithm.

Equation	Multiplications
$e(n)=\left[s(n)-\mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1)\right]$	L
$\mathbf{h}(n)=\mathbf{h}(n-1)+2\mu_{LMS}\mathbf{x}(n)e(n)$	L
Total Complexity	$2L$

2.1.2. The NLMS Algorithm. As indicated by Equation (8) LMS's step size is restricted by its region of stability which is determined by the energy in the excitation signal. For signals that have time-varying short-time energy, like speech, a constant step-size means the speed of convergence will vary with the short-time energy. NLMS overcomes this problem by normalizing the step size every update with the squared Euclidian norm of the excitation vector, $\mathbf{x}(n)$. NLMS can be derived by considering a sample-by-sample cost function that minimizes the size of the coefficient update under the constraint that the *a posteriori error* (the error after the coefficient update) for that sample period is zero. Thus,

$$J_n=\delta\mathbf{r}^{\mathrm{T}}(n)\mathbf{r}(n)+\hat{e}^2(n) \tag{10}$$

where, $\mathbf{r}(n)$ is the coefficient update vector at sample period n, $\hat{e}(n)$ is the a posteriori error, and δ is a weighting factor between the size of the update and the a posteriori error.

The a posteriori error can be expressed as,

$$\begin{aligned}\hat{e}(n) &= s(n) - \mathbf{x}^T(n)\mathbf{h}(n) \\ &= s(n) - \mathbf{x}^T(n)\left[\mathbf{h}(n-1) + \mathbf{r}(n)\right] \\ &= e(n) - \mathbf{x}^T(n)\mathbf{r}(n)\end{aligned} \tag{11}$$

Thus, the cost function can be written as,

$$J_n = \delta \mathbf{r}^{\mathrm{T}}(n)\mathbf{r}(n) + \left[e(n) - \mathbf{x}^T(n)\mathbf{r}(n)\right]^2 \tag{12}$$

Setting the gradient of J_n with respect to $\mathbf{r}(n)$ and setting the result to zero, we find this cost function is minimized when,

$$\mathbf{r}(n) = \left[\mathbf{x}^T(n)\mathbf{x}(n) + \delta\right]^{-1}\mathbf{x}(n)e(n) \tag{13}$$

NLMS algorithm can be written in the two steps of its usual implementation form as,

$$e(n) = \left[s(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1)\right] \tag{14}$$

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mu_{NLMS}\left[\mathbf{x}^T(n)\mathbf{x}(n) + \delta\right]^{-1}\mathbf{x}(n)e(n) \tag{15}$$

where, the NLMS step-size parameter, μ_{NLMS} has been added as a *relaxation* factor and the stability range of μ_{NLMS} for NLMS is $0 < \mu_{NLMS} < 1$. The parameter δ in the NLMS coefficient update is also known as the regularization parameter. It is seen that when δ is non-zero (it is always non-negative) the coefficient update is prevented from becoming unstable when $\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) = 0$.

Placing equation (14) into (15) we may write the update as,

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \frac{\mu_{NLMS}}{\left\|\mathbf{x}(n)\right\|^2 + \delta}\mathbf{x}(n)\left[s(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1)\right] \tag{16}$$

Recalling that the desired signal is,

$$s(n) = \mathbf{x}^T(n)\mathbf{h}_{true} + y(n) \tag{17}$$

where, $y(n)$ is the noise signal from the echo path and using this in equation (16) the update is expressed as,

$$\mathbf{h}(n)=\left(\mathbf{I}-\frac{\mu_{NLMS}}{\|\mathbf{x}(n)\|^2+\delta}\mathbf{x}(n)\mathbf{x}(n)^T\right)\mathbf{h}(n-1) + \frac{\mu_{NLMS}}{\|\mathbf{x}(n)\|^2+\delta}\mathbf{x}(n)\mathbf{x}(n)^T\mathbf{h}_{true} + \frac{\mu_{NLMS}}{\|\mathbf{x}(n)\|^2+\delta}\mathbf{x}(n)\,y(n) \tag{18}$$

This expression of NLMS is useful for making a few observations. The matrix

$$\mathbf{P}(n)=\frac{1}{\|\mathbf{x}(n)\|^2}\mathbf{x}(n)\mathbf{x}(n)^T \tag{19}$$

occurs in both the first and second terms of equation (18). It is a rank one projection matrix and can be written in the form,

$$\mathbf{P}(n)=\mathbf{U}(n)\begin{bmatrix}1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0\end{bmatrix}\mathbf{U}^T(n) \tag{20}$$

where, $\mathbf{U}(n)$ is an L-dimensional unitary matrix whose first column is $\mathbf{x}(n)/\|\mathbf{x}(n)\|$.

When $\mu_{NLMS}=1$ and $\delta=0$, the first term of equation (18) is the old coefficient vector multiplied by the rank *L*-1, *L* dimensional projection matrix,

$$\mathbf{Q}(n)=\mathbf{I}-\frac{1}{\|\mathbf{x}(n)\|^2}\mathbf{x}(n)\mathbf{x}(n)^T = \mathbf{I}-\mathbf{P}(n)=\mathbf{U}(n)\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{bmatrix}\mathbf{U}^T(n) \tag{21}$$

Thus, one-dimension of $\mathbf{h}(n-1)$ is *forgotten* in each coefficient update. The second term in the equation (18) is the true coefficient vector $\mathbf{h}_{true}$ multiplied by a rank 1, *L*-dimensional projection matrix, $\mathbf{P}(n)$. So, in the second term, one dimension of the actual echo path is *learned* in each coefficient update.

Now, we return to the cases where $\mu_{NLMS} \neq 1$ and $\delta \neq 0$. Equation (18)'s third term is where the effect of noise is manifested. Here, we see the noise, $y(n)$ is multiplied by the vector, $\frac{\mu_{NLMS}}{\|\mathbf{x}(n)\|^2 + \delta}\mathbf{x}(n)$, whose magnitude is,

$$\xi = \frac{\mu_{NLMS}\|\mathbf{x}(n)\|}{\|\mathbf{x}(n)\|^2 + \delta} \tag{22}$$

For sufficiently large *L*, $\|\mathbf{x}(n)\|^2$ may be approximated by $L\sigma_x^2$. Therefore,

$$\xi \approx \frac{\mu_{NLMS} L\sigma_x}{L\sigma_x^2 + \delta} \tag{23}$$

This ξ is known as the noise amplification factor. Both μ_{NLMS} and δ help control the size of the noise amplification factor. When the excitation signal's energy, σ_x^2 is very small, ξ is limited in its growth by the regularization factor δ in its denominator.

A high regularization value results in slower convergence and very low regularization leads to a larger noise amplification factors. Choosing the correct regularization value is important for optimizing the performance of the adaptive filter [8]. In general, compared to LMS, NLMS with regularization is faster and more stable for all kinds of excitation signals (white noise, colored noise and speech).

The computational complexity of the NLMS filter for one sample period is summarized in Table 2.2. The total memory required is around 2L. The total complexity is a little higher than LMS. O_{NLMS} is a constant associated with the computational complexity of the required inverse in equation (15).

Table 2.2. Complexity of NLMS Algorithm.

Equation	Multiplications
$e(n)=\left[s(n)-\mathbf{x}^{\mathrm{T}}(n)\mathbf{h}(n-1)\right]$	L
$\varepsilon(n)=\left[\left\|\mathbf{x}(n)\right\|^2+\delta\right]^{-1}e(n)$	O_{NLMS}
$\mathbf{h}(n)=\mathbf{h}(n-1)+\mu_{NLMS}\mathbf{x}(n)\varepsilon(n)$	L
Total Complexity	$2L+O_{NLMS}$

2.2. AFFINE PROJECTION ALGORITHMS

2.2.1. Affine Projection Algorithm, APA. APA is a generalization of NLMS. Where the coefficient update NLMS can be viewed as a rank-1 affine projection, a rank-N projection with $N \geq 1$ is made in APA. As the projection rank increases, the convergence speed of the adaptive filter increases as well, unfortunately so does the computational complexity [9,10,11].

The N[th]-order affine projection algorithm, in a relaxed and regularized form, can be defined as,

$$\mathbf{e}(n)=\mathbf{s}(n)-\mathbf{X}^{T}(n)\mathbf{h}(n-1) \tag{24}$$

$$\boldsymbol{\varepsilon}(n)=[\mathbf{X}^{T}(n)\mathbf{X}(n)+\delta\mathbf{I}]^{-1}\mathbf{e}(n) \tag{25}$$

$$\mathbf{h}(n)=\mathbf{h}(n-1)+\mu_{APA}\mathbf{X}^{T}(n)\boldsymbol{\varepsilon}(n) \tag{26}$$

The APA variables are defined as follows:

- The excitation signal matrix, $\mathbf{X}(n)$, is L by N and has the structure,

  $$\mathbf{X}(n)=[\mathbf{x}(n),\mathbf{x}(n-1),\cdots,\mathbf{x}(n-N+1)] \tag{27}$$

  where, again $\mathbf{x}(n)=[x(n),\cdots,x(n-L+1)]^{T}$ is the *excitation* vector.

- The adaptive tap weight vector is $\mathbf{h}(n)=[h_0(n),\cdots,h_{L-1}(n)]^{T}$, where $h_i(n)$ is the i^{th} coefficient at sample period n.

- $\mathbf{e}(n)=\left[e_0(n),e_1(n),\cdots,e_{N-1}(n)\right]^T$ is the *N*- length vector consists of background noise and residual echo.
- The *N*-length *desired response* vector, $\mathbf{s}(n)=\left[s(n),s(n-1),\cdots,s(n-N+1)\right]^T$, where, $s(n)=\mathbf{x}^T(n)\mathbf{h}_{true}+y(n)$, is the system output consisting of the response of the echo path impulse response, $\mathbf{h}_{\text{true}}$, to the excitation and the additive system noise, $y(n)$.
- μ_{APA} is the adaptation constant in the range $0\le\mu_{APA}\le 1$ and δ is the regularization parameter.
- *N* defines the rank of affine projections in the solution space and it is called as the order of APA.
- *L* is the length of the adaptive filter.

Similar to NLMS representation, when $\mu_{APA}=1$, $\delta=0$, and there is no noise ($y(n)=0$), the APA coefficient update can be defined as [9,11],

$$\mathbf{h}(n)=\left[\mathbf{I}-\mathbf{P}_{\mathbf{APA}}(n)\right]\mathbf{h}(n-1)+\mathbf{P}_{\mathbf{APA}}(n)\mathbf{h}_{\text{true}} \tag{28}$$

where, $\mathbf{P}_{\mathbf{APA}}(n)$ now has rank *N*.

$$\mathbf{P}_{\text{APA}}(n)=\mathbf{X}(n)\left(\mathbf{X}^{\text{T}}(n)\mathbf{X}(n)\right)^{-1}\mathbf{X}^{\text{T}}(n)=\mathbf{U}\begin{pmatrix}\mathbf{1} & & & & \\ & \ddots & & & \\ & & \mathbf{1} & & \\ & & & \mathbf{0} & \\ & & & & \ddots \\ & & & & & \mathbf{0}\end{pmatrix}\mathbf{U}^{\text{T}} \tag{29}$$

and,

$$\mathbf{I}-\mathbf{P}_{\text{APA}}(n)=\mathbf{U}\begin{pmatrix}\mathbf{0} & & & & \\ & \ddots & & & \\ & & \mathbf{0} & & \\ & & & \mathbf{1} & \\ & & & & \ddots \\ & & & & & \mathbf{1}\end{pmatrix}\mathbf{U}^{\text{T}} \tag{30}$$

has nullity *N*. From equations (28), (29) and (30), it can be seen that for every update, *N* dimensions of $\mathbf{h}(n-1)$ are forgotten and *N* dimensions of $\mathbf{h}_{\text{true}}$ are learned [9,11] as opposed to 1 each for NLMS.

When the projection order of APA is 1, it is equivalent to NLMS. However, the convergence of APA gets better with the increased in the projection order and APA demonstrates very good convergence properties with colored excitation signals.
The computational complexity of APA for one sample period is shown in Table 2.3. O_{APA} is a constant associated with the complexity of the inverse required in equation (25). The total memory required is roughly $2L+ O_{APA}[N^2]$.

Table 2.3. Complexity of APA Algorithm.

Equation	Multiplications	Memory
$\mathbf{e}(n)=\mathbf{s}(n)-\mathbf{X}^T(n)\mathbf{h}(n-1)$	NL	L
$\boldsymbol{\varepsilon}(n)=[\mathbf{X}^T(n)\mathbf{X}(n)+\delta\mathbf{I}]^{-1}\mathbf{e}(n)$	$O_{APA}N^2$	N^2
$\mathbf{h}(n)=\mathbf{h}(n-1)+\mu_{APA}\mathbf{X}^T(n)\boldsymbol{\varepsilon}(n)$	NL	L
Total Complexity	$2NL+O_{APA}[N^2]$	$2L+N^2$

2.2.2. Fast Affine Projection Algorithm, FAP. APA is computationally complex and demands a lot of memory for an implementation. The complexity of APA increases greatly with increase in projection dimensions. To address this problem, FAP was introduced in 1990's. The computational complexity and required memory of FAP are roughly the same as NLMS [2,10]. The problem was solved in three steps.

First, the complexity of $\mathbf{e}(n)$ update was reduced to *L* from *NL*. Considering APA equations (24), (25) and (26) as a reference, $\mathbf{e}(n)$ is defined as [2,10],

$$\mathbf{e}(n)=\mathbf{s}(n)-\mathbf{X}^T(n)\mathbf{h}(n-1)=\begin{bmatrix} e(n) \\ (1-\mu)\bar{\mathbf{e}}(n-1) \end{bmatrix} \tag{31}$$

where, the *N*-1 length vector $\bar{\mathbf{e}}(n-1)$ contains the *N*-1 first elements of $\mathbf{e}(n)$. Therefore, the complexity *L*, is required to calculate $\mathbf{e}(n)$ and complexity *N*-1, for $\bar{\mathbf{e}}(n)$ calculation.

Second, the complexity of the $\boldsymbol{\varepsilon}(n)$ update was reduced to $O_{APA}[N]$ from $O_{APA}[N^2]$. Defining,

$$\mathbf{R}(n) = \mathbf{X}^T(n)\mathbf{X}(n) + \delta\mathbf{I} \tag{32}$$

And using this in equation (25), we can express the *normalized error vector* as,

$$\boldsymbol{\varepsilon}(n) = \mathbf{R}^{-1}(n)\mathbf{e}(n) \tag{33}$$

$\mathbf{R}(n)$ may be updated from $\mathbf{R}(n-1)$ using two rank-1 updates,

$$\mathbf{R}(n) = \mathbf{R}(n-1) + \boldsymbol{\alpha}(n)\boldsymbol{\alpha}^T(n) - \boldsymbol{\alpha}(n-L)\boldsymbol{\alpha}^T(n-L) \tag{34}$$

where, $\boldsymbol{\alpha}(n) = \left[x(n), \cdots, x(n-N+1)\right]^T$ and $\mathbf{R}(0) = \delta\mathbf{I}$. Note that the regularization parameter is set at initialization and then persists indefinitely. Because $\boldsymbol{\alpha}(n)$ (which is also the first row of the excitation matrix) is shift invariant, we may use sliding windowed fast recursive least squares (SW-FRLS) techniques to reduce the computational complexity of equation (33) [2, 13].

Finally, the complexity of $\mathbf{h}(n)$ update is reduced to *L* from *NL*. The $\mathbf{h}(n)$ update is replaced by an $\hat{\mathbf{h}}(n)$ update. Where $\hat{\mathbf{h}}(n)$ only uses the last vector of $\mathbf{X}(n)$ in its update. $\hat{\mathbf{h}}(n)$ is defined as,

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu\mathbf{x}(n-N+1)E_{N-1}(n) \tag{35}$$

where, $E_{N-1}(n) = \left[\varepsilon_{N-1}(n) + \varepsilon_{N-2}(n-1)\cdots + \varepsilon_0(n-(N-1))\right]$. The error vector $\mathbf{e}(n)$ is then calculated using $\hat{\mathbf{h}}(n)$ instead of $\mathbf{h}(n)$.

The FAP algorithm with $\mu = 1$ can summarized as follows [2,10]:

1. Initialize, $E_a(n) = \delta$ and $\mathbf{a}_0 = [1, 0, \cdots, 0]^T$.
2. Update $E_a(n)$ and $\mathbf{a}(n)$ with FRLS [14,11].
3. $\tilde{\mathbf{r}}_{xx}(n) = \tilde{\mathbf{r}}_{xx}(n-1) + x(n)\tilde{\boldsymbol{\alpha}}(n) + x(n-L)\tilde{\boldsymbol{\alpha}}(n-L)$ (36)
4. $\hat{e}(n) = s(n) + \mathbf{x}^T(n)\hat{\mathbf{h}}(n-1)$ (37)
5. $e_n = \hat{e}_n + \hat{\mathbf{r}}_{xx}^T(n)\bar{\mathbf{E}}(n-1)$ (38)
6. $\mathbf{E}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{E}}(n) \end{bmatrix} + \frac{1}{E_a(n)}\mu\boldsymbol{\varepsilon}_n$ (39)
7. $\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu \mathrm{x}(\mathrm{n\text{-}N+1})E_{N-1}(n)$ (40)

The complexity of FAP *without* relaxation is *2L+14N* which, when *L<<N*, is close to NLMS. The details are shown in Table 2.4. The total memory required by FAP is around 2*L*. The complexity of FAP *with* the relaxation parameter is 2*L*+20*N*; see [2,9,10,11] for complete details.

Table 2.4. Complexity of FAP without relaxation.

Equation	Multiplications
Update $E_a(n)$ and $\mathbf{a}(n)$ with FRLS.	10*N*
$\tilde{\mathbf{r}}_{xx}(n) = \tilde{\mathbf{r}}_{xx}(n-1) + x(n)\tilde{\boldsymbol{\alpha}}(n) + x(n-L)\tilde{\boldsymbol{\alpha}}(n-L)$	2*N*
$\hat{e}(n) = s(n) + \mathbf{x}^T(n)\hat{\mathbf{h}}(n-1)$	*L*
$e_n = \hat{e}_n + \hat{\mathbf{r}}_{xx}^T(n)\bar{\mathbf{E}}(n-1)$	*N*
$\mathbf{E}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{E}}(n) \end{bmatrix} + \frac{1}{E_a(n)}\mu\boldsymbol{\varepsilon}_n$	*N*
$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu \mathrm{x}(\mathrm{n\text{-}N+1})E_{N-1}(n)$	*L*
Total Complexity	2*L*+14*N*

2.3. SUB-BAND ADAPTIVE FILTER

Sub-band adaptive filters demonstrate the desirable properties of fast convergence at low computational complexity when compared to full-band adaptive filters. However, sub-band adaptive filters introduce some latency due to their analysis and synthesis filters. They also need careful designing to avoid aliasing effects [3,4].

Figure 2.2, shows the block diagram of a sub-band adaptive filter [1]. In sub-band adaptive filtering, the sub-band analysis filter bank divides the full-band excitation and the line signals into M sub-bands of equal width. An adaptive filter attempts to adjust the FIR coefficients in each sub-band to predict the echo path coming from the room (in the case of acoustic echo) or hybrid network (in the case of line echo). The error residual signals in each sub-band are obtained and passed on to the synthesis filter bank after non-linear processing. The sub-band synthesis filter then combines the M sub-band residual signals into a single full-band output signal [1]. Generally, a complex NLMS adaptive filter is used to estimate the adaptive FIR coefficients in each sub-band. Other adaptive filters like LMS, APA or FAP can also be used instead of NLMS filter. In this thesis complex NLMS adaptive filters are used.

Sub-band signals are sub-sampled by a factor *R*, sub-sampling of each sub-band decreases the overall computational complexity when compared to the full-band rate. Also, the sub-band adaptive filter lengths are *R* times shorter; therefore, sub-band structure provides *R* times as much time to do the processing. Computational savings over full-band processing is thus $\frac{M}{R^2} = \frac{2}{R}$, for M=2R.

Consider the conceptual operation of an analysis filter. Let $x(n)$ denote the full-band excitation signal. The m^{th} sub-band signal $x_m(k)$ is generated by multiplying the full-band signal $x(n)$ with the complex exponential $e^{-j2\pi mn/M}$, to shift the spectrum of $x(n)$ down in frequency by m sub-bands, and then the result is convolved with an analysis low-pass filter [1]. Finally, the signal is then sub-sampled by the factor *R*. For the efficient implementation of the analysis filter, polyphase filters are used. Figure 2.3 shows a polyphase implementation of the sub-band analysis filter. In the Figure 2.3, $h(n)$ is the analysis low-pass filter.

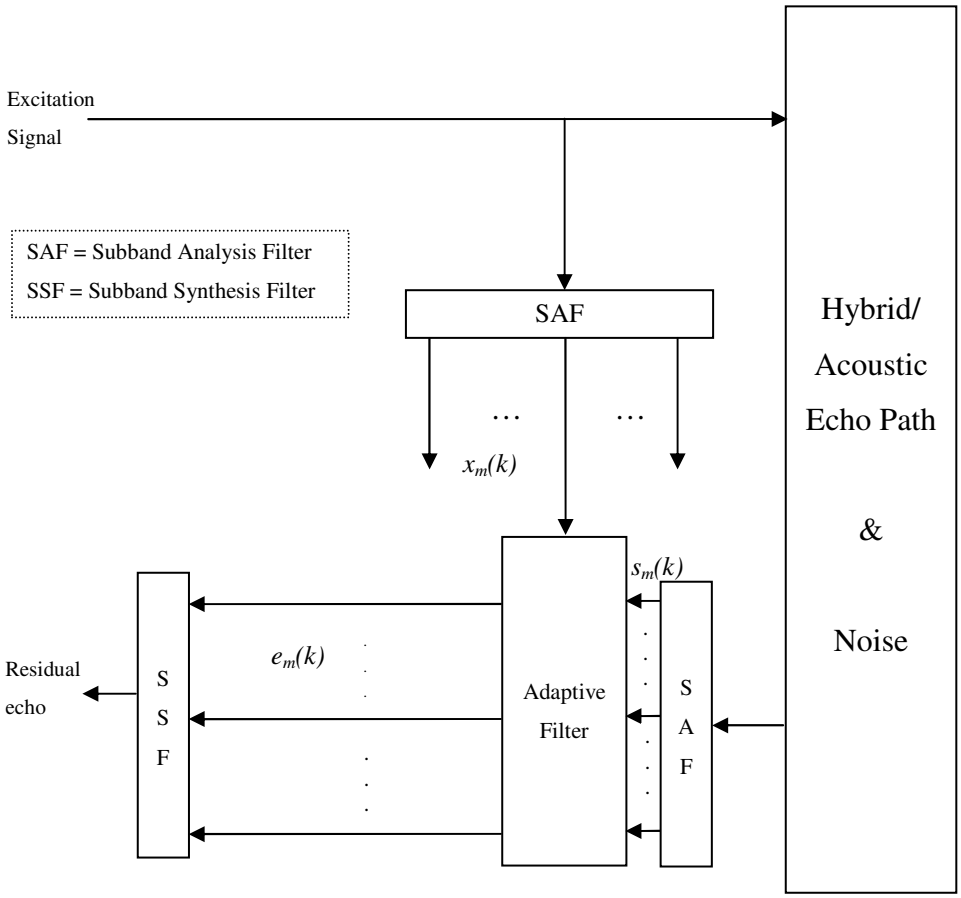

Figure 2.2. Sub-Band Adaptive Filter – Schematic Diagram.

The sub-band error signals, $e_m(k)$, are reconstructed by the synthesis filter bank. The sub-band error signals are up-sampled by *R*. Up-sampling results in images repeating in the frequency domain. Hence, the up-sampled signal is convoluted with a low-pass synthesis filter. These up-sampled, low-passed signals are then shifted up in frequency by m sub-bands by multiplication with the complex exponential $e^{j2\pi mn/M}$ [1]. The polyphase implementation of the sub-band synthesis filter is shown in Figure 2.4. In the Figure 2.4, $\hat{e}(n)$ is the reconstructed residual error signal and $g(n)$ is the synthesis low pass filter.

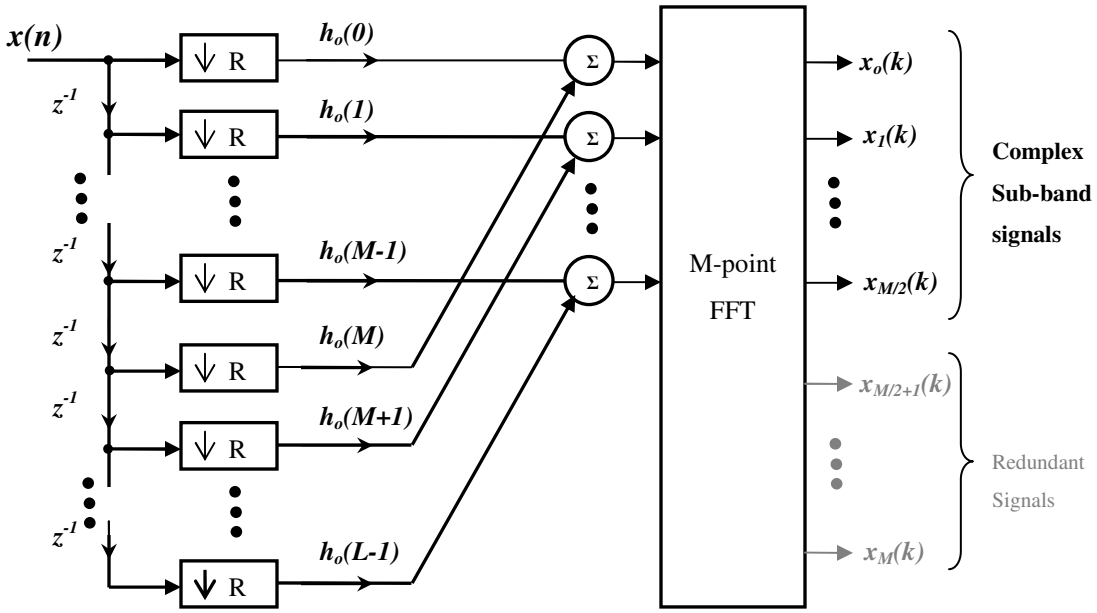

Figure 2.3. Polyphase Implementation of Analysis Filter Bank.

Let an excitation signal be band limited to f_o . Then the cross-over frequency of each sub-band would be $\frac{f_o}{M}$. To avoid aliasing $\frac{f_o}{R}$ must be greater than $\frac{f_o}{M}$, i.e. *R* must be less than *M*. As *R* is chosen close to *M*, the transition band of the analysis filter must narrow. Narrow transition is achieved with a low pass analysis filter of increased length and hence, the latency and computational complexity of the analysis filter increases. Therefore, it is desirable to make *R* large and *M* small to reduce the adaptive filter's complexity.

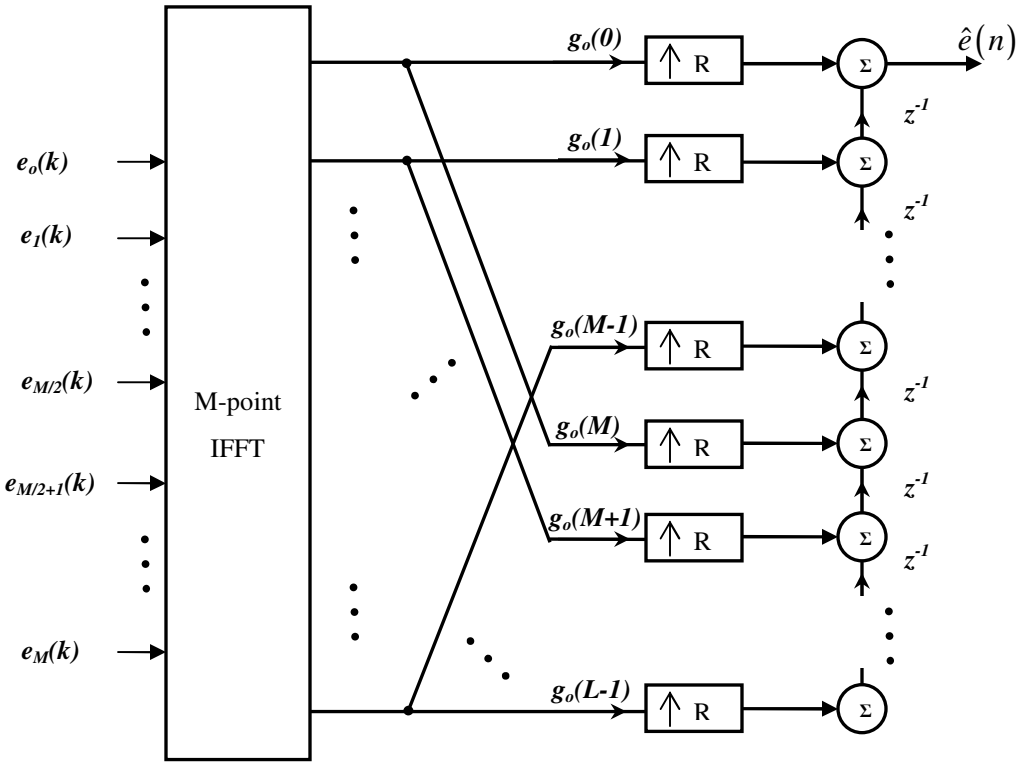

Figure 2.4. Polyphase Implementation of Synthesis Filter Bank.

3. VARIABLE REGULARIZED FAST AFFINE PROJECTIONS

3.1. INTRODUCTION

A wide variety of adaptive filters are now available in the signal processing community. Each has their advantages and disadvantages. The affine projection algorithm (APA) [15,9] has received considerable attention over the past 15 years or so because it's attributes provide a nice compromise between normalized least mean squares (NLMS) [8] and fast recursive least squares (FRLS) [14]. NLMS is computationally quite efficient and numerically stable, but converges rather slowly when a colored excitation signal is used. FRLS, is less computationally efficient and somewhat difficult to stabilize numerically, but has fast convergence for colored excitation.

The NLMS coefficient update method may be viewed as a one-dimensional affine projection in the parameter space. Under this view, APA is a generalization of NLMS in that it performs an *N*-dimensional affine projection each sample period [9,2]. When N is greater than or equal to the order of the source model that creates the excitation signal, APA's convergence properties are roughly the same as FRLS's [9]. Depending on the exact implementation, APA generally enjoys a much greater degree of numerical stability than FRLS algorithms – even the so-called stabilized ones. However, depending on *N*, APA's computational complexity can be higher than FRLS. To address this defect, the fast affine projection (FAP) [9,2,10,16] was introduced in the early 1990's. FAP reduced the computational complexity to roughly that of NLMS.

As the affine projection order, *N*, increases from one, a simple scalar inversion of the excitation vector's norm in NLMS becomes an *N*-by-*N* excitation sample covariance matrix inversion in APA. Often, with highly colored noise excitation, this sample covariance is ill-conditioned and to prevent undue noise amplification, a regularization parameter, δ is added to the matrix diagonal prior to inversion. In [17] a method for dynamically estimating an optimal regularization parameter for APA was described and the subsequent improvement in convergence for stationary excitation signals was demonstrated. This thesis work re-derives the optimal regularization for FAP. Traditionally, FAP's regularization is implemented by way of an initialization parameter that remains fixed thereafter; this static-regularization problem is overcome by using only

a very small initial regularization and then using the noise-injection technique to vary the regularization as determined by the VR method.

This section is arranged as follows: Sub-section 3.2 is a brief review of APA and FAP, Sub-section 3.3 presents the derivation of the variable regularization parameter for FAP and the use of the noise injection method. Finally, simulation results are presented in Sub-section 3.4 and conclusions in Sub-section 3.5.

3.2. REVIEW OF APA AND FAP

3.2.1. The Affine Projection Algorithm. This section presents a brief review of APA and a review of FAP to the extent that the VR algorithm for it may be derived. For a complete derivation of FAP, see [9,2,10,16].

$$\mathbf{e}(n) = \mathbf{s}(n) - \mathbf{X}^T(n)\mathbf{h}(n-1) \tag{41}$$

$$\boldsymbol{\varepsilon}(n) = [\mathbf{X}^T(n)\mathbf{X}(n) + \delta\mathbf{I}]^{-1}\mathbf{e}(n) \tag{42}$$

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mathbf{X}^T(n)\boldsymbol{\varepsilon}(n) \tag{43}$$

The excitation signal matrix, $\mathbf{X}(n)$, is L by N and has the structure,

$$\mathbf{X}(n) = [\mathbf{x}(n), \mathbf{x}(n-1), \cdots, \mathbf{x}(n-N+1)] \tag{44}$$

where, the $\mathbf{x}(n) = [x(n), \cdots, x(n-L+1)]^T$. The adaptive tap weight vector is $\mathbf{h}(n) = [h_0(n), \cdots, h_{L-1}(n)]^t$, where $h_i(n)$ is the i^{th} coefficient at sample period n. The N-length vector, $\mathbf{e}(n)$, consists of background noise and residual echo left uncancelled by the echo canceller's L-length adaptive tap weight vector, $\mathbf{h}(n)$. The N-length vector, $s(n)$, is the system output consisting of the response of the echo path impulse response, $\mathbf{h}_{true}$ to the excitation and the additive system noise, $\mathbf{y}(n)$,

$$\mathbf{s}(n) = \mathbf{X}^T(n)\mathbf{h}_{ture} + \mathbf{y}(n) \tag{45}$$

The scalar δ is the regularization parameter for the sample autocorrelation matrix inverse used in (42), the calculation of the *N*-length normalized residual echo vector, $\boldsymbol{\varepsilon}(n)$. Where $\mathbf{X}^T(n)\mathbf{X}(n)$ may have eigenvalues close to zero, creating problems for the inverse, $\mathbf{X}^T(n)\mathbf{X}(n)+\delta\mathbf{I}$ has δ as its smallest eigenvalue which, if large enough, yields a well behaved inverse. The step-size parameter μ is the relaxation factor. As in NLMS, the algorithm is stable for $0 \le \mu < 2$.

Defining the coefficient error vector as $\Delta\mathbf{h}(n) = \mathbf{h}_{true} - \mathbf{h}(n)$, the error vector, $\mathbf{e}(n)$ may be written as,

$$\mathbf{e}(n) = \mathbf{X}^T(n)\Delta\mathbf{h}(n) + \mathbf{y}(n) \tag{46}$$

If the order of projection, *N* is set to one, relations (41), (42) and (43) reduce to the familiar NLMS algorithm. Thus, APA is a generalization of NLMS.

3.2.2. The Fast Affine Projection Algorithm. The complexity of APA is $2LN + K_{inv}N^2$ multiplies per sample period, where K_{inv} is a constant associated with the complexity of the inverse required in (42). FAP performs a complete *N*-dimensional APA update each sample period with $2L + O(N)$ multiplies per sample [9,2,10,16]. The development of FAP involves reducing the computational complexity of each of the steps in equations (41), (42) and (43). For the variable regularization derivation in Section 3.3 it is needed to review only the FAP's computational reduction of equation (43).

3.2.3. Fast Adaptive Coefficient Vector Calculation. The "trick" used in FAP to reduce the computational complexity of the coefficient update equation for $\mathbf{h}(n)$ is to introduce an alternate coefficient vector, $\hat{\mathbf{h}}(n)$, whose update each sample period consists only of adding a weighted version of the last column of $\mathbf{X}(n)$. This requires only *L* multiplications as opposed to *NL* for the update of equation (43). FAP also provides a method for calculating $\mathbf{e}(n)$ from $\hat{\mathbf{h}}(n)$ which is not shown in this work.

From (43) the APA tap update is,

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mu\mathbf{X}(n)\boldsymbol{\varepsilon}(n) \tag{47}$$

One can also express the current echo path estimate, $\mathbf{h}(n)$, in terms of the original echo path estimate, $\mathbf{h}(0)$, and the subsequent $\mathbf{X}(i)'s$ and $\boldsymbol{\varepsilon}(i)'s$,

$$\mathbf{h}(n) = \mathbf{h}(0) + \mu \sum_{i=0}^{n-1} \mathbf{X}(n-i)\boldsymbol{\varepsilon}(n-i) \tag{48}$$

Now, expanding the vector/matrix multiplication,

$$\mathbf{h}(n) = \mathbf{h}(0) + \mu \sum_{i=0}^{n-1} \sum_{j=0}^{N-1} \mathbf{x}(n-j-i)\boldsymbol{\varepsilon}_j(n-i) \tag{49}$$

Assuming that $\mathbf{x}(n) = 0$ for $n \le 0$, (49) can be rewritten as,

$$\mathbf{h}(n) = \mathbf{h}(0) + \mu \sum_{k=0}^{N-1} \mathbf{x}(n-k) \sum_{j=0}^{k} \boldsymbol{\varepsilon}_j(n-k+j) + + \mu \sum_{k=N}^{n-1} \mathbf{x}(n-k) \sum_{j=0}^{N-1} \boldsymbol{\varepsilon}_j(n-k+j) \tag{50}$$

If the first term and the second pair of summations on the right side of (50) are defined as,

$$\hat{\mathbf{h}}(n-1) = \mathbf{h}(0) + \mu \sum_{k=N}^{n-1} \mathbf{x}(n-k) \sum_{j=0}^{N-1} \boldsymbol{\varepsilon}_j(n-k+j) \tag{51}$$

and the first pair of the summations in (50) are recognized as a vector-matrix multiplication,

$$\mathbf{X}(n)\mathbf{E}(n) = \mu \sum_{k=0}^{N-1} \mathbf{x}(n-k) \sum_{j=0}^{k} \boldsymbol{\varepsilon}_j(n-k+j) \tag{52}$$

where,

$$\mathbf{E}(n) = \begin{bmatrix} \varepsilon_0(n) \\ \varepsilon_1(n) + \varepsilon_0(n-1) \\ \vdots \\ \varepsilon_{N-1}(n) + \varepsilon_{N-2}(n-1) \cdots + \varepsilon_0(n-(N-1)) \end{bmatrix} \tag{53}$$

then, (50) can be expressed as,

$$\mathbf{h}(n) = \hat{\mathbf{h}}(n-1) + \mu \mathbf{X}(n)\mathbf{E}(n) \tag{54}$$

It is easily seen from (51) that,

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu \mathbf{x}\left(n-(N-1)\right)\sum_{j=0}^{N-1} \boldsymbol{\varepsilon}_j (n-N+1+j) \tag{55}$$

or,

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu \mathbf{x}\left(n-(N-1)\right) E_{N-1}(n) \tag{56}$$

where, $E_{N-1}(n)$ is the last element of $\mathbf{E}(n)$ (note, the elements are numbered from 0 to *N*-1).

3.3. VARIABLE REGULARIZATION FOR FAP

Now make the regularization parameter variable with time, explicitly denoting it as $\delta(n)$ [18]. In the following derivation it is assumed that the relaxation parameter, $\mu = 1$. Similar to the approach in [10] chose the optimization criterion which minimizes the cost function [18],

$$J = E\left(\left\|\Delta\hat{\mathbf{h}}(n)\right\|^2\right) - E\left(\left\|\Delta\hat{\mathbf{h}}(n-1)\right\|^2\right) \tag{57}$$

where,

$$\Delta\hat{\mathbf{h}}(n) = \mathbf{h}_{true} - \hat{\mathbf{h}}(n) \tag{58}$$

is the FAP coefficient error vector. Applying (58) to (56) yields,

$$\mathrm{h}_{true} - \Delta\hat{\mathbf{h}}(n) = \mathrm{h}_{true} - \Delta\hat{\mathbf{h}}(n-1) + \mathbf{x}(n-N+1) E_{N-1}(n) \tag{59}$$

$$\Delta\hat{\mathbf{h}}(n) = \Delta\hat{\mathbf{h}}(n-1) - \mathbf{x}(n-N+1) E_{N-1}(n) \tag{60}$$

From (53), it is seen that

$$E_{N-1}(n) = \varepsilon_{N-1}(n) + \varepsilon_{N-2}(n-1) \cdots + \varepsilon_0\left(n-(N-1)\right) \tag{61}$$

Defining,

$$\mathbf{S}(n) = [\mathbf{X}^T(n)\mathbf{X}(n) + \delta\mathbf{I}]^{-1}, \tag{62}$$

Now, make the simplifying assumption that the sample covariance matrix, $\mathbf{X}^T(n)\mathbf{X}(n)$ is fixed and equal to $L\mathbf{R}_x$ where $\mathbf{R}_x$ is the correlation matrix of $\mathbf{x}(n)$. This is a reasonable assumption when $N \ll L$. This implies that

$$\mathbf{S} \approx [L\mathbf{R}_x + \delta\mathbf{I}]^{-1} \tag{63}$$

where, the time index is removed to emphasize the assumption that $\mathbf{S}$ is now non-time-varying. Under these assumptions (42) becomes

$$\boldsymbol{\varepsilon}(n) \approx \mathbf{S}\mathbf{e}(n) \tag{64}$$

Since it is assumed that the step-size, $\mu = 1$, from [2] it is known that FAP sets

$$\mathbf{e}(n) = \left[e(n), \mathbf{0}_{N-1}^T\right]^T \tag{65}$$

where, $\mathbf{0}_k$ is an k-length all zero vector. Thus,

$$\boldsymbol{\varepsilon}(n) \approx \mathbf{S}\mathbf{p}_0 e(n) \tag{66}$$

where, $\mathbf{p}_k = \left[\mathbf{0}_k^T,\ 1,\ \mathbf{0}_{N-k-1}^T\right]^T \quad 0 \le k \le N-1$. By careful inspection one can observe that

$$\varepsilon_{N-1-k}(n-k) \approx \mathbf{p}_{N-1-k}^T \mathbf{S}\mathbf{p}_0 e(n-k) \tag{67}$$

Thus, with some manipulation one can rewrite (61) as,

$$E_{N-1}(n) = \mathbf{p}_0^T \mathbf{S}\breve{\mathbf{e}}^R(n) \tag{68}$$

where, the R in the superscript of $\breve{\mathbf{e}}(n)$ denotes the "reverse" of the vector and $\breve{\mathbf{e}}(n)$ is given by,

$$\breve{\mathbf{e}}(n) = \begin{bmatrix} e(n) & e(n-1) & \cdots & e(n-N+1) \end{bmatrix}^T \tag{69}$$

is a history of the *N* most recent FAP residual errors – not to be confused with $\mathbf{e}(n)$ of (41) and certainly not that of (65). One can now express (60) as,

$$\Delta\hat{\mathbf{h}}(n) = \Delta\hat{\mathbf{h}}(n-1) - \mathbf{x}(n-N+1)\mathbf{p}_0^T\mathbf{S}\breve{\mathbf{e}}^R(n) \tag{70}$$

Using (70) in (57) , write the cost function as,

$$\begin{aligned} J = E\left(\left\|\Delta\hat{\mathrm{h}}(n-1)\right\|^2\right) + E\left(\left\|\mathbf{x}(n-N+1)\right\|^2\left(\mathbf{p}_0^T\mathbf{S}\breve{\mathbf{e}}^R(n)\right)^2\right) \\ - E\left(2\Delta\hat{\mathbf{h}}^T(n-1)\mathbf{x}(n-N+1)\mathbf{p}_0^T\mathbf{S}\breve{\mathbf{e}}^R(n)\right) \\ - E\left(\left\|\Delta\hat{\mathrm{h}}(n-1)\right\|^2\right) \end{aligned} \tag{71}$$

$$J = E\left(\left\|\mathbf{x}(n-N+1)\right\|^2\left(\mathbf{p}_0^T\mathbf{S}\breve{\mathbf{e}}^R(n)\right)^2\right) - E\left(2\Delta\hat{\mathbf{h}}^T(n-1)\mathbf{x}(n-N+1)\mathbf{p}_0^T\mathbf{S}\breve{\mathbf{e}}^R(n)\right) \tag{72}$$

Assuming the excitation signal is white,

$$\mathbf{S} \approx \frac{1}{\delta(n) + L\sigma_x^2}\mathbf{I} \tag{73}$$

Then,

$$\mathbf{p}_0^T\mathbf{S}\breve{\mathbf{e}}^R(n) = \frac{\breve{e}_0{}^R(n)}{\delta(n) + L\sigma_x^2} = \frac{e(n-N+1)}{\delta(n) + L\sigma_x^2} \tag{74}$$

Assuming that $\Delta\hat{\mathbf{h}}^T(n-1)$ changes slowly enough that $\Delta\hat{\mathbf{h}}^T(n-1) \approx \Delta\hat{\mathbf{h}}^T(n-N)$ and the filter is close enough to convergence that $\Delta\hat{\mathbf{h}}^T(n-N) \approx \Delta\mathbf{h}^T(n-N)$, then

$$\Delta\hat{\mathbf{h}}^T(n-1)\mathbf{x}(n-N+1) \approx \Delta\mathbf{h}^T(n-N)\mathbf{x}(n-N+1) \tag{75}$$

Using this in (46) at the sample period $n-N+1$ yields,

$$\Delta\hat{\mathbf{h}}^T(n-1)\mathbf{x}(n-N+1) \approx e(n-N+1) - y(n-N+1) \tag{76}$$

Applying (73) and (76) to (72), and assuming $\left\|\mathbf{x}(n-N+1)\right\|^2 \approx L\sigma_x^2$, the performance index can be written as,

$$J = \frac{\left(L\sigma_x^2\right)\left(E\left(e^2(n-N+1)\right)\right)}{\left(\delta(n)+L\sigma_x^2\right)^2} - \frac{E\left(e(n-N+1) - y(n-N+1)\right)e(n-N+1)}{\left(\delta(n)+L\sigma_x^2\right)} \tag{77}$$

$$J = \left(\frac{L\sigma_x^2}{\left(\delta(n)+L\sigma_x^2\right)^2} - \frac{2}{\left(\delta(n)+L\sigma_x^2\right)}\right)E\left(e(n-N+1)^2\right) + \frac{2E\left(y(n-N+1)e(n-N+1)\right)}{\left(\delta(n)+L\sigma_x^2\right)} \tag{78}$$

From (76), can be written as,

$$E\left(y(n-N+1)e(n-N+1)\right) = E\left(y(n-N+1)\left(\Delta\hat{\mathbf{h}}^T(n-1)\mathbf{x}(n-N+1)\right) + y(n-N+1)\right) \tag{79}$$

Assuming that the excitation energy is independent from the background noise,

$$E\left(y(n-N+1)\left(\Delta\hat{\mathbf{h}}^T(n-1)\mathbf{x}(n-N+1)\right)\right) \approx 0 \tag{80}$$

Applying (79) , (80) to (78) and assuming $E\left(y(n-N+1)^2\right) = \sigma_y^2$, the cost function J can be written as,

$$J \approx \frac{2\sigma_y^2}{\delta(n)+L\sigma_x^2} - \frac{L\sigma_x^2 + 2\delta(n)}{\left(\delta(n)+L\sigma_x^2\right)^2}E\left(e^2(n-N+1)\right) \tag{81}$$

Consider $E\left(e^2(n-N+1)\right) = \psi$ for readability. Minimizing J with respect to $\delta(n)$ yields,

$$\frac{\partial J}{\partial \delta(n)} = \frac{-2\sigma_y^2}{\left(\delta(n)+L\sigma_x^2\right)^2} + \frac{2\left(2\delta(n)+L\sigma_x^2\right)\psi}{\left(\delta(n)+L\sigma_x^2\right)^3} - \frac{2\psi}{\left(\delta(n)+L\sigma_x^2\right)^2} = 0 \tag{82}$$

$$-\sigma_y^2\left(\delta(n)+L\sigma_x^2\right)+\left(2\delta(n)+L\sigma_x^2\right)\psi-\left(\delta(n)+L\sigma_x^2\right)\psi = 0 \tag{83}$$

$$-\sigma_y^2\left(\delta(n)+L\sigma_x^2\right)+\left(\delta(n)+L\sigma_x^2\right)\psi+\delta(n)\psi-\left(\delta(n)+L\sigma_x^2\right)\psi = 0 \tag{84}$$

$$\delta(n) = \frac{L\sigma_y^2\sigma_x^2}{E\left(e^2\left(n-N+1\right)\right)-\sigma_y^2} \tag{85}$$

As in [17] estimate $E\left(e^2\left(n-N+1\right)\right)$ with a time average. From the simulations it is found that a time average of length *L* is sufficient. The estimated $E\left(e^2\left(n-N+1\right)\right)$ quantity may result in a magnitude lower than σ_y^2. Therefore, when the denominator in equation (85) goes negative the value of $\delta(n)$ is limited to δ_{MAX}. δ_{MAX} is given as

$$\delta_{MAX} = \frac{\sigma_x^2}{\delta_{\text{limit}}} \tag{86}$$

The regularization of (85) is applied to FAP via the technique of noise injection. Noise injection is a method of regularizing a signal's covariance by adding gaussian white noise. The standard deviation of the noise is set to the square root of the desired regularization value. In VR-FAP, this noise is only added in the excitation signal input to the sliding window FRLS part where the calculation of the forward and backward linear predictors and their estimation error energies are calculated [9,2,10,16].

3.4. SIMULATIONS

The performance of the VR-FAP was compared with FAP and VR-APA adaptive filters. The measure of the performance was given by the misadjustment or the coefficient error in *dB*, κ. κ is defined as,

$$\kappa = 10\log 10\left(\left(\mathbf{h}_{true}-\mathbf{h}\right)^T\left(\mathbf{h}_{true}-\mathbf{h}\right)\right) \tag{87}$$

where, $\mathbf{h}_{true} = [h_{true,0}, h_{true,1}, \cdots, h_{true,L-1}]^T$ is the true echo path and $\mathbf{h}=[h_0, h_1, \cdots, h_{L-1}]^T$ is the estimated echo path by the adaptive filters. L is the length of the filter, i.e. number of coefficients. For all the simulations, L was set to 512 and the true echo path used for the simulations is as shown in Figure 3.1.

Figure 3.2 shows a comparison of the convergence of FAP, VR-APA and VR-FAP. The excitation signal was white Gaussian noise. The length of the echo path was set to L=512. The projection order, N=2 and the additive noise, $\mathbf{y}(n)$, was set 30dB down from the echo signal. The δ_{limit} for VR-FAP was set to $(0.001)\sigma_x^2$. For FAP the fixed regularization was $\delta = (5)\sigma_x^2$. For VR-APA the parameter γ (referred to as δ in [17]) was set to 0.05. VR-FAP performs very well when compared to FAP; the VR-FAP coefficient error goes below -50dB while FAP bottomed-out at -29dB. VR-APA and VR-FAP have similar initial convergence rate but VR-APA only reaches -43dB.

Figure 3.3 shows the convergence curves of the FAP, VR-APA and VR-FAP for colored noise input. The colored noise was generated using and auto regressive model with one pole at z-0.95, AR1(0.95). The total length of the echo path was L=512 and the additive noise, $y(n)$, was set to 30dB lower than the echo. The order of the projection, N was 2. δ for VR-FAP and FAP are the same as above. γ for VR-APA is set to 0.08. The performance of the proposed algorithm is significantly improved over FAP. VR-FAP has fast initial convergence rate and finally converges to a value around -32dB where as FAP settles down at around -17dB. For all the simulations the sampling frequency was considered to be 8000Hz.

Figure 3.4 shows the convergence curves of the FAP, VR-APA and VR-FAP for speech excitation. The total length of the echo path was L=512 and the additive noise, $y(n)$, was set to 30dB lower than the echo. The order of the projection, N was 2. Initial δ for VR-FAP was set to $\delta_{limit} = (0.1)\sigma_x^2$ and the fixed regularization for FAP was set to $\delta = (10)\sigma_x^2$. γ for VR-APA is set to 10. The proposed VR-FAP algorithm performs well with the speech excitation. VR-FAP finally converges to a value around -18dB and FAP settles down at around -15dB. For all the simulations the sampling frequency was considered to be 8000Hz.

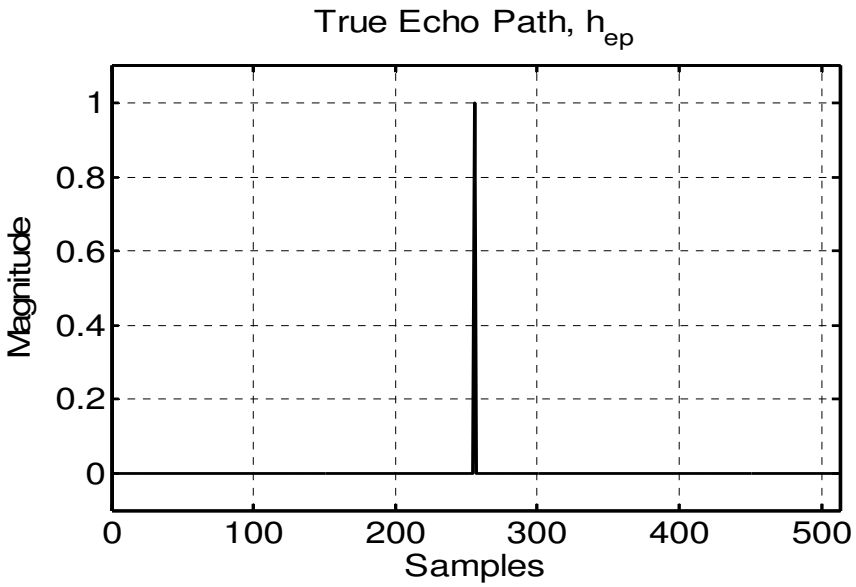

Figure 3.1. True Echo Path, $\mathbf{h}_{ep}$.

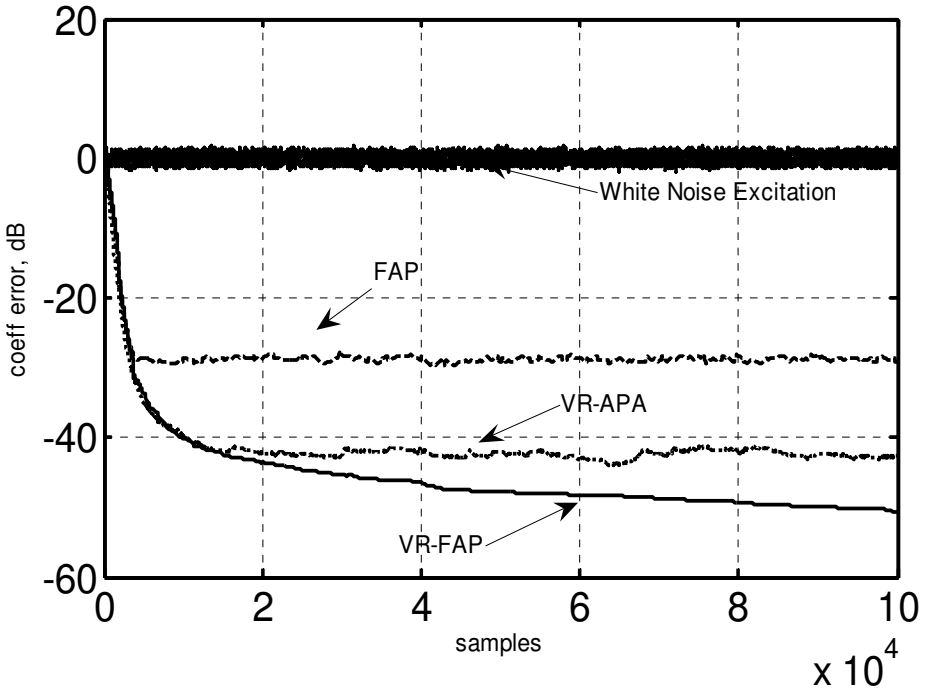

Figure 3.2. Coefficient Error (in *dB*) for White Excitation Noise, L=512, N=2, SNR=30*dB*.

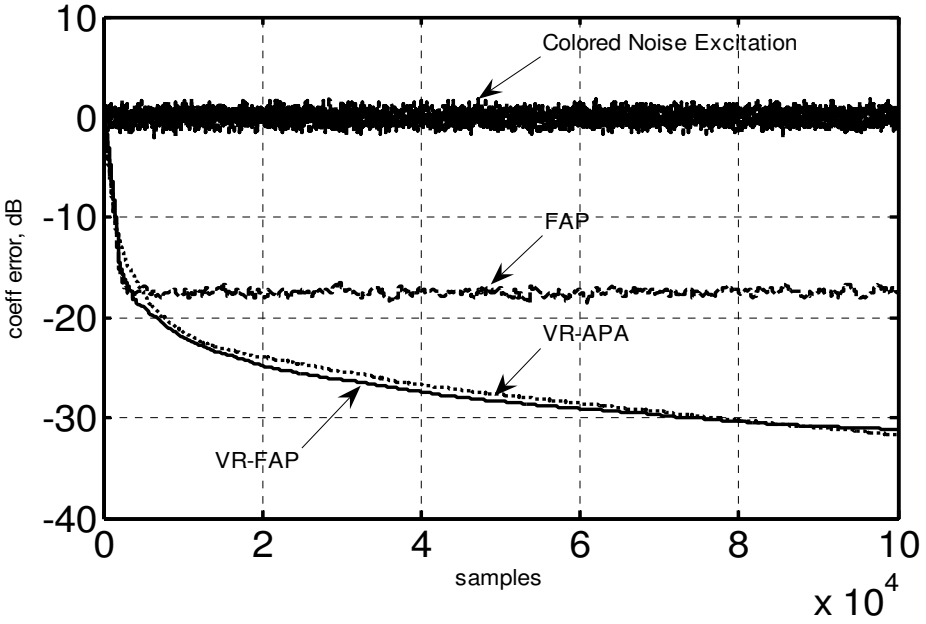

Figure 3.3. Coefficient Error (in *dB*) for Colored Noise AR1(0.95), L=512, N=2, SNR=30*dB*.

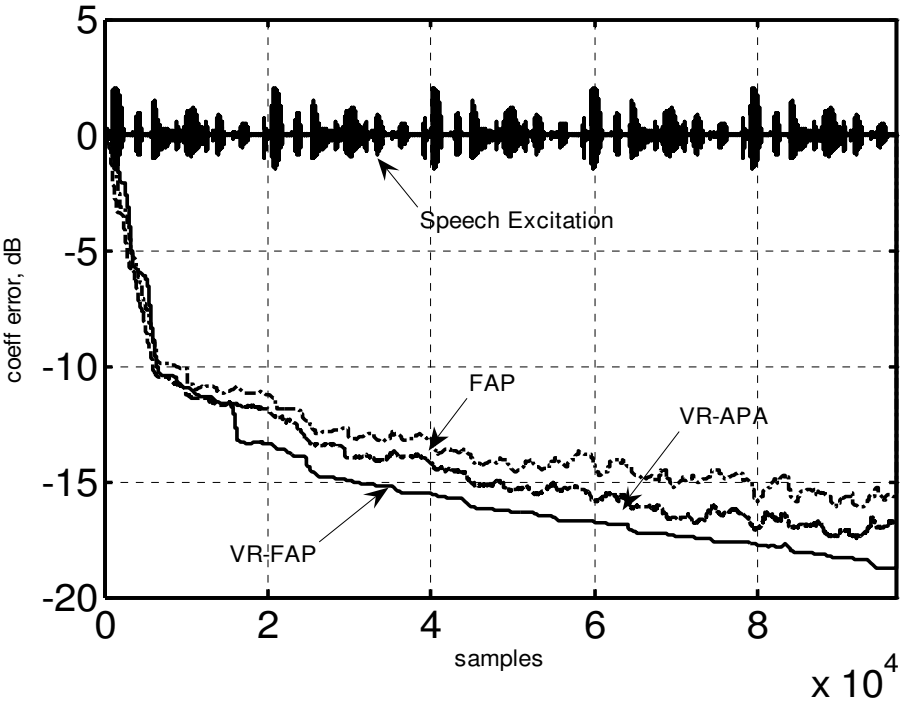

Figure 3.4. Coefficient Error (in *dB*) for Speech Excitation, L=512, N=2, SNR=30*dB*.

4. COMPUTATIONAL INTELLIGENCE METHODS

Computational intelligence (CI) is a field which involves the study of adaptive mechanisms to facilitate intelligent behavior in complex and varying environments [5]. These mechanisms include artificial intelligence (AI) topics namely artificial neural networks (ANN), swarm intelligence, evolutionary computing and fuzzy systems [5]. The ability to learn and adapt to new situations can be defined as the intelligent behavior demonstrated by AI techniques [5]. Particle swarm optimization (PSO) is one of the swarm intelligence based algorithms used for optimizing a set of parameters [6]. In this thesis work an ANN combined with PSO is used to search for a solution in a multi dimensional space.

4.1. ARTIFICIAL NEURAL NETWORK, ANN

ANN, also known as Multi Layer Perceptron, MLP can be defined as interconnected group of simple processing units called neurons [5]. These neurons are generally connected in parallel and layer fashion to perform the required mathematical or logical operation. The neuron connections are weighted with so-called *synaptic weights* [5]. A single neuron consists of synaptic weights, a summation or product operator and an activation function (also called a transfer function). Figure 4.1 shows a model of an artificial neuron with a summation unit [5,19].

The output of a neuron is defined as,

$$y = \psi\left(\sum_{j=1}^{n} w_{ij} x_j + \Theta\right) \tag{88}$$

where, Θ is the external threshold. It is also called as bias term or offset. $\Psi(n)$ is the activation function. There can be unipolar or bipolar activation functions (e.g. sigmoid, hard limiter, ramp, hyperbolic, Gaussian). The weights, w_{ji} are the synaptic weights of the neuron. These weights are adjusted by optimization algorithms to get the desired output. The inputs are the x_j 's and y is the output of the neuron [5].

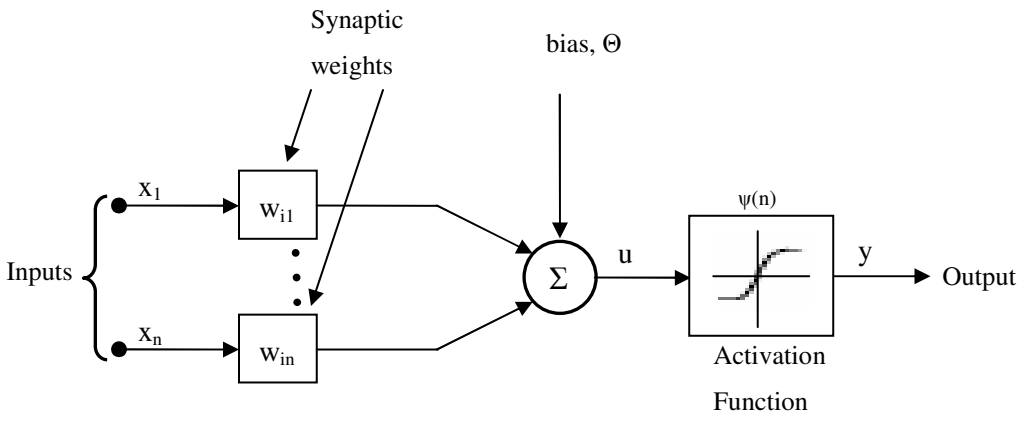

Figure 4.1. Artificial Neuron with Summation Unit.

A neural network can be designed for an adjustable number of layers, neurons, inputs and outputs. A schematic representation of a $3\times2\times2$ feed forward multi-layer perceptron organized in three layers, input, hidden and output is shown in Figure 4.2.

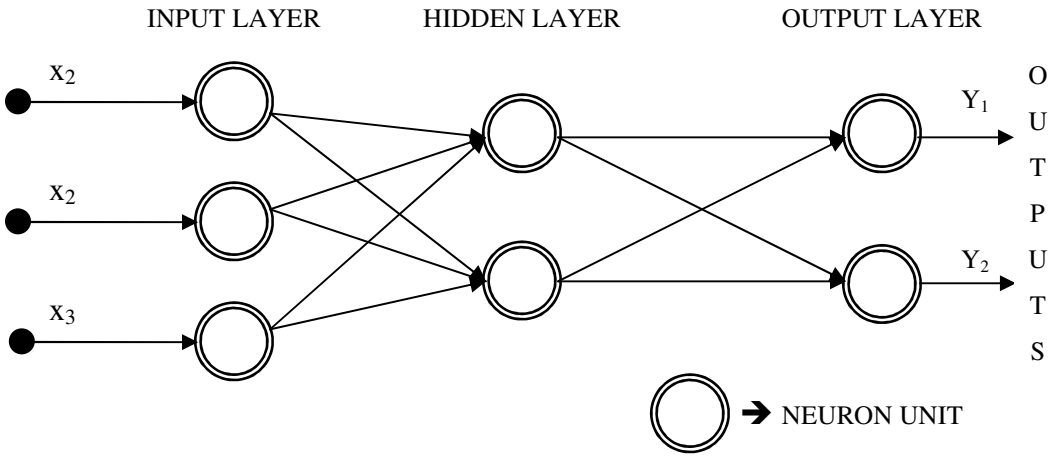

Figure 4.2. Schematic Diagram of an Artificial Neural Network.

4.2. PARTICLE SWARM OPTIMIZATION (PSO)

Particle swarm optimization (PSO) is an algorithm which belongs to the swarm intelligence field [6,7,20,19]. PSO adjusts the synaptic weights of the NN to give the desired optimal outputs and parameters based on the *fitness function* or *performance index* of the NN outputs.

Discovery of the PSO algorithm was based on the observations of a simplified social model where the individuals arrive at the solution working together in a collective manner [6]. PSO consists of a group (population) of *particles*, which search for the best solution in a given 'n' dimensional space. The number of dimensions of the space being searched is defined by the number of NN weights. The particles in the search *fly* within the boundaries of the given dimensional space at different *velocities* which are dynamically adjusted according to their personal and social (population) historical behavior [6,7,20]. Therefore, each particle of the population is a potential solution. Every particle records its personal best values, 'pbest', recorded with reference to the fitness of the NN output. The best of the 'pbest' is considered to be the global best (the best fit particle of the population), 'gbest', and is updated continuously. The particles move towards or search for the solution in the 'n' dimensional space with the help of velocity. Each particle has a different velocity and also different velocity in each dimension. The velocity is calculated with the past experience, referring to the 'pbest' and 'gbest' values. An inertia term is also involved. The velocity calculation also involves randomness. This random nature helps avoid convergence to local minima. Thus, the particles are approach the solution in every iteration/epoch. Boundaries defined for the velocities and the weights of the neural network act as the total search space boundaries [6,7,20].

In this work, PSO is used for offline training of the NN. The pseudo code for particle swarm optimizer is as follows [6,7,19,20]:

Step 1: Randomly initialize the weights of the particles, velocity, pbest, fitness vector and define the number of particles, velocity and weight boundaries, inertia weight, cognitive acceleration term and social acceleration term, and the number of epochs (or iterations).

Step 2: Evaluate the fitness of each particle using the fitness function by providing all the inputs (batch).

Step 3: For each particle, update the fitness vector and pbest vectors (pbest) if the current fitness is better.

Step 4: Find 'gbest' value. Exit if the 'target fitness' is achieved, else go to Step 5.

Step 5: Inform each particle of the new gbest vector.

Step 6: Calculate the velocity, using the velocity equation.

$$V_{id} = w * V_{id} + C_1 * rand() * (P_{bestid} - X_{id}) + C_2 * rand() * (G_{bestid} - X_{id}); \quad (89)$$

where,

V_{id}, is the velocity vector.

X_{id}, is the current weights of each particle.

w, is the inertia term.

C_1, is the Cognitive acceleration term.

C_2, is the Social acceleration term.

P_{bestid}, is the vector of weights which stores the personal best of each particle.

G_{bestid}, is the vector of weights which stores the Global best of all particles.

$rand()$, is a function generating the random variable of uniform distribution

Step 7. Update the weights of the particles with the velocity and go to step 2.

$$X_{id} = X_{id} + V_{id} \quad (90)$$

It can be observed from equation (89) that near the optimal solution, $(P_{bestid} - X_{id})$ and $(G_{bestid} - X_{id})$ become very small [7,6]. Due to this effect, velocity, V_{id} damps out in some particles, resulting in losing the search space's exploration property which limits the convergence of the algorithm [7]. Therefore, a so-called *differential evolution* (DE)

operator derived from evolutionary computing techniques is applied to improve the G_{bestid} solutions to overcome this defect. PSO with DE is known as DEPSO. The differential evolution process involves identifying 'n' random particles for P_{bestid} solutions in every epoch. These randomly identified P_{bestid} solutions are *mutated* with the gbest particle in the current epoch to generate an *offspring* using the differential operator [7]. The offspring is tested for fitness and compared with its gbest parent. If the offspring is found to be a better fit than the gbest parent, the offspring replaces the gbest parent particle in the population. Thus, this hybrid DEPSO-gbest version overcomes the problem of zero damping and stimulates the search for a better solution [7]. The DEPSO-gbest version introduces the following computations in the PSO algorithm for every epoch/iteration [7],

Step1. Calcluate one offspring for every epoch.

$$T_{id} = G_{bestid} + \delta_{N,d} \tag{91}$$

$$\bar{\delta}_N = \frac{\left(\sum_1^N \vec{\Delta}\right)}{N} \tag{92}$$

where, CR is the so-called *crossover mutation* constant, T_{id} is the offspring to be tested for fitness, $\vec{\Delta}$ is the difference vector of the two randomly chosen P_{bestid} ,

$$\vec{\Delta}_d = P_{bestid,A} - P_{bestid,B} \tag{93}$$

where, $P_{bestid,A}$ and $P_{bestid,B}$ are the randomly selected P_{bestid} from the population.

Step2. Check for the fitness of T_{id} . If T_{id} is found to have better fitness than G_{bestid} , replace G_{bestid} with T_{id} in the population.

Step3. Go to the PSO algorithm steps.

DEPSO has two versions. When applied to P_{bestid}, it is known as the DEPSO-pbest version. When applied to G_{bestid}, it is known as the DEPSO-gbest version [7]. Both of these versions generally have similar convergence properties and perform better than PSO. This thesis work uses the DEPSO-gbest version to solve the optimization problem.

5. OPTIMIZATION OF ANALYSIS AND SYNTHESIS FILTER DESIGN PARAMETER – A COMPUTATIONAL INTELLIGENCE APPROACH

5.1. FILTER DESIGN PARAMETERS

Sub-band Analysis and Synthesis filter banks are a special type of modulated low pass FIR filters [3,4]. The non-ideal characteristics of the filter banks distort the signals in the sub-bands due to the pass-band ripple and aliasing effects from the neighboring sub-bands (in-band aliasing). This aliasing appears as noise to the adaptive filters within the sub-bands and result in improper adaptation of the coefficients and hence, degrades the performance of the overall sub-band adaptive filter. To avoid such in-band aliasing, the filter bank is desired to have a very sharp cut-off and also very low stop band ripple. However, practical always have some aliasing, therefore in order to yield good reconstruction, the synthesis filter bank is designed such that the aliasing terms are cancelled out and the residual aliasing in the output full-band signal is near zero [4]. It is also desired to have minimal latency through the analysis and synthesis filters. The design of such filters involves choosing optimal values for several parameters [4].

An analysis filter is designed with reference to an objective function. In this thesis work, the objective function used for analysis filter design is [4],

$$e_h = \alpha_h + \beta_h \tag{94}$$

where, α_h is the pass-band response error and β_h is the distortion due to in-band aliasing, α_h and β_h can be defined as [4]. The pass-band response error is the integral of the squared magnitude of the error of the designed prototype with respect to the ideal desired response over the pass band frequencies,

$$\alpha_h = \frac{1}{2\omega_p}\int_{-\omega_p}^{\omega_p}\left|H\left(e^{j\omega}\right) - H_{LP}\left(e^{j\omega}\right)\right|^2 d\omega \tag{95}$$

Where we define the ideal desired response as a pure delay of τ_h whose Fourier transform is,

$$H_{LP}\left(e^{j\omega}\right) = e^{-j\omega\tau_h}, \quad \omega \in \left[-\omega_p, \omega_p\right] \tag{96}$$

where, ω_p is the pass-band edge frequency, $H_{LP}\left(e^{j\omega}\right)$ is the Fourier transform of the desired low pass filter response, τ_h is the group delay of the prototype analysis filter bank, and $H\left(e^{j\omega}\right)$ is the Fourier transform of the desired response of the analysis filter to be designed.

The distortion due to in-band aliasing is the integral of the sum of the aliased part of the mean magnitude square of the aliased part of the designed filter after sub-sampling,

$$\beta_h = \frac{1}{2\pi}\int_{-\pi}^{\pi}\sum_{k=1}^{D-1}\left|H\left(e^{\frac{j\omega}{D}}W_D^k\right)\right|^2 d\omega \tag{97}$$

where, D is the number of distortion terms due to aliasing, equal to the sub-sampling factor minus 1. Now consider, $\mathbf{h} = \left[h_0, h_1, \cdots, h_{N_h-1}\right]^T$ and $\boldsymbol{\varphi}_{\mathbf{h}}(z) = \left[1, z^{-1}, \ldots, z^{-N_h+1}\right]^T$, $H(z)$ can be written as,

$$H(z) = \mathbf{h}^T\boldsymbol{\varphi}_{\mathbf{h}}(z) \tag{98}$$

Applying (98) to (95), (97) and simplifying, α_h and β_h in quadratic form can be written as [4],

$$\alpha_h = \mathbf{h}^T\mathbf{A}\mathbf{h} - 2\mathbf{h}^T\mathbf{b} + 1 \tag{99}$$

$$\beta_h = \mathbf{h}^T\mathbf{C}\mathbf{h} \tag{100}$$

where,

$$A_{i,k} = \frac{\sin\left(\omega_p(k-i)\right)}{\omega_p(k-i)} \tag{101}$$

$$b_i = \frac{\sin\left(\omega_p(\tau_h - i)\right)}{\omega_p(\tau_h - i)} \tag{102}$$

where $i,k = 0,\ldots,N_h-1$. Matrix **A** has a toeplitz structure and **b** is a vector. Matrix **C** also has a toeplitz structure and is defined as [4],

$$C_{i,k} = \frac{\varphi(k-i)\sin\left(\frac{\pi(k-i)}{D}\right)}{\pi(k-i)} \tag{103}$$

where,

$$\varphi(n) = D\sum_{t=-\infty}^{\infty}\delta(n-tD)-1 \tag{104}$$

Therefore, from (99) and (100), e_h can be written as,

$$e_h = \mathbf{h}^T(\mathbf{A}+\mathbf{C})\mathbf{h} - 2\mathbf{h}^T\mathbf{b} + 1 \tag{105}$$

Minimizing e_h in (105) with respect to $\mathbf{h}$, the required filter is derived as [4],

$$\mathbf{h} = (\mathbf{A}+\mathbf{C})^{-1}\mathbf{b} \tag{106}$$

Equation (106) is derived by minimizing the objective function (94). The objective function in (94) considers the pass-band response error and in-band aliasing distortion but does not specify the stop-band edge or the stop-band roll-off factor. A sharp cut-off of the analysis filter is desired for better performance of the adaptive filter. Hence, a constraint with a specified stop-band edge and the roll-off factor when added to the objective function will result in significant performance improvement. The stop-band roll-off constraint, χ_h can be defined as,

$$\chi_h = \frac{1}{2(\pi-\omega_s)}\left[\int_{-\pi}^{-\omega_s}\left|H\left(e^{j\omega}\right)\right|^2 d\omega + \int_{\omega_s}^{\pi}\left|H\left(e^{j\omega}\right)\right|^2 d\omega\right] \tag{107}$$

where, $H\left(e^{j\omega}\right)$ is the prototype analysis filter response and ω_s is the stop-band edge.

Therefore, the objective function for the analysis filter design can be re-defined as,

$$\hat{e}_h = \alpha_h + \beta_h + \gamma\chi_h \tag{108}$$

where, γ is the scalar weighting on $\boldsymbol{\lambda}$. Setting $\gamma > 1$, will result in producing a filter with more emphasis on stop-band roll-off factor than the pass-band response error and the in-band aliasing distortion. Now, the required filter $\mathbf{h}$ is derived using the objective function (108). The prototype analysis filter response $H(z)$ can be expressed in terms of its impulse response $h(n)$ as,

$$H(z) = \sum_{n=0}^{N_h-1} h(n) z^{-n} \tag{109}$$

Considering, $\mathbf{h} = \left[h_0, h_1, \cdots, h_{N_h-1}\right]^T$ and $\boldsymbol{\varphi}_{\mathbf{h}}(z) = \left[1, z^{-1}, \ldots, z^{-N_h+1}\right]^T$, $H(z)$ from (98) can be written as,

$$H\left(e^{j\omega}\right) = \mathbf{h}^T \boldsymbol{\varphi}_{\mathbf{h}}\left(e^{j\omega}\right) \tag{110}$$

$$\left|H\left(e^{j\omega}\right)\right|^2 = H\left(e^{j\omega}\right) H^*\left(e^{j\omega}\right) = \mathbf{h}^T \boldsymbol{\varphi}_{\mathbf{h}}\left(e^{j\omega}\right) \boldsymbol{\varphi}_{\mathbf{h}}^H\left(e^{j\omega}\right) \mathbf{h} \tag{111}$$

Consider,

$$\boldsymbol{\varphi}_{\mathbf{h}}\left(e^{j\omega}\right) \boldsymbol{\varphi}_{\mathbf{h}}^H\left(e^{j\omega}\right) = e^{-j\omega i} e^{j\omega k} = e^{-j\omega(i-k)} = \boldsymbol{\varphi}_{\mathbf{h}}^H(\omega) \tag{112}$$

where, $i, k = 0, \ldots, N_h - 1$. Applying (111) and (112) in (107), yields

$$\chi_h = \frac{1}{2(\pi - \omega_s)} \left[\int_{-\pi}^{-\omega_s} \mathbf{h}^T \boldsymbol{\varphi}_{\mathbf{h}}^H(\omega) \mathbf{h} d\omega + \int_{\omega_s}^{\pi} \mathbf{h}^T \boldsymbol{\varphi}_{\mathbf{h}}^H(\omega) \mathbf{h} d\omega \right] \tag{113}$$

Re-writing equation (113) in quadratic form, yields

$$\chi_h = \mathbf{h}^T \boldsymbol{\lambda} \mathbf{h} \tag{114}$$

where, $\boldsymbol{\lambda}$ is given as,

$$\boldsymbol{\lambda} = \frac{1}{2(\pi - \omega_s)} \left[\int_{-\pi}^{-\omega_s} \boldsymbol{\varphi}_{\mathbf{h}}^H(\omega) d\omega + \int_{\omega_s}^{\pi} \boldsymbol{\varphi}_{\mathbf{h}}^H(\omega) d\omega \right] \tag{115}$$

Substituting (112) in (115) yields,

$$\lambda_{i,k} = \frac{1}{2(\pi - \omega_s)}\left[\int_{-\pi}^{-\omega_s} e^{-j\omega(i-k)} d\omega + \int_{\omega_s}^{\pi} e^{-j\omega(i-k)} d\omega\right] \tag{116}$$

where, $i,k = 0,\ldots,N_h - 1$. Integrating equation (116), yields,

$$\lambda_{i,k} = \frac{1}{2(\pi - \omega_s)}\left[\frac{-1}{j(i-k)}\left(e^{j\omega_s(i-k)} - e^{j\pi(i-k)} + e^{-j\pi(i-k)} - e^{-j\omega_s(i-k)}\right)\right] \tag{117}$$

Simplifying (117), yields,

$$\lambda_{i,k} = \frac{1}{2(\pi - \omega_s)}\left[\frac{-1}{j(i-k)}\left(2j\sin\left(\omega_s(i-k)\right) - 2j\sin\left(\pi(i-k)\right)\right)\right] \tag{118}$$

From (116), it can be observed that for $i = k$, $\lambda_{i,k} = 1$. Therefore, $\boldsymbol{\lambda}$ is a toeplitz matrix defined as,

$$\lambda_{i,k} = \frac{\left(\sin\left(\pi(k-i)\right) - \sin\left(\omega_s(k-i)\right)\right)}{(\pi - \omega_s)(k-i)}, \textit{ for all } i \neq k$$
$$\lambda_{i,k} = 1, \textit{ for all } i = k \tag{119}$$

Substituting (114), (100) and (99) to (108) yields,

$$\hat{e}_h = \mathbf{h}^T(\mathbf{A} + \gamma\boldsymbol{\lambda} + \mathbf{C})\mathbf{h} - 2\mathbf{h}^T\mathbf{b} + 1 \tag{120}$$

Minimizing $\hat{e}_h$ with respect to $\mathbf{h}$, the required filter is derived as [4],

$$\frac{\partial \hat{e}_h}{\partial \mathbf{h}} = 2(\mathbf{A} + \gamma\boldsymbol{\lambda} + \mathbf{C})\mathbf{h} - 2\mathbf{b} = \mathbf{0} \tag{121}$$

$$\mathbf{h} = (\mathbf{A} + \gamma\boldsymbol{\lambda} + \mathbf{C})^{-1}\mathbf{b} \tag{122}$$

Comparing (122) to (106), one can observe that the stop band constraint, χ_h introduced the term $\boldsymbol{\lambda}$ in the filter design equation (106). Thus $\mathbf{h}$ in (122) defines the desired analysis filter.

The design of the synthesis filter is based on the analysis filter because the performance focus is on the analysis and synthesis filter bank as a whole [4,3]. Therefore, the objective function, e_f, of the synthesis filter is defined as the combination of the total response error, $\alpha_f(h)$ and residual aliasing distortion, $\beta_f(h)$. The objective function e_f is given as [4],

$$e_f(h) = \alpha_f(h) + v\beta_f(h) \tag{123}$$

where,

$$\alpha_f(h) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|A_0\left(e^{j\omega}\right) - e^{-j\omega\tau_T}\right|^2 d\omega \tag{124}$$

and,

$$\beta_f(h) = \frac{1}{2\pi}\sum_{d=1}^{D-1}\sum_{m=0}^{M-1}\int_{-\pi}^{\pi}\left|A_{m,d}\left(e^{j\omega}\right)\right|^2 d\omega \tag{125}$$

where, τ_T is the total delay of the analysis and synthesis filter bank, *M* is the number of sub-bands, and *D* is the number of distortion terms. The term $A_{m,d}\left(e^{j\omega}\right)$ defines the analysis and synthesis filter bank's transfer function as a whole [4] and,

$$A_0\left(e^{j\omega}\right) = \sum_{m=0}^{M-1} A_{m,d}\left(e^{j\omega}\right), \text{ where } d = 0 \tag{126}$$

$A_0\left(e^{j\omega}\right)$ is the transfer function of the analysis-synthesis system whose outcome is the desired reconstructed signal spectrum [4]. $A_{m,d}\left(e^{j\omega}\right)$, $d = 1,\ldots,D-1$ in (125) measures the total residual aliasing introduced in all the sub-bands [4]. The value *v*, is a scalar weighting factor for the synthesis filter. Now, similar to the analysis filter, the required synthesis filter can be derived as [4],

$$f = \left(\mathbf{E} + v\mathbf{P}\right)^{-1}\mathbf{g} \tag{127}$$

where, $i,k = 0,\ldots,N_f - 1$. N_f is the length of the synthesis filter. The matrix $\mathbf{E}$ and vector $\mathbf{g}$ are defined as [4],

$$E_{i,k} = \frac{M^2}{D^2}\sum_{t=-\infty}^{\infty} h^*\left(kM - i\right)h\left(tM - k\right) \tag{128}$$

$$g_i = \frac{M}{D}h\left(\tau_{\mathrm{T}} - i\right) \tag{129}$$

The matrix $\mathbf{P}$ is defined as [4],

$$P_{i,k} = \frac{M}{D^2}\sum_{l=-\infty}^{\infty} h^*\left(l + k\right)h\left(l + i\right)\varphi\left(i - k\right) \tag{130}$$

$$\varphi\left(n\right) = D\sum_{t=-\infty}^{\infty}\delta\left(n - tD\right) - 1 \tag{131}$$

With reference to the filter design technique discussed above, a few of the parameters are chosen by outside design considerations. Typically, these include the number of sub-bands, *M*, and the sub-sampling factor, *R*. These parameters influence both the performance and complexity of the overall filterbank. Section 2.3 discusses the selection criteria for *M* and *R*. The other adjustable parameters are as follows [4,3],

- The length of the analysis filter, N_h
- The length of the synthesis filter, N_f
- The passband edge frequency, ω_p
- The stopband edge frequency, ω_s
- The weighting of synthesis filter design, *v*
- The delay or latency of the analysis filter, τ_h
- The total delay or latency of analysis and synthesis filter-bank, τ_T
- The stop-band constraint weighting in the analysis filter, γ

The total latency, τ_T introduced by both analysis and synthesis filter banks is usually determined prior to the design the filter bank. The analysis portion of the total delay, τ_h, is then adjusted for optimal performance. Generally τ_h is chosen to be

around $0.5\tau_T$. However, in this thesis τ_h is chosen by the optimization algorithm considering a pre-defined τ_T. The pass and stop band frequencies ω_p and ω_s determine the bandwidth of the filters and influence the total filter response error and residual alias distortion of the system [4]. The parameter, γ is a scalar which defines the weighting on the stop-band constraint used for generating the analysis filter. The higher the value of γ, the greater the emphasis on the stop-band roll off factor. The weighting parameter, *v* is used in the synthesis filter bank design to trade-off the importance of the filter bank response error and the residual aliasing distortion [4]. When $v > 1$, the design emphasis on reducing the residual aliasing distortion is greater than the total response error. In greater general total response error can be tolerated when compared to the residual aliasing distortion. This is because aliasing affects the sub-band adaptive filter's adaptation rate and depth. As a result, the parameters *v*, γ, ω_p, ω_s, τ_h, N_h and N_f have a direct impact on the sub-band adaptive filter's performance (convergence). Choosing the right values for these parameters for optimal adaptive filter performance is a very tedious and time consuming job. Hence, this thesis proposes an offline computational intelligence optimization approach for the selection of the parameter values.

The performance of the adaptive filter can be measured in terms of the echo return loss enhancement, ERLE. ERLE is the measure of the attenuation of the echo in the send path of the echo canceller.

5.2. COMPUTATIONAL INTELLIGENCE APPROACH FOR FILTER DESIGN PARAMETERS' SELECTION

The optimization system is shown in Figure 5.1. The particles in DEPSO-gbest which correspond to the synaptic weights of the ANN are randomly initialized. The NN is provided with a constant input which acts as a fixed energy source. Each set of synaptic weights or particle produces an output of the NN which is a set of parameters for the design of the analysis and synthesis filter. These parameters used as inputs to the sub-band analysis/sub-band filter bank design procedure of the previous section. The resulting filter banks are then used in the sub-band adaptive filter for performance evaluation. The performance of the sub-band adaptive filter is measured in terms of the fitness function

which is a weighted combination of the ERLE of the sub-band adaptive filter and the ripple measure of the analysis-synthesis filter bank. Based on this fitness measure, DEPSO optimizes the weights of the NN to generate parameters with better fitness measure.

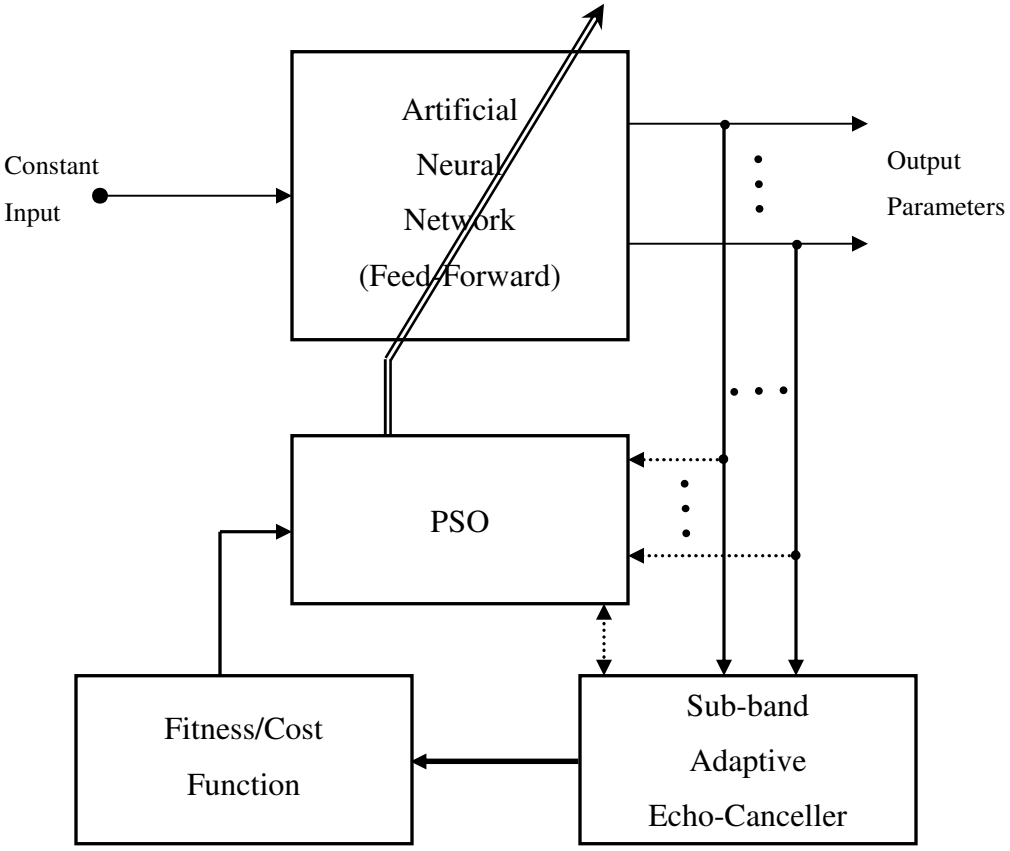

Figure 5.1. Parameter Optimization System.

A feed-forward NN consisting of sixteen neurons distributed in three layers is utilized. There are four in the input layer is four, five in the hidden layer, and seven in the output layer. The hidden layer neuron use sigmoid functions in the output where the input and the output layer neurons use linear functions. Each parameter has a different magnitude range and hence, the outputs of NN are subjected to the range constraints as discussed in Section 4.2.2. A population of thirty particles is used to train the NN. The sub-band echo canceller is tested for ERLE using a white noise excitation signal.

5.2.1. Cost Function. Both the ERLE and ripple measures are recorded in decibels and combined to develop the fitness function,

$$J = r_{dB} - \alpha \sum_{i=0}^{2} erle(i) \tag{132}$$

where, r_{dB} is the ripple measure of the analysis-synthesis filter bank's response (reconstructed signal) and $erle(i)$ is a set of three ERLE measurements recorded during the adaptive filter's adaptation period. The weighting constant α is used to weight the emphasis of the optimization between r_{dB} and ERLE.

In order to measure r_{dB}, a rectangular pulse of unit magnitude is given as input to the analysis-synthesis filter bank and the output is evaluated for the reconstruction ability. The synthesis filter's output looks like a delayed version of the input but with distorted pulse transition and ripples in the unit magnitude region. The amount of ripple in the unit magnitude region is measured in decibels it is defined as,

$$r_{dB} = 20\log_{10}\left(\frac{\max(\mathbf{r}) - \min(\mathbf{r})}{avg(\mathbf{r})}\right) \tag{133}$$

where, $\mathbf{r}$ is a vector which is the output of synthesis filter bank. The vector $\mathbf{r}$ records the reconstructed signal constituting only the required unit magnitude region. The function $\max(\mathbf{r})$ finds the maximum value and the function $\min(\mathbf{r})$ finds the minimum value in $\mathbf{r}$. $avg(\mathbf{r})$ is be defined as,

$$avg(\mathbf{r}) = \frac{1}{T}\sum_{n=0}^{T-1} r(n) \tag{134}$$

where, T is the length of vector $\mathbf{r}$. The first sample time and the length in the vector $\mathbf{r}$ is user defined.

ERLE is measured in three equally spaced time segments during the filter adaptation. The time segments are generally measured near the final convergence. For simulations, ERLE is measured at 0.5,1 and 1.5 secs. ERLE can be defined as,

$$erle = 10\log_{10}\sum_{n=1}^{MR}\hat{s}^2(n) - 10\log_{10}\sum_{n=1}^{MR}e^2(n) \tag{135}$$

where, $\hat{s}(n)$ is the pure echo signal without the near-end signal and near-end background noise. $e(n)$ is the residual echo in the send path of the echo canceller. *MR* defines the size of the data segment considered for ERLE measurement.

5.2.2. Constraints. Two types of constraints can be given to the optimization system such that the dimension of the search space is reduced. This helps the optimization system to search for a better solution faster.

Boundary conditions for the particles in the DEPSO algorithm can be imposed to constrain the solution to reasonable values. DEPSO optimizes the weights of the NN as the particles fly towards the optimal solution in a multi-dimensional search space at different velocities. Boundary conditions to the DEPSO algorithm are applied by limiting the maximum allowed velocity to each particle and limiting the weight values of the NN to a certain range. The boundary conditions are important for DEPSO to avoid any divergence of the particles from the solution and also helps find the solution faster. For simulations, the maximum allowed velocity of each particle is set to 0.5 and the NN weights are allowed to take values between ± 0.5. The above said boundaries for the DEPSO algorithm were chosen manually using a trial and error approach [19]. The simulations show that 100 epochs were good enough to find an optimal set of parameters.

Secondly, the output parameters can be restricted to a certain range of values using prior knowledge of the parameters. The dependence/relation between the parameters can also be exploited to put constraints on the search space. Limiting the range of the parameters reduce the multi-dimensional search space and hence, assists the optimization system in finding the required solution faster. The total delay, τ_{T} of the system can always be assumed to be more than the analysis filter bank delay, τ_h. The length of the analysis and synthesis filter bank can be limited to a certain range based on the processing band-width provided to the sub-band adaptive filter. Utilizing prior information, the parameters are restricted to a certain range of values as tabulated in

Table 5.1. Consider *M* as the number of sub-bands, *R* as the sub-sampling factor and f_s as the sampling frequency. For simulations τ_T is pre-defined to either M/f_s or $2M/f_s$ secs.

Table 5.1. Range Constraints on the Parameters.

Parameter	Range Limits
N_h	$(4\times R) < N_h < (16\times R)$ N_h can take only integer values
N_f	$(4\times R) < N_f < (16\times R)$ N_f can take only integer values
τ_h	$\tau_h < \tau_T$
ω_p	$\frac{0.6\pi}{M} < \omega_p < \frac{2\pi}{M}$ *rad/sample*
ω_s	$\frac{0.6\pi}{M} < \omega_s < \frac{2\pi}{M}$ *rad/sample*
v	$1 < v < 2000$
γ	$4000 < \gamma < 500000$

5.3. SIMULATIONS

For simulations an echo path characterized by a 20th order low pass elliptic filter is used. Figure 5.2 shows the response of the echo path.

Figure 5.3 shows the convergence curve of the proposed optimization system which used a $4\times5\times7$ feed forward neural network along with gbest version of DEPSO. The DEPSO-gbest version was populated with 30 particles. The other parameters were set as, $w = 0.8$, $C_1 = 2$, $C_2 = 2$, the maximum velocity of each particle $vmax = 0.5$ and the NN weights were allowed to be between ± 0.5. The sub-band echo canceller was set to $M = 32$, $R = 16$ and $\tau_T = 8ms$. A white noise excitation signal is used and a 20th order low pass elliptic filter with the pass-band cut-off at 0.5π characterized the echo path. The

sampling frequency, f_s of the excitation signal is 8kHz. No background noise is added to the Line signal. Table 5.2 tabulates the filter design parameters identified by the NN-DEPSO optimization system. The designed analysis and synthesis filter bank response has a ripple, r_{dB} = -56dB. The ripple was measured between the time 70ms and 150ms. Figure 5.4 shows the response of the ideal and designed analysis and synthesis filter bank's response. $\alpha = 0.1$ is used in cost function (135).The Figure 5.5 shows the cancellation of echo with respect to the number of input samples. Figure 5.6 shows the ERLE performance plot with respect to time. The ERLE was measured considering 64ms data segment. The ERLE at 2$secs$ is around 38dB.

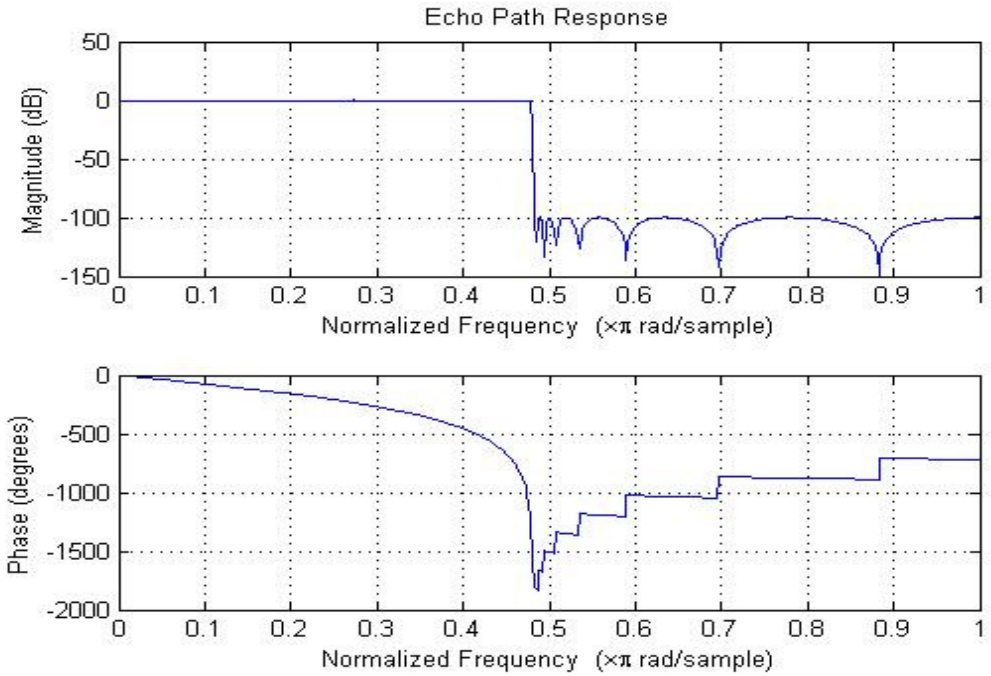

Figure 5.2. Response of the Echo Path Channel.

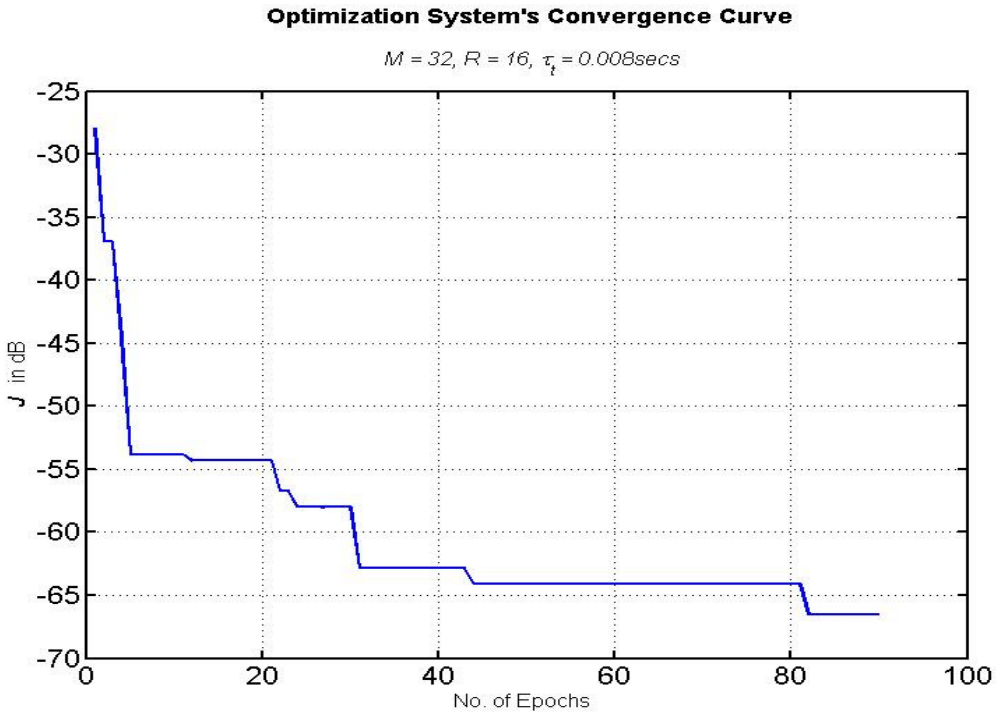

Figure 5.3. DEPSO-NN Convergence Curve, M=32, R=16, $\tau_T = 8ms$.

Table 5.2. Parameters Identified by the Optimization System, M=32, R=16, $\tau_T = 8ms$.

Parameter	Identified Values
N_h	240
N_f	80
τ_h	1.9848ms
ω_p	0.045π rad/sample
ω_s	0.0625π rad/sample
v	1916.7
γ	17220

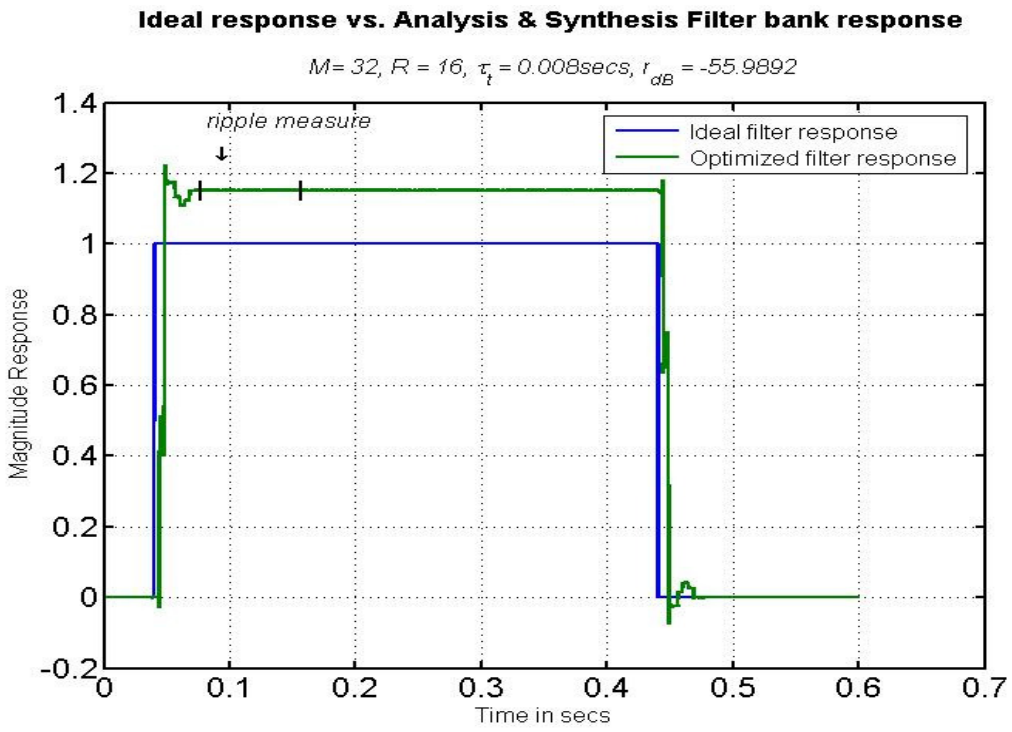

Figure 5.4. Analysis & Synthesis Filter Bank Response, M=32, R=16, $\tau_T = 8ms$.

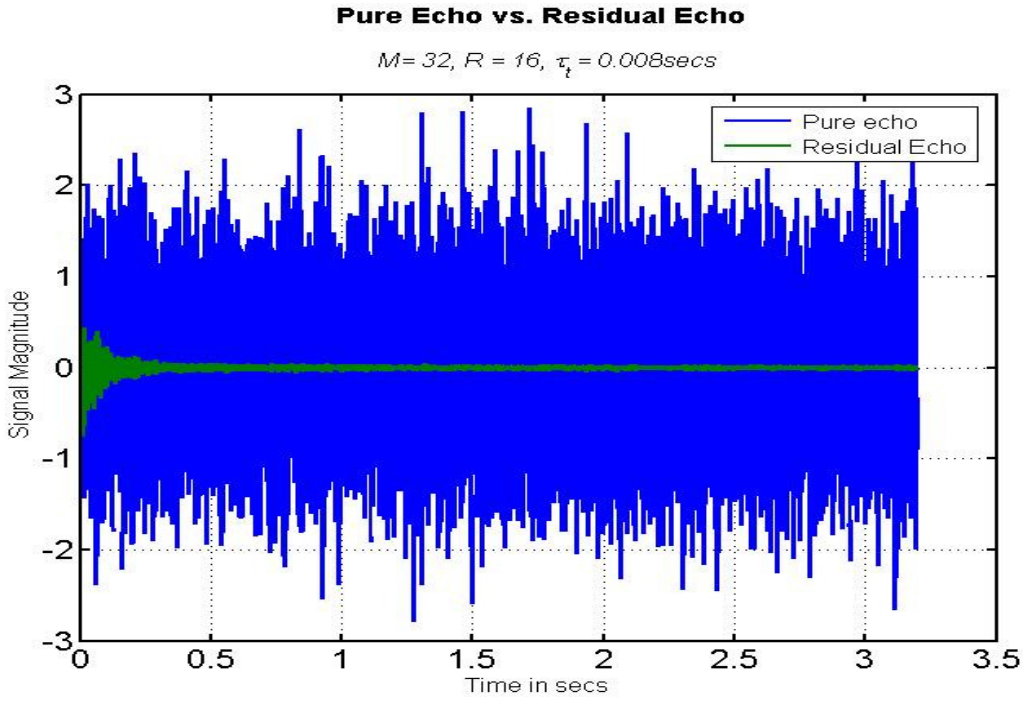

Figure 5.5. Pure Echo vs. Residual Echo, M=32, R=16, $\tau_T = 8ms$.

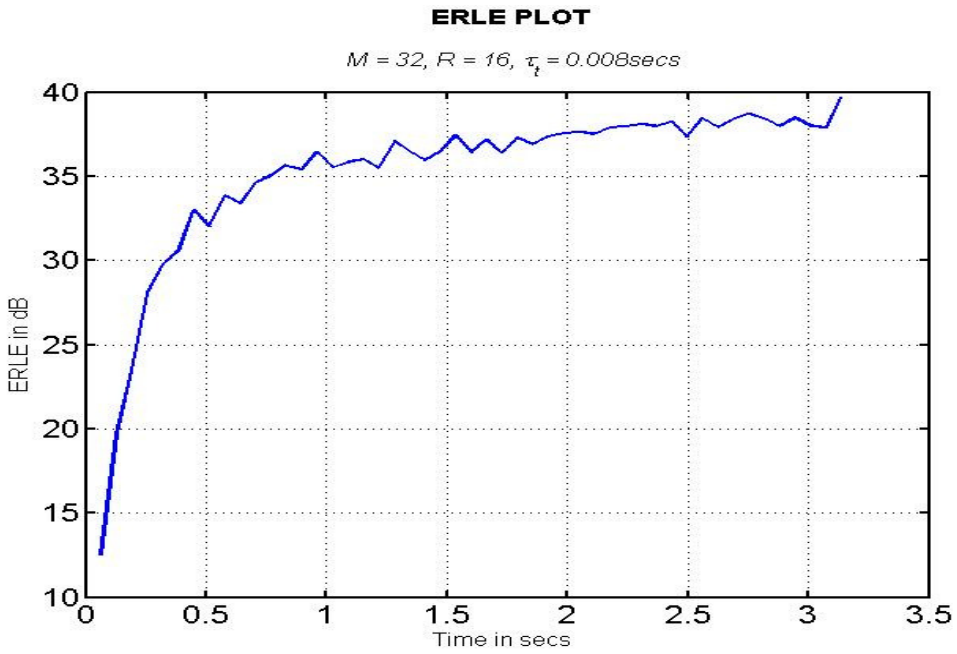

Figure 5.6. ERLE Plot, M=32, R=16, $\tau_T = 8ms$.

Figure 5.7 shows the convergence curve of the proposed optimization system which used a $4\times5\times7$ feed forward neural network along with gbest version of DEPSO. The DEPSO-gbest version was populated with 30 particles. The other parameters were set as, $w = 0.8$, $C_1 = 2$, $C_2 = 2$, the maximum velocity of each particle $vmax = 0.5$ and the NN weights were allowed to be between ± 0.5. The sub-band echo canceller was set to $M = 32$, $R = 16$ and $\tau_T = 4ms$. A white noise excitation signal is used and a 20^{th} order low pass elliptic filter with the pass-band cut-off at 0.5π characterized the echo path. The sampling frequency, f_s of the excitation signal is $8kHz$. No background noise is added to the Line signal. Table 5.3 tabulates the filter design parameters identified by the NN-DEPSO optimization system. The designed analysis and synthesis filter bank response had a ripple, r_{dB} = -45dB. The ripple was measured between the time 63ms and 150ms. Figure 5.8 shows the response of the ideal and designed analysis and synthesis filter bank's response. $\alpha = 0.1$ is used in cost function (135). The Figure 5.9 shows the cancellation of echo with respect to the number of input samples. Figure 5.10 shows the

ERLE performance plot with respect to time. The ERLE was measured considering 64*ms* data segment. The ERLE at 2*secs* is around 32*dB*.

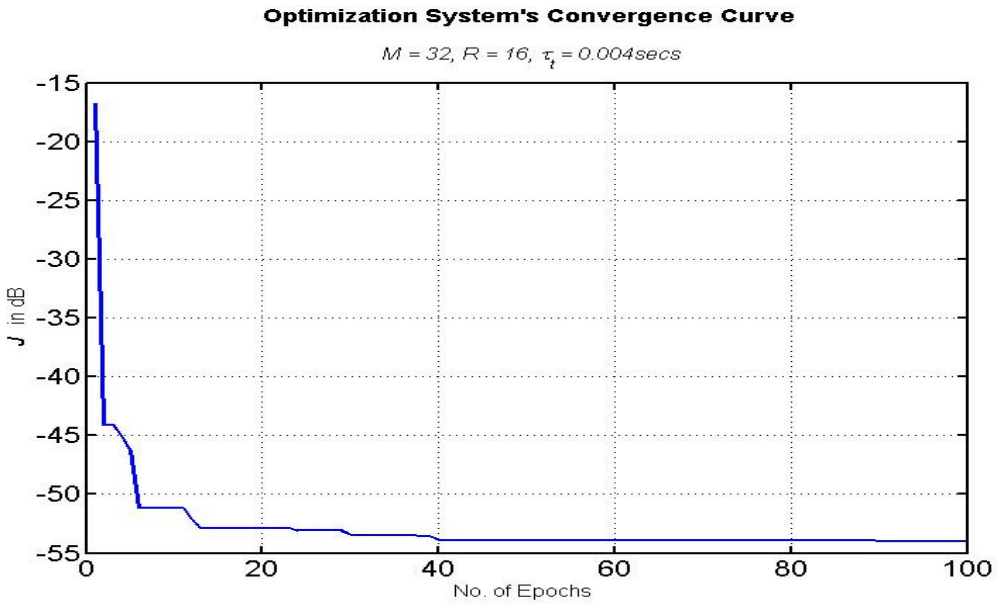

Figure 5.7. DEPSO-NN Convergence Curve, *M*=32, *R*=16, $\tau_T = 4ms$.

Table 5.3. Parameters Identified by the Optimization System, *M*=32, *R*=16, $\tau_T = 4ms$.

Parameter	Identified Values
N_h	64
N_f	96
τ_h	2.9296*ms*
ω_p	0.01875π *rad/sample*
ω_s	0.0305625π *rad/sample*
v	2000
γ	4000

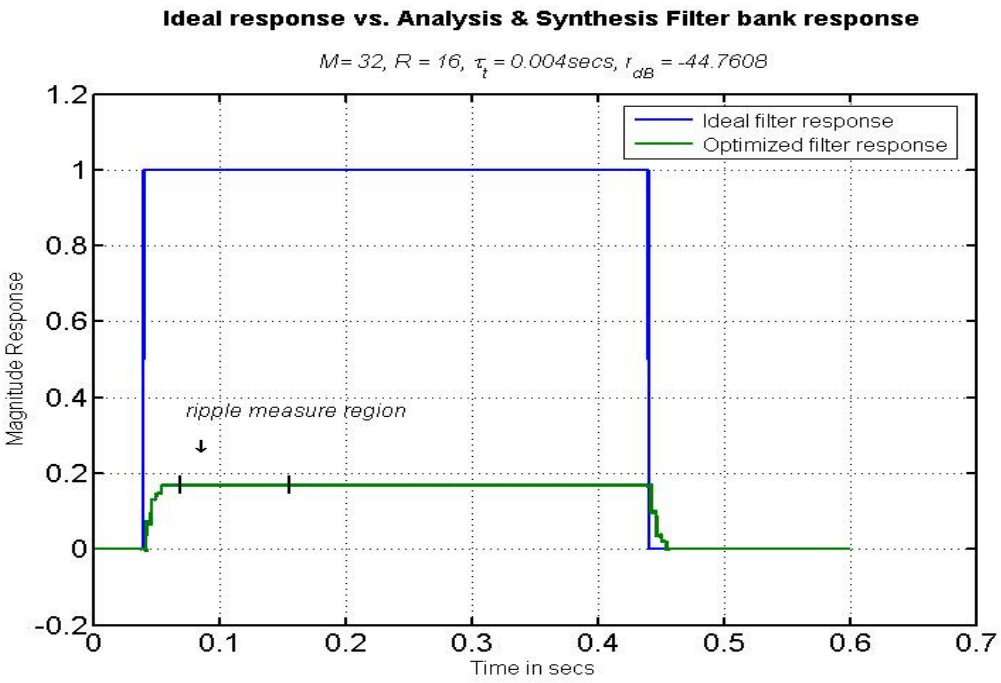

Figure 5.8. Analysis & Synthesis Filter Bank Response, M=32, R=16, $\tau_T = 4ms$.

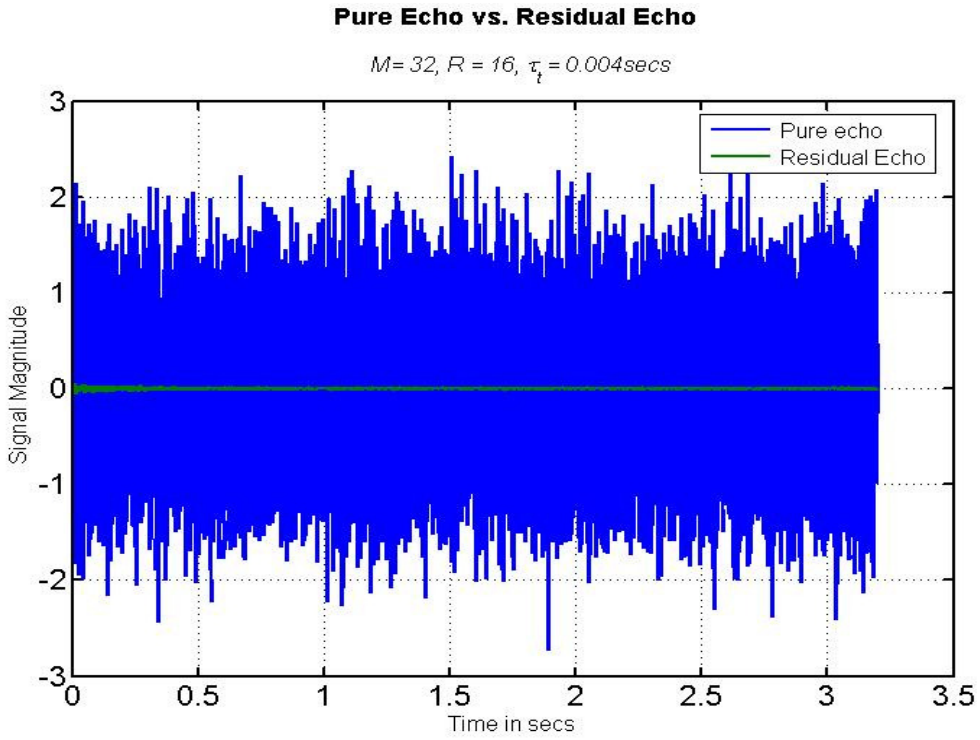

Figure 5.9. Pure Echo vs. Residual Echo, M=32, R=16, $\tau_T = 4ms$.

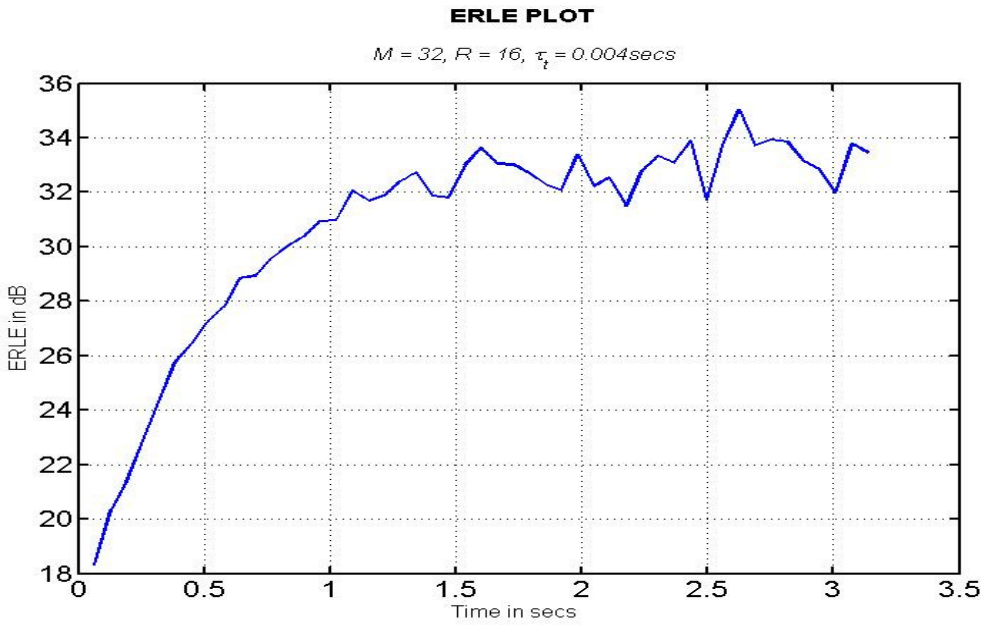

Figure 5.10. ERLE Plot, *M*=32, *R*=16, $\tau_T = 4ms$.

Figure 5.11 shows the convergence curve of the proposed optimization system which used a $4\times5\times7$ feed forward neural network along with gbest version of DEPSO. The DEPSO-gbest version was populated with 30 particles. The other parameters were set as, $w = 0.8$, $C_1 = 2$, $C_2 = 2$, the maximum velocity of each particle $vmax = 0.5$ and the NN weights were allowed to be between ± 0.5. The sub-band echo canceller was set to $M = 16$, $R = 8$ and $\tau_T = 4ms = \frac{2M}{f_s}$. A white noise excitation signal is used and a 20th order low pass elliptic filter with the pass-band cut-off at 0.5π characterized the echo path. The sampling frequency, f_s of the excitation signal is 8*kHz*. No background noise is added to the Line signal. Table 5.4 tabulates the filter design parameters identified by the NN-DEPSO optimization system. The designed analysis and synthesis filter bank response had a ripple, r_{dB} = -36*dB*. The ripple was measured between the time 33*ms* and 150*ms*. Figure 5.12 shows the response of the ideal and designed analysis and synthesis filter bank's response. $\alpha = 2$ is used in cost function (135). The Figure 5.13 shows the cancellation of echo with respect to the number of input samples. Figure 5.14 shows the

ERLE performance plot with respect to time. The ERLE was measured considering 64*ms* data segment. The ERLE at 2*secs* is around 36*dB*.

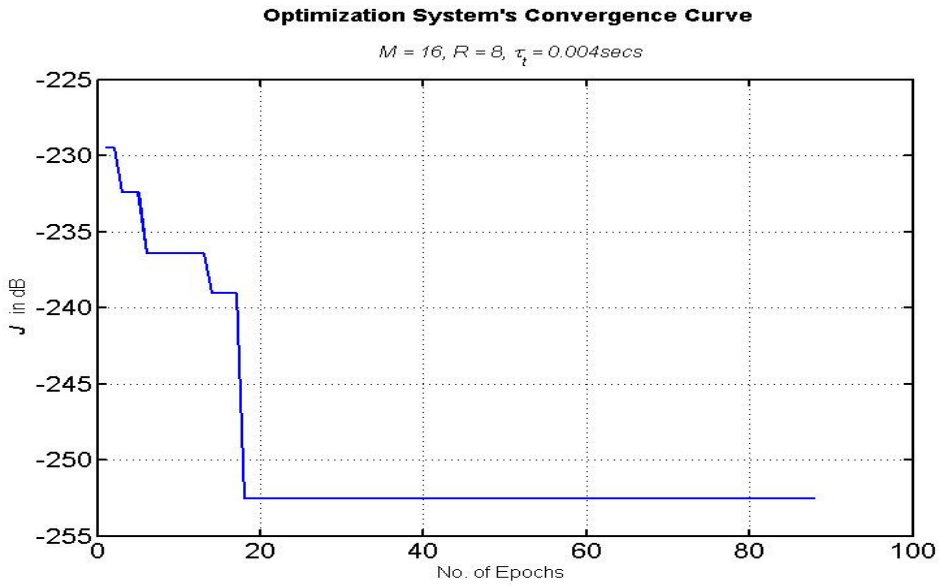

Figure 5.11. DEPSO-NN Convergence Curve, M=16, R=8, $\tau_T = 4ms$.

Table 5.4. Parameters Identified by the Optimization System, M=16, R=8, $\tau_T = 4ms$.

Parameter	Identified Values
N_h	88
N_f	32
τ_h	0ms
ω_p	0.115378π rad/sample
ω_s	0.0375π rad/sample
v	2000
γ	3084.9

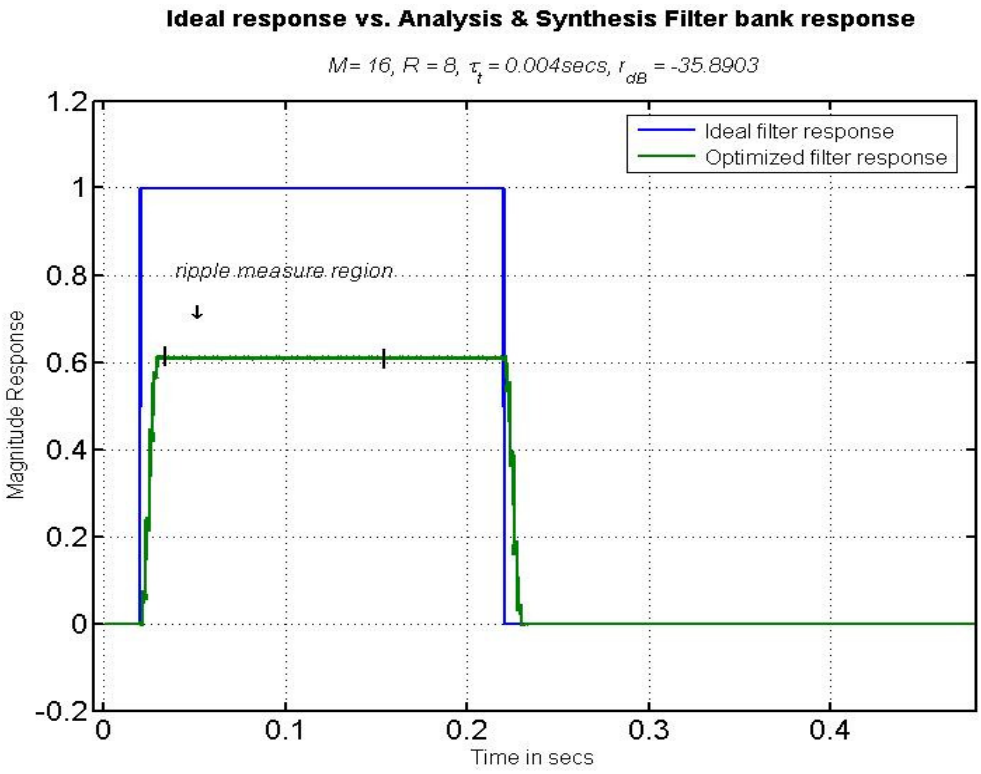

Figure 5.12. Analysis & Synthesis Filter Bank Response, M=16, R=8, $\tau_T = 4ms$.

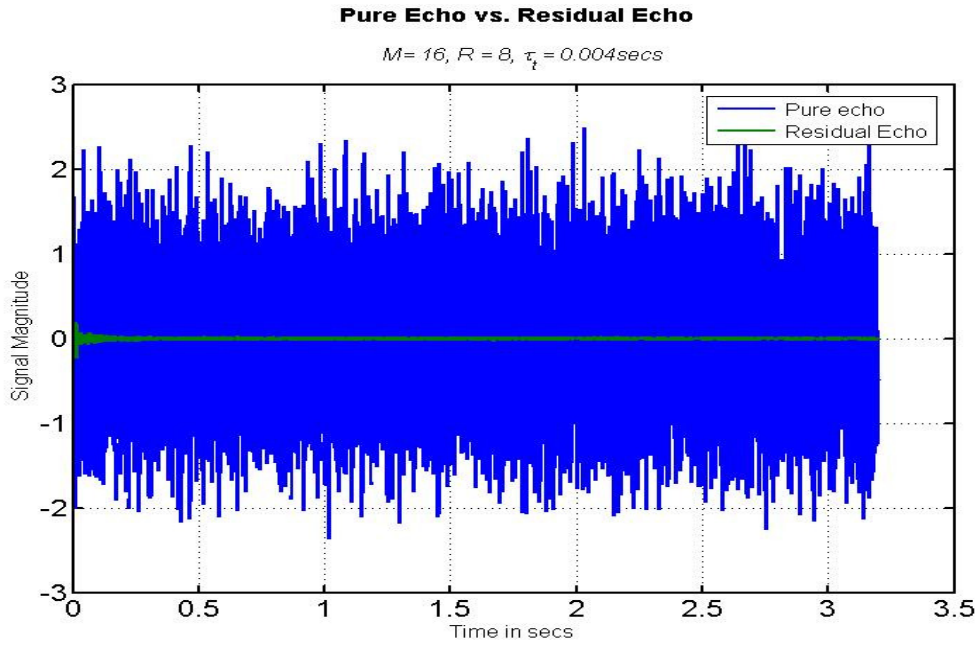

Figure 5.13. Pure Echo vs. Residual Echo, M=32, R=16, $\tau_T = 4ms$.

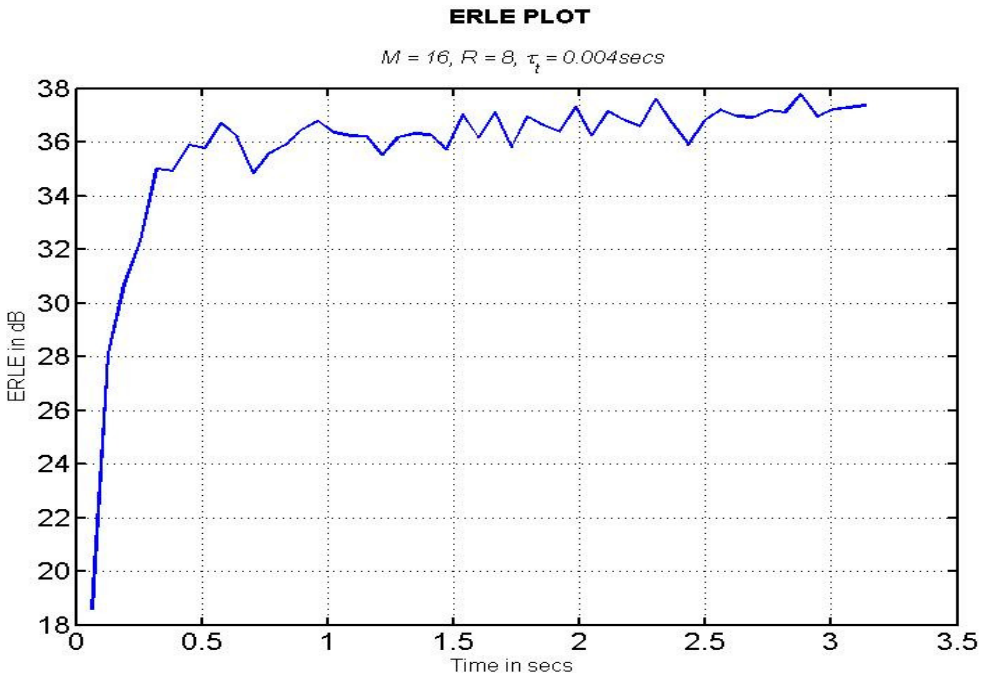

Figure 5.14. ERLE Plot, M=16, R=8, $\tau_T = 4ms$.

6. CONCLUSION

This thesis work introduced a variable regularization method for the fast affine projection algorithm (VR-FAP). The regularization parameter varies as a function of the excitation, measurement noise, and residual error energies. Because of the dependence on the last parameter, VR-FAP demonstrates the desirable property of fast initial convergence and deep convergence (low misadjustment)/immunity to measurement noise. Conventionally, FAP's regularization is only set at initialization, to overcome this, noise-injection technique is used. Noise in the filter update is injected with the noise-power proportional to the variable regularization parameter. As with their fixed regularization versions, VR-FAP is considerably less complex than VR-APA and simulations verify that they both have similar convergence properties for white noise and colored noise excitations. The VR-FAP performs well even with the speech excitation signal.

This thesis work introduced a computational intelligence approach for optimizing the filter design parameters in order to improve the performance of the sub-band echo canceller. A Hybrid Artificial Intelligence system consisting of Artificial Neural Network and gbest version of DEPSO was utilized to find the optimal filter parameters. Simulations were performed to identify filter design parameters for a combination of different number of sub-bands, sub-sampling factor and total filter delay, using the proposed optimization system. The results show that the sub-band adaptive filter was able to comfortably cancel the echo by $30dB$. Therefore, the proposed parameter optimization system can be used to improve the convergence of the adaptive filters.

BIBLIOGRAPHY

[1] J. Benesty, T.Gänsler, D.R. Morgan, M.M. Sondhi, S. L. Gay, *Advances in Network and Acoustic Echo Cancellation*, Springer-Verlag Berlin Heidelberg 2001.

[2] S.L. Gay, "Fast Projection Algorithms with Application to Voice Excited Echo Cancellers," *Ph.D. Dissertation,* Rutgers University, Piscataway, New Jersey, October, 1994.

[3] RE Crochiere and LR Rabiner, "*Multirate Digital Signal Processing*," Chapter 7, pp. 289-404, Prentice-Hall, Englewood Cliffs, NJ, USA, 1983.

[4] JM de Haan, N Grbic, I Claesson, and SE Nordholm, "Filter bank design for subband adaptive microphone arrays," in *IEEE Trans. On Speech and Audio Processing*, vol. 11, no. 1, Jan. 2003.

[5] Witold Pedrycz, "*Computational Intelligence: An Introduction*," Boca Raton, Fla.: CRC Press, 1998.

[6] J. Kennedy, R. Eberhart, "Particle Swarm Optimization," in *Proceedings of the IEEE Trans. Neural Network*, vol. 4, pp. 1942-1948, 1995.

[7] Wen-Jun Zhang, Xiao-Feng Xie, "DEPSO: Hybrid Particle Swarm with Differential Evolution Operator," in *Proceedings of the IEEE Trans. Systems, Man and Cybernetics*, vol. 4, pp. 3816-3821, 2003.

[8] S. Haykin, "*Adaptive Filter Theory*," 3rd ed., Upper Saddle River, NJ: Prentice Hall, 2002.

[9] S.L. Gay, "Affine Projection Algorithms," in *Least-Mean-Square Adaptive Filters*, S. Haykin, B. Widrow, Eds., John Wiley & sons Inc., Chapter 7, pp. 241-291, 2003.

[10] S.L. Gay, S. Tavathia, "The Fast Affine Projection Algorithm," in *Proceedings of the IEEE Trans. Acoust.*, *Speech, Signal Process.,* vol.5, pp. 3023-3026, May 1995.

[11] S. L. Gay, J. Benesty, "*Acoustic Signal Processing For Telecommunication*," Massachusetts, Kluwer Academic Publishers, 2000.

[12] S. L. Gay, "A Fast Converging, Low Complexity Adaptive Filtering Algorithm," in *IEEE workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 4-7, 1993.

[13] Steven L. Gay, "Dynamically regularized Fast RLS with application to Echo Cancellation," in *Proc. IEEE ICASSP*, pp. 957–960, 1996.

[14] J.M. Cioffi, T. Kailath, "Windowed Fast Transversal Filters Adaptive Algorithms with Normalization," *IEEE Trans. Acoust.*, *Speech, Signal Process.,* Vol. ASSP-33, No. 3, June 1985.

[15] K. Ozeki, T. Umeda, "An Adaptive Filtering Algorithm Using an Orthogonal Projection to an Affine Subspace and Its Properties," Electron. Commun. Japan., vol.67-A, no.5, pp.19-27, May 1984.

[16] M. Tanaka, Y. Kaneda, S. Makino, J. Kojima, "Fast Projection Algorithm and Its Step Size Control," in *Proceedings of the IEEE Trans. Acoust.*, *Speech, Signal Process.,* vol.2, pp. 945-948, May 1995.

[17] H. Rey, L.R. Vega, S. Tressens, J. Benesty, "Optimum Variable Explicit Regularized Affine Projection Algorithm," in *Proceedings of the IEEE Trans. Acoust.*, *Speech, Signal Process.,* vol.3, pp. 197-200, May 2006.

[18] Deepak Challa, Steven L. Grant, Asif Mohammad, "Variable Regularized Fast Affine Projections," in *IEEE Conference On Audio, Speech and Signal Processing*, April 2007.

[19] Venu G. Gudise, Ganesh K. Venayagamoorthy, "*Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks*," in *Proceedings of the IEEE Trans. Swarm Intelligence Symposium*, pp. 110-117, 2003.

[20] http://www.swarmintelligence.org/tutorials.php PSO tutorial (July 19th 2007).

VITA

Deepak Kumar Challa was born on 17th September, 1982 in Hyderabad, a city in the southern part of India. He received his Bachelors in Electronics and Communications Engineering from Siddaganga Institute of Technology, Tumkur, India in July 2003. He worked as an intern at Dandamizz Research Group, Bangalore, India on GSM technologies during his bachelors program. He worked for two years on Audio, Video Codecs' implementation on VLSI platforms at Future Techno Designs Ltd., Bangalore, India and Conexant Systems India Pvt. Ltd., Hyderabad, India. He joined the University of Missouri Rolla in August 2005 for the masters program specializing in Digital Signal Processing and Communications Engineering. He was involved in research with Dr. Steven L. Grant on Echo Cancellation and Blind Source Separation techniques during his masters study. He also served as an Intern in the Audio Systems team at Qualcomm Inc., CA, U.S.A. He received his Master of Science Degree in Electrical Engineering in December 2007. His current research interests include Echo Cancellation, Adaptive Filtering, Blind Source Separation and Audio/Video Technologies.