CHANNEL ESTIMATION TECHNIQUES FOR OFDM SYSTEMS IN DISPERSIVE TIME-VARYING CHANNELS

by

XUEGUI SONG

B.Eng., Northwestern Polytechnical University, P. R. China, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The College of Graduate Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
(OKANAGAN)

May 2009

Abstract

Coherent modulation is more effective than differential modulation for orthogonal frequency division multiplexing (OFDM) systems requiring high data rate and spectral efficiency. Channel estimation is therefore an integral part of the receiver design. In this thesis, two iterative maximum likelihood based channel estimation algorithms are proposed for an OFDM system in dispersive time-varying channels. A multipath channel model is proposed for OFDM uplink transmission in macrocellular systems. The multipath fading channel is modeled such that the channel state can be determined by estimating the unknown channel parameters. A second-order Taylor series expansion is adopted to simplify the channel estimation problem. Based on the system model, an iterative maximum likelihood based algorithm is first proposed to estimate the discrete-time channel parameters. The mean square error performance of the proposed algorithm is analyzed using a small perturbation technique. Based on a convergence rate analysis, an improved iterative maximum likelihood based channel estimation algorithm is presented using a successive overrelaxation method. Numerical experiments are performed to confirm the theoretical analyses and show the improvement in convergence rate of the improved algorithm.

Table of Contents

Appendices

List of Tables

List of Figures

List of Acronyms

Acronyms	Definitions
3GPP	Third Generation Partnership Project
BER	Bit-Error Rate
BPSK	Binary Phase-Shift Keying
CIR	Channel Impulse Response
CRB	Cramer-Rao Bound
DAB	Digital Audio Broadcasting
DFT	Discrete Fourier Transform
DVB	Digital Video Broadcasting
FFT	Fast Fourier Transform
ICI	Inter-Carrier Interference
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
ISI	Intersymbol Interference
LOS	Line-of-Sight
LS	Least Square
LTE	Long Term Evolution
ML	Maximum Likelihood
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
OFDM	Orthogonal Frequency Division Multiplexing
P/S	Parallel-to-Serial
PAPR	Peak-to-Average Power Ratio

RF	Radio Frequency
S/P	Serial-to-Parallel
SC-FDMA	Single-Carrier Frequency Division Multiple Access
SNR	Signal-to-Noise Ratio
SOR	Successive Overrelaxation
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Access Network

Acknowledgments

I am deeply grateful to my supervisor Dr. Julian Cheng for his enthusiasm, guidance, advice, encouragement, and support. I will continue to be influenced by his rigorous scholarship, clarity in thinking, and professional integrity.

I would like to thank Dr. Zheng Du from Huawei Technology Co. and Dr. Norman C. Beaulieu from University of Alberta, Edmonton for their feedbacks, constructive comments, and valuable suggestions on my research work.

I would like to express my thanks to Dr. Paramjit Gill for his willingness to serve as my external examiner. I would also like to thank Dr. Richard Klukas and Dr. Jonathan Holzman for serving on the committee. I really appreciate their valuable time and constructive comments on my thesis.

I owe many people for their generosity and support during my two-year study at the University of British Columbia. I would like to thank my dear colleagues Chiun-Shen Liao, Junfeng Zhao, James Jianchen Sun, Mingbo Niu, Nick Kuan-Hsiang Huang, Ning Wang, Xian Jin, and Yeyuan Xiao for sharing their academic experiences and constructive viewpoints generously with me during our discussions. I would also like to thank my dear friends Eric Jinglong Ma, Junning Li, Pan Luo, Yi Zhang, and Yu Hui for sharing in my excitement and encouraging me when I was in frustration during this journey. I would also like to thank all my friends in Vancouver, Beijing, Chengdu, and Xi'an.

Finally, I would like to thank my parents for their patience, understanding and support over all these years. All my achievements would not have been possible without their constant encouragement and support.

Chapter 1

Introduction

1.1 Background and Motivation

The earliest prototype of wireless communications can be traced back to the pre-industrial age. In those years, information was transmitted over line-of-sight (LOS) distances using smoke signals, torch signaling, flashing mirrors, signal flares, or semaphore flags. However, true wireless communications, where information is transmitted in terms of electromagnetic waves through complex physical mediums, started from the first experiments with radio communication by M. G. Marconi in the 1890s. Since then, wireless communication systems have been developing and evolving with a rapid pace. In the intervening hundred years, many types of wireless communication systems have flourished, and often later disappeared. By far the most successful wireless communication system has been the cellular system. It is reported that the number of mobile phone subscribers is on the order of 4 billion currently and will rise to 5.6 billion in 2013 worldwide. Undoubtedly, cellular phones have radically changed the life of people and become a critical business tool in most countries.

In the last two decades, with the explosive growth of wireless communications, wireless services have migrated from the conventional voice-centric services to data-centric services. Therefore, one main target for modern wireless communications is to provide the possibility for high end-user data rates. One straightforward way to meet this requirement is to increase the transmission bandwidth because it is well known that the achievable end-user data rates are limited by transmission bandwidth and transmitter power. However, the transmitted signal in a wireless communication system with wider transmission bandwidth can incur increased corruption due to time dispersion (or equivalently frequency selectivity) of the radio channel. To counteract such signal

corruption, researchers have proposed techniques such as receiver-side equalization, multicarrier transmission and specific single-carrier transmission. Orthogonal frequency division multiplexing (OFDM) is a typical multicarrier transmission scheme which can increase the overall transmission bandwidth without suffering from increased signal corruption due to radio channel frequency selectivity. In OFDM systems, data is transmitted in parallel by modulating a number of closely-spaced orthogonal subcarriers, thereby converting a frequency-selective channel into multiple flat fading subchannels. Moreover, intersymbol interference (ISI) can be eliminated by inserting a guard interval between two consecutive OFDM symbols. With these attractive properties OFDM has been adopted by wireless standards such as digital audio broadcasting (DAB), digital video broadcasting (DVB), wireless local area network (WLAN), and wireless metropolitan access network (WMAN) [1–3].

There are several key challenging problems associated with OFDM systems: performance optimization, time and frequency synchronization, channel estimation, and peak-to-average power ratio (PAPR) reduction [4]. In this thesis, we will focus on the channel estimation problem and study channel estimation techniques for an OFDM system operating in dispersive time-varying channels.

1.2 Literature Review

Channel estimation is a challenging problem in wireless systems because mobile radio channels are highly dynamic. To avoid channel estimation, one can adopt a differential modulation technique instead of coherent modulation. However, such a system typically results in lower data rates and can incur a 3-4 dB penalty in signal-to-noise ratio (SNR) [5]. For OFDM systems which aim to provide high data rate and spectral efficiency, coherent modulation is more effective; hence, channel estimation is often required as an integral part of the receiver design.

Channel estimation for OFDM systems in slow fading channels has been widely studied [6–8]. However, those channel estimation techniques were developed under the assumption that the channel is constant over one OFDM symbol, an assumption that may not hold in some mobile applications. Channel estimation for OFDM systems in fast fading channels has been studied in

the following work. In [9], Li *et al.* presented a minimum mean square error (MMSE) estimator by exploiting both time-domain and frequency-domain correlations of the frequency response of rapid dispersive fading channels. In a related work, Moon and Choi introduced a channel estimation algorithm by adopting a Gaussian interpolation filter or a cubic spline interpolation filter [10]. However, algorithms in both [9] and [10] require knowledge of channel statistics, which may not be available. To make the estimation algorithm independent of the channel statistics, Li discussed in [11] a robust implementation of the MMSE pilot-symbol-aided estimator, which does not depend on channel statistics. In [12], Zhao and Huang proposed a method employing low-pass filtering in a transform domain to estimate the channel transfer function for the pilot subcarriers. Then a high-resolution interpolation is adopted to obtain the channel transfer function for non-pilot subcarriers. In [13], Chang and Su discussed a pilot-aided channel estimation method for OFDM systems operating in Rayleigh fading channels. The channel responses at pilot subcarriers are estimated using a least square (LS) estimator and then interpolated to non-pilot subcarriers using a 2-D regression surface function. This estimator also does not require channel statistics. Recently, Merli and Vitetta proposed a maximum likelihood (ML) based algorithm for joint carrier frequency offset and channel impulse response (CIR) estimation for OFDM systems in [14]. The authors adopted a second-order Taylor series to simplify the estimation problem and derived an approximate closed-form solution to the estimation problem. However, it was derived under the assumption that the channel does not change over one OFDM symbol. In [14], the frequency offset is assumed to be constant for all multipaths implying that the Doppler frequency shift is neglected for individual multipath.

In this thesis, we study the channel estimation problem for OFDM systems in dispersive time-varying fading channels. Two ML-based channel estimation algorithms which do not require channel statistics are proposed. Unlike the existing solutions in the literature, our channel estimation algorithms are based on a unique channel model which is particularly appropriate for OFDM uplinks in a macrocellular system. In this channel model, the channel state can be determined by estimating a number of unknown channel parameters which do not change over several OFDM symbols. This property of our developed channel model allows us to estimate the channel pa-

rameters once and then employ them to determine the channel state for the next several OFDM symbols.

1.3 Thesis Outline and Contributions

This thesis consists of five chapters. Chapter 1 presents background knowledge of wireless communication developments and technologies. In modern wireless communication, a high end-user data rate is one main objective of system design and therefore wider transmission bandwidth is desired. However, an increased transmission bandwidth will increase the frequency selectivity of a radio channel and thus cause severe corruption to the transmitted signal. This problem can be solved by using multicarrier transmission techniques such as OFDM.

Chapter 2 provides detailed technical background for the entire thesis. Firstly, fading characteristics of wireless propagation environments are presented and classified into large-scale fading and small-scale fading from the view of propagation terrain. Secondly, the basic concept of OFDM and its implementation are explained. Finally, we list the advantages and drawbacks of OFDM compared with a conventional single-carrier transmission.

In Chapter 3, a channel estimation problem is formulated based on a discrete-time CIR model. This channel model is particularly appropriate for OFDM uplinks in a macrocellular system. The channel model developed here allows the CIR to vary within one OFDM symbol. To facilitate the channel estimation problem we adopt a truncated Taylor series expansion to approximate the inter-carrier interference (ICI) caused by Doppler frequency shifts. We find that a second-order Taylor series expansion is sufficient for our estimation problem. Then an iterative ML-based algorithm is proposed to estimate the discrete-time channel parameters. Our proposed channel estimation algorithm is particularly useful for an OFDM system which has already compensated the frequency offset due to local oscillator mismatch. The mean square error (MSE) performance of the proposed algorithm in large SNR regions is analyzed using a small perturbation technique, and then demonstrated by simulated results.

In Chapter 4, the convergence performance of the proposed algorithm is analyzed. Based on the analysis, an improved fast converging iterative channel estimation algorithm is presented

using a successive overrelaxation (SOR) method to accelerate the proposed algorithm in Chapter 3. Simulated results are also presented to demonstrate the MSE and convergence performance for the improved algorithm.

Chapter 5 summarizes the whole thesis and lists our contributions in this work. In addition, some future work related to our current research is suggested.

Chapter 2

Wireless Channels and OFDM Technique

In this chapter, we will present some background knowledge concerning wireless channels and the OFDM technique. We first address the fading characteristics of wireless propagation environments, then introduce the basic concept and fast Fourier transform (FFT) implementation of OFDM. Finally, advantages and drawbacks of the OFDM technique are compared with conventional single-carrier modulation.

2.1 Wireless Channels

In a wireless communication system, the transmitted signal typically undergoes attenuation and distortion over the transmission path. The overall effect on the transmitted signal caused by the transmission path is one main source of system performance degradation in any wireless system. To address and compensate for the attenuation and distortion caused by the transmission path, researchers have studied wireless channels extensively and proposed different channel models [5], [15–18]. These effects, which are mainly due to path loss, shadowing, scattering and reflecting effects caused by unpredictable objects between the transmitter and receiver, can primarily be categorized into large-scale fading and small-scale fading (see Fig. 2.1). In this section, we will briefly describe both types of fading.

2.1.1 Large-Scale Fading

Large-scale fading, which is caused by path loss of the signal, can be characterized as a function of transmitted distance and shadowing effects of large obstacles such as buildings and hills. This phenomenon happens when the mobile moves over a large distance (of the order of the cell size),

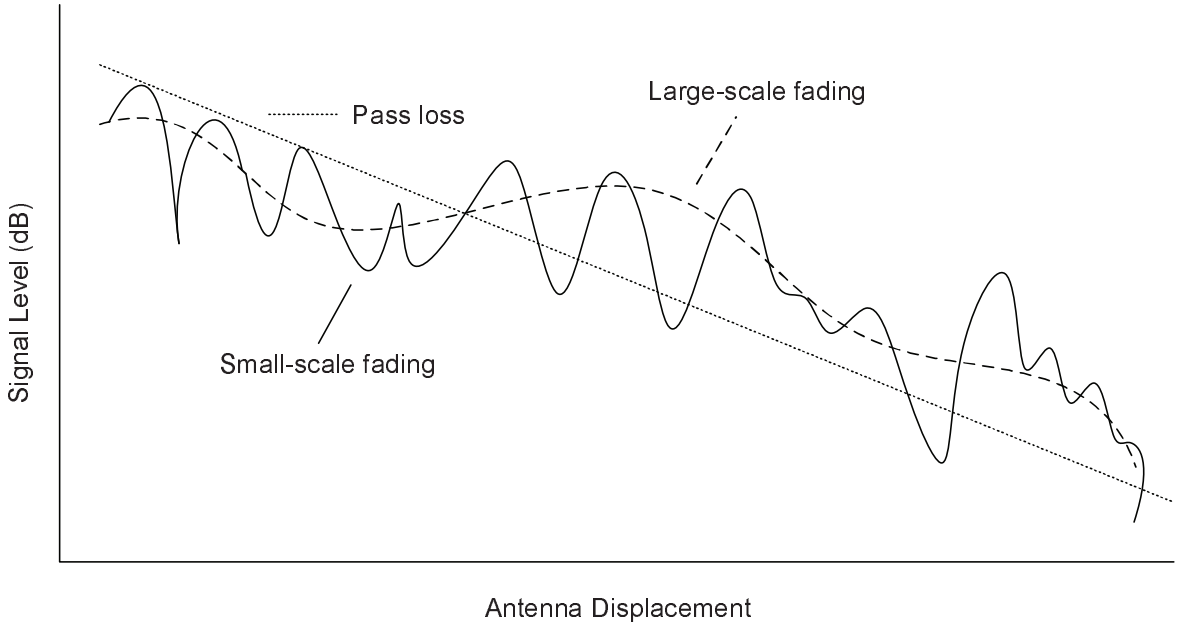

Figure 2.1: Signal power fluctuation versus range in wireless channels.

and is typically frequency independent [18]. In an ideal LOS channel with no obstructions between the transmitter and the receiver, the received signal power P_r is given by

$$P_r = P_t \left(\frac{\lambda}{4\pi d} \right)^2 G_t G_r \tag{2.1}$$

where P_t is the transmitted signal power, λ is the wavelength, d is the distance from the transmitter to the receiver, and G_t, G_r are respectively the power gains of the transmit and receive antennas. The power attenuation $P_L = P_r/P_t$ is also referred to as free space path loss and it is obvious from (2.1) that the received signal power P_r is inversely proportional to the square of the distance d between the transmit and receive antennas. However, in real transmission environments, the received signal power P_r does not obey this free space path loss model and it varies randomly due to the terrain. Usually a ray tracing method can be used to trace the signal propagation through a wireless channel. Unfortunately, the free space model and ray tracing method cannot model complex propagation environments accurately. Based on empirical measurements, some empirical models for path loss in typical wireless environments have been developed to predict the average received signal power as the transmitted distance d varies, i.e., the Okumura model, the Hata

model, the European Cooperative for Science and Technology (COST) model, and the piecewise linear (multi-slope) model [17], [19]. In addition to path loss, the transmitted signal is also subject to shadowing, which is caused by changes in reflecting surfaces and scattering objects along the transmission path. The shadowing causes random attenuation to the transmitted signal. Typically, the log-normal shadowing model, which has been confirmed empirically to accurately model the variation in received power, is used to characterize this random attenuation.

2.1.2 Small-Scale Fading

Small-scale fading is due to the constructive and destructive addition of different multipath components introduced by the channel between the transmitter and receiver. Therefore, it is also referred to as multipath fading. This phenomenon usually occurs over a distance of several signal wavelengths and is frequency dependent. Since the transmitted signal over a multipath fading channel experiences randomness we must characterize multipath fading channels statistically. Frequency selectivity and the time-varying nature, which depend on the relative relation between parameters of the transmitted signal (i.e., signal bandwidth and symbol duration) and parameters of multipath fading channels (i.e., delay spread and Doppler spread), are two important characteristics of multipath fading channels. In the following, we will briefly describe these two characteristics of multipath fading channels.

Depending on the relative relation between transmitted signal bandwidth and delay spread (or equivalently coherence bandwidth B_C), multipath fading channels can be categorized into frequency-nonselective (flat) fading channels and frequency-selective fading channels. The parameter coherence bandwidth is the reciprocal of the delay spread which is defined as the span of the delays of duplicates of the transmitted signal arriving at the receiver via different paths. When the transmitted signal bandwidth is small compared with the coherence bandwidth B_C, the channel is called frequency-nonselective or flat fading channel. For a flat fading channel, the spectral components of the transmitted signal are affected in a similar manner so that the multipath components are not resolvable. Otherwise, if the transmitted signal bandwidth is large compared with the coherence bandwidth B_C, the channel is said to be frequency-selective. For a frequency-selective fad-

ing channel, the spectral components of the transmitted signal are affected by different amplitude gains and phase shifts. In a frequency-selective fading channel, different multipath components with delay differences significantly exceeding the inverse of the transmitted signal bandwidth are resolvable. Typically, such a frequency-selective fading channel can be modeled as a tapped delay line filter with time-variant tap coefficients.

In a similar way, the multipath fading channel can be categorized as slow fading or fast fading based on the relative relation between symbol duration and Doppler spread (or equivalently coherence time T_C). The parameter coherence time, which measures the period of time over which the channel effect on the transmitted signal does not change, is defined as the reciprocal of the Doppler spread. The fading channel is said to be slow fading if the symbol duration is small compared with the channel coherence time T_C; otherwise, it is considered to be fast fading. In a slow fading channel, the transmitted signal is affected by the same amplitude gain and phase shift over at least one symbol duration, while the amplitude gain and phase shift vary within one symbol duration in a fast fading channel.

According to the discussion above, we have four types of multipath channel, i.e., slow flat fading channel, slow frequency-selective fading channel, fast flat fading channel, and fast frequency-selective fading channel. In this thesis, we focus on the channel state estimation of a fast frequency-selective fading channel for an OFDM system. The channel is characterized using the CIR as [17]

$$h(\tau,t) = \sum_{l}^{L(t)} \gamma_l(t)\delta(\tau - \tau_l(t)) \tag{2.2}$$

where $L(t)$ is the number of resolvable multipaths, $\tau_l(t)$ is the delay of the lth multipath, and $\gamma_l(t)$ is the corresponding complex amplitude. We will look into the details of this channel model in the next chapter.

2.2 OFDM Technique

OFDM has become a promising multicarrier modulation technique and has received considerable interest in the past decade. In OFDM systems, data is transmitted in parallel by modulating a

number of closely-spaced orthogonal subcarriers, thereby converting a frequency-selective channel into multiple flat fading subchannels [4], [20]. Moreover, ISI can be eliminated by inserting a guard interval between two consecutive OFDM symbols. With these attractive properties OFDM has been adopted by wireless standards such as DAB, DVB, WLAN, and WMAN [1–3]. In this section, we will present the basic concept and FFT implementation of OFDM and list the advantages and drawbacks of the OFDM technique compared with conventional single-carrier modulation.

2.2.1 Basic Concept

OFDM is a special case of multicarrier transmission which uses parallel data transmission and frequency division multiplexing. The earliest development of this concept can be traced back to the 1950s [21] and the idea was published in the mid-1960s [22], [23]. In an OFDM system, the available bandwidth is divided into a collection of narrow sub-bands, and the data is transmitted in parallel by modulating these closely-spaced orthogonal subcarriers. Let $X[k]$ denote the complex symbols to be transmitted by an OFDM system, such that the OFDM (modulated) signal can be expressed as

$$s(t) = \sum_{k=0}^{N-1} X[k] e^{j2\pi f_k t}, \qquad 0 \le t \le T_s \tag{2.3}$$

where $f_k = f_0 + k\Delta f$, $j^2 = -1$, and N denotes the number of subcarriers in the system. Parameters T_s and Δf are called symbol duration and subcarrier spacing of the OFDM system, respectively. These two parameters must satisfy the orthogonality condition ($T_s \Delta f = 1$) to guarantee that the OFDM signal can be demodulated properly by the receiver.

To demonstrate this orthogonal property, we first let

$$\varphi_k(t) = \begin{cases} e^{j2\pi f_k t}, & 0 \le t \le T_s \\ 0, & \text{otherwise}, \end{cases} \qquad k = 0, 1, \cdots, N-1 \tag{2.4}$$

and rewrite (2.3) as

$$s(t) = \sum_{k=0}^{N-1} X[k] \varphi_k(t). \tag{2.5}$$

Using the definition of $\varphi_k(t)$, one can show

$$\begin{aligned}\frac{1}{T_s}\int_0^{T_s}\varphi_k(t)\varphi_l^*(t)dt &= \frac{1}{T_s}\int_0^{T_s}e^{j2\pi(f_k-f_l)t}dt \\ &= \frac{1}{T_s}\int_0^{T_s}e^{j2\pi(k-l)\Delta ft}dt \\ &= \delta[k-l] \end{aligned} \tag{2.6}$$

where $\delta[k-l]$ is the delta function defined as

$$\delta[n] = \begin{cases} 1, & n=0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

From (2.6) one can see that $\{\varphi_k(t)\}_{k=0}^{N-1}$ is a set of orthogonal functions. Using this orthogonal property, the received OFDM signal $r(t)$ (in an ideal case $r(t)=s(t)$) can be demodulated at the receiver by

$$\begin{aligned}\frac{1}{T_s}\int_0^{T_s}r(t)e^{-j2\pi f_k t}dt &= \frac{1}{T_s}\int_0^{T_s}s(t)e^{-j2\pi f_k t}dt \\ &= \frac{1}{T_s}\int_0^{T_s}\left(\sum_{l=0}^{N-1}X[l]\varphi_l(t)\right)\varphi_k^*(t)dt \\ &= \sum_{l=0}^{N-1}X[l]\delta[l-k] \\ &= X[k]. \end{aligned} \tag{2.8}$$

According to the discussion above, we provide an illustrative description of a basic OFDM modulator in Fig. 2.2 and the basic principle of OFDM demodulation in Fig. 2.3.

2.2.2 OFDM Implementation Using FFT

The relationship between OFDM and the discrete Fourier transform (DFT) was first addressed in [24]. It has been shown that the DFT can be applied to an OFDM system as part of the modulation and demodulation processes. In practice, the DFT can be implemented with the computationally efficient FFT. Here we will briefly discuss the FFT implementation of OFDM.

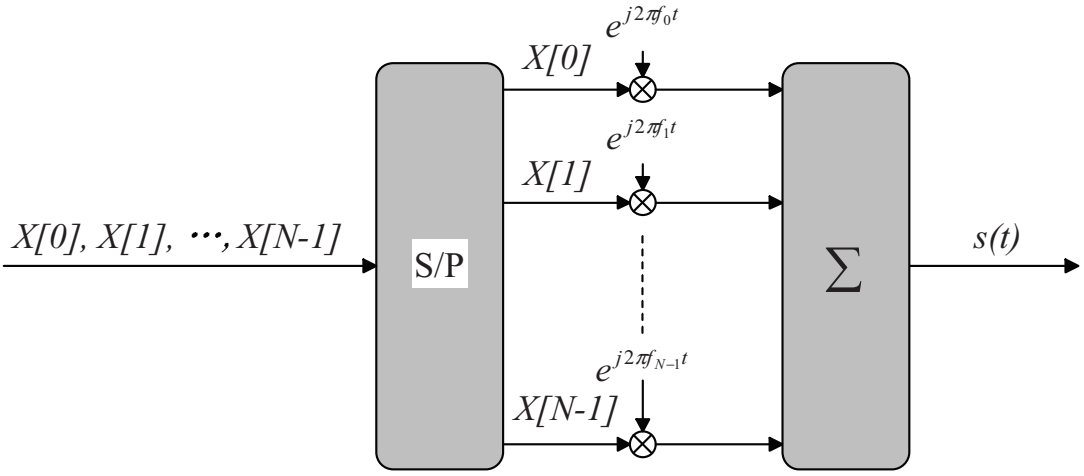

Figure 2.2: Basic principle of OFDM modulation.

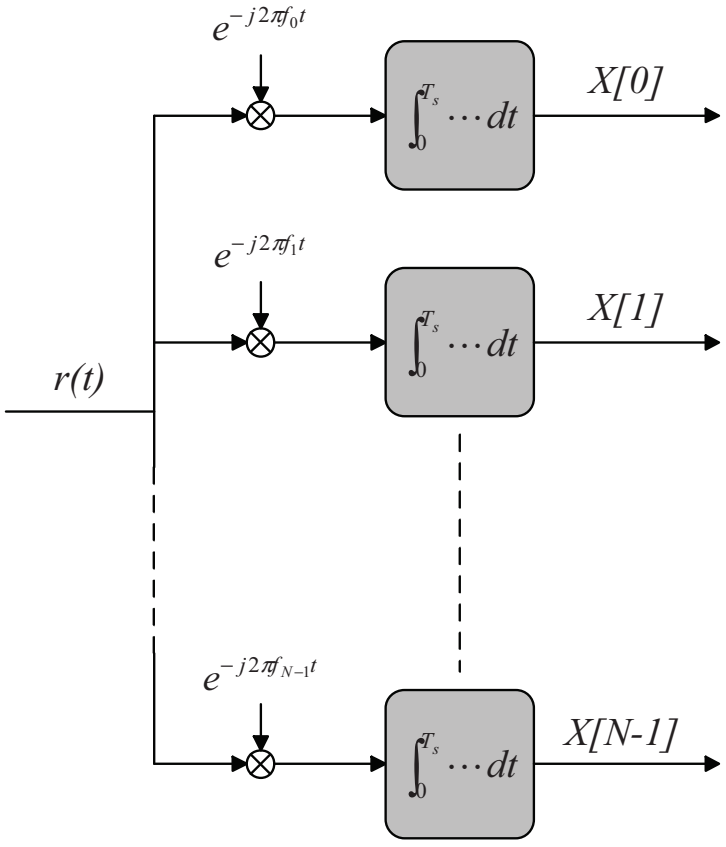

Figure 2.3: Basic principle of OFDM demodulation.

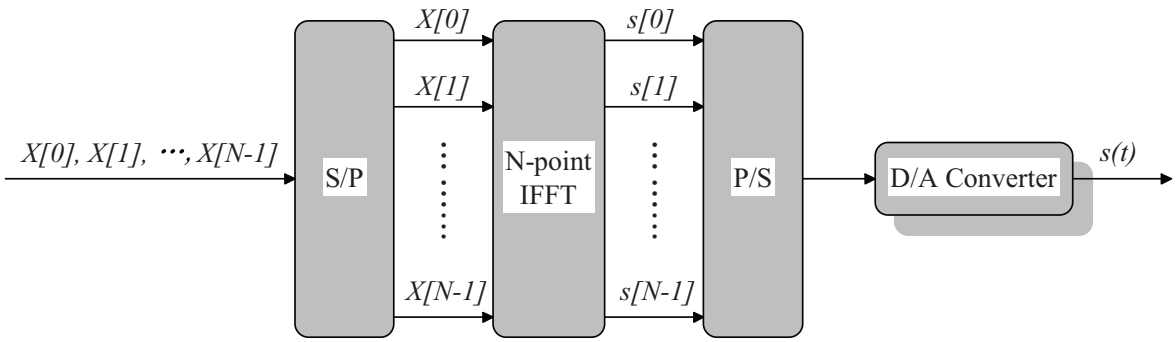

Figure 2.4: OFDM modulation by means of IFFT processing.

From (2.3), we can sample $s(t)$ at an interval of $T_{sa} = T_s/N$. Then the sampled OFDM signal becomes

$$\begin{aligned} s[n] &\stackrel{\triangle}{=} s(t)|_{t=nT_{sa}} \\ &= \sum_{k=0}^{N-1} X[k] e^{\frac{j2\pi f_k n T_s}{N}}, \qquad n = 0,1,\ldots,N-1. \end{aligned} \tag{2.9}$$

Without loss of generality, we can set $f_0 = 0$ so that $f_k T_s = k$ and

$$\begin{aligned} s[n] &= \sum_{k=0}^{N-1} X[k] e^{\frac{j2\pi kn}{N}} \\ &= \text{IDFT}\{X[k]\}, \qquad n = 0,1,\ldots,N-1 \end{aligned} \tag{2.10}$$

where IDFT denotes the inverse discrete Fourier transform. Therefore, as illustrated in Fig. 2.4, OFDM modulation can be implemented by means of IDFT (or the computationally efficient inverse fast Fourier transform (IFFT)) processing followed by digital-to-analog conversion. Similarly, the demodulation at the receiver can be carried out using efficient FFT processing, as illustrated in Fig. 2.5.

2.2.3 Advantages and Drawbacks of OFDM

As a special case of multicarrier transmission scheme, OFDM has the following key advantages over conventional single-carrier transmission [20]:

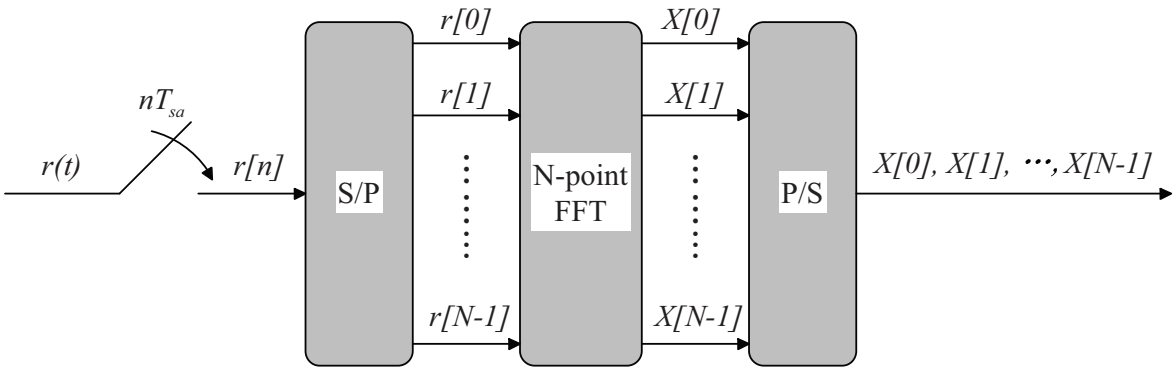

Figure 2.5: OFDM demodulation by means of FFT processing.

- OFDM is an efficient way to deal with multipath fading; for a given channel delay spread, the implementation complexity is much lower than that of a conventional single-carrier system with an equalizer.
- In relatively slow time-varying channels, it is possible to significantly enhance the capacity by adapting the data rate per subcarrier according to the SNR of that particular subcarrier.
- OFDM is robust against narrowband interference, because such interference only affects a small percentage of the subcarriers.
- OFDM makes single-frequency networks possible, which is especially attractive for broadcasting applications.

However, OFDM also has some drawbacks compared with conventional single-carrier modulation [20]:

- OFDM is more sensitive to frequency offset and phase noise. The frequency offset and phase noise will destroy the orthogonality among subcarriers and hence introduces ICI.
- OFDM signals have relatively large PAPR, which tends to reduce the power efficiency of the RF amplifier.
- OFDM needs an adaptive or coded scheme to overcome spectral nulls in the channel.

Chapter 3

ML-based Channel Estimation for OFDM Systems in Dispersive Time-Varying Channels

In this chapter, by taking into account the ICI component, we propose an ML-based channel estimation algorithm (which does not require channel statistics) for OFDM systems in dispersive time-varying fading channels. A channel estimation problem is formulated based on a discrete-time CIR model. This channel model is particularly appropriate for OFDM uplinks in a macro-cellular system. The channel model developed here allows the CIR to vary within one OFDM symbol. To facilitate the channel estimation problem we adopt a truncated Taylor series expansion to approximate the ICI caused by Doppler frequency shifts. We find that a second-order Taylor series expansion is sufficient for our estimation problem. Then an iterative ML-based algorithm is proposed to estimate the discrete-time channel parameters. Using a small perturbation technique, we analyze the performance of the proposed algorithm in large SNR regions. Our computer simulations demonstrate that the proposed algorithm can estimate the desired parameters accurately, and the simulated performances agree with our theoretical analysis.

3.1 System Model

We consider an OFDM system with N subcarriers. For each OFDM symbol, we denote the transmitted symbols as $X[0], X[1], \ldots, X[N-1]$. After the IFFT, the time-domain OFDM signal can be

expressed as [25]

$$s[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{\frac{j2\pi kn}{N}}, \qquad n = 0,1,\ldots,N-1 \tag{3.1}$$

where $j^2 = -1$. Each OFDM symbol is extended with a cyclic prefix and transmitted after appropriate pulse shaping.

Recalling (2.2), we can describe a multipath CIR by

$$h(\tau,t) = \sum_{l=1}^{L} \gamma_l(t)\,\delta(\tau-\tau_l) \tag{3.2}$$

where $\gamma_l(t)$ is the complex amplitude of the lth multipath. The number of resolvable multipaths $L(t)$ and the delay of the lth multipath $\tau_l(t)$ are assumed to be L and τ_l, respectively. According to [6], [9], the CIR can be well approximated by a sampled discrete-time CIR $h[m,n] \triangleq h(mT_{sa}, nT_{sa})$, where T_{sa} is the sampling interval defined as $T_{sa} = T_s/N$ and T_s is the OFDM symbol duration. In this thesis, we assume a time-varying multipath channel and express the discrete-time CIR as [12], [26]

$$h[m,n] = \sum_{l=1}^{L} h_l e^{\frac{j2\pi f_l n}{N}} \delta[m-n_l] \tag{3.3}$$

where h_l is the complex amplitude for the lth multipath, f_l is the corresponding Doppler frequency shift normalized by $1/T_s$, and n_l is the delay in samples for the lth multipath. The detailed derivation of this discrete-time CIR model is given in Appendix A. It should be emphasized that the discrete-time CIR in (3.3) is only valid for OFDM uplink transmission in macrocellular systems. Without loss of generality, we assume $n_1 \le n_2 \le \cdots \le n_L$. It is seen from (3.3) that the CIR can vary within one OFDM symbol duration.

3.2 Formulation of Channel Estimation Problem

Our goal is to estimate the channel parameters h_l, f_l, for $l = 1,2,\ldots,L$, in the discrete-time CIR in (3.3). We assume that the length of the cyclic prefix is longer than the delay spread of the multipath channel and that the receiver accurately removes the cyclic prefix before implementing the FFT.

For the channel model given in (3.3), one can express the output of the FFT as

$$\begin{aligned} Y[k] &= \frac{1}{N}\sum_{l=1}^{L}\sum_{k'=0}^{N-1}\sum_{n=0}^{N-1} X[k']e^{\frac{j2\pi k'(n-n_l)}{N}}h_l e^{\frac{j2\pi f_l n}{N}}e^{-\frac{j2\pi kn}{N}} + w[k] \\ &= \sum_{l=1}^{L}\sum_{k'=0}^{N-1} e^{-\frac{j2\pi k' n_l}{N}}X[k']h_l \frac{1-e^{j2\pi(k'-k+f_l)}}{N\left(1-e^{\frac{j2\pi(k'-k+f_l)}{N}}\right)} + w[k] \\ &= \sum_{l=1}^{L}\sum_{k'=0}^{N-1} e^{-\frac{j2\pi k' n_l}{N}}X[k']h_l R(k',k,f_l) + w[k] \\ &= \alpha_k X[k] + \beta_k + w[k], \qquad k = 0,1,\ldots,N-1 \end{aligned} \tag{3.4a}$$

where

$$\alpha_k = \sum_{l=1}^{L} e^{-\frac{j2\pi k n_l}{N}}h_l R(k,k,f_l) \tag{3.4b}$$

and

$$\beta_k = \sum_{l=1}^{L}\sum_{\substack{k'=0 \\ k'\neq k}}^{N-1} e^{-\frac{j2\pi k' n_l}{N}}X[k']h_l R(k',k,f_l). \tag{3.4c}$$

In (3.4a) $w[k]$ is an additive complex Gaussian noise component with zero mean and variance σ^2, i.e., $CN\left(0,\sigma^2\right)$, and

$$\begin{aligned} R(k',k,f_l) &= \frac{1}{N}\sum_{n=0}^{N-1} e^{\frac{j2\pi n(k'-k+f_l)}{N}} \\ &= \frac{1-e^{j2\pi(k'-k+f_l)}}{N\left(1-e^{\frac{j2\pi(k'-k+f_l)}{N}}\right)} \end{aligned} \tag{3.5}$$

represents the interference of $X[k']$ on the kth subcarrier caused by a normalized Doppler frequency shift f_l. In (3.4b) and (3.4c), α_k and β_k denote the multiplicative distortion at the desired subchannel and the additive ICI term, respectively. In the absence of Doppler frequency shifts ($f_l = 0, l = 1,2,\ldots,L$), we have $R(k',k,0) = \delta[k'-k]$ resulting in the conventional model without ICI among subcarriers. In the special case when $f_l = \varepsilon$, for $l = 1,2,\ldots,L$, our estimation problem

is mathematically equivalent to the joint channel and frequency offset estimation problem that has been considered in [14], [27] and [28].

In practical mobile communication scenarios, f_l is typically less than 0.1. Consider an OFDM system, for example, with $N = 256$ subcarriers and a subcarrier spacing of 7.81 kHz. A mobile terminal moving at a speed of 84.4 km/h will result in a normalized Doppler frequency $f_l = 0.05$ with a carrier frequency $f_c = 5$ GHz. Exploiting the fact that f_l has small values, one can consider a Taylor series expansion for $R(k',k,f_l)$ as

$$R(k',k,f_l) = \begin{cases} 1 + j\pi\frac{N-1}{N}f_l - \pi^2\frac{2N^2-3N+1}{3N^2}f_l^2 - j\pi^3\frac{(N-1)^2}{3N^2}f_l^3 + \cdots, & k' = k \\ -\frac{j2\pi}{(1-\omega)N}f_l + 2\pi^2\frac{(2-N)\omega+N}{(1-\omega)^2N^2}f_l^2 + j4\pi^3\frac{N^2(1-\omega)^2+3\omega(1+\omega-N\omega+N)}{3N^3(1-\omega)^3}f_l^3 + \cdots, & k' \neq k \end{cases} \tag{3.6a}$$

where

$$\omega = \exp\left(\frac{j2\pi(k'-k)}{N}\right). \tag{3.6b}$$

Applying (3.6a) to (3.4a) and collecting the terms with the same power of f_l, one obtains

$$Y[k] = \sum_{l=1}^{L} h_l \left\{A_l[k] + B_l[k]f_l + C_l[k]f_l^2 + D_l[k]f_l^3 + \cdots\right\} + w[k] \tag{3.7}$$

for $k = 0, 1, \ldots, N-1$, or, in vector form,

$$\vec{Y} = \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 + \vec{D}_l f_l^3 + \cdots\right) + \vec{w}. \tag{3.8}$$

The first three leading coefficients $A_l[k]$, $B_l[k]$, $C_l[k]$ in (3.7) can be shown to be

$$A_l[k] = X[k]e^{\frac{-j2\pi kn_l}{N}} \tag{3.9a}$$

and

$$B_l[k] = j\pi\frac{N-1}{N}X[k]e^{\frac{-j2\pi kn_l}{N}} - \frac{j2\pi}{(1-\omega)N}\sum_{\substack{k'\neq k \\ k'=0}}^{N-1} X[k']e^{\frac{-j2\pi k'n_l}{N}} \tag{3.9b}$$

and

$$C_l[k] = -\pi^2 \frac{2N^2-3N+1}{3N^2} X[k] e^{\frac{-j2\pi k n_l}{N}} + 2\pi^2 \frac{(2-N)\omega + N}{(1-\omega)^2 N^2} \sum_{\substack{k'\neq k \\ k'=0}}^{N-1} X[k'] e^{\frac{-j2\pi k' n_l}{N}}. \tag{3.9c}$$

The channel estimation technique in this thesis will be based on (3.8). In (3.8), the coefficient vectors $\vec{A}_l$, $\vec{B}_l$, $\vec{C}_l$, ... can be derived given the transmitted training sequence $X[k]$, $k = 0, 1, \ldots, N-1$. We will assume the receiver has knowledge of the training sequence $X[k]$ and performs parameter estimation of the h_l's and f_l's based on (3.8).

We note that the Taylor expansion in (3.6a) is an approximation for $R(k', k, f_l)$ and the accuracy of the approximation depends on the order of the Taylor expansion. On the other hand, the order of the Taylor expansion also determines the computational complexity of the channel estimation algorithm. In this work we employ a second-order Taylor expansion for $|f_l| < 0.1$.

In order to assess the accuracy of the second-order Taylor series expansion, we compare the MSE of the received OFDM symbol using several approximation orders. Here the MSE is defined as the mean squared difference between the exact value of the signal at the receiver ($Y[k]$) and its approximation ($\widehat{Y}[k]$)

$$\text{MSE} \triangleq E\left\{ \left| Y[k] - \widehat{Y}[k] \right|^2 \right\} \tag{3.10}$$

where $E\{x\}$ is the ensemble average of a random sequence x. $Y[k]$ and $\widehat{Y}[k]$ are calculated according to (3.4a) and (3.7) assuming $w[k] = 0$, respectively. From Fig. 3.1 one observes that the MSE increases with the normalized Doppler frequency f_l as expected. Fig. 3.1 also shows that the first-order approximation may be inadequate. However, the difference between the second-order approximation and third-order approximation is negligible. Furthermore, with a typical value of f_l, e.g. $f_l = 0.05$, the MSE of the second-order approximation is about -50 dB and, thus, can be ignored in our estimation problem. Therefore, we can conclude that a second-order approximation

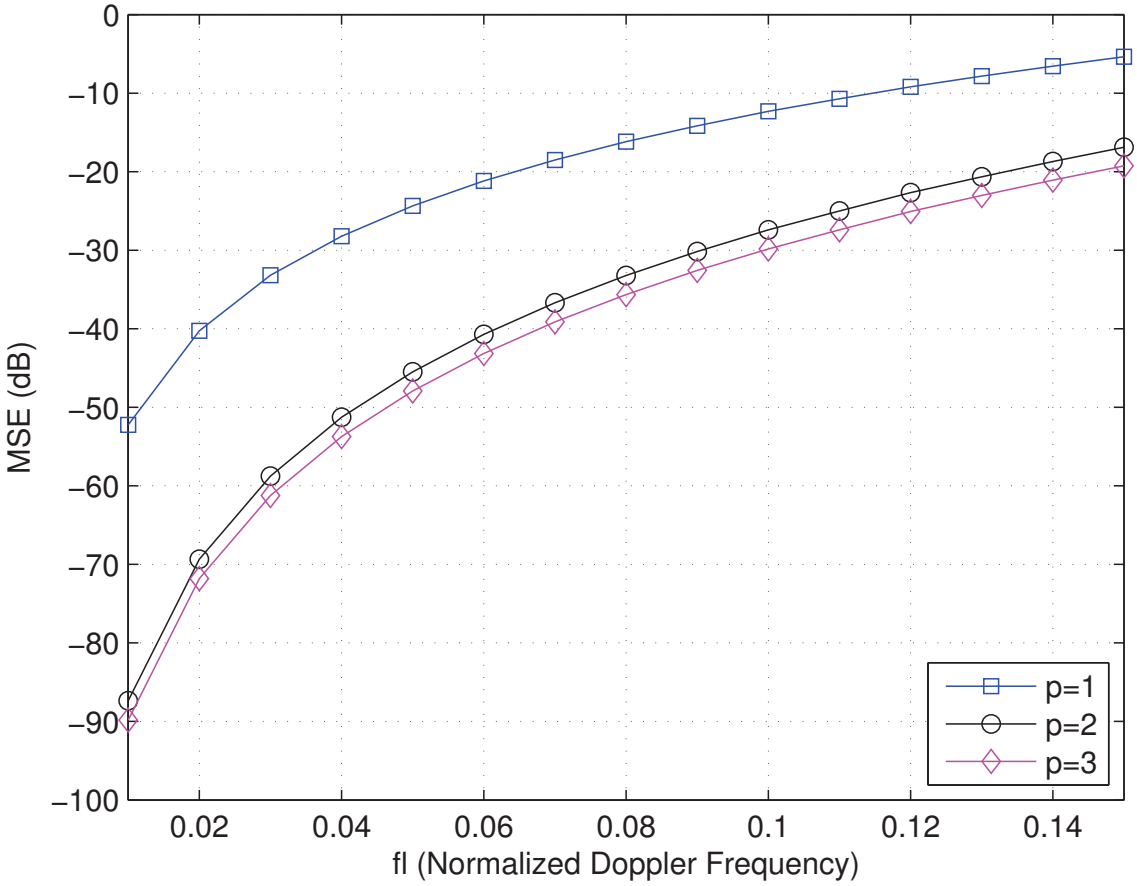

Figure 3.1: An MSE comparison for three different approximation orders.

is sufficient for our problem and modify (3.8) to be

$$\vec{Y} \approx \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) + \vec{w}. \tag{3.11}$$

The likelihood function of the received symbol is then given by

$$\begin{aligned} f\left(\vec{Y} \left| \{h_l, f_l\}_{l=1}^{L} \right. \right) = & \frac{1}{\pi^N \sigma^N} \exp \left\{ - \left[\vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) \right]^H \right. \\ & \left. \times \left[\vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) \right] / \sigma^2 \right\} \end{aligned} \tag{3.12}$$

where $(\cdot)^H$ denotes the Hermitian transpose of the argument matrix. One can therefore express the ML estimation for the parameters h_l's and f_l's as

$$\left\{\hat{h}_l, \hat{f}_l\right\}_{l=1}^{L} = \arg \min_{\{h_l, f_l\}_{l=1}^{L}} \left| \vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) \right|^2 \tag{3.13}$$

where $\hat{h}_l$'s and $\hat{f}_l$'s are the solutions to

$$\frac{\partial}{\partial h_{l'}} \ln f \left(\vec{Y} \left| \{h_l, f_l\}_{l=1}^{L} \right. \right) = 0 \tag{3.14}$$

and

$$\frac{\partial}{\partial f_{l'}} \ln f \left(\vec{Y} \left| \{h_l, f_l\}_{l=1}^{L} \right. \right) = 0, \tag{3.15}$$

respectively. From (3.12), we can get

$$\begin{aligned} \ln f \left(\vec{Y} \left| \{h_l, f_l\}_{l=1}^{L} \right. \right) = \ln \frac{1}{\pi^N \sigma^N} - \frac{1}{\sigma^2} & \left[\vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) \right]^H \\ & \times \left[\vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) \right] \end{aligned} \tag{3.16}$$

and show that (3.14) and (3.15) are equivalent to the following equations

$$\left[\vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) \right]^H \left(\vec{A}_{l'} + \vec{B}_{l'} f_{l'} + \vec{C}_{l'} f_{l'}^2 \right) = 0 \tag{3.17}$$

$$\Re \left\{ \left[\vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 \right) \right]^H h_{l'} \left(\vec{B}_{l'} + 2\vec{C}_{l'} f_{l'} \right) \right\} = 0, \tag{3.18}$$

respectively. Here $\Re\{\cdot\}$ denotes the real part of the complex argument. It is worth mentioning that we need to pay attention to the properties of complex derivatives [29] when obtaining (3.17) and (3.18). Furthermore, we also need to note that (3.17) and (3.18) are non-linear in the h_l's and f_l's. A technique to solve these two equations will be presented in the next section.

Table 3.1: Algorithm 1

Initialization: Initialize $h_l^{(0)}$ and $f_l^{(0)}$, for $l = 1, 2, \ldots, L$. Set the iteration counter $i = 0$.

Iterations:

Step 1: For each $l = 1, 2, \ldots, L$, calculate

$$\tilde{h}_l^{(i+1)} = \arg\min_{h_l} L\left(h_l, \tilde{f}_l^{(i)}\right)$$

$$\tilde{f}_l^{(i+1)} = \arg\min_{f_l} L\left(\tilde{h}_l^{(i+1)}, f_l\right)$$

Step 2: If $\max_l \left(\frac{\left|\tilde{h}_l^{(i+1)} - \tilde{h}_l^{(i)}\right|}{\left|\tilde{h}_l^{(i)}\right|} \right) \times 100\% > \delta_h$ or $\max_l \left(\frac{\left|\tilde{f}_l^{(i+1)} - \tilde{f}_l^{(i)}\right|}{\left|\tilde{f}_l^{(i)}\right|} \right) \times 100\% > \delta_f$, let $i = i + 1$ and go to *Step 1*, otherwise output $\tilde{h}_l^{(i+1)}$'s and $\tilde{f}_l^{(i+1)}$'s.

3.3 An Iterative Channel Estimation Algorithm

In this section, we propose an iterative approach to estimate the $\hat{h}_l$'s and $\hat{f}_l$'s based on (3.13). The most straightforward approach is the coordinate-ascent algorithm [30] as shown in Table 3.1, where the cost function $L(h_l, f_l)$ is defined as

$$\begin{aligned} L(h_l, f_l) = \Bigg| \vec{Y} - \sum_{j=1}^{l-1} h_j^{(i+1)} \left(\vec{A}_j + \vec{B}_j f_j^{(i+1)} + \vec{C}_j \left(f_j^{(i+1)} \right)^2 \right) \\ - \sum_{j=l+1}^{L} h_j^{(i)} \left(\vec{A}_j + \vec{B}_j f_j^{(i)} + \vec{C}_j \left(f_j^{(i)} \right)^2 \right) - h_l \left(\vec{A}_l + \vec{B}_l f_l + C_l {f_l}^2 \right) \Bigg|^2 . \end{aligned} \tag{3.19}$$

Note that in Algorithm 1, the first equation in *Step 1* is a linear least-square problem and the second equation in *Step 1* requires finding the roots for a third-order polynomial. Both equations can be solved using well-known numerical methods. As discussed in [30], the coordinate-ascent algorithm is a special case of the space-alternating generalized expectation-maximization algorithm and has the attractive property of increasing the likelihood value at each iteration.

3.4 Performance Analysis of the Channel Estimation Algorithm

In this section, we analyze the MSE performance of the proposed algorithm in large SNR regions assuming the f_l's are small.

From the discussion in Section 3.2 we can see that the ML-based estimation algorithm aims to minimize the cost function in (3.13). Using a small perturbation technique [31], one can obtain from (3.13) a linear equation for the perturbation of the estimates as

$$\underbrace{\begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1L} \\ R_{21} & R_{22} & \cdots & R_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ R_{L1} & R_{L2} & \cdots & R_{LL} \end{pmatrix}}_{R} \underbrace{\begin{pmatrix} \Delta\vec{\theta}_1 \\ \Delta\vec{\theta}_2 \\ \vdots \\ \Delta\vec{\theta}_L \end{pmatrix}}_{\Delta\vec{\theta}} = \underbrace{\begin{pmatrix} \vec{n}_1 \\ \vec{n}_2 \\ \vdots \\ \vec{n}_L \end{pmatrix}}_{\vec{n}} \tag{3.20a}$$

where

$$R_{ij} = \begin{pmatrix} \Re\left\{\vec{A}_j^H \vec{A}_i\right\} & \Im\left\{\vec{A}_j^H \vec{A}_i\right\} & \Re\left\{h_j^* \vec{B}_j^H \vec{A}_i\right\} \\ -\Im\left\{\vec{A}_j^H \vec{A}_i\right\} & \Re\left\{\vec{A}_j^H \vec{A}_i\right\} & -\Im\left\{h_j^* \vec{B}_j^H \vec{A}_i\right\} \\ \Re\left\{h_i \vec{A}_j^H \vec{B}_i\right\} & \Im\left\{h_i \vec{A}_j^H \vec{B}_i\right\} & \Re\{h_i h_j^* \vec{B}_j^H \vec{B}_i\} \end{pmatrix} \tag{3.20b}$$

and

$$\Delta\vec{\theta}_l = \begin{pmatrix} \Re\{\Delta h_l\} \\ \Im\{\Delta h_l\} \\ \Delta f_l \end{pmatrix} \quad \vec{n}_l = \begin{pmatrix} \Re\left\{\vec{w}^H \vec{A}_l\right\} \\ -\Im\left\{\vec{w}^H \vec{A}_l\right\} \\ \Re\left\{h_l \vec{w}^H \vec{B}_l\right\} \end{pmatrix} \tag{3.20c}$$

where $\Re\{\cdot\}$ and $\Im\{\cdot\}$ respectively denote the real part and imaginary part of the argument, $\{\cdot\}^*$ denotes the complex conjugate of the argument, $\Delta h_l = \hat{h}_l - h_l$ and Δf_l[1] $= \hat{f}_l - f_l$ are the estimate perturbations caused by Gaussian noise $\vec{n}_l$. A detailed derivation of (3.20) is given in Appendix B. When the Gaussian noise is absent, we have $\hat{h}_l = h_l$ and $\hat{f}_l = f_l$.

Equation (3.20) indicates that the perturbations $\Delta\vec{\theta}_l$ subject to the Gaussian noise components

[1] Here Δf_l does not denote the subcarrier spacing.

$\vec{n}_l$, $l = 1,2,\ldots,L$, are also Gaussian, and one can express $\Delta\vec{\theta}_l$ as

$$\begin{pmatrix} \Delta\vec{\theta}_1 \\ \Delta\vec{\theta}_2 \\ \vdots \\ \Delta\vec{\theta}_L \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1L} \\ R_{21} & R_{22} & \cdots & R_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ R_{L1} & R_{L2} & \cdots & R_{LL} \end{pmatrix}^{-1} \begin{pmatrix} \vec{n}_1 \\ \vec{n}_2 \\ \vdots \\ \vec{n}_L \end{pmatrix} \tag{3.21}$$

or in a more compact form

$$\Delta\vec{\theta} = R^{-1}\vec{n}. \tag{3.22}$$

To determine the variance of $\Delta\vec{\theta}_l$ subject to Gaussian noise components $\vec{n}_l$, we have

$$\begin{aligned} \vec{n}_i\vec{n}_j^H &= \begin{pmatrix} \Re\left\{\vec{w}^H\vec{A}_i\right\} \\ -\Im\left\{\vec{w}^H\vec{A}_i\right\} \\ \Re\left\{h_i\vec{w}^H\vec{B}_i\right\} \end{pmatrix} \begin{pmatrix} \Re\left\{\vec{w}^H\vec{A}_j\right\} \\ -\Im\left\{\vec{w}^H\vec{A}_j\right\} \\ \Re\left\{h_j\vec{w}^H\vec{B}_j\right\} \end{pmatrix}^H \\ &= \begin{pmatrix} \Re\left\{\vec{w}^H\vec{A}_i\right\}\Re\left\{\vec{w}^H\vec{A}_j\right\} & -\Re\left\{\vec{w}^H\vec{A}_i\right\}\Im\left\{\vec{w}^H\vec{A}_j\right\} & \Re\left\{\vec{w}^H\vec{A}_i\right\}\Re\left\{h_j\vec{w}^H\vec{B}_j\right\} \\ -\Im\left\{\vec{w}^H\vec{A}_i\right\}\Re\left\{\vec{w}^H\vec{A}_j\right\} & \Im\left\{\vec{w}^H\vec{A}_i\right\}\Im\left\{\vec{w}^H\vec{A}_j\right\} & -\Im\left\{\vec{w}^H\vec{A}_i\right\}\Re\left\{h_j\vec{w}^H\vec{B}_j\right\} \\ \Re\left\{h_i\vec{w}^H\vec{B}_i\right\}\Re\left\{\vec{w}^H\vec{A}_j\right\} & -\Re\left\{h_i\vec{w}^H\vec{B}_i\right\}\Im\left\{\vec{w}^H\vec{A}_j\right\} & \Re\left\{h_i\vec{w}^H\vec{B}_i\right\}\Re\left\{h_j\vec{w}^H\vec{B}_j\right\} \end{pmatrix} \end{aligned} \tag{3.23}$$

and

$$\begin{aligned} E\left\{\vec{n}_i\vec{n}_j^H\right\} &= \frac{\sigma^2}{2}E\left\{\begin{pmatrix} \Re\left\{\vec{A}_j^H\vec{A}_i\right\} & \Im\left\{\vec{A}_j^H\vec{A}_i\right\} & \Re\left\{h_j^*\vec{B}_j^H\vec{A}_i\right\} \\ -\Im\left\{\vec{A}_j^H\vec{A}_i\right\} & \Re\left\{\vec{A}_j^H\vec{A}_i\right\} & -\Im\left\{h_j^*\vec{B}_j^H\vec{A}_i\right\} \\ \Re\left\{h_i\vec{A}_j^H\vec{B}_i\right\} & \Im\left\{h_i\vec{A}_j^H\vec{B}_i\right\} & \Re\left\{h_ih_j^*\vec{B}_j^H\vec{B}_i\right\} \end{pmatrix}\right\} \\ &= \frac{\sigma^2}{2}E\left\{R_{ij}\right\} \\ &= \frac{\sigma^2}{2}R_{ij}. \end{aligned} \tag{3.24}$$

In obtaining (3.24), we have used the fact that $\vec{w}$ is a complex Gaussian noise vector whose elements have independent real and imaginary parts with zero mean and variance equal to $\sigma^2/2$.

Therefore, using (3.20a), (3.22) and (3.24), one can show straightforwardly that

$$E\left\{\begin{pmatrix}\vec{n}_1\\ \vdots\\ \vec{n}_L\end{pmatrix}\left(\vec{n}_1^H\cdots\vec{n}_L^H\right)\right\}=\frac{\sigma^2}{2}R \tag{3.25}$$

and

$$E\left\{\begin{pmatrix}\Delta\vec{\theta}_1\\ \vdots\\ \Delta\vec{\theta}_L\end{pmatrix}\left(\Delta\vec{\theta}_1^H\cdots\Delta\vec{\theta}_L^H\right)\right\}=\frac{\sigma^2}{2}R^{-1}. \tag{3.26}$$

Generally speaking, the elements in R_{ij}, $i\neq j$, are non-zero and, thus, the small perturbations for different taps, $\Delta\vec{\theta}_l$, are correlated. When $L<<N$ and a pseudo-random sequence $X[k]$ is used as the training symbols, the values of R_{ij}, $i\neq j$, are generally much smaller than those in R_{ii} and, thus, negligible [25]. For such cases, one can rewrite (3.26) as

$$E\left\{\Delta\vec{\theta}_l^H\Delta\vec{\theta}_l\right\}=\frac{\sigma^2}{2}R_{ll}^{-1},\quad l=1,2,\ldots,L \tag{3.27}$$

where

$$\begin{aligned}R_{ll}&=\begin{pmatrix}\Re\left\{\vec{A}_l^H\vec{A}_l\right\} & \Im\left\{\vec{A}_l^H\vec{A}_l\right\} & \Re\left\{h_l^*\vec{B}_l^H\vec{A}_l\right\}\\ -\Im\left\{\vec{A}_l^H\vec{A}_l\right\} & \Re\left\{\vec{A}_l^H\vec{A}_l\right\} & -\Im\left\{h_l^*\vec{B}_l^H\vec{A}_l\right\}\\ \Re\left\{h_l\vec{A}_l^H\vec{B}_l\right\} & \Im\left\{h_l\vec{A}_l^H\vec{B}_l\right\} & \Re\{h_l^*h_l\vec{B}_l^H\vec{B}_l\}\end{pmatrix}\\ &=\begin{pmatrix}\vec{A}_l^H\vec{A}_l & 0 & \Re\left\{h_l^*\vec{B}_l^H\vec{A}_l\right\}\\ 0 & \vec{A}_l^H\vec{A}_l & -\Im\left\{h_l^*\vec{B}_l^H\vec{A}_l\right\}\\ \Re\left\{h_l\vec{A}_l^H\vec{B}_l\right\} & \Im\left\{h_l\vec{A}_l^H\vec{B}_l\right\} & |h_l|^2\vec{B}_l^H\vec{B}_l\}\end{pmatrix}\end{aligned} \tag{3.28}$$

and R_{ll}^{-1} is shown to be

$$R_{ll}^{-1} = \frac{1}{|h_l|^2 \vec{A}_l^H \vec{A}_l \left(\zeta - \left|\vec{B}_l^H \vec{A}_l\right|^2\right)} \times \begin{pmatrix} \zeta |h_l|^2 - \left(\Im\left\{h_l \vec{A}_l^H \vec{B}_l\right\}\right)^2 & \Re\left\{h_l \vec{A}_l^H \vec{B}_l\right\} \Im\left\{h_l \vec{A}_l^H \vec{B}_l\right\} & -\vec{A}_l^H \vec{A}_l \Re\left\{h_l \vec{A}_l^H \vec{B}_l\right\} \\ \Re\left\{h_l \vec{A}_l^H \vec{B}_l\right\} \Im\left\{h_l \vec{A}_l^H \vec{B}_l\right\} & \zeta |h_l|^2 - \left(\Re\left\{h_l \vec{A}_l^H \vec{B}_l\right\}\right)^2 & -\vec{A}_l^H \vec{A}_l \Im\left\{h_l \vec{A}_l^H \vec{B}_l\right\} \\ -\vec{A}_l^H \vec{A}_l \Re\left\{h_l \vec{A}_l^H \vec{B}_l\right\} & -\vec{A}_l^H \vec{A}_l \Im\left\{h_l \vec{A}_l^H \vec{B}_l\right\} & \left(\vec{A}_l^H \vec{A}_l\right)^2 \end{pmatrix} \tag{3.29}$$

where ζ is defined as $\zeta = \vec{A}_l^H \vec{A}_l \vec{B}_l^H \vec{B}_l$ for the sake of convenience. Equation (3.27) indicates that the perturbations $\Delta\vec{\theta}_l$ for different taps can be treated independently. However, for the same l, the perturbations $\Delta h_l = \Re\{\Delta h_l\} + j\Im\{\Delta h_l\}$ and Δf_l are still correlated because the non-diagonal elements in R_{ll} are not negligible.

It is of interest to obtain some explicit results for the estimation performance. Motivated by [32], we consider the asymptotic behavior with a white training sequence. Under this condition, we have

$$\vec{A}_l^H \vec{A}_l = 1 \tag{3.30a}$$

$$\vec{B}_l^H \vec{A}_l = -\frac{j\pi(N-1)}{N} \tag{3.30b}$$

$$\vec{B}_l^H \vec{B}_l = \frac{2\pi^2(N-1)(2N-1)}{3N^2} \tag{3.30c}$$

where the derivations of (3.30a)-(3.30b) are trivial, and a detailed derivation of (3.30c) can be found in Appendix C. According to the definition of $\Delta\vec{\theta}_l$, we can obtain

$$\begin{aligned} \Delta\vec{\theta}_l \Delta\vec{\theta}_l^H &= \begin{pmatrix} \Re\{\Delta h_l\} \\ \Im\{\Delta h_l\} \\ \Delta f_l \end{pmatrix} \begin{pmatrix} \Re\{\Delta h_l\} \\ \Im\{\Delta h_l\} \\ \Delta f_l \end{pmatrix}^H \\ &= \begin{pmatrix} (\Re\{\Delta h_l\})^2 & \Re\{\Delta h_l\}\Im\{\Delta h_l\} & \Re\{\Delta h_l\}\Delta f_l \\ \Re\{\Delta h_l\}\Im\{\Delta h_l\} & (\Im\{\Delta h_l\})^2 & \Im\{\Delta h_l\}\Delta f_l \\ \Re\{\Delta h_l\}\Delta f_l & \Im\{\Delta h_l\}\Delta f_l & \Delta f_l^2 \end{pmatrix} \end{aligned} \tag{3.31}$$

and

$$\begin{aligned}\Delta h_l^* \Delta h_l &= |\Delta h_l|^2 \\ &= (\Re\{\Delta h_l\})^2 + (\Im\{\Delta h_l\})^2 .\end{aligned} \tag{3.32}$$

Therefore, we can obtain the expression for the estimation performance as

$$\begin{aligned}E\{\Delta h_l^* \Delta h_l\} &= E\left\{(\Re\{\Delta h_l\})^2 + (\Im\{\Delta h_l\})^2\right\} \\ &= E\left\{(\Re\{\Delta h_l\})^2\right\} + E\left\{(\Im\{\Delta h_l\})^2\right\}.\end{aligned} \tag{3.33}$$

Applying (3.27), (3.29) and (3.30) to (3.33), one can obtain

$$\begin{aligned}E\{\Delta h_l^* \Delta h_l\} &= \frac{\sigma^2\left(2\vec{A}_l^H\vec{A}_l\vec{B}_l^H\vec{B}_l - \left|\vec{B}_l^H\vec{A}_l\right|^2\right)}{2\vec{A}_l^H\vec{A}_l\left(\vec{A}_l^H\vec{A}_l\vec{B}_l^H\vec{B}_l - \left|\vec{B}_l^H\vec{A}_l\right|^2\right)} \\ &= \frac{(5N^2-6N-1)\sigma^2}{2(N^2-1)}.\end{aligned} \tag{3.34}$$

Similarly, we can obtain

$$\begin{aligned}E\left\{\Delta f_l^2\right\} &= \frac{\sigma^2\vec{A}_l^H\vec{A}_l}{2|h_l|^2\left(\vec{A}_l^H\vec{A}_l\vec{B}_l^H\vec{B}_l - \left|\vec{B}_l^H\vec{A}_l\right|^2\right)} \\ &= \frac{3N^2\sigma^2}{2\pi^2(N^2-1)|h_l|^2}.\end{aligned} \tag{3.35}$$

When $N \to \infty$, we have

$$\lim_{N\to\infty} E\{\Delta h_l^* \Delta h_l\} = \lim_{N\to\infty} \frac{(5N^2-6N-1)\sigma^2}{2(N^2-1)} = \frac{5\sigma^2}{2} \tag{3.36}$$

and

$$\lim_{N\to\infty} E\left\{\Delta f_l^2\right\} = \lim_{N\to\infty} \frac{3N^2\sigma^2}{2\pi^2(N^2-1)|h_l|^2} = \frac{3\sigma^2}{2\pi^2|h_l|^2}. \tag{3.37}$$

In comparison, the Cramer-Rao bound (CRB) at large SNR for the classical frequency offset estimator of Moose is [33], [34]

$$CRB_{Moose}(f_l) = \left(\frac{2}{2\pi}\right)^2 \frac{2\sigma^2}{|h_l|^2 \vec{A}_l^H \vec{A}_l} = \frac{2\sigma^2}{\pi^2 |h_l|^2}. \tag{3.38}$$

Thus, from (3.37) and (3.38), we can see that the estimation algorithm considered in this paper can achieve $10\log_{10}\frac{2}{3/2} = 1.25$ dB gain over Moose's estimator. In Section 3.5, it will be seen that simulations performed to test the analytical results confirm the theoretical analysis.

3.5 Numerical Results and Discussions

In this section, we present some numerical results by considering a discrete baseband OFDM uplink in our simulation. The complex amplitude h_l is randomly chosen, and the normalized Doppler frequency f_l is chosen to be less than 0.1. The total number of subcarriers is fixed at $N = 256$. A binary phase-shift keying (BPSK) signaling scheme is adopted in the simulation. We set the stopping threshold values for the proposed algorithm as $\delta_h = 1\%$ and $\delta_f = 1\%$. A second-order Taylor series expansion is adopted in the simulation. Here the MSE is adopted as the performance measure.

Fig. 3.2 plots the MSE performance comparison between Algorithm 1 and Moose's estimator for f_l in a fading channel with $L = 1$. Here h_l is randomly chosen and f_l is set to be $f_l = 0.05$. The MSEs are obtained by averaging over 2000 trials, and the theoretical performance predictions of $\hat{f}_l$ are obtained using (3.37) and (3.38). From Fig. 3.2, we observe that the simulated MSE performance of both algorithms and their theoretical predictions are in excellent agreement over a wide range of SNR values. The simulated performance curve for Algorithm 1 exhibits an error floor, which is caused by the approximation error when the average SNR value is greater than 25 dB. This is the cause of discrepancy between the simulated curve and the theoretical curve for SNR greater than 25 dB. However, the values of MSE in these regions are small and the error floor is negligible. Fig. 3.2 also shows that Algorithm 1 outperforms Moose's estimator by about 1.2 dB over a wide range of SNR values.

Fig. 3.3 plots the average MSE performance of $\hat{h}_l$ in a multipath fading channel with $L = 3$ for the proposed algorithm and Figs. 3.4-3.6 plot MSE performance of $\hat{f}_l$ for the same system with three different f_l values. Here the h_l's are arbitrarily chosen to be $\vec{h}_l = [0.2944 + 1.6236j, -1.3362 - 0.6918j, 0.7143 + 0.858j]$, and the f_l's are set to be $\vec{f}_l = [0.04, 0.02, 0.06]$, respectively. The MSEs are obtained by averaging over 1500 trials, and the theoretical performance predictions of $\hat{h}_l$ and $\hat{f}_l$ are obtained using (3.36) and (3.37), respectively. From Figs. 3.3-3.6, we observe that the simulated MSE performance of $\hat{h}_l$ and $\hat{f}_l$ for Algorithm 1 and their theoretical predictions have excellent agreement over a wide range of SNR values. Figs. 3.4-3.6 also show that the MSE performance degrades with an increasing value of f_l, owing to increased (deterministic) approximation errors. From Fig. 3.4, it is seen for $f_l = 0.02$ that the simulated result and the theoretical result agree over the entire SNR range. Figs. 3.3, 3.5 and 3.6 suggest that the simulated results overestimate the theoretical results when the average SNR value is large. This is because we adopt a second-order Taylor series expansion to simplify the estimation problem, and the error floor is caused by the approximation error. However, the values of MSE in these regions are small and the error floor is negligible.

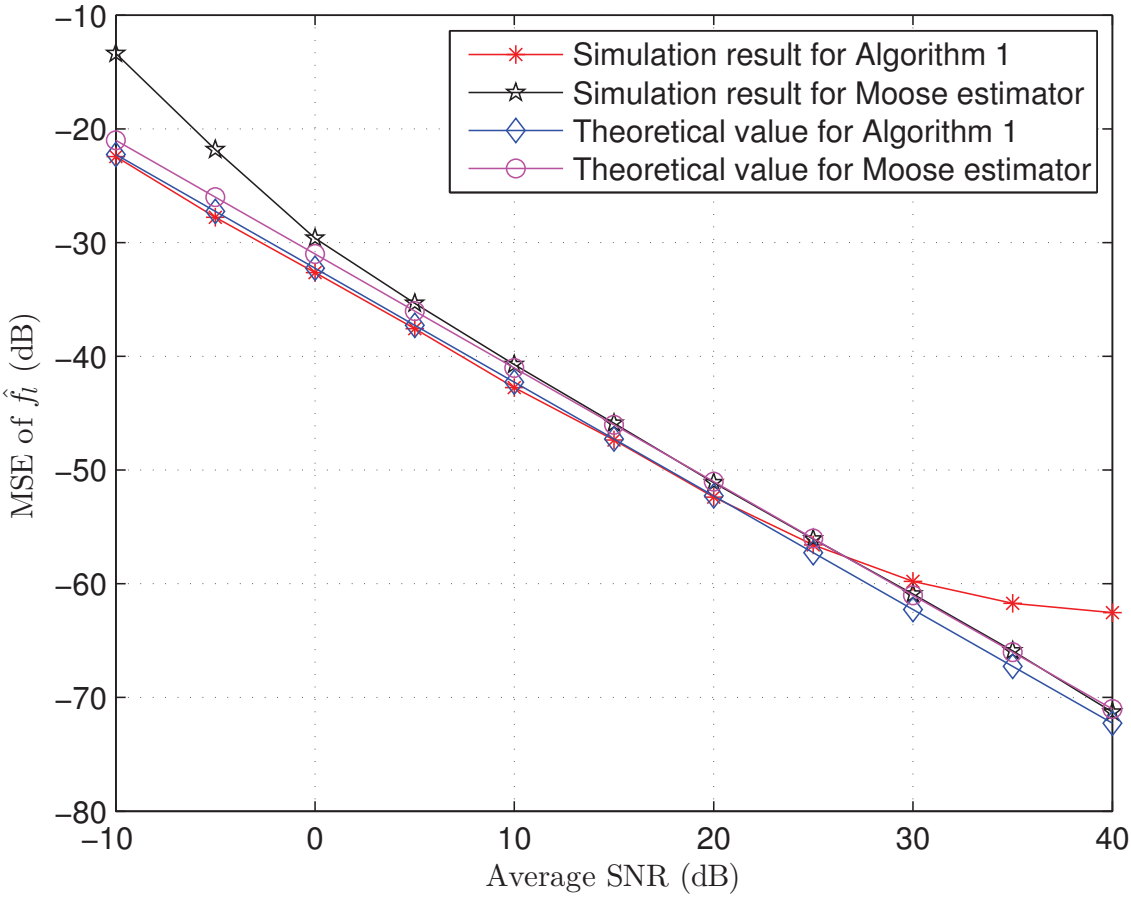

Figure 3.2: An MSE performance comparison between Algorithm 1 and Moose's estimator for $\hat{f}_l$ when $f_l = 0.05$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

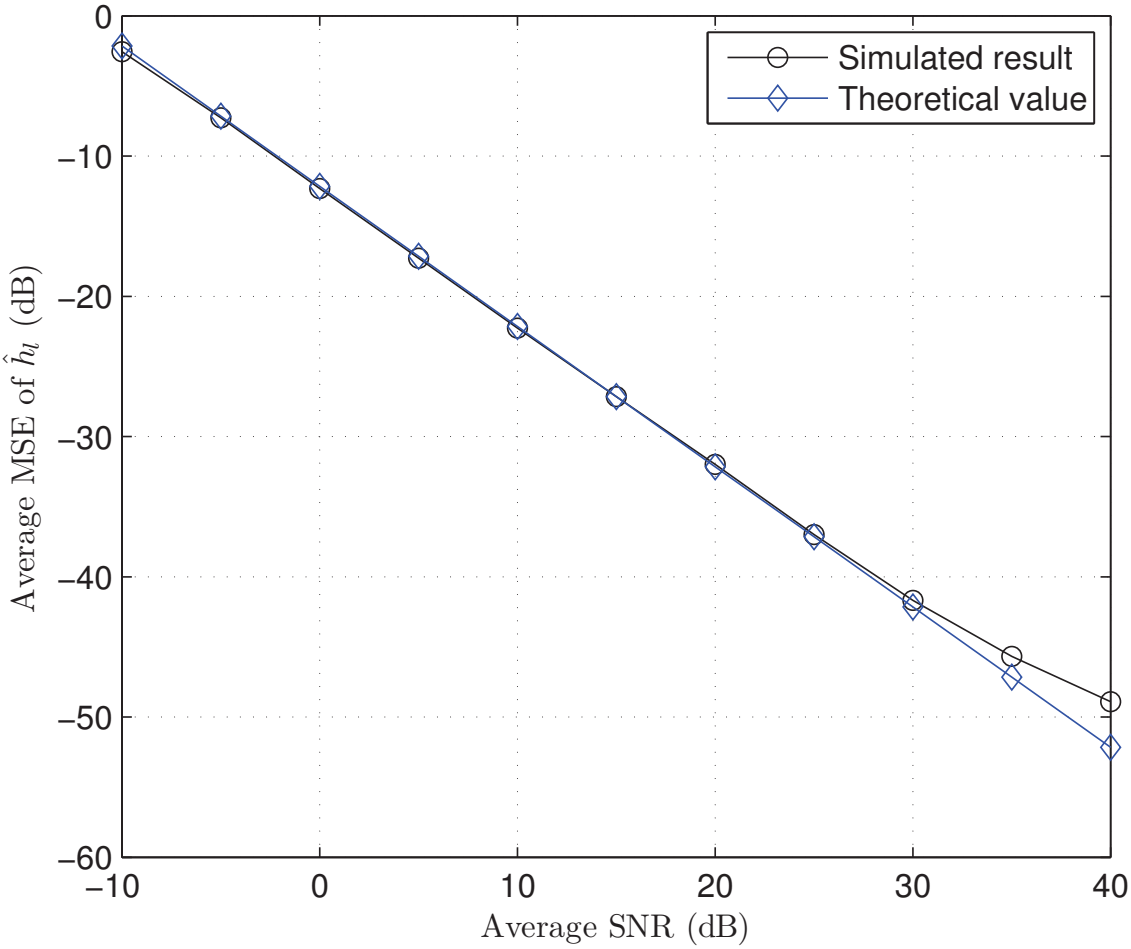

Figure 3.3: An average MSE performance comparison between the simulated results and the theoretical values for $\hat{h}_l$ of Algorithm 1 when $\delta_h = 1\%$ and $\delta_f = 1\%$.

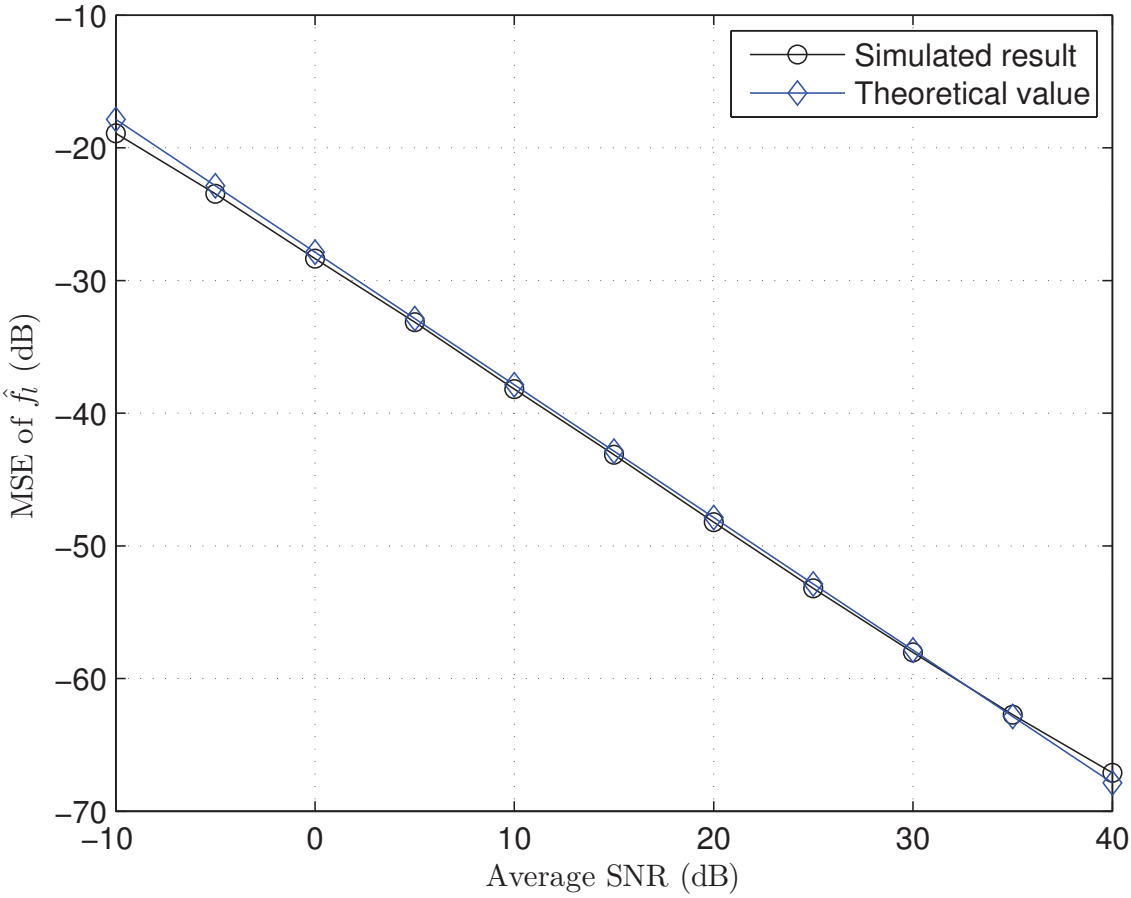

Figure 3.4: An MSE performance comparison between the simulated results and the theoretical values for $\hat{f}_l$ of Algorithm 1 when $f_l = 0.02$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

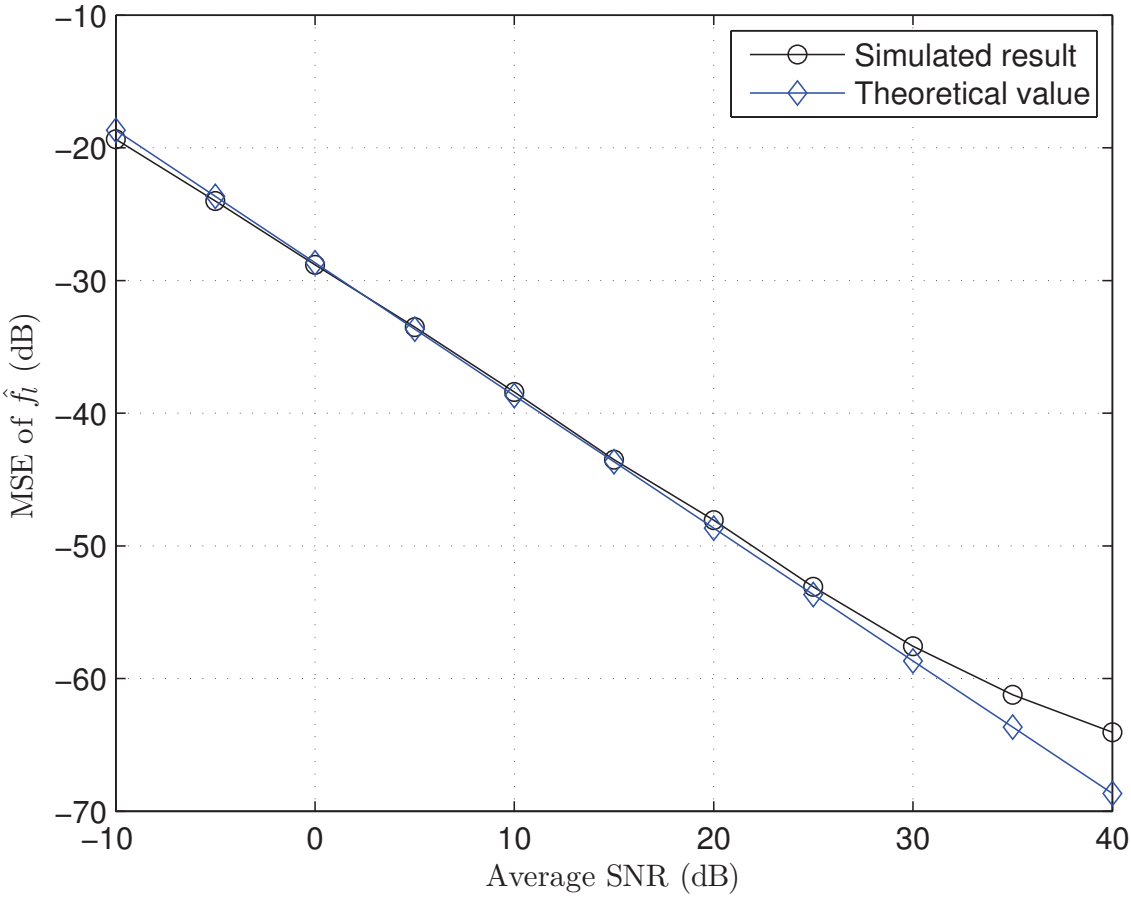

Figure 3.5: An MSE performance comparison between the simulated results and the theoretical values for $\hat{f}_l$ of Algorithm 1 when $f_l = 0.04$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

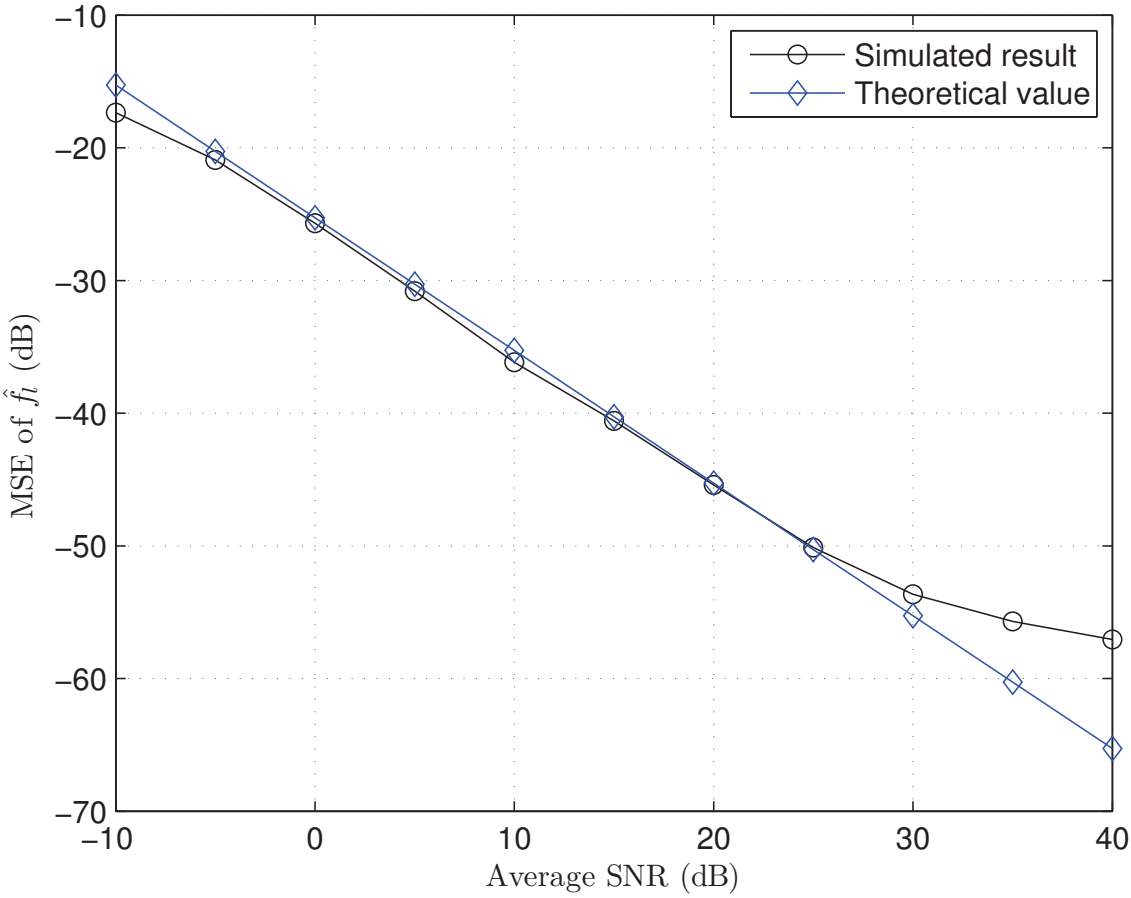

Figure 3.6: An MSE performance comparison between the simulated results and the theoretical values for $\hat{f}_l$ of Algorithm 1 when $f_l = 0.06$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

Chapter 4

Convergence Study of ML-based Channel Estimation Algorithms

In Chapter 3, by taking into account the ICI component, we proposed an ML-based channel estimation algorithm (which does not require channel statistics) for OFDM systems in dispersive time-varying fading channels. Our proposed channel estimation algorithm is iterative. In this chapter, we will study the convergence performance of the iterative algorithm presented in Chapter 3 analytically. Furthermore, we propose an improved fast converging channel estimation algorithm for OFDM systems in dispersive time-varying fading channels. An SOR method [35] is adopted to accelerate our proposed channel estimation algorithm. Our computer simulations demonstrate that the improved algorithm, which achieves the same MSE performance as the algorithm proposed in Chapter 3, has better convergence performance.

4.1 Convergence Performance Analysis of the Proposed Algorithm

In this section we first analyze the convergence rate of Algorithm 1 proposed in Chapter 3. Based on the convergence rate analysis we propose an improved fast converging algorithm using an SOR method [35] in the next section. We first rewrite the linear expression in (3.20a) for small pertur-

bation as a linear Gauss-Seidel iteration [35, eq. (3.15), page 72]

$$D\begin{pmatrix}\Delta\vec{\theta}_1^{(i+1)}\\ \Delta\vec{\theta}_2^{(i+1)}\\ \vdots\\ \Delta\vec{\theta}_L^{(i+1)}\end{pmatrix} = -L\begin{pmatrix}\Delta\vec{\theta}_1^{(i+1)}\\ \Delta\vec{\theta}_2^{(i+1)}\\ \vdots\\ \Delta\vec{\theta}_L^{(i+1)}\end{pmatrix} - U\begin{pmatrix}\Delta\vec{\theta}_1^{(i)}\\ \Delta\vec{\theta}_2^{(i)}\\ \vdots\\ \Delta\vec{\theta}_L^{(i)}\end{pmatrix} + \begin{pmatrix}\vec{n}_1\\ \vec{n}_2\\ \vdots\\ \vec{n}_L\end{pmatrix} \tag{4.1}$$

or, equivalently,

$$D\Delta\vec{\theta}^{(i+1)} = -L\Delta\vec{\theta}^{(i+1)} - U\Delta\vec{\theta}^{(i)} + \vec{n} \tag{4.2}$$

where D is a diagonal matrix whose entries are the diagonal elements of R; L and U are the lower and upper triangular matrices of R, i.e., $L+D+U=R$. Applying (3.20a) to (4.1), we can obtain

$$D\begin{pmatrix}\Delta\vec{\theta}_1^{(i+1)}\\ \Delta\vec{\theta}_2^{(i+1)}\\ \vdots\\ \Delta\vec{\theta}_L^{(i+1)}\end{pmatrix} = -L\begin{pmatrix}\Delta\vec{\theta}_1^{(i+1)}\\ \Delta\vec{\theta}_2^{(i+1)}\\ \vdots\\ \Delta\vec{\theta}_L^{(i+1)}\end{pmatrix} - U\begin{pmatrix}\Delta\vec{\theta}_1^{(i)}\\ \Delta\vec{\theta}_2^{(i)}\\ \vdots\\ \Delta\vec{\theta}_L^{(i)}\end{pmatrix} + R\begin{pmatrix}\Delta\vec{\theta}_1\\ \Delta\vec{\theta}_2\\ \vdots\\ \Delta\vec{\theta}_L\end{pmatrix}, \tag{4.3}$$

i.e.,

$$(D+L)\begin{pmatrix}\Delta\vec{\theta}_1^{(i+1)} - \Delta\vec{\theta}_1\\ \Delta\vec{\theta}_2^{(i+1)} - \Delta\vec{\theta}_2\\ \vdots\\ \Delta\vec{\theta}_L^{(i+1)} - \Delta\vec{\theta}_L\end{pmatrix} = -U\begin{pmatrix}\Delta\vec{\theta}_1^{(i)} - \Delta\vec{\theta}_1\\ \Delta\vec{\theta}_2^{(i)} - \Delta\vec{\theta}_2\\ \vdots\\ \Delta\vec{\theta}_L^{(i)} - \Delta\vec{\theta}_L\end{pmatrix} \tag{4.4}$$

or, in a more compact form as

$$(D+L)\left(\Delta\vec{\theta}^{(i+1)} - \Delta\vec{\theta}\right) = -U\left(\Delta\vec{\theta}^{(i)} - \Delta\vec{\theta}\right). \tag{4.5}$$

The convergence rate[2] of (4.5) is determined by the largest (in an absolute value sense) eigenvalue of $(D+L)^{-1}U$ [35].

The Gauss-Seidel iteration is a simple and efficient algorithm. However, by exploiting the

[2]With this convention, the smaller the largest eigenvalue is, the faster the iterative algorithm will converge.

structure of the Fisher information matrix R, one may adopt an overrelaxation philosophy to accelerate the convergence rate. We will next introduce an SOR method [35] to achieve faster convergence.

Corresponding to the linear iteration (4.2), an overrelaxed linear iteration can be expressed as [35, eq. (3.22), page 73]

$$D\Delta\vec{\theta}^{(i+1)} = \eta\left(-L\Delta\vec{\theta}^{(i+1)} - U\Delta\vec{\theta}^{(i)} + \vec{n}\right) + (1-\eta)D\Delta\vec{\theta}^{(i)} \tag{4.6}$$

where η is the relaxation factor. Applying (3.20a) to (4.6), we can obtain

$$(D+\eta L)\left(\Delta\vec{\theta}^{(i+1)} - \Delta\vec{\theta}\right) = [(1-\eta)D - \eta U]\left(\Delta\vec{\theta}^{(i)} - \Delta\vec{\theta}\right). \tag{4.7}$$

Similarly to the Gauss-Seidel iteration in (4.5), the convergence rate of the successive overrelaxed iteration in (4.7) depends on the largest absolute value of the eigenvalues of $(D+\eta L)^{-1}[(1-\eta)D - \eta U]$ [35]. To achieve the fastest convergence rate for (4.7), one is required to minimize the largest eigenvalue of $(D+\eta L)^{-1}[(1-\eta)D - \eta U]$ by choosing the optimal value for η. To simplify the analysis, we again assume that $R_{ij} = 0$ for $i \neq j$. Under this assumption, D, L and U are block diagonal matrices composed of 3×3 matrices D_l, L_l and U_l, $l = 1,\ldots,L$. Thus, for the first step, we consider the optimization on the eigenvalues of $(D_l+\eta L_l)^{-1}[(1-\eta)D_l - \eta U_l]$ separately. For the sake of convenience, we denote

$$\begin{aligned}
a_l &= \vec{A}_l^H \vec{A}_l, \\
b_l &= |h_l|^2 \vec{B}_l^H \vec{B}_l, \\
c_l &= \Re\left\{h_l^* \vec{B}_l^H \vec{A}_l\right\}, \\
d_l &= \Im\left\{h_l^* \vec{B}_l^H \vec{A}_l\right\},
\end{aligned}$$

and obtain

$$\begin{aligned} M_\eta &\triangleq D_l + \eta L_l \\ &= \begin{pmatrix} a_l & 0 & 0 \\ 0 & a_l & 0 \\ \eta c_l & -\eta d_l & b_l \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \eta c_l/a_l & -\eta d_l/a_l & 1 \end{pmatrix} \begin{pmatrix} a_l & 0 & 0 \\ 0 & a_l & 0 \\ 0 & 0 & b_l \end{pmatrix} \end{aligned} \tag{4.8}$$

$$\begin{aligned} N_\eta &\triangleq (1-\eta)D_l - \eta U_l \\ &= \begin{pmatrix} (1-\eta)a_l & 0 & -\eta c_l \\ 0 & (1-\eta)a_l & \eta d_l \\ 0 & 0 & (1-\eta)b_l \end{pmatrix}. \end{aligned} \tag{4.9}$$

It is straightforward to show that

$$M_\eta^{-1} N_\eta = \begin{pmatrix} 1-\eta & 0 & -\eta c_l/a_l \\ 0 & 1-\eta & \eta d_l/a_l \\ -\eta(1-\eta)c_l/b_l & \eta(1-\eta)d_l/b_l & (1-\eta)+\eta^2(c_l^2+d_l^2)/a_l b_l \end{pmatrix}. \tag{4.10}$$

The eigenvalues of (4.10) are the roots of the characteristic equation

$$\lambda^3 - \left[3(1-\eta) + \frac{\eta^2(c_l^2+d_l^2)}{a_l b_l}\right]\lambda^2 + (1-\eta)\left[3(1-\eta) + \frac{\eta^2(c_l^2+d_l^2)}{a_l b_l}\right]\lambda - (1-\eta)^3 = 0. \tag{4.11}$$

For the sake of simplicity, we denote

$$k_l \triangleq \frac{c_l^2+d_l^2}{a_l b_l} = \frac{\left|\vec{B}_l^H \vec{A}_l\right|^2}{\vec{A}_l^H \vec{A}_l \vec{B}_l^H \vec{B}_l} \tag{4.12}$$

and simplify (4.11) to

$$\lambda^3 - \left[3(1-\eta)+\eta^2 k_l\right]\lambda^2 + (1-\eta)\left[3(1-\eta)+\eta^2 k_l\right]\lambda - (1-\eta)^3 = 0. \tag{4.13}$$

It is shown in Appendix D that the optimal value of η which minimizes the largest eigenvalue (in absolute value) of $M_\eta^{-1}N_\eta$, is

$$\eta = \frac{2(1-\sqrt{1-k_l})}{k_l}, \tag{4.14}$$

and the corresponding largest absolute value for the eigenvalues of $M_\eta^{-1}N_\eta$ is

$$|\lambda| = |1-\eta| = \left|1 - \frac{2(1-\sqrt{1-k_l})}{k_l}\right|. \tag{4.15}$$

Generally, an SOR method requires knowledge about the structure of the Fisher information matrix. It is observed in (3.20b) that this matrix (the Fisher information matrix R_{ll}) depends on the channel realization h_l. However, (4.12) and (4.14) indicate that for a given 3×3 Fisher information matrix R_{ll}, the optimal η value does not depend on the channel realization h_l. Instead, it only relies on the training sequence. In other words, given the training sequence, the receiver can employ η in (4.14) to achieve a fast convergence rate for the overrelaxed iteration.

Typically convergence accelerating algorithms require full knowledge of the Fisher information matrix R_{ll} to achieve better convergence rates. This implies that one needs to estimate h_l and it renders such algorithms less useful. In contrast, the overrelaxation method proposed in this work only requires information about the training sequence and, therefore, is more robust. This property makes the overrelaxation method an attractive approach to achieve a fast convergence rate in practical receivers.

From (4.12) and (4.14), it is observed that the η value depends on k_l and may vary for different l. Substitution of (3.30) into (4.12) yields

$$\begin{aligned} k_l &= \frac{\pi^2(N-1)^2/N^2}{2\pi^2(N-1)(2N-1)/3N^2} \\ &= \frac{3(N-1)}{2(2N-1)} \end{aligned} \tag{4.16}$$

which is, in fact, independent of l assuming a white training sequence is employed. By letting $N \to \infty$, we have the limiting value η, from (4.14) and (4.16), as

$$\begin{aligned} \eta &= \lim_{N\to\infty} \frac{2\left(1-\sqrt{1-k_l}\right)}{k_l} \\ &= \frac{4}{3} \end{aligned} \tag{4.17}$$

which leads to a convergence rate of $|1-\eta| = 1/3$.

In comparison, considering the Gauss-Seidel iteration by letting $\eta = 1$, one can express (4.10) as

$$M_\eta^{-1} N_\eta = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_l \end{pmatrix} \tag{4.18}$$

and conclude that the largest eigenvalue for the iteration is k_l. Thus, the largest eigenvalue for the Gauss-Seidel iteration is $k_l = 3/4$, which is larger than that of 1/3 for the SOR method, implying a faster convergence for the later[3].

In each Gauss-Seidel iteration, a value that maximizes the likelihood function is produced as an output. In contrast, an SOR method adopts a less aggressive strategy by combining the Gauss-Seidel solution and the previous value linearly as

$$h_l^{(i+1)} = \eta \tilde{h}_l^{(i+1)} + (1-\eta) h_l^{(i)} \tag{4.19a}$$

$$f_l^{(i+1)} = \eta \tilde{f}_l^{(i+1)} + (1-\eta) f_l^{(i)} \tag{4.19b}$$

where η is an overrelaxation parameter, and $\tilde{h}_l^{(i+1)}$ and $\tilde{f}_l^{(i+1)}$ are the solutions to the Gauss-Seidel iteration. When $\eta = 1$, the SOR method degenerates to a Gauss-Seidel iteration.

[3]We should comment that the η value in (4.17) is only asymptotically optimal. In Section 4.3, we will demonstrate that this suboptimal solution already shows significant improvement over the Gauss-Seidel iteration.

4.2 An Improved Channel Estimation Algorithm

The above discussion is based on the linear approximation (3.20a) near the ML estimation result. Algorithm 1 is, however, a nonlinear estimator. Using the overrelaxation method, we can propose an improved new iterative ML algorithm as shown in Table 4.1. Note that the function $L(h_l, f_l)$ in *Step 1* is the cost function defined in (3.19) and $\eta = 4/3$, which is the asymptotically optimal value given by (4.17). Our numerical results will show that the improved algorithm can achieve good performance, both in terms of MSE and convergence rate.

Table 4.1: Algorithm 2

Initialization: Initialize $h_l^{(0)}$ and $f_l^{(0)}$, for $l = 1, 2, \ldots, L$. Set the iteration counter $i = 0$.

Iterations:

Step 1: For each $l = 1, 2, \ldots, L$, calculate

$$\tilde{h}_l^{(i+1)} = \arg\min_{h_l} L\left(h_l, f_l^{(i)}\right)$$

$$\tilde{f}_l^{(i+1)} = \arg\min_{f_l} L\left(h_l^{(i+1)}, f_l\right)$$

Let $h_l^{(i+1)} = \eta\tilde{h}_l^{(i+1)} + (1-\eta)h_l^{(i)}$ and $f_l^{(i+1)} = \eta\tilde{f}_l^{(i+1)} + (1-\eta)f_l^{(i)}$.

Step 2: If $\max_l \left(\frac{\left|h_l^{(i+1)} - h_l^{(i)}\right|}{\left|h_l^{(i)}\right|} \right) \times 100\% > \delta_h$ or $\max_l \left(\frac{\left|f_l^{(i+1)} - f_l^{(i)}\right|}{\left|f_l^{(i)}\right|} \right) \times 100\% > \delta_f$, let $i = i+1$ and go to *Step 1*, otherwise output $h_l^{(i+1)}$'s and $f_l^{(i+1)}$'s.

4.3 Numerical Results and Discussions

In this section, we present some numerical results by considering the same discrete baseband OFDM uplink transmission discussed in Chapter 3 in our simulation. All the simulation parameters adopted here are the same as those considered in Section 3.5. Here we compare the MSE performance and convergence performance of Algorithm 1 and Algorithm 2 to demonstrate the performance of the improved algorithm.

Fig. 4.1 plots the average MSE performance of $\hat{h}_l$ for the two proposed algorithms and Figs. 4.2-4.4 plot MSE performance of $\hat{f}_l$ for the same system with three different f_l values. The MSEs are obtained by averaging over 2000 trials, and the theoretical performance predictions of $\hat{h}_l$ and $\hat{f}_l$ are

obtained using (3.36) and (3.37), respectively. From Figs. 4.1-4.4, we observe that the simulated MSE performance of $\hat{h}_l$ and $\hat{f}_l$ for Algorithm 1 and Algorithm 2, along with their theoretical predictions, have excellent agreement over a wide range of SNR values. We can notice a slight difference between the MSE performance of Algorithm 1 and Algorithm 2 when the SNR values are greater than 30 dB. However, the MSEs of both algorithms in this SNR region are less than -50 dB and the differences are therefore negligible. We can conclude that the two algorithms have almost the same performance in terms of MSE.

Fig. 4.5 plots the average number of iterations required by the two proposed algorithms to achieve the MSE performance shown in Figs. 4.1-4.4. As expected, Fig. 4.5 shows that the average number of iterations of both algorithms decreases with increasing value of SNR up to 20 dB. The average number of iterations of both algorithms is unchanged when the average SNR value is greater than 20 dB. It is shown in Fig. 4.5 that Algorithm 2 converges faster than Algorithm 1, and it achieves 63% improvement in the number of iterations in the large SNR region. From Fig. 4.5, it is seen that the asymptotically optimal η value obtained under a high SNR assumption also works when the SNR value is low.

Figs. 4.6 and 4.7 respectively plot the average MSE performance of $\hat{h}_l$ and $\hat{f}_l$ versus the number of iterations. For both figures, the average SNR value is fixed at 25 dB, and we observe that the MSE for Algorithm 2 decreases faster than that of Algorithm 1. Our numerical results clearly demonstrate that Algorithm 2, which uses an SOR technique, can achieve faster convergence than Algorithm 1.

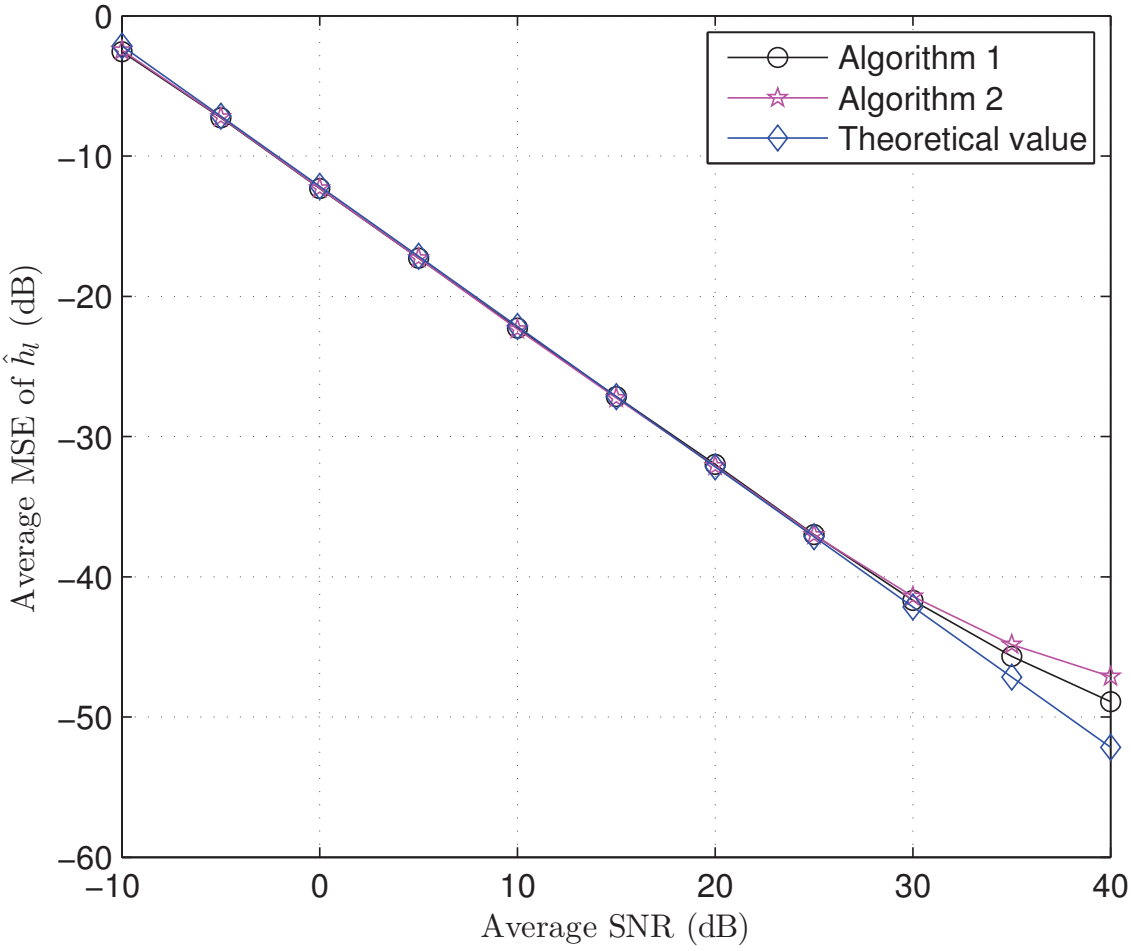

Figure 4.1: An average MSE performance comparison between Algorithm 1 and Algorithm 2, along with the theoretical values for $\hat{h}_l$, when $\delta_h = 1\%$ and $\delta_f = 1\%$.

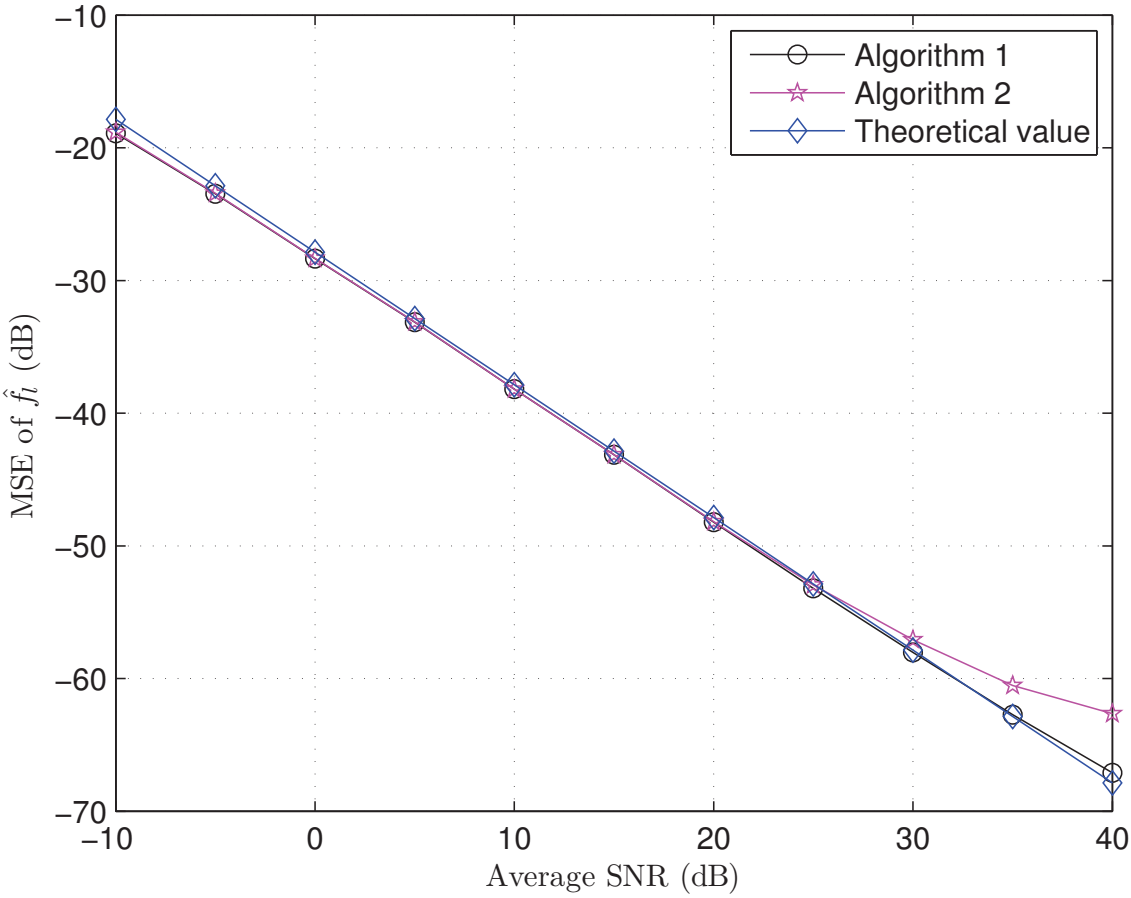

Figure 4.2: An MSE performance comparison between Algorithm 1 and Algorithm 2, along with the theoretical values for $\hat{f}_l$, when $f_l = 0.02$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

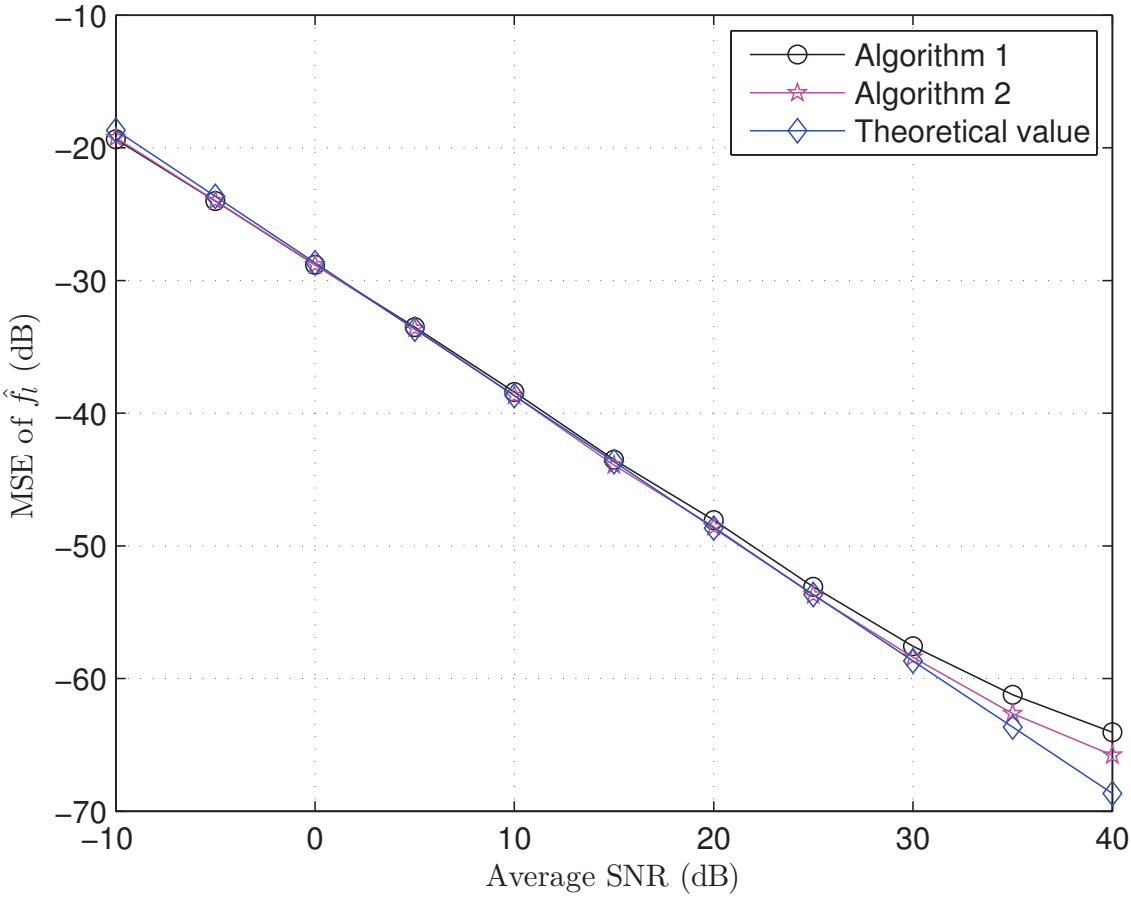

Figure 4.3: An MSE performance comparison between Algorithm 1 and Algorithm 2, along with the theoretical values for $\hat{f}_l$, when $f_l = 0.04$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

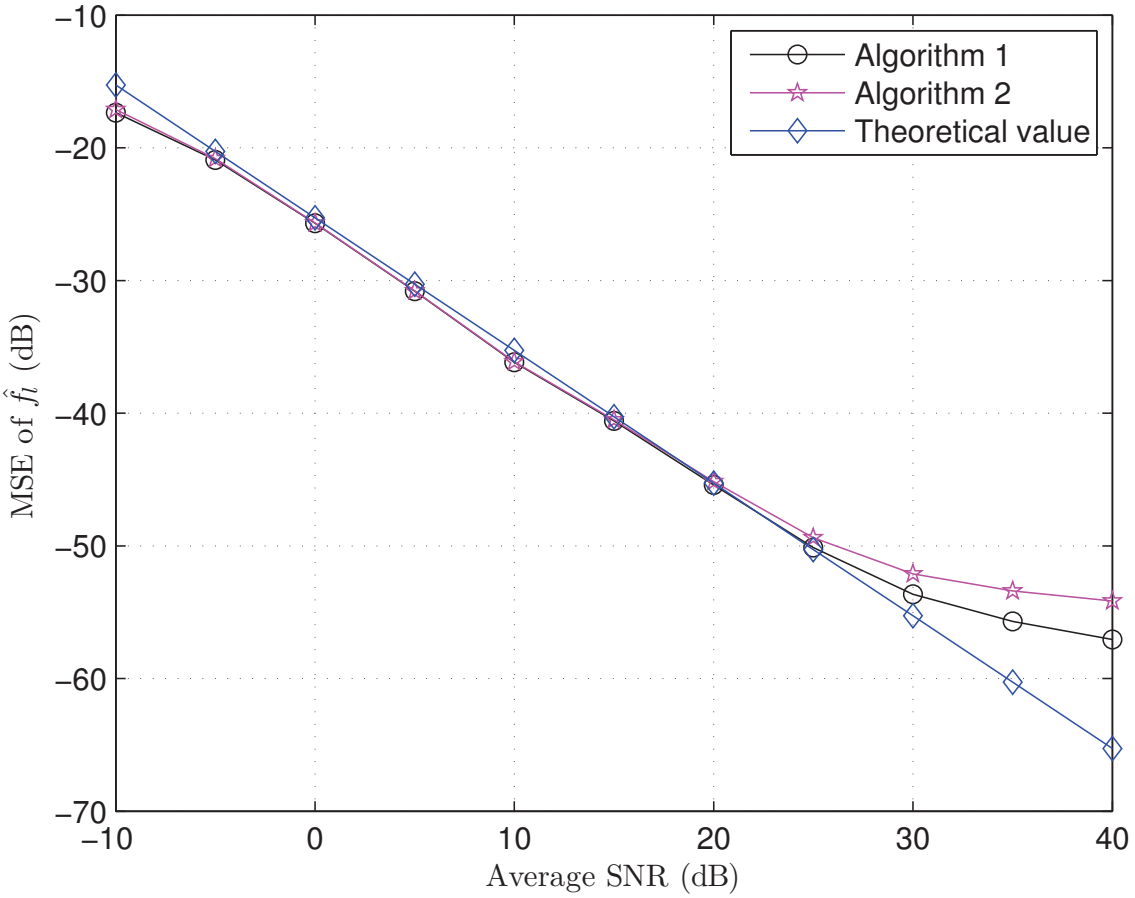

Figure 4.4: An MSE performance comparison between Algorithm 1 and Algorithm 2, along with the theoretical values for $\hat{f}_l$, when $f_l = 0.06$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

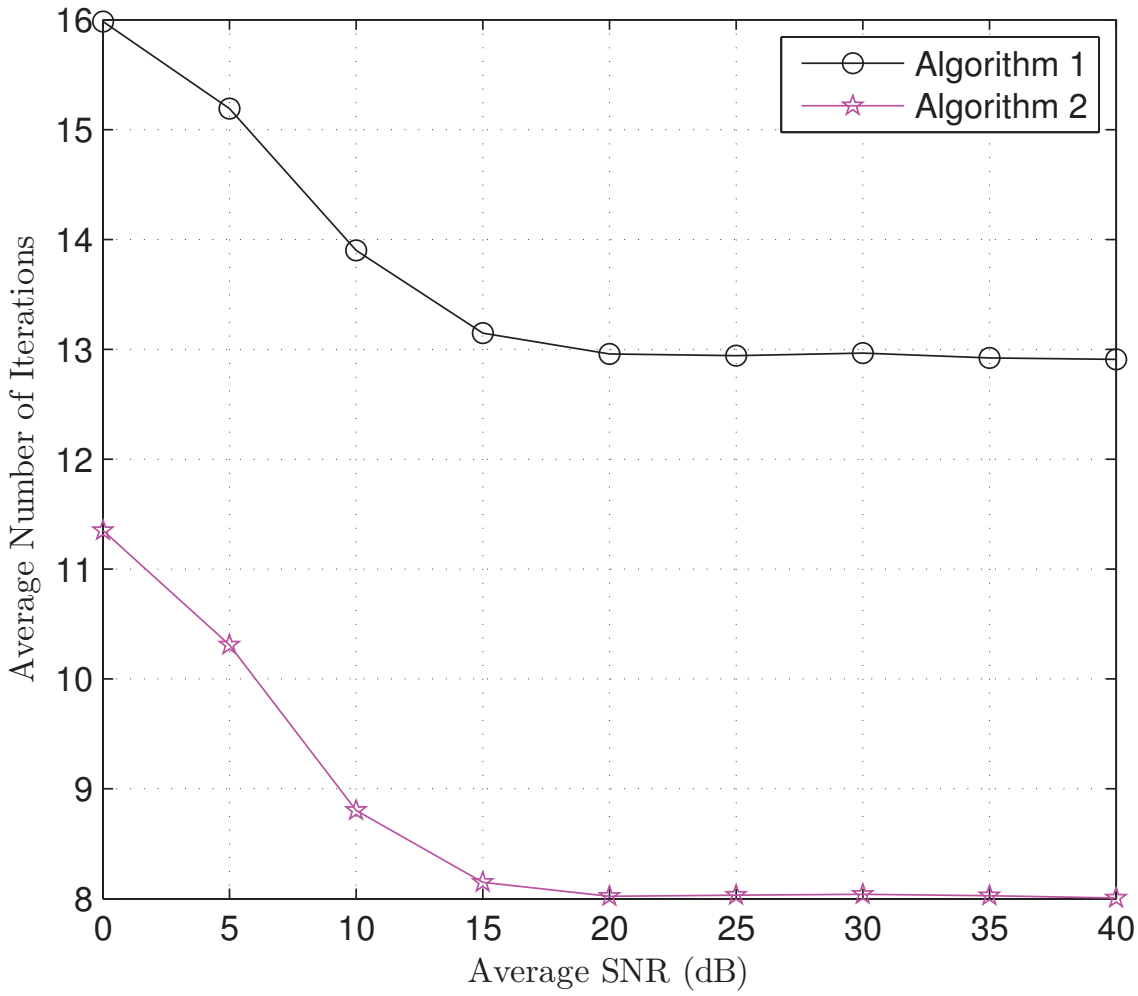

Figure 4.5: An average number of iterations comparison between Algorithm 1 and Algorithm 2 when $\eta = 4/3$, $\delta_h = 1\%$ and $\delta_f = 1\%$.

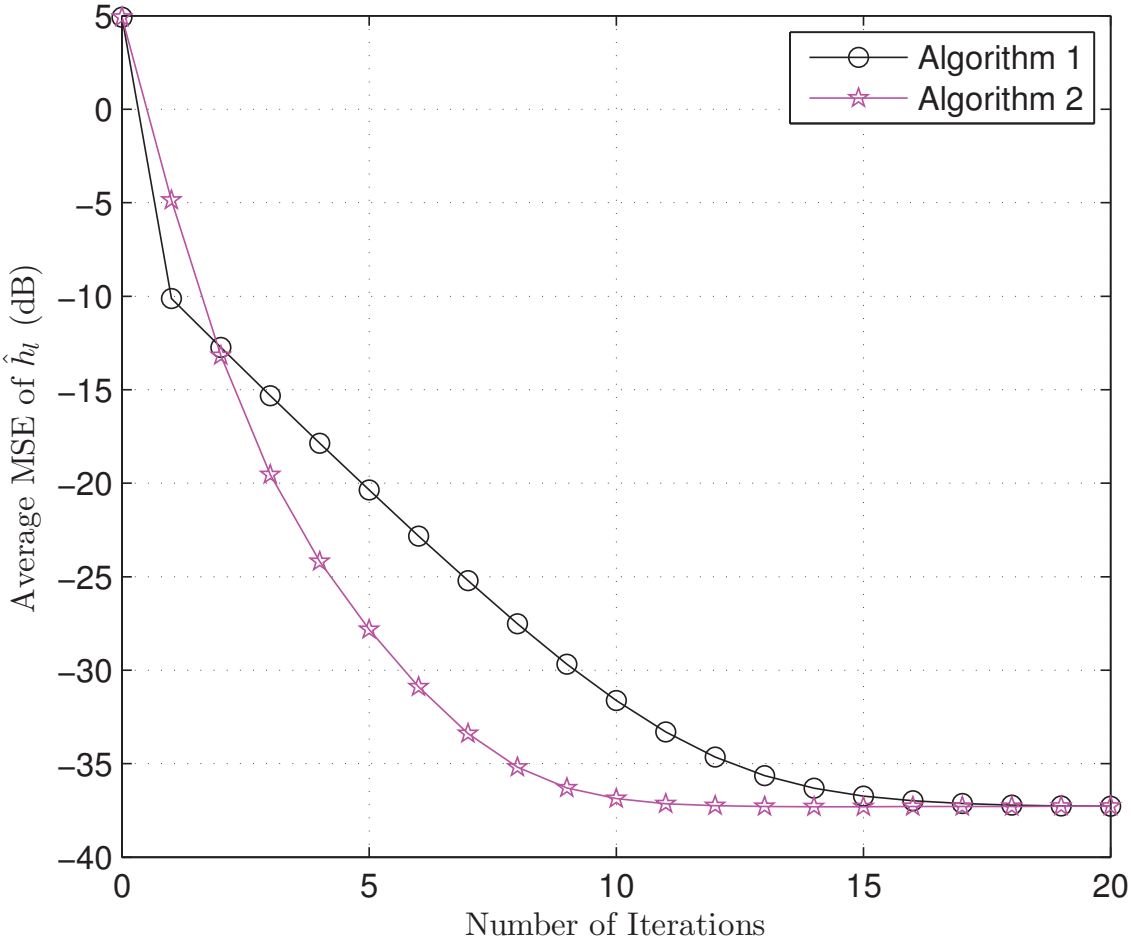

Figure 4.6: The average MSE performance versus number of iterations for $\hat{h}_l$ of Algorithm 1 and Algorithm 2 when $\eta = 4/3$ and SNR $= 25$.

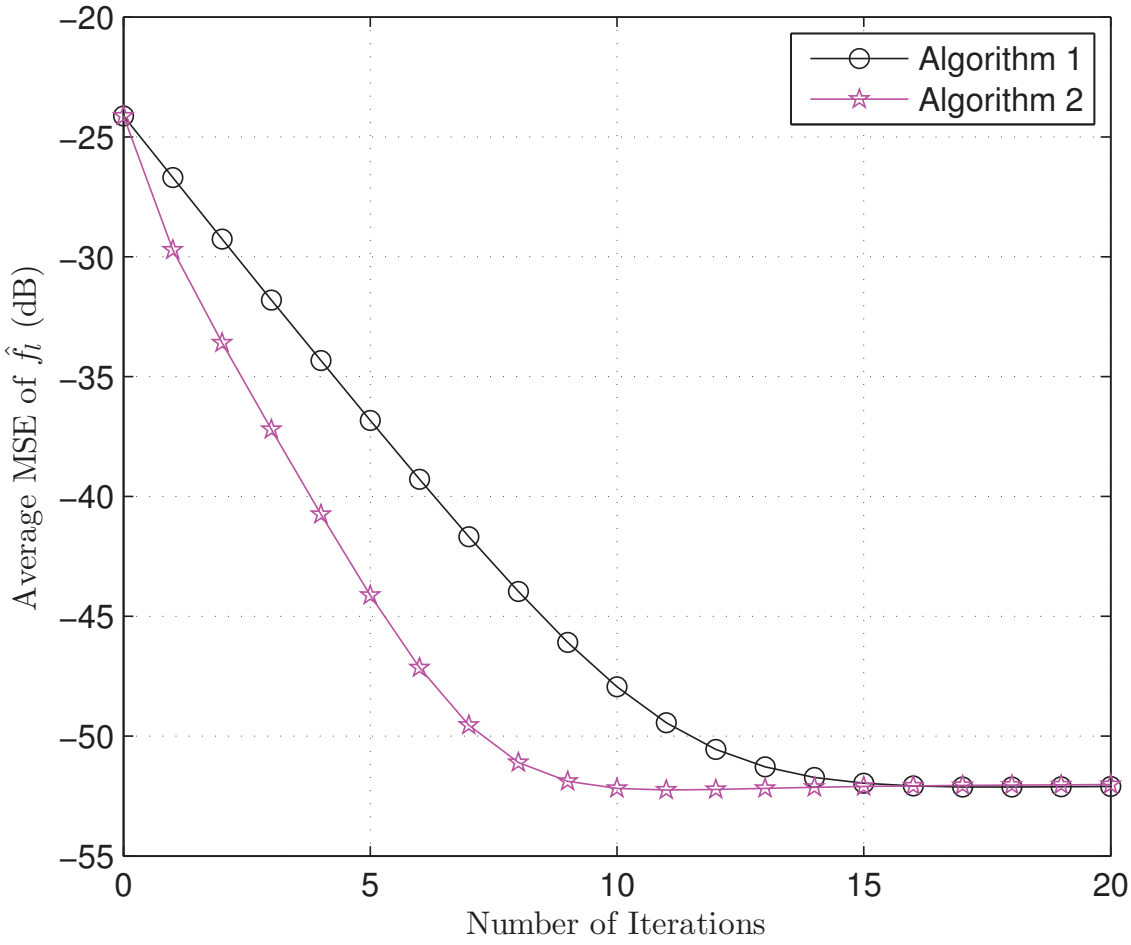

Figure 4.7: The average MSE performance versus number of iterations for $\hat{f}_l$ of Algorithm 1 and Algorithm 2 when $\eta = 4/3$ and SNR $= 25$ dB.

Chapter 5

Conclusions

In this chapter, we first summarize the contributions of this work, and then suggest some future work.

5.1 Summary of Contributions

Through the numerical simulations and mathematical analyses, the performance of our proposed algorithms have been verified. In the simulation, we do not compare the performance of our proposed algorithms with any other solutions because of the uniqueness of our channel model. However, we do compare the simulated estimate's performance with the CRB, which is the best one can achieve. The contributions of this thesis can be summarized as follows.

1. A discrete CIR model, which allows the CIR to vary within one OFDM symbol, has been developed for OFDM uplink transmission in macrocellular systems. The CIR is characterized by the channel parameters h_l and f_l, where h_l is the complex amplitude for the lth multipath and f_l is the corresponding Doppler frequency shift normalized to the duration of an OFDM symbol. In this case, the channel state can be determined by estimating the unknown channel parameters so that the channel estimation problem can be simplified.

2. Taking into account the ICI component, we formulate the channel estimation problem based on our proposed discrete CIR model. Due to the fact that f_l is typically less than 0.1 in practical mobile communication scenarios, the ICI factor $R(k', k, f_l)$ is represented by a truncated Taylor series expansion to facilitate the channel estimation problem. We also demonstrate that a second-order Taylor series approximation is sufficient for a typical channel estimation problem.

3. We propose two iterative ML-based channel estimation algorithms, which do not require channel statistics, for OFDM systems in dispersive time-varying fading channels. These two algorithms can estimate the desired parameters accurately in order to provide channel state information for the receiver. Our proposed channel estimation algorithms are particularly useful for an OFDM system which has already compensated for the frequency offset due to local oscillator mismatch. The MSE performance of the proposed algorithms in large SNR regions has been analyzed using a small perturbation technique. Since our algorithms are iterative, the convergence performance is also analyzed. Based on the analysis, an SOR method is adopted to accelerate the proposed algorithm. The performance analyses are verified by our simulated results.

5.2 Future work

In this work, we have proposed two iterative ML-based channel estimation algorithms and demonstrated their performance in terms of MSE. However, channel estimation is only one part of the receiver design. Based on accurate channel estimates, we will design a receiver structure based on the proposed channel model. The bit-error rate (BER) performance of such a system will also be investigated.

Single-carrier frequency division multiple access (SC-FDMA) is an OFDM based technique that has been adopted for the Long Term Evolution (LTE) uplink transmission by the third Generation Partnership Project (3GPP). We believe that our proposed channel model and channel estimation algorithms can be applied to such a system. In our future research, we will study the channel estimation problem for SC-FDMA systems using the dispersive, time-varying channel model proposed in this thesis.

Bibliography

[1] M. K. Ozdemir and H. Arslan, "Channel estimation for wireless OFDM systems," *IEEE Communications Surveys and Tutorials*, vol. 9, no. 2, pp. 18–48, July 2007.

[2] I. Koffman and V. Roman, "Broadband wireless access solutions based on OFDM access in IEEE 802.16," *IEEE Communications Magazine*, vol. 40, no. 4, pp. 96–103, Apr. 2002.

[3] I. Barhumi, G. Leus, and M. Moonen, "Optimal training design for MIMO-OFDM systems in mobile wireless channels," *IEEE Transactions on Signal Processing*, vol. 51, no. 6, pp. 1615–1624, June 2003.

[4] Y. Li and G. L. Stüber, *Orthogonal Frequency Division Multiplexing for Wireless Communications*. New York: Springer Publications, 2006.

[5] J. G. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2000.

[6] J.-J. van de Beek, O. Edfors, M. Sandell, S. Wilson, and P. Börjesson, "On channel estimation in OFDM systems," in *Proc. IEEE Vehicular Technology Conference, VTC 1995*, Chicago, IL, USA, July 25–28 1995, vol. 2, pp. 815–819.

[7] O. Edfors, M. Sandell, J.-J. van de Beek, S. K. Wilson, and P. O. Börjesson, "OFDM channel estimation by singular value decomposition," *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 931–939, July 1998.

[8] P. Chen and H. Kobayashi, "Maximum likelihood channel estimation and signal detection for OFDM systems," in *Proc. IEEE International Conference on Communications, ICC 2002*, New York, NY, USA, May 2002, vol. 3, pp. 1640–1645.

[9] Y. Li, L. J. Cimini, and N. R. Sollenberger, "Robust channel estimation for OFDM systems with rapid dispersive fading channels," *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 902–915, July 1998.

[10] J. K. Moon and S. I. Choi, "Performance of channel estimation methods for OFDM systems in a multipath fading channels," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 161–170, Feb. 2000.

[11] Y. Li, "Pilot-symbol-aided channel estimation for OFDM in wireless systems," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 4, pp. 1207–1215, July 2000.

[12] Y. Zhao and A. Huang, "A novel channel estimation method for OFDM mobile communications systems based on pilot signals and transform-domain processing," in *Proc. IEEE Vehicular Technology Conference, VTC 1997*, Phoenix, AZ, USA, May 4–7 1997, vol. 3, pp. 2089–2093.

[13] M. X. Chang and Y. T. Su, "Model-based channel estimation for OFDM signals in Rayleigh fading," *IEEE Transactions on Communications*, vol. 50, no. 4, pp. 540–544, Apr. 2002.

[14] F. Z. Merli and G. M. Vitetta, "Iterative ML-based estimation of carrier frequency offset, channel impulse response and data in OFDM transmissions," *IEEE Transactions on Communications*, vol. 56, no. 3, pp. 497–506, Mar. 2008.

[15] G. L. Stüber, *Principles of Mobile Communications*. Norwell, MA: Kluwer, 1996.

[16] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River: Prentice-Hall, 2001.

[17] A. Goldsmith, *Wireless Communications*. New York: Cambridge University Press, 2005.

[18] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. New York: Cambridge University Press, 2005.

[19] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge, UK: Cambridge Univeristy Press, 2003.

[20] R. van Nee and R. Prasad, *OFDM for Wireless Multimedia Communications*. Boston: Artech House, 2000.

[21] R. R. Mosier and R. G. Clabaugh, "Kineplex, a bandwidth efficient binary transmission system," *IEEE Transactions on Power Apparatus and Systems*, vol. 76, pp. 723–728, Jan. 1958.

[22] R. W. Chang, "Synthesis of band limited orthogonal signals for multichannel data transmission," *Bell System Technical Journal*, vol. 45, no. 12, pp. 1775–1796, Dec. 1966.

[23] B. R. Salzberg, "Performance of an efficient parallel data transmission system," *IEEE Transactions on Communication Technology*, vol. COM-15, no. 6, pp. 805–813, Dec. 1967.

[24] S. B. Weinstein and P. M. Ebert, "Data transmission by frequency-division multiplexing using the discrete Fourier transform," *IEEE Transactions on Communication Technology*, vol. COM-19, no. 5, pp. 628–634, Oct. 1971.

[25] A. Gorokhov and J.-P. Linnartz, "Robust OFDM receivers for dispersive time-varying channels: equalization and channel acquisition," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 527–583, Apr. 2004.

[26] R. Steele and L. Hanzo, *Mobile Radio Communications*. New York: Wiley-IEEE Press, 1999.

[27] X. Ma, H. Kobayashi, and S. C. Schwartz, "Joint frequency offset and channel estimation for OFDM," in *Proc. IEEE Global Telecommunications Conference, GLOBECOM 2003*, San Francisco, CA, USA, Dec. 1–5 2003, vol. 1, pp. 15–19.

[28] T. Cui and C. Tellambura, "Joint frequency offset and channel estimation for OFDM systems using pilot symbols and virtual carriers," *IEEE Transactions on Wireless Communications*, vol. 6, no. 4, pp. 1193–1202, Apr. 2007.

[29] K. B. Petersen and M. S. Petersen, *The Matrix Cookbook*. http://matrixcookbook.com, 2008.

[30] J. A. Fessler and A. O. Hero, "Space-alternating generalized expectation-maximization algorithm," *IEEE Transactions on Signal Processing*, vol. 42, no. 10, pp. 2664–2677, Oct. 1994.

[31] K. Li and H. Liu, "Joint channel and carrier offset estimation in CDMA communications," *IEEE Transactions on Signal Processing*, vol. 47, no. 7, pp. 1811–1822, July 1999.

[32] P. Stoica and O. Besson, "Training sequence design for frequency offset and frequency-selective channel estimation," *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1910–1917, Nov. 2003.

[33] P. H. Moose, "A technique for orthogonal frequency division multiplexing offset correction," *IEEE Transactions on Communications*, vol. 42, no. 10, pp. 2908–2914, Oct. 1994.

[34] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, Dec. 1997.

[35] D. M. Young, *Iterative Solution of Large Linear Systems*. New York: Academic Press, 1971.

[36] G. T. RAN, "Spatial channel model for Multiple Input Multiple Output (MIMO) simulations," Technical Report, Dec. 2008.

[37] G. Cardano and T. R. Witmer, *The Great Art, or: The Rules of Algebra*, 1st ed. Cambridge: MIT Press, 1968.

Appendix A

Derivation of the Discrete CIR Model

For a multipath CIR described in (3.2), each resolvable path consists of a number of non-resolvable components so that $\gamma_l(t)$ can be expressed as

$$\gamma_l(t) = \sum_i \alpha_{l,i} e^{-j\phi_{l,i}(t)} \tag{A.1}$$

where $\alpha_{l,i}$ and $\phi_{l,i}(t)$ are respectively the complex amplitude and the phase shift incurred by the ith non-resolvable component of the lth resolvable path. The phase shift $\phi_{l,i}(t)$ is due to the time delay and Doppler frequency shift and is given by

$$\phi_{l,i}(t) = 2\pi f_c \tau_l - 2\pi f_{D_{l,i}} t \tag{A.2}$$

where f_c is the carrier frequency and $f_{D_{l,i}}$ is the Doppler frequency shift at the ith non-resolvable component of the lth resolvable path. In this work, we consider an uplink transmission in a macro-cellular system where the angular spread is small for those non-resolvable components [36]. Consequently, we can assume $f_{D_{l,i}} = f_{D_l}$ and

$$\phi_{l,i}(t) = 2\pi f_c \tau_l - 2\pi f_{D_l} t. \tag{A.3}$$

Applying (A.3) to (A.1), one can obtain

$$\gamma_l(t) = e^{j2\pi f_{D_l} t} \underbrace{\sum_i \alpha_{l,i} e^{-j2\pi f_c \tau_l}}_{h_l} \tag{A.4}$$

and rewrite (3.2) as

$$h(\tau,t) = \sum_{l=1}^{L} h_l e^{j2\pi f_{D_l} t} \delta(\tau - \tau_l). \tag{A.5}$$

Therefore, the discrete-time CIR can be expressed as

$$\begin{aligned} h[m,n] &\stackrel{\triangle}{=} h(mT_{sa}, nT_{sa}) \\ &= \sum_{l=1}^{L} h_l e^{\frac{j2\pi f_{D_l} T_s n T_{sa}}{N T_{sa}}} \delta(mT_{sa} - T_{sa}\tau_l/T_{sa}) \\ &= \sum_{l=1}^{L} h_l e^{\frac{j2\pi f_l n}{N}} \delta[m - n_l] \end{aligned} \tag{A.6}$$

where T_{sa} is the sampling interval defined as $T_{sa} = T_s/N$ and T_s is the OFDM symbol duration, f_l is the Doppler frequency shift at the lth multipath normalized by $1/T_s$, i.e., $f_l = f_{D_l} T_s$, and n_l is the corresponding delay in samples, i.e., $n_l = \lfloor \tau_l/T_{sa} \rfloor$.

Appendix B

Derivation of (3.20)

Recalling (3.8) and (3.13), the cost function we try to minimize is

$$\begin{aligned} J(h_l, f_l) &= \left| \vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 + \cdots \right) \right|^2 \\ &= \left| \vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 + \cdots \right) \right|^H \left| \vec{Y} - \sum_{l=1}^{L} h_l \left(\vec{A}_l + \vec{B}_l f_l + \vec{C}_l f_l^2 + \cdots \right) \right| \end{aligned} \tag{B.1}$$

where $(\cdot)^H$ denotes the Hermitian transpose of the argument matrix. According to (30), (31), and (32) in [31], we can obtain

$$\Delta\vec{\theta}_i = \begin{pmatrix} \Re\{\Delta h_i\} \\ \Im\{\Delta h_i\} \\ \Delta f_i \end{pmatrix} \tag{B.2}$$

$$\vec{n}_i = - \begin{pmatrix} \frac{\partial J(h_l, f_l)}{\partial \Re\{h_i\}} \\ \frac{\partial J(h_l, f_l)}{\partial \Im\{h_i\}} \\ \frac{\partial J(h_l, f_l)}{\partial f_i} \end{pmatrix} \tag{B.3}$$

and

$$R_{ij} = \begin{pmatrix} \frac{\partial^2 J(h_l, f_l)}{\partial \Re\{h_i\} \partial \Re\{h_j\}} & \frac{\partial^2 J(h_l, f_l)}{\partial \Re\{h_i\} \partial \Im\{h_j\}} & \frac{\partial^2 J(h_l, f_l)}{\partial \Re\{h_i\} \partial f_j} \\ \frac{\partial^2 J(h_l, f_l)}{\partial \Im\{h_i\} \partial \Re\{h_j\}} & \frac{\partial^2 J(h_l, f_l)}{\partial \Im\{h_i\} \partial \Im\{h_j\}} & \frac{\partial^2 J(h_l, f_l)}{\partial \Im\{h_i\} \partial f_j} \\ \frac{\partial^2 J(h_l, f_l)}{\partial f_i \partial \Re\{h_j\}} & \frac{\partial^2 J(h_l, f_l)}{\partial f_i \partial \Im\{h_j\}} & \frac{\partial^2 J(h_l, f_l)}{\partial f_i \partial f_j} \end{pmatrix} \tag{B.4}$$

where $\Re\{\cdot\}$ and $\Im\{\cdot\}$ respectively denote the real part and imaginary part of the argument, and $\Delta h_i = \hat{h}_i - h_i$ and $\Delta f_i = \hat{f}_i - f_i$ are the perturbation of tne estimates caused by Gaussian noise $\vec{n}_i$.

In the following, we will calculate the elements of $\vec{n}_i$.

Using (B.1), one can obtain

$$\frac{\partial J(h_l,f_l)}{\partial \Re\{h_i\}} = \left[-\left(\vec{A}_i+\vec{B}_i f_i+\vec{C}_i f_i^2+\cdots\right)\right]^H \left[\vec{Y}-\sum_{l=1}^{L} h_l\left(\vec{A}_l+\vec{B}_l f_l+\vec{C}_l f_l^2+\cdots\right)\right] + \left[\vec{Y}-\sum_{l=1}^{L} h_l\left(\vec{A}_l+\vec{B}_l f_l+\vec{C}_l f_l^2+\cdots\right)\right]^H \left[-\left(\vec{A}_i+\vec{B}_i f_i+\vec{C}_i f_i^2+\cdots\right)\right]. \tag{B.5}$$

Recalling from (3.8), we have

$$\vec{w} = \vec{Y}-\sum_{l=1}^{L} h_l\left(\vec{A}_l+\vec{B}_l f_l+\vec{C}_l f_l^2+\cdots\right) \tag{B.6}$$

and can rewrite (B.5) as

$$\frac{\partial J(h_l,f_l)}{\partial \Re\{h_i\}} = \left[-\left(\vec{A}_i+\vec{B}_i f_i+\vec{C}_i f_i^2+\cdots\right)\right]^H \vec{w} + \vec{w}^H\left[-\left(\vec{A}_i+\vec{B}_i f_i+\vec{C}_i f_i^2+\cdots\right)\right]. \tag{B.7}$$

Using the assumption that f_l's are small, we can neglect the terms containing f_i in (B.7) and further simplify (B.7) as

$$\begin{aligned}\frac{\partial J(h_l,f_l)}{\partial \Re\{h_i\}} &\approx \left(-\vec{A}_i\right)^H \vec{w} + \vec{w}^H\left(-\vec{A}_i\right)\\ &= \left(-\vec{w}^H\vec{A}_i\right)^H + \left(-\vec{w}^H\vec{A}_i\right)\\ &= -2\Re\left\{\vec{w}^H\vec{A}_i\right\}.\end{aligned} \tag{B.8}$$

In a similar way, we can obtain

$$\frac{\partial J(h_l,f_l)}{\partial \Im\{h_i\}} = 2\Im\left\{\vec{w}^H\vec{A}_i\right\} \tag{B.9}$$

and

$$\frac{\partial J(h_l,f_l)}{\partial f_i} = -2\Re\left\{h_i\vec{w}^H\vec{B}_i\right\} \tag{B.10}$$

so that we obtain

$$\vec{n}_i = -\begin{pmatrix} \frac{\partial J(h_l,f_l)}{\partial \Re\{h_i\}} \\ \frac{\partial J(h_l,f_l)}{\partial \Im\{h_i\}} \\ \frac{\partial J(h_l,f_l)}{\partial f_i} \end{pmatrix} = 2\begin{pmatrix} \Re\left\{\vec{w}^H\vec{A}_i\right\} \\ -\Im\left\{\vec{w}^H\vec{A}_i\right\} \\ \Re\left\{\vec{h}_i\vec{w}^H\vec{B}_i\right\} \end{pmatrix}. \tag{B.11}$$

Based on the derivation above, it is straightforward to calculate the elements of R_{ij} in a similar way and obtain

$$R_{ij} = 2\begin{pmatrix} \Re\left\{\vec{A}_j^H\vec{A}_i\right\} & \Im\left\{\vec{A}_j^H\vec{A}_i\right\} & \Re\left\{h_j^*\vec{B}_j^H\vec{A}_i\right\} \\ -\Im\left\{\vec{A}_j^H\vec{A}_i\right\} & \Re\left\{\vec{A}_j^H\vec{A}_i\right\} & -\Im\left\{h_j^*\vec{B}_j^H\vec{A}_i\right\} \\ \Re\left\{h_i\vec{A}_j^H\vec{B}_i\right\} & \Im\left\{h_i\vec{A}_j^H\vec{B}_i\right\} & \Re\{h_ih_j^*\vec{B}_j^H\vec{B}_i\} \end{pmatrix} \tag{B.12}$$

where $\{\cdot\}^*$ denotes the complex conjugate of the argument. Applying (B.11) and (B.12) to (32) in [31], we can obtain (3.20).

Appendix C

Derivation of (3.30c)

From (3.7) and (3.8), we have

$$\vec{B}_l^H \vec{B}_l = \sum_{k=0}^{N-1} B_l^H[k] B_l[k]. \tag{C.1}$$

Applying (3.9b) to (C.1) and using the uncorrelation property of white training sequence , one obtains

$$\vec{B}_l^H \vec{B}_l = \frac{\pi^2 (N-1)^2}{N^2} \sum_{k=0}^{N-1} |X[k]|^2 + \frac{4\pi^2}{N^2} \sum_{k=0}^{N-1} \sum_{\substack{k'\neq k \\ k'=0}}^{N-1} \frac{1}{|1-\omega|^2} \left|X[k']\right|^2 \tag{C.2}$$

where $\omega = \exp\left(\frac{j2\pi(k'-k)}{N}\right)$. Assuming $|X[k]| = \frac{1}{\sqrt{N}}$, for $k = 0, 1, \ldots, N-1$, we can simplify (C.2) to be

$$\vec{B}_l^H \vec{B}_l = \frac{\pi^2 (N-1)^2}{N^2} + \frac{4\pi^2}{N^3} \sum_{k=0}^{N-1} \sum_{\substack{k'\neq k \\ k'=0}}^{N-1} \frac{1}{|1-\omega|^2}. \tag{C.3}$$

For the sake of simplicity, we denote

$$\begin{aligned} s[k] &\triangleq \sum_{\substack{k'\neq k \\ k'=0}}^{N-1} \frac{1}{|1-\omega|^2} \\ &= \sum_{\substack{k'\neq k \\ k'=0}}^{N-1} \frac{1}{\left|1 - e^{\frac{j2\pi(k'-k)}{N}}\right|^2}, \qquad k = 0, 1, \ldots, N-1 \end{aligned} \tag{C.4}$$

and rewrite (C.3) as

$$\vec{B}_l^H \vec{B}_l = \frac{\pi^2 (N-1)^2}{N^2} + \frac{4\pi^2}{N^3} \sum_{k=0}^{N-1} s[k]. \tag{C.5}$$

When N is fixed, exploiting the fact that $s[k]$ is constant for $k = 0, 1, \ldots, N-1$, one can obtain

$$s[k] \triangleq s = \sum_{n=1}^{N-1} \frac{1}{\left|1 - e^{\frac{j2\pi n}{N}}\right|^2}, \qquad k = 0, 1, \ldots, N-1 \tag{C.6}$$

and further simplify (C.5) to be

$$\vec{B}_l^H \vec{B}_l = \frac{\pi^2 (N-1)^2}{N^2} + \frac{4\pi^2}{N^2} s. \tag{C.7}$$

Using Parseval's relationship, one can show

$$\begin{aligned} s &= \sum_{n=1}^{N-1} \frac{1}{\left| 1 - e^{\frac{j2\pi n}{N}} \right|^2} \\ &= \frac{1}{4} \sum_{n=1}^{N-1} \frac{1}{\sin^2\left(\frac{n\pi}{N}\right)} \\ &= \frac{(N-1)(N+1)}{12}. \end{aligned} \tag{C.8}$$

Finally substituting of (C.8) into (C.7), one obtains (3.30c) as desired, i.e.,

$$\vec{B}_l^H \vec{B}_l = \frac{2\pi^2 (N-1)(2N-1)}{3N^2}. \tag{C.9}$$

Appendix D

Derivation of the Absolute Value for the Largest Eigenvalue of $M_\eta^{-1}N_\eta$

According to algebraic theory, the roots of a cubic equation

$$x^3+\alpha x^2+\beta x+\gamma=0 \tag{D.1}$$

are given by [37]

$$x=v-u-\frac{\alpha}{3} \tag{D.2}$$

where

$$u=\sqrt[3]{\frac{q}{2}\pm\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}} \tag{D.3a}$$

$$v=\frac{p}{3u} \tag{D.3b}$$

$$p=\beta-\frac{\alpha^2}{3} \tag{D.3c}$$

$$q=\gamma+\frac{2\alpha^3-9\alpha\beta}{27}. \tag{D.3d}$$

After some manipulations, one can calculate that the roots for the characteristic equation in (4.12) are given by

$$p=-\frac{3(1-\eta)+\eta^2k_l}{3}\eta^2k_l \tag{D.4a}$$

and

$$q = -\frac{1}{27}\left[9(1-\eta)+2\eta^2 k_l\right]\eta^4 k_l^2. \tag{D.4b}$$

To minimize the largest absolute value of the roots of (4.12), we let

$$\delta = \frac{q^2}{4}+\frac{p^3}{27} = 0 \tag{D.5}$$

and have three equal roots. After applying (D.4) into (D.5), one can show that η is required to satisfy

$$\eta^2 k_l - 4\eta + 4 = 0 \tag{D.6}$$

or

$$\eta = \frac{2(1-\sqrt{1-k_l})}{k_l}. \tag{D.7}$$

After substitution of η into u and v, it can be shown that

$$u = \sqrt[3]{\frac{q}{2}} = -\frac{2}{3}(1-\eta) \tag{D.8a}$$

$$v = \frac{2}{3}(1-\eta) \tag{D.8b}$$

$$x = v - u - \frac{\alpha}{3} = \alpha = 1-\eta. \tag{D.8c}$$

Thus, the absolute value for the largest eigenvalue of $M_\eta^{-1}N_\eta$ is

$$|\lambda| = |1-\eta| = \left|1-\frac{2(1-\sqrt{1-k_l})}{k_l}\right|. \tag{D.9}$$

Appendix E

Matlab Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                             %%
%%                          Main Function                                      %%
%%                                                                             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear;
%% Parameter Initialization %%

REPEAT = 2000;          % Number of repeat times, for purpose of average
Nfft = 256;             % Size of FFT
L = 3;                  % L-path channel
% fl = ones(1, L);      % Normalized Doppler frequency (1 by L vector)
% hl = zeros(1, L);     % Amplitude for the path (1 by L vector)
nl = 0 : (L - 1);       % Delay of the path (1 by L vector)
n_modu_type = 1;        % The type of modulation scheme
                        % 1=BPSK, 2=QPSK, 4=16QAM, 6=64QAM
Tx = 1;                 % Tx=1 => Transmitor; Tx=0 => Receiver
SNR = -10: 5: 40;
hl = [(0.2944+1.6236j)  (-1.3362-0.6918j)   (0.7143+0.8580j)];
fl = [0.02 0.04 0.06];
delta_h_2 = 0.01;       % Stoping threshold
delta_f_2 = 0.01;       % Stoping threshold
hl_initial = ones(2, L);
fl_initial = ones(2, L);
hl_initial = (1 + 1j) * hl_initial;  % Initial value
fl_initial = 0.05 * fl_initial;      % Initial value

%% For Algorithm 1 %%
hl_hat_ave_p2_A1 = zeros(length(SNR), L);
fl_hat_ave_p2_A1 = zeros(length(SNR), L);
hl_MSE_p2_A1 = zeros(length(SNR), L);
fl_MSE_p2_A1 = zeros(length(SNR), L);
hl_err_ave_p2_A1 = zeros(length(SNR), L);
```

```
fl_err_ave_p2_A1 = zeros(length(SNR), L);
iteration_p2_A1 = zeros(length(SNR), 1);

%%  For Algorithm 2  %%
hl_hat_ave_p2_A2 = zeros(length(SNR), L);
fl_hat_ave_p2_A2 = zeros(length(SNR), L);
hl_MSE_p2_A2 = zeros(length(SNR), L);
fl_MSE_p2_A2 = zeros(length(SNR), L);
hl_err_ave_p2_A2 = zeros(length(SNR), L);
fl_err_ave_p2_A2 = zeros(length(SNR), L);
iteration_p2_A2 = zeros(length(SNR), 1);

%%  For theoretical value  %%
nvar = zeros(length(SNR), 1);   % Noise variance
theoretic_fl = zeros(length(SNR), (L + 1));

for snr_index = 1 : length(SNR)
    %%  For Algorithm 1  %%
    hl_hat_temp_p2_A1 = zeros(1, L);
    fl_hat_temp_p2_A1 = zeros(1, L);
    hl_MSE_temp_p2_A1 = zeros(1, L);
    fl_MSE_temp_p2_A1 = zeros(1, L);
    hl_err_temp_p2_A1 = zeros(1, L);
    fl_err_temp_p2_A1 = zeros(1, L);
    Iteration_temp_p2_A1 = 0;

    %%  For Algorithm 2  %%
    hl_hat_temp_p2_A2 = zeros(1, L);
    fl_hat_temp_p2_A2 = zeros(1, L);
    hl_MSE_temp_p2_A2 = zeros(1, L);
    fl_MSE_temp_p2_A2 = zeros(1, L);
    hl_err_temp_p2_A2 = zeros(1, L);
    fl_err_temp_p2_A2 = zeros(1, L);
    Iteration_temp_p2_A2 = 0;

    %%  For theoretical value  %%
    nvar_temp = 0;   % Noise variance

    for repeat_index = 1 : REPEAT
        %%  Generate Data  %%
        Xbit = round(rand(1, (Nfft * n_modu_type)));
        %Xbit = randint(1, Nfft * n_modu_type);
```

```
    %%  Modulation (Gray Mapping)  %%
    X = mapper(Xbit, n_modu_type, Tx);
    X = X / sqrt((X * X'));  %%% Normalize X

    %%  Calculate Coefficiences  %%
    [Al, Bl, Cl, Dl] = calcu_coeff(Nfft, L, nl, X);

    %%  Calculate the output of FFT (Y[k]) using Eq. 4  %%
    [Y, nvar_i] = calcu_corrupted(Nfft,L,fl,hl,nl,X,SNR(snr_index));
    nvar_temp = nvar_temp + nvar_i;

    %%  Channel Estimation using Algorithm 1  %%
    temp_iter_p2 = 0;
    p = 2;       % The order of the Taylor expansion.
                 % It can not work properly when p=1.
    [hl_hat_ave_p2_A1(snr_index, :), fl_hat_ave_p2_A1(snr_index, :), iteration_p2_A1(snr_index)]...
        = channel_est_A1(Nfft, L, p, Al, Bl, Cl, Dl, Y, delta_h_2, delta_f_2, hl_initial, fl_initial);
    iteration_p2_A1(snr_index) = 1000 - iteration_p2_A1(snr_index);

    %%  Channel Estimation using Algorithm 2  %%
    [hl_hat_ave_p2_A2(snr_index, :), fl_hat_ave_p2_A2(snr_index, :), iteration_p2_A2(snr_index)]...
        = channel_est_A2(Nfft, L, p, Al, Bl, Cl, Dl, Y, delta_h_2, delta_f_2, hl_initial, fl_initial);
    iteration_p2_A2(snr_index) = 1000 - iteration_p2_A2(snr_index);

    %%  Calculate the MSE  %%
    %%  For Algorithm 1  %%
    hl_hat_temp_p2_A1=hl_hat_temp_p2_A1+hl_hat_ave_p2_A1(snr_index, :);
    fl_hat_temp_p2_A1=fl_hat_temp_p2_A1+fl_hat_ave_p2_A1(snr_index, :);
    hl_temp_p2_A1 = abs((hl_hat_ave_p2_A1(snr_index, :) - hl)) .^ 2;
    fl_temp_p2_A1 = abs((fl_hat_ave_p2_A1(snr_index, :) - fl)) .^ 2;
    hl_MSE_temp_p2_A1 = hl_MSE_temp_p2_A1 + hl_temp_p2_A1;
    fl_MSE_temp_p2_A1 = fl_MSE_temp_p2_A1 + fl_temp_p2_A1;
    Iteration_temp_p2_A1 = Iteration_temp_p2_A1 + iteration_p2_A1(snr_index);

    %%  For Algorithm 2  %%
    hl_hat_temp_p2_A2=hl_hat_temp_p2_A2+hl_hat_ave_p2_A2(snr_index, :);
    fl_hat_temp_p2_A2=fl_hat_temp_p2_A2+fl_hat_ave_p2_A2(snr_index, :);
    hl_temp_p2_A2 = abs((hl_hat_ave_p2_A2(snr_index, :) - hl)) .^ 2;
    fl_temp_p2_A2 = abs((fl_hat_ave_p2_A2(snr_index, :) - fl)) .^ 2;
    hl_MSE_temp_p2_A2 = hl_MSE_temp_p2_A2 + hl_temp_p2_A2;
    fl_MSE_temp_p2_A2 = fl_MSE_temp_p2_A2 + fl_temp_p2_A2;
    Iteration_temp_p2_A2 = Iteration_temp_p2_A2 + iteration_p2_A2(snr_index);
end
```

```
    %%  For Algorithm 1  %%
    hl_hat_ave_p2_A1(snr_index, :) = hl_hat_temp_p2_A1 / REPEAT;
    fl_hat_ave_p2_A1(snr_index, :) = fl_hat_temp_p2_A1 / REPEAT;
    hl_MSE_p2_A1(snr_index, :) = hl_MSE_temp_p2_A1 / REPEAT;
    fl_MSE_p2_A1(snr_index, :) = fl_MSE_temp_p2_A1 / REPEAT;
    hl_err_ave_p2_A1(snr_index, :) = hl_hat_ave_p2_A1(snr_index, :) - hl;
    fl_err_ave_p2_A1(snr_index, :) = fl_hat_ave_p2_A1(snr_index, :) - fl;
    iteration_p2_A1(snr_index) = Iteration_temp_p2_A1 / REPEAT;

    %%  For Algorithm 2  %%
    hl_hat_ave_p2_A2(snr_index, :) = hl_hat_temp_p2_A2 / REPEAT;
    fl_hat_ave_p2_A2(snr_index, :) = fl_hat_temp_p2_A2 / REPEAT;
    hl_MSE_p2_A2(snr_index, :) = hl_MSE_temp_p2_A2 / REPEAT;
    fl_MSE_p2_A2(snr_index, :) = fl_MSE_temp_p2_A2 / REPEAT;
    hl_err_ave_p2_A2(snr_index, :) = hl_hat_ave_p2_A2(snr_index, :) - hl;
    fl_err_ave_p2_A2(snr_index, :) = fl_hat_ave_p2_A2(snr_index, :) - fl;
    iteration_p2_A2(snr_index) = Iteration_temp_p2_A2 / REPEAT;

    %%  For theoretical value  %%
    nvar(snr_index) = nvar_temp / REPEAT;
end

%%  Plot Figure  %%
hl_temp_p2_A1 = zeros(length(SNR), 1);
fl_temp_p2_A1 = zeros(length(SNR), 1);
theoretic_hl = nvar * 5 /2;
hl_normal = abs(hl) .^ 2;
for i = 1 : L
    theoretic_fl(:, i) = nvar / hl_normal(i);
    theoretic_fl(:, (L + 1)) = theoretic_fl(:, (L + 1))+theoretic_fl(:, i);
end
theoretic_fl = theoretic_fl / 2 / pi^2 * 3;
theoretic_fl(:, (L + 1)) = theoretic_fl(:, (L + 1)) / L;
hl_temp_p2_A2 = zeros(length(SNR), 1);
fl_temp_p2_A2 = zeros(length(SNR), 1);
for i = 1 : L
    hl_temp_p2_A1 = hl_MSE_p2_A1(:, i) + hl_temp_p2_A1;
    hl_temp_p2_A2 = hl_MSE_p2_A2(:, i) + hl_temp_p2_A2;
end
hl_ave_MSE_p2_A1 = hl_temp_p2_A1' / L;
hl_ave_MSE_p2_A2 = hl_temp_p2_A2' / L;
hl_ave_MSE_p2_A1 = 10 * log10(hl_ave_MSE_p2_A1);
```

```
fl_ave_MSE_p2_A1 = 10 * log10(fl_MSE_p2_A1);
hl_ave_MSE_p2_A2 = 10 * log10(hl_ave_MSE_p2_A2);
fl_ave_MSE_p2_A2 = 10 * log10(fl_MSE_p2_A2);
theoretic_hl_final = 10 * log10(theoretic_hl);
theoretic_fl_final = 10 * log10(theoretic_fl);
for p = 2 : 3
    switch p
    case 2
       plot(SNR, hl_ave_MSE_p2_A1, '-ko');
       hold on
       plot(SNR, hl_ave_MSE_p2_A2, '-mp');
       hold on
       plot(SNR, theoretic_hl_final, '-bd');
       grid on;
       legend( 'Algorithm 1', 'Algorithm 2', 'Theoretical value' );
       xlabel( 'Average SNR (dB)', 'interpreter', 'latex' );
       ylabel( 'Average MSE of $\hat h_l$ (dB)', 'interpreter', 'latex' );
    end
end

for i = 1 : L
    figure
    plot(SNR, fl_ave_MSE_p2_A1(:, i), '-ko');
    hold on
    plot(SNR, fl_ave_MSE_p2_A2(:, i), '-mp');
    hold on
    plot(SNR, theoretic_fl_final(:, i), '-bd');
    grid on;
    legend( 'Algorithm 1', 'Algorithm 2', 'Theoretical value' );
    xlabel( 'Average SNR (dB)', 'interpreter', 'latex' );
    ylabel( 'MSE of $\hat f_l$ (dB)', 'interpreter', 'latex' );
end
x_d = SNR(3:end);
y_d_A1 = iteration_p2_A1(3:end);
y_d_A2 = iteration_p2_A2(3:end);
figure
plot(x_d, y_d_A1, '-ko');
hold on
plot(x_d, y_d_A2, '-mp');
hold on
grid on
legend( 'Algorithm 1', 'Algorithm 2' );
xlabel( 'Average SNR (dB)', 'interpreter', 'latex' );
```

```
ylabel( 'Average Number of Iterations', 'interpreter', 'latex' );

%   The End   %
```

```
function data_mapped = mapper( data_input, n_modu_type, Tx )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                             %%
%%       File: mapear.m                                                        %%
%%                                                                             %%
%%       Function: Map the input bits into the modulated symbols               %%
%%                 according to the modulation type decided by                 %%
%%                 the parameter 'n_modu_type' at the transmitor.              %%
%%                 Or demap the received symbols into bits at the              %%
%%                 receiver.                                                   %%
%%                                                                             %%
%%       Parameters: data_input => input bits                                  %%
%%                   n_modu_type => modulation type                            %%
%%                   (1=BPSK, 2=QPSK, 4=16QAM, 6=64QAM)                        %%
%%                   Tx => flag indicates the transmitor or receiver           %%
%%                   (1=transmitor, 0=receiver)                                %%
%%                                                                             %%
%%       Outputs: A matrix of the mapped symbols according to the              %%
%%                modulation scheme used at the transmitor side.               %%
%%                Or a vector of the demapped bits according to the            %%
%%                modulation scheme.                                           %%
%%                                                                             %%
%%       Notice: The length of 'data_input' must be a integer multiple         %%
%%               of 'n_modu_type' at the transmitor side.                      %%
%%                                                                             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%  Decide the constellation first  %%
[M, M1, M2, type_mapping, c] = parameters_constellation( n_modu_type );
alphabet = bit_symbol(M, type_mapping, M1, M2);
if n_modu_type ~= 1
    constellation_gray = alphabet(:,3) + 1j * alphabet(:,2);
else
    constellation_gray = [-1 1]';
end
l = length(data_input);

%%  Realize the mapping or demapping  %%
if Tx==1
    matrix_data = reshape (data_input, n_modu_type, l/n_modu_type);
    m_data_decimal = bi2de (matrix_data', 'left-msb');
    for i = 1 : (l/n_modu_type)
```

```
        v_data_decimal = m_data_decimal(i);
        v_encoded = genqammod(v_data_decimal,constellation_gray);
        output(:,i) = v_encoded;
    end
    data_mapped = c .* output;

elseif Tx==0
    data_normalized = data_input ./ c;
    for i = 1 : l
        v_data_mapped = data_normalized (i);
        v_decoded = genqamdemod(v_data_mapped,constellation_gray);
        data_decimal(:, i) = v_decoded;
        data_mapped = de2bi(data_decimal, n_modu_type, 'left-msb')';
        data_mapped = data_mapped(:)';
    end
end
```

```
function [M, M1, M2, type_mapping, c] = parameters_constellation( n_modu_type )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                            %%
%%       File: parameters_constellation.m                                     %%
%%                                                                            %%
%%       Function: Decide the constellation parameters according              %%
%%                 to modulation scheme to be used.                           %%
%%                                                                            %%
%%       Parameters: n_modu_type => type of modulation scheme                 %%
%%                   (1=BPSK, 2=QPSK, 4=16QAM, 6=64QAM)                       %%
%%                                                                            %%
%%       Outputs: The parameters which are used to call another               %%
%%                function to calculate the constellation using               %%
%%                Gray labeling.                                              %%
%%                                                                            %%
%%       Notice: Here we should pay attention to the parameter 'c'            %%
%%               We use it to normalize the constellation.                    %%
%%                                                                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch n_modu_type
    case 1                          % For BPSK (As a PAM)
        type_mapping = 'MPSK';      % Only take into account Aic
        M = 2;
        M1 = 0;
        M2 = 0;
        c = 1;
    case 2                          % For QPSK (Like 4-QAM)
        type_mapping = 'QAM';
        c = 1/sqrt(2);
    case 4                          % For 16-QAM
        type_mapping = 'QAM';
        c = 1/sqrt(10);
    case 6                          % For 64-QAM
        type_mapping = 'QAM';
        c = 1/sqrt(42);
end
if n_modu_type ~= 1
    M = 0;
    M1 = sqrt(2^n_modu_type);
    M2 = sqrt(2^n_modu_type);
end
```

```
function [alphabet] = bit_symbol( M, type, M1, M2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                             %%
%%     File: bit_symbol.m                                                      %%
%%                                                                             %%
%%     Function: Return the alphabet of the constellation according            %%
%%               to the modulation scheme used. Here the encoding              %%
%%               is done by Gray labeling                                      %%
%%                                                                             %%
%%     Parameters: M => total number of points in the constellation            %%
%%                  It should equal to 2^k, where k is the number of           %%
%%                  bits that are grouped together to form a symbol,           %%
%%                  except QAM, which can come as the product of the           %%
%%                  arguments M1 and M2.                                       %%
%%                  type => type of modulation scheme                          %%
%%                  PAM, MPSK, QAM(rectangular).                               %%
%%                  M1 & M2 => number of points on each axle                   %%
%%                                                                             %%
%%     Outputs: the alphabet of the constellation according                    %%
%%              to the modulation scheme used.                                 %%
%%                                                                             %%
%%     Notice: When using QAM, M is the product of M1 by M2.                   %%
%%             Here the encoding is done by Gray labelin                       %%
%%                                                                             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%  Block initialization  %%
if strcmp(type, 'QAM')
    k1 = ceil( log2(M1) );
    k2 = ceil( log2(M2) );
    M1 = 2 ^ k1;
    M2 = 2 ^ k2;
    M = M1 * M2;
    Aicd = zeros(1, k1);
    Aisd = zeros(1, k2);
    table1 = zeros(M1, 2);
    table2 = zeros(M2, 2);
    alphabet = zeros(M, 3);
    d1 = 0 : 1 : M1-1;
    d1 = d1';
    d2 = 0 : 1 : M2-1;
    d2 = d2';
```

```
        ind1 = bi2de(fliplr(gray2bi(fliplr(de2bi(d1)))));
        table1 = [d1, ind1+1];
        ind2 = bi2de(fliplr(gray2bi(fliplr(de2bi(d2)))));
        table2 = [d2, ind2+1];
else
        k = ceil(log2(M));
        M = 2 ^ k;
        Aicd = zeros(1, k);
        Aisd = zeros(1, k);
        table = zeros(M, 2);
        alphabet = zeros(M, 3);
        d = 0 : 1 : M-1;
        d = d';
        ind = bi2de(fliplr(gray2bi(fliplr(de2bi(d)))));
        table = [d, ind+1];
end

%%  Block computation  %%
if strcmp(type, 'PAM')
    Aicd = -(M-1) : 2 : M-1;
    Aisd = [];
    % Use the alphabet
    for i=1 : M
        index = find_index(i-1,table);
        alphabet(i, :) = [i-1, Aicd(index), 0];
    end
elseif strcmp(type, 'MPSK')
    angle = 0:2*pi/M:2*pi*(M-1)/M;
    Aicd = cos(angle);
    Aisd = sin(angle);
    % Use the alphabet
    for i=1 : M
        index = find_index(i-1,table);
        alphabet(i, :) = [i-1, Aicd(index), Aisd(index)];
    end
elseif strcmp(type, 'QAM')
    Aicd = -(M1-1) : 2 : M1-1;
    Aisd = (M2-1) : -2 : -(M2-1);
    % Use the alphabet
    for i=1 : M1
        for j=1 : M2
            index1 = find_index(i-1,table1);
            index2 = find_index(j-1,table2);
```

```
            l = i + M1 * (j-1);
            alphabet(l, :) = [l-1, Aicd(index1), Aisd(index2)];
        end
    end
end
```

```
function b = gray2bi( g )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                              %%
%%   GRAY2BI converts Gray encoded sequence into the binary                    %%
%%   sequence. It is assumed that the most significant bit is                   %%
%%   the left hand side bit.                                                    %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % copy the msb:
    b(:,1) = g(:,1);

    for i = 2:size(g,2),
        b(:,i) = xor( b(:,i-1), g(:,i) );
    end

return;
```

```
function indice = find_index(num,table)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                              %%
%%        File: find_index.m                                                    %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dim = size(table);
num_col = dim(1);

for i=1:num_col
    if num==table(i,1);
        indice = table(i,2);
        return
    end
end
```

```
function [Al, Bl, Cl, Dl] = calcu_coeff(Nfft, L, nl, X)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                             %%
%%       File: calcu_coeff.m                                                   %%
%%                                                                             %%
%%       Function: Calculate the coefficients of Equation 8.                   %%
%%                 Then output Al[k], Bl[k], Cl[k] and Dl[k].                  %%
%%                                                                             %%
%%       Parameters: Nfft => Size of FFT                                       %%
%%                   L => L-path channel                                       %%
%%                   nl => Delay of the path                                   %%
%%                   X => Training sequence                                    %%
%%                                                                             %%
%%       Outputs: Al, Bl, Cl and Dl                                            %%
%%                                                                             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Al = zeros(L, Nfft);
for l = 1 : L
    for k = 1 : Nfft
        Al(l, k) = X(k) * exp(-1j * 2 * pi * (k - 1) * nl(l) / Nfft);

        w = exp(1j * 2 * pi * ((1 : Nfft) - k + 1e-13) / Nfft);
        temp = X .* exp(-1j * 2 * pi * (0 : (Nfft - 1)) * nl(l) / Nfft)...
            .* (-2 * 1j * pi / Nfft ./ (1 - w));
        temp(k) = Al(l, k) * 1j * pi * (Nfft - 1) / Nfft;
        Bl(l, k) = sum(temp);

        temp = X .* exp(-1j * 2 * pi * (0 : (Nfft - 1)) * nl(l) / Nfft)...
            .* (2 * pi^2 .* (1 - 2 * w ./ (w-1) / Nfft) ./ (1-w) / Nfft);
        temp(k) = -Al(l, k) * pi^2 * (2 / 3 - 2 / 3 / Nfft / Nfft - (Nfft-1) / Nfft / Nfft);
        Cl(l,k) = sum(temp);

        temp = X .* exp(-1j * 2 * pi * (0 : (Nfft - 1)) * nl(l) / Nfft)...
            .* (1j * 4 * pi^3 * (Nfft^2 * (1-w).^2 + 3 * w .* (Nfft + 1 ...
            + w + Nfft * w)) / 3 / Nfft^3 / (1-w).^3);
        temp(k) = -Al(l, k) * 1j * pi^3 * (Nfft-1)^2 / 3 / Nfft^2;
        Dl(l, k) = sum(temp);
    end
end
```

```
function [Y, nvar] = calcu_corrupted(Nfft, L, fl, hl, nl, X, SNR)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%        File: calcu_corrupted.m                                           %%
%%                                                                          %%
%%        Function: Calculate the corrupted signal, undergoing fading,      %%
%%                  adding AWGN according to Equation (4).                  %%
%%                                                                          %%
%%        Parameters: Nfft => Size of FFT                                   %%
%%                    L => L-path channel                                   %%
%%                    fl => Normalized Doppler frequency                    %%
%%                    hl => Complex valued channel gain                     %%
%%                    nl => Delay of the path                               %%
%%                    X => Training sequence                                %%
%%                    SNR => SNR value                                      %%
%%                                                                          %%
%%        Outputs: The corrupted signal at the receiver side.               %%
%%                                                                          %%
%%        Notice: We do not add AWGN if we just want the corrupted          %%
%%                signal for calculating the equivalent SNR to              %%
%%                quantify the approximation. Then we only have             %%
%%                6 input arguments without SNR.                            %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch nargin
    case 6
        Y = zeros(1, Nfft);
        R = zeros(L, 2*Nfft-1);
        kk = (1 - Nfft) : (Nfft - 1);

        % Calculate the received signal without AWGN
        for l = 1 : L
            R(l, :) = (1 - exp(j * 2 * pi * fl(l))) ./ (1 - exp(j * 2 ...
                * pi .* (kk + fl(l) + 1e-13) / Nfft)) / Nfft;
        end
        for k = 1 : Nfft
            for l = 1 : L
                for k_ = 1 : Nfft
                    Y(k) = Y(k) + exp(-j * 2 * pi * (k_-1) * nl(l) / ...
                        Nfft) * X(k_) * hl(l) * R(l, (Nfft-k+k_));
                end
```

```
            end
        end
        nvar = 0;
    case 7
        SNR_l = 10 ^ (SNR / 10);
        Y = zeros(1, Nfft);
        R = zeros(L, 2*Nfft-1);
        kk = (1 - Nfft) : (Nfft - 1);

        % Calculate the received signal without AWGN
        for l = 1 : L
            R(l, :) = (1 - exp(j * 2 * pi * fl(l))) ./ (1 - exp(j * 2 ...
                * pi .* (kk + fl(l) + 1e-13) / Nfft)) / Nfft;
        end
        for k = 1 : Nfft
            for l = 1 : L
                for k_ = 1 : Nfft
                    Y(k) = Y(k) + exp(-j * 2 * pi * (k_-1) * nl(l) / ...
                        Nfft) * X(k_) * hl(l) * R(l, (Nfft-k+k_));
                end
            end
        end

        % Add AWGN to the signal
        Es = mean(abs(Y) .^ 2);
        nvar = Es / SNR_l;
        AWGNoise = sqrt(nvar / 2) .* (randn(1, Nfft) + 1j.*randn(1, Nfft));
        Y = Y + AWGNoise;
end
```

```
function [hl_hat, fl_hat, count] = channel_est_A1(Nfft, L, p, Al, Bl, ...
    Cl, Dl, Y, delta_h, delta_f, hl_initial, fl_initial)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                              %%
%% File: channel_est_A1.m                                                       %%
%%                                                                              %%
%% Function: Estimate the channel parameters (hl & fl) using Algorithm 1.%%
%%                                                                              %%
%% Parameters:                                                                  %%
%%             Nfft => Size of FFT                                              %%
%%             L => L-path channel                                              %%
%%             p => the order of Taylor Series Expansion                        %%
%%             Al, Bl, Cl, Dl => the coefficients of the approximation          %%
%%             Y => the received signal                                         %%
%%             delta_h => the threshold of hl (stops the iteration)             %%
%%             delta_f => the threshold of fl                                   %%
%%             hl_initial => the initial value of estimate 'hl_hat'             %%
%%             fl_initial => the initial value of estimate 'fl_hat'             %%
%%                                                                              %%
%% Outputs: The esimates 'hl_hat', 'fl_hat' and the iteration 'count'           %%
%%                                                                              %%
%% Notice: The order of accuracy depends on Nfft and p. We have                 %%
%%         different accuracy for different path. The iteration time            %%
%%         depends on the initial value and the threshold.                      %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Initialize hl and fl %%%
if nargin == 10
    YYY = Y.';
    AAA = Al.';
    BBB = Bl.';
    temp_H = inv(Al * AAA) * Al * YYY;
    temp_F = inv(Bl * BBB) * Bl * (YYY - AAA * temp_H) ./ temp_H;
    hl_initial = [temp_H temp_H];
    fl_initial = [temp_F temp_F];
    hl_initial = hl_initial.';
    fl_initial = fl_initial.';
end
count = 0;
hl_hat = zeros(1, L);
fl_hat = zeros(1, L);
```

```
switch p
    case 2    % second order approximation
        iteration = 1000;
        while (iteration)
            for l = 1 : L
                Alpha_h = zeros(1, Nfft);
                Alpha_f = zeros(1, Nfft);
                if l == 1
                    for k = 2 : L
                        Alpha_h = Alpha_h + hl_initial(2, k) .* (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                elseif l == L
                    for k = 1 : L - 1
                        Alpha_h = Alpha_h + hl_initial(2, k) .* (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                else
                    for k = 1 : l - 1
                        Alpha_h = Alpha_h + hl_initial(2, k) .* (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                    for k = l + 1 : L
                        Alpha_h = Alpha_h + hl_initial(2, k) .*  (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                end
                % update hl
                Alpha_h = Y - Alpha_h;
                Beta_h = Al(l, :) + fl_initial(2, l) .* Bl(l, :) + fl_initial(2, l)^2 .* Cl(l, :);
                temp = hl_initial(2, l);
                hl_initial(2, l) = Beta_h * Alpha_h' / (Beta_h * Beta_h');
                hl_initial(2, l) = conj(hl_initial(2, l));
                hl_initial(1, l) = temp;
                % update fl
                Alpha_f = Alpha_h - hl_initial(2, l) .* Al(l, :);
                Beta_f = hl_initial(2, l) .* Bl(l, :);
                Eta_f = hl_initial(2, l) .* Cl(l, :);
                a = Eta_f * Eta_f' * 4;
                b = (Beta_f * Eta_f' + Eta_f * Beta_f') * 3;
                c = (Beta_f * Beta_f' - Alpha_f * Eta_f' - Eta_f * Alpha_f') * 2;
                d = -(Alpha_f * Beta_f' + Beta_f * Alpha_f');
                ppp = [a b c d];
```

```
            rrr = roots(ppp);
            temp = fl_initial(2, l);
            fl_initial(2, l) = rrr(length(rrr));
            fl_initial(1, l) = temp;
        end
        if (max(abs(hl_initial(2, :) - hl_initial(1, :)) ./ abs(hl_initial(1, :))) < delta_h)...
                && (max(abs(fl_initial(2, :) - fl_initial(1, :)) ./ abs(fl_initial(1, :))) < delta_f)
            count = iteration;
            iteration = 0;
        else
            iteration = iteration - 1;
        end
    end
    hl_hat = hl_initial(2, :);
    fl_hat = fl_initial(2, :);

case 3   % third order approximation
    iteration = 1000;
    while (iteration)
        for l = 1 : L
            Alpha_h = zeros(1, Nfft);
            Alpha_f = zeros(1, Nfft);
            if l == 1
                for k = 2 : L
                    Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                        + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
                end
            elseif l == L
                for k = 1 : L - 1
                    Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                        + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
                end
            else
                for k = 1 : l - 1
                    Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                        + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
                end
                for k = l + 1 : L
                    Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                        + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
                end
            end
            % update hl
```

```
            Alpha_h = Y - Alpha_h;
            Beta_h = Al(l, :) + fl_initial(2, l) .* Bl(l, :) + fl_initial(2, l)^2 .* Cl(l, :)...
                + fl_initial(2, l)^3 .* Dl(l, :);
            temp = hl_initial(2, l);
            hl_initial(2, l) = Beta_h * Alpha_h' / (Beta_h * Beta_h');
            hl_initial(2, l) = conj(hl_initial(2, l));
            hl_initial(1, l) = temp;
            % update fl
            Alpha_f = Alpha_h - hl_initial(2, l) .* Al(l, :);
            Beta_f = hl_initial(2, l) .* Bl(l, :);
            Eta_f = hl_initial(2, l) .* Cl(l, :);
            Theta_f = hl_initial(2, l) .* Dl(l, :);
            a = Theta_f * Theta_f' * 6;
            b = (Eta_f * Theta_f' + Theta_f * Eta_f') * 5;
            c = (Beta_f * Theta_f' + Eta_f * Eta_f' + Theta_f * Beta_f') * 4;
            d = (Beta_f * Eta_f' + Eta_f * Beta_f' - Alpha_f * Theta_f' - Theta_f * Alpha_f') * 3;
            e = (Beta_f * Beta_f' - Alpha_f * Eta_f' - Eta_f * Alpha_f') * 2;
            f = -(Alpha_f * Beta_f' + Beta_f * Alpha_f');
            ppp = [a b c d e f];
            rrr = roots(ppp);
            temp = fl_initial(2, l);
            fl_initial(2, l) = rrr(length(rrr));
            fl_initial(1, l) = temp;
        end
        if (max(abs(hl_initial(2, :) - hl_initial(1, :)) ./ abs(hl_initial(1, :))) < delta_h)...
                && (max(abs(fl_initial(2, :) - fl_initial(1, :)) ./ abs(fl_initial(1, :))) < delta_f)
            count = iteration;
            iteration = 0;
        else
            iteration = iteration - 1;
        end
    end
    hl_hat = hl_initial(2, :);
    fl_hat = fl_initial(2, :);
end
```

```
function [hl_hat, fl_hat, count] = channel_est_A2(Nfft, L, p, Al, Bl, ...
    Cl, Dl, Y, delta_h, delta_f, hl_initial, fl_initial)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                              %%
%% File: channel_est_A2.m                                                       %%
%%                                                                              %%
%% Function: Estimate the channel parameters (hl & fl) using Algorithm 2.%%
%%                                                                              %%
%% Parameters:                                                                  %%
%%             Nfft => Size of FFT                                              %%
%%             L => L-path channel                                              %%
%%             p => the order of Taylor Series Expansion                        %%
%%             Al, Bl, Cl, Dl => the coefficients of the approximation          %%
%%             Y => the received signal                                         %%
%%             delta_h => the threshold of hl (stops the iteration)             %%
%%             delta_f => the threshold of fl                                   %%
%%             hl_initial => the initial value of estimate 'hl_hat'             %%
%%             fl_initial => the initial value of estimate 'fl_hat'             %%
%%                                                                              %%
%% Outputs: The esimates 'hl_hat', 'fl_hat' and the iteration 'count'           %%
%%                                                                              %%
%% Notice: The order of accuracy depends on Nfft and p. We have                 %%
%%         different accuracy for different path. The iteration time            %%
%%         depends on the initial value and the threshold.                      %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Initialize hl and fl %%%
if nargin == 10
    YYY = Y.';
    AAA = Al.';
    BBB = Bl.';
    temp_H = inv(Al * AAA) * Al * YYY;
    temp_F = inv(Bl * BBB) * Bl * (YYY - AAA * temp_H) ./ temp_H;
    hl_initial = [temp_H temp_H];
    fl_initial = [temp_F temp_F];
    hl_initial = hl_initial.'
    fl_initial = fl_initial.'
end
eta = 4 / 3;
count = 0;
hl_hat = zeros(1, L);
```

```
fl_hat = zeros(1, L);

switch p
    case 2    % second order approximation
        iteration = 1000;
        while (iteration)
            for l = 1 : L
                Alpha_h = zeros(1, Nfft);
                Alpha_f = zeros(1, Nfft);
                if l == 1
                    for k = 2 : L
                        Alpha_h = Alpha_h + hl_initial(2, k) .* (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                elseif l == L
                    for k = 1 : L - 1
                        Alpha_h = Alpha_h + hl_initial(2, k) .* (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                else
                    for k = 1 : l - 1
                        Alpha_h = Alpha_h + hl_initial(2, k) .* (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                    for k = l + 1 : L
                        Alpha_h = Alpha_h + hl_initial(2, k) .* (Al(k, :)...
                            + fl_initial(2, k) .* Bl(k, :) + fl_initial(2, k)^2 .* Cl(k, :));
                    end
                end
                % update hl
                Alpha_h = Y - Alpha_h;
                Beta_h = Al(l, :) + fl_initial(2, l) .* Bl(l, :) + fl_initial(2, l)^2 .* Cl(l, :);

                temp1 = Beta_h * Alpha_h' / (Beta_h * Beta_h');  % h(i+1)
                temp1 = conj(temp1);
                temp2 = eta * temp1 + (1 - eta) * hl_initial(2, l);
                temp = hl_initial(2, l);
                hl_initial(1, l) = temp;
                hl_initial(2, l) = temp2;
                % update fl
                Alpha_f = Alpha_h - hl_initial(2, l) .* Al(l, :);
                Beta_f = hl_initial(2, l) .* Bl(l, :);
                Eta_f = hl_initial(2, l) .* Cl(l, :);
```

```
            a = Eta_f * Eta_f' * 4;
            b = (Beta_f * Eta_f' + Eta_f * Beta_f') * 3;
            c = (Beta_f * Beta_f' - Alpha_f * Eta_f' - Eta_f * Alpha_f') * 2;
            d = -(Alpha_f * Beta_f' + Beta_f * Alpha_f');
            ppp = [a b c d];
            rrr = roots(ppp);
            temp1 = rrr(length(rrr));    %   f(i+1)
            temp2 = eta * temp1 + (1 - eta) * hl_initial(2, l);
            temp = fl_initial(2, l);
            fl_initial(1, l) = temp;
            fl_initial(2, l) = temp1;
        end
        if (max(abs(hl_initial(2, :) - hl_initial(1, :)) ./ abs(hl_initial(1, :))) < delta_h)...
                && (max(abs(fl_initial(2, :) - fl_initial(1, :)) ./ abs(fl_initial(1, :))) < delta_f)
            count = iteration;
            iteration = 0;
        else
            iteration = iteration - 1;
        end
    end
    hl_hat = hl_initial(2, :);
    fl_hat = fl_initial(2, :);

case 3  % third order approximation
    iteration = 1000;
    while (iteration)
        for l = 1 : L
            Alpha_h = zeros(1, Nfft);
            Alpha_f = zeros(1, Nfft);
            if l == 1
                for k = 2 : L
                    Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                        + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
                end
            elseif l == L
                for k = 1 : L - 1
                    Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                        + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
                end
            else
                for k = 1 : l - 1
                    Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                    + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
```

```
        end
        for k = l + 1 : L
            Alpha_h = Alpha_h + hl_initial(2,k) .* (Al(k,:) + fl_initial(2,k) .* Bl(k,:)...
                + fl_initial(2, k)^2 .* Cl(k, :) + fl_initial(2, k)^3 .* Dl(k, :));
        end
    end
    % update hl
    Alpha_h = Y - Alpha_h;
    Beta_h = Al(l, :) + fl_initial(2, l) .* Bl(l, :) + fl_initial(2, l)^2 .* Cl(l, :)...
        + fl_initial(2, l)^3 .* Dl(l, :);

    temp1 = Beta_h * Alpha_h' / (Beta_h * Beta_h');  % h(i+1)
    temp1 = conj(temp1);
    temp2 = eta * temp1 + (1 - eta) * hl_initial(2, l);
    temp = hl_initial(2, l);
    hl_initial(1, l) = temp;
    hl_initial(2, l) = temp2;
    % update fl
    Alpha_f = Alpha_h - hl_initial(2, l) .* Al(l, :);
    Beta_f = hl_initial(2, l) .* Bl(l, :);
    Eta_f = hl_initial(2, l) .* Cl(l, :);
    Theta_f = hl_initial(2, l) .* Dl(l, :);
    a = Theta_f * Theta_f' * 6;
    b = (Eta_f * Theta_f' + Theta_f * Eta_f') * 5;
    c = (Beta_f * Theta_f' + Eta_f * Eta_f' + Theta_f * Beta_f') * 4;
    d = (Beta_f * Eta_f' + Eta_f * Beta_f' - Alpha_f * Theta_f' - Theta_f * Alpha_f') * 3;
    e = (Beta_f * Beta_f' - Alpha_f * Eta_f' - Eta_f * Alpha_f') * 2;
    f = -(Alpha_f * Beta_f' + Beta_f * Alpha_f');
    ppp = [a b c d e f];
    rrr = roots(ppp);
    temp1 = rrr(length(rrr));    %   f(i+1)
    temp2 = eta * temp1 + (1 - eta) * hl_initial(2, l);
    temp = fl_initial(2, l);
    fl_initial(1, l) = temp;
    fl_initial(2, l) = temp1;
end
if (max(abs(hl_initial(2, :) - hl_initial(1, :)) ./ abs(hl_initial(1, :))) < delta_h)...
        && (max(abs(fl_initial(2, :) - fl_initial(1, :)) ./ abs(fl_initial(1, :))) < delta_f)
    count = iteration;
    iteration = 0;
else
    iteration = iteration - 1;
end
```

```
        end
        hl_hat = hl_initial(2, :);
        fl_hat = fl_initial(2, :);
end
```