# Adaptive Equalization for Periodically Varying Fading Channels

**Qadri A. A. Mayyala**

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Electrical and Electronics Engineering

Eastern Mediterranean University
January 2012
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Electrical and Electronic Engineering.

Assoc. Prof. Dr. Aykut Hocanın
Chair, Department of Electrical Electronic and Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Electrical and Electronic Engineering.

| | |
|---|---|
| Assoc. Prof. Dr. Aykut Hocanın<br>Co-Supervisor | Prof. Dr. Osman Kukrer<br>Supervisor |

Examining Committee

1. Prof. Dr. Hüseyin Özkaramanlı
2. Prof. Dr. Osman Kukrer
3. Prof. Dr. Runyi Yu
4. Assoc. Prof. Dr. Aykut Hocanın
5. Assoc. Prof. Dr. Hasan Demirel

# ABSTRACT

The problem of identification and tracking of periodically varying systems is considered. Multipath fading channel imposes significant constraints and limitations on wireless communication applications. When the multipath is caused by a few strong reflectors, the channel behaves as a system with poly-periodically time-varying response. The channel impulse response is then modeled by a linear combination of a finite set of complex exponentials whose frequencies are termed by Doppler frequencies. This model is well-motivated in radio cellular telephony and aeronautical radio communication.

While the system coefficients start varying rapidly in time, the commonly used adaptive least mean squares (LMS) and weighted least squares (WLS) algorithms are unable to track the variations effectively. The key point is to employ basis functions (BF) expansion algorithms, which are more specialized adaptive filters. Unfortunately, this type of estimators is numerically very demanding and has a limited mean square estimation error (MSE) performance.

This thesis explores two existing adaptive equalization algorithms, namely, exponentially weighted basis function (EWBF), gradient basis function Gradient-BF, and contributes by proposing a new efficient BF estimator termed as recursive inverse basis function (RIBF) estimator. Furthermore, a frequency-adaptive version of RIBF estimator is derived. Computer simulations are carried out, using Matlab software package, to evaluate the proposed RIBF estimator performance. The new BF estimator outperforms the EWBF estimator by large computational complexity savings. Moreover, RIBF is superior to the Gradient-BF and EWBF estimators since it shows further reduction in the mean square parameter estimation error. These advantages results in significant gains when applied in wireless communications to reduce BER, SNR and channel bandwidth requirements.

**Keywords:** Basis function algorithms, systems identification, nonstationary processes, periodically varying systems, adaptive filters.

# ÖZ

Bu tezde periyodik olarak değişen sistemlerin tanınma ve takib edilme problemi üzerinde durulmuştur. Kablosuz iletişim uygulamalarında çok yollu sönümlü kanal problemi ciddi sınırlamalar getirir. Böyle durumlarda kanalın darbe tepkimesi, frekansları Doopler frekansı olan sınırlı sayıda karmaşık eksponansiyelin doğrusal bileşimi ile modellenebilir.

Bu uygulamalarda kullanılan en az ortalama kareler ve ağıtlıklı en az kareler algoritmaları sistem parametrelerindeki hızlı değişimleri takip etmede yetersiz kalmaktadır. Temel işlev açılım algoritmaları bu durumlarda daha başarılı olmakla birlikte, gerektirdikleri sayısal işlemlerin çok olması ve kestirim hata başarımlarının sınırlı olması bu algoritmaların temel zorluklarıdır.

Bu tez var olan bazı uyarlamalı eşitleme algoritmalarını araştırır. Bunlar ekponansiyel ağırlıklı temel işlev (EWBF) ve eğim temel işlev (Gradient-BF) algoritmaları ve bunların türevleridir. Ayrıca yeni olarak dönüşümşü tersleme temel işlev (RIBF) algoritması önerilmiş, ve frekans-uyarlamalı türevi elede edilmiştir. Yeni önerilen algoritmaların var olanlara göre, daha düşük hesaplama karmaşıklığı ve daha düşük kestirim hatası bakımından daha avantajlı oldukları benzetim yolu ile gösterilmiştir.

**Anahtar Kelimeler:** Temel işlev algoritmaları, sistem tanıma, durağan olmayan süreçler, periyodik olarak değişen sistemler.

To:

My Beloved Country, Palestine

# ACKNOWLEDGMENT

I would like firstly to express my great thanks to my supervisors, Prof. Dr. Osman Kukrer for his wonderful and boundless help and guidance during my research, and Assoc. Prof. Dr. Aykut Hocanın, Chairman of the Department of Electrical and Electronics Engineering, Eastern Mediterranean University, for his great contribution and immense support in my thesis. I appreciate his kind help in many issues during this stage.  I admire their vast knowledge and enthusiasm of doing research.

I would like also to express my gratitude to the Electrical and Electronic Engineering Department staff, which they nourish me with a great knowledge during this stage.

I convey my special thanks to my friends and colleagues specially Nazzal, Abu Swuan, Nurellari and Opeyemi.

My most special thanks goes to my family in Palestine, my great parents, who always gives me their full capability to support me in every aspects of my life, another warm thanks goes to my fantastic siblings which I have, specially Assist. Prof. Dr. Samer for his generous support throughout my life aspects. I’m really indebt to all of them.

I would like to dedicate this study to all of the above mentioned, as well as to my lovely fiancé who inspires my life. Without all of them; I would not have been able to do this.

I do not forget that, the first and last thanks is all for our God.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| AWGN | Additive white Gaussian noise |
| BER | Bit error rate |
| BF | Basis Function |
| DCT | Discrete cosine transform |
| DFE | Decision feedback Equalization |
| DFT | Discrete Fourier transform |
| EWBF | Exponentially weighted basis function |
| FB | Fixed basis |
| FDLMS | Frequency-domain least mean square |
| FIR | Finite impulse response |
| FSE | Fractionally spaced equalizer |
| ISI | Intersymbol Interference |
| KLT | Karhunen-Loeve transform |
| LMS | Least mean square |
| LTE | Linear Transversal Equalizer |
| MA | Moving average |
| MLSE | Maximum likelihood sequence estimation |
| MSE | Mean square error |
| NLMS | Normalized least mean square |

| | |
|---|---|
| QAM | Quadrature amplitude modulation |
| RB | Running basis |
| RI | Recursive inverse |
| RIBF | Recursive inverse basis function |
| RLS | Recursive least squares |
| TDLMS | Transform-domain least mean square |
| VA | Virterbi algorithm |
| WLS | Weighted least squares |
| $\boldsymbol{\alpha}$ | Vector of coefficients |
| $\mathbf{f}(s)$ | Vector of basis functions |
| $\boldsymbol{\theta}(s)$ | Vector of periodically varying system coefficients |
| $\boldsymbol{\psi}(s)$ | Forward generalized regression vector |
| $\boldsymbol{\zeta}(s)$ | Backward generalized regression vector |
| $\mathbf{R}_{+}(s)$ | Forward-time average input correlation matrix |
| $\mathbf{R}_{-}(s)$ | Backward-time average input correlation matrix |
| $\mathbf{P}_{+}(s)$ | Forward-time average input/desired crosscorrelation vector |
| $\mathbf{P}_{-}(s)$ | Backward-time average input/desired crosscorrelation vector |
| $\boldsymbol{K}_{+}(s)$ | Forward gain factor |
| $\boldsymbol{K}_{-}(s)$ | Backward gain factor |

# Chapter1

# INTRODUCTION

## 1.1 Adaptive Equalization

Digital transmission systems such as voice and video communications are superior to analog transmission in mainly due to its higher reliability in noisy environments. However, a majority of digital communication transmission is faced with a common phenomenon known as intersymbol interference (ISI). In this case, the received data (pulses), which correspond to different symbols, are not completely separated. They are smeared out and the separation depends on the transmission media which causes the ISI. The transmission media are mainly cable lines or cellular communication.

Traditionally, either static or dynamic equalizers solve the problem of amplitude and phase dispersion that results in ISI of the received signals. Hence, it is clear that careful equalization is required whenever we need to employ a reliable digital transmission system.

Commonly, the design of any communication system, that includes a transmitter and a receiver, is build on the assumption that the channel transfer function is completely known. However, in most communication systems, the channel transfer functions are not completely known in order to allow us to design filters to eliminate the channel effects. The time varying nature of the channel in cellular communication makes the transfer function change with time (non stationary environment). It is therefore difficult to incorporate static filters to deal with such cases. To solve these problems, adaptive equalizers are designed to work in such environment to decrease Bit Error Rate (BER).

Simply saying, the main task of the equalizer is to cancel the effects of ISI and channel noise that is present in the channel. We may ignore the effects of the channel noise since the main challenge is to utilize the bandwidth as much as possible. In general, equalizers estimate the inverse of the channel impulse response and apply to the received signal as all filters do. Then combination of both channel and equalizer will give a flat frequency response and linear phase [1]. As shown in Figure (1.1).

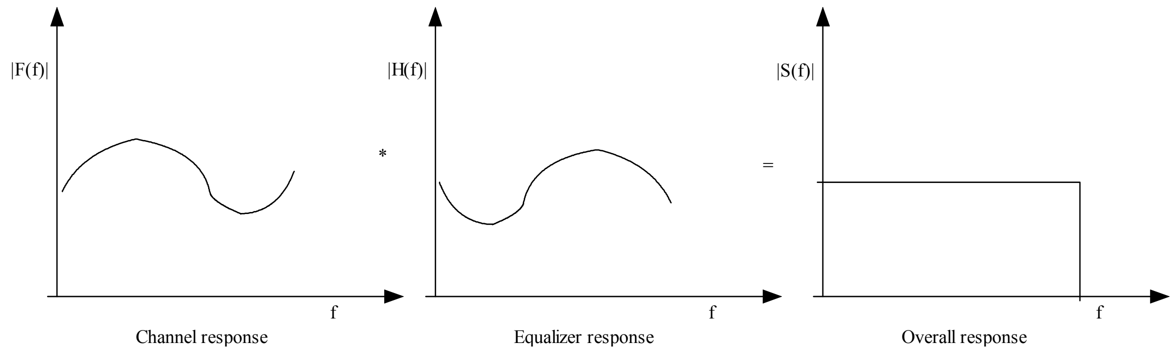

Figure 1.1: Concept of Equalizer [13]

However, the static equalizer shows superiority in terms of price and easiness to design, but its noise performance is not very good. As mentioned earlier both the channel and system transfer function might not be known, and possibly, the channel impulse response will differ with time. At this point static equalizer may fail to cancel the channel effects and BER would increase consequently. Hence, the need for adaptive equalizer comes on scene. An adaptive equalizer is an equalizer filter that automatically adapts to time varying properties of communication channel [4].

## 1.2 Channel Equalization Modes

In high transmission rate demand, the intersymbol interference (ISI) problem obviously becomes serious problem that needs a special attention. Channel Equalization technique solves such a problem by designing an equalizer that have an impulse response, as the combination of the channel and equalizer is close to the original symbols. The channel or the transmitter impulse is mostly unknown or varying with time as said earlier. Therefore, the need for adaptive equalizer becomes necessary to track channel characteristics.

Channel equalization system should complete two modes [4] in order to reconstruct desired signal from the distorted one *x* (*n*) as shown in figure (1.2) below:

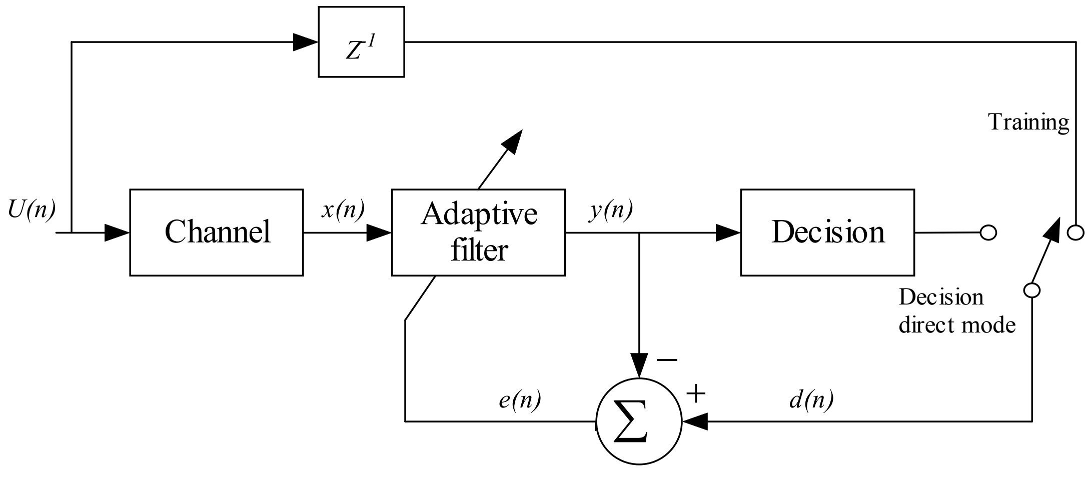

Figure 1.2: Digital Transmission System Using Channel Equalizer [13]

The two modes are as follows:

- Training Mode: this mode starts before the regular data transmission, in order to help in discovering adaptive filter coefficients which are considered as a channel impulse response for the distorted version signal *x* (*n*) of the original one *u*(*n*). The delayed version of the transmitted signal considered as the desired response *d* (*n*) for the adaptive filter need to be compared with the current output *y* (*n*) to get minimized error *e* (*n*) difference between both of them. The idea here is that the adaptive filter proceed iteratively and adjusts its coefficient to be maintained closer as possible to the unique optimum set of tap coefficients. After the convergence of the error according to some criteria as will be discussed later *y* (*n*) should be close as much as possible to *d* (*n*), and this means that the adaptive filter coefficient can be used as a compensator for the distorted signal.

- Decision-Direct Mode: this mode could be considered as steady state mode of the adaptive channel equalization system. After getting the suitable coefficients of the adaptive filter. The equalization process switches to Decision-direct mode in order start compensating the received signals in a proper way. Afterword, the system may be able to go back again to check if any change has happened on the channel

## 1.3 Modeling of the Multipath Fading Channels

Usually, to describe fading channels, the impulse response of the channels should be modeled by stochastic process approach. In this thesis, we used a deterministic approach which shows to be more suitable for our application. In this part we showed that the fading channels can be represented as a Moving Average (MA) model. Under some approximations, the land and aeronautical mobile radio channels both are shown to present poly-periodic time variations, and the frequency of the prior time variation is so called Doppler frequency.

The time-varying multipath channel is mainly characterized by the time spread introduced to the transmitted signal. Hence, the signal at the receiver input is the sum of the attenuated and delayed replica signals of the original transmitted signal. Herein, let's consider the communication model depicted in Figure (1.3), represented in the equivalent baseband (low-pass) representation. This simplified model is proposed by Forney [18].

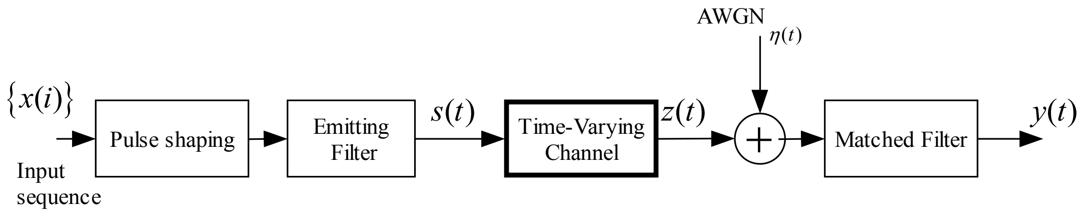

Figure 1.3: Communication Model [18]

The equivalent received baseband signal is [15]

$$z(t) = \sum_{l=1}^{k} \alpha_l(t) e^{-j2\pi f_c \tau_l(t)} s(t - \tau_l(t)) \tag{1. 1}$$

Where $k$ is the number of paths, $\tau_l(t)$ is the time-varying delay presented in each $l$ path, $\alpha_l(t)$ is the time-varying attenuation presented in the m path as well, $f_c$ carrier frequency, and $s(t)$ is the transmitted signal.

The time-varying attenuation $\alpha_l(t)$ changes significantly, only if the channel experiences large dynamic changes. Hence, for quite large time interval, the attenuation can be considered roughly as a constant. Furthermore, another approximation can be done by considering the time varying delay $\tau_l(t)$ as constant for quit large number of symbols, since the time variation during the symbol period is so small and can be ignored, APPENDIX A, then we can approximate $s(t-\tau_l(t)) \approx s(t-\tau_l)$.Whereas, the term $2\pi f_c \tau_l(t)$ is affected significantly even though $\tau_m$ changes very slightly, owing to the high carrier frequency $f_c$. Thus the signal $z(t)$ becomes

$$z(t) = \sum_{l=1}^{k} \alpha_l e^{-j2\pi f_c \tau_l(t)} s(t-\tau_l) \tag{1. 2}$$

When the mobile body has a constant velocity, the propagation delay can be approximated by a linear function of time ($f_c \tau_l(t) \square \lambda_{0l} + f_l t$), APPENDIX A. The received signal is then

$$z(t) = \sum_{l=1}^{k} \beta_l e^{-j2\pi f_l t} s(t-\tau_l) \tag{1. 3}$$

where $\beta_l = \alpha_l e^{-j2\pi\lambda_{0l}}$

Then, the sampled version of the received signal can be shown to be

$$z(n) = \sum_{m=1}^{k} \beta_m e^{-j2\pi \tilde{f}_l n} s(n-p_l) \tag{1. 4}$$

where $\tilde{f}_l = f_l T_s, \tau_l \square p_l T_s, p_l$ is an integer, and $T_s$ is the sampling period (which is equal to the symbol period). Actually, the Eq. (1.4) justifies the choice of exponentially basis functions. After that, we can write the noiseless received signal as

$$z(n) = \sum_{i=0}^{q} a_i(n) s(n-i) \tag{1. 5}$$

where $q = \max\{p_l, l = 1,\dots,k\}$ is the model order. If $i \notin \{p_l\}$, the respective parameters $b_i$ are zero. Eq. (1.5) represents the channel impulse response as an MA model with periodically time-varying parameters.

To generate an overall model for the system communication in Figure (1.3), let $\{x(i)\}$ be the symbol sequence. The baseband emitted signal will be

$$s(t) = \sum_{i=-\infty}^{+\infty} x(i) g_1(t - iT_s) \tag{1. 6}$$

where $T_s$ is the symbol period, $g_1$ is the overall impulse response of the cascaded connection between the shaping pulse and the emission filter. Now, substituting Eq. (1.6) into Eq. (1.1) yield

$$z(t) = \sum_{l=1}^{k} \alpha_l(t) e^{-j2\pi f_c \tau_l(t)} \sum_{i=-\infty}^{+\infty} x(i) g_1(t - iT_s - \tau_l(t)) \tag{1.7}$$

The low-pass received signal then becomes,

$$y(t) = z(t) * g_2(t) + \eta(t) * g_2(t) \triangleq w(t) + v(t) \tag{1.8}$$

where $g_1$ is the impulse response of the matched filter, and '$*$'stands for convolution) . Assuming that both of the impulse response $g_1, g_2$ have finite duration, and after some derivation, for more information see reference [15], we can easily reached to the following a complete MA model of the given communication system in Figure (1.3) (discrete-time input/output relationship of the channel)

$$y(n) = \sum_{i=0}^{q} a_i(n) s(n-i) + v(n), \tag{1. 9}$$

with

$$a_i(n) = \sum_{l=1}^{k} a_{il} e^{-j2\pi \tilde{f}_l n} \tag{1. 10}$$

where $q$ is the memory length of the channel, $a_{il}$ are complex constant coefficients, $\tilde{f}_l$ are the corresponding Doppler frequencies which given by [4].

$$\tilde{f}_l = \frac{v}{c} T_s \tag{1. 11}$$

where $v$ is the mobile body speed, c is the speed of the light.

We conclude that the mobile radio channel can be modeled by a linear periodically time varying filter [4], as long as the time delays considered as linearly time varying. The nonstationary time-varying channel's response is extended over exponential basis functions as in Eq. (1.9) and (1.10). Hence, to recognize the nonstationary case channel impulse response, then we have encountered the problem of estimating of the channel constant coefficients $\{a_{il}\}$, and their respective channel Doppler shift $\tilde{f}_l$.

## 1.4 Adaptive Filters and Adaption Algorithms

Adaptive filters play an important role in the adaptive channel equalization systems. The main structure of the adaptive filter is shown in the block diagram below:

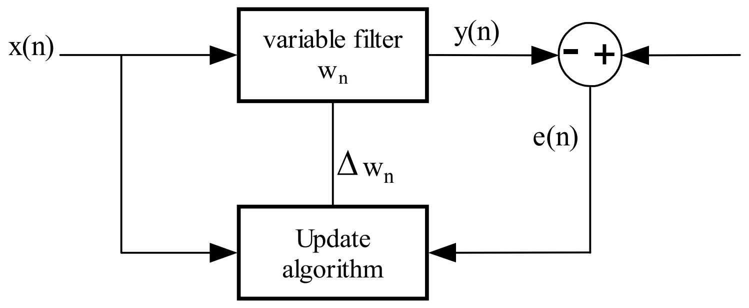

Figure 1.4: Adaptive Filter Concept [32]

In general, the input to the adaptive filters is the desired signal *d* (n) plus the noise *v*(n), where sometimes the input signal considered as the desired signal, multiplied by distorted function(i.e. time variant system) is defined as follows :

$$x(n) = d(n) + v(n) \tag{1. 12}$$

The variable filter has a Finite Impulse Response (FIR) structure. In such structures, the impulse response is equal to the filter coefficients, which are updated recursively by the specified updating algorithm. The coefficient weight for a filter of order (*p*) is defined as follows:

$$w_n = [w_n(0), \quad w_n(1), \quad \ldots w_n(p)]^T \tag{1. 13}$$

The cost function or the error function e(n), defined as the difference between the desired and estimated output signal y(n) is given as

$$e(n) = d(n) - y(n) \tag{1. 14}$$

The estimated output is found by the variable FIR filter by the convolution between the input signal and filter impulse response; this operation could be expressed in vector notation as:

$$y(n) = w_n * x(n) \tag{1. 15}$$

where $x(n) = [x(n), \quad x(n-1), \quad \ldots, \quad x(n-p)]^T$ is an input signal vector. Afterwards, the variable filter updates the filter coefficient at every instant or iteration

$$w_{n+1} = w_n + \Delta w_n \tag{1. 15}$$

Where $\Delta w_n$ is a correction updating factor for the filter coefficients and depends on how the adaptive algorithm is utilized from the input signal and the generated error.

## 1.5 Motivation and Contributions of the Thesis

Research work in this thesis makes several contributions to the area of adaptive equalization and communication fields. First, an efficient new BF estimator, termed as recursive inverse basis function (RIBF), is proposed. Next, a frequency-adaptive version of RIBF is developed by means of a simple gradient search strategy. The new BF estimator outperforms the EWBF estimator by providing considerable complexity savings. Moreover, RIBF is superior to the Gradient-BF and EWBF estimators since it shows further reduction in the mean square parameter estimation error without using any correction code. These advantages results in significant advantages when applied in wireless communications to reduce BER, SNR and channel bandwidth requirements.

## 1.6 Organization of the Thesis

This thesis has six chapters, two core chapters: Chapters Four and Five, where the contributions and verifications are located. And it has four supporting chapters, Chapters One, Two, Three and Six, which presents principles, conduct surveys and draw conclusions.

Chapter One sets out the problem statement and discusses the thesis contribution and motivations and gives a thesis outline.

Chapter Two discusses the development and categorization of adaptive equalization techniques. Then it presents some of the important adaptive equalization structures, namely, Linear Transversal Equalizer (LTE), Decision Feedback Equalization (DFE), Maximum Likelihood Sequence Estimation (MLSE) Equalizer and Fractionally Spaced Equalizer (FSE).

Chapter Three shows the principles and development of FIR adaptive algorithms and explore their main characteristics. Then, it contains a brief survey for the most important adaptive algorithms, such as Stochastic Gradient Family (LMS, NLMS, TDLMS and Newton-LMS), Least Squares (RLS) and the newly proposed RI algorithm. This chapter furnishes the reader with the necessary background theory and information on the adaptive algorithms theory.

Chapter Four introduces the BF algorithms which are a special form of adaptive filters that fit the periodically time-varying systems. It then introduces the Exponentially Weighted Basis Function EWBF estimators in two forms; Running Basis (RB) and Fixed Basis (FB) algorithms. Furthermore, it shows the BF-gradient estimators in both running and fixed basis forms. Finally, it presents the new

Recursive Inverse Basis Function (RIBF) estimator, as well as, introducing its frequency-adaptive version.

Chapter Five presents an example where the proposed algorithms are successfully applied in multipath fading channels. Accordingly, it shows the simulation results of the mentioned BF algorithms using MATLAB software package. Additionally, it presents the performance and the superiority of the proposed RIBF-estimator over the other competitive algorithms. It investigates the performance capability of the proposed frequency-adaptive version of the EWBF estimator.

Eventually, Chapter Six draws conclusions and suggests improvements for the current work.

# Chapter 2

# ADAPTIVE EQUALIZATION TECHNIQUES

The techniques of adaptive equalization have been developed during the last two decades to cope with the market demand for efficient, high speed and reliable communication devices. Hence, many equalization techniques have been proposed for combating ISI on band-limited time-fading channels. These techniques can be subdivided into two main types, linear and nonlinear equalization. The categorization referred to whether the output of equalizer is affected by the decision maker output or not as shown in Figure (2.1) [12]. If the decision maker output (digital stream $d(t)$) feeds back to control the equalizer, then the equalization is non linear, otherwise it is linear. Different filter structures have been used to implement each type that is available in the literature [13]. Among many we can mention Fractionally Spaced Equalizer, Blind Equalization, Linear Phase Equalizer, T-shaped Equalizer, Decision Feedback Equalization, Dual Mode Equalizer and Linear Transversal Equalizer. Associated with each structure there is a class of adaptive algorithms that may be used to adaptively adjust the parameters of the equalizer. Figure (2.2) [21] illustrates a general classification for equalization techniques according to the filter types, structures and algorithms employed.

## 2.1 Linear Equalization Techniques

The linear equalizer may be implemented as a finite-duration impulse response (FIR) filter. The most widely used equalizer structure is the Linear Transversal Equalizer (LTE).

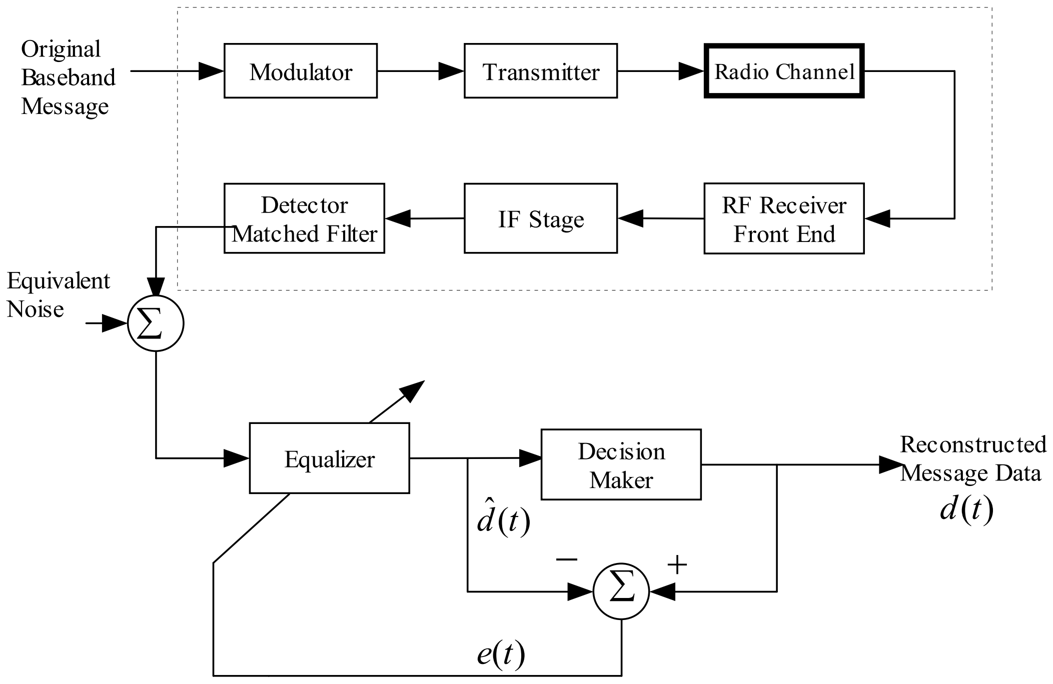

Figure 2.1: Block Diagram of a Simplified Communication System Using an Adaptive Equalizer at the Receiver [12]

The LTE is made up of tapped delay lines which store samples from the input sequence as one per symbol period $T_s$, and give a sum of the weighted input sequence with the filter weights, in an attempt to synthesize the converse of the channel effects. Some texts refer to such a structure as symbol-spaced linear equalizer [13], since the input and output rates are equal.

As mentioned earlier, in order to achieve appropriate initialization for the filter coefficients, a short training sequence maybe transmitted within the start-up period.

The most common criteria used in the optimization of the filter coefficients is the Mean Square Error (MSE) between the desired output of the equalizer and its actual output. The most common algorithm proposed to minimize MSE value is the Least Mean Square (LMS) Algorithm (Widrow and Hoff, 1960), which is simple but suffers from slow converges rate.

The low converge rate of the LMS algorithm is due to the presence of only one parameter, namely the step size parameter that controls the adaptation of the process.

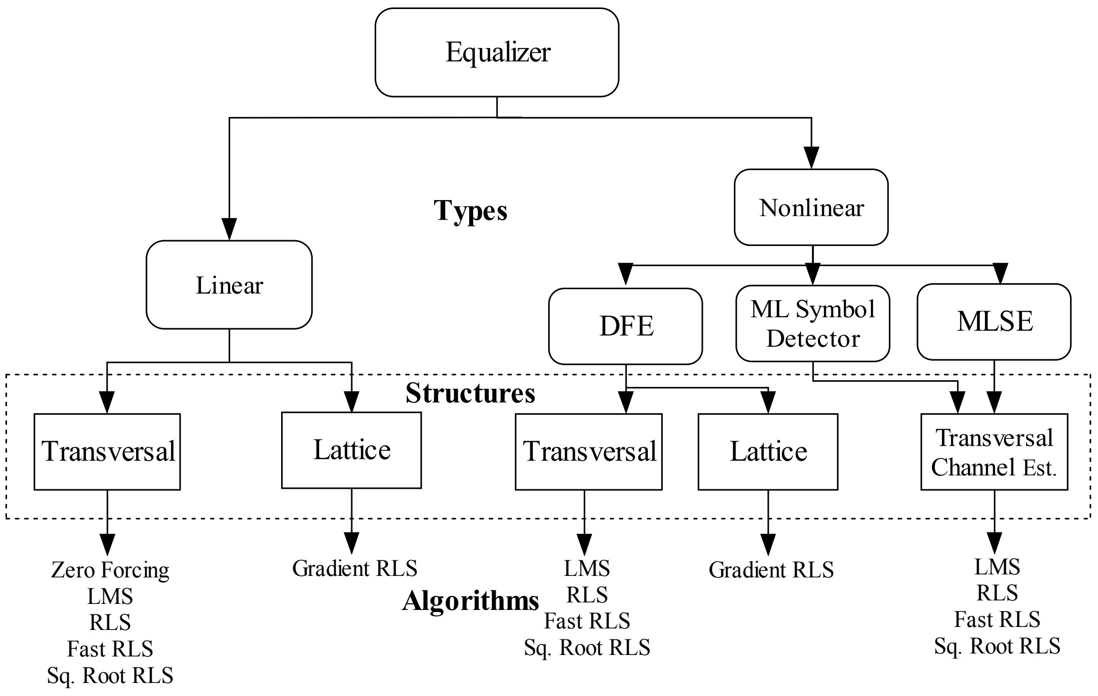

Figure 2.2: Classification of Equalizers [21]

Another common criterion is the Least Squares method (LS). LS-method is used in a deterministic frame to minimize the sum of the exponentially weighted squares error recursively in the case of the LTE equalizer. This adaptive algorithm is referred to as Recursive Least Squares (RLS) algorithm. RLS algorithm is converges faster than LMS algorithm at the expense of high computational complexity. After that, many other RLS- based algorithms appropriate for LTE equalizer have been designed to solve the RLS computational complexity. Among many we can count square-root RLS algorithm [23], fast RLS, and the algorithm which is appropriate to implement the RLS criterion as a lattice structure, which is called as RLS Lattice (RLSL) algorithm [29].

Both, the transversal and lattice linear equalizers are all-zero filters. Hence, obtaining an Infinite-duration Impulse response (IIR) structure can be easily done by adding a filter section containing poles. However, the addition of poles may affect the equalizer in term of its stability; as a result, adaptive IIR equalizers are rarely used.

## 2.2 Nonlinear Equalization Techniques

Nonlinear equalizers are applied in communication channels where the channel distortion is too severe and it's insufficient for the linear equalizer to handle this distortion. Linear equalizers do not perform well on the channels which have spectral nulls in their frequency response characteristics. As it attempts to fix the channel distortion problem, the linear equalizer places a large gain in the neighborhood of the spectral null. As a result, significant enhancement for the current noise at those frequencies will occur.

In the last three decades, very efficient nonlinear equalizers have been proposed, in an attempt to compensate for linear equalizer drawbacks. One of them is the Decision Feedback Equalization (DFE). The second one is a sequence detection algorithm, which is based on the criterion of maximum likelihood sequence estimation (MLSE). DFE equalizer is implemented efficiently by the Viterbi Algorithm (VA). As an equalizer, MLSE was firstly proposed by (Forney, 1972). We briefly investigate the key characteristics of these methods.

### 2.2.1 Decision Feedback Equalization

The action of the DFE equalizer is to feed back a weighted sum of past decisions to cancel the ISI that they cause in the present signaling interval [20]. In other words, the ISI distortion carried on the present input that was introduced by previous pulse is subtracted. The DFE can be implemented either by transversal or lattice form. The DFE transversal structure is composed of two filters, a feedforward filter and a feedback filter. The later is fed by the decision of the detector nonlinear outputs, which is the reason to classify the DFE as a nonlinear equalizer.

The advantage of the DFE comes from the presence of the feedback filter, which is an additional component that works on the noiseless quantized level to remove ISI. Whereas, the nonlinearity of the DFE may leads to an instability problem, especially when an incorrect decision propagates to affect the feedback filter weights.

### 2.2.2 Maximum Likelihood Sequence Estimation (MLSE) Equalizer

MLSE is the same as LMS algorithm, is optimum in a sense that it minimizes the probability of symbol error. But LMS works only when the channel does not introduce any amplitude distortion to the signal, which is the familiar problem that requires attention in the mobile communication applications. MLSE deals with such a problem by choosing the data sequence with maximum probability as the output. MLSE acts by testing all the data sequence that comes from sampling the analog signal of the matched filter output. The MLSE requires knowledge of the channel characteristics; in addition to that, it requires the knowledge of the statistical distribution of the noise smearing the signal. Hence, in case of ISI that covers many symbols, the statistical computational complexity becomes impractical, therefore

MLSE may be considered as a benchmark for comparison purposes with other algorithm's performance [21].

The MLSE Equalizer was first proposed as an equalizer by Forney in 1972, and it has recently been implemented successfully in mobile radio channels.

## 2.3 Fractionally Spaced Equalizer (FSE)

The optimum receiver for a digital communication channel smeared by Additive White Gaussian Noise (AWGN) that is well-known is composed of a matched filter which is sampled periodically at the symbol rate. Furthermore, if the smeared received signal is corrupted by ISI, then the sample needs to be treated by either linear or non linear equalizer. Hence, the matched filter prior to the equalizer, in the presence of channel distortion, must be matched with the channel distorted signal. Whereas, in practice the channel impulse response is mostly unknown, accordingly the optimum matched filter must be adaptively estimated. Another suboptimum solution, where is the matched filter is matched with the transmitted signal, may result in undesired degradation in the receiver performance [21]. In fact, The Fractionally Spaced Equalizer (FSE) incorporates the equalization and matched filtering functions into a single filter structure.

The FSE works by receiving $K$ input signal samples involved in producing one output signal sample and then updating the filter weights. Therefore, the input sample rate can be expressed as ($K/T$), and the output samples rate ($1/T$) (also equals to the rate of updating the weights). Consequently, it is called fractional. The FSE has many advantages [24]. One of them is that it has the ability to not be affected by ISI (aliasing) problem and the sample rate is less than the symbol rate.

FSE’s is currently used almost in all commercially high speed modems over voice frequency channels.

# Chapter 3

# ADAPTIVE FILTERING ALGORITHMS

In this chapter, we show the development of adaptive algorithms and explore their main characteristics which show their superiority to others. Firstly, we investigate the classical adaptive linear filtering algorithm, which encompasses the steepest descent method and we show the main obstacles encountered when trying to put such filters in practical situations. We also briefly show the development of adaptive filter algorithms which have widespread applications. The family of the gradient stochastic algorithms are the following; Least Mean Square (LMS) algorithm (Widrow and Hoff, 1960), Normalized version of LMS algorithm (NLMS), Frequency-Domain least-mean-square (FDLMS), Transform Domain LMS (TDLMS), and Newton LMS. The most important characteristic of the LMS algorithm is its simplicity. We also consider the Recursive least-squares (RLS) algorithm which may be viewed as a special case of the Kalman filter [1]. The main characteristic of the RLS algorithm is faster convergence compared with LMS algorithms since it utilizes the data from the starting point, however, this significant performance leads to more complexity. Finally, we investigate the newly proposed Recursive inverse (RI) algorithm which has considerable reductions in complexity and better performance than the RLS algorithm [3].

## 3.1 Steepest Descent Algorithm

At this point it would be wholesome to introduce a well-known optimization technique, known as the steepest descent method. This method is recursive, in the sense that starting from the initial value for the tap-weight vector, it improves with increased number of iterations [1]. And at the end of this process the weights will converges to the Wiener solution, which is mainly what we are searching for.

In Figure 3.1, we consider a transversal filter, with input vector samples drawn from a wide stationary stochastic source at time *n*

$$\mathbf{u}(\mathrm{n}) = [u(n), \quad u(n-1), \quad \ldots \quad ,u(n-M+1)]^T \tag{3.1}$$

the vector of tap weights, $\mathbf{w}(\mathrm{n})$, is given by

$$\mathbf{w}(\mathrm{n}) = [w_0(n), \quad w_0(n), \quad \ldots w_{M-1}(n)]^T \tag{3.2}$$

The difference between the estimated output $\hat{d}(n)$ and the desired response *d(n)*, generates an estimation error *e(n)*, given by

$$e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{w}^H(n)\mathbf{u}(\mathrm{n}) \tag{3.3}$$

Actually, if both vector of inputs **u**(n) and the desired response *d*(n) are jointly stationary, then the mean square error (MSE) = $E\left[e(n)e^*(n)\right]$ or cost function $\boldsymbol{J}(n)$ at time *n* is a quadratic function of the weight taps given as

$$\boldsymbol{J}(n) = \sigma_d^2 + \mathbf{w}^{\mathrm{H}}(\mathrm{n})\,\mathbf{p}\text{-}\mathbf{p}^{\mathrm{H}}\mathbf{w}(\mathrm{n}) + \mathbf{w}^{\mathrm{H}}(\mathrm{n})\,\mathbf{R}\,\mathbf{w}(\mathrm{n}) \tag{3.4}$$

where $\sigma_d^2$ = desired response *d(n)* variance,

**p** = cross correlation vector between input vector **u**(n) and the desired response *d(n)*

**R** = auto correlation matrix of the input vector **u**(n)

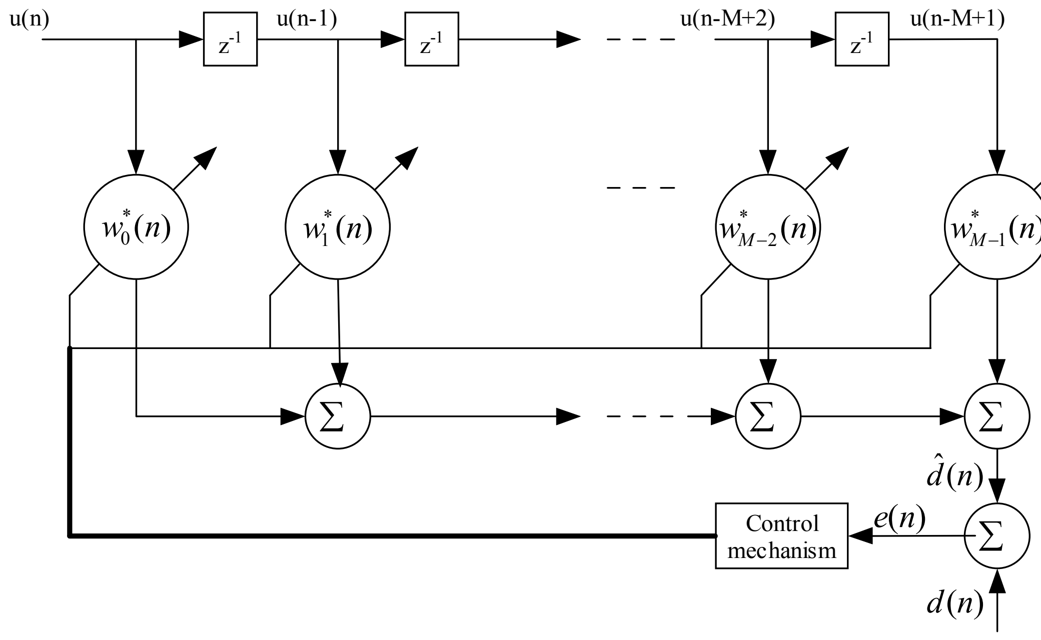

Figure 3.1: Adaptive Transversal Filter Structure [1]

It is clear from Eq.(3.4) that the cost function ($\boldsymbol{J}(n)$)changes with time since the tap weights **w**(n) are changing with time as well. This leads to changing estimated error (*e*(n)), and this change signifies the fact that the error process is a time-varying process (non stationary).

We may visualize the dependence of the mean square error ($\boldsymbol{J}(n)$)on the tap weight vector (**w**(n) )elements, as a bowl-shaped surface with unique minimum [1]. Adaptive filters search for that minimum point to achieve the optimum weight vector $w_0$ which is the optimum solution of the Wiener-Hopf equation, given as

$$\mathbf{R}\mathbf{w}_0 = \mathbf{p} \tag{3.5}$$

and the minimum point of the cost function

$$\boldsymbol{J}_{\min} = \sigma_d^2 - \mathbf{p}^{\mathrm{H}}\mathbf{w}_0 \tag{3.6}$$

Solving the Wiener-Hopf equation (3.5) is straightforward, even though, it has computational difficulties especially when the input data rate is high and the number of tap weights increase. However, the steepest descent method is an alternative approach to solving the Wiener-Hopf equation iteratively. It is one of the oldest optimization methods for searching a multidimensional performance surface. Let $\nabla(\boldsymbol{J}(n))$ denote the gradient vector at time *n*, and ($\mathbf{w}$(n) = $\mathbf{a}$(n) + j$\mathbf{b}$(n)) the complex weight vector. Then, the new update value of the weight vector at *n + 1* is computed recursively as

$$\mathbf{w}(\mathrm{n}+1) = \mathbf{w}(\mathrm{n}) + \frac{1}{2}\mu\left[-\nabla(\boldsymbol{J}(n))\right] \tag{3.7}$$

and the gradient $\nabla(J(n))$ of the cost function is given by:

$$\nabla(\boldsymbol{J}(n)) = \begin{bmatrix} \dfrac{\partial J(n)}{\partial a_0(n)} + j\dfrac{\partial J(n)}{\partial b_0(n)} \\ \dfrac{\partial J(n)}{\partial a_1(n)} + j\dfrac{\partial J(n)}{\partial b_1(n)} \\ \vdots \\ \dfrac{\partial J(n)}{\partial a_{M-1}(n)} + j\dfrac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix} = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(\mathrm{n}) \tag{3.8}$$

In the steepest-descent algorithm we assumed the autocorrelation **R** matrix and cross correlation **p** are known, so we can find the gradient of the cost function and then we could find the update values of the weight-taps using the following steepest descent algorithm recursive equation

$$\mathbf{w}(\mathrm{n}+1)=\mathbf{w}(\mathrm{n})+\mu\left[\mathbf{p}-\mathbf{R}\mathbf{w}(\mathrm{n})\right] \tag{3.9}$$

where the $\mu$ parameter controls the size of correction applied to the weight vector every iteration, so we call it the step-size parameter or weighting constant.

## 3.2 Stochastic Gradient algorithms

### 3.2.1 Least Mean Square (LMS) algorithm

Using the exact measurements of the gradient vector $\nabla(\boldsymbol{J}(n))$ in Eq. (3.6) at each iteration, with suitable choice of the step size, the steepest descent algorithm would converge to $\mathbf{w}_0$, which is the optimum solution of the Wiener-Hopf equation. Nevertheless, exact measurements of the gradient vector are not possible, since it needs prior knowledge of the correlation matrix **R** and crosscorrelation vector **p** .

In reality, we are dealing with a stream of data sequences that are coming from the source, so filters such as the steepest descent may fail in this case. Algorithms like least mean square (LMS) come out to adapt such a sequence of data. A great feature of the LMS algorithm is that it does not require previous knowledge of the correlation parameters (**R** matrix and **p** vector) nor does it need matrix inversion.

If we substitute the estimated values of autocorrelation matrix **R** and cross correlation vector **p** to the gradient vector values, which they depend on the instantaneous values of the tap input vector **u**(n) and the desired response *d*(n) as follows

$$\mathbf{R}(n)=\mathbf{u}(\mathrm{n})\,\mathbf{u}^{\mathrm{H}}(n) \tag{3.10}$$

and

$$\hat{\mathbf{p}}(n) = \mathbf{u}(\mathrm{n})d^*(n) \tag{3.11}$$

then, instantaneous estimation of the gradient vector will be

$$\hat{\nabla}(\boldsymbol{J}(n)) = -2\mathbf{u}(\mathrm{n})d^*(n) + 2\mathbf{u}(\mathrm{n})\mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{w}}(n) \tag{3.12}$$

Substituting the estimated value of the gradient cost vector $\hat{\nabla}(\boldsymbol{J}(n))$ in Eq. (3.12) to the steepest descent recursive formula in Eq. (3.6), we obtain a new recursive algorithm for updating the weight vector:

$$\hat{\mathbf{w}}(\mathrm{n}+1) = \hat{\mathbf{w}}(\mathrm{n}) + \mu\, \mathbf{u}(\mathrm{n})\left[d^*(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{w}}(\mathrm{n})\right] \tag{3.13}$$

(The hat sign over the symbols as used here is to distinguish the tap weights vector from the values obtained from steepest descent algorithm). Now, if we look carefully at the estimate tap weights in the preceding equation, we may rewrite it in terms of the filter output and estimation error as follows:

$$\hat{\mathbf{w}}(\mathrm{n}+1) = \hat{\mathbf{w}}(\mathrm{n}) + \mu\, \mathbf{u}(\mathrm{n})\, e^*(n) \tag{3.14}$$

where $e$(n) is the estimated error

$$e(n) = d(n) - y(n) \tag{3.15}$$

and y(n) is the filter output

$$y(n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(\mathrm{n}) \tag{3.16}$$

Eqs. (3.14) to (3.16) describe the complex version of Least Mean Square (LMS) algorithm, which is considered as a member of stochastic gradient algorithms. The correction term ( $\mu\, \mathbf{u}(\mathrm{n})\, e^*(n)$ in Eq.(3.14) added to the estimate value of tap weights $\hat{\mathbf{w}}(n)$ to predict the new estimate of tap weight function can take any direction and changes randomly. The tap weight vector search starts from initial value $\hat{\mathbf{w}}(0) = 0$.

LMS algorithm can be represented in a single flow diagram, as a close loop feedback model as shown in Figure (3.8). The simplicity of the LMS algorithm is obvious, which needs *2M+1* complex multiplications and *2M* complex additions, in each iteration for *M* tap weights in the adaptive transversal filter.

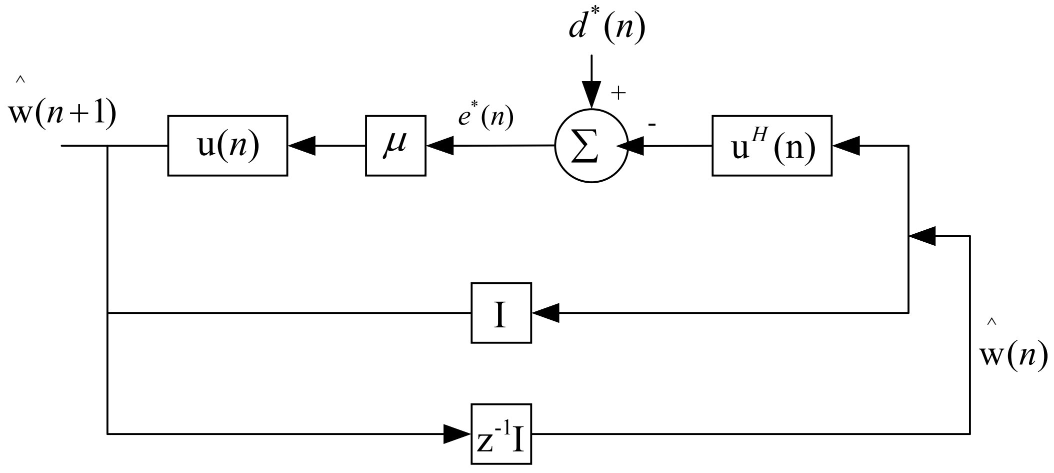

Figure 3.2: Single-Flow Graph Representation of LMS-Algorithm [1]

Finally, it would be proper to mention that the LMS algorithm remains stable, when the step size $\mu$ remains inside the following band [1]:

$$0 < \mu < \frac{1}{3\mathrm{tr}[\mathrm{R}]} \tag{3.17}$$

where: $\mathrm{tr}[\mathrm{R}] = \sum_{i=1}^{N-1} r_{ii}$ .

For stationary input signals and sufficiently small $\mu$ [6], the speed of convergence of the LMS algorithm is dependent on the eigenvalue spread, which is the ratio of the maximum to the minimum eigenvalues ($\lambda_{\max}(\mathbf{R})/\lambda_{\min}(\mathbf{R})$) of the input autocorrelation matrix **R.** It's often referred to the mentioned ratio which measures the ill-conditioning of the matrix as the condition number.

### 3.2.2 Normalized Least Mean Square (NLMS) Algorithm

The correction factor $\mu\, \mathbf{u}(\mathrm{n})\, e^*(n)$ in LMS algorithm is directly proportional to input data sample stream **u**(n).Hence, any large change in the input sequence affects the stability and the convergence of the tap weights vector to its optimum value. This large input value **u**(n) causes a gradient noise amplification problem [1].

Normalized LMS algorithm (NLMS)[1] comes out to solve the problem of variation in the signal amplitude at the filter input by considering a time variant step size. In brief, we may implement NLMS algorithm as a natural modification of the conventional LMS algorithm, but instead of considering constant step size $\mu$, we normalize the correction factor $\mu\, \mathbf{u}(\mathrm{n})\, e^*(n)$ to the square Euclidean norm of the input tap vector **u**(n). However, to examine this we may derive the NLMS algorithm as a solution to the minimal optimization problem (Goodwin and Sin, 1984) using the method of Lagrange multipliers. The problem states that for input vector **u**(n) and desired response *d*(n) find the weight vector **w** at *n+1* in order to minimize the square Euclidean norm of the change of $\hat{\mathbf{w}}(\mathrm{n}+1)$ to its original tap weights $\hat{\mathbf{w}}(\mathrm{n})$ as follows

$$\delta\hat{\mathbf{w}}(\mathrm{n}+1) = \hat{\mathbf{w}}(\mathrm{n}+1) - \hat{\mathbf{w}}(\mathrm{n}) \tag{3.18}$$

Subject to the constraint

$$\hat{\mathbf{w}}^H(\mathrm{n}+1)\mathbf{u}(n) = d(n) \tag{3.19}$$

---

[1] This algorithm has been suggested independently by Nagumo and Nado (1967) and Albert and Gardner (1967) under different names. Its name “Normalized LMS algorithm” was invented by Bitmead and Anderson.

As mentioned earlier, to solve this optimization problem we use the Lagrange multipliers method. The square norm of the change in the weight vector can be expressed as follows

$$\begin{aligned}\|\delta\hat{\mathbf{w}}(\mathrm{n}+1)\|^2 &= \delta\hat{\mathbf{w}}^H(\mathrm{n}+1)\delta\hat{\mathbf{w}}(\mathrm{n}+1)\\ &= \left[\hat{\mathbf{w}}(\mathrm{n}+1)-\hat{\mathbf{w}}(\mathrm{n})\right]^H\left[\hat{\mathbf{w}}(\mathrm{n}+1)-\hat{\mathbf{w}}(\mathrm{n})\right]\\ &= \sum_{k=0}^{M-1}\left|\hat{\mathbf{w}}_k(\mathrm{n}+1)-\hat{\mathbf{w}}_k(\mathrm{n})\right|^2 \end{aligned} \tag{3.20}$$

Then, we define the weight vector $\hat{\mathbf{w}}(\mathrm{n})$, tap input $\mathbf{u}(n\text{-}k)$ and the desired response $d$(n) for $k$=0, 1, . . . ,$M$-1 in their respective real and imaginary parts:

$$\begin{aligned}&\hat{\mathbf{w}}(\mathrm{n})=\mathbf{a}_k(n)+j\mathbf{b}_k(n)\\ &\mathbf{u}(n-k)=\mathbf{u}_1(n-k)+j\mathbf{u}_2(n-k)\\ &d(n)=d_1(n)+jd_2(n)\end{aligned} \tag{3.21}$$

Now, using the complex representations to modify the constraints of the optimization problem in Eq. (3.18) and (3.19), we have

$$\left|\delta\hat{\mathbf{w}}(\mathrm{n}+1)\right|^2 = \sum_{k=0}^{M-1}(\left[a_k(n+1)-a_k(n)\right]^2+\left[b_k(n+1)-b_k(n)\right]^2) \tag{3.22}$$

By the same way substituting in Eq. (3.20), and writing it as an equivalent pair of the real and imaginary part of the real representation, (for more details see Haykin (1991)). Then we may also write the real valued-cost function ***J***(n) for the constraints of the given optimization problem in one single relation as follows

$$\begin{aligned} \boldsymbol{J}(n) &= \left|\delta\hat{\mathbf{w}}(\mathrm{n}+1)\right|^2 \\ &= \sum_{k=0}^{M-1}\left(\left[a_k(n+1)-a_k(n)\right]^2+\left[b_k(n+1)-b_k(n)\right]^2\right) \\ &= \lambda_1\left[d_1(n)-\sum_{k=0}^{M-1}\left(a_k(n+1)u_1(n-k)+b_k(n+1)u_2(n-k)\right)\right] \\ &+ \lambda_2\left[d_2(n)-\sum_{k=0}^{M-1}\left(a_k(n+1)u_2(n-k)+b_k(n+1)u_1(n-k)\right)\right] \end{aligned} \tag{3.23}$$

where both $\lambda_1$ and $\lambda_2$ are Lagrange multipliers. Now, by differentiating the cost function ***J***(n) with respect to the tap weight parameters $a_k(n+1)$ & $b_k(n+1)$ ,and then setting the results equal to zero, we can find their optimum values,

$$\begin{aligned} \frac{\partial J(n)}{\partial a_k(n+1)} &= 0 = 2\left[a_k(n+1)-a_k(n)\right]-\lambda_1 u_1(n-k)-\lambda_2 u_2(n-k) \\ \frac{\partial J(n)}{\partial b_k(n+1)} &= 0 = 2\left[b_k(n+1)-b_k(n)\right]-\lambda_1 u_1(n-k)+\lambda_2 u_2(n-k) \end{aligned} \tag{3.24}$$

Using the complex representation form of $\hat{\mathbf{w}}(\mathrm{n})$ and $\mathbf{u}(\mathrm{n\text{-}k})$ in Eq.(3.21) to combine Eq.(3.24) into a single complex one, we have

$$2\left[\hat{w}_k(n+1)-\hat{w}_k(n)\right]=\lambda^* u(n-k), \qquad k=0,1,\ldots,M-1 \tag{3.25}$$

Where $\lambda=\lambda_1+j\lambda_2$ is the complex Lagrange multiplier.

Now to solve for the unknown $\lambda^*$, we multiply both sides of Eq.(3.25) by $u^*(n-k)$, then sum up over all the possible integer values utilized from the vector representation of the sums, we have

$$\lambda^* = \frac{2}{\left\|\mathbf{u}(n)\right\|^2}\left[d^*(n)-\hat{\mathbf{w}}^H(n)\mathbf{u}^*(n)\right] \tag{3.26}$$

We may write the Lagrange multiplier in terms of error *e*(n):

$$\lambda^* = \frac{2}{\|\mathbf{u}(n)\|^2} e^*(n) \tag{3.27}$$

Finally, substituting the Lagrange multiplier $\lambda^*$ value into Eq. (3.25), we obtained

$$\begin{aligned}\delta \hat{w}(n+1) &= \hat{w}_k(n+1) - \hat{w}_k(n) \\ &= \frac{1}{\|u(n)\|^2} u(n) e^*(n)\end{aligned} \tag{3.28}$$

Equivalently we may rewrite Eq. (3.28) in vector form as follows:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{1}{\|\mathbf{u}(n)\|^2} \mathbf{u}(n) e^*(n) \tag{3.29}$$

In order to control the step change introduced over the tap weight vector, we may introduce a real positive scaling factor $\tilde{\mu}$, then we may express Eq. (3.29) in iterative form which is the Normalized Least Mean Square (NLMS) update equation for M-by-1 tap weight vector:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2} \mathbf{u}(n) e^*(n) \tag{3.30}$$

On this recursion formula we may show the following observations:

1. The term "Normalized" comes when we normalize the vector product term $\mathbf{u}(n)e^*(n)$ with respect to the square Euclidean norm of the input vector **u**(n).
2. The adaption step size $\tilde{\mu}$ for NLMS algorithm is dimensionless, while the adaption step size $\mu$ for LMS algorithm has an inverse power dimension.
3. We may show that the NLMS algorithm is a new version of LMS algorithm, but with time variant adaption step size parameter $\mu(n)$ is as follow:

$$\mu(n) = \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2} \tag{3.31}$$

4. The NLMS algorithm converges in the mean square sense when the constant step size $\mu$ satisfies the following requirement (Hsia, 1983):

$$0 < \tilde{\mu} < 2 \tag{3.32}$$

5. Despite NLMS algorithm comes to solve numerical problem, it may add a new numerical problem. Specifically, when the input vector becomes small, the division of the gradient part over square value of the input norm becomes too large. To solve such a problem, we add a new constant to the input norm square. The recursion formula of the NLMS algorithm in Eq. (3.30) becomes:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{a + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n) \tag{3.33}$$

where $a > 0$, if $a = 0$ Eq. (3.33) comes back again to the original NLMS recursion equation Eq.(3.30).

### 3.2.3 The Transform Domain LMS (TDLMS) Algorithm

In this adaptive algorithm approach, one of the drawbacks of LMS algorithm which is the slow convergence property has been improved on. This section material was drawn from [5]. In general, as mentioned early for a stationary input sequence and proper selection of small adapting constant $\mu$, the rate of convergence is dependent on the eigenvalues spread (ratio of the maximum to the minimum eigenvalues) of the input autocorrelation sequence **R**. if this ratio is large, then a slow convergence rate can be expected. An approach to accelerate the convergence rate is to somehow transform the input signal **u**(n) into another signal with the corresponding autocorrelation matrix having a smaller eigenvalue spread [7],[8]. Varying the adaptive filtering process to some orthogonal transform domain guarantees the increase of the convergence rate.

A block diagram of the transform domain LMS adaptive filter is shown in Fig. (3.9), the input sequence vector **u** transformed into another vector sequence $\mathbf{u}_T$

$$\begin{aligned}\mathbf{u}_T(n) &= \mathrm{T}\mathbf{u}(\mathrm{n}) \\ &= \left[u_{T,0}(n),\quad u_{T,1}(n),\quad \dots\quad ,u_{T,M-1}(n)\right]^T\end{aligned} \tag{3.34}$$

where $\mathrm{T}$ is a unitary matrix of rank $M$ (i.e. $\mathrm{TT}^* = I_M$ ).

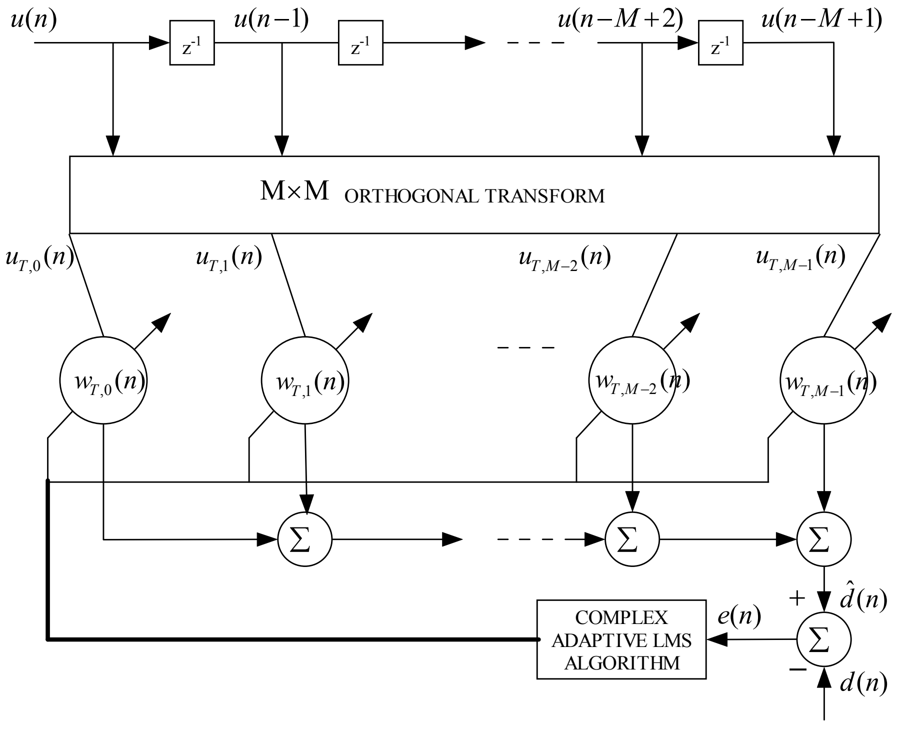

Figure 3.3: Block Diagram of the Transform Domain LMS-Adaptive Filter [5]

Then , we multiplied the transferred input vector $\mathbf{u}_T(n)$ by the transferred domain weight vector $\mathrm{w}_T(n)$ to form the adaptive output sequence, given below

$$\hat{\mathbf{d}}(n) = \mathbf{u}_T^T(n)\mathbf{w}_T(n) \tag{3.35}$$

where $\mathbf{w}_T(n) = \left[ w_{T,0}(n), \quad w_{T,1}(n), \quad \ldots \quad , w_{T,M-1}(n) \right]^T$ is the transform domain of the weight vector .

Now, the weight update vector equation becomes:

$$\mathbf{w}_T(n+1) = \mathbf{w}_T(n) + 2\mu\hat{\mathbf{D}}^{-1}e(n)\mathbf{u}_T(n) \tag{3.36}$$

where,

$$e(n) = d(n) - \mathbf{w}_T^T(n)\mathbf{u}_T(n) \tag{3.37}$$

$\hat{\mathbf{D}}$ is the estimate of the diagonal matrix **D**, which its (*i,i*) elements correspond to the power estimate $\hat{\sigma}^2_{\mathrm{u}_T}(n)$ of the input sequence $\mathbf{u}_T(n)$ as follows:

$$\mathbf{D} = \begin{bmatrix} E\left[u_{T,0}^2(n)\right] & 0 & \cdots & 0 \\ 0 & E\left[u_{T,2}^2(n)\right] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E\left[u_{T,M-1}^2(n)\right] \end{bmatrix} \tag{3.38}$$

The estimated diagonal matrix $\hat{\mathbf{D}}$ can be achieved recursively by

$$\hat{\mathbf{D}}(n) = \beta\hat{\mathbf{D}}(n-1) + (1-\beta)\mathrm{diag}(\mathbf{u}_T^2(n)) \tag{3.39}$$

where $\beta$ is positive constant close to 1, $\hat{\mathbf{D}}(n) = \mathrm{diag}\left[\hat{\sigma}^2_{u_{T,0}}(n), \hat{\sigma}^2_{u_{T,1}}(n), \ldots, \hat{\sigma}^2_{u_{T,M-1}}(n)\right]$.

The inverse of matrix **D** is valid as long as the input sequence autocorrelation matrix **R** is positive definite. By applying matrix **D** we get the normalized tap input vector (not to confuse with NLMS algorithm), as it is obvious from the weights equation (3.36), as follows:

$$\mathbf{u}_{T,normalized}(n) = \mathbf{D}^{-1/2}\mathbf{u}_T(n) \tag{3.40}$$

This normalization can convert $\mathbf{R}_T$ to a normalized matrix $\mathbf{R}_{T,normalized}$ where its eigenvalue spread will be much smaller than that of **R**. Thereby, by choosing a proper orthogonal transform, the convergence behavior of LMS algorithm in the transform domain is expected to have an improved convergence performance over the corresponding time domain algorithm.

Moreover, if the orthogonal transform $\mathbf{T}$ is chosen to render $\mathbf{R}_T$ completely diagonal, then the eigenvalue spread becomes unity. Adaptive filters implemented in such a situation, have been shown to have the best convergence performance. The corresponding $\mathbf{T}$ transform which reduced the time domain M-vector adaption problem to the M-scalar in the transform domain is the famous as Karhunen-Loeve Transform (KLT). In most real application KLT is difficult to apply, since it is dependent on **R** itself.

The most popular orthogonal transform are Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT), as shown in the APPENDEX C. Choosing DFT as a transform domain in this case, implies to a recognized name which is called Frequency Domain Least Mean Square Algorithm (FDLMS), (Shankar and Peterson, 1981). Both recursively whitened the input sequence **u**(n) , then the transformed input signal $\mathbf{u}_T(n)$ with reduced dynamic spectrum range is introduced for adaptation process.

Finally, TDLMS algorithm which implemented by Equations (3.36) and (3.37), has a wide range application especially in signals which have large spectral dynamic range such as speech signals. Furthermore, it may also have a good demand when the input signals are obtained by sampling continuous signal after it passes a low-pass (anti-aliasing) filter. If the sampling frequency is twice as greater than the filter's cut off frequency, and if the filter has small stopband amplitude, then the dynamic range spectrum of the input sequence will be wide, despite whether the dynamic range spectrum of the continuous input signal is narrow. So applying the time delay filters in such application may lead to lack in the convergence rate, whereas the orthogonal filters show superior over the time delay filters in terms of convergence performance.

### 3.2.4 Newton-LMS Algorithm

This section material was drawn from [5]. Newton's method as the steepest method is an iterative method used in the literature as an approach for searching for the critical points in solving optimization problem. Using the steepest descent algorithm Eq. (3.9) and Wiener Hopf Eq.(3.5), the estimated weight vector can be written as

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \mu\left[\mathrm{R}\mathbf{w}(n) - \mathrm{R}\mathbf{w}_0\right] \\ &= \mathbf{w}(n) - \mu\mathbf{R}\left[\mathbf{w}(n) - \mathbf{w}_0\right]\end{aligned} \tag{3.41}$$

From LMS algorithms which uses the steepest descent method, we can conclude that all of the LMS algorithms suffers from low convergence speed when the autocorrelation matrix has wide of eigenvalue spread. This problem might be clear in Eq.(3.41) due to the existence of autocorrelation matrix of the input signal **R** which makes the steepest descent algorithm suffers always from slow convergence problem.

Newton's method solves the problem of slow convergence. Instead of $\mu$ in Eq. (3.7) we substituted matrix of adaptive constant $\mu\mathbf{R}^{-1}$, to give the Newton's update equation as shown below

$$\mathbf{w}(\mathrm{n}+1)=\mathbf{w}(\mathrm{n})+\frac{1}{2}\mu\mathbf{R}^{-1}\left[-\nabla(\boldsymbol{J}(n))\right] \tag{3.42}$$

This replacement in the Newton's equation guarantees an improvement in the speed of the convergence. This is because the inverse of the autocorrelation matrix $\mathbf{R}^{-1}$, affects the gradient vector direction in terms of rotation to search for the minimum point on the searching surface.

To find out the Newton-LMS algorithm, considering the same process as what has been done in LMS algorithm, that is, replacing the stochastic gradient vector by its instantaneous estimate vector $\hat{\nabla}(\boldsymbol{J}(n))$ and processed. Then Eq. (3.32) becomes

$$\mathbf{w}(\mathrm{n}+1)=\mathbf{w}(\mathrm{n})+\mu\mathbf{R}^{-1}e(n)\mathbf{u}(\mathrm{n}) \tag{3.43}$$

Since equation (3.43) assumes that the matrix $\mathbf{R}^{-1}$ needs to be a known matrix, it may be called an ideal Newton-LMS algorithm. In practice, which assumes the validation of $\mathbf{R}^{-1}$ is not possible, makes the use Newton-LMS is insufficient. Therefore, using the iterative method to estimate $\mathbf{R}^{-1}$, as we will show later, makes Newton-LMS more applicable.

It can be shown that TDLMS and Newton-LMS algorithms have two different representations for the same algorithm by choosing Karhunen Loeve transform (KLT) at the filter input signal. Then, for a proper transform, TDLMS efficiently represented Newton-LMS algorithm. More details can be found in [4].

## 3.3 Recursive Least Squares (RLS) Algorithm

This section material was drawn from [5]. Filter design problem to find out the optimum parameters which are correspond to give the desired output response for a given input sequence have two formulations; one of these formulation is the statistical design approach, which has been discussed so far. And the other one is the deterministic structure in the filter-solving problem. Wiener filter and the LMS with its derivatives are considered to belong to the statistical framework since the minimizing of the least mean square (statistical quantity) parameter's criteria is considered in their designs. In this section, solving the optimization problem using the deterministic point of view is being presented using the method of least squares. Same as the standard LMS algorithm has wide derivatives with wide range of applications according to their figure of merit, the algorithms of least squares method also has. Using the least squares method and matrix inversion Lemma relation, the Recursive Least Squares (RLS) algorithm is presented to find out the desired response recursively. This algorithm has shown faster convergence rate than LMS with increased the computational complexity. This is because it exploits the data input sequence from the time of initialization until the time the weighted vector is updated.

As the LMS algorithm derivation criteria minimize the error in mean square sense, RLS minimizes the error in least square sense. In least squares method the adaptive filter taps calculated by minimizing the cost function giving by

$$\xi(n)=\sum_{i=1}^{n}\beta(n,i)\left|e(i)\right|^{2} \tag{3.44}$$

Where this cost function $\xi(n)$ assumes that all the past input sequence and the desired response output should be available to minimize the error at the present output. The error *e(i)* is the difference between the estimated desired output and the desired output given by

$$\begin{aligned} e(i) &= d(i) - y(i) \\ &= d(i) - \mathbf{w}^H(n)\mathbf{u}(i) \end{aligned} \tag{3.45}$$

where *d(i)* is the desire response at time *i*, $\mathbf{u}(i)$ is the tap input vector at time *i*

$$\mathbf{u}(i) = \left[u(i), \quad u(i-1), \quad \ldots \quad ,u(i-M+1)\right]^T \tag{3.46}$$

and **w**(n) is the tap weight vector at time *n* giving by

$$\mathbf{w}(\mathrm{n}) = \left[w_0(n), \quad w_0(n), \quad \ldots w_{M-1}(n)\right]^T \tag{3.47}$$

Note that the tap weights are constant in the time observation interval $1 \le i \le n$ for which the error cost function $\xi(n)$ is defined.

$\beta(n,i)$ is the weighting factor which has the property of $0 < \beta(n,i) \le 1$ . The special weighting factor that has been used is the exponential weighting factor or forgetting factor

$$\beta(n,i) = \lambda^{n-i} \tag{3.48}$$

where $\lambda$ is a positive constant close to, but less than one. Forgetting factor clearly works by putting more weight for the present and near present samples and forget the distant past samples. Putting $\lambda = 1$ gives the ordinary method of least squares (LS) which has an infinite memory by considering all the samples from the starting point of the simulation. Whereas, in case of $\lambda < 1$, trying to emphasize the weight of the most present samples, which becomes a most criterion in case of nonstationary input.

Roughly speaking, the memory approximately equal to $(1-\lambda)^{-1}$. Then the exponentially weighted least squares cost function is minimized as follows

$$\xi(n) = \sum_{i=1}^{n} \lambda^{n-i} \left|e(i)\right|^{2} \tag{3.49}$$

At minimum cost error function $\xi(n)$, the optimum tap weight vector $\hat{\mathrm{w}}(n)$ in matrix form is giving by the following normal equation:

$$\mathbf{\Phi}(\mathrm{n})\hat{\mathbf{w}}(n) = \mathbf{z}(n) \tag{3.50}$$

The M-by-M autocorrelation matrix $\Phi(\mathrm{n})$ for the filter input **u**(n) is defined by

$$\mathbf{\Phi}(\mathrm{n}) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}(i)\mathbf{u}^{H}(i) \tag{3.51}$$

Also the M-by-1 cross-correlation vector **z**(n) between the filter input **u**(n) and the desired response *d(n)* is given by

$$\mathbf{z}(\mathrm{n}) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}(i) d^{*}(i) \tag{3.52}$$

Expanding the correlation matrix $\mathbf{\Phi}(\mathrm{n})$ as follows:

$$\mathbf{\Phi}(\mathrm{n}) = \lambda \left[ \sum_{i=1}^{n-1} \lambda^{n-1-i} \mathbf{u}(i)\mathbf{u}^{H}(i) \right] + \mathbf{u}(n)\mathbf{u}^{H}(n) \tag{3.53}$$

Comparing the first term in the left hand side of Eq. (3.53) with Eq.(3.51), it can be deduced that this term represents the correlation matrix $\mathbf{\Phi}(\mathrm{n})$ of the input samples at time *n-1*. Hence, the following recursion equation for correlation matrix is introduced:

$$\mathbf{\Phi}(\mathrm{n}) = \lambda \mathbf{\Phi}(n-1) + \mathbf{u}(n)\mathbf{u}^{H}(n) \tag{3.54}$$

The second term $\mathbf{u}(i)\mathbf{u}^H(i)$ in Eq. (3.54) represents the correction applied to the old value of the correlation matrix $\mathbf{\Phi}(n-1)$ during the updating process.

This also can be done for the cross-correlation vector in Eq. (3.52) to get the following recursive update equation for cross-correlation between the tap input vector and desired response

$$\mathbf{z}(\mathrm{n}) = \lambda \mathbf{z}(n-1) + \mathbf{u}(n)d^*(n) \tag{3.55}$$

To attain the least square estimate of the tap weight vector $\hat{\mathrm{w}}(n)$ in Eq. (3. 49), it's clear that the inverse of the correlation matrix $\mathbf{\Phi}(\mathrm{n})^{-1}$ should be available. In practice, we try to avoid the inversion at all, because it is considered time consuming where trying to find the least square estimate $\hat{\mathbf{w}}(n)$ recursively. Achieving both of these requirements are possible by using a linear algebra, i.e. by using matrix inversion lemma, see APPENDIX B.

Applying matrix inversion lemma for Eq. (3.54) and taking into account that the initial condition has been chosen to ensure that the correlation matrix $\mathbf{\Phi}(\mathrm{n})$ is positive definite (nonsingular matrix). Considering the parameters in matrix inversion lemma as follows

$$\begin{aligned} &\mathbf{A} = \lambda \mathbf{\Phi}(n-1) \\ &\mathbf{B} = \mathbf{u}(n) \\ &\mathbf{C} = \mathbf{u}^H(n) \\ &\mathbf{D}^{-1} = -1, \end{aligned} \tag{3.56}$$

Substituting these parameters definitions in the matrix inversion lemma (APPENDEX B), gives the following recursive equation for the inverse autocorrelation matrix

$$\mathbf{\Phi}^{-1}(\mathrm{n}) = \lambda^{-1}\mathbf{\Phi}(n-1) - \frac{\lambda^{-2}\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^{H}(n)\mathbf{\Phi}^{-1}(n-1)}{1+\lambda^{-1}\mathbf{u}^{H}(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n)} \tag{3.57}$$

For convenient representation assume that

$$\mathbf{P}(\mathrm{n})=\mathbf{\Phi}^{-1}(\mathrm{n}) \tag{3.58}$$

and

$$\mathbf{k}(\mathrm{n})=\frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^{H}(n)\mathbf{P}(n-1)\mathbf{u}(n)} \tag{3.59}$$

Then, the inverse correlation matrix in terms of these definitions can be written as follows

$$\mathbf{P}(\mathrm{n}) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1) \tag{3.60}$$

where **P**(n) is an M-by-M inverse of the autocorrelation input matrix $\mathbf{\Phi}(\mathrm{n})$, and **k**(n) is M-by-1 vector which is called gain factor. Rearranging the gain factor in a different way as follows

$$\mathbf{k}(\mathrm{n}) = \left[\lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(\mathrm{n})\mathbf{u}^{H}(n)\mathbf{P}(n-1)\right]\mathbf{u}(n) \tag{3.61}$$

Considering Eq. (3.61) and comparing it with Eq. (3.60), the term inside the bracket at left hand side of Eq. (3.61) equal to **P** (n). Hence Eq. (3.61) can be written in terms of the input and autocorrelation inverse as given

$$\begin{aligned}\mathbf{k}(\mathrm{n}) &= \mathbf{P}(n)\mathbf{u}(n) \\ &= \mathbf{\Phi}^{-1}(\mathrm{n})\mathbf{u}(n)\end{aligned} \tag{3.62}$$

From the previous formula, it can be seen that the gain factor definition, which is equal to the transformation version from the tap input vector done by the correlation inverse.

To find the least square update equation for the tap weight vector, substitute the value of recursive formula of the cross correlation matrix **z**(n) in Eq. (3.50)

$$\begin{aligned}\hat{\mathrm{w}}(n) &= \mathbf{\Phi}^{-1}(n)\mathbf{z}(n)=\mathbf{P}(n)\mathbf{z}(n)\\ &= \mathbf{P}(n)\lambda\mathbf{z}(n-1)+\mathbf{P}(n)\mathbf{u}(n)d^{*}(n)\\ &= \mathbf{\Phi}^{-1}(n-1)\mathbf{z}(n-1)-\mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{z}(n-1)+\mathbf{P}(n)\mathbf{u}(n)d^{*}(n)\\ &= \hat{\mathbf{w}}(n-1)-\mathbf{k}(n)\mathbf{u}^{H}(n)\hat{\mathbf{w}}(n-1)+\mathbf{P}(n)\mathbf{u}(n)d^{*}(n)\end{aligned} \tag{3.63}$$

Note that in Eq. (3.63), **P** (n) **u** (n) equals to the gain factor **k**(n). After substituting and arranging equation (3.63), the least squares estimate of the tap weight vector becomes

$$\hat{\mathbf{w}}(n)=\hat{\mathbf{w}}(n-1)+\mathbf{k}(n)\varepsilon^{*}(n) \tag{3.64}$$

where, $\varepsilon(n)$ is the prior estimation error given by

$$\begin{aligned}\varepsilon(n) &= d(n)-\mathbf{u}^{T}(n)\hat{\mathbf{w}}^{*}(n-1)\\ &= d(n)-\hat{\mathbf{w}}^{H}(n-1)\mathbf{u}(n)\end{aligned} \tag{3.65}$$

Where, the scalar product $\hat{\mathrm{w}}^{H}(n-1)\mathrm{u}(n)$ is the desired response of the filter at the previous tap weight vector found at *(n-1).* From the tap weight vector adjustment equation and from the prior estimation error equation, the RLS algorithm can easily be represented in block diagram shown in Figure (3.10).

RLS algorithm comprise of Eq. (3.59) which represent the gain factor, Eq. (3.65) that gives the prior estimate of error, Eq. (3.64) that also forms the estimation of the tap weight vector and Eq. (3.60) which is the inverse estimate of the correlation function. RLS algorithm is summarized in table (3.1). The initialization of RLS algorithm should be chosen in a way to avoid the singularity of the correlation matrix. The tap weight vector is also preferably initialized with the null vector.

Finally, the RLS algorithm represents a solution for minimizing the cost function of the optimization problem (Sayed and Kailath, 1994) given by:

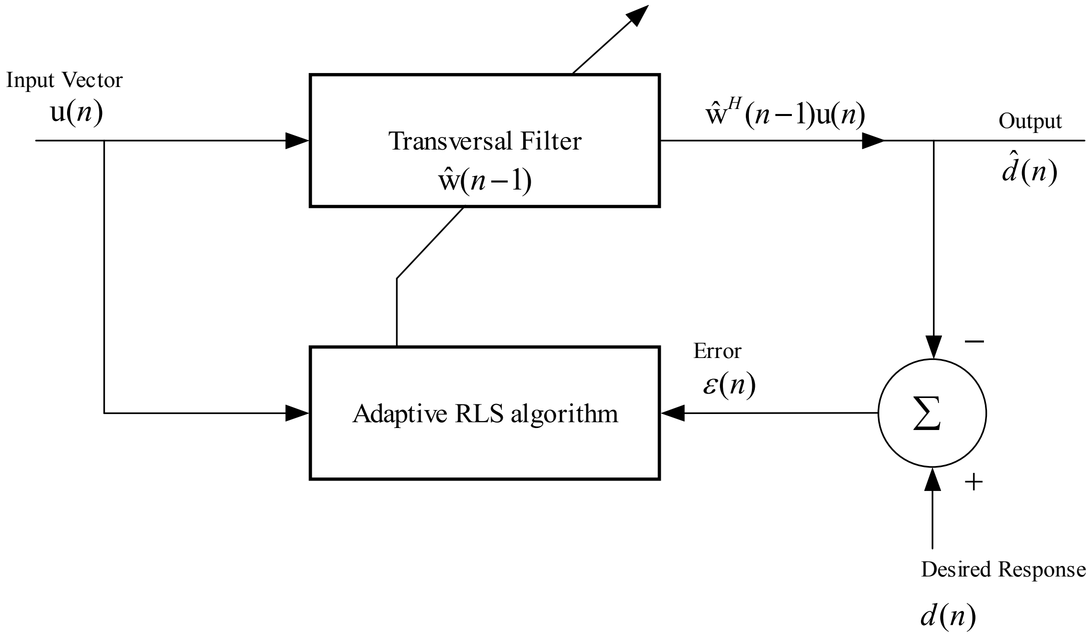

Figure 3.4: Block Diagram Representation of RLS-Adaptive Filter [1]

$$\min_{\mathbf{w}(n)} \left[ \delta\lambda^n \left\|\mathbf{w}(n)\right\|^2 + \sum_{i=1}^{n} \lambda^{n-i} \left|e(i)\right|^2 \right] \tag{3.66}$$

where e(i) is error in the following equation

$$e(i) = d(i) - \mathbf{w}^H(n)\mathbf{u}(i) \tag{3.67}$$

It is important to mention that the RLS algorithm suffers from computational complexity; it grows in proportion with the square value of the filter length, which makes things to get worse in implementation and speed point of view.

Table 3.1: Summary of RLS-Algorithm

| | |
|---|---|
| Initialization: | $\mathbf{P}(0) = \delta^{-1}\mathrm{I}$ <br> $\hat{\mathbf{w}}(0)=0$ <br> $\delta$ : is small positive constant. <br> I : M-by-M identity matrix |
| Iteration: | $\mathbf{k}(\mathrm{n})=\dfrac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)}$ <br> $\varepsilon(n) = d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)$ <br> $\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\varepsilon^*(n)$ <br> $\mathbf{P}(\mathrm{n}) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)$ |

This serious drawback of the RLS algorithm has been challenging for the last two decades. To solve the problem, researchers have created many algorithms, which are proportional with the filter length, and are commonly referred to as the *fast* RLS algorithms. These invented algorithms utilize of the prediction and filtering concepts in an elegant way to present the efficient of computational complexity.RLS lattice (RLSL) algorithm [29], which is one of the most numerically robust implementation of fast RLS algorithms. Also, Fast Transversal RLS (FTRLS) which is known as fast transversal filter (FTF) [22], is an alternative implementation which has solved the problem of complexity efficiently.

## 3.4 Recursive Inverse (RI) algorithm

This section material was drawn from [1].Recursive inverse (RI) is one of the newly proposed algorithms, which has shown superior performance over others in some charactersistics, such as computational complexity and speed of convergence. In addition, it has solved the problem of correlation inverse by excluding it from the updating equation.

To derive the update equation of RI algorithm, we start from the Wiener-Hopf equation. The equation is written again for convenience

$$\mathbf{w}(n) = \mathbf{R}^{-1}(n)\mathbf{p}(\mathrm{n}) \tag{3.68}$$

where **R**(n),**p**(n) are the autocorrelation matrix of the input vector **u**(n) and the cross-correlation vector between the input $\mathbf{u}(n)$ and the desired response $d(n)$, respectively. Note that, as the filter coefficients vector is updated, the solution of Eq. (3.68) is required in each iteration. Furthermore, there is a need for the autocorrelation matrix to be nonsingular at each iteration [1].

Considering the estimation of the correlation parameters iteratively, the equations (3.54) and (3.55) that estimate the correlations matrix and vector, are written again for real evaluation cases, with new symbol notations

$$\mathbf{R}(\mathrm{n}) = \lambda\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}^T(n) \tag{3.69}$$

$$\mathbf{p}(\mathrm{n}) = \lambda\mathbf{p}(n-1) + \mathbf{u}(n)d(n) \tag{3.70}$$

where $\lambda$ is the forgetting factor, usually close and smaller than one.

Solving the Wiener-Hopf equation (3.68) iteratively at each time step *n*, gives the filter update equation (3.9), which converges to the optimum Wiener solution and it is given by (the equation written again for convenience)

$$\mathbf{w}_{k+1}(n) = \left[\mathrm{I} - \mu\mathbf{R}(n)\right]\mathbf{w}_k(n) + \mu\mathbf{p}(\mathrm{n}) \tag{3.71}$$

for $k = 0,1,2,\ldots$.

And, if $\mu$ satisfies the convergence condition in [1]

$$\mu < \frac{2}{\Lambda_{\max}(\mathbf{R}(n))} \tag{3.72}$$

where $\Lambda_{\max}(\mathbf{R}(n))$ is the maximum eigenvalue of correlation matrix **R**(n). Referring to correlation update equation (3.69) and taking the expectation of **R**(n):

$$\bar{\mathbf{R}}(\mathrm{n}) = \lambda\bar{\mathbf{R}}(n-1) + \mathbf{R}_{uu} \tag{3.73}$$

where $\mathbf{R}_{uu}$=$E\left\{\mathbf{u}(n)\mathbf{u}^T(n)\right\}$, $\bar{\mathbf{R}}$(n)=E$\left\{\mathbf{R}(n)\right\}$. Solving the difference equation (3.73) yields

$$\bar{\mathbf{R}}(\mathrm{n}) = \frac{1-\lambda^n}{1-\lambda}\mathbf{R}_{uu} \tag{3.74}$$

Considering the maximum value, when $n \to \infty$

$$\bar{\mathbf{R}}(\infty) = \frac{1}{1-\lambda}\mathbf{R}_{uu} \tag{3.75}$$

Eq. (3.74) stated that the eigenvalues in the estimated correlation matrix $\bar{\mathrm{R}}$(n) increase exponentially as $(1-\lambda^n)$, with a maximum limit $(1-\lambda)^{-1}$ times that of the original correlation matrix. Since the step-size $\mu$ is restricted to satisfy Eq. (3.72) as a maximum limit might be possible to approach in order to converge for optimum Wiener solution, then we get

$$\mu < \frac{2(1-\lambda)}{\Lambda_{\max}(\mathbf{R}_{uu})} = \mu_{\max} \tag{3.76}$$

This equation shows that the step-size $\mu$ takes values which are much smaller than that required in Eq. (3.72). Now, it would be more convenient to replace the step size with a variable one so that

$$\mu(n) < \frac{2}{\Lambda_{\max}(\mathbf{R}(\mathrm{n}))} = \left(\frac{1}{(1-\lambda^n)}\right)\left(\frac{2(1-\lambda)}{\Lambda_{\max}(\mathrm{R}_{uu})}\right) \tag{3.77}$$

It is clear in the equation (3.77) that the second part at right hand side, is equal to the maximum allowable limit for the step size if $\mathrm{R}_{uu}$ is used. Hence, Eq. (3.77) becomes

$$\mu(n) < \frac{\mu_{\max}}{1-\lambda^n} = \frac{\mu_0}{1-\lambda^n} \tag{3.78}$$

where $\mu_0 < \mu_{\max}$.

It's clear in the steepest descent update equation (3.74), that there is a high iterative complexity cost. Consequently, replacing the step-size constant with the variable one makes only one iteration at each time step to be adequate [3]. Hence, the final update equation of the RI algorithm becomes

$$\mathbf{w}(n) = \left[\mathrm{I} - \mu(n)\mathbf{R}(n)\right]\mathbf{w}(n-1) + \mu(n)\mathbf{p}(\mathrm{n}) \tag{3.79}$$

The RI algorithm has many superior advantages over the others; it does not need update in the inverse of the matrix as the other does, which some time leads to face numerical stability problems. Furthermore, it has considerable reduction in the computational complexity than the others such as standard RLS algorithm.

Table 3.2: Summary of RI-Algorithm

| | |
|---|---|
| Initialization: | $\mathbf{P}(0) = 0$ |
| | $\mathbf{R}(0) = 0$ |
| | $\hat{\mathbf{w}}(0)=0$ |
| | $\mu_0$ : is small positive constant. |
| | $\lambda$ Smaller and less than one. |
| Iteration: | |
| | $\mathbf{R}(\mathrm{n}) = \lambda\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}^T(n)$ |
| | $\mathbf{p}(\mathrm{n}) = \lambda\mathbf{p}(n-1) + \mathbf{u}(n)d(n)$ |
| | $\mu(n) = \dfrac{\mu_0}{1-\lambda^n}$ |
| | $\mathbf{w}(n) = \left[\mathbf{I} - \mu(n)\mathbf{R}(n)\right]\mathbf{w}(n-1) + \mu(n)\mathbf{p}(\mathrm{n})$ |

## 3.5 Applications of Adaptive Filtering

The capability of adaptive filters to track variation in unknown environment, and to follow the changing in the statistics of the input signal, gives the adaptive filters a wide range of application in control and signal processing. Indeed, adaptive filters have been successfully applied to communications devices, sensor, radar and biomedical engineering. Despite this variety in the applications, all filters have the same structure, the input vector and the desired response in finding out the estimate error, which is responsible for adjustment in the taps of the filter. However, these varieties of applications are owing to the manner of how we extract and use the desired response. Subsequently, we may distinguish four basic classes of adaptive filtering application [13], as shown in the following table.

Table 3.3: Four Basic Classes of adaptive Filtering and their Applications

| **Class of adaptive filtering** | **Application** | **Description** |
|---|---|---|
| **Identification** | System identification, | Given an unknown dynamical system, the purpose system identification is to design an adaptive filter that provides an approximate to the system. |
| | Layered earth modeling | In exploration seismology, a layered modeled of the earth is developed to unravel of the complexities of the earth's surface. |
| **Inverse Modeling** | Equalization | Given a channel of unknown impulse response, the purpose of an adaptive equalizer is to operate on the channel output such that the cascade connection of the channel followed by the equalizer provides an ideal transmission medium. |
| **Prediction** | Predictive coding, | The adaptive prediction is used to develop a model of a signal of interest (e.g., a speech signal); rather than encode the signal directly, in predictive coding the prediction error is encoded for transmission or storage. Typically, the prediction error has a smaller variance than the original signal, Hence, the |

|  |  |  |
|---|---|---|
|  |  | bases for improved coding. |
|  | Spectrum analysis | In this application, predictive modeling is used to estimate the power spectrum of a signal of interest. |
| **Interference Cancelation** | Noise Cancellation, | The purpose of the adaptive noise canceller is to subtract noise from a received signal in adaptively controlled manner so as to improve the signal-to-noise ratio. Echo cancellation, experience on telephone circuits, is a special form of noise cancellation. Noise cancellation is also used electrocardiography. |
|  | Beamforming | A beamforming is spatial filter consist of an array of antenna elements with adjustable weights (coefficient). The twin purposes of an adaptive beamformer are to adaptively control the weights so as to cancel interfering signal. |

# Chapter 4

# ALGORITHMS FOR IDENTIFICATION OF PERIODICALLY VARYING SYSTEMS

Adaptive filters have many classes, owing to verities of applications to which they are applied. Adaptive equalization is one of the most famous applications that have a special situation against others, especially in considering nonstationary environment, i.e. radio channels which have time varying property. Still it is possible to apply adaptive filters in nonstationary environments under certain assumptions, which they will be clear later.

In this Chapter we set up BF algorithms which are a special form of adaptive filters that fit the periodically time-varying systems. After that, we introduce the Exponentially Weighted Basis Function EWBF estimators in two forms; Running Basis (RB) and Fixed Basis (FB) algorithms. Furthermore, we show the BF-gradient estimators in both running and fixed basis forms. Finally, we propose the new Recursive Inverse Basis Function (RIBF) estimator, as well as, introduce its frequency-adaptive version.

Finally, most of this chapter materials and terminologies have been referred to the paper that was proposed by Niedzwiecki and Klaput in 2003 [2]. Our contribution is the proposing of the new RIBF and its Frequency-adaptive version extended from the existing algorithm that was proposed by Ahmad, Kukrer and Hocanin [3].

## 4.1 Operation of Adaptive Filters in Nonstationary Environment

In this situation it might be important to ask these questions: how can we apply adaptive filters in general for time-variant systems instead of time-invariant one? And how will its performance be in this situation?

Anyway, when we apply an adaptive filter to a nonstationary environment, the tap weight coefficients assume time varying form which means the values change at different iteration. Hence, the adaptive filter will have an extra duty to continuously keep seeking for the optimum value (minimum error) on the time varying error surface.

Let $\mathbf{w}_0(n)$ be the optimum time-varying vector coefficients of the transversal filter that works in the nonsationary situation, where $n$ is the number of iteration. Also, let $\hat{\mathbf{w}}(n)$ be the estimated vector coefficients. Hence, the adaptive algorithms tries on their own approach get the optimum situation; this makes the estimated tap weights $\hat{\mathbf{w}}(n)$ to get as close to the time varying optimum tap weight vector $\mathbf{w}_0(n)$ as possible, as follows

$$\begin{aligned}\boldsymbol{\varepsilon}(n) &= \hat{\mathbf{w}}(n) - \mathbf{w}_0(n) \\ &= \left(\hat{\mathbf{w}}(n) - E\left[\hat{\mathbf{w}}(n)\right]\right) + \left(E\left[\hat{\mathbf{w}}(n)\right] - \mathbf{w}_0(n)\right) \\ &= \boldsymbol{\varepsilon}_1(n) + \boldsymbol{\varepsilon}_2(n)\end{aligned} \qquad (4.1)$$

where the expectation indicates the ensemble average.

It is easy from equation (4.1) to identify two kinds of error; the first one stands for the difference between the estimated tap coefficients and ensemble average of the tap coefficients $E\left[\hat{\mathbf{w}}(n)\right]$. This is due to the adaptive algorithm error minimizing criteria (i.e. for LMS case, it’s due to the errors in the estimate used for gradient vector [1]).

This difference is called the weight vector noise $\boldsymbol{\varepsilon}_1(n)$. The second one is weight vector lag difference $\boldsymbol{\varepsilon}_2(n)$; which stands for any difference between the ensemble average $E\left[\hat{\mathbf{w}}(n)\right]$ and the optimum coefficients $\mathbf{w}_0(n)$ (it's the target value) which is due to lag in the adaptive process. In a stationary process, it is assumed that the ensemble average $E\left[\hat{\mathbf{w}}(n)\right]$ is equal to the optimum value $\mathbf{w}_0(n)$. Therefore, the weight vector lag difference disappears, against the nonstationary processes which usually consider the weight vector lag difference.

## 4.2 Exponential Basis Function Estimators

In this thesis, we considered the problem of tracking and identification of the periodically varying system which is summed up by equations (1.9) and (1.10) given by, the following equation written again for convenience in vector form

$$y(t) = \boldsymbol{\varphi}^T(t)\boldsymbol{\theta}(t) + \upsilon(t) \tag{4.2}$$

where $t = 1, 2, \dots$ denotes the normalized discrete time, $\upsilon(t)$ is an additive white noise, *y(t)* is the system output. $\boldsymbol{\varphi}(t)$ is the regression vector consisting of the past input samples, given by

$$\boldsymbol{\varphi}(t) = \left[u(t), u(t-1), \dots, u(t-n+1)\right]^T \tag{4.3}$$

and $\boldsymbol{\theta}(t)$ is the vector of periodically time varying system coefficients given by

$$\boldsymbol{\theta}(t) = \left[\theta_1(t), \theta_2(t), \dots, \theta_n(t)\right]^T \tag{4.5}$$

where $n$ is the number of system coefficients . The $i$th system coefficient is given as a linear combination of basis exponentials as follows

$$\theta_i(t) = \sum_{l=1}^{k} a_{il} e^{jw_l t} \tag{4.6}$$

*k stands* for number of different signal paths (fading path  or number of basis function).

As mentioned earlier, adaptive filters could be employed efficiently for periodically time varying system under certain conditions. We will consider that the frequencies $w_1, w_2, \ldots, w_k$ are given or estimated a prior, the parameters or system weighting coefficients $a_{il}$ which are unknown constants and independent of time, or slowly (possibly) time-varying quantities, so the main time-dependent terms are the basis function, and lastly, the input sequence is considered as a wide-sense stationary WSS ergodic process with known positive definite covariance matrix given by

$$\mathbf{\Phi}_0 = E\left[\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^T(t)\right] > 0 \tag{4.7}$$

As also mentioned earlier that, under certain conditions one of the applications that admit such formulation is adaptive equalization of rapidly fading communication channels [2]. As we know that in modern wireless communication systems, the main distortion which the transmitted signal experiences during transmission is caused mostly by the multipath effect; the transmitted signal reaches most probably a moving receiver (with constant speed) with different time delays, and when this effect is dominant through different reflectors, all of this implies an unknown periodic channel impulse response which can be modeled by Eq. (4.2). Then y(t) in this particular case stands for the sampled baseband signal received by the mobile radio system, *u(t)* for time sequence $t = 1, 2, \ldots$ is the transmitted symbols sequence(which includes the traning sequence) *k* is the number of signal paths, and finally, $w_1, w_2, \ldots, w_k$ are the corresponding Doppler shifts.

Actually, in case the system parameters change very rapidly, the conventional estimators such as Weighted Least Squares WLS and LMS become unsatisfactory. Hence, it would be more advantageous to use a special form of estimators such as the Basis Function (BF) algorithms, which focus on the parameter estimation aspect, such as Eq. (4.6). BF estimators suffer from computational complexity which is very demanding.

The basis function set consist of the basis exponentials $\{f_1(s), f_2(s), \ldots, f_k(s), s \in T_t\}$, where $T_t = [1, 2, \ldots, t]$ is the expanding time analysis window. Each basis element is given by

$$f_l(s) = e^{jw_l s}, \qquad l = 1, \ldots, k \tag{4.8}$$

It is instructive to mention that, the basis function are linearly independent in the time frame $T_t$, if all the frequency components differ, i.e., $w_i \neq w_l$ for $i \neq l$.

The running basis (RB) function algorithm corresponds to the following forward-time description of coefficient changes[2]

$$\theta_i(s) = \sum_{l=1}^{k} a_{il} f_l(s), \qquad s \in T_t, \quad i = 1, \ldots, n \tag{4.9}$$

or in vector form by making use of the kronecker product, APPENIX D,

$$\begin{aligned} \boldsymbol{\theta}(s) &= \left(\mathbf{I}_n \otimes \mathbf{f}^T(s)\right)\boldsymbol{\alpha}(s) \\ &= \mathbf{D}(s)\boldsymbol{\alpha}(s) \end{aligned} \tag{4.10}$$

where $\boldsymbol{\theta}(s)$ is the vector of periodically time varying system coefficients given by Eq. (4.5), $\mathbf{D}(s) = \mathbf{I}_n \otimes \mathbf{f}^T(s)$, $\mathbf{I}_n$ is $n$-by-$n$ identity matrix, $\boldsymbol{\alpha}(s)$ is $n$-by-$k$ matrix of

[2]The description is stacked to the terminology introduced in [16]

unknown coefficients, which we may represent here as a vector of dimension *nk*-by-*1,* given by

$$\boldsymbol{\alpha} = \left[a_{11}, a_{12}, \ldots, a_{1k}, \ldots, a_{n1}, a_{n2}, \ldots, a_{nk}\right]^T \tag{4.11}$$

$\mathbf{f}(s)$, vector of basis functions given by

$$\mathbf{f}(s) = \left[f_1(s), f_2(s), \ldots, f_k(s)\right]^T \tag{4.12}$$

Even though the vector of coefficients $\boldsymbol{\alpha}$ in Eq. (4.11) is written as a constant, its components $a_{il}$ are often slowly time varying. Actually, adaptive filter techniques are usually used in such applications, because of their ability to track slow variations in the parameters. However, using the kronecker product, we may facilitate the analysis further more. Hence, we can write the generalized form of the regression vector by

$$\begin{aligned}\boldsymbol{\psi}(\mathrm{s}) &= \left[u(s)\mathbf{f}^T(s), u(s-1)\mathbf{f}^T(s), \ldots, u(s-n+1)\mathbf{f}^T(s)\right]^T \\ &= \boldsymbol{\varphi}(s) \otimes \mathbf{f}(s)\end{aligned} \tag{4.13}$$

We need to estimate, we may rewrite the system equation (4.2) in terms of shorthand notation as follows

$$y(s) = \boldsymbol{\psi}^T(\mathrm{s})\boldsymbol{\alpha} + \upsilon(s), \quad s \in T_t \tag{4.14}$$

Clearly, if this modeling assumption is satisfied, which assumes that the coefficients vector is constant, then the unknown parameters might be considered as time-invariant elements. Therefore, Eq. (4.14) becomes a linear equation with respect to the unknown vector $\boldsymbol{\alpha}$.

Herewith, we may demonstrate that the problem of identification of a linear time-varying system of order *n* is switched to a problem of coefficient estimation of a linear time-invariant system of order *nk*.

## 4.3 Exponentially Weighted Basis Function (EWBF) Estimators

In this section, we focus on the most frequently used method of localization, which is the famous exponential forgetting technique, in order to apply to unknown parameter ($\boldsymbol{\alpha}$) estimation. As mentioned before, that the optimization least squares (LS) method in such cases suffers from an obvious drawback; as the time instant which we need to estimate the parameter at is increased, the estimator becomes less robust in the sense of its capability to match any parameter variations. In addition to that, it has an infinite memory that considers all the input samples from the starting point. However, the method of weighted least squares counters such drawbacks, by its ability to track small variations and it has finite memory, as pointed out earlier. Using the exponentially weighted least squares method, we can get the following estimate of $\boldsymbol{\alpha}$:

$$\begin{aligned}\hat{\boldsymbol{\alpha}}(t) &= \arg\min_{\boldsymbol{\alpha}} \sum_{i=0}^{t-1} \lambda^i \left| y(t-i) - \boldsymbol{\psi}^T(t-i)\boldsymbol{\alpha} \right|^2 \\ &= \left(\mathbf{R}_+^*(t)\right)^{-1} \mathbf{s}_+(t)\end{aligned} \tag{4.15}$$

where

$$\mathbf{R}_+(t) = \sum_{i=0}^{t-1} \lambda^i \boldsymbol{\psi}(t-i)\boldsymbol{\psi}^H(t-i) = \lambda \mathbf{R}_+(t-1) + \boldsymbol{\psi}(t)\boldsymbol{\psi}^H(t) \tag{4.16}$$

$$\mathbf{s}_+(t) = \sum_{i=0}^{t-1} \lambda^i y(t-i)\boldsymbol{\psi}^*(t-i) = \lambda \mathbf{s}_+(t-1) + y(t)\boldsymbol{\psi}^*(t) \tag{4.17}$$

(the + ve sign is used here to indicate forward-time model), $\mathbf{R}_+(t)$ is the *nk*-by-*nk* input correlation matrix $\boldsymbol{\psi}(t)$, and $\mathbf{s}_+(t)$ is the *(nk*-by-*1)* crosscorrelation matrix between the desired response $y(t)$ and the input vector $\boldsymbol{\psi}(t)$.

## 4.3.1 Running Basis Algorithm (RB-EWBF)

To introduce the running basis version (RB) of exponentially weighted basis function (EWBF). We applied the well known matrix inversion lemma, APENDIX B, to Eq. (4.16), considering that the initial value is chosen properly to ensure that the correlation matrix $\mathbf{R}_+(t)$ is positive definite. The parameters in the matrix inversion lemma are chosen as follows

$$\begin{aligned}
&\mathrm{A} = \lambda \mathbf{R}_+(t) \\
&\mathrm{B} = \boldsymbol{\psi}(t) \\
&\mathrm{C} = \boldsymbol{\psi}^H(t) \\
&\mathrm{D} = -1
\end{aligned} \tag{4.18}$$

Substituting these parameters in the matrix inversion lemma, and for more convenience we substituted $\boldsymbol{P}_+(t)$ to be equal to the inverse of the correlation matrix ($\boldsymbol{P}_+(t) = \mathbf{R}_+^{-1}(t)$), we then obtained the following recursive equation for the inverse autocorrelation matrix $\boldsymbol{P}_+(t)$

$$\boldsymbol{P}_+(t) = \lambda^{-1}\left[\boldsymbol{P}_+(t-1) - \boldsymbol{P}_+(t-1)\frac{\boldsymbol{\psi}(t)\boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)}{\lambda + \boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)\boldsymbol{\psi}(t)}\right] \tag{4.19}$$

Considering the gain factor $\boldsymbol{K}_+(t)$ to be

$$\boldsymbol{K}_+(t) = \frac{\boldsymbol{P}_+(t-1)\boldsymbol{\psi}(t)}{\lambda + \boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)\boldsymbol{\psi}(t)} \tag{4.20}$$

then, Eq. (4.19) becomes

$$\boldsymbol{P}_+(t) = \lambda^{-1}\left[\boldsymbol{P}_+(t-1) - \boldsymbol{K}_+(t)\boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)\right] \tag{4.21}$$

By considering Eq. (4.20) and representing it in a different form, it gives the definition of the gain factor, as follows

$$\boldsymbol{K}_+(t) = \left[\lambda^{-1}\boldsymbol{P}_+(t-1) - \lambda^{-1}\boldsymbol{K}_+(t)\boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)\right]\boldsymbol{\psi}(t) \tag{4.22}$$

It's clear from Eq. (4.22) that the term inside the bracket equals the inverse of the correlation matrix $\boldsymbol{P}_+(t)$ in Eq. (21). Hence, the gain factor can be explained in terms of the inverse correlation matrix and the general form of the regression vector $\boldsymbol{\psi}(t)$, as follows:

$$\boldsymbol{K}_+(t) = \boldsymbol{P}_+(t)\boldsymbol{\psi}(t) \tag{4.23}$$

To get the RB-EWBF estimator for the coefficient vector $\hat{\boldsymbol{\alpha}}(t)$, we substituted the recursive form of the crosscorrelation vector $\boldsymbol{P}_+(t)$ given in Eq. (4.17) in the main equation (4.15), then we obtain

$$\begin{aligned}
\hat{\boldsymbol{\alpha}}(t) &= \boldsymbol{P}_+^*(t)\lambda\mathbf{s}_+(t-1) + \boldsymbol{P}_+^*(t)y(t)\boldsymbol{\psi}^*(t) \\
&= \lambda^{-1}\left[\boldsymbol{P}_+(t-1) - \boldsymbol{K}_+(t)\boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)\right]^* \lambda\mathbf{s}_+(t-1) + \boldsymbol{P}_+^*(t)y(t)\boldsymbol{\psi}^*(t) \\
&= \hat{\boldsymbol{\alpha}}(t-1) - \boldsymbol{K}_+^*(t)\boldsymbol{\psi}^T(t)\hat{\boldsymbol{\alpha}}(t-1) + y(t)\boldsymbol{P}_+^*(t)\boldsymbol{\psi}^*(t) \\
&= \hat{\boldsymbol{\alpha}}(t-1) - \boldsymbol{K}_+^*(t)\boldsymbol{\psi}^T(t)\hat{\boldsymbol{\alpha}}(t-1) + y(t)\boldsymbol{K}_+^*(t) \\
&= \hat{\boldsymbol{\alpha}}(t-1) + \boldsymbol{K}_+^*(t)\varepsilon(t)
\end{aligned} \tag{4.24}$$

where, $\varepsilon(t)$ is the a priori estimation error given by

$$\varepsilon(t) = y(t) - \boldsymbol{\psi}^T(t)\hat{\boldsymbol{\alpha}}(t-1) \tag{4.25}$$

Since we present here the running basis version of the exponentially weighted basis function, we may write the basis function vector $\mathbf{f}(t)$ in terms of $\mathbf{f}(t-1)$, as follows

$$\mathbf{f}(t) = \mathbf{A}\mathbf{f}(t-1) \tag{4.26}$$

where, $\mathbf{A} = \operatorname{diag}\left\{e^{jw_1}, e^{jw_2}, \ldots, e^{jw_k}\right\}$.

In the RLS-type recursive algorithms, to avoid inversion of the correlation matrix $\mathbf{R}_+(t)$ at the beginning of the estimation, we choose the following initial condition:

$$\hat{\boldsymbol{\alpha}}(0) = 0,$$
$$\boldsymbol{P}_+(0) = \eta \mathbf{I}_{nk}$$

where $\eta = \delta^{-1}$ is a large positive constant number, which is a standard initialization for all RLS-type algorithms [1].

The complete RB-EWBF estimator is summed up in the following Table (4.1).

Table 4.1: Summary of RB-EWBF Estimator

| | |
|---|---|
| Initialization: | $\boldsymbol{P}_+(0) = \eta \mathbf{I}_{nk}$<br>$\hat{\boldsymbol{\alpha}}(0) = 0$<br><br>$\eta$: is a large positive constant.<br><br>$\mathbf{I}_{nk}$ : $nk$-by- $nk$ identity matrix |
| Iteration: | $\hat{\boldsymbol{\theta}}(t)=\mathbf{D}(t)\hat{\boldsymbol{\alpha}}(t)$<br>$\hat{\boldsymbol{\alpha}}(t) = \hat{\boldsymbol{\alpha}}(t-1) + \boldsymbol{K}_+^*(t)\varepsilon(t)$<br>$\varepsilon(t) = y(t) - \boldsymbol{\psi}^T(t)\hat{\boldsymbol{\alpha}}(t-1)$<br>$\boldsymbol{\psi}(t)=\boldsymbol{\varphi}(t)\otimes\mathbf{f}(t)$<br>$\mathbf{f}(t) = \mathbf{A}\mathbf{f}(t\text{-}1)$<br>$\boldsymbol{K}_+(t) = \dfrac{\boldsymbol{P}_+(t-1)\boldsymbol{\psi}(t)}{\lambda + \boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)\boldsymbol{\psi}(t)}$<br>$\boldsymbol{P}_+(t) = \dfrac{1}{\lambda}\left[\boldsymbol{P}_+(t-1) - \boldsymbol{K}_+(t)\boldsymbol{\psi}^H(t)\boldsymbol{P}_+(t-1)\right]$ |

## 4.3.2 Fixed Basis Algorithm (FB EWBF)

This is another equivalent form that represents the EWBF, which has slight reduction in the computational complexity over the running basis approach. It can be derived from the RB-EWBF by using the linear time-varying transform (rotating frame) given by

$$\begin{aligned}\hat{\boldsymbol{\beta}}(t) &= \mathbf{A}_n^{(t+1)}\hat{\boldsymbol{\alpha}}(t)\\ \boldsymbol{P}_{-}(t) &= \mathbf{A}_n^{-(t+1)}\boldsymbol{P}_{+}(t)\mathbf{A}_n^{(t+1)}\\ \boldsymbol{K}_{-}(t) &= \mathbf{A}_n^{-t}\boldsymbol{K}_{+}(t)\end{aligned} \tag{4.27}$$

where

$$\mathbf{A}_n = \mathbf{I}_n \otimes \mathbf{A}$$

From Eq. (4.10), substitute for the value of $\hat{\boldsymbol{\alpha}}(t)$ given in Eq. (4.27), we obtain

$$\hat{\boldsymbol{\theta}}(t) = \mathbf{D}_0\hat{\boldsymbol{\beta}}(t) \tag{4.28}$$

where

$$\begin{aligned}\hat{\boldsymbol{\theta}}(t) &= \mathbf{D}(t)\mathbf{A}_n^{-(t+1)}\\ &= \left(\mathbf{I}_n \otimes \mathbf{f}^T(t)\right)\left(\mathbf{I}_n \otimes \mathbf{A}^{-(t+1)}\right)\\ &= \mathbf{I}_n \otimes \left(\mathbf{A}^{-(t+1)}\mathbf{f}(t)\right)^T\\ &= \mathbf{I}_n \otimes \mathbf{f}^H(1)\end{aligned} \tag{4.29}$$

The last two results in Eq. (4.29) follow from the Kronecker products identity, given in APPENDIX D, and from the fact that

$$\mathbf{A}^{-s}\mathbf{f}(t) = \mathbf{f}(t-s) = \mathbf{f}^*(s-t) \tag{4.30}$$

Following the time-varying transform in Eq. (4.27), we try to find the vector of coefficients $\hat{\boldsymbol{\beta}}(t)$ by substituting the value of $\hat{\boldsymbol{\alpha}}(t)$ from Eq. (4.24). This yields

$$\begin{aligned}\hat{\boldsymbol{\beta}}(t) &= \mathrm{A}_n^{(t+1)}\hat{\boldsymbol{\alpha}}(t) \\ &= \mathrm{A}_n^{(t+1)}\left[\hat{\boldsymbol{\alpha}}(t-1)+\boldsymbol{K}_+^*(t)\varepsilon(t)\right] \\ &= \mathrm{A}_n\left[\hat{\boldsymbol{\beta}}(t-1)+\boldsymbol{K}_-^*(t)\varepsilon(t)\right]\end{aligned} \tag{4.31}$$

Consider rewriting the prior estimation error $\varepsilon(t)$ given in Eq. (4.25) in terms of the coefficient vector. Direct substitution gives

$$\varepsilon(t) = y(t) - \boldsymbol{\zeta}^T(\mathrm{t})\hat{\boldsymbol{\beta}}(t-1) \tag{4.32}$$

where $\boldsymbol{\zeta}(\mathrm{t})$ is the generalized regression vector given by

$$\begin{aligned}\boldsymbol{\zeta}(t) &= \mathrm{A}_n^{-t}\psi(t) = \left(\mathbf{I}_n \otimes \mathrm{A}^{-t}\right)\left(\varphi(t)\otimes\mathbf{f}(t)\right) \\ &= \varphi(t)\otimes\mathbf{f}(0) = \varphi(t)\otimes\left[1,\ldots,1\right]^T\end{aligned} \tag{4.33}$$

The last result in Eq. (4.33) follows from the Kronecker products identity, and from the fact given in Eq. (4.30). Now, by considering Eq. (4.20) and the value of $\boldsymbol{\zeta}(\mathrm{t})$ in Eq. (4.33) we can write the gain factor $\boldsymbol{K}_-(t)$ as

$$\begin{aligned}\boldsymbol{K}_-(t) &= \mathrm{A}_n^{-t}\boldsymbol{K}_+(t) \\ &= \frac{\boldsymbol{P}_-(t-1)\boldsymbol{\zeta}(\mathrm{t})}{\lambda + \boldsymbol{\zeta}^H(\mathrm{t})\boldsymbol{P}_+(t-1)\boldsymbol{\zeta}(\mathrm{t})}\end{aligned} \tag{4.34}$$

The time-varying transform of the inverse autocorrelation $\boldsymbol{P}_-(t)$, by direct substitution of the value of $\boldsymbol{P}_+(t)$ given in Eq. (4.21), can be represented by

$$\begin{aligned}\boldsymbol{P}_-(t) &= \mathrm{A}_n^{-(t+1)}\boldsymbol{P}_+(t)\mathrm{A}_n^{(t+1)} \\ &= \frac{1}{\lambda}\mathrm{A}_n^*\left[\boldsymbol{P}_-(t-1)-\boldsymbol{K}_-(t)\boldsymbol{\zeta}^H(\mathrm{t})\boldsymbol{P}_-(t-1)\right]\mathrm{A}_n\end{aligned} \tag{4.35}$$

It's clear in the FB-EWBF algorithm that; the evaluation process of the generalized regression vector $\boldsymbol{\zeta}(\mathrm{t})$ in Eq. (4.33) does not involve any arithmetic operation; hence, there is no need to update the basis vector $\mathbf{f}(t)$. Furthermore, the matrix $\mathbf{A}_n$ is diagonal; all of these properties show that, there is reduction in the computational cost over the RB-EWBF estimator (as we will show later) when the time-invariant coordinates are changed with the time-varying coordinates. The FB-EWBF is summarized in table (4.2).

It is interesting to note that the reparameterized (FB-EWBF) algorithm given in table (4.2) is referred to the following backward time-description of parameter changes:

$$\theta_i(t-s+1)=\sum_{l=1}^{k} b_{il} f_l^*(s), \qquad s\in T_t, \quad i=1,\ldots,n \tag{4.36}$$

where

$$b_{il}=e^{j\omega_l(t+1)}a_{il}, \qquad , \quad i=1,\ldots,n \quad l=1,\ldots,k$$

The backward time-description in Eq. (4.36) and the forward time-description given in Eq. (4.9) are both equal as have been shown in [16]. Nevertheless, RB and FB-EWBF are both strictly input-output equivalent [26], i.e. they give identical parameter estimates $\hat{\theta}(t)$ for identical data set $(u(s), y(s), \; s=1,\ldots,t)$ and appropriately chosen initial conditions.

Table 4.2: Summary of FB-EWBF Estimator

| | |
|---|---|
| Initialization: | $\boldsymbol{P}_{-}(0) = \eta \mathbf{I}_{nk}$<br>$\hat{\boldsymbol{\beta}}(0) = 0$<br>$\eta$ : is a large positive constant.<br>$\mathbf{I}_{nk}$ : $nk$-by- $nk$ identity matrix |
| Iteration: | $\hat{\boldsymbol{\theta}}(\mathrm{t}) = \mathbf{D}_0 \hat{\boldsymbol{\beta}}(t)$<br>$\hat{\boldsymbol{\beta}}(t) = \mathrm{A}_n \left[ \hat{\boldsymbol{\beta}}(t-1) + \boldsymbol{K}_{-}^{*}(t) \varepsilon(t) \right]$<br>$\varepsilon(t) = y(t) - \boldsymbol{\zeta}^{T}(\mathrm{t}) \hat{\boldsymbol{\beta}}(t-1)$<br>$\boldsymbol{K}_{-}(t) = \dfrac{\boldsymbol{P}_{-}(t-1)\boldsymbol{\zeta}(\mathrm{t})}{\lambda + \boldsymbol{\zeta}^{H}(\mathrm{t}) \boldsymbol{P}_{+}(t-1) \boldsymbol{\zeta}(\mathrm{t})}$<br>$\boldsymbol{P}_{-}(t) = \dfrac{1}{\lambda} \mathrm{A}_n^{*} \left[ \boldsymbol{P}_{-}(t-1) - \boldsymbol{K}_{-}(t) \boldsymbol{\zeta}^{H}(\mathrm{t}) \boldsymbol{P}_{-}(t-1) \right] \mathbf{A}_n$ |

## 4.4 Gradient Estimators

These algorithms are extended from the standard LMS algorithm, which are used in the time-invariant case. It is well known that RLS type algorithms such as EWBF, converge fast, whereas they are very demanding regarding computational complexity. However, in contrast, the stochastic gradient algorithm does not have any, as we show later. Therefore, using the LMS method, we can estimate the parameter vector $\hat{\boldsymbol{\alpha}}(t)$, by minimizing the following error power problem:

$$\hat{\boldsymbol{\alpha}}(t) = \arg \min_{\boldsymbol{\alpha}} E\left\{ \left| y(t) - \boldsymbol{\psi}^{T}(t) \boldsymbol{\alpha} \right|^{2} \right\} \tag{4.37}$$

### 4.4.1 Running Basis-Gradient Algorithm

The low complexity and simple stochastic gradient algorithm can be obtained easily by replacing the inverse correlation matrix $\boldsymbol{P}_{+}$ in RB-EWBF estimator, that need ($nk$-

by-*nk)* matrix updating at each iteration, with the scalar step size $\mu > 0$ [4]. Then the gain factor in Eq. (4.23) becomes

$$\boldsymbol{K}_{+}(t) = \mu\boldsymbol{\psi}(t) \tag{4.38}$$

The basis function obtained is termed as a running-basis (RB) gradient algorithm, and is summarized in Table (4.3).

Table 4.3: Summary of RB-Gradient Estimator

| | |
|---|---|
| Initialization: | $\hat{\boldsymbol{\alpha}}(0) = 0$ <br> $\mu$ : small number |
| Iteration: | $\hat{\boldsymbol{\theta}}(t)=\mathbf{D}(t)\hat{\boldsymbol{\alpha}}(t)$ <br> $\hat{\boldsymbol{\alpha}}(t) = \hat{\boldsymbol{\alpha}}(t-1) + \mu\boldsymbol{\psi}^{*}(t)\varepsilon(t)$ <br> $\boldsymbol{\psi}(t)=\boldsymbol{\varphi}(t)\otimes\mathbf{f}(t)$ <br> $\mathbf{f}(t) = \mathbf{Af}(t\text{-}1)$ <br> $\varepsilon(t) = y(t) - \boldsymbol{\psi}^{T}(t)\hat{\boldsymbol{\alpha}}(t-1)$ |

## 4.4.2 Fixed Basis-Gradient Algorithm

The equivalent extension of the RB-gradient to Fixed Basis (FB) algorithm easily can easily be obtained, by using the linear time-varying transform as have been done in the FB-EWBF, by using different system of coordinates, or by replacing $\boldsymbol{P}_{-}$ matrix by the step size $\mu > 0$. Then the generated estimator is called as fixed basis-gradient estimator, and is summarized in Table (4.4)

Table 4.4: Summary of FB-Gradient Estimator

| | |
|---|---|
| Initialization: | $\hat{\boldsymbol{\alpha}}(0)=0$ |
| | $\mu$ : small number |
| Iteration: | $\hat{\boldsymbol{\theta}}(\text{t})=\mathbf{D}_0\hat{\boldsymbol{\beta}}(t)$ <br> $\hat{\boldsymbol{\beta}}(t)=\mathrm{A}_n\left[\hat{\boldsymbol{\beta}}(t-1)+\mu\boldsymbol{\zeta}^*(t)\varepsilon(t)\right]$ <br> $\varepsilon(t)=y(t)-\boldsymbol{\zeta}^T(\text{t})\hat{\boldsymbol{\beta}}(t-1)$ |

We show later that, the FB-gradient estimator shown in table (4.4), is superior to the running basis version in terms of computational complexity.

## 4.5 Recursive Inverse Basis Function (RIBF) Estimators

The newly proposed Recursive Inverse (RI) algorithm has been applied successfully in many signal processing areas like image processing [25], ALE [3], and channel equalization [17]. In this section we propose the Recursive Inverse Basis Function (RIBF) estimator and its adaptive-frequency version. Its application on fading channel equalization problem is discussed in chapter 5 to show the superiority of the proposed algorithm over the others.

### 4.5.1 Running Basis (RB-RIBF) Algorithm

Using the Wiener-Hopf equation (3.68) and rewriting it in terms of system coefficients ( $\hat{\boldsymbol{\alpha}}$ ) estimation, we have

$$\hat{\boldsymbol{\alpha}}(t)=\mathbf{R}^{-1}(t)\mathbf{s}(\text{t}) \tag{4.39}$$

Solving the Wiener-Hopf Equation iteratively by considering the estimation of the correlation parameters iteratively in complex form and following the same derivation procedure introduced for RI-algorithm in section (3.4), then, we can easily achieve the RB-RIBF estimator as shown in table (4.5).

## 4.5.2 Doppler Frequency Estimation

The proposed equalization algorithms would not be inclusive without a method for estimating and tracking the slight changes of the Doppler frequencies, i.e. although the BF estimators are robust to small local changes in frequencies around the known specified values, they fail to identify the system properly in the presence of a frequency drift [2].

Table 4.5: Summary of RB-RIBF Estimator

| | |
|---|---|
| Initialization: | $\mathbf{s}_+(0)=0$ <br> $\hat{\boldsymbol{\alpha}}(0)=0$ <br> $\mathbf{R(0)}=0$ |
| Iteration: | $\hat{\boldsymbol{\theta}}(t)=\mathbf{D}(t)\hat{\boldsymbol{\alpha}}(t)$ <br> $\mu(t)=\dfrac{\mu_0}{1-\lambda^t}$ <br> $\hat{\boldsymbol{\alpha}}(t)=\left[\mathbf{I}_{nk}-\mu(t)\mathbf{R}_+(t)\right]\hat{\boldsymbol{\alpha}}(t-1)+\mu(t)\mathbf{s}_+(\mathrm{t})$ <br> $\boldsymbol{\psi}(t)=\boldsymbol{\varphi}(t)\otimes\mathbf{f}(t)$ <br> $\mathbf{f}(t)=\mathbf{Af}(t\text{-}1)$ <br> $\mathbf{R}_+(\mathrm{t})=\lambda\mathbf{R}_+(\mathrm{t}-1)+\boldsymbol{\psi}(t)\boldsymbol{\psi}^H(t)$ <br> $\mathbf{s}_+(\mathrm{t})=\lambda\mathbf{s}_+(\mathrm{t}-1)+\mathrm{y}(t)\boldsymbol{\psi}^*(t)$ |

Therefore, we adopt a simple gradient search strategy, which is proposed in [4], to derive the frequency-adaptive version of the RIBF algorithm.

Let the gradient be

$$J(t,\omega)=\frac{1}{2}\left|\varepsilon(t,\omega)\right|^2 \tag{4.40}$$

where $\boldsymbol{\omega}=\left[\omega_1,\ldots,\omega_k\right]^T$ are the instantaneous frequencies to be tracked.

The simple gradient algorithm which minimizes (4.40) can be stated in the form

$$\hat{\boldsymbol{\omega}}(t+1)=\hat{\boldsymbol{\omega}}(t)-\mu\nabla J(\hat{\boldsymbol{\omega}}(t)) \tag{4.41}$$

where $\nabla J(\hat{\boldsymbol{\omega}}(t))$ denotes the derivative of $J(t,\omega)$ with respect to the frequencies $\boldsymbol{\omega}$, evaluated at the instant $\hat{\boldsymbol{\omega}}(t)$, and $\mu > 0$ is a small adaption constant.

In the case of frequency estimation we did a little modification in the form of presenting the general regression vector and the estimated parameters by arranging them according to their respective different frequency components ($\omega_1,\ldots,\omega_k$). The original periodically time-varying coefficients are,

$$\theta_i(t) = \sum_{l=1}^{k} a_{il} e^{j\omega_l t}$$

Representing $\boldsymbol{\alpha}_l = \left[a_{1l},\ldots,a_{nl}\right]^T$ the vector of system coefficients corresponds to a particular frequency $\omega_l$. Similarly, let $\boldsymbol{\psi}_l(t)=\boldsymbol{\varphi}(t)e^{j\omega_l t}$ be the generalized regression vector that corresponds to the *l*th frequency component. Accordingly, the error will be given as

$$\varepsilon(t) = y(t) - \sum_{l=1}^{k} \hat{\boldsymbol{\psi}}_l^T(t)\hat{\boldsymbol{\alpha}}_l(t-1) = y(t) - \hat{\boldsymbol{\psi}}^T(t)\hat{\boldsymbol{\alpha}}(t-1)$$

where $\hat{\boldsymbol{\psi}}(t) = \left[\hat{\boldsymbol{\psi}}_1^T(t),\ldots,\hat{\boldsymbol{\psi}}_l^T(t)\right]^T$, and $\hat{\boldsymbol{\alpha}}(t-1) = \left[\hat{\boldsymbol{\alpha}}_1^T(t),\ldots,\hat{\boldsymbol{\alpha}}_l^T(t)\right]^T$.

Therefore,

$$\begin{aligned}\nabla J_l(\hat{\boldsymbol{\omega}}(t)) &= \frac{\partial J(t,\boldsymbol{\omega})}{\partial \omega_l(t)}|_{\omega_l} \\ &= \mathrm{Re}\left\{ j\varepsilon(t)e^{-j\omega_l}\boldsymbol{\varphi}^H(t)\hat{\boldsymbol{\alpha}}_l^*(t-1)\right\} \\ &= \mathrm{Im}\left\{\varepsilon^*(t)\hat{\boldsymbol{\psi}}_l^T\hat{\boldsymbol{\alpha}}_l^*(t-1)\right\}\end{aligned}$$

where $\hat{\boldsymbol{\psi}}_l(t)=\hat{f}_l(t)\boldsymbol{\varphi}(t)$, $\quad \hat{f}_l(t) = e^{j\hat{\omega}_l t}\hat{f}_l(t-1)$.

The number and the values of system frequencies have to be initialized properly in order to avoid divergence of the frequency adaptive BF-estimators. In the channel identification, the known (training) part can be employed for this purpose. The angular frequencies of the periodically-varying systems can be initialized based on the analysis method of the higher order input/output signal statistics (these methods are further described in [4],[15]) or using the methods of sliding window least squares estimates of system coefficients, proposed in [28]. Table (4.6) summarizes the frequency-adaptive version of the RIBF algorithm.

Table 4.6: Summary of the Frequency-Adaptive RIBF Estimator.

| | |
|---|---|
| Initialization: | $\mathbf{s}_{+}(0)=0$<br>$\hat{\boldsymbol{\alpha}}(0)=0$<br>$\mathbf{R}_{+}(\mathbf{0})=0$<br>$\hat{\boldsymbol{\omega}}^{0}=\left[\omega_{l}^{0},\ldots,\omega_{k}^{0}\right]^{T}$<br>$\mu_0$ : is small positive constant.<br>$\lambda$ Less but much close to one. |
| Iteration: | $\hat{f}_l(t)=e^{j\hat{\omega}_l t}\hat{f}_l(t-1)$<br>$\hat{\boldsymbol{\psi}}_l(t)=\hat{f}_l(t)\boldsymbol{\varphi}(t),$<br>$l=1,\ldots,k$<br>$\hat{\boldsymbol{\psi}}(t)=\left[\hat{\boldsymbol{\psi}}_1^{T}(t),\ldots,\hat{\boldsymbol{\psi}}_l^{T}(t)\right]^{T}$<br>$\varepsilon(t)=y(t)-\hat{\boldsymbol{\psi}}^{T}(t)\hat{\boldsymbol{\alpha}}(t-1)$<br>$g_l(t)=\operatorname{Im}\left\{\varepsilon^{*}(t)\hat{\boldsymbol{\psi}}^{T}\hat{\boldsymbol{\alpha}}_l^{*}(t-1)\right\}$<br>$\hat{\boldsymbol{\omega}}(t+1)=\hat{\boldsymbol{\omega}}(t)-\mu g_l(t)$<br>$l=1,\ldots,k$<br>$\mu(t)=\dfrac{\mu_0}{1-\lambda^{t}}$<br>$\hat{\boldsymbol{\alpha}}(t)=\left[\mathbf{I}_{nk}-\mu(t)\mathbf{R}_{+}(t)\right]\hat{\boldsymbol{\alpha}}(t-1)+\mu(t)\mathbf{s}_{+}(\mathrm{t})$<br>$\mathbf{R}_{+}(\mathrm{t})=\lambda\mathbf{R}_{+}(\mathrm{t}-1)+\boldsymbol{\psi}(t)\boldsymbol{\psi}^{H}(t)$<br>$\mathbf{s}_{+}(\mathrm{t})=\lambda\mathbf{s}_{+}(\mathrm{t}-1)+\mathrm{y}(t)\boldsymbol{\psi}^{*}(t)$<br>$\hat{\boldsymbol{\theta}}(t)=\sum_{l=1}^{k}\hat{f}_l(t)\hat{\boldsymbol{\alpha}}_l(t)$ |

## 4.6 Computational Complexity

The computational complexity is an important measure, especially in real time applications. In the implementation of a real time system, hardware limitations may influence system performance. Therefore, computational complexity is one of the important characteristics that we refer to in the adaptive algorithm comparison.

Table (4.7) illustrates a comparison of the computational complexity for all the estimators that have been described. The computational complexity stands for the number of complex multiply/add operations per sampling time. In the evaluation we considered the fact that some of the involved matrices are Hermitian (i.e. $\boldsymbol{P}_+, \boldsymbol{P}_-$ and $\mathbf{R}_+$ ), hence, only their upper ( lower) triangular parts have to be updated.

The proposed RIBF-algorithms have a clear computational advantage over the EWBF-estimators (in both versions RB and FB), it reduces the complexity by $kn(kn-1)$ multiply/add operations. In addition to the reduction in the computational complexity, it also has the capability to converge to smaller error values in terms of mean square error (MSE) as we show in the simulation chapter. Interestingly, despite this significant reduction in the proposed RIBF estimator, more reduction may be achieved, if we use an approximation for the autocorrelation matrix to be as a Toeplitz matrix.

There is no doubt that the Gradient estimator has the minimum computational complexity over all the others due to its simplicity, as is clear in Table (4.7). However, the price paid for reduced complexity of gradient algorithms is their initial slow convergence.

Table 4.7: Comparison of Computational Complexity of Different BF-Algorithms

| **Algorithm** | **Complexity** |
|---|---|
| **RB-EWBF** | $2(kn)^2 + 6kn + k$ |
| **FB-EWBF** | $2(kn)^2 + 6kn$ |
| **RB-Gradient** | $4kn + k + 1$ |
| **FB-Gradient** | $3kn + k + 1$ |
| **RB-RIBF** | $(kn)^2 + 7kn + k$ |

# Chapter 5

# COMPUTER SIMULATIONS

In this section we set up an example where the proposed algorithms are applied in multipath fading channels. Accordingly, we simulated the mentioned algorithms using MATLAB software package. We present, as well, the performance and the superiority of the proposed RIBF-estimator over others. And we investigated the performance capability of the proposed frequency-adaptive version of the EWBF estimator.

## 5.1 The Test Case

The test case, which was adopted from [2] and [4], involve a periodically time-varying channel as stated in equations (4.2) and (4.6). There are two channel taps (coefficients, *n*=2). Usually, the channel coefficient's number is a small number that depends on the transmission-channel's memory. Each channel-tap is equal to the linear combination of three basis functions (number of dispersive paths, *k*=3), given by:

| | |
|---|---|
| $f_1(t) = 1,$ | First (direct) path. |
| $f_2(t) = e^{j(2\pi/T_2)},$ | Second (reflector) path. |
| $f_3(t) = e^{j(2\pi/T_3)},$ | Third (reflector) path. |

where $T_2 = 120$ and $T_3 = 200$ sample periods. These numbers are chosen to be close to the real values meant for the carrier frequency $f_c = 900$ MHz, a bit rate around 20 kbit/s, and a vehicle moving at 100 km/h.

Considering the simple adaptive equalization scheme that is shown in Figure (5.1), the input is assumed to be the 4-QAM (generated as $u(t) = \pm 1 \pm j$ with variance $\sigma_u^2 = 2$). After that, the input is filtered through the time-varying channel with coefficients ($\alpha$). The received signal $\theta(t)$ is corrupted with AWGN ($\sigma_v^2 = 0.4373$) noise. Consequently, the receiving end signal has an average SNR that is equal to 15 dB.

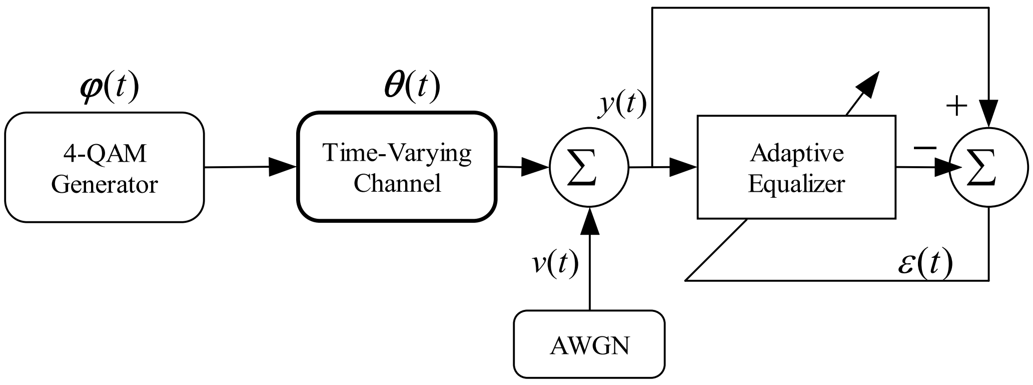

Figure 5.1: Simple Adaptive Equalization Scheme

The original 4-QAM signal constellation is shown in Figure (5.2.a), whereas, Figure (5.2.b) illustrates the effect of channel response over the transmitted signal.

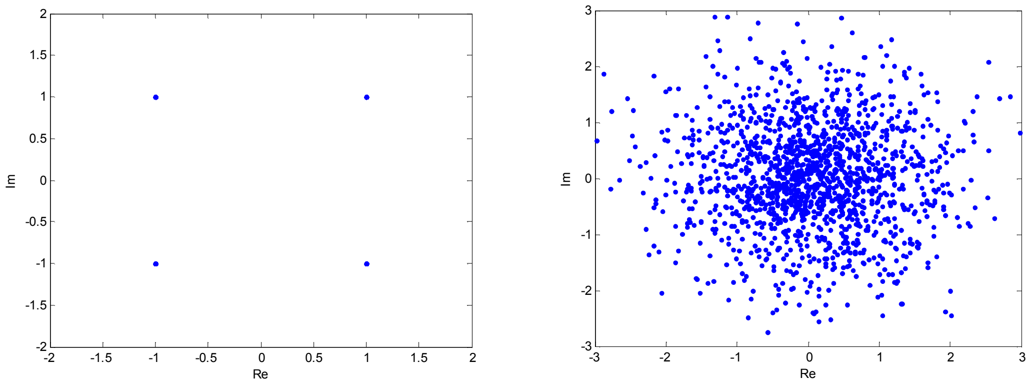

Figure 5.2: a. 4-QAM Installation Diagram, b. The Received Signal Before Equalization.

## 5.2 Matlab Simulation

In this section, we provide the performance of the proposed RB-RIBF, RB-EWBF and RB-Gradient estimators in terms of the mean square parameter estimation errors. In all the simulations results were obtained through Monte Carlo trials (the error averaged over 200 simulation runs). Figure (5.3) shows the MSE performance for RIBF estimator for different forgetting constants $\lambda$, with constant step size $\mu_0 = 0.000018$. Since the forgetting factor controls the memory of the estimator (it is an important characteristic that show the number of the past samples used to track the parameters effectively). Therefore, there is a trade-off between tracking speed and the memory size.

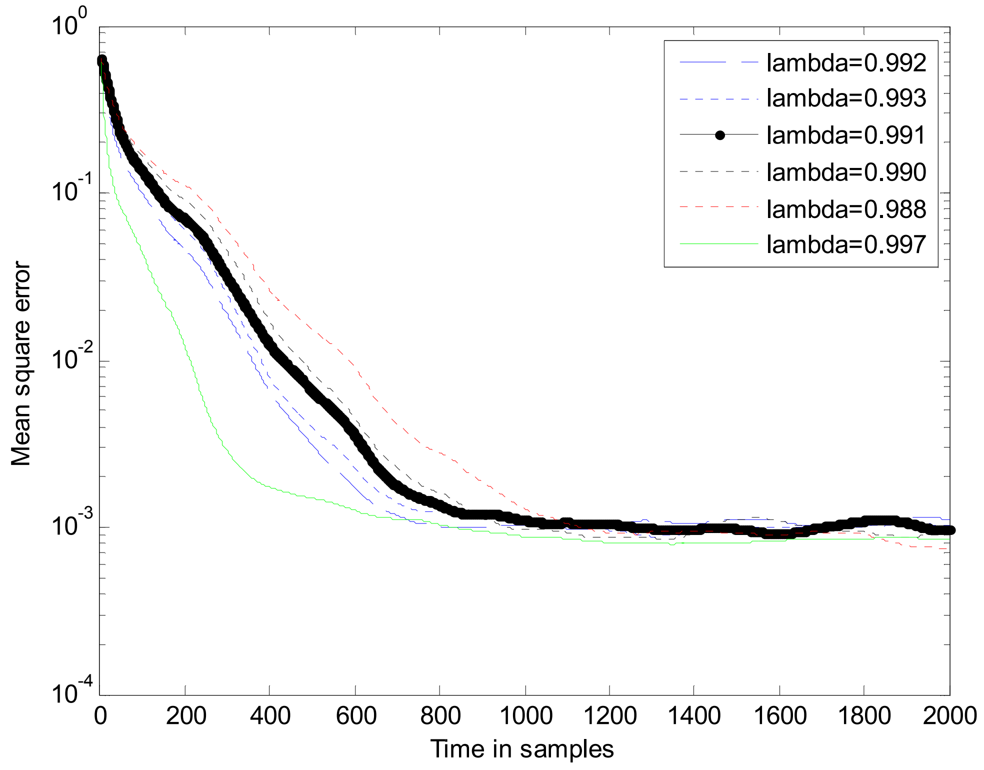

Figure 5.3: Mean Square Error Performance of the RIBF Estimator for Different Forgetting Factor and Constant Adapting Factor $\mu_0 = 0.000018$.

Clearly, Figure (5.3) shows that effect of the estimation memory on the tracking speed behavior, as the estimation memory increase the RIBF estimator behaves to converge faster. In addition to that, the steady-state MSE occupies a lower value. Furthermore, the variable step size has a different effect on the RIBF. Increasing the maximum value of the variable step size $\mu_0$, helps the estimator to converge faster, even though the MSE level goes up to a slightly higher value. This behavior is so obvious in Figure (5.4) which illustrates the performance of RIBF estimator for different adaptation step sizes $\mu_0$ at a constant forgetting factor $\lambda = 0.99$. At $\mu_0 = 2.0\times10^{-5}$, the MSE steady state value is 69 dB at time $t = 1050$.

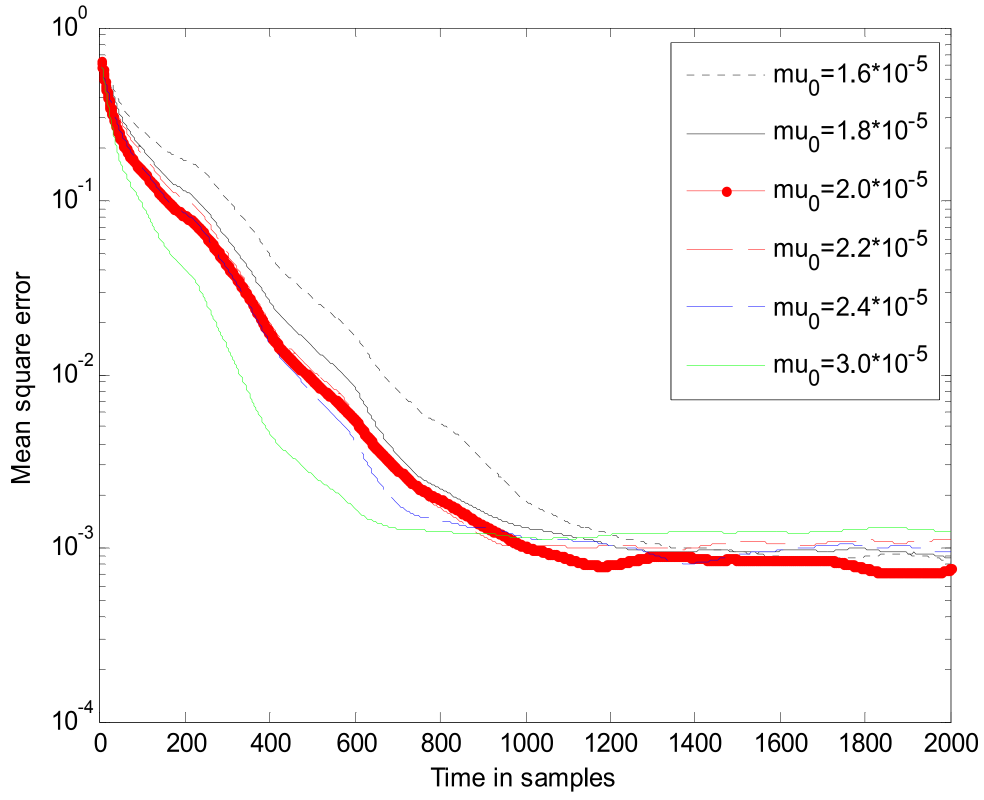

Figure 5.4: Mean Square Error Performance of the RIBF Estimator for Different Adapting Factor and Constant Forgetting Factor $\lambda = 0.99$.

Whereas, at $\mu_0 = 3.0 \times 10^{-5}$ the MSE goes up to a higher level (50 dB) and it converge faster at time $t = 650$. It is clear that, as the adaptation constant increases, the estimator converges faster and become less accurate.

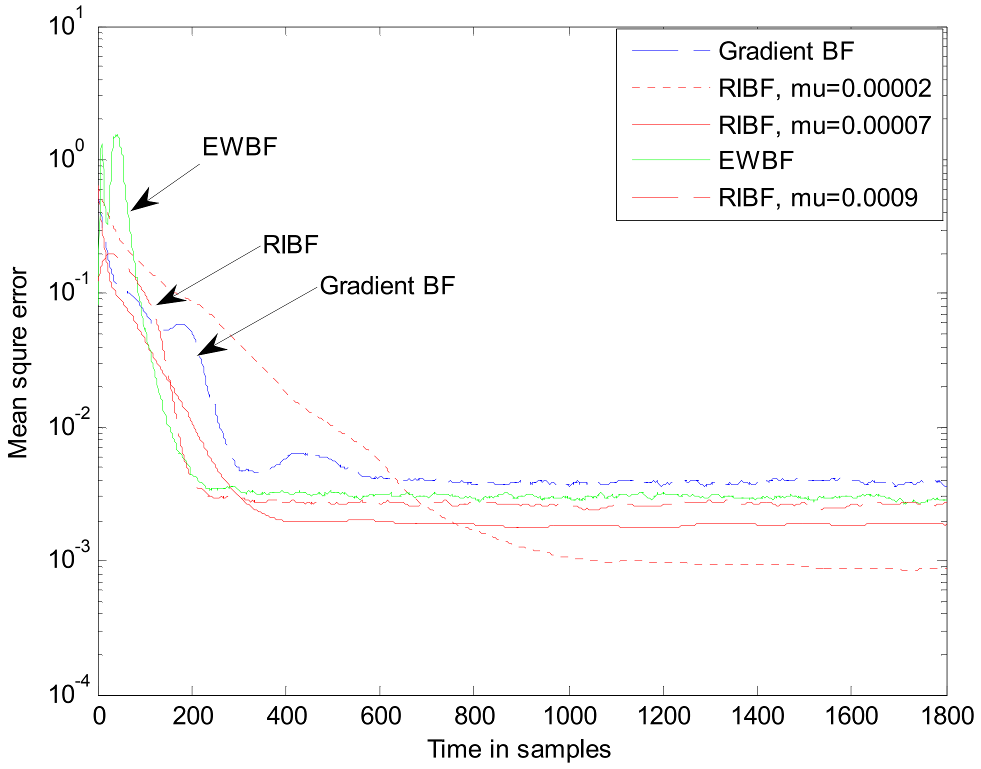

Figure 5.5: Comparison of Mean Square Parameter Estimation Error for Three Estimation Algorithms; Gradient-BF, EWBF and RIBF.

Figure (5.5) sets up the development of the mean square parameter estimation error (ensemble average over 200 realizations) that is obtained for the RB-EWBF algorithm as given in table (4.1), the Gradient-BF algorithm in table (4.3), and the proposed RIBF estimator summarized in table (4.5). All of the results in Figure (5.4) are taken for fixed forgetting factor ($\lambda = 0.99$) that corresponds to an estimation memory equals to approximately (58) samples. Besides, the step size (adaptation constant) for the gradient algorithm was set to $\mu = 0.00572$; such a value has misadjustment equivalent in the steady state to the other BF's missadjustment (which is the ratio of the algorithm MSE to the theoretical MSE), i.e., all the filters approximately have the same memory.

For the sack of comparison, Figure (5.5) demonstrate clearly the advantages of the RIBF estimator in terms of accuracy (by reaching MSE values for different step sizes ($\mu_0$)).The RIBF estimator superior to the EWBF estimator by about 8 db (at $\mu_0 = 0.00002$) and the Gradient-BF estimator by 12 dB. Even though, when the estimators coverage to the steady state MSE value at same time $t = 200$, the RIBF algorithm still have a slightly reduction in the MSE (about 1.3 dB) over EWBF and 1.55 dB over the Gradient-BF estimator.

The significant reduction in MSE value that has shown in Figure (5.5) combined with low computational complexity cost that illustrated in Table (4.7), promises with low BER without using any error correction code.

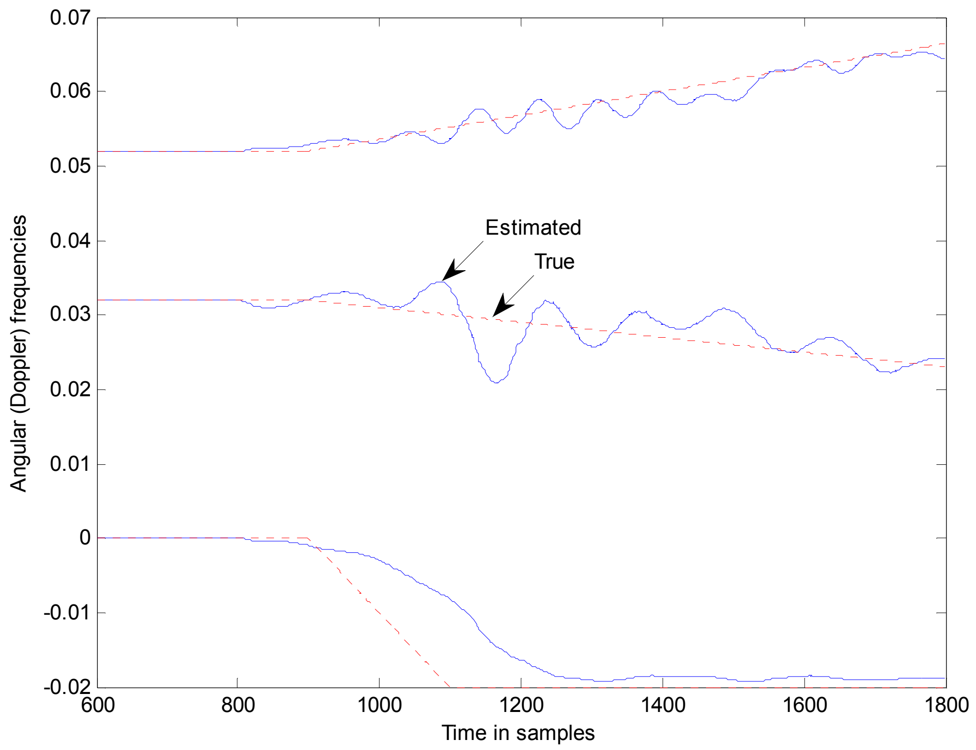

Figure 5.6: Adaptive Frequency-RIBF Estimator Response, True Frequencies (dotted lines) and Their Estimates (solid lines).

In all the previous simulations, the Doppler frequencies are assumed to be constant and known or estimate. Nevertheless, in the next simulation we introduce a frequency variation with different scales to different paths, and we investigate the adaptive-frequency RIBF estimator performance.

Figure (5.6) illustrates the tracking frequency capabilities of the developed RIBF estimator described in Table (4.6). The simulation settings were exactly the same as in the test case (that were ran over 200 realizations), except that the adaptation constant of gradient search algorithm's set to be ($\mu = 0.00035$) and the other parameters of RIBF are set as ($\lambda = 0.99$, $\mu_0 = 23 \times 10^{-6}$). The frequency drift starts after the estimator has reached its steady-state (that is at t =800).

It is clear, that the estimator has a good capability of tracking the frequency variations. It results with small oscillations in the paths that have slight drift, in contrast with the paths which has a significant drift (i.e. the direct path which has frequency drift 0.02).

Besides, Figure (5.7) shows the MSE performance of the adaptive frequency-RIBF estimator. It's clear that when the frequencies drift starts (nonstationary process) the MSE exhibit to jump and settle down a new MSE.

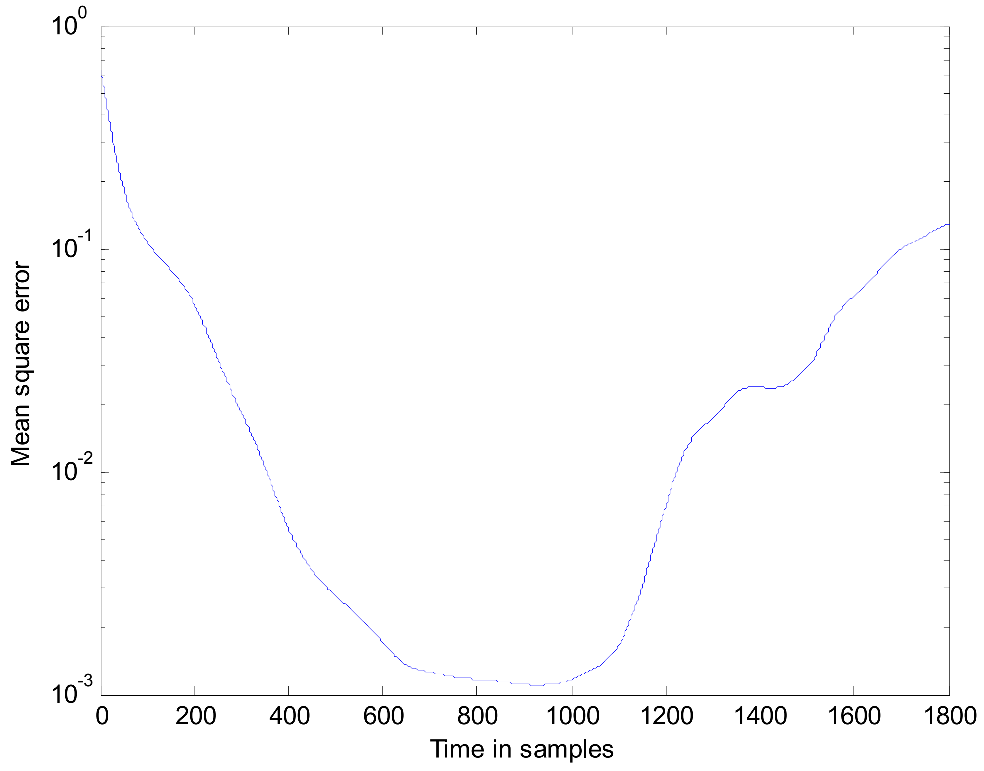

Figure 5.7: Mean Square Error Performance of the Adaptive Frequency-RIBF Estimator.

# Chapter 6

# CONCLUSIONS AND FUTURE WORK

## 6.1 Conclusions

The problem of identification and tracking of periodically varying systems has been considered. The classical basis function algorithms which were used to estimate and track the time-varying coefficients in such systems have a good tracking performance, yet they computationally are demanding. We have used the recursive inverse algorithm, which has been applied successfully in many digital signal processing problems, to propose a new recursive algorithm termed as the recursive inverse basis function estimator (RIBF). The proposed algorithm outperformed the classical basis function schemes in terms of complexity reduction and had better tracking performance, it superior to the EWBF by reducing the computational complexity by $kn(kn-1)$ multiply/add operations and it shows further reduction by 8 dB in the mean square parameter estimation error, whereas the other estimators can't reaches that level. Furthermore, an adaptive-frequency version of the recursive inverse exponential basis function algorithm was derived by employing a simple gradient search strategy. The tracking properties of the introduced algorithms were investigated by means of computer simulations.

## 6.2 Future Work

Digital Signal Processing (DSP) field has vast applications in many disciplines especially in communication area. In this thesis we explored a fundamental problem in mobile communication which is multipath fading channels. And we proposed a new adaptive estimator, which can handle part of the dispersive channel regarding to estimation of the channel coefficient (due to amplitude dispersion), assuming that the Doppler frequencies (phase dispersion) are given or estimated. As a future work, we are going to explore the problem of identification of periodically varying channels in more details. Then, we will try to enroll the RI adaptive algorithm for the more general case, which can be summed up as estimation and tracking of complex-valued quasi-periodically varying systems which are discussed in [26], [27]. The quasi-periodically time-varying is governed by the following two equations:

$$y(t) = \boldsymbol{\varphi}^T(t)\boldsymbol{\theta}(t) + \upsilon(t)$$

and

$$\theta_i(t) = \sum_{l=1}^{k} a_{il}(t) e^{j\sum_{s=1}^{t} w_l t}, \qquad i = 1,\ldots,n.$$

In this case it is clear that the amplitude and frequencies are time varying. There for, the system parameters are changing over time in a periodic-like but not exactly in a periodic manner.

# REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.

[2] M. Niedzwiecki and T. Klaput, "Fast algorithms for identification of periodically varying systems," *IEEE Transactions on Signal Processing*, vol. 51, no. 12, pp. 3270-3279, December 2003.

[3] M. S. Ahmad, O. Kukrer and A. Hocanin, “Recursive Inverse adaptive filtering algorithm,” *Elsevier Journal of Digital Signal Processing*, DOI:10.1016 j.dsp.2011.03.001, March 2011.

[4] M. K. Tsatsanis and G. B. Giannakis, “Modeling and equalization of rapidly fading channels,” *Int. J. Adaptive Contr. Signal Process.*, vol.10, pp. 159–176, 1996.

[5] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications*, New York: Wiley, 1998.

[6] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., “Stationary and non-stationary learning characteristics of the LMS adaptive filter,” *Proc. IEEE*, vol. 64, pp. 1151-1162, Aug. 1976.

[7] S. S. Narayan, A. M. Peterson, and M. J. Narashima, “Transform domain LMS algorithm*,” IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 609–615, Jun. 1983.

[8] S. Shankar Narayan and A. M. Peterson, “Frequency domain LMS algorithm,” *Proc.ZEEE*,vol. 69, pp.124-126, Jan.1981.

[9] Zhou, Y., Chan, S.C., and Ho, K.L.: ‘A new LMS/newton algorithm for robust adaptive filtering in impulsive noise’. *EUSIPCO* 2004, pp . 705– 708.

[10] G.-O. Glentis, K. Berberidis, and S. Theodoridis, “Efficient least squares adaptive algorithms for FIR transversal filtering,” *Signal Processing Mag*., vol. 16, pp. 13–41, July 1999.

[11] D. J. Tylavsky and G. R. L. Sohie, “Generalization of the Matrix Inversion Lemma”, *Proceedings of the IEEE*, vol. 74, no. 7, pp. 1050-1052, July.1986.

[12] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[13] G. Malik and A. S. Sappal, “Adaptive Equalization Algorithms: An Overview” *Int. J. of Advanced Computer Science and Applications*, Vol. 2, No.3 March 2011.

[14] U. S. H. Qureshi, “Adaptive equalization,” *Proc. IEEE*, vol. 73, no. 9, pp. 1347–1387, Sep. 1985.

[15] J. Bakkoury, D. Roviras, M. Ghogho, and F. Castanie, “Adaptive MLSE receiver over rapidly fading channels,” *Signal Process.*, vol. 80, pp.1347–1360, 2000.

[16] M. Niedz′wiecki, “Recursive functional series modeling approach to identification of time-varying plants – More bad news than good?,” *IEEE Trans. Automat. Contr.*, vol. 35, pp. 610–616, May 1990.

[17] M. S. Ahmad, O. Kukrer and A. Hocanin, "An Efficient Recursive Inverse Adaptive Filtering Algorithm for Channel Equalization", *IEEE European Wireless Conference*, April, 2010,

[18] G. D. Fomey, Jr., “Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interferences,” *IEEE Trans. Inform. Theory*, vol IT-18, pp. 363-378, May 1972.

[19] D. Gorge, R. Brwen, and J. Storey, "An adpative decision feedback equalizer," *IEEE Trans. Comm.*, vol.19, pp.28 I-293, June 1971.

[20] C. A. Belfiori and J. H. Park Jr., “Decision-feedback equalization*,” Proc. IEEE*, vol. 67, pp. 1143-1156, Aug. 1979.

[21] J. G. Proakis, “Adaptive equalization for TDMA digital mobile radio,” *IEEE Trans. Veh. Technol.*, vol. 40, pp. 333–341, May 1991.

[22] J. M. Cioffi and T. Kailath, “Fast recursive least squares transversal filters for adaptive filtering,” *IEEE Trans. Acoust., Speech Signal Processing*, vol. ASSP–32, pp. 304–337, Apr. 1984.

[23] G. J. Bierman, “Factorization methods for discrete sequential estimation*,” in* *Mathematics in Science and Engineering*. New York: Academic, 1977, vol. 128.

[24] J. R. Treichler, I. Fijalkow, and C. R. Johnson, Jr., “Fractionally spaced equalizers,” *IEEE Signal Process. Mag.*, vol. 13, no. 3, pp. 65–81, May 1996.

[25] M. S. Ahmad, O. Kukrer and A. Hocanin, "A 2-D recursive inverse adaptive algorithm ", *SIViP*, March, 2010,

[26] M. Niedz´wiecki and P. Kaczmarek, “Estimation and tracking of complex-valued quasi-periodically varying systems,” *Automatica*, vol. 41,pp. 1503–1516, 2005.

[27] M. Niedz´wiecki and P. Kaczmarek, “Identification of quasi-periodically varying systems using the combined nonparametric/parametric approach*,” IEEE Trans. Signal Process.*, vol. 53, no. 12, pp. 4588–4598, Dec. 2005.

[28] T. Kłaput and M. Niedz´wiecki, “A novel approach to estimation of Doppler frequencies of a time-varying communication channel,” in *Proc. Amer. Contr. Conf.*, Anchorage, AK, 2002, pp. 3213–3218.

[29] M. Cioffi and T. Kailath, “ Fast, recursive-least-squares transversal filters for adaptive filtering,” *IEEE Trans. Acoust., Speech, Signal Processing*. Vol. ASSP-32, pp. 304-337. Apr. 1984.

[30] M. Niedz´wiecki and T. Kłaput, “Do WLS adaptive filters provide better tracking performance than LMS filters?,” *Bull. Polish Acad. Sci., Tech. Sci. Series*, vol. 49, pp. 517–525, 1999.

[31] L. Guo and L. Ljung, “Performance analysis of general tracking algorithms,” *IEEE Trans. Automat. Contr.* , vol. 40, pp. 1388–1402, Aug.1995.

[32] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. NewYork: Wiley, 1996.

# APPENDICES

## Appendix A: Simple Propagation Scenario

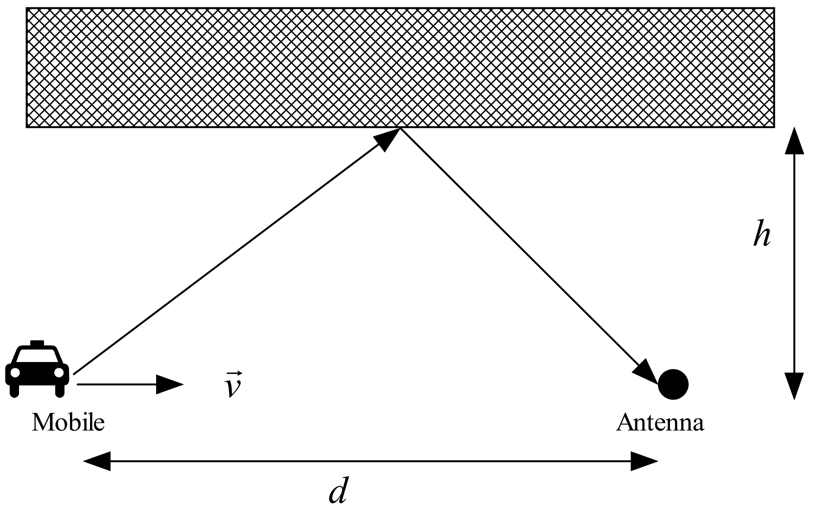

Figure A.1: Simple Propagation Scenario

Depending on a simple propagation delay scenario considered on Figure (A.1), we will study the variation of $\tau_l(t)$. Figure (A.1) shows a mobile body with constant velocity $v$, and an infinite obstacle.

The propagation delay on the reflected path is

$$\tau_r(d) = 2\frac{\sqrt{(d/2)^2 + h^2}}{c} \tag{A.1}$$

where $c$ is the light speed, $d$ is the mobile-receiving antenna distance, and $h$ is the antenna-obstacle distance.

The derivative of $\tau_l(t)$ with respect to $d$ is giving by

$$\frac{\partial \tau_r}{\partial d} = \frac{1}{c\sqrt{1/4 + (h/d)^2}} \tag{A.2}$$

The maxima of $\tau_r(t)$ happens when $d$ goes infinity, so the most important slope for $\tau_r(t)$ is given from Eq. (A.1) as $\tau_r(t) \approx d/c$.

Now, with $d = d_0 - vt$, where $d_0$ is the distance at $t=0$, by dividing over $c$ we have

$$\tau_r(t) = \frac{d_0}{c} - \frac{v}{c}t \tag{A.3}$$

From the former equation we see that $\tau_r(t)$ is a linearly time-variant parameter. With the current mobile speeds ($v \approx 300$ km/h) the slope of $\tau_r(t)$ is about $3 \times 10^{-7}$. For mobile communication with bit rate 20 kbps, with QAM modulation, the variation of $\tau_r(t)$ during the symbol period is about $75 \times 10^{-4}$ ns, which is very small. So, for a large number of symbols, $\tau_r(t)$ can be considering constant.

# Appendix B: Matrix Inversion Lemma

The matrix inversion Lemma is giving by

$$\left(A - BD^{-1}C\right)^{-1} = A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1} \tag{B.1}$$

# Appendix C: Discrete Cosine Transform

We explore herein the Discrete Cosine Transform DCT as an example of the orthogonal transform. The DCT of a data sequence $\{x(n), x(n-1), \ldots, x(n-N+1)\}$ is defined as [5],

$$x_{DCT,k}(n) = \sum_{l=0}^{N-1} c_{kl} x(n-l), \qquad k = 0,1,\ldots N-1 \tag{C.1}$$

where

$$c_{kl}(n) = \begin{cases} \dfrac{1}{N}, & k = 0 \quad \text{and} \quad l = 0,1,\ldots N-1, \\ \sqrt{\dfrac{2}{N}} \cos\dfrac{\pi(2l+1)k}{2N}, & k,l = 0,1,\ldots N-1 \end{cases} \tag{C.2}$$

are the DCT coefficients. Eq. (C.1) can be written as a linear system as follows

$$\mathbf{x}_{DCT}(n) = \mathbf{T}_{DCT}\mathbf{x}(n) \tag{C.3}$$

where $\mathbf{T}_{DCT}$ is the $N \times N$ DCT matrix.

# Appendix D: Kronecker Product

The Kronecker product of $\boldsymbol{X}\otimes\boldsymbol{Y}\left[im\times jn\right]$ of two matrices $\boldsymbol{X}\left[i\times j\right]$and $\boldsymbol{Y}\left[m\times n\right]$ is defined as

$$\boldsymbol{X}\otimes\boldsymbol{Y}\triangleq\begin{bmatrix} x_{11}\boldsymbol{Y} & \cdots & x_{1j}\boldsymbol{Y} \\ \vdots & \ddots & \vdots \\ x_{i1}\boldsymbol{Y} & \cdots & x_{ij}\boldsymbol{Y} \end{bmatrix} \tag{D.1}$$

And the Kronecker product identity given by

$$\left(\boldsymbol{X}\otimes\boldsymbol{Y}\right)\left(\boldsymbol{P}\otimes\boldsymbol{Q}\right)=\boldsymbol{XP}\otimes\boldsymbol{YQ} \tag{D.2}$$