國立交通大學

電 子 工 程 系

碩 士 論 文

使用減低複雜度的重新取樣架構於不對等接收與傳送速率系統中之可適性訊號處理

Adaptive signal processing for mismatched rate systems by using a reduced complexity re-sampling architecture

研 究 生：陳毓成

指導教授：桑梓賢　教授

中 華 民 國 九 十 四 年 六 月

使用減低複雜度的重新取樣架構於不對等接收與傳送速率系統中之可適性訊號處理

Adaptive signal processing for mismatched rate systems by using a reduced complexity re-sampling architecture

研 究 生：陳 毓 成　　Student：Yu-Cheng Chen

指導教授：桑 梓 賢　　Advisor：Tzu-Hsien Sang

國 立 交 通 大 學

電 子 工 程 學 系

碩 士 論 文

A Thesis

Submitted to Department of Electronics Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electronics Engineering

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

使用減低複雜度的重新取樣架構於不對等接收與傳送速率系統中之可適性訊號處理

研究生：陳毓成　　　　　　　　　　　指導教授：桑梓賢 博士

國立交通大學電子工程學系(研究所)碩士班

摘　　要

在不對等接收與傳送速率的系統中，並使用一減低複雜度的架構下，我們提出一些有效的可適性訊號處理演算法以減低因重新取樣後而增加的運算量。使用這個減低複雜度架構的 LMS 演算法已經在[1]中被討論過。而在這篇論文中我們將會分析其收斂行為。另外一方面，當使用不理想的重新取樣濾波器而衍生出來的實際問題也會在這篇論文中討論。最後，我們將會使用模擬的結果來描述所討論的問題以及驗證這些演算法的效能。

Adaptive signal processing for mismatched rate systems by using a reduced complexity re-sampling architecture

student: *Yu-Cheng Chen* Advisor: Dr. *Tzu-Hsien Sang*

Department (Institute) of Electronics Engineering

National Chiao Tung University

ABSTRACT

Based on a reduced-complexity structure for mismatched rate adaptive signal processing, we present several efficient adaptive algorithms to reduce the computation cost due to the increasing amount of data after the re-sampling block. LMS algorithm using this reduced-complexity structure has been investigated in [1] and we shall give an analysis of its convergence behavior. On the other hand, practical issues such as problems introduced by imperfect re-sampling blocks will also be discussed. Furthermore, simulation results will be shown to verify the performance of the algorithms and illustrate our problems in discussion.

誌　　　謝

首先我要感謝我的指導教授-桑梓賢老師，這兩年來除了給我論文方面許多重要的啟發之外，從老師身上也學到了不少待人處世的道理。雖然在口試過了之後老師仍盡善盡美的要求我們要將演算法實現到 DSP 板上，以致於讓我到工作前最後的假期泡湯了，可是我還是感謝他，謝謝他讓我學到更多寶貴的經驗。

接下來我要感謝我的好同學們，景大、啟仁、勝毅、欣德、毓堂、毛律、弘安、小科，兩年來沒有你們一起打拼，我現在不會順利的在這邊寫我的論文致謝；還要感謝我的女朋友雅靜，謝謝妳在我失意的時候傾聽我所有的抱怨，謝謝妳體諒我因為研究而對妳的冷落。研究所能夠認識你們這些朋友實在是件很幸福的事，大家要繼續保持連絡啊！

最後我要感謝婆婆跟老爸老媽，謝謝他們從小到大對我無怨無悔的照顧與付出。

毓成　94.7.於颱風天下午的窗前

CONTENT

LIST OF TABLES

LIST OF FIGURES

Chapter1　Introduction

Adaptive linear filters have been successfully used in areas such as modeling of unknown systems, linear prediction, adaptive noise canceling, channel equalization systems with high-speed digital communication, echo cancellation, and in many other applications. Applying these adaptive signal processing algorithms to systems with identical transmit rate, say, ***R_tx*** samples per sec, and receive rate, say, ***R_rx*** samples per sec, is straight forward. However, if a system is with unequal transmit and receive rate, there must exist a rate matching function that accommodates the receiving signal to the filtered signal operating under the same rate for correct data processing. Such mismatched transmit and receive rate scheme can be seen, for example, the echo cancellation filter in the recent applications such as ADSL and VDSL systems. Figure 1.1 shows the conventional rate matching block diagram.

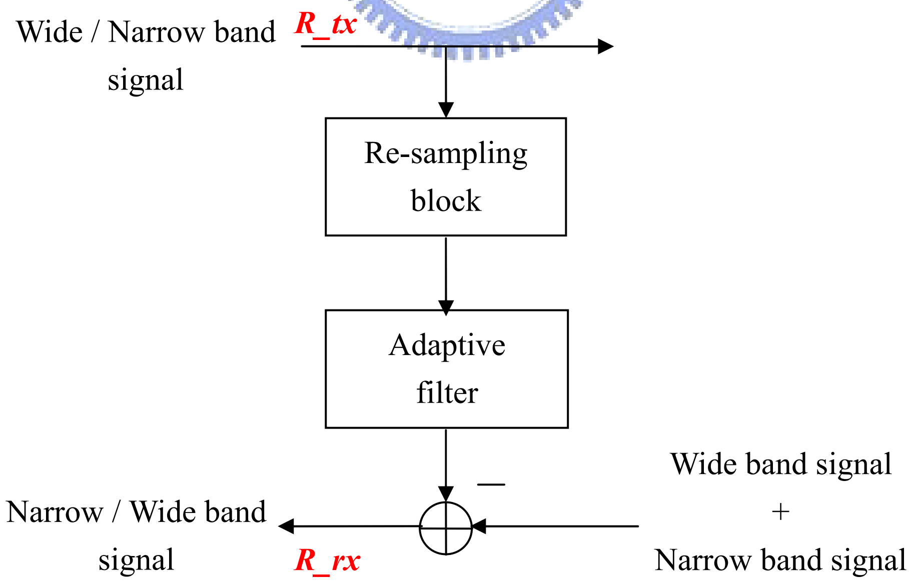
Wide / Narrow band signal
R_tx
Re-sampling block
Adaptive filter
—
Wide band signal + Narrow band signal
Narrow / Wide band signal
R_rx

Figure.1.1 Conventional rate matching structure. ***R_rx ≠R_tx.***

There are two situations, one is the higher transmission rate than the receive rate, and the other is the reverse. For the former, conventional configuration using algorithms such as sub-band adaptive filtering or simply down-sampling the transmitted signal can handle that well. Here, we will not consider that further. But for the latter (***R_rx*** > ***R_tx***), the adaptive filter must operate at a higher receive rate which results in an inefficient design. Such configuration shown in figure 1.1 with higher receive rate will increase the length of the filter proportional to ***R_rx/R_tx*** and then will quadratic increase the computation complexity of the following adaptive signal processing. For a long system path such as acoustic echo in telephone communication system, such increasing may be inhibited.

Alper et al. [1] have proposed an alternative structure shown in Figure 1.2.

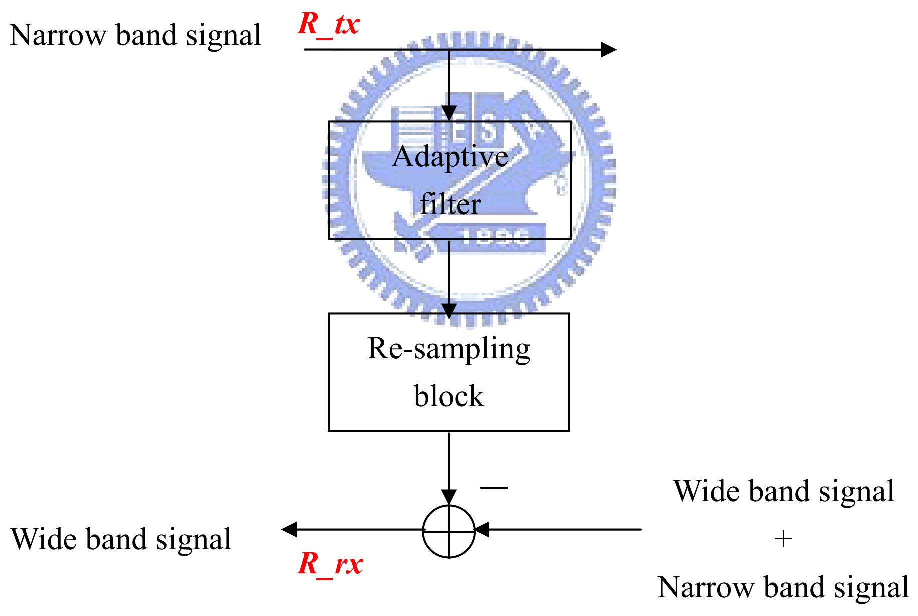

Figure 1.2 Reduced complexity rate matching structure. ***R_rx*** > ***R_tx***.

The configuration reverses the order of the re-sampling block and the filter in the conventional structure. In this structure, the adaptive filter is able to operate at the lower transmit rate and thus yields a more efficient design.

1.1: Definition of the mismatched ratio

Consider echo cancellation in ADSL system. In this system, the demanding downstream bandwidth is typically a multiple of the bandwidth of the upstream. Particularly, it allows the downstream band to overlap with the upstream band. Suppose the upstream signal is band-limited within f_1 and is sampled at a frequency f_{s1} that is at least larger than twice of f_1 to avoid samples aliasing. The downstream signal is band-limited within f_2 and is sampled a frequency f_{s2} which is also a value twice more than f_2. Then the mismatched ratio is defined as

$$\text{Mismatched ratio: } M = \frac{f_{s2}}{f_{s1}} = \frac{\boldsymbol{R\_rx}}{\boldsymbol{R\_tx}} \tag{1.1}$$

For example, suppose the transmit signal and the receive signal are both over-sampled twice, and the upstream and downstream bandwidths are 256 KHz and 1 MHz, respectively, then $M = 4$.

In this paper, we will consider ***R_rx*** is an integer multiple of ***R_tx***. Based on the mismatched rate scheme, we will consider the LMS (Least Mean Square) and the RLS (Recursive Least Square) algorithms in chapter 2. We will also give a convergence analysis of the reduced complexity mismatched rate LMS in chapter 3. Some performance effects introduced by non-ideal re-sampling blocks will be discussed in chapter 4. In chapter 5, simulation will be shown to verify the performance of the algorithms and illustrate our problems in discussion. We conclude this paper in chapter 6.

Finally, small capital symbol in bold face denotes a vector, and capital symbol in bold face denotes a matrix in this paper. Transposition of a vector/matrix is denoted as $\{.\}^T$. Dimension of a symbol will denote beside with lower case if necessary.

Chapter2 Matched and Mismatched Rate Adaptive Algorithms

Here two popular adaptive algorithms will present, one is LMS, and the other is RLS. In this chapter, we will show how the reduced complexity mismatched rate LMS and RLS is derived. We will first introduce the necessary derivations of conventional LMS and RLS in matched rate environment. The mismatched-rate LMS and RLS with reduced complexity will use their analogies.

2.1: Matched rate adaptive algorithms

2.1.1: General descriptions

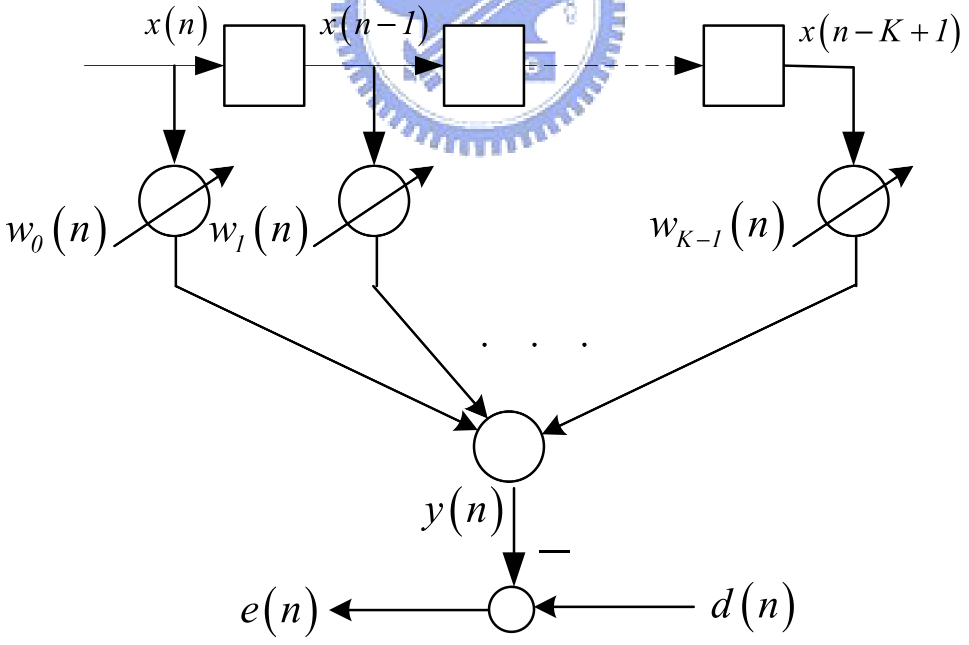

Figure 2.1 A K-tap transversal adaptive filter

In Figure 2.1, *d*(*n*) is the desired signal generated by a system we are going to identify.

$d(n)$ can be viewed as a linear combination of the last K samples of the input signal, corrupted by independent zero-mean plant noise $v(n)$. Our aim in this application is to estimate an unknown plant through minimization of the output error $e(n)$ in the mean square sense. For purposes of analysis, we consider the plant to be a transversal FIR filter.

Referring to Figure 2.1, the input signal vector at the nth sampling instant is denoted by

$$\mathbf{x}(n) = \left[x(n),\ \ldots\ ,x(n-K+1)\right]^T, \tag{2.1}$$

and the set of weights of the adaptive transversal filter is denoted by

$$\mathbf{w}(n) = \left[w_0(n),\ \ldots\ ,w_{K-1}(n)\right]^T. \tag{2.2}$$

The nth output sample is

$$y(n) = \sum_{i=0}^{K-1} w_i(n)x(n-i) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n) = \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}(n). \tag{2.3}$$

The input signal vector and the desired response are assumed to be wide-sense stationary. Denoting the desired signal as $d(n)$, the error at the nth time is

$$e(n) = d(n) - y(n) = d(n) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n) = d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}(n). \tag{2.4}$$

The square of this error is

$$e^2(n) = d^2(n) - 2d(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{w}(n) + \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{w}(n). \tag{2.5}$$

The mean square error (MSE), ξ, defined as the expected value of $e^2(n)$, is

$$\begin{aligned}\xi &\equiv E\left[e^2(n)\right]\\ &= E\left[d^2(n)\right] - 2E\left[d(n)\mathbf{x}^{\mathrm{T}}(n)\right]\mathbf{w}(n) + \mathbf{w}^{\mathrm{T}}(n)E\left[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\right]\mathbf{w}(n)\\ &= E\left[d^2(n)\right] - 2\mathbf{p}^{\mathrm{T}}\mathbf{w}(n) + \mathbf{w}^{\mathrm{T}}(n)\mathbf{R}\mathbf{w}(n),\end{aligned} \tag{2.6}$$

where the cross-correlation vector between the input signal and the desired response is defined as

$$E\left[d(n)\mathbf{x}^{\mathrm{T}}(n)\right] \equiv \mathbf{p}, \tag{2.7}$$

and the input autocorrelation matrix $\mathbf{R}$ is defined as

$$E\left[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\right] \equiv \mathbf{R} = \mathbf{R}^{\mathrm{T}}. \tag{2.8}$$

2.1.2: Wiener filter

The minimum ξ_{min} of ξ can be obtained by differentiating ξ with respect to $\mathbf{w}$ (time index is omitted for clarity). It can be written in vector gradient form:

$$\nabla\xi = \left[\frac{\partial\xi}{\partial w_0}\ \frac{\partial\xi}{\partial w_1}\ \cdots\ \frac{\partial\xi}{\partial w_{K-1}}\right]^{\mathrm{T}} \tag{2.9}$$

where ∇ denotes the gradient operator.

By applying matrix differentiations listed in appendix, the gradient vector can be written as:

$$\begin{aligned}\nabla\xi &= \nabla\left\{E\left[d^2(n)\right] - 2\mathbf{w}^{\mathrm{T}}\mathbf{p} + \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}\right\} = \nabla\left(-2\mathbf{w}^{\mathrm{T}}\mathbf{p}\right) + \nabla\left(\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}\right) \\ &= -2\mathbf{p} + \left(\mathbf{R} + \mathbf{R}^{\mathrm{T}}\right)\mathbf{w}.\end{aligned} \tag{2.10}$$

Since $\mathbf{R}$ is symmetric, (2.10) can be written as:

$$\nabla\xi = \mathbf{2Rw} - \mathbf{2p}\,. \tag{2.11}$$

By setting the gradient to zero, we can obtain the well-known *Wiener-Hopf equation*:

$$\mathbf{R}\mathbf{w}_{\mathbf{o}} = \mathbf{p}\ , \tag{2.12}$$

where

$$\mathbf{w} = \mathbf{w}_{\mathbf{0}} = \mathbf{R}^{-1}\mathbf{p} \tag{2.13}$$

is the optimal weight vector known as the *Wiener filter tap vector* or the *Wiener solution*.

Replacing $\mathbf{w}(n)$ by $\mathbf{w}_{\mathbf{o}}$ and $\mathbf{R}\mathbf{w}_{\mathbf{o}}$ by $\mathbf{p}$ in (2.6), we obtain

$$\begin{aligned}\xi_{min} &= E\left[d^2(n)\right] - \mathbf{w}_{\mathbf{o}}^{\mathrm{T}}\mathbf{p} \\ &= E\left[d^2(n)\right] - \mathbf{w}_{\mathbf{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathbf{o}}.\end{aligned} \tag{2.14}$$

This is the minimum mean-square error that can be achieved by the transversal Wiener filter and is obtained when its tap weights are chosen according to the optimum solution given by (2.13).

Recall (2.6), the MSE ξ can be rearranged as follows:

$$\xi = \mathbf{w}^{\mathbf{T}}(n)\mathbf{R}\mathbf{w}(n) - \mathbf{p}^{\mathbf{T}}\mathbf{w}(n) - \mathbf{w}^{\mathbf{T}}(n)\mathbf{p} + E\left[d^2(n)\right].$$

We substitute $\mathbf{p}$ with (2.12) and reform the above equation with additional term $\mathbf{w}_{\mathbf{o}}^{\mathbf{T}}\mathbf{R}\mathbf{w}_{\mathbf{o}}$, we obtain

$$\begin{aligned}\xi &= \mathbf{w}^{\mathbf{T}}(n)\mathbf{R}\mathbf{w}(n) - \mathbf{w}_{\mathbf{o}}^{\mathbf{T}}(n)\mathbf{R}^{\mathbf{T}}\mathbf{w}(n) - \mathbf{w}^{\mathbf{T}}(n)\mathbf{R}\mathbf{w}_{\mathbf{o}} + \mathbf{w}_{\mathbf{o}}^{\mathbf{T}}\mathbf{R}\mathbf{w}_{\mathbf{o}} + E\left[d^2(n)\right] - \mathbf{w}_{\mathbf{o}}^{\mathbf{T}}\mathbf{R}\mathbf{w}_{\mathbf{o}} \\ &= \left(\mathbf{w}(n) - \mathbf{w}_{\mathbf{o}}\right)^{\mathbf{T}}\mathbf{R}\left(\mathbf{w}(n) - \mathbf{w}_{\mathbf{o}}\right) + E\left[d^2(n)\right] - \mathbf{w}_{\mathbf{o}}^{\mathbf{T}}\mathbf{R}\mathbf{w}_{\mathbf{o}}.\end{aligned}$$

Substitute the right most two terms of the above equation with (2.14), we get

$$\xi = \xi_{min} + \left(\mathbf{w}(n) - \mathbf{w}_{\mathbf{o}}\right)^{\mathbf{T}}\mathbf{R}\left(\mathbf{w}(n) - \mathbf{w}_{\mathbf{o}}\right) . \tag{2.15}$$

We will use (2.15) to study the MSE convergence behavior in chapter 3.

2.1.3: Steepest-Descent algorithm

Directly find out the Wiener solution of $\mathbf{w}$ is not practical. Instead of solving (2.12), the steepest-descent algorithm provides a general scheme that iteratively searches for the minimum point of any convex function. By starting with an initial guess of $\mathbf{w}$, the general iterative update procedure is:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla\xi \tag{2.16}$$

where μ is a positive scalar step-size, and ξ is the function whose minimum is the goal we are searching for.

Substitute (2.11) into (2.16), we derive the steepest-descent update equation:

$$\mathbf{w}(n+1)=\mathbf{w}(n)-2\mu\left(\mathbf{R}\mathbf{w}(n)-\mathbf{p}\right). \tag{2.17}$$

The reason why we outline the steepest-descent method here is because one of the important issues in this paper: LMS convergence behavior, will take advantage of its exact description of the stochastic learning curve.

2.1.4: LMS

LMS is a robust algorithm that is notified by its simplicity of computation and performance of tracking ability. Conventional LMS algorithm is a stochastic implementation of the steepest-descent method. It simply replaces the cost function $\xi=E\left[e^2(n)\right]$ by its instantaneous coarse estimate $e^2(n)$ in (2.16):

$$\mathbf{w}(n+1)=\mathbf{w}(n)-\mu\nabla e^2(n). \tag{2.18}$$

Noting that the gradient differentiates respect to $\mathbf{w}$, the last term can be written as:

$$\nabla e^2(n)=2e(n)\nabla e(n)=2e(n)\nabla\left(d(n)-\mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)\right)=-2e(n)\mathbf{x}(n). \tag{2.19}$$

Substitute (2.19) into (2.18), the LMS recursive update equation can be written as:

$$\mathbf{w}(n+1)=\mathbf{w}(n)+2\mu e(n)\mathbf{x}(n). \tag{2.20}$$

2.1.5: RLS

On the goal to obtain the optimum solution, the method of least squares provides a different point of view whose method is primarily based on a deterministic framework, while the LMS provides a statistical framework. In the method of least-squares, at any time instant $n > 0$, the adaptive filter parameters are calculated so that the cost function

$$\zeta(n) = \sum_{k=1}^{n} \lambda_n(k) e_n^2(k) \tag{2.21}$$

is minimized.

As RLS is named, it recursively update the adaptive filter tap-weights with feedback estimation error $\hat{e}_{n-1}(n)$ and the gain vector $\mathbf{k}(n)$ in the form:

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\hat{e}_{n-1}(n). \tag{2.22}$$

The following contents of this sub section will give a brief demonstration of how the update equation holds.

Rewrite (2.21) in matrix form:

$$\zeta(n) = \mathbf{e}^{\mathrm{T}}(n)\mathbf{\Lambda}(n)\mathbf{e}(n), \tag{2.23}$$

where

$$\mathbf{e}(n) = \left[e_n(1)\ e_n(2)\ \cdots\ e_n(n)\right]^{\mathrm{T}} \text{ is a collection of errors,} \tag{2.24}$$

and

$$\mathbf{\Lambda}(n) = diag\left[\ \lambda^{n-1}\ \lambda^{n-2}\ \cdots\ 1\right],\ 0 < \lambda < 1, \text{ is the forgetting factor.}$$

There is a corresponding filtered signal $y(n)$ for the desired response $d(n)$ to obtain a specified residue error in the relation of

$$e_i(n) = d(i) - y_n(i) = d(i) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(i),\ i = 1 \sim n. \tag{2.25}$$

The corresponding desired response vector

$$\mathbf{d}(n)=\left[d(1)\ d(2)\ \cdots\ d(n)\right]^{\mathrm{T}} \tag{2.26}$$

and the corresponding filtered signal vector

$$\begin{aligned}\mathbf{y}(n)&=\left[y_n(1)\ y_n(2)\ \cdots\ y_n(n)\right]^{\mathrm{T}}\\ &\equiv \mathbf{w}^{\mathrm{T}}(n)\hat{\mathbf{X}}(n)=\hat{\mathbf{X}}^{\mathrm{T}}(n)\mathbf{w}(n)\end{aligned} \tag{2.27}$$

where

$$\hat{\mathbf{X}}(n)\equiv\left[\mathbf{x}(1)\ \mathbf{x}(2)\ \cdots\ \mathbf{x}(n)\right] \tag{2.28}$$

is the observed input data matrix.

The cost function shown in (2.23) can now be written as

$$\zeta(n)=\mathbf{d}^{\mathrm{T}}(n)\mathbf{\Lambda}(n)\mathbf{d}(n)-2\boldsymbol{\theta}_{\lambda}^{\mathrm{T}}(n)\mathbf{w}(n)+\mathbf{w}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}(n)\mathbf{w}(n), \tag{2.29}$$

where the cross correlation with the desired signal vector

$$\begin{aligned}\boldsymbol{\theta}_{\lambda}(n)&=\hat{\mathbf{X}}(n)\mathbf{\Lambda}(n)\mathbf{d}(n)\\ &=\mathbf{x}(n)d(n)+\lambda\mathbf{x}(n-1)d(n-1)+\cdots\\ &=\lambda\boldsymbol{\theta}_{\lambda}(n-1)+\mathbf{x}(n)d(n)\end{aligned} \tag{2.30}$$

and the correlation matrix

$$\begin{aligned}\mathbf{\Psi}_{\lambda}(n)&=\hat{\mathbf{X}}(n)\mathbf{\Lambda}(n)\hat{\mathbf{X}}^{\mathrm{T}}(n)\\ &=\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)+\lambda\mathbf{x}(n-1)\mathbf{x}^{\mathrm{T}}(n-1)+\cdots\\ &=\lambda\mathbf{\Psi}_{\lambda}(n-1)+\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n).\end{aligned} \tag{2.31}$$

Differentiate (2.29) with respect to $\mathbf{w}(n)$ and set the resulting gradient equation to zero, we can write down the *normal equation* for a linear least-squares filter:

$$\mathbf{\Psi}_{\lambda}(n)\hat{\mathbf{w}}(n)=\boldsymbol{\theta}_{\lambda}(n) \tag{2.32}$$

where $\hat{\mathbf{w}}(n)$ is the estimate of filter tap-weight in the least-squares sense.

It follows the least-squares solution:

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{\Psi}_{\lambda}^{-1}(n)\mathbf{\theta}_{\lambda}(n) \\ &= \lambda\mathbf{\Psi}_{\lambda}^{-1}(n)\mathbf{\theta}_{\lambda}(n-1)+\mathbf{\Psi}_{\lambda}^{-1}(n)\mathbf{x}(n)d(n).\end{aligned} \tag{2.33}$$

Substitute (2.33) into (2.29), the minimum value of $\zeta(n)$ is obtained as

$$\begin{aligned}\zeta_{min}(n) &= \mathbf{d}^{\mathrm{T}}(n)\mathbf{\Lambda}(n)\mathbf{d}(n)-\mathbf{\theta}_{\lambda}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n)\mathbf{\theta}_{\lambda}(n) \\ &= \mathbf{d}^{\mathrm{T}}(n)\mathbf{\Lambda}(n)\mathbf{d}(n)-\mathbf{\theta}_{\lambda}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n).\end{aligned} \tag{2.34}$$

In (2.33), it is clear that if we want to find the optimum solution of $\hat{\mathbf{w}}(n)$, $\mathbf{\Psi}_{\lambda}^{-1}(n)$ must be solved.

Using (2.31) and the matrix inverse lemma shown in appendix B, we have

$$\begin{aligned}\mathbf{\Psi}_{\lambda}^{-1}(n) &= \lambda^{-1}\mathbf{\Psi}_{\lambda}^{-1}(n-1)-\frac{\lambda^{-2}\mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n-1)}{1+\lambda^{-1}\mathbf{x}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)} \\ &\equiv \lambda^{-1}\left[\mathbf{\Psi}_{\lambda}^{-1}(n-1)-\mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n-1)\right],\end{aligned} \tag{2.35}$$

where

$$\mathbf{k}(n) \equiv \frac{\lambda^{-1}\mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}. \tag{2.36}$$

Write (2.36) out, we have

$$\mathbf{k}(n) = \lambda^{-1}\left[\mathbf{\Psi}_{\lambda}^{-1}(n-1)-\mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n-1)\right]\mathbf{x}(n). \tag{2.37}$$

Substitute (2.35) into (2.37), we obtain

$$\mathbf{k}(n) = \mathbf{\Psi}_{\lambda}^{-1}(n)\mathbf{x}(n). \tag{2.38}$$

Rewrite (2.33) with (2.38), we have

$$\hat{\mathbf{w}}(n) = \lambda\mathbf{\Psi}_{\lambda}^{-1}(n)\mathbf{\theta}_{\lambda}(n-1)+\mathbf{k}(n)d(n). \tag{2.39}$$

Now replace $\mathbf{\Psi}_{\lambda}^{-1}(n)$ with the recursion derived in (2.35), we get

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{\theta}_{\lambda}(n-1)+\mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{\theta}_{\lambda}(n-1)+\mathbf{k}(n)d(n)\\ &= \hat{\mathbf{w}}(n-1)-\mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n-1)+\mathbf{k}(n)d(n)\\ &= \hat{\mathbf{w}}(n-1)+\mathbf{k}(n)\left(d(n)-\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n-1)\right)\\ &\equiv \hat{\mathbf{w}}(n-1)+\mathbf{k}(n)e_{n-1}(n).\end{aligned} \quad (2.40)$$

Here the estimation error $e_{n-1}(n)$ is defined as:

$e_{n-1}(n)=d(n)-\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n-1)$, determined by past weights and current input. (2.41)

In summary, the standard RLS update procedures is as follows:

1. Update the gain vector: (2.36)

$$\begin{aligned}&\mathbf{u}(n)=\mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n),\\ &denk=\lambda+\mathbf{x}^{\mathrm{T}}(n)\mathbf{u}(n),\\ &\mathbf{k}(n)=\frac{\mathbf{u}(n)}{denk}.\end{aligned}$$

2. Update the estimation error $e_{n-1}(n)$: (2.41)

$$e_{n-1}(n)=d(n)-\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n-1).$$

3. Update the tap weights $\hat{\mathbf{w}}(n)$: (2.40)

$$\hat{\mathbf{w}}(n)=\hat{\mathbf{w}}(n-1)+\mathbf{k}(n)e_{n-1}(n).$$

4. Update $\mathbf{\Psi}_{\lambda}^{-1}(n)$: (2.35)

$$\begin{aligned}\mathbf{\Psi}_{\lambda}^{-1}(n) &= \lambda^{-1}\left[\mathbf{\Psi}_{\lambda}^{-1}(n-1)-\mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{\Psi}_{\lambda}^{-1}(n-1)\right]\\ &= Tri\left\{\lambda^{-1}\left[\mathbf{\Psi}_{\lambda}^{-1}(n-1)-\mathbf{k}(n)\mathbf{u}^{\mathrm{T}}(n)\right]\right\}\end{aligned}.$$

The last equation has two purposes. One is to save the computation and the other is to stabilize the RLS convergence in implementation. $Tri\{\}$ is the operator that extracting the lower or upper part of the operating matrix and copies them to the either side except the diagonal.

2.2: Mismatched-rate adaptive algorithms with reduced complexity

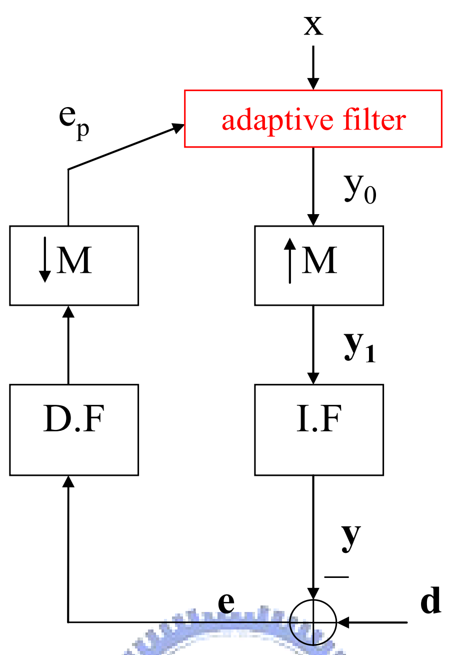

Figure 2.2 Reduced complexity adaptive filter.

The most significant difference between the matched rate and the mismatched rate system is that the number of the received desired signal samples in the mismatched rate system is M (or even larger) times more than that in the matched rate system. To cooperate with the characteristics of the received signal **d**, the identifying filtered signal **y** must be designed, at least, to match **d** in number. On the other hand, the reduced complexity structure is meaningfully proposed because the structure is using conventional adaptive algorithms even though it is in an environment that gives an increasing amount of data. In figure 2.2, the reduced complexity structure shows its clever collaboration. "IF" and "DF" denote the interpolation filter and the decimation filter respectively. The structure uses up-sample block to treat the filter output to match the number of received signal, and uses down-sample block to treat the filter input to maintain the low complexity. Notice that if extra wide band noise was added such that the signal **d** is not a narrow band signal, the down-sample block may give a mechanism that suppresses the disturbance and makes the filter input less noisy.

2.2.1: General descriptions

In order to derive the reduced complexity adaptive filter update rule, we now write each necessary component out. Starts from the output of the adaptive filter y_0, all necessary components following the arrow direction in figure 2.2 will be shown step by step. The operating function of each component will be also described in detail.

Let N_I and N_D denote the order of the interpolation and the decimation filter. We start from y_0, a simply K-tap linear transversal filter output:

$$y_0(n)=\sum_{i=0}^{K-1} w_i(n)x(n-i)=\mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)=\mathbf{x}^{\mathrm{T}}(n)\mathbf{w}(n). \tag{2.42}$$

In the initial state, the reduced complexity adaptive filter starts to train until it collects L filtered outputs, where L is a number we will show later. In other words, the filter starts to train when n = L, and at the same time, $y_0(1),\ldots,y_0(L)$ have already collected. Reform the filtered outputs into a vector form

$$\begin{aligned}\mathbf{y_0}(n)=\begin{bmatrix} y_0(n) \\ \vdots \\ y_0(n-L+1)\end{bmatrix}&=\begin{bmatrix}\sum_{i=0}^{K-1} w_i(n)x(n-i) \\ \vdots \\ \sum_{i=0}^{K-1} w_i(n)x(n-L+1-i)\end{bmatrix} \\ &\equiv \mathbf{X}(n)\mathbf{w}(n),\end{aligned} \tag{2.43}$$

where the input data are organized in a Hankel matrix form as

$$\mathbf{X}(n)_{L\times K}\equiv\begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-K+1) \\ x(n-1) & x(n-2) & \cdots & x(n-K) \\ \vdots & \vdots & \vdots & \vdots \\ x(n-L+1) & x(n-L) & \cdots & x(n-L-K+2)\end{bmatrix}. \tag{2.44}$$

From (2.44), we can see that the adaptive filter pays *L+K-1* latencies before starting to train.

Passing $\mathbf{y}_0(n)$ through an M times zero insertion up-sampler, we write the resulting vector $\mathbf{y}_1(n)$ in matrix form:

$$\mathbf{y}_1(n) \equiv \mathbf{\Omega}\, \mathbf{y}_0(n) = \mathbf{\Omega}\, \mathbf{X}(n)\mathbf{w}(n), \tag{2.45}$$

where $\mathbf{\Omega}$ is a zero insertion matrix. For example, if $M = 2$, $\mathbf{\Omega}$ can be written as

$$\mathbf{\Omega}_{J\times L} = \begin{bmatrix} 1\,0\,0 \cdots & & 0 \\ 0\,0\,0 \cdots & & 0 \\ 0\,1\,0 \cdots & & 0 \\ 0\,0\,0 \cdots & & 0 \\ \vdots & & \vdots \\ 0\,0\,0 \cdots & & 0\,1\,0 \\ 0\,0\,0 \cdots & & 0\,0\,0 \\ 0\,0\,0 \cdots & & 0\,0\,1 \\ 0\,0\,0 \cdots & & 0\,0\,0 \end{bmatrix}$$

where $J = N_D+N_I+M-2$, which is a number designed to fit the following matrix operation.

In view of $\mathbf{\Omega}$, we can now determine that $L = ceil(J/M)$.

Next we determine the signal $\mathbf{y}(n)$ that passes $\mathbf{y}_1(n)$ through the interpolation filter as:

$$\mathbf{y}(n) \equiv \mathbf{F}\, \mathbf{y}_1(n) = \mathbf{F}\, \mathbf{\Omega}\, \mathbf{X}(n)\mathbf{w}(n), \tag{2.46}$$

where $\mathbf{F}$ is the convolution matrix with dimension N_D+M-1 by J. Suppose the interpolation filter coefficients are $f_1 \cdots f_{N_I}$, $\mathbf{F}$ can be written as:

$$\mathbf{F} = \begin{bmatrix} f_1 & f_2 & \cdots & f_{N_I-1} & f_{N_I} & & & \mathbf{0} \\ & f_1 & f_2 & \cdots & & f_{N_I-1} & f_{N_I} & \\ & & \ddots & & & & \ddots & \\ \mathbf{0} & & & f_1 & f_2 & \cdots & f_{N_I-1} & f_{N_I} \end{bmatrix}. \tag{2.47}$$

In (2.46), it shows that there are N_D+M-1 samples to match the desired response. Actually, only first M samples are needed and the rest of them are not in use at present. One may question that if the rest of samples are useless, why do we need to compute them? This answer lies in whether we use the decimation filter or not. If we neglect the decimation filter, the dimension of $\mathbf{y}(n)$ is M and no redundant computations will be needed. However, in specific application such as echo cancellation in ADSL, the downlink high frequency components will exist in the operating band of the received signal and is recognized as a disturbance for echo identification (notice that we only want the signal generated by the echo path in the received signal for echo cancellation). If there is no barrier such as the low pass decimation filter, the feedback error will be no longer clean and the performance will consequently decay.

Return to our discussion, the error vector $\mathbf{e}(n)$ with dimension N_D+M-1 is

$$\mathbf{e}(n)=\mathbf{d}(n)-\mathbf{y}(n).$$

Suppose the decimation filter coefficients are $g_1 \cdots g_{N_D}$, and one of the *first M* phases, say, phase p, is selected to do the decimation. Then we can write the down sampled error $e_p(n)$, a scalar, as:

$$e_p(n)=\mathbf{g_p}\mathbf{e}(n), \tag{2.48}$$

where the combined effect of decimation filter and the decimator can be written as:

$$\mathbf{g_p}=\left[zeros(p-1)\ g_1 \cdots g_{N_D}\ zeros(M-p)\right],\ 1\le p\le M. \tag{2.49}$$

Here "*zeros*(k)" means there are k zeros line in a row, $k\ge 0$.

Finally, we combine the effects of the up-sample block and the down-sample block into a vector $\mathbf{h_p}$, dimension 1 by L:

$$\mathbf{h_p}=\mathbf{g_p}\ \mathbf{F}\ \mathbf{\Omega}, \tag{2.50}$$

where p is the decimator chosen phase.

The study of the combined effect $\mathbf{h}_\mathrm{p}$ is one of the major topics in this paper.

2.2.2: LMS

The cost function here is $e_p^2(n)$.

By using (2.46), (2.48) and (2.50), the square of the down sampled error can be written as:

$$\begin{aligned} e_p^2(n) &= \left[\mathbf{g}_\mathrm{p}\ \mathbf{e}(n)\right]^2 = \left[\mathbf{g}_\mathrm{p}\ \mathbf{d}(n) - \mathbf{g}_\mathrm{p}\ \mathbf{y}(n)\right]^2 \\ &= \left[\mathbf{g}_\mathrm{p}\ \mathbf{d}(n) - \mathbf{g}_\mathrm{p}\ \mathbf{F}\ \boldsymbol{\Omega}\ \mathbf{X}(n)\mathbf{w}(n)\right]^2 \\ &= \left[\mathbf{g}_\mathrm{p}\ \mathbf{d}(n) - \mathbf{h}_\mathrm{p}\ \mathbf{X}(n)\mathbf{w}(n)\right]^2 \\ &= \left[\mathbf{d}^\mathrm{T}(n)\mathbf{g}_\mathrm{p}^\mathrm{T} - \mathbf{w}^\mathrm{T}(n)\mathbf{X}^\mathrm{T}(n)\mathbf{h}_\mathrm{p}^\mathrm{T}\ \right]^2. \end{aligned} \tag{2.51}$$

Using the analogies of (2.18), differentiate (2.51) with respect to $\mathbf{w}$, we get

$$\begin{aligned} \nabla e_p^2(n) &= 2e_p(n)\nabla e_p(n) \\ &= -2e_p(n)\mathbf{X}^\mathrm{T}(n)\mathbf{h}_\mathrm{p}^\mathrm{T}. \end{aligned} \tag{2.52}$$

Substitute (2.52) into (2.18), we have the LMS update rule:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e_p(n)\mathbf{X}^\mathrm{T}(n)\mathbf{h}_\mathrm{p}^\mathrm{T}. \tag{2.53}$$

Notice that since $\mathbf{h}_\mathrm{p}$ is determined once the IF and DF are determined, rather than calculating the whole $\mathbf{h}_\mathrm{p}\mathbf{X}(n)$, one may use the shifting property of $\mathbf{X}(n)$ such that only the latest L data have to be processed during each update.

2.2.2.1: Simplified version

Since both the interpolation filter and the decimation filter are low pass filters, the combined effect $\mathbf{h_p}$ can be designed to have an impulse response similar to sinc function with one dominant coefficient. The simplified version $\hat{\mathbf{h}}_\mathbf{p}$ set the dominant coefficient of $\mathbf{h_p}$ one and set the rest of them zeros. Now we can get

$$\hat{\mathbf{h}}_\mathbf{p}\mathbf{X}(n)=\left[x(n-d)\ x(n-d-1)\ \cdots\ x(n-d-K-1)\right],$$

where d is the index of the largest element of $\mathbf{h_p}$. Then the update rule of LMS can be written as

$$\mathbf{w}(n+1)=\mathbf{w}(n)+2\mu e_p(n)\mathbf{x}(n-d). \tag{2.54}$$

Observe (2.54), it is just the same as the conventional LMS update equation in (2.20).

Compared to the original LMS, the simplified version replaces the color factor $\mathbf{h_p}$ with a delta function $\hat{\mathbf{h}}_\mathbf{p}$. Before the original LMS updates, it has to process the input data by the combined effect $\mathbf{h_p}$. As we shall discuss in chapter 4, the combined effect will become a color factor of the input data if the distortion exists within the up-sample and down-sample blocks. The simplified version can avoid such color effect of the input data and hence the convergence speed will be faster than the original LMS in most of the cases.

Performance comparison between the reduced complexity adaptive algorithm and its simplified version will be shown in section 5.1.

2.2.3: RLS

Reduced complexity RLS algorithm is similar to the conventional RLS algorithm by simply modifies the input data sequence. In this paper, we call this reduced complexity RLS for mismatched rate system by "vector input" RLS.

Recall (2.44), the Hankel input data matrix can be written as

$$\mathbf{X}(n)_{L\times K} = \left[\bar{\mathbf{x}}(n)\ \bar{\mathbf{x}}(n-1)\ \cdots\ \bar{\mathbf{x}}(n-K+1)\right],$$

where

$$\bar{\mathbf{x}}(n)_{L\times 1} = \left[x(n)\ x(n-1)\ \cdots\ x(n-L+1)\right]^{\mathrm{T}}.$$

Then the "vector input" RLS data sequence has the form:

$$\mathbf{h}_{\mathrm{p}}\mathbf{X}(n) = \left[\mathbf{h}_{\mathrm{p}}\bar{\mathbf{x}}(n)\ \mathbf{h}_{\mathrm{p}}\bar{\mathbf{x}}(n-1)\ \cdots\ \mathbf{h}_{\mathrm{p}}\bar{\mathbf{x}}(n-K+1)\right]_{1\times K}. \tag{2.55}$$

The difference between the "vector input" RLS and the conventional RLS is that the update of the "vector input" RLS data sequence needs to compute $\mathbf{h}_{\mathrm{p}}\bar{\mathbf{x}}(n)$, which requires L multiplications and L-1 additions, and the conventional RLS needs just a shift. The "vector input" RLS is named because it requires a vector $\bar{\mathbf{x}}(n)$ with length L to compute the latest input data feasible for updating the mismatched rate RLS system.

The "vector input" RLS modifies the conventional RLS by its input from $x(n)$ to $\mathbf{h}_{\mathrm{p}}\bar{\mathbf{x}}(n)$, so the cross correlation vector becomes

$$\boldsymbol{\theta}_{\lambda}(n) = \lambda\boldsymbol{\theta}_{\lambda}(n-1) + \mathbf{X}(n)\mathbf{h}_{\mathrm{p}}^{\mathrm{T}}\, d(n), \tag{2.56}$$

and the correlation matrix becomes

$$\boldsymbol{\Psi}_{\lambda}(n) = \lambda\boldsymbol{\Psi}_{\lambda}(n-1) + \mathbf{X}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{p}}^{\mathrm{T}}\,\mathbf{h}_{\mathrm{p}}\mathbf{X}(n). \tag{2.57}$$

By simply substitute $\mathbf{x}(n)$ in conventional RLS with $\left[\mathbf{h_p X}(n)\right]^{\mathrm{T}}$, we summarize the reduced complexity mismatched rate RLS as follows.

1. Update the gain vector:

$$\mathbf{u}(n) = \mathbf{\Psi}_{\lambda}^{-1}(n-1)\mathbf{X}^{\mathrm{T}}(n)\mathbf{h}_{\mathbf{p}}^{\mathrm{T}},$$
$$denk = \lambda + \mathbf{h_p X}(n)\mathbf{u}(n),$$
$$\mathbf{k}(n) = \frac{\mathbf{u}(n)}{denk}.$$

2. Update the filtering error $e_p(n)$:

$$\mathbf{y}(n) = \mathbf{F}\ \mathbf{\Omega}\ \mathbf{X}(n)\mathbf{w}(n)$$
$$e_p(n) = \mathbf{g_p}\left[\mathbf{d}(n) - \mathbf{y}(n)\right], \text{ note that } \mathbf{h} = \mathbf{g_p F}\ \mathbf{\Omega}.$$

3. Update the tap weights $\mathbf{w}(n)$:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)e_p(n).$$

4. Update $\mathbf{\Psi}_{\lambda}^{-1}(n)$:

$$\mathbf{\Psi}_{\lambda}^{-1}(n) = Tri\left\{\lambda^{-1}\left[\mathbf{\Psi}_{\lambda}^{-1}(n-1) - \mathbf{k}(n)\mathbf{u}^{\mathrm{T}}(n)\right]\right\}.$$

RLS simplified version can use the same update methods listed above by replacing $\mathbf{h_p}$ by $\hat{\mathbf{h}}_{\mathbf{p}}$, where $\hat{\mathbf{h}}_{\mathbf{p}}$ is exactly the same described in section 2.2.2.1.

Applying conventional RLS into this structure is straight forward, but there are still something special we have to take care with, we point them out in the next sub-section.

2.2.3.1: Initialization problems

Two kinds of initialization scheme we will demonstrate in this paper, one happens in the beginning of the training, and the other may happen in the midway.

For the initialization in the beginning of training, there are several kinds of popular data matrix windowing [2]. Here we will use two of them. One is pre-windowing, and the other is covariance windowing. Using the pre-windowing for the data matrix is to start the input data sequence from all zero and no latencies are paid. Using covariance windowing for the data matrix is to fill the input data sequence with only data and we have to pay *L-K+1* latencies for full-filling the data matrix. In this paper, we often use the covariance windowing. The pre-windowing is used in some special cases such as re-initialization problem.

In the reduced complexity structure, the performance of the pre-windowing version is a little worse than the covariance windowing version which pays latencies for initializing the sequence with $\mathbf{h_p X}(n)$. We will see the demonstrations in section 5.2.1.

The initialization problem can apply not only in the beginning of the training, but also in the midway. In real time applications, the limited memory size in DSP and the power consumption may be big factors for continuous adaptation of RLS. One of the straight forward solutions is to stop adaptation for a while and wait until the previous computations complete offline. To maintain the performance under this stagnant adaptation, the re-initialization is developed for this RLS computation-relaxed case.

The other application for the RLS re-initialization is the adaptation for time varying channels. If the RLS adaptive filter has finished training for a channel before it changes, we may periodically modify the existing RLS parameters to fit the current channel condition. For such modification, re-initialization may be used to maintain the tracking performance. We will not consider this case in our simulation experiment.

For the computation-relaxed problems, the performance of the transient state (convergence) and the steady state (tracking) of the adaptive filter are both under challenges due to un-consistent adaptations. We can assert that the performance will degradation if the algorithm has no modification for this situation. In section 5.2.2, we will demonstrate the re-initialization phenomenon.

2.3: Remarks

The adaptive filters demonstrated in section 2.2 are all designed based on minimizing $e_p^2(n)$ no matter in its statistical property or in exact value. Because of the filter inputs $\mathbf{h_p X}(n)$ and $e_p^2(n)$ are all operating in a lower rate, it is natural to see that the complexity of the mismatched rate system adaptation is similar to that of the conventional adaptive algorithm. Table 2.1 shows the number of multiplications at each update iteration applying different architectures for these algorithms.

	LMS	Simplified LMS	RLS
Reduced complexity structure. Fig.1.2	$(L+1)(K+N_D+M)$	$(L+1)(K+N_D+M)-L$	$2K^2+(L+2)K$ $+(L+1)(N_D+M-1)+L$
Conventional structure. Fig.1.1	~M^2 times more than the above.	"	~M^2 times more than the above.

Table 2.1: Number of multiplications in update equations.

At last, the performance comparison between LMS and RLS by inputting the same sequence (input data and desired responses) will also be shown in section 5.3.

Chapter3 Convergence Behavior of mismatched Rate LMS

In this chapter, the convergence behavior of the reduced complexity mismatched rate LMS algorithm will be addressed. Besides its MSE learning curve, we will also give a prediction of its convergence speed and optimum convergence bound. In this chapter, we will first show some basic concepts about how conventional LMS is modeled for its convergence behavior. Some tools such as the steepest-descent method and Wiener-Hopf equation introduced in section 2.1 will be used to analyze the LMS convergence. Then the reduced complexity algorithm will be studied.

3.1: Matched rate LMS convergence behavior

Before entering the LMS convergence section, we first introduce the convergence analysis of the steepest-descent method, which will give us some deterministic insights of the convergence behavior of such stochastic adaptive algorithms. The convergence analysis is following the derivations listed in [4], [5].

3.1.1: The steepest-descent method

The steepest-descent method provides us a deterministic way to derive some necessary convergence indices such as the time constants that we will use later. To study the convergence behavior of stochastic adaptive algorithm, we start from analyzing their MSE (mean square error).

Recall (2.15), the learning curve can be written as

$$\xi(n)=\xi_{min}+\left(\mathbf{w}(n)-\mathbf{w_o}\right)^{\mathbf{T}}\mathbf{R}\left(\mathbf{w}(n)-\mathbf{w_o}\right).$$

Note that the learning curve is defined as the MSE ξ versus the time index *n*.

This formula shows that the convergence goes to a steady state once its filter tap-weights approach $\mathbf{w_0}$**,** at that moment, $\xi \cong \xi_{min}$. Here, given the stationary signals *x*(n) and *d*(n), the optimum tap-weight vector $\mathbf{w_0}$, can be determined according to the Wiener-Hopf equation listed in (2.13).

However, with only (2.15), there is not enough information to give further insights such as convergence speed and some other interesting issues. The right-hand term of (2.15) can be expanded

$$\xi(n)=\xi_{min}+\left(\mathbf{w}(n)-\mathbf{w_o}\right)^{\mathbf{T}}\mathbf{Q\Lambda Q^T}\left(\mathbf{w}(n)-\mathbf{w_o}\right). \tag{3.1}$$

Here we use the unitary similarity decomposition property of a symmetric matrix, **R** is decomposed as

$$\mathbf{R}=\mathbf{Q\Lambda Q^T}, \tag{3.2}$$

where $\mathbf{\Lambda}$ is a diagonal matrix consisting of the eigenvalues $\lambda_0,\lambda_1,\cdots,\lambda_{K-1}$ of **R** and the columns of **Q** contain the corresponding orthonormal eigenvectors.

Define the weight-error vector **v**(*n*) as

$$\mathbf{v}(n)=\mathbf{w}(n)-\mathbf{w_o}, \tag{3.3}$$

we can rewrite (3.1) as

$$\xi(n)=\xi_{min}+\mathbf{v^T}(n)\mathbf{Q\Lambda Q^T}\mathbf{v}(n). \tag{3.4}$$

Now define the transformed weight-error vector **v'**(*n*) as **v'**(*n*) = $\mathbf{Q^T}$ **v**(*n*), we can get

$$\begin{aligned}\xi(n)&=\xi_{min}+\mathbf{v}'(n)^{\mathbf{T}}\mathbf{\Lambda v}'(n)\\&=\xi_{min}+\sum_{i=0}^{K-1}\lambda_i v_i'^2(n).\quad\text{(scalar form)}\end{aligned} \tag{3.5}$$

To derive the transformed weight-error *v'*(*n*) for the steepest-descent method, its update equation (2.16) is rewritten as

$$\mathbf{w}(n+1)=(\mathbf{I}-2\mu\mathbf{R})\mathbf{w}(n)+2\mu\mathbf{p}. \tag{3.6}$$

where **I** is the *K* by *K* identity matrix.

Subtract $\mathbf{w_0}$ from both sides of (3.6) and rearrange the result to obtain

$$\mathbf{w}(n+1)-\mathbf{w_o}=(\mathbf{I}-2\mu\mathbf{R})(\mathbf{w}(n)-\mathbf{w_o}) \tag{3.7}$$

here we replace **p** by (2.12). Substitute into (3.7), we obtain

$$\mathbf{v}(n+1)=(\mathbf{I}-2\mu\mathbf{R})\mathbf{v}(n). \tag{3.8}$$

Substituting (3.8) into (3.7) and replacing **I** with $\mathbf{QQ^T}$, rewrite (3.8) as

$$\begin{aligned}\mathbf{v}(n+1)&=\left(\mathbf{QQ^T}-2\mu\mathbf{Q\Lambda Q^T}\right)\mathbf{v}(n)\\&=\mathbf{Q}(\mathbf{I}-2\mu\mathbf{\Lambda})\mathbf{Q^T}\mathbf{v}(n).\end{aligned} \tag{3.9}$$

Then transform **v**(*n*) to **v'**(*n*) by pre-multiplying (3.9) by $\mathbf{Q^T}$, the recursive equation (3.9) is now reformed as

$$\mathbf{v}'(n+1)=(\mathbf{I}-2\mu\mathbf{\Lambda})\mathbf{v}'(n). \tag{3.10}$$

The vector recursion (3.10) can be separated into the scalar recursive equations

$$v_i'(n+1)=(1-2\mu\lambda_i)v_i'(n),\ \text{for } i=0,1,...,K-1, \tag{3.11}$$

where $v_i'(n)$ is the *i*th element of the vector $\mathbf{v}'(n)$.

Starting with a set of initial values $v_0'(0),v_1'(0),\ldots,v_{K-1}'(0)$ and iterating *n* times, the scalar recursion can be written as

$$v_i'(n)=(1-2\mu\lambda_i)^n v_i'(0),\ \text{for } i=0,1,...,K-1. \tag{3.12}$$

Substitute (3.12) into (3.5), we now derive the learning behavior of the steepest descent method:

$$\begin{aligned}\xi(n) &= \xi_{min} + \sum_{i=0}^{K-1} \lambda_i v'^2_i(n) \\ &= \xi_{min} + \sum_{i=0}^{K-1} \lambda_i (1-2\mu\lambda_i)^{2n} v'^2_i(0).\end{aligned} \tag{3.13}$$

If μ is property selected, then (3.13) converges to ξ_{min} as n increases.

In (3.13), the MSE is consisted of the sum of K exponentially decaying terms each of which corresponds to one of the modes of convergence of the algorithm.

3.1.2: Eigenvalue spread

Each exponential term in (3.13) can be characterized by a time constant which is obtained as follows.

Let

$$(1-2\mu\lambda_i)^{2n} = e^{-n/\tau_i} \tag{3.14}$$

and define τ_i as the time constant associated with the exponential term $(1-2\mu\lambda_i)^{2n}$.

Solving (3.14) for τ_i, we get

$$\tau_i = \frac{-1}{2\ln(1-2\mu\lambda_i)}. \tag{3.15}$$

When $2\mu\lambda_i = 1$, $\ln(1-2\mu\lambda_i) \cong -2\mu\lambda_i$. Substitute this in (3.15), we obtain

$$\tau_i \cong \frac{1}{4\mu\lambda_i}. \tag{3.16}$$

We can see in (3.16) that large eigenvalue λ_i causes small time constant, so there is a faster convergence speed for the tap-weight coefficient $w_i, i = 0,1,...,K-1$. However, the MSE convergence is determined by the whole tap-weight coefficients, a single time constant cannot thoroughly govern the convergence speed. We observe this situation in (3.12). The necessary and sufficient condition of convergence is

$$\left(\mathbf{I}-2\mu\mathbf{\Lambda}\right)^{n}\rightarrow\mathbf{0}\text{, as n}\rightarrow\infty,$$

or
$$\begin{bmatrix} \left(1-2\mu\lambda_0\right)^{n} & 0 & \cdots & 0 \\ 0 & \left(1-2\mu\lambda_0\right)^{n} & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \left(1-2\mu\lambda_0\right)^{n} \end{bmatrix}\rightarrow\mathbf{0}\text{, as n}\rightarrow\infty. \quad (3.17)$$

So that if the maximum of these eigenvalues, say, λ_{max}, is large, and others are relatively small, the convergence speed will still be delayed by the other eigenvalues. Hence, the eigenvalue spread provides a good merit for the convergence speed. It is defined as

$$\text{Eigenvalue spread } \chi\left(\mathbf{R}\right)=\frac{\lambda_{max}}{\lambda_{min}}. \quad (3.18)$$

Eigenvalue spread is a condition number of matrix **R**. **R** is ill-conditioned if the eigenvalue spread is large. Some analysis [4], [5] have shown that the eigenvalue spread is relative to the spectrum of the input data sequence. The eigenvalue spread is bounded by the ratio of the maximum and the minimum of the input data spectrum and is approximately the same when the sequence length is long enough. In this observation, it is natural to conclude that the input data with a flatter spectrum has a smaller eigenvalue spread, and accordingly the adaptive filter will have a faster convergence speed.

3.1.3: LMS convergence behavior

Assume that the input signal $x(n)$ and the desired response $d(n)$ are zero-mean stationary process and are all jointly Gaussain-distributed random variables. And at time n, the tap-weight vector $\boldsymbol{w}(n)$ is independent of the input vector $\mathbf{x}(n)$ and the desired response $d(n)$. Based on these assumptions, we start to analyze the mean square error $E[e^2(n)]$ of the conventional LMS algorithm. Such analysis can be seen in [4], [5].

Recall (2.20), the update equation of LMS, subtracting $\mathbf{w_0}$ from both sides, we obtain:

$$\mathbf{v}(n+1) = \mathbf{v}(n) + 2\mu e(n)\mathbf{x}(n). \tag{3.19}$$

where $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w_0}$ is the weight-error vector. Also notes that

$$\begin{aligned} e(n) &= d(n) - y(n) = d(n) - \mathbf{w}^{\mathbf{T}}(n)\mathbf{x}(n) = d(n) - \mathbf{x}^{\mathbf{T}}(n)\mathbf{w}(n) \\ &= d(n) - \mathbf{x}^{\mathbf{T}}(n)\mathbf{w_0} - \mathbf{x}^{\mathbf{T}}(n)\left(\mathbf{w}(n) - \mathbf{w_0}\right) \\ &= e_o(n) - \mathbf{x}^{\mathbf{T}}(n)\mathbf{v}(n) \end{aligned} \tag{3.20}$$

where

$$e_o(n) = d(n) - \mathbf{x}^{\mathbf{T}}(n)\mathbf{w_0} \tag{3.21}$$

is the estimation error when the filter tap weights are optimum.

Squaring both sides of (3.20) and taking the expectation on both sides, we obtain

$$e(n) = E\left[e_o^2(n)\right] + E\left[\left(\mathbf{v}^{\mathbf{T}}(n)\mathbf{x}(n)\right)^2\right] - 2E\left[e_o^2(n)\mathbf{v}^{\mathbf{T}}(n)\mathbf{x}(n)\right]. \tag{3.22}$$

Noting that $\mathbf{v}^{\mathbf{T}}(n)\mathbf{x}(n) = \mathbf{x}^{\mathbf{T}}(n)\mathbf{v}(n)$ and using the independent assumption between the tap weights and the input signal, the second term on the right-hand side of (3.22) can be expanded as

$$\begin{aligned} E\left[\left(\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\right)^2\right] &= E\left[\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{v}(n)\right] \\ &= E\left[\mathbf{v}^{\mathrm{T}}(n)E\left[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\right]\mathbf{v}(n)\right] \\ &= E\left[\mathbf{v}^{\mathrm{T}}(n)\mathbf{R}\,\mathbf{v}(n)\right]. \end{aligned} \tag{3.23}$$

Noting that $\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)$ is a scalar, rewrite (3.5) as

$$\begin{aligned} E\left[\left(\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\right)^2\right] &= tr\left\{E\left[\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{v}(n)\right]\right\} \\ &= tr\left\{E\left[\mathbf{v}^{\mathrm{T}}(n)E\left[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\right]\mathbf{v}(n)\right]\right\} \\ &= E\left[tr\left\{\mathbf{v}^{\mathrm{T}}(n)\mathbf{R}\,\mathbf{v}(n)\right\}\right] \\ &= E\left[tr\left\{\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{R}\right\}\right] \\ &= tr\left\{E\left[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\right]\mathbf{R}\right\}. \end{aligned} \tag{3.24}$$

Here we use the matrix trace property: $tr\{\mathbf{AB}\} = tr\{\mathbf{BA}\}$ for facilitating the term exchanging operation. Define the correlation matrix of the weight-error vector $\mathbf{v}(n)$ as

$$\mathbf{K}(n) \equiv E\left[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\right], \tag{3.25}$$

the above result reduces (3.6) to

$$E\left[\left(\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\right)^2\right] = tr\left\{\mathbf{K}(n)\mathbf{R}\right\}. \tag{3.26}$$

Using the independent assumption and noting that $e_o(n)$ is a scalar, the last term on the right-hand side of (3.22) can be written as

$$E\left[e_o^2(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\right] = E\left[\mathbf{v}^{\mathrm{T}}(n)\right]E\left[\mathbf{x}(n)e_o^2(n)\right] = 0, \tag{3.27}$$

Using (3.26) and (3.27) in (3.22), we obtain

$$\xi = E\left[e^2(n)\right] = \xi_{min} + tr\left\{\mathbf{K}(n)\mathbf{R}\right\}. \tag{3.28}$$

where $\xi_{min} = E\left[e_o^2(n)\right]$, that is, the minimum mean-square error (MSE) at the filter output.

(3.28) can be written in a form that is more convenient for future analysis. Recall (3.2), by decomposing **R** and using the matrix trace term commute rule, we obtain

$$\xi = E\left[e^2(n)\right] = \xi_{min} + tr\left\{\mathbf{K}'(n)\mathbf{\Lambda}\right\}. \tag{3.29}$$

where $\mathbf{K}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{K}(n)\mathbf{Q}$. Furthermore, using (3.25) and definition **v'**(n) = $\mathbf{Q}^{\mathrm{T}}$ **v**(n) from section 3.1, we find that

$$\mathbf{K}'(n) \equiv E\left[\mathbf{v}'(n)\mathbf{v}'(n)^{\mathrm{T}}\right]. \tag{3.30}$$

Noting that $\mathbf{\Lambda}$ is a diagonal matrix composed of eigenvalues of **R**, (3.29) can be expanded as

$$\xi = E\left[e^2(n)\right] = \xi_{min} + \sum_{i=0}^{K-1} \lambda_i k'_{ii}(n) \tag{3.31}$$

where $k'_{ij}(n)$ is the *ij*th element of the matrix $\mathbf{K}'(n)$.

The LMS on average follows the same trajectory as the steepest-descent method [4]. Despites the noisy variation of the filter tap weight, the learning curve of the LMS matches closely the theoretical results of the steepest-descent method. To this end, (3.13) is applicable and the time constant

$$\tau_i \cong \frac{1}{4\mu\lambda_i} \tag{3.32}$$

can be used for predicting the transient behavior of the LMS algorithm. Consequently, the eigenvalue spread introduced in section 3.1.1.1, can be used to indicate the convergence speed of the LMS algorithm given an input data autocorrelation matrix.

3.2: Mismatched rate LMS convergence behavior

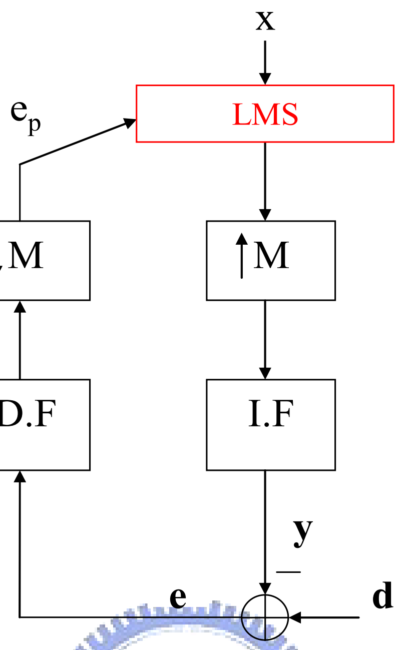

Figure. 3.1 Reduced complexity mismatched rate LMS filter.

Let the down sampled desired signal $d_p(n)$ as

$$d_p(n) = \mathbf{g_p d}(n),\ 1 \le p \le M \tag{3.33}$$

where $\mathbf{g_p} = \left[zeros(p-1)\ g_1\ \text{L}\ \ g_{N_D}\ zeros(M-p) \right],\ 1 \le p \le M$.

Rewrite the down sampled error (2.51) as

$$\begin{aligned} E\left[e_p^2\right] &= E\left[\left(\mathbf{g_p d} - \mathbf{g_p y}\right)^2\right] = E\left[\left(d_p - \mathbf{h_p X w}\right)^2\right] \\ &= E\left[d_p^2\right] - \mathbf{w}^{\mathrm{T}} E\left[\mathbf{X}^{\mathrm{T}} \mathbf{h}_{\mathrm{p}}^{\mathrm{T}} d_p\right] - E\left[d_p \mathbf{h_p X}\right]\mathbf{w} + \mathbf{w}^{\mathrm{T}} E\left[\mathbf{X}^{\mathrm{T}} \mathbf{h}_{\mathrm{p}}^{\mathrm{T}} \mathbf{h_p X}\right]\mathbf{w}. \end{aligned} \tag{3.34}$$

Define the cross correlation vector of the mismatched rate system as

$$E\left[\mathbf{X}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{p}}^{\mathrm{T}} d_p\right] \equiv \mathbf{q_p} \tag{3.35}$$

and the autocorrelation matrix as

$$E\left[\mathbf{X}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{p}}^{\mathrm{T}} \mathbf{h_p X}(n)\right] \equiv \mathbf{R_p} = \mathbf{R}_{\mathrm{p}}^{\mathrm{T}}. \tag{3.36}$$

Substitute (3.35) and (3.36) into (3.34), we can get:

$$E\left[e_p^2\right] = E\left[d_p^2\right] - 2\mathbf{w}^{\mathbf{T}}\mathbf{q}_{\mathbf{p}} + \mathbf{w}^{\mathbf{T}}\mathbf{R}_{\mathbf{p}}\mathbf{w},\ 1 \le p \le M\ . \tag{3.37}$$

Using the Wiener-Hopf equation (2.12), the optimum tap-weight is

$$\mathbf{w}_{\mathbf{o,p}} = \mathbf{R}_{\mathbf{p}}^{\mathbf{-1}}\mathbf{q}_{\mathbf{p}}\ . \tag{3.38}$$

With this optimum tap-weight, the convergence of this structure has an optimum to approach. Similar to (2.15), (3.37) can be written as

$$E\left[e_p^2\right] = \xi_{min,\,p} + \left(\mathbf{w} - \mathbf{w}_{\mathbf{o,\,p}}\right)^{\mathbf{T}} \mathbf{R}_{\mathbf{p}} \left(\mathbf{w} - \mathbf{w}_{\mathbf{o,\,p}}\right),\ 1 \le p \le M\ . \tag{3.39}$$

where $\xi_{min,\,p}$ is the minimum mean square error of the mismatched rate LMS training system and can be written as

$$\xi_{min,\,p} = E\left[d_p^2\right] - \mathbf{q}_{\mathbf{p}}^{\mathbf{T}}\ \mathbf{R}_{\mathbf{p}}^{\mathbf{-1}}\ \mathbf{q}_{\mathbf{p}},\ 1 \le p \le M\ . \tag{3.40}$$

One must notice that the lower case of "*p*" besides these symbols. It represents which phase of the error vector **e**(*n*) is chosen and the convergence of MSE and its MMSE are all relative to it. The effects of choosing different phase (performance, convergence speed) will be discussed in the next chapter.

Similar to the LMS convergence derivation, by replacing the autocorrelation matrix with $\mathbf{R}_{\mathbf{p}}$, we can obtain the behavior of the mean square value of e_p as:

$$\xi_p(n) = \xi_{min,\,p} + tr\left\{\mathbf{K}_{\mathbf{p}}(n)\mathbf{R}_{\mathbf{p}}\right\},\ 1 \le p \le M\ . \tag{3.41}$$

Here we define $\mathbf{v}_{\mathbf{p}}(n) = \mathbf{w}(n) - \mathbf{w}_{\mathbf{o,p}}$ is the weight-error vector with respect to phase *p*, and define the correlation matrix of the weight-error vector $\mathbf{v}_{\mathbf{p}}(n)$ as

$$\mathbf{K}_{\mathbf{p}}(n) \equiv E\left[\mathbf{v}_{\mathbf{p}}(n)\mathbf{v}_{\mathbf{p}}^{\mathbf{T}}(n)\right]. \tag{3.42}$$

Decompose $\mathbf{R}_{\mathbf{p}}$ as:

$$\mathbf{R}_{\mathbf{p}} = \mathbf{Q}_{\mathbf{p}}\mathbf{\Lambda}_{\mathbf{p}}\mathbf{Q}_{\mathbf{p}}^{\mathbf{T}}, \tag{3.43}$$

where $\mathbf{\Lambda_p}$ is a diagonal matrix consisting of the eigenvalues $\lambda_{0,p}, \lambda_{1,p} \cdots , \lambda_{K-1,p}$ of $\mathbf{R_p}$ and the columns of $\mathbf{Q_p}$ contain the corresponding orthonormal eigenvectors.

Using (3.43), we can write (3.42) as

$$\xi_p(n) = \xi_{min,\, p} + tr\left\{\mathbf{K_p}'(n)\mathbf{\Lambda_p}\right\},\ 1 \le p \le M \tag{3.44}$$

Notice that $\mathbf{v_p}'(n) = \mathbf{Q_p^T v_p}(n)$, then

$$\begin{aligned}\mathbf{K_p}'(n) &= E\left[\mathbf{v_p}'(n)\mathbf{v_p}'(n)^{\mathbf{T}}\right] \\ &= \mathbf{Q_p^T K_p}(n)\mathbf{Q_p}\end{aligned}. \tag{3.45}$$

(3.44) can be also written in scalar form as

$$\xi_p(n) = E\left[e_p^2(n)\right] = \xi_{min,\, p} + \sum_{i=0}^{K-1} \lambda_{i,p} k'_{ii,\, p}(n),\ 1 \le p \le M \tag{3.46}$$

where $k'_{ij,\, p}(n)$ is the *ij*th element of the matrix $\mathbf{K_p}'(n)$.

The above can be seen as the convergence behavior of the mismatched rate system, but for problems such as echo cancellation, equation (3.46) may not be able to provide us enough information about the echo attenuation level. For such problems, the "true error" we concern is the input of the down-sampling block, the error vector **e**(*n*). Unlike *e*$_p$(*n*), it only gives a partial information, the value of **e**(*n*) (norm) is the direct result of the interpolated filtered signal and the received signal. It gives us a more complete information about the system identification. Here, we take **e**(*n*) as our performance index and the rest of this chapter will be devoted to derive its convergence behavior.

Now, we focus on the derivation of $E\left[\left\|\mathbf{e}(n)\right\|^2\right]$.

Recall (2.48) and (2.49), we collect all *e*$_p$(*n*)s, $p = 1 \sim M$ in a vector $\mathbf{e_p}(n)$

$$\mathbf{e}_{\mathrm{p}}(n)=\begin{bmatrix} e_1(n) \\ \mathrm{M} \\ e_M(n) \end{bmatrix}=\mathbf{Ge}(n) \tag{3.47}$$

where

$$\mathbf{G}_{M\times(N_D+M-1)}=\begin{bmatrix} g_1 & g_2 & \mathrm{L} & g_{N_{D-1}} & g_{N_D} & & & \mathbf{0} \\ & \mathrm{O} & & & & & \mathrm{O} & \\ \mathbf{0} & & g_1 & g_2 & \mathrm{L} & & g_{N_{D-1}} & g_{N_D} \end{bmatrix} \tag{3.48}$$

is a convolution matrix filled with coefficients of the decimation filter.

In (3.47), we can see that

$$\mathbf{e}(n)=\mathbf{G}^{-1}\mathbf{e}_{\mathrm{p}}(n) \quad \text{if } \mathbf{G}^{-1} \text{ exits.} \tag{3.49}$$

But as we can see in (3.48), the dimension of the convolution matrix makes the existence of $\mathbf{G}^{-1}$ impossible. If we force a matrix **A** such that **AG = I**, we may find the left inverse matrix of **G**. However, to simplify this derivation and make it intuitive, we will not focus on the derivation of the left inverse matrix of **G**. Instead, we will study the convergence of each sampling phase of **e**(*n*) individually.

3.2.1: Convergence analysis of each phase

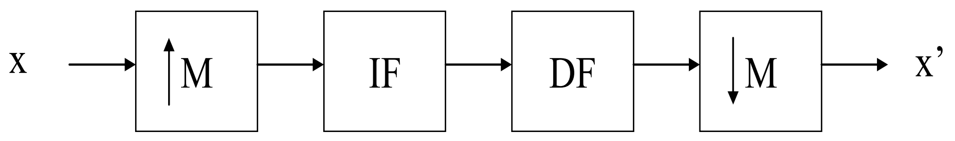

Figure. 3.2 Combination of the interpolator and the decimator.

Observing the path through the up-sampling and down-sampling block, a typical combination of the interpolator and the decimator can be plotted as shown in figure

3.2. In this figure, both the cut-off frequencies of the interpolation filter and the decimation filter are π / M. Such combination can be simplified by removing one of the low pass filters as we have known in the digital signal processing text book. The motivation of the following structure which help us to analyze the convergence of $\mathbf{e}(n)$ comes from this simplification.

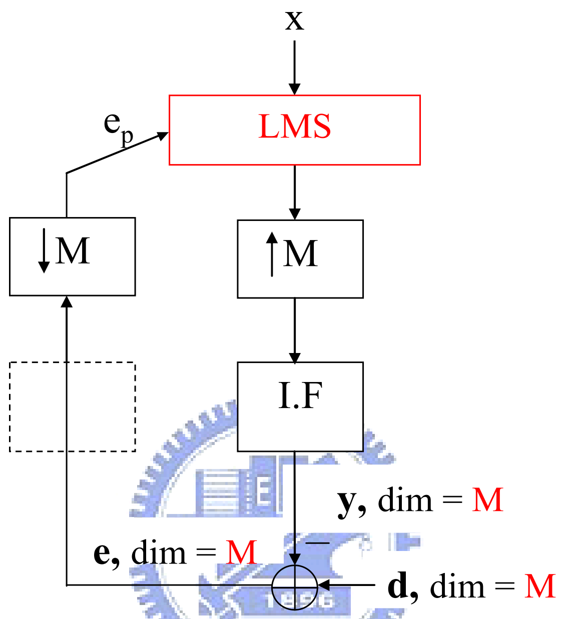

Figure. 3.3 A convergence analysis convenient structure.

The structure in figure 3.3 neglects the decimation filter. In this simplified structure, the convolution matrix shown in (3.48) is now simplified to an identity matrix with dimension *M*, and (3.49) now can be written as

$$\mathbf{e}(n) = \mathbf{e}_{\mathrm{p}}(n),\ 1 \le p \le M\ . \tag{3.50}$$

Hence, we can get

$$\begin{aligned} E\left[\left\|\mathbf{e}(n)\right\|^2\right] &= E\left[\mathbf{e}^{\mathrm{T}}(n)\mathbf{e}(n)\right] = E\left[\mathbf{e}_{\mathrm{p}}^{\mathrm{T}}(n)\mathbf{e}_{\mathrm{p}}(n)\right] \\ &= \sum_{p=1}^{M} E\left[e_p^2(n)\right]. \end{aligned} \tag{3.51}$$

Finally, substitute (3.46) into (3.51), the desired learning behavior can be written as:

$$
\begin{aligned}
E\left[\left\|\mathbf{e}(n)\right\|^2\right] &= \sum_{p=1}^{M} E\left[e_p^2(n)\right] \\
&= \sum_{p=1}^{M} \xi_{min,\,p} + \sum_{p=1}^{M}\sum_{i=0}^{K-1} \lambda_{i,\,p} k'_{ii,\,p}(n).
\end{aligned}
\tag{3.52}
$$

The analysis of the simplified LMS can be obtained using the same way. Modifying the autocorrelation matrix $\mathbf{R_p}$ with

$$
\hat{\mathbf{R}}_{\mathbf{p}} = E\left[\mathbf{X}^{\mathrm{T}}(n)\hat{\mathbf{h}}_{\mathbf{p}}^{\mathrm{T}}\hat{\mathbf{h}}_{\mathbf{p}}\mathbf{X}(n)\right],\ 1 \le p \le M\ , \tag{3.53}
$$

we can get the convergence behavior of the simplified LMS by applying eigenvalues of $\hat{\mathbf{R}}_{\mathbf{p}}$ to the methods listed above.

As we have mentioned before, one must notice that the performance of such a mismatched rate system without a decimation filter has a degraded performance due to the leakage of high frequency components. Hence the analysis of the structure without a decimation filter has a little difference to the structure with a decimation filter. However, given high SNR and no additional disturbance presented in high frequency, simulations in section 5.4 will demonstrate that the difference is not too much and the analysis introduced in this chapter is still practical in use for analyzing the mismatched rate reduced complexity structure.

Eigenvalues of the autocorrelation matrix $\mathbf{R_p}$ will be used to evaluate the eigenvalue spread and will be taken as a convergence speed index of this reduced complexity structure. One may see the experiment result in section 5.5 that the larger eigenvalue spread of $\mathbf{R_p}$ implies a slower convergence of $E\left[\left\|\mathbf{e}(n)\right\|^2\right]$ with decimation phase p.

Chapter4 Effects introduced by non-ideal re-sampling blocks

This chapter discusses the phenomenon of the performance difference when we choose different decimation phase p. The motivation to research the characteristics of all the decimation phases is from the derivation of convergence behavior. In practical use, it is enough to choose a "well-behaved" phase among these candidates. But the study of the other phases can provide us thorough information about how the finite length up/down sample filter affects the system performance.

Different selection of p may produce different response of $\mathbf{h_p}$. Because it is correlated with the update equations listed in chapter 2, when we accidentally choose a phase that suffers from distortion, the performance will consequently degrade. In figure 4.1, if the input x is a WSS (wide-sense stationary) signal, the output x' should be also a WSS signal after passing through the combined structure of M-fold interpolator and the M-fold decimator (We will prove this in section 4.1). However, some non-ideal effects of the intermediate low pass filter will distort this ideal all pass system.

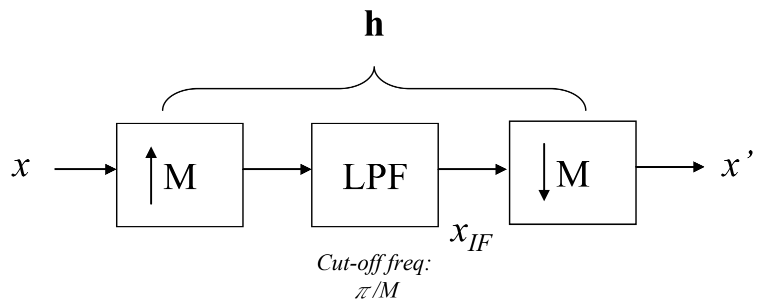

Figure 4.1 Combination of equal rate up and down sample blocks.

In this chapter, we will focus on distortion introduced by such non-ideal filter in LMS algorithm. After quantifying such distortion, we will provide a method to avoid the performance degradation.

4.1: Properties of random signals through a multi-rate system

Some basic conceptions are introduced when passing a random signal through a multi-rate system. We will show some statistical concepts about *x'* in figure 4.1 given a WSS (Wide Sense Stationary) signal *x*. In this section, all WSS and CWSS processes are assumed to be zero mean.

4.1.1: Preliminaries

A. *L*-fold Interpolator

Passing an input sequence *x*(*n*) into an *L*-fold interpolator results in

$$y_I(n)=\begin{cases} x(n/L), & \text{if } n \text{ is an integer multiple of } L \\ 0, & \text{otherwise.} \end{cases},$$

whose z transform is

$$Y_I(z)=X(z^L). \tag{4.1}$$

B. *M*-fold Decimator

Passing an input sequence *x*(*n*) into an *M*-fold decimator results in

$$y_D(n)=x(nM),$$

whose z transform is

$$Y_D(z)=\frac{1}{M}\sum_{k=0}^{M-1}X\left(z^{1/M}W_M^k\right).\ W_M=e^{-j2\pi k/M}. \tag{4.2}$$

C. Cyclo-WSS Process

Let $\mathbf{R}_{xx}(n,\ k)=E\left[x(n)\ x^*(n-k)\right]$ denote the autocorrelation function of *x*(*n*).

The process is said to be (CWSS)$_L$ if

$$E\left[x(n)\right]=E\left[x(n+kL)\right],\quad \forall n,\ k\text{, and}$$

$$\mathbf{R}_{xx}(n,\ k) = \mathbf{R}_{xx}(n+L,\ k),\quad \forall n,\ k.$$

D. Linear Periodically Time-Varying (LPTV) system

A system is said to be LPTV with period L (denoted as $(\text{LPTV})_L$) if the output $y(n)$ in response to input $x(n)$ can be written as

$$y(n) = \sum_{k=-\infty}^{\infty} h(n,\ k)\, x(n-k)$$

where the $h(\ n,\ k\ )$ is an LTI system with property

$$h(n,\ k) = h(n+L,\ k),\quad \forall n,\ k.$$

The LPTV system can be implemented as figure 4.2.

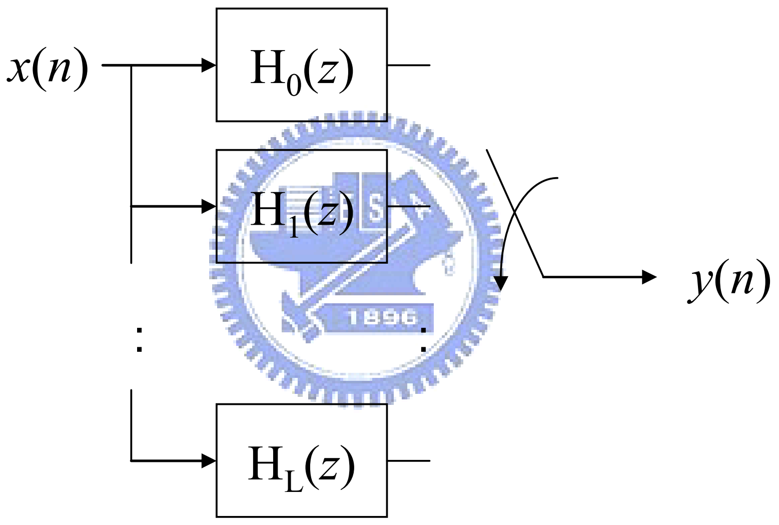

Figure 4.2: Implementation of an $(\text{LPTV})_L$ system.

Note: The output at time n is one of the L filters, its order is controlled by number (n *mod* L).

4.1.2: Properties of the signal passing through a multi-rate system

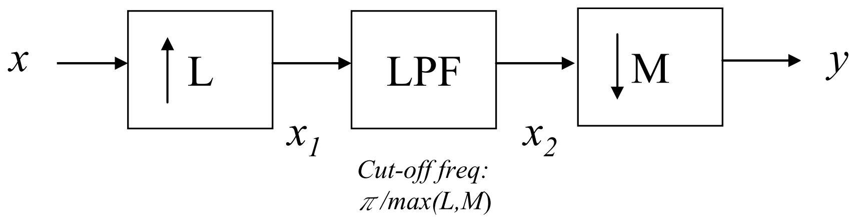

Figure 4.3: A general multi-rate filter.

For each output node in figure 4.3, we outline three important properties.

Property A:

If we pass a WSS signal $x(n)$ through an L-fold interpolator, the output $x_1(n)$ is CWSS with period L.

Property B:

If we pass a signal $x_1(n)$, $(\mathrm{CWSS})_L$, through an $(\mathrm{LPTV})_L$ system, the output $x_2(n)$ is CWSS with period L.

Property C:

If we pass a signal $x_2(n)$, $(\mathrm{CWSS})_L$, through an M-fold decimator, the output $y(n)$ is CWSS with period K, where $K = L / \gcd(L, M)$. gcd{,} is the greatest common factor operator.

In [3], the above properties have been proved and a theory is developed to make the output $y(n)$ to be WSS in general multi-rate systems.

Consider the multi-rate system in figure 4.1 in our case. From properties listed above, it is clear that for an equal up sampling and down sampling rate system, passing a wide-sense stationary signal x to the filter results in a wide-sense stationary signal x'.

4.2: Quantification of distortion

In this section, we will first find out the source of distortion. Then the method to evaluate the power of the non-ideal effect is presented. Finally, a frequency domain representation for each node of the reduced complexity structure is derived to provide a theoretical analysis of distortion. This theoretical result will be used to compare with the simulation of the actual distortion signal.

4.2.1: Source of distortion

Based on the conclusion in section 4.1, the output of the combination of the equal up-sampling and down-sampling rate blocks should be WSS given a WSS input signal. That is, the combined effect should be an all pass filter in frequency domain, and a delta function in time domain. However, some combined effects may produce impulse responses such as appearance of two peaks or asymmetry and distort all-pass property. Because insertion of zeros (interpolation) has nothing to modify, it is easy to assert that the distortion comes from the intermediate low pass filter and the decimation process. Reference to figure 4.1 and set $M = 2$, we show a brief demonstration of how such non-ideal side lobes of the interpolation filter affects in frequency domain.

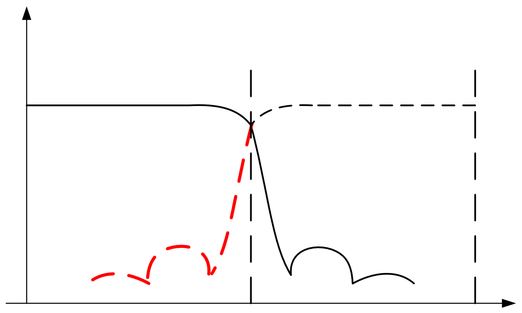

Figure 4.4: Brief demo for non-ideal IF side lobes effect. $M = 2$.

For M = 2, figure 4.4 shows that the side lobes belonged to the image of 0.5π ~ π affects the frequency components in the neighborhood of 0.5π. Mapping to our system, the filtered output **y** plays the same role as x_{IF} in this case. That is, we can see the distortion from the filtered output, or the direct identification result, **e**(n).

Distortion of the filtered output means that the adaptive filter operating band is polluted by adjacent image. This implies that the system identification cannot perfectly match the response of the desired signal. Such distortion will reflect on the performance. In section 5.7, we will demonstrate this situation in the spectrum point of view between the distorted filtered signal and the desired signal. Besides, the eigenvalue spread will be shown to point out the performance degradation. It is expected that the adaptive filter suffered from distortion has a larger eigenvalue spread.

4.2.2: Level of distortion

Next, we present our method to evaluate the distortion power level. Integration shown in (4.3) will be used for our evaluation.

$$P_d = \int_a^{\pi/M} \left| \Phi_y(\omega) - \Phi_d(\omega) \right| d\omega . \tag{4.3}$$

In (4.3), $\Phi_y(\omega)$ denotes the power spectrum of the distorted filtered signal in the steady state , and $\Phi_d(\omega)$ denotes the power spectrum of the corresponding desired response. Note that the only desired variation in (4.3) is the distortion power, so spectrum samples in our experiment are taken from the Fourier transform of the steady state in time domain where the noisy adaptation has a comparably smaller influence.

From the previous section, we know that the distortion occurs in the neighborhood of

π/M. So the upper bound of the (4.3) is π/M for sure. What we have to do now is to decide the lower bound a.

A distorted combined effect of $\mathbf{h_p}$ is illustrated below.

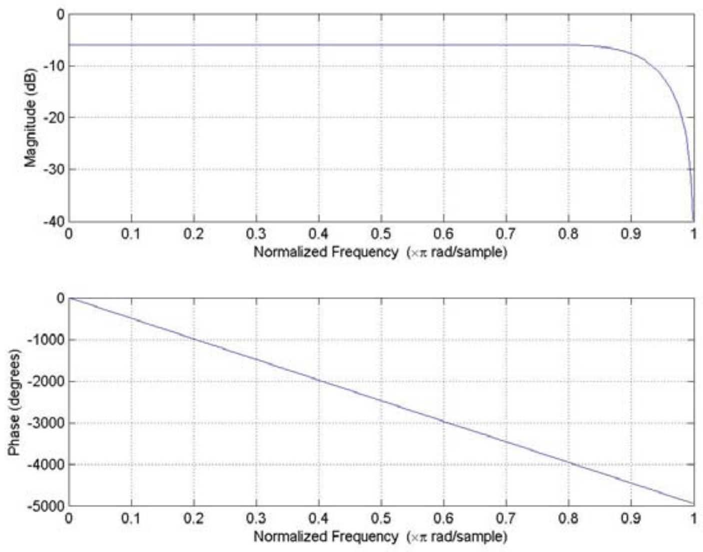

Figure 4.5: Frequency response of a distorted combined effect $\mathbf{h_p}$.

We can see that this distorted response has decay at high frequencies. By setting a discrimination threshold, the distorted interval of $\mathbf{h_p}$ is determined once the threshold is exceeded. We first set a default distorted portion for each rate M. The distortion portion defines a default distance between the upper bound and the lower bound of $\mathbf{h_p}$, the lower bound must not exceed the default interval to prevent a wrong estimation of the lower bound.

Because of the fact that the frequency response is stretched M times wider after the M-fold decimation, we can determine the distortion interval at the filtered output is $1/M$ of the estimating distortion interval of $\mathbf{h_p}$. Take figure 4.5 as an example. The estimated distortion interval of $\mathbf{h_p}$ is approximately 0.15π, so the effective distortion interval of the filtered output is approximately $0.15\pi/M$. In (4.3), the lower bound a is now set as $0.85\pi/M$ in this case.

Notice that if there's no distortion, $a = \pi/M$.

4.2.3: Frequency domain representation of the reduced complexity structure

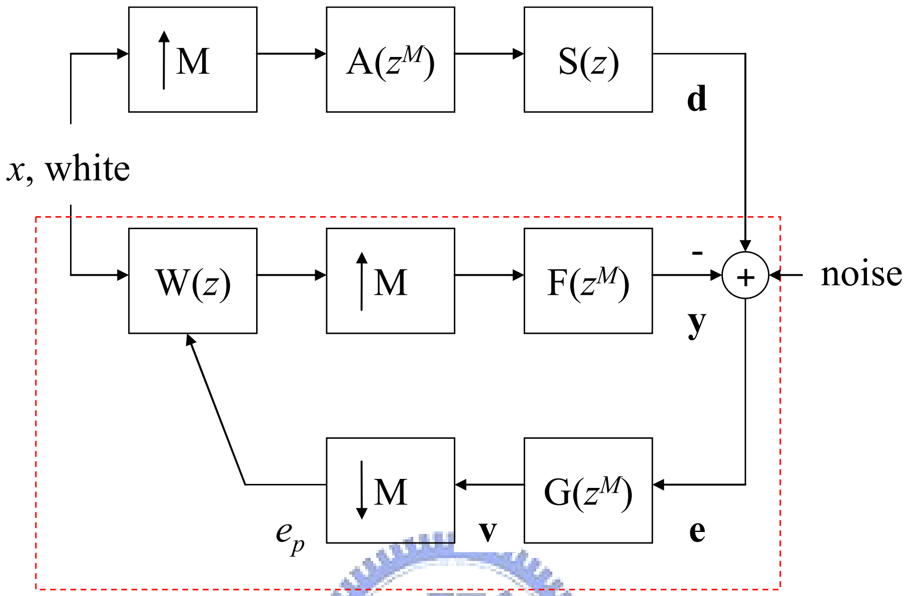

Figure 4.6: Complete system model in our case.

For the problem of echo cancellation, S(z) is replaced by the z-transform of the echo path. In figure 4.6, the echo path is modeled at the up-most branch, and the adaptive filter part is the other two branches bracketed with the dashed line. In our system, the echo is sampled at the Nyquist rate. So the echo path is operating in a bandwidth 2M times more narrow than the input signal and we up-sample the input signal to match this requirement. Here, A(z^M) is an interpolation filter for channel output.

Now we list each node in our system model in z-transform domain. Their transformations will be denoted in capital form.

Use (4.1), the z-transform of the desired signal **d** can be represented as

$$D(z) = A(z^M)S(z)X(z^M) \tag{4.4}$$

and the filtered signal **y** can be represented as

$$Y(z) = F(z^M) W(z^M) X(z^M). \tag{4.5}$$

The z-transform of the error signal **e** is

$$E(z) = D(z) - Y(z). \tag{4.6}$$

In section 5.5, we will use equation (4.6) to be our theoretical comparison.

In equation (4.5), we can see that we interpolate the filtered signal to match the desired signal in number. A signal after interpolation has images of the compressed version of the original signal distributed periodically in the high frequency domain. If the interpolation low-pass filter is not ideal, the frequency components beyond the cut-off frequency will not completely filter out. This will influence the performance because these partially suppressed images make the interpolated filtered signal unable perfectly fit the high frequency components of the desired signal. Such high frequency components can be considered as a smaller-level distortion (compared to the distortion in the neighborhood of π/M) in different mismatched rate systems. In section 5.5, spectrums of the filtered signals and the desired signal of different mismatched rates will be demonstrated. We will see that given different mismatched ratio M, different conditions of partially suppressed images will appear in the high frequency domain.

4.3: Filter design issue to avoid such degradation

In section 4.2.1, we know that the distortion introduced by the interpolation filter and the different decimation phase is the source, except the noise, that will degrade the performance of the LMS. Because the relatively weak response of $\mathbf{h}_\mathbf{p}$ in the high frequency, the adaptive filter cannot perfectly match the desired response. On the other hand, the distorted $\mathbf{h}_\mathbf{p}$ in the update equation is just like the coloring filter that correlates the input random signal in the conventional LMS. In our knowledge of the adaptive filter theory, such correlation will enlarge the eigenvalue spread and cause the convergence slower.

Since the combined response $\mathbf{h}_\mathbf{p}$ is determined once the intermediate low pass filter is set, we can evaluate its frequency response beforehand and choose a flat one to avoid such degradation caused by the imperfection design of filter.

Note that the flat frequency response $\mathbf{h}_\mathbf{p}$ can be found not only by changing the intermediate filter, but also by checking all the decimation phases for their corresponding $\mathbf{h}_\mathbf{p}$. In our simulations, even though there is a certain decimation phase that has larger distortion, there still exist other possible “well behaved” phases that have flat response of $\mathbf{h}_\mathbf{p}$ given the same intermediate filter. These “well behaved” phases will achieve better performance given the same parameters. If we evaluate the frequency responses $\mathbf{h}_\mathbf{p}$ of all the decimation phases, a good choice may be found among them.

In section 5.5, given M and the “possibly distorted” filter coefficients, we will demonstrate the coexistence of different performance among each decimation phase. A measurement which represents their corresponding performance difference, called DESR, will also be shown. We shall introduce it in the next section.

4.3.1: Distortion to pure Error Signal Ratio (DESR)

Finally, we remark a measurement of distortion. To quantify the performance difference between the "well behaved" phase and the distorted phase, we give a performance index, i.e., the Distortion to pure Error Signal Ratio (DESR). It is defined as follows.

Suppose phase 1 suffers from distortion. Then the MSE ratio between phase 1 and the MMSE (DESR) is:

$$DESR \equiv \frac{E\left[\mathbf{e}_1^2\right]}{E\left[\mathbf{e}_o^2\right]}. \tag{4.7}$$

Because $E\left[\mathbf{e}_o^2\right]$ provides the optimum solution for solving the system identification problem and the distortion is the only degradation of this performance difference, the distorted $E\left[\mathbf{e}_1^2\right]$ can be written as

$$E\left[\mathbf{e}_1^2\right] = E\left[\mathbf{e}_o^2\right] + E\left[\mathbf{e}_d^2\right]. \tag{4.8}$$

$E\left[\mathbf{e}_d^2\right]$ represents the distortion power.

Now DESR can be written as

$$DESR = \frac{E\left[\mathbf{e}_1^2\right]}{E\left[\mathbf{e}_o^2\right]} \cong \frac{E\left[\mathbf{e}_d^2\right]}{E\left[\mathbf{e}_o^2\right]} + 1. \tag{4.9}$$

From (4.13), if the performance is scaled in dB, we can see that the distance between $E\left[\mathbf{e}_1^2\right]$ and MMSE is the distortion power.

Chapter5 Simulations

In this chapter, promised simulations in the previous sections will be shown. Before entering this chapter, we shall give the environment specifications in our simulation.

Environment settings:

Channel: A 400 taps FIR with a low pass shape, cutoff frequency = *π/M/2* (Divide by 2 because we sample the channel at the Nyqusit rate). SNR = 150dB.

Adaptive filter settings:

	LMS parameters	RLS parameters
M=1	410 taps	410 taps
	$\mu = 0.002$	$\lambda = 0.98$
	IF order = 1	IF order = 1
M=2	205 taps	205 taps
	$\mu = 0.0078$	$\lambda = 0.985$
	IF order = 55	IF order = 37
M=4	105 taps	105 taps
	$\mu = 0.0313$	$\lambda = 0.975$
	IF order = 65	IF order = 45

Table 5.1: LMS and RLS filter settings

Both rates($M = 2, 4$) use Kaiser low pass filter with minimum -133 dB side-lobe attenuation of the Fourier transform of the window.

Note1: In our simulations, DF coefficients are set equal to the IF coefficients.

Note2: It is equal to the conventional adaptive algorithm if $M = 1$.

Note3: The performance index is the normalized mean square value of the vector $\mathbf{e}(n)$. We will rapidly run ten times to obtain an ensemble average.

Note4: "Covariance windowing" is used for input data sequence if no specification.

5.1: Original adaptive algorithm V.S. its simplified version

5.1.1: LMS algorithm

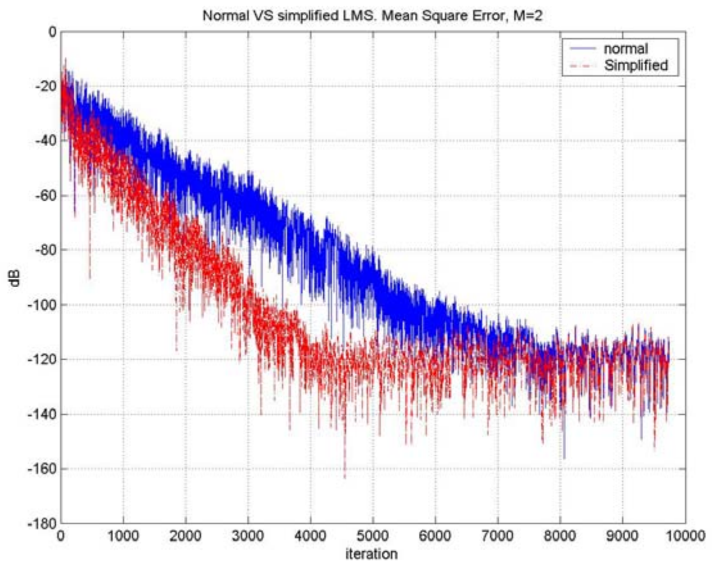

Figure 5.1.1: MSE of original LMS VS simplified version, $M = 2$.

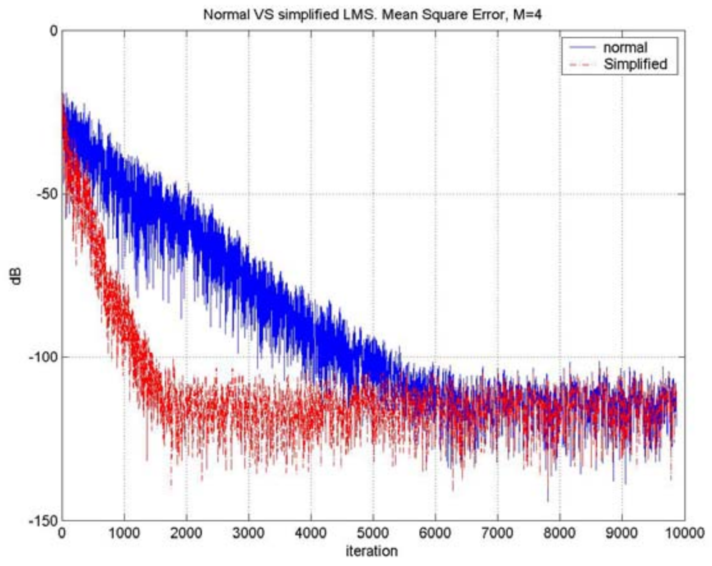

Figure 5.1.2: MSE of original LMS VS simplified version, $M = 4$.

We can see that the simplified version converges faster than the original one. As we have mentioned in section 2.2.2.1, the simplified version of the LMS achieves such speed by assuming $\mathbf{h_p}$ a delta function.

5.1.2: RLS algorithm

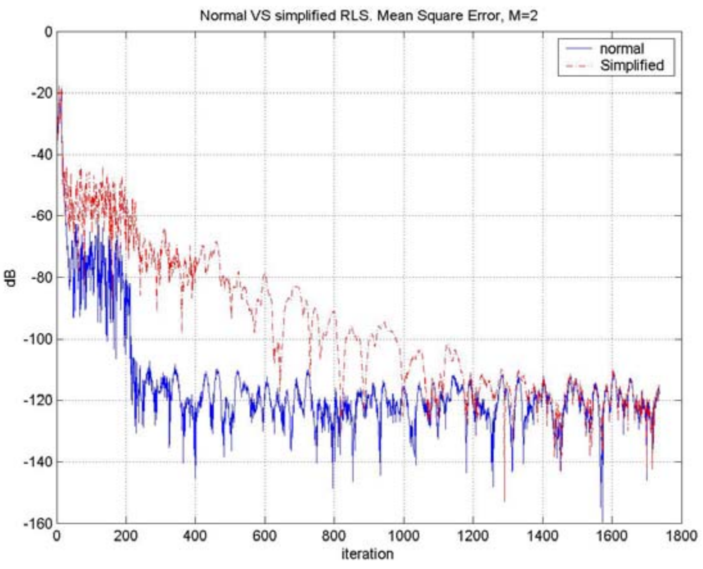

Figure 5.1.3: MSE of original RLS VS simplified version, $M = 2$.

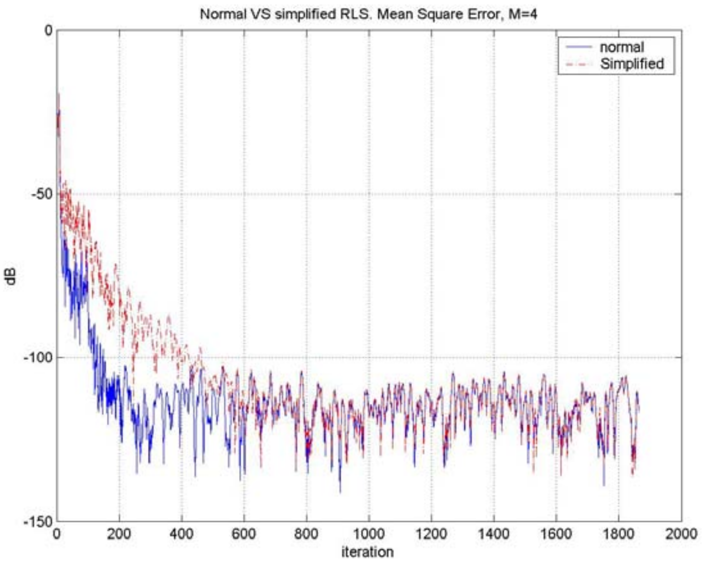

Figure 5.1.4: MSE of original RLS VS simplified version, $M = 4$.

For RLS, the situation is reversed. The reason is simple. Because RLS's fast convergence speed results from exact system modeling in deterministic form, it is natural to see that the original RLS algorithm, which provides a more complete modeling, converges faster than the simplified version.

5.2: RLS initializations.

Section 5.2.1 will demonstrate the initialization in the beginning, and section 5.2.2 will demonstrate the re-initialization problem.

5.2.1: Pre-windowing V.S. RLS using covariance windowing

Note1: Covariance windowing initializes the input data sequence with $\mathbf{h}_{\mathbf{p}}\mathbf{X}(n)$, where n is the latency time.

Note2: Notices that during the latency time, the sequence with initialization sets its error value as 1, and after 10 times ensemble average, the MSE during the latency becomes 0.1 = -10dB.

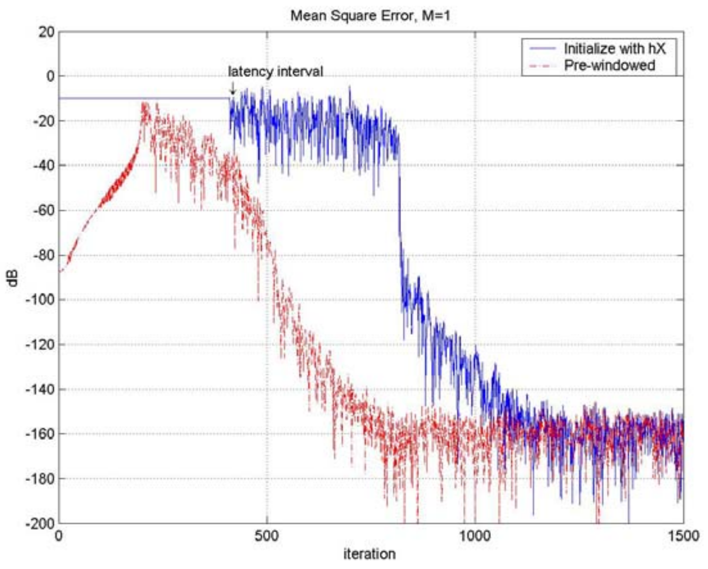

Figure 5.2.1: RLS using pre-windowing VS covariance windowing, $M = 1$, $n = 410$.

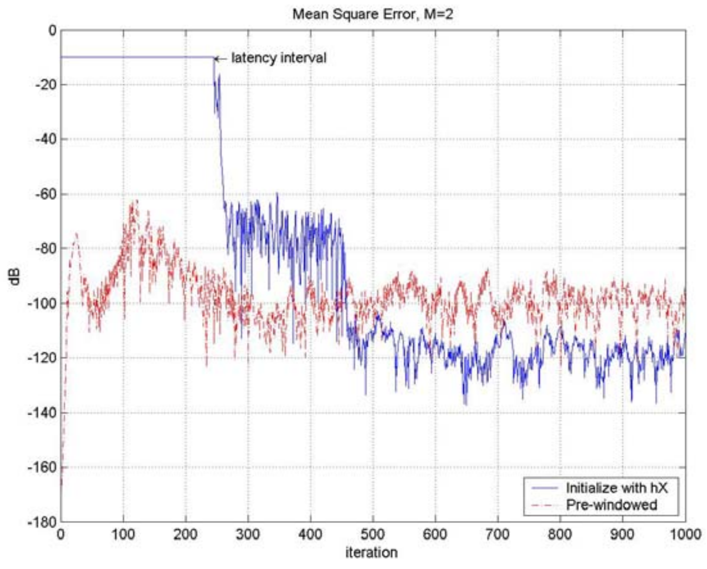

Figure 5.2.2: RLS using pre-windowing VS covariance windowing, *M* = 2, *n* = 264.

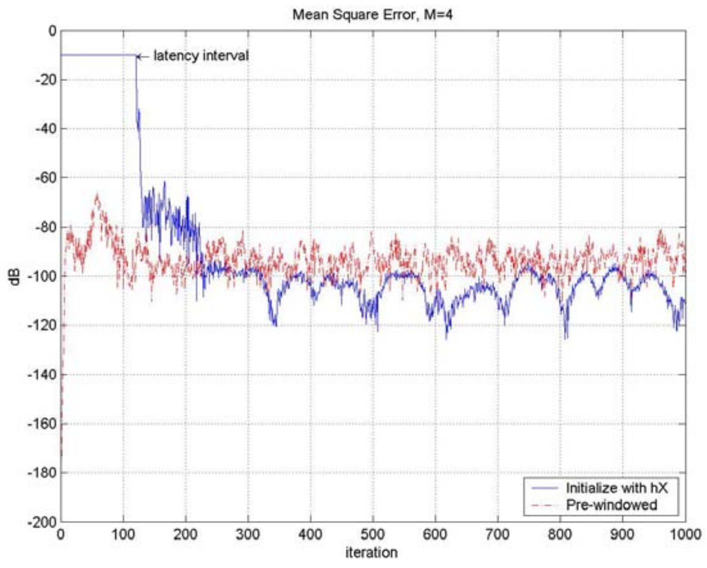

Figure 5.2.3: RLS using pre-windowing VS covariance windowing, *M* = 4, *n* = 137.

We can see in simulations that pre-windowing works as well as the covariance windowing in the matched rate system. But in the mismatched rate systems we can see that the covariance windowing is better. The latency is paid of worthy even though its convergence is late coming result.

5.2.2: Re-initialization for computation-released pre-windowed RLS

Note: RLS adapts 1 iteration and stop for several iterations. While stopping adaptation, one of the test sequence still keep on updating the latest input data with $\mathbf{h}\bar{\mathbf{x}}(n)$, and the other test sequence does nothing but shifting the latest input data with zero.

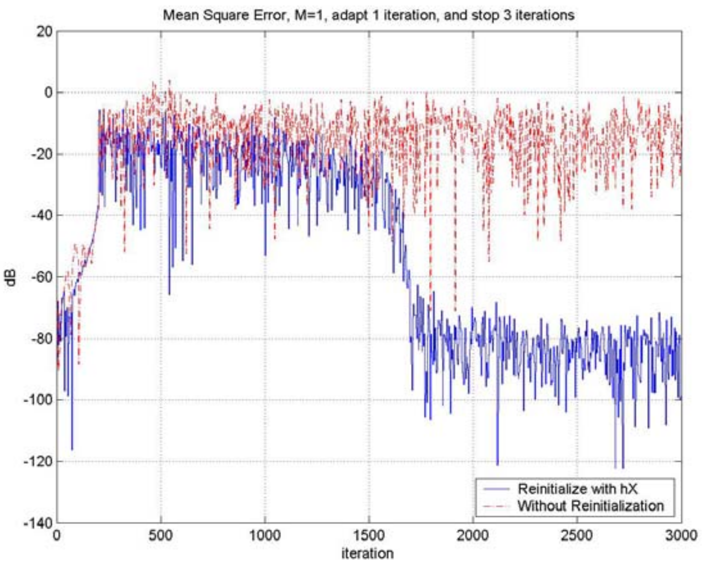

Figure 5.2.4: With VS Without re-initialization, *M*=1, stops 3 iterations.

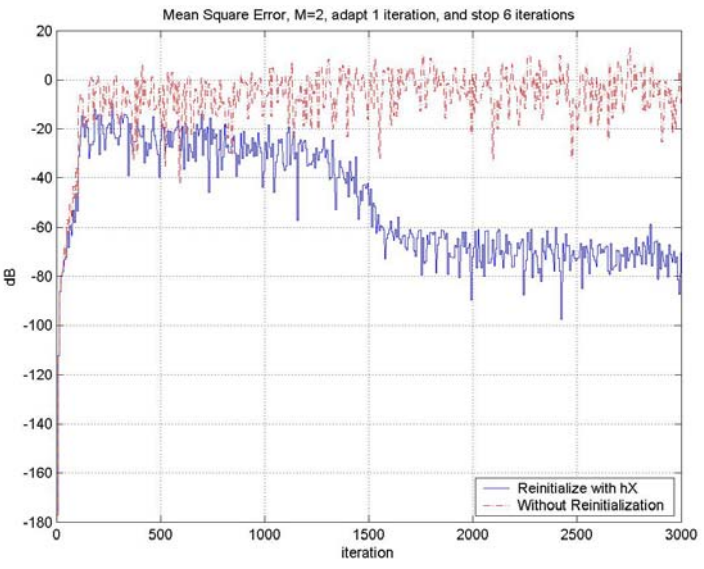

Figure 5.2.5: With VS Without re-initialization, *M*=2, stops 6 iterations.

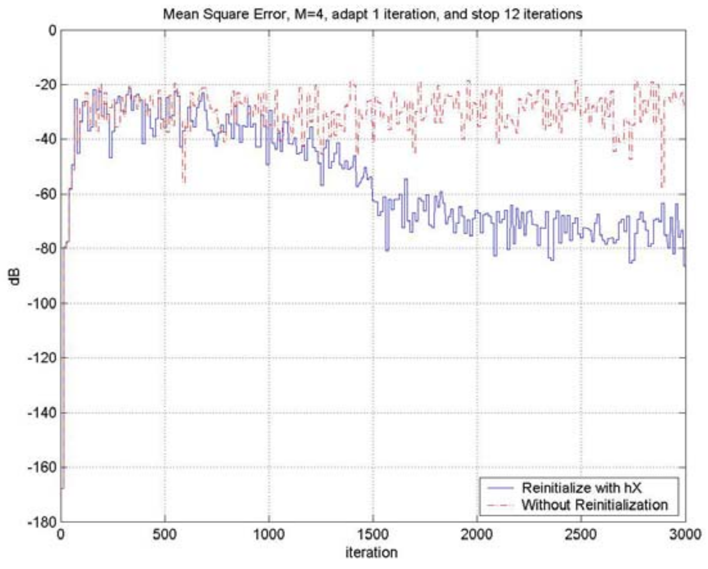

Figure 5.2.6: With VS Without re-initialization, *M*=4, stops 12 iterations.

We choose the number of stop iterations so that these experiments using re-initialization converge at about 1700 iterations for all *M*. The previous error record is kept during stop adaptation. This is why the granular-like curves appear these figures.

Our simulations explore two things. One is the obvious degrading performance and of the MSE if we periodically stop adapting the filter. This is reasonable since we sacrifice the performance to release the computation high tide. The other thing is sequence with continuously update its input data has a stronger resistance against the periodical stagnant adaptation thanks to its forgetting property of past data.

5.3: LMS V.S. RLS

Note: MSE comparison of both of the algorithms presented. We can look for literature [5] the similar comparison between LMS and RLS for M = 1. Here, we provide the mismatched versions for M = 2, 4.

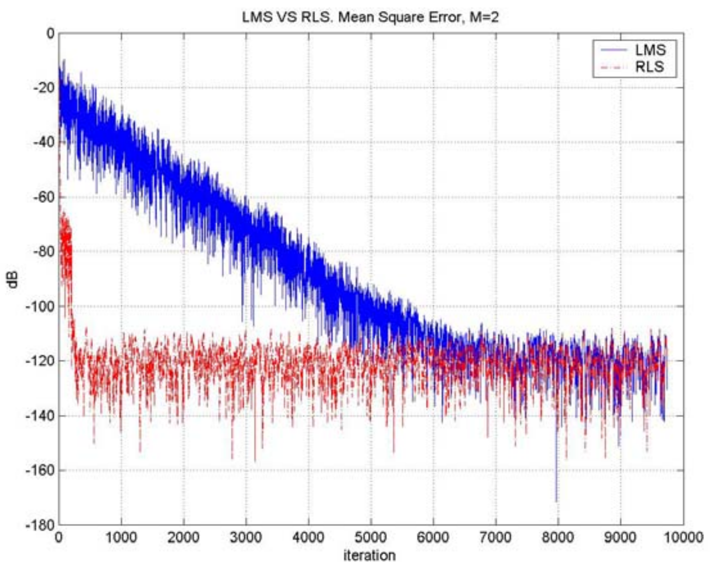

Figure 5.3.1: LMS VS RLS, $M = 2$.

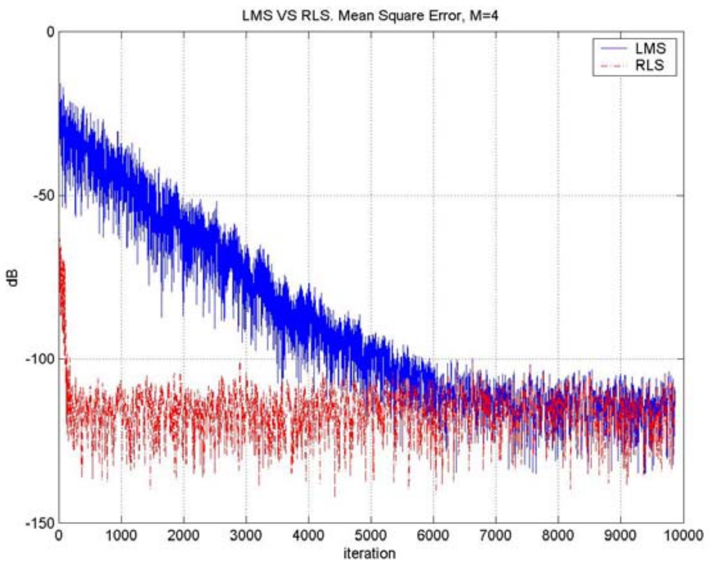

Figure 5.3.2: LMS VS RLS $M = 4$.

As expected, RLS converges faster than the LMS using the same input sequence in the cost of larger computation.

5.4: LMS performance comparison between structures with DF and without DF

Note: For a clear demonstration, we averaged 20 samples for an averaged MSE.

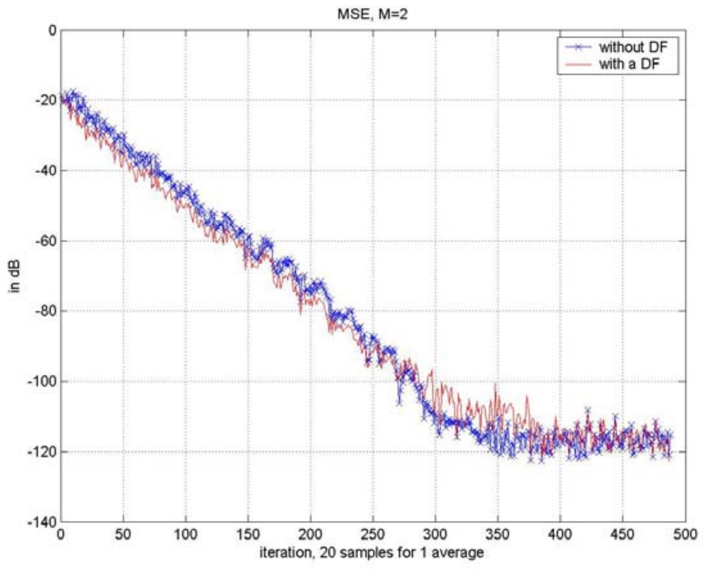

Figure 5.4.1: LMS using structures with DF and without DF, M=2.

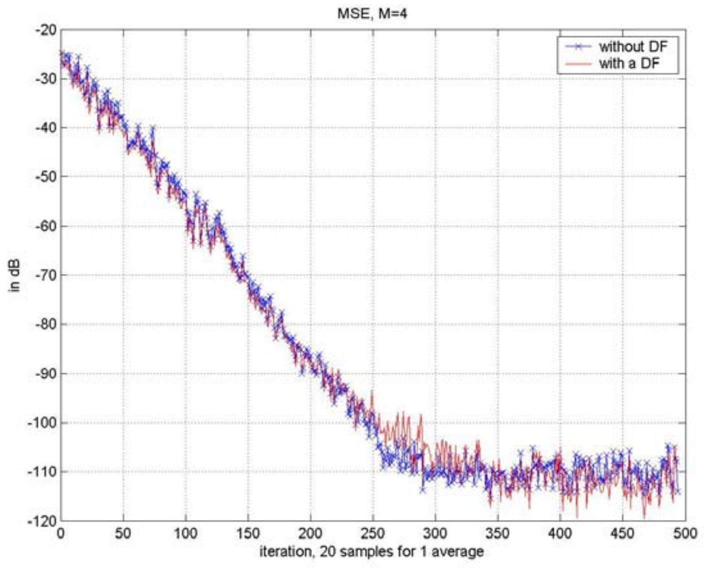

Figure 5.4.2: LMS using structures with DF and without DF, M=4.

Given high SNR and no additional disturbance presented in high frequency, the difference between the structure with a DF and without a DF is not significant. LMS convergence behavior of the analysis convenient structure presented in section 3.2.1 is now demonstrated to be similar to the actual reduced complexity structure.

5.5: LMS Different performance when sampling different phase

Note: For a clear demonstration, we averaged 20 samples for an averaged MSE.

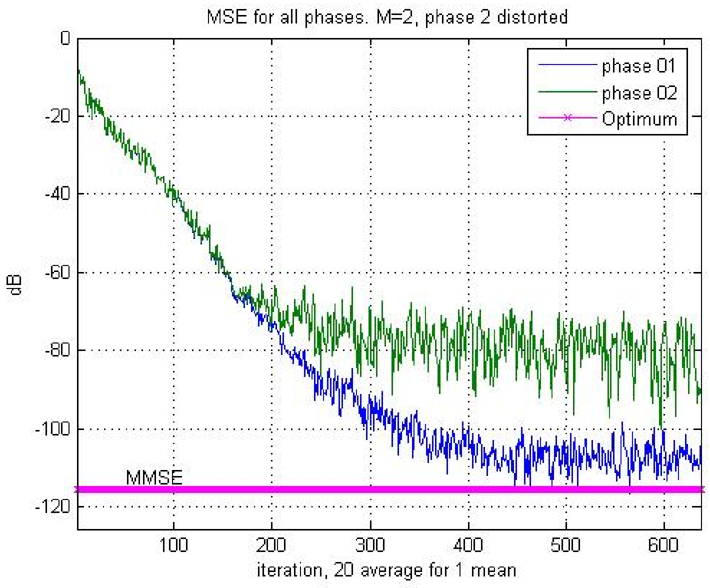

Figure 5.5.1: LMS, MSE of all phases, M=2, MMSE=-115.7071 dB

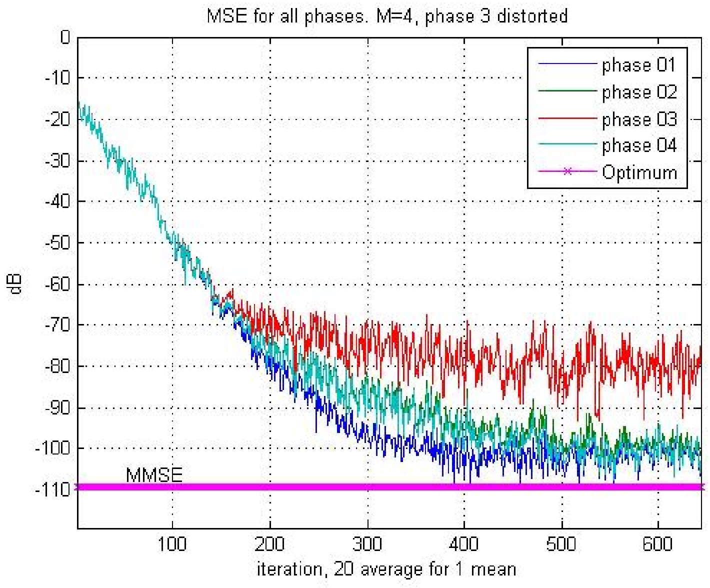

Figure 5.5.2: LMS, MSE of all phases, M=4, MMSE= -109.2926 dB

In these two plots, the distorted phase is phase 2 for M = 2 and phase 3 for M = 4. Their convergence is influenced by the non-ideal intermediate filter as we mentioned in chapter 4. The other phases are well behaved. In figure 5.5.2, we can see that the "well behaved" phases converge almost together.

In the following simulations, we will show their corresponding power spectrums of the filtered signal of each phase and the desired signal.

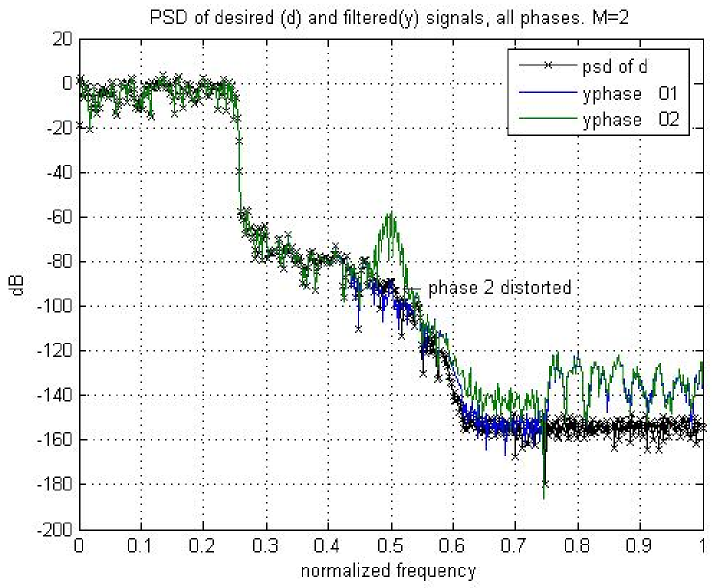

Figure 5.5.3: Demo of distortion in power spectrum, *M*=2

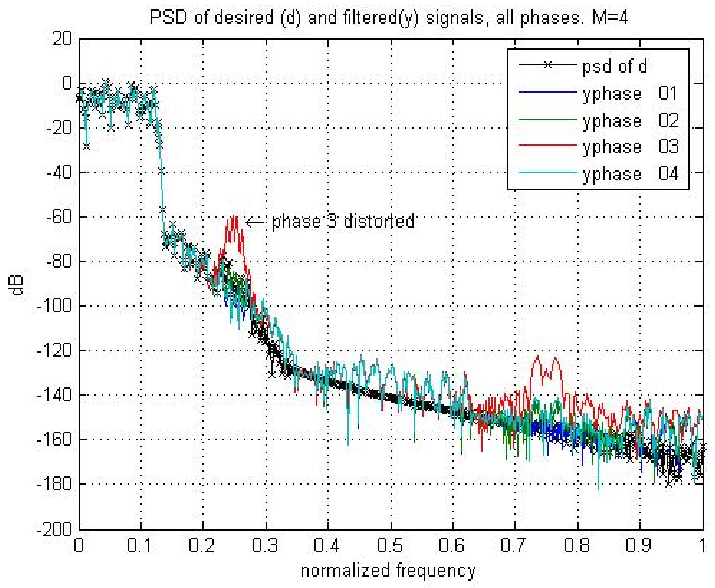

Figure 5.5.4: Demo of distortion in power spectrum, *M*=4

Notice the phase 2 in figure 5.5.3 and phase 3 in figure 5.5.4, the maximum distortion can be seen from the peaks appear in the neighborhood of frequency 0.5π and 0.25π in these two plots. Rest of the distortions in the high frequency components is the result of the partially suppressed images that we have mentioned in section 4.2.3.

Next, we draw out the spectrum of the error vector signals for each phase and the comparison theoretical curve. Note that the theoretical curve is a result that substitutes the tap-weights with the Wiener filter coefficients implicitly in equation (4.6).

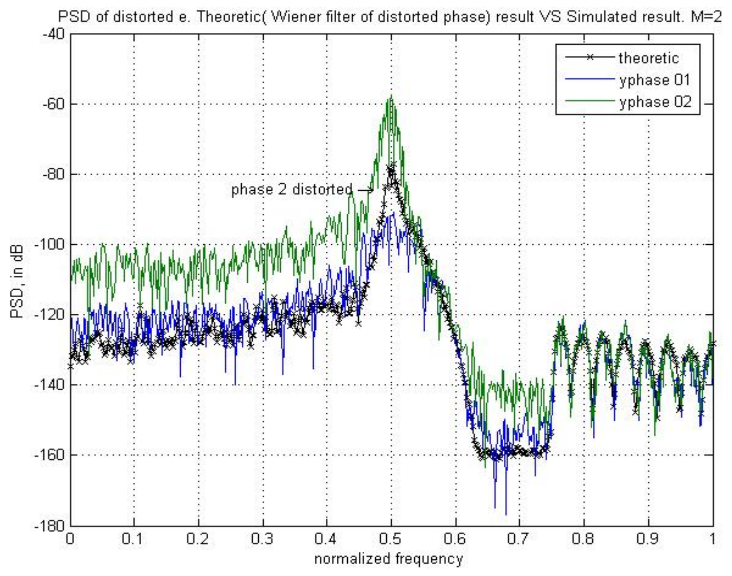

Figure 5.5.5: Theoretical distortion VS simulated distortion, M=2.

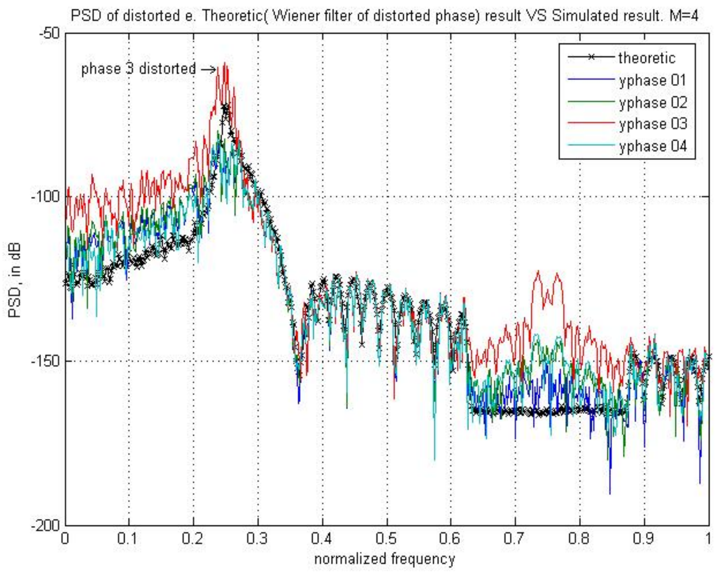

Figure 5.5.6: Theoretical distortion VS simulated distortion, M=4.

We can see that even substitute with the optimum tap-weights, phase 2 in figure 5.5.5 (phase 3 in figure 5.5.6) still suffers from distortion.

Some characteristics of the distortion phenomenon are summarized in table 5.2 and table 5.3.

Maximum distortion occurs on phase	2
Estimated distortion effective interval	0.40625π ~ 0.5π
Estimated distortion power of distorted phase	-80.5991 dB
DESR	35.1080 dB
MMSE	-115.7071 dB

Table 5.2: Features of distortion, $M = 2$.

Maximum distortion occurs on phase	3
Estimated distortion effective interval	0.17188π ~ 0.25π
Estimated distortion power of distorted phase	-84.6417 dB
DESR	26.5561 dB
MMSE	-109.2926 dB

Table 5.3: Features of distortion, $M = 4$.

The second row in table 5.2 and 5.3 is the estimated distortion effective interval in the neighborhood of π/M using the methods presented in section 4.2. With this result, the distortion power is calculated and listed in the third row. DESR using the estimated distortion power in these two cases is shown in the fourth row. On the other hand, DESR can be seen from the steady state MSE difference between the distorted phase and the MMSE shown in figure 5.5.1 and 5.5.2. Finally, the predicted minimum MSE values are listed in the last rows.

In table 5.4 and 5.5, we show that the distorted phase has a larger eigenvalue spread and a larger distortion power relative to the others. Their corresponding slower convergence is demonstrated in figure 5.7.1 and 5.7.2.

	Phase 1	Phase 2
Eigenvalue spread	43	1408
Distortion power (dB)	-102.7427	-80.5991

Table 5.4: Eigenvalue pread table, $M = 2$.

	Phase 1	Phase 2	Phase 3	Phase 4
Eigenvalue spread	12	20	942	20
Distortion power (dB)	-100.5926	-99.6247	-82.7365	-98.8549

Table 5.5: Eigenvalue spread table, $M = 4$.

Conclusion

The reduced complexity adaptive filter is designed based on minimizing $e_p^2(n)$, no matter in its statistical property (LMS) or in exact value (RLS), it all operates at a lower rate. In this paper, it is shown that the complexity of the reduced complexity system is similar to that of the conventional adaptive algorithm and the complexity will not grow proportional to the quadratic value of the mismatched rate.

The LMS convergence behavior for this mismatched rate reduced complexity model is presented. For this system, several issues such as the colored effect $\mathbf{h_p}$ introduced by imperfection of the intermediate filter and the RLS initialization problems have also been discussed. Such problems were addressed in our simulations and we have provided a possible solution for them. On the other hand, the simulations are conducted to illustrate these two adaptive algorithms in their original version and simplified version. A comparison between LMS and RLS is also presented.

Appendix A: Matrix differentiation conventions

Given a matrix $\mathbf{R}$ and an n by 1 column vector $\mathbf{x} = \left[x_1 \; x_2 \; \cdots \; x_n \right]^{\mathrm{T}}$, the conventions can be written as:

$$\frac{\partial}{\partial \mathbf{x}} \left(\mathbf{Rx} \right) = \mathbf{R}^{\mathrm{T}} \tag{a1}$$

$$\frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x}^{\mathrm{T}} \mathbf{R} \right) = \mathbf{R}. \tag{a2}$$

Using (a1), (a2) and the chain rule, quantities in [] are considered constant, we have:

$$\frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x}^{\mathrm{T}} \mathbf{Rx} \right) = \frac{\partial}{\partial \mathbf{x}} \left(\left[\mathbf{x}^{\mathrm{T}} \mathbf{R} \right] \mathbf{x} \right) + \frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x}^{\mathrm{T}} \left[\mathbf{Rx} \right] \right) = \left[\mathbf{x}^{\mathrm{T}} \mathbf{R} \right]^{\mathrm{T}} + \left[\mathbf{Rx} \right] = \left(\mathbf{R}^{\mathrm{T}} + \mathbf{R} \right) \mathbf{x} \tag{a3}$$

$$\text{If } \mathbf{R} \text{ is symmetric, } \frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x}^{\mathrm{T}} \mathbf{Rx} \right) = \mathbf{2Rx} \tag{a4}$$

Appendix B: Matrix inversion lemma

Suppose B and D are invertible matrix, then:

$$\left(\mathbf{B} + \mathbf{CDC}^{\mathrm{T}} \right)^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1} \mathbf{C} \left(\mathbf{D}^{-1} + \mathbf{C}^{\mathrm{T}} \mathbf{B}^{-1} \mathbf{C} \right)^{-1} \mathbf{C}^{\mathrm{T}} \mathbf{B}^{-1} \tag{a5}$$

References

[1] Alper Erdogan, Bijit Halder, Tzu-Hsien Sang and Ahmet Karakas, "*Efficient implementation of echo cancellation for applications with asymmetric rates*," (ICASSP '03). 2003 IEEE International Conference on, Volume: 6, 6-10 April 2003. Pages: VI - 233-6 vol.6.

[2] N. Kalouptsidis, S. Theodoridis, *Adaptive System Identification and Signal Processing Algorithms, chapter 5,* Prentice Hall international series in acoustics, speech, and signal processing, 1993.

[3] Sathe, V. and Vaidyanathan, P. P. "*Effect of multirate systems on the statistical properties of random signals*," IEEE Trans. On Signal Processing, vol. ASSP-41, March 1993.

[4] B. Farhang-Boroujeny, *Adaptive Filters Theory and Applications*, John Wiley & Sons, 1998.

[5] Simon Haykin, *Adaptive Filters Theory Fourth Edition*, Prentice-Hall Inc., New Jersey, 2002.

[6] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*, Prentice Hall Signal Processing Series, 2003.