Decision Weighted Adaptive Algorithms with Applications to Wireless Channel Estimation

by

Shane Martin Haas

B.S. Math The University of Kansas, Lawrence, 1998

B.S.E.E. The University of Kansas, Lawrence, 1998

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

_____

Professor in Charge

_____

_____

Committee Members (2)

_____

Date of Acceptance

Acknowledgments

I cannot begin to thank the countless people that have influenced my life and education. I apologize in advance to all the people I forget to mention. I want to thank my advisor Glenn Prescott for your sound advice and the hundreds of hours you spent helping me with homework problems and this thesis. Thank you for always making time for me and for your infinite patience. I want to thank the other members of my committee, Dave Petr and Joe Evans, for their time and direction. I cannot imagine how boring and time consuming reading one thesis after another is. I just hope you can get through this one with as little pain and boredom as possible (except for Dr. Petr who I hope suffers just one tenth the amount of pain (any more would be too cruel of a wish) he inflicted on us with his EECS 360 homework assignments on convolution...although admittedly I did need them for this thesis).

I would also like to thank Bozenna Pasik-Duncan and Tyrone Duncan who have been so generous with their time to help me on the sticky details of the proofs presented in this thesis. I cannot imagine what my experience at KU would have been like if I had not taken calculus from Dr. Pasik-Duncan my freshman year. I thank you for your never-ending support for me and your dedication to my education.

I certainly cannot forget the many long hours spent in the ITTC wireless lab with Craig Sparks, Rich Killoy, and Ken Filardo, where many of the ideas presented in this thesis developed. Thanks for the all the enlightening conversations, the frustrating hours of troubleshooting, and the rewarding satisfaction of demonstrating the RDRN system in Washington D.C. I would also like to thank Roel Jonkman who most graciously let me use his thesis template. I also thank Tom Mulinazzi for his support, without which, my timely graduation would not have been possible.

Lastly, I want to thank my family and friends. Thank you Kassy for your support and uncanny ability to always make me laugh (also, thank you for the sandwiches you brought me so late at night; I was really hungry). I want to thank my wonderful family for their unfaltering support and love. Thank you Mom for always being there for me. And Dad, I thank you for instilling in me the desire to always keep learning and to never be satisfied with what I already know. Thank you for your advice and direction. I will always miss you.

Abstract

This thesis proposes and studies novel modifications to the least mean squares (LMS) and weighted recursive least squares (WRLS or weighted RLS) adaptive algorithms to estimate the impulse response of a wireless communications channel blindly without the aid of a training or probe sequence. Specifically, we use knowledge of receiver decision quality to weight the LMS and WRLS estimators to increase their robustness to hard decision errors and channel noise. We propose two classes of these decision weighted algorithms: 1) soft decision weighted, where algorithm weights are a function of receiver soft decisions; and 2) ideal decision weighted, where algorithm weights are a function of decision error knowledge. We compared the performance of these decision weighted estimators to their non-weighted blind and non-blind counterparts through simulations over free-space propagation three path (Rummler) and mobile radio channel models. Our results show that the decision weighted LMS (DWLMS) has significant advantages over ordinary LMS in environments with low signal-to-noise ratios (SNR) and high symbol error rates (SER). The decision weighted recursive least squares, however, had mixed results. The soft decision weighted RLS (SDWRLS) had poorer performance than other WRLS algorithms, but the ideal decision weighted RLS (IDWRLS) had similar performance to other WRLS. To improve the performance of the SDWRLS, we also propose and simulate modifications to its estimator update structure. These modifications allow the SDWRLS to perform similar to the soft decision weighted LMS.

Contents

List of Figures

Chapter 1

Introduction

1.1 What is an Adaptive Algorithm?

In its most general form, an adaptive algorithm is a procedure that changes its parameters as it gains more knowledge of its possibly changing environment. Preferably the algorithm will change its parameters in a fashion that optimizes some criteria such as the mean squared difference between two given signals.

The simplest form of an adaptive algorithm is illustrated in Figure 1.1. In this system, a human-being adjusts the parameters of an adjustable filter to try to minimize or maximum some aspect of the system's performance. In this manual approach the adaptive algorithm is implemented in the person's brain based on observations through their five senses. A more automated approach is shown in Figure 1.2. In this configuration, a computer or signal processor performs the adjustments based on an adaptive algorithm and a set of input signals. This thesis describes two such classes of algorithms, least squares (LS) and least mean squares (LMS), and modifications to them to provide better performance in the application of system identification.

1.2 What is System Identification?

In its broadest definition, system identification is the characterization of the input-output relationship of a system. If the system is linear and time-invariant, the problem reduces to finding the system's impulse response, or equivalently, its transfer function. If this transfer function is rational, i.e. it can be written as the ratio of two polynomials, then the problem becomes the estimation of the polynomial coefficients.

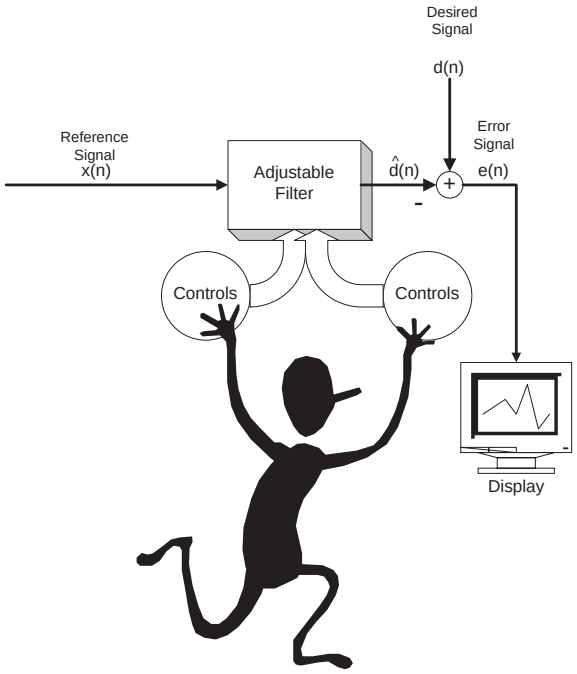

Figure 1.1: Illustration of a manual adaptive algorithm

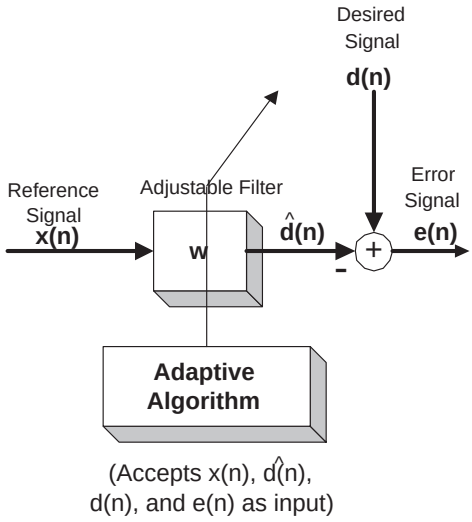

Figure 1.2: Illustration of an automatic adaptive algorithm

Adaptive algorithms are commonly[1] used to perform system identification ([34], pg. 7), [8], ([22], Ch. 9-10). Figure 1.3 illustrates how adaptive algorithms and adjustable filters are applied to the problem of system identification. For this application, the reference signal is the input to the system and the desired signal is the output of the system. The algorithm usually tries to adjust the adaptive filter in such a way as to drive the mean squared error signal to zero. If this mean squared error signal is zero, then the adjustable filter, whose parameters are completely known, replicates the system in question whose parameters are unknown. In other words, the parameters of the adjustable model or filter provide a good estimate of the parameters of the unknown system.

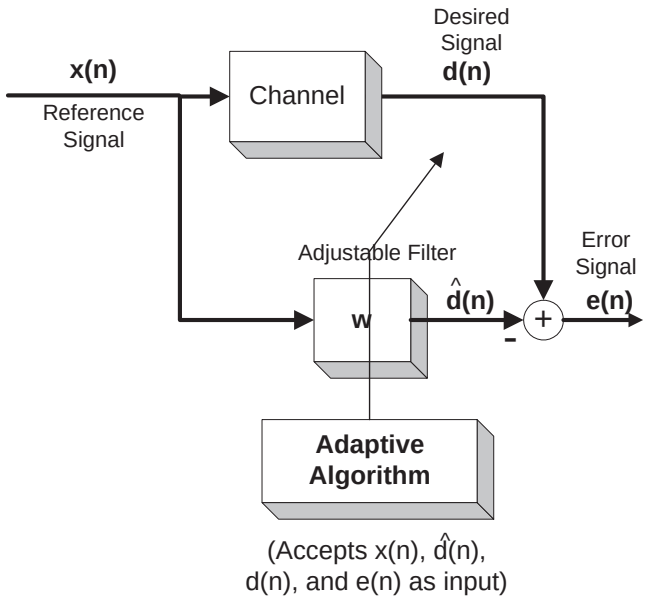

Figure 1.3: An adaptive filter applied to the problem of system identification

1.3 Benefits of Channel Estimation

The problem of system identification when applied to telecommunication media or channels is often called channel estimation [40], [6], [3]. In general, more accurate knowledge of a communication system's environment leads to better performance. Specifically, RAKE receivers [27] use knowledge of the channel impulse response to adapt a matched filter to the response

[1] Other system identification methods include sounding or cross-correlation techniques ([34], pg. 7), [39].

of the pulse shaping filters and the channel. In addition, maximum likelihood sequence detector receivers [25] rely on channel estimates to detect a sequence of symbols in such a way as to minimize the probability of making an error. Furthermore, adaptive channel equalizers use channel estimates to implement the inverse of the channel [23], [7], ([28], pg. 265). Other benefits of channel estimation include the possibility of a communication system to adaptively change its data rate or modulation scheme to reduce the intersymbol interference caused by transmitting data too quickly through the band-limited channel.

1.4 Training Sequence Versus Blind Estimation

In most communication systems, considerable distances separate the transmitter and receiver; therefore, the estimator at the receiver does not have practical access to the transmitted signal that enters the channel. Blind algorithms are those that do not rely upon this knowledge of the transmitted signal. A popular class of blind algorithms are decision directed or decision feedback algorithms. These algorithms rely upon the demodulated and detected sequence at the receiver to reconstruct the transmitted signal. Unlike in Figure 1.3, they then use this reconstructed signal as if it were the actual transmitted signal. An obvious downfall of these methods is that a decision or bit error at the receiver will cause the construction of an incorrect transmitted signal. In the case of channel estimation, this decision error will introduce a bias in the channel estimate, making it less accurate. In this thesis, we propose modifications to standard adaptive algorithms to make them less susceptible to these decision errors.

Although the receiver might not have direct access to the transmitted signal, if the transmitter periodically sends a known training or probe sequence, the receiver can use this training sequence to reconstruct the transmitted waveform. While this method will produce more accurate estimates of the channel during the training interval, these estimates become out of date between these intervals, unlike the continually updated estimates of the blind techniques. Another drawback of training sequence methods is that the training sequence occupies valuable bandwidth, reducing the throughput of the communication system. For example, training sequences account for 22% of GSM's [6] and 9% of IS-54's [11] total bandwidth.

1.5 Thesis Overview

This thesis is structured as follows. Chapter 2 describes the theoretical development of the channel estimation problem. It starts by formulating the general channel estimation problem

then proceeds to describe multiple phase shift keying (MPSK), the modulation technique used in later simulations. We then characterize wireless communication channels as linear systems. The next section provides background on representing real bandpass systems as complex low-pass systems using Hilbert transformation techniques. Next, we explore properties of two popular adaptive algorithms, linear estimators and least mean squares (LMS) estimators. We consider both training and decision directed applications, making modifications to the decision directed techniques to reduce their susceptibility to decision errors and noise. Chapter 3 describes the methodology used to simulate the algorithms described in Chapter 2. Chapter 4 presents the results of the simulations, and Chapter 5 discusses each algorithm's performance.

Chapter 2

Theoretical Development

This chapter develops the theory necessary to understand the channel estimation problem. It starts by formulating the problem of channel estimation, then proceeds to describe multiple phase shift keying (MPSK) modulation. Next, we show how to model a wireless channel as a transversal or tap-delay line filter. We then describe how to model a real bandpass channel as a complex low-pass channel, a conversion that greatly eases simulation. Finally, we introduce linear and least mean squares estimators and derive some of their properties.

2.1 Problem Formulation

A conceptual block diagram illustrating the wireless communication channel estimation is shown in Figure 2.1. A symbol source generates symbols (information) that pass through a modulator to produce the transmitted signal. The transmitted signal passes through the channel where it becomes distorted and corrupted with noise. The demodulator reduces the received signal to a single sample for each symbol interval. Based on this sample, the decision block decides which symbol the transmitter sent. The symbols then flow into the symbol sink, a symbolic representation of the remaining information path.

The goal of the estimator blocks is to estimate the impulse response of the channel. The training sequence estimator has the luxury of knowing the transmitted signal while the decision directed estimator must reconstruct it from the detected symbols (hard decisions). The decision weighted estimator is a special case of the decision directed estimator. In its non-realizable ideal form, decision weighted estimators use knowledge of decision errors to reduce their vulnerability to these errors. In its implementable form, they rely upon soft decision information to reduce this vulnerability.

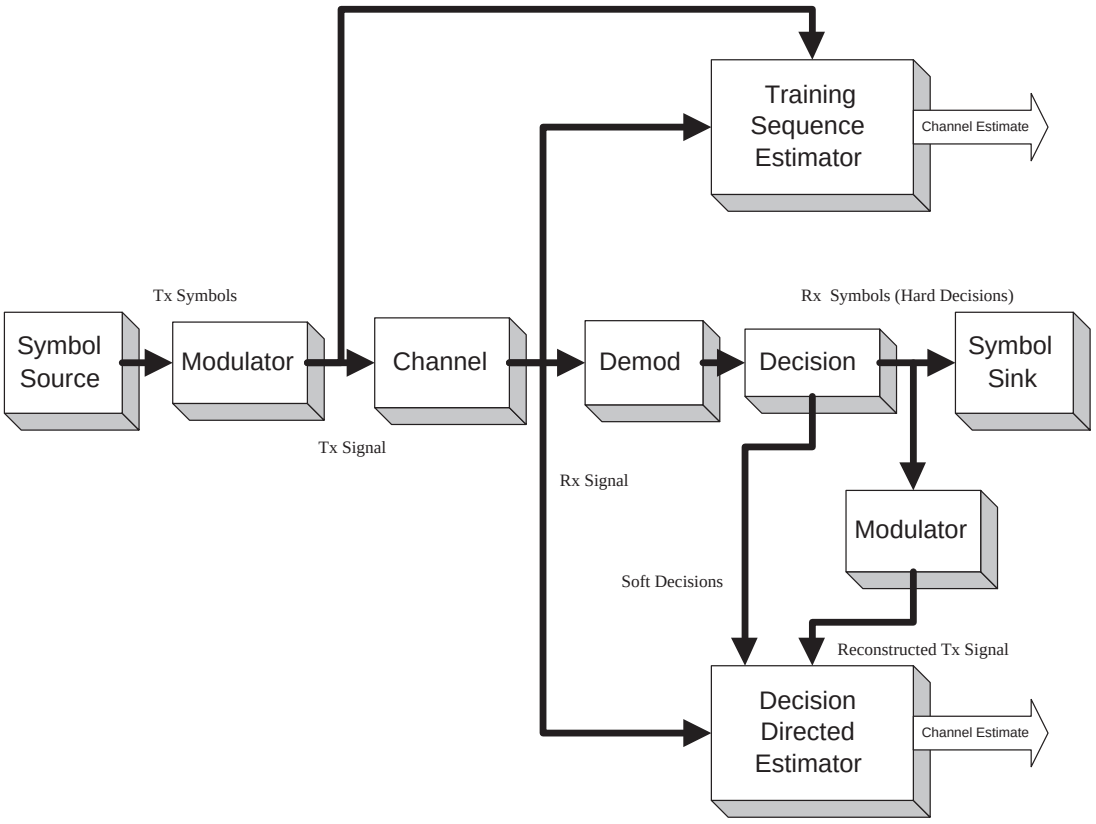

Figure 2.1: Conceptual block diagram of the wireless communication channel estimation problem

2.2 Multiple Phase Shift Keying (MPSK)

Although estimation algorithms are not inherently dependent upon the modulation scheme, we will consider multiple phase shift keying (MPSK) modulation for implementation and simulation purposes. This section provides an overview of MPSK modulation and demodulation techniques ([33],pp. 142-144).

2.2.1 Modulation

An MPSK modulator takes a symbol $i \in \{1, \cdots, M\}$ where M is the number of possible symbols and produces the signalling waveform during its symbol interval

$$s_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi f_c t + \frac{2\pi i}{M}\right) \qquad \text{for } 0 \leq t < T \tag{2.1}$$

where T is the symbol interval duration, E is the energy of $s_i(t)$ over the symbol interval, and f_c is the carrier frequency.

We can rewrite 2.1 as

$$\begin{aligned} s_i(t) &= \sqrt{\frac{2E}{T}} \cos\left(\frac{2\pi i}{M}\right) \cos(2\pi f_c t) - \sqrt{\frac{2E}{T}} \sin\left(\frac{2\pi i}{M}\right) \sin(2\pi f_c t) \\ &= a_{i1}\psi_1(t) + a_{i2}\psi_2(t) \end{aligned} \tag{2.2}$$

where

$$\psi_1(t) = \sqrt{\frac{2}{T}} \cos(2\pi f_c t) \tag{2.3}$$

$$\psi_2(t) = -\sqrt{\frac{2}{T}} \sin(2\pi f_c t) \tag{2.4}$$

$$a_{i1} = \sqrt{E} \cos(\phi_i) \tag{2.5}$$

$$a_{i2} = \sqrt{E} \sin(\phi_i) \tag{2.6}$$

$$\phi_i = \frac{2\pi i}{M} \tag{2.7}$$

We make the following remarks about MPSK modulation based on its signalling waveform when T is an integer multiple of $1/2f_c$, or in practice when $T \gg 1/2f_c$.

Remark 2.1 The signals $\psi_1(t)$ and $\psi_2(t)$ have unit energy.

Proof.

$$\begin{aligned}\int_0^T \psi_1^2(t)dt &= \frac{2}{T}\int_0^T \frac{1}{2}\left(1+\cos\left(2\pi 2 f_c t\right)\right)dt \\ &= 1+\frac{1}{T}\int_0^T \cos\left(2\pi 2 f_c t\right)dt \\ &\approx 1\end{aligned}$$

Similarly,

$$\int_0^T \psi_2^2(t)dt \approx 1$$

□

Remark 2.2 (Orthogonality) The signals $\psi_1(t)$ and $\psi_2(t)$ are orthogonal.

Proof.

$$\begin{aligned}\int_0^T \psi_1(t)\psi_2(t)dt &= -\frac{2}{T}\int_0^T \cos\left(2\pi f_c t\right)\sin\left(2\pi f_c t\right)dt \\ &= -\frac{1}{T}\int_0^T \sin\left(2\pi 2 f_c t\right)dt \\ &\approx 0\end{aligned}$$

□

Remark 2.3 (Orthonormal Basis) From the previous two remarks and (2.2), $\psi_1(t)$ and $\psi_2(t)$ form an orthonormal basis for the signal space $\{s_i(t) \mid i = 1, \cdots, M; 0 \leq t < T\}$.

Figure 2.2 shows (2.2) in a block diagram implementation. The function of the baseband MPSK modulator block is to calculate (2.5) and (2.6) for each symbol.

2.2.2 Coherent Demodulation

Because $\psi_1(t)$ and $\psi_2(t)$ form an orthonormal basis for the signaling set, performing an inner product with the modulated signal $s_i(t)$ and $\psi_j(t)$ yields the jth coordinate, a_{ij}, for symbol i. Figure 2.3 shows this geometric interpretation of MPSK modulation and demodulation. Notice that the angle ϕ_i is nothing more than the $\arctan(a_{i2}/a_{i1})$. Using this observation, an MPSK demodulator can be implemented as shown in Figure 2.4 ([33],pg. 144). The demodulation process begins with inner product of each basis function with the received signal $r_i(t)$. The

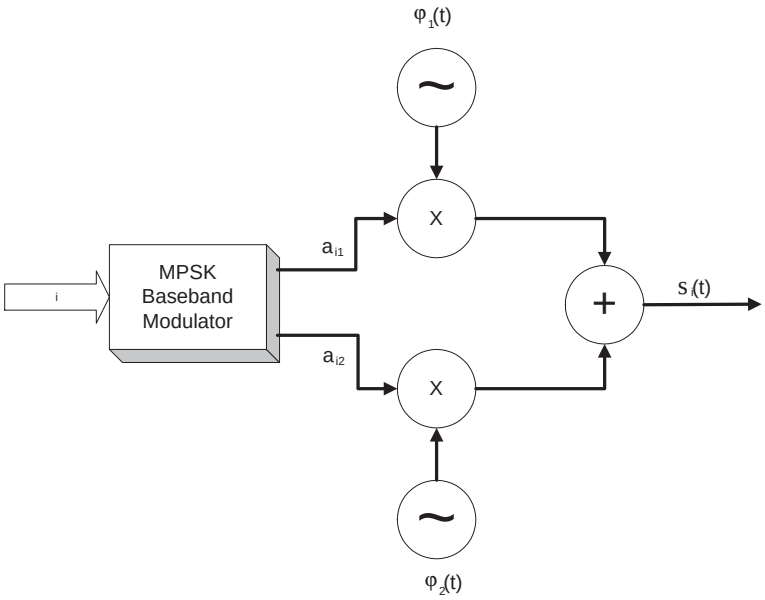

Figure 2.2: Block diagram of an MPSK modulator

angle of the resulting coordinate θ_i is then computed and compared to the known set of possible signal coordinate angles $\{\phi_i | i = 1, \cdots, M\}$. The receiver then chooses the symbol i whose coordinate is angularly closest to the received angle θ_i.

2.3 Characterizing Wireless Communication Channels

In its most general definition the channel represents everything between the information source and sink. Because the designer of a wireless communication system usually specifies most of its elements between the source and sink, with the exception of the free-space medium, we will restrict our definition of channel to this free-space medium. A number of models have been proposed to account for free-space effects [31], [32], [30], [17]. These models try to emulate the most severe distortion caused by wireless channels, which is multipath distortion. For a more complete description of modeling communication channels please refer to ([18],pp. 362-386).

As seen in Figure 2.5, several paths can exist between the transmitter and receiver of a wireless communication system. These paths are caused by the reflections, refractions, and scattering of the electromagnetic waves carrying the information off of objects such as buildings, trees, ground, atmospheric layers, etc. These multiple paths create the multipath phenomenon, characterized further by a combination of two types of multipath, specular and diffuse.

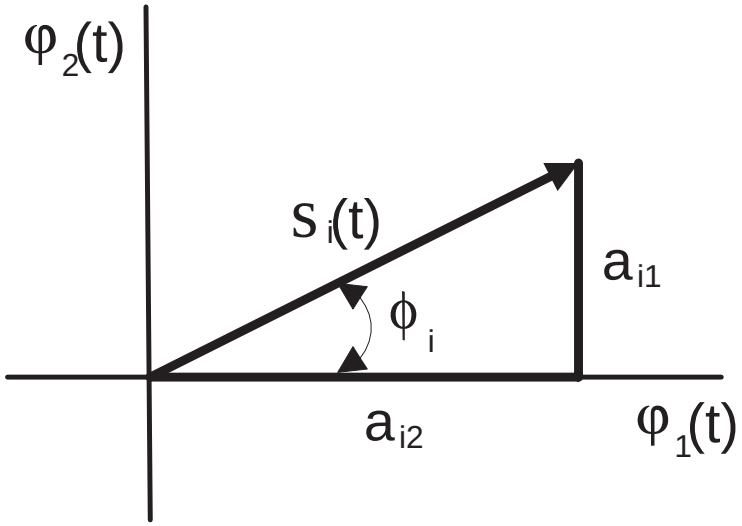

Figure 2.3: Geometric interpretation of MPSK signalling

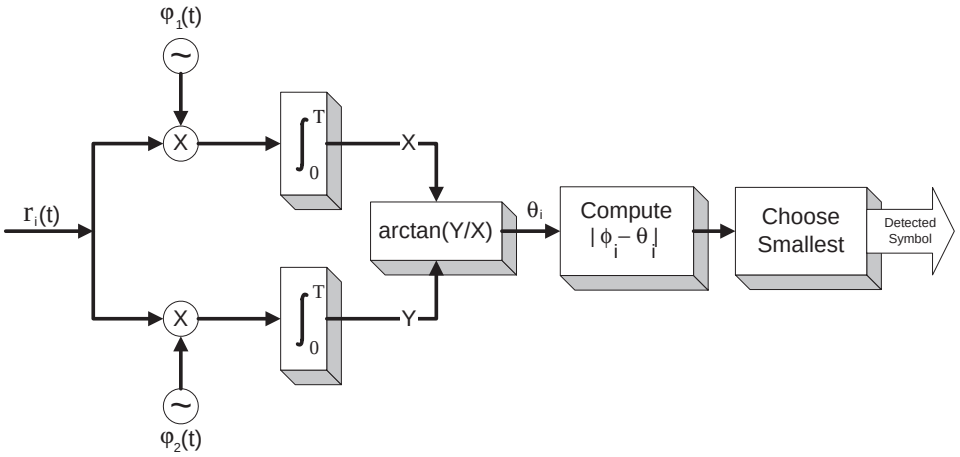

Figure 2.4: Block diagram of an MPSK demodulator

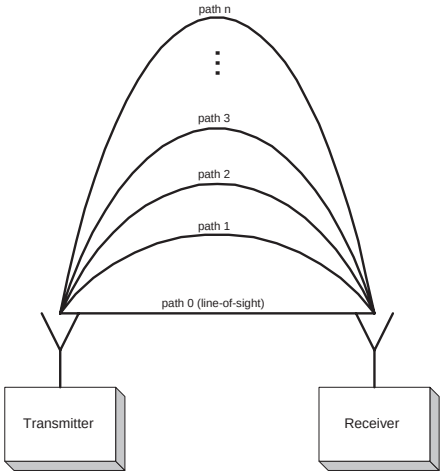

Figure 2.5: The wireless channel multipath phenomenon

2.3.1 Specular Multipath

When the paths between the transmitter and receiver are discrete, each with its own attenuation and time delay, the multipath is called specular. The signal at the receiver, $y(t)$, can then be modeled as the summation of scaled and delayed versions of the transmitted signal, $s(t)$, written

$$y(t) = \sum_n \alpha_n(t) s(t - \tau_n(t)) \tag{2.8}$$

where $\alpha_n(t)$ and $\tau_n(t)$ are time-varying attenuations and delays for each discrete path. Because of its form in (2.8), discrete multipath is naturally represented as a tapped delay line with time-varying coefficients and possibly time-varying tap spacings.

The differential delay (or delay spread) is defined as the difference between the largest and smallest path delays. If the differential delay of the channel is on the order of the reciprocal of the information bandwidth, then frequency selective fading of the information signal is noticeable. Frequency selective fading manifests itself in the time domain as intersymbol interference, or the “bleeding” of previous symbol interval energy into the current symbol interval.

2.3.2 Diffuse Multipath

When the paths between the transmitter and receiver form a continuum, the multipath is called diffuse. In this case, the received signal $y(t)$ is related to the transmitted signal $s(t)$ by

$$y(t) = \int_{-\infty}^{+\infty} \alpha(\tau; t)s(t-\tau)d\tau \tag{2.9}$$

where $\alpha(\tau; t)$ is the channel response at time t to an impulse transmitted at time τ. Under widesense stationary uncorrelated scattering (WSSUS) assumptions [2], the delay cross-power spectral density is defined as

$$R_c(\tau; \Delta t) = \frac{1}{2}E\left\{\widetilde{h}^*(\tau, t)\widetilde{h}(\tau, t+\Delta t)\right\} \tag{2.10}$$

where * denotes complex conjugation and

$$\widetilde{h}(\tau, t) = \alpha(\tau; t)e^{-j2\pi f_c \tau} \tag{2.11}$$

is the low-pass time variant impulse response of the channel. Evaluating $R_c(\tau; \Delta t)$ at $\Delta t = 0$ yields the delay power spectrum or multipath intensity profile, $R_c(\tau) = R_c(\tau; 0)$. The range of τ for which $R_c(\tau)$ is approximately non-zero is called the multipath delay spread. The scattering function, $S(\tau, v)$, is defined as the Fourier transform of $R_c(\tau; \Delta t)$ with respect to Δt.

$$S(\tau, v) = \int_{-\infty}^{+\infty} R_c(\tau; \Delta t)e^{-j2\pi v \Delta t}d(\Delta t) \tag{2.12}$$

Although represented here as a time-varying phenomenon, if the bandwidth of the scattering function for each delay τ is small with respect to the reciprocal of the symbol interval duration, the channel can be modeled as a linear time-invariant (LTI) system for that duration.

If we assume that the signal is band-limited, then using results from [19], the time-varying diffuse multipath channel can be described as a tapped-delay line with time-varying coefficients and fixed tap spacings ([18], pg. 376). We can then express the low-pass complex envelope of the output signal as

$$\widetilde{y}(t) = \sum_m \frac{1}{W}\widetilde{h}\left(\frac{m}{W}; t\right)\widetilde{s}\left(t - \frac{m}{W}\right) \tag{2.13}$$

where W is the bandwidth of the input signal and $\widetilde{s}(t)$ is the low-pass complex envelope of the transmitted signal. The number of taps in the delay line is $ceil(T_m W) + 1$, where T_m is the

multipath delay spread ([18], pg. 377), ([25], pg. 137).

2.4 Conversion of Real Bandpass Signals and Systems to Complex Low-Pass Signals and Systems

For simulation purposes, complex low-pass systems are preferable to real bandpass systems because they are easier to sample. The purpose of this section is to convert a real bandpass system into a complex low-pass system. Consider a carrier modulated signal $x(t)$ with slowly varying amplitude $r(t)$ and phase $\phi(t)$ described by

$$\begin{aligned} x(t) &= r(t)\cos\left(2\pi f_c t + \phi(t)\right) \\ &= Re\left\{r(t)e^{j\phi(t)}e^{j2\pi f_c t}\right\} \\ &= Re\left\{v(t)e^{j2\pi f_c t}\right\} \end{aligned}$$

where $Re\{z\}$ denotes the real part of z and

$$v(t) = r(t)e^{j\phi(t)}$$

The complex low-pass signal $v(t)$ is often called the complex low-pass equivalent or complex envelope of the signal $x(t)$. If the bandwidth of $x(t)$ is less than or equal to the carrier frequency then we will show via Hilbert transformation techniques an equivalent relationship between bandpass filtering $x(t)$ and low-pass filtering $v(t)$ ([18], pp. 36-45).

Definition 2.4 (Hilbert Transform) The Hilbert transform, denoted by $\widehat{x}(t)$, of a signal $x(t)$ is defined by the relationship

$$\widehat{x}(t) = \mathcal{H}\left\{x(t)\right\} = -j\mathcal{F}^{-1}\left\{X(f)\text{sgn}(f)\right\}$$

where $X(f)$ is the Fourier transform of $x(t)$, $\text{sgn}(f)$ is the signum function, and $\mathcal{F}^{-1}$ is the inverse Fourier transform operator.

Definition 2.5 (Preenvelope) Define the preenvelope or analytic signal of $x(t)$ as

$$z_x(t) = x(t) + j\widehat{x}(t)$$

Denote the the Fourier transform of $z_x(t)$ as $Z_x(f)$.

Definition 2.6 (Complex Envelope) Define the complex envelope of a signal $x(t)$ as

$$\widetilde{x}(t) = z_x(t)e^{-j2\pi f_c t}$$

Remark 2.7 Notice the relationship between $x(t)$, $z_x(t)$, and $\widetilde{x}(t)$ when $x(t)$ is real.

$$\begin{aligned} x(t) &= Re\left\{z_x(t)\right\} && (2.14)\\ &= Re\left\{\widetilde{x}(t)e^{j2\pi f_c t}\right\} && (2.15)\end{aligned}$$

Lemma 2.8 (Preenvelope of an Amplitude and Phase Modulated Signal) Let $x(t)$ be an amplitude and phase modulated signal described by

$$x(t) = p(t)\cos(2\pi f_c t) - q(t)\sin(2\pi f_c t)$$

where $p(t)$ and $q(t)$ are low-pass signals with absolute bandwidths less than or equal to f_c. Then the preenvelope of $x(t)$ is

$$z_x(t) = [p(t) + jq(t)]e^{j2\pi f_c t}$$

Proof. Denote the Fourier transforms of $p(t)$ and $q(t)$ as $P(f)$ and $Q(f)$, respectively. Then

$$\begin{aligned}\widehat{X}(f) &= -jX(f)\mathrm{sgn}(f)\\ &= -j\left(\frac{1}{2}\left[P(f-f_c)+P(f+f_c)\right] - \frac{1}{2j}\left[Q(f-f_c)-Q(f+f_c)\right]\right)\mathrm{sgn}(f)\\ &= -j\left(\frac{1}{2}\left[P(f-f_c)-P(f+f_c)\right] - \frac{1}{2j}\left[Q(f-f_c)+Q(f+f_c)\right]\right)\\ &= \frac{1}{2j}\left[P(f-f_c)-P(f+f_c)\right] + \frac{1}{2}\left[Q(f-f_c)+Q(f+f_c)\right]\end{aligned}$$

Taking the inverse Fourier transform yields

$$\widehat{x}(t) = p(t)\sin(2\pi f_c t) + q(t)\cos(2\pi f_c t)$$

Hence,

$$\begin{aligned} z_x(t) &= x(t) + j\widehat{x}(t) \\ &= p(t)\cos(2\pi f_c t) - q(t)\sin(2\pi f_c t) + jp(t)\sin(2\pi f_c t) + jq(t)\cos(2\pi f_c t) \\ &= p(t)\cos(2\pi f_c t) + jp(t)\sin(2\pi f_c t) + j\left(q(t)\cos(2\pi f_c t) + jq(t)\sin(2\pi f_c t)\right) \\ &= \left(p(t) + jq(t)\right)e^{j2\pi f_c t} \end{aligned}$$

□

Lemma 2.9 (Fourier Transform of the Preenvelope) Let $z_x(t)$ be the preenvelope of $x(t)$ with respective Fourier transforms $Z_x(f)$ and $X(f)$. Then

$$Z_x(f) = 2X(f)u(f)$$

where $u(f)$ is the unit step function.

Proof. Denote the Fourier transform of $\widehat{x}(t)$ as $\widehat{X}(f)$. Then

$$\begin{aligned} Z_x(f) &= X(f) + j\widehat{X}(f) \\ &= X(f) + X(f)\text{sgn}(f) \\ &= 2X(f)u(f) \end{aligned}$$

where we have used

$$j\mathcal{F}\{\widehat{x}(t)\} = X(f)\text{sgn}(f)$$

and

$$2u(f) = 1 + \text{sgn}(f)$$

□

Theorem 2.10 (Preenvelope Convolution) Let $z_h(t)$ and $z_x(t)$ be the preenvelopes of signals $h(t)$ and $x(t)$, respectively. If

$$y(t) = \int_{-\infty}^{+\infty} h(t-\tau)x(\tau)d\tau$$

then

$$z_y(t) = \frac{1}{2}\int_{-\infty}^{+\infty} z_h(t-\tau) z_x(\tau) d\tau$$

where $z_y(t)$ is the preenvelope of $y(t)$.

Proof. From Lemma 2.9 write

$$\begin{aligned} Z_x(f) &= 2X(f)u(f) \\ Z_h(f) &= 2H(f)u(f) \\ Z_y(f) &= 2Y(f)u(f) \end{aligned}$$

Because

$$Y(f) = H(f)X(f)$$

we can write

$$2Y(f)u(f) = \frac{1}{2}\left(2H(f)u(f)\right)\left(2X(f)u(f)\right)$$

Hence,

$$Z_y(f) = \frac{1}{2} Z_h(f) Z_x(f)$$

Taking the inverse Fourier transform gives the desired result.

□

Corollary 2.11 (Complex Envelope Convolution) Let $h(t)$ be the impulse response of a bandpass LTI system centered at f_c. Then passing $x(t)$ through this system yields $y(t)$ with a complex envelope given by

$$\widetilde{y}(t) = \int_{-\infty}^{+\infty} \widetilde{h}(t-\tau)\widetilde{x}(\tau) d\tau$$

where $\widetilde{x}(t)$ is the complex envelope of $x(t)$ and $\widetilde{h}(t)$ is the complex envelope of $h(t)$ scaled by *one-half*.

Proof. From Theorem 2.10 write

$$\begin{aligned} z_y(t) &= \frac{1}{2}\int_{-\infty}^{+\infty} z_h(t-\tau) z_x(\tau) d\tau \\ &= \int_{-\infty}^{+\infty} \widetilde{h}(t-\tau) e^{j2\pi f_c(t-\tau)} \widetilde{x}(\tau) e^{j2\pi f_c \tau} d\tau \\ &= e^{j2\pi f_c t} \int_{-\infty}^{+\infty} \widetilde{h}(t-\tau)\widetilde{x}(\tau) d\tau \end{aligned}$$

Hence

$$\widetilde{y}(t) = z_y(t) e^{-j2\pi f_c t} = \int_{-\infty}^{+\infty} \widetilde{h}(t-\tau)\widetilde{x}(\tau) d\tau$$

□

Corollary 2.12 (Amplitude and Phase Modulated Signal Through a Bandpass System) Passing an amplitude and phase modulated signal $x(t)$, described by

$$x(t) = p(t)\cos(2\pi f_c t) - q(t)\sin(2\pi f_c t)$$

where the absolute bandwidths of $q(t)$ and $p(t)$ are less than or equal to f_c, through a bandpass LTI system with impulse response $h(t)$ produces an output $y(t)$ whose complex envelope is

$$\widetilde{y}(t) = Re\left\{\widetilde{h}(t)\right\} \otimes p(t) - Im\left\{\widetilde{h}(t)\right\} \otimes q(t) + j\left(Im\left\{\widetilde{h}(t)\right\} \otimes p(t) + Re\left\{\widetilde{h}(t)\right\} \otimes q(t)\right)$$

where $\widetilde{x}(t)$ is the complex envelope of $x(t)$, $\widetilde{h}(t)$ is the complex envelope of $h(t)$ scaled by *one-half*, and $\otimes$ denotes the convolution operation.

Proof. From Lemma 2.8 and Definition 2.6 we have

$$\widetilde{x}(t) = p(t) + jq(t)$$

Applying Corollary 2.11 gives

$$\begin{aligned} \widetilde{y}(t) &= \widetilde{h}(t) \otimes \widetilde{x}(t) \\ &= \left(Re\left\{\widetilde{h}(t)\right\} + jIm\left\{\widetilde{h}(t)\right\}\right) \otimes (p(t) + jq(t)) \\ &= Re\left\{\widetilde{h}(t)\right\} \otimes p(t) - Im\left\{\widetilde{h}(t)\right\} \otimes q(t) + j\left(Im\left\{\widetilde{h}(t)\right\} \otimes p(t) + Re\left\{\widetilde{h}(t)\right\} \otimes q(t)\right) \end{aligned}$$

□

Remark 2.13 (Low-Pass MPSK Modulation and Demodulation) Applying Lemma 2.8 to the MPSK signalling waveform in (2.1) yields the complex envelope

$$\widetilde{s}_i(t) = \sqrt{\frac{2E}{T}} \cos\left(\frac{2\pi i}{M}\right) + j\sqrt{\frac{2E}{T}} \sin\left(\frac{2\pi i}{M}\right)$$

Furthermore, the mixing stage of the MPSK demodulator is built into the bandpass to low-pass conversion process. Notice that the output of the top integrator in Figure 2.2 when $T \gg 1/f_c$ is

$$\begin{aligned}
X &= \sqrt{\frac{2}{T}} \int_{-\infty}^{+\infty} r(t) \cos(2\pi f_c t)\, dt \\
&= \sqrt{\frac{2}{T}} \int_{-\infty}^{+\infty} Re\left\{\widetilde{r}(t) e^{j2\pi f_c t}\right\} \cos(2\pi f_c t)\, dt \\
&= \sqrt{\frac{1}{2T}} \int_{-\infty}^{+\infty} Re\left\{\widetilde{r}(t) e^{j2\pi f_c t}\left(e^{j2\pi f_c t} + e^{-j2\pi f_c t}\right)\right\} dt \\
&= \sqrt{\frac{1}{2T}} \int_{-\infty}^{+\infty} Re\left\{\widetilde{r}(t)\right\} dt + \sqrt{\frac{1}{2T}} \int_{-\infty}^{+\infty} Re\left\{\widetilde{r}(t) e^{j2\pi 2 f_c t}\right\} dt \\
&\approx \sqrt{\frac{1}{2T}} \int_{-\infty}^{+\infty} Re\left\{\widetilde{r}(t)\right\} dt
\end{aligned}$$

Similarly,

$$Y \approx \sqrt{\frac{1}{2T}} \int_{-\infty}^{+\infty} Im\left\{\widetilde{r}(t)\right\} dt$$

Thus, when simulating a baseband MPSK system (or in general, a quadrature multiplexing system) the real component of the received complex envelope is the in-phase component of the received signal and the imaginary component is the quadrature component.

2.5 Adaptive Algorithms

We will consider two classes of adaptive algorithms for the purpose of estimating the impulse response of the channel: linear estimators and least mean squares (LMS) estimators.

2.5.1 Linear Estimators

2.5.1.1 Training Sequence Estimation

2.5.1.1.1 The System Identification Problem When the receiver has access to the transmitted signal, or when the receiver and transmitter prenegotiate a training sequence, the estimation of the channel is the classical system identification problem ([22],pp. 169-228),[8]. A block diagram of this problem's structure is shown in Figure 2.6.

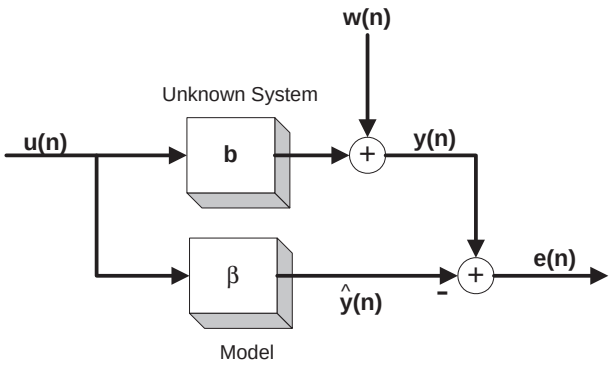

Figure 2.6: A block diagram of the training sequence system identification problem

From Figure 2.6 define the following relationships:

$$y(n) = b_1 u_1(n) + \cdots + b_M u_M(n) + w(n) \tag{2.16}$$

$$\widehat{y}(n) = \beta_1 u_1(n) + \cdots + \beta_M u_M(n) \tag{2.17}$$

$$e(n) = y(n) - \widehat{y}(n) = y(n) - (\beta_1 u_1(n) + \cdots + \beta_M u_M(n)) \tag{2.18}$$

where $u_i(n)$, $i = 1, \ldots, M$ are real-valued discrete-time random processes, $b_1, \cdots, b_M, \beta_1, \cdots, \beta_M \in \mathcal{R}$ where $\mathcal{R}$ is the set of real numbers, M is a positive integer, and $w(n)$ is a mean zero real-valued discrete-time random process. After observing the system for N sample periods we can

write the collected data in matrix form as

$$\mathbf{y} = \begin{bmatrix} y(1) & y(2) & \cdots & y(N) \end{bmatrix}^T \tag{2.19}$$

$$\mathbf{w} = \begin{bmatrix} w(1) & w(2) & \cdots & w(N) \end{bmatrix}^T \tag{2.20}$$

$$\mathbf{e} = \begin{bmatrix} e(1) & e(2) & \cdots & e(N) \end{bmatrix}^T \tag{2.21}$$

$$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \cdots & b_M \end{bmatrix}^T \tag{2.22}$$

$$\beta = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_M \end{bmatrix}^T \tag{2.23}$$

$$\mathbf{U} = \begin{bmatrix} u_1(1) & \cdots & u_M(1) \\ \vdots & \ddots & \vdots \\ u_1(N) & \cdots & u_M(N) \end{bmatrix} \tag{2.24}$$

with

$$\mathbf{y} = \mathbf{U}\mathbf{b} + \mathbf{w} \tag{2.25}$$

$$\mathbf{e} = \mathbf{y} - \mathbf{U}\beta \tag{2.26}$$

where ^{T} is the matrix transpose operator. Define the error or loss function $J(\beta)$ as

$$J(\beta) = \mathbf{e^T R e} = (\mathbf{y} - \mathbf{U}\beta)^{\mathbf{T}} \mathbf{R} (\mathbf{y} - \mathbf{U}\beta) \tag{2.27}$$

where $\mathbf{R}$ is a positive definite symetric N x N matrix of weighting coefficients, possibly functions of $\mathbf{U}$. The choice of $\mathbf{R}$ depends on the desired application and estimator properties. For example, when

- $\mathbf{R} = diag(a_1, \cdots, a_N)$ [1] with $a_i \in \mathcal{R},\ i = 1 \ldots N$ then (2.27) is the loss function for weighted least squares estimators [4], [12]
- $\mathbf{R} = diag(\lambda^{N-1}, \cdots, \lambda, 1)$ where $0 < \lambda < 1$, then (2.27) is the loss function for exponentially weighted least squares estimators ([34], pp. 96-98)
- $\mathbf{R} = \mathbf{I}$ then (2.27) is the loss function for ordinary least squares estimators ([22], pg. 189)
- $\mathbf{R} = \mathbf{W^{-1}}$ where $\mathbf{W} = \mathcal{E}\left\{\mathbf{w}\mathbf{w}^{\mathbf{T}}\right\}$ is the covariance of the noise process, then (2.27) is the loss function for Markov estimators ([8], pg. 189)

[1] $diag(a_1, \cdots, a_N)$ is defined as an N x N matrix whose diagonal elements are $a_1, \cdots, a_N$

2.5.1.1.2 Estimator Derivation

We now derive the general form for linear estimators.

Theorem 2.14 (Training Sequence Linear Estimator) The estimator $\widehat{\beta}$ of the system defined by (2.16) to (2.26) that minimizes the loss function (2.27) satisfies the so-called "normal" equation

$$\mathbf{U^T R U}\widehat{\beta} = \mathbf{U^T R y}$$

and if $\mathbf{U^T R U}$ is non-singular then

$$\widehat{\beta} = \left(\mathbf{U^T R U}\right)^{-1}\mathbf{U^T R y}$$

Proof. Simplify (2.27) to get

$$\begin{aligned} J(\beta) &= (\mathbf{y} - \mathbf{U}\beta)^{\mathbf{T}}\,\mathbf{R}\,(\mathbf{y} - \mathbf{U}\beta) \\ &= \mathbf{y^T R y} - \mathbf{y^T R U}\beta - \beta^{\mathbf{T}}\mathbf{U^T R y} + \beta^{\mathbf{T}}\mathbf{U^T R U}\beta \\ &= \mathbf{y^T R y} - 2\beta^{\mathbf{T}}\mathbf{U^T R y} + \beta^{\mathbf{T}}\mathbf{U^T R U}\beta \end{aligned}$$

Differentiating with respect to β yields

$$\frac{\partial J(\beta)}{\partial \beta} = -2\mathbf{U^T R y} + 2\mathbf{U^T R U}\beta \tag{2.28}$$

For some vector $\widehat{\beta}$ we set this gradient equal to zero, leading to an expression for $\widehat{\beta}$ at which an extremum of J occurs.

$$\mathbf{U^T R U}\widehat{\beta} = \mathbf{U^T R y} \tag{2.29}$$

Because R is positive definite, the extremum at $\widehat{\beta}$ is a minimum. If $\mathbf{U^T R U}$ is non-singular then

$$\widehat{\beta} = \left(\mathbf{U^T R U}\right)^{-1}\mathbf{U^T R y} \tag{2.30}$$

□

Recall that for $\mathbf{R} = diag(a_1, \cdots, a_N)$ the estimator (2.30) is the weighted least squares (WLS) estimator, and for $\mathbf{R} = diag(\lambda^{N-1}, \cdots, \lambda, 1)$ where $0 < \lambda < 1$ it is an exponentially weighted least squares estimator (EWLS). The form for these estimators in (2.30) is not desirable for implementation because it requires the storage of all the past data of the system. We will now derive a recursive form for these estimators that is suitable for digital implementation. For

further analysis of the WLS estimator refer to [12] and [4], and for the EWLS refer to ([22], pg. 191) and ([34],pp. 96-98).

Corollary 2.15 (Recursive Weighted Least Squares Estimator) Let

$$\mathbf{u}(n) = \begin{bmatrix} u_1(n) & u_2(n) & \cdots & u_M(n) \end{bmatrix}^T$$

and

$$\mathbf{R} = \begin{bmatrix} \lambda^{N-1}a_1 & 0 & \cdots & 0 \\ 0 & \lambda^{N-2}a_2 & & \vdots \\ \vdots & \ddots & & \\ 0 & & \lambda a_{N-1} & 0 \\ 0 & \cdots & 0 & a_N \end{bmatrix}$$

where $a_i \in \mathcal{R}$ and $0 < \lambda \leq 1$ such that $\mathbf{R}$ is positive definite. If $\mathbf{U^T R U}$ is non-singular, then (2.30) is valid and can be written recursively[1] as

$$\widehat{\beta}_n = \widehat{\beta}_{n-1} + a_n \mathbf{H}_n^{-1} \mathbf{u}(n) e(n) \tag{2.31}$$

$$\mathbf{H}_n = \lambda \mathbf{H}_{n-1} + a_n \mathbf{u}(n) \mathbf{u}^T(n) \tag{2.32}$$

$$e(n) = y(n) - \mathbf{u}(n)^T \widehat{\beta}_{n-1} \tag{2.33}$$

Proof. With

$$\mathbf{u}(n) = \begin{bmatrix} u_1(n) & u_2(n) & \cdots & u_M(n) \end{bmatrix}^T \tag{2.34}$$

$\mathbf{U}$ becomes

$$\mathbf{U} = \begin{bmatrix} \mathbf{u^T}(\mathbf{1}) \\ \vdots \\ \mathbf{u^T}(\mathbf{N}) \end{bmatrix} \tag{2.35}$$

[1] Applying the matrix inversion lemma can further reduce computational complexity by eliminating the matrix inverse ([22], pp. 190-191), ([34], pg. 98).

For $\mathbf{R} = diag(\lambda^{N-1}a_1, \cdots, \lambda a_{N-1}, a_N)$, define $\mathbf{H_n}$ to be the product $\mathbf{U^T R U}$ after collecting n intervals of data (i.e. $N = n$). Expanding this product yields

$$\begin{aligned}
\mathbf{H}_n &= \left[\ \mathbf{u}(1)\cdots\mathbf{u}(n)\ \right]\begin{bmatrix} \lambda^{n-1}a_1 & 0 & \cdots & 0 \\ 0 & \lambda^{n-2}a_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & a_n \end{bmatrix}\begin{bmatrix} \mathbf{u}^T(1) \\ \vdots \\ \mathbf{u}^T(n) \end{bmatrix} \\
&= \left[\ \lambda^{n-1}a_1\mathbf{u}(1)\cdots a_n\mathbf{u}(n)\ \right]\begin{bmatrix} \mathbf{u}^T(1) \\ \vdots \\ \mathbf{u}^T(n) \end{bmatrix} \\
&= \sum_{k=1}^{n}\lambda^{n-k}a_k\mathbf{u}(k)\mathbf{u}^T(k) \\
&= a_n\mathbf{u}(n)\mathbf{u}^T(n) + \lambda\sum_{k=1}^{n-1}\lambda^{n-1-k}a_k\mathbf{u}(k)\mathbf{u}^T(k) \\
&= a_n\mathbf{u}(n)\mathbf{u}^T(n) + \lambda\mathbf{H}_{n-1} \qquad (2.36)
\end{aligned}$$

Similarly, define $\mathbf{s}_n$ to be the product of $\mathbf{U^T R y}$ after collecting n intervals of data (i.e. $N = n$). Expanding this product yields

$$\begin{aligned}
\mathbf{s}_n &= \left[\ \mathbf{u}(1)\cdots\mathbf{u}(n)\ \right]\begin{bmatrix} \lambda^{n-1}a_1 & 0 & \cdots & 0 \\ 0 & \lambda^{n-2}a_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & a_n \end{bmatrix}\begin{bmatrix} y(1) \\ \vdots \\ y(n) \end{bmatrix} \\
&= \left[\ \lambda^{n-1}a_1\mathbf{u}(1)\cdots a_n\mathbf{u}(n)\ \right]\begin{bmatrix} y(1) \\ \vdots \\ y(n) \end{bmatrix} \\
&= \sum_{k=1}^{n}\lambda^{n-k}a_k\mathbf{u}(k)y(k) \\
&= a_n\mathbf{u}(n)y(n) + \lambda\sum_{k=1}^{n-1}\lambda^{n-1-k}a_k\mathbf{u}(k)y(k) \\
&= a_n\mathbf{u}(n)y(n) + \lambda\mathbf{s}_{n-1} \qquad (2.37)
\end{aligned}$$

Hence, we can write (2.30) after collecting n intervals of data as

$$\widehat{\beta}_n = \mathbf{H}_n^{-1}\mathbf{s}_n \tag{2.38}$$

and (2.29) as

$$\mathbf{H}_n\widehat{\beta}_n = \mathbf{s}_n \tag{2.39}$$

From (2.36), (2.37), (2.38), and (2.39) we derive a recursive form of the WLS estimator.

$$\begin{aligned}
\widehat{\beta}_n &= \mathbf{H}_n^{-1}\mathbf{s}_n \\
&= \mathbf{H}_n^{-1}\left(a_n\mathbf{u}(n)y(n) + \lambda\mathbf{s}_{n-1}\right) \\
&= \mathbf{H}_n^{-1}\left(a_n\mathbf{u}(n)y(n) + \lambda\mathbf{H}_{n-1}\widehat{\beta}_{n-1}\right) \\
&= a_n\mathbf{H}_n^{-1}\mathbf{u}(n)y(n) + \mathbf{H}_n^{-1}\lambda\mathbf{H}_{n-1}\widehat{\beta}_{n-1} \\
&= a_n\mathbf{H}_n^{-1}\mathbf{u}(n)y(n) + \mathbf{H}_n^{-1}\left(\mathbf{H}_n - a_n\mathbf{u}(n)\mathbf{u}^T(n)\right)\widehat{\beta}_{n-1} \\
&= \widehat{\beta}_{n-1} + a_n\mathbf{H}_n^{-1}\mathbf{u}(n)\left(y(n) - \mathbf{u}(n)^T\widehat{\beta}_{n-1}\right) \\
&= \widehat{\beta}_{n-1} + a_n\mathbf{H}_n^{-1}\mathbf{u}(n)e(n)
\end{aligned}$$

where

$$e(n) = y(n) - \mathbf{u}(n)^T\widehat{\beta}_{n-1}$$

□

2.5.1.1.3 Properties of Training Sequence Linear Estimators We will now derive several properties of linear estimators. ([8], pp. 185-190)

Definition 2.16 (Linearity) An estimator $\widehat{\beta}$ is linear if for some matrix $\mathbf{Q}$

$$\widehat{\beta} = \mathbf{Q}\mathbf{y}$$

Proposition 2.17 (Linearity of Linear Estimators) As the name suggests, linear estimators are linear because they fit the form

$$\widehat{\beta} = \mathbf{Q}\mathbf{y} \tag{2.40}$$

where $\mathbf{Q}$ in this case is

$$\mathbf{Q} = \left(\mathbf{U^T R U}\right)^{-1} \mathbf{U^T R} \tag{2.41}$$

Definition 2.18 (Unbiasness) An estimator $\widehat{\beta}$ is unbiased if its expectation equals the true parameter, i.e.

$$\mathcal{E}\left\{\widehat{\beta}\right\} = \mathbf{b}$$

Notation 2.19 Denote the joint probability density function of the three stochastic processes $y(n)$, $\mathbf{u}(n)$, and $w(n)$ as p. This distribution is a function of the true parameter $\mathbf{b}$. Denote the expectation with respect to p as $\mathcal{E}$.

Proposition 2.20 (Unbiasness of Training Sequence Linear Estimators) If

$$\mathcal{E}\left\{\mathbf{w}|\mathbf{U}\right\} = 0 \tag{2.42}$$

then

$$\mathcal{E}\left\{\widehat{\beta}\right\} = \mathbf{b} \tag{2.43}$$

Proof. Substitute (2.25) into (2.30) and simplify to get

$$\begin{aligned}
\widehat{\beta} &= \left(\mathbf{U^T R U}\right)^{-1} \mathbf{U^T R} \left(\mathbf{U b} + \mathbf{w}\right) \\
&= \mathbf{b} + \left(\mathbf{U^T R U}\right)^{-1} \mathbf{U^T R w}
\end{aligned}$$

Taking the expectation conditioned upon $\mathbf{U}$ gives

$$\begin{aligned}
\mathcal{E}\left\{\widehat{\beta} \mid \mathbf{U}\right\} &= \mathbf{b} + \mathcal{E}\left\{\left(\mathbf{U^T R U}\right)^{-1} \mathbf{U^T R w} \mid \mathbf{U}\right\} \\
&= \mathbf{b} + \left(\mathbf{U^T R U}\right)^{-1} \mathbf{U^T R} \mathcal{E}\left\{\mathbf{w} \mid \mathbf{U}\right\} \\
&= \mathbf{b}
\end{aligned}$$

Again taking the expectation yields

$$\begin{aligned}
\mathcal{E}\left\{\widehat{\beta}\right\} &= \mathcal{E}\left\{\mathcal{E}\left\{\widehat{\beta} \mid \mathbf{U}\right\}\right\} \\
&= \mathbf{b}
\end{aligned}$$

□

Proposition 2.21 (Covariance of Linear Unbiased Estimators) If

$$\mathcal{E}\{\mathbf{w}|\mathbf{U}\} = 0$$

Then the covariance of (2.30) is

$$\begin{aligned} \mathbf{cov}\left\{\widehat{\beta}\right\} &= \mathcal{E}\left\{\left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\} && (2.44)\\ &= \mathcal{E}\left\{\mathbf{QWQ^T}\right\} && (2.45)\end{aligned}$$

where

$$\begin{aligned} Q &= \left(\mathbf{U^T R U}\right)^{-1}\mathbf{U^T R} && (2.46)\\ W &= \mathcal{E}\left\{\mathbf{ww^T} \mid \mathbf{U}\right\} && (2.47)\end{aligned}$$

Proof. Simplify the argument of the covariance expression to get

$$\begin{aligned} \left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}} &= \left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\\ &= \left(\mathbf{Qy} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\mathbf{Qy} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\\ &= \left(\left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right) + \mathbf{Qw}\right)\left(\left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right) + \mathbf{Qw}\right)^{\mathbf{T}}\\ &= \left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\\ &\quad + \left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)(\mathbf{Qw})^{\mathbf{T}}\\ &\quad + \mathbf{Qw}\left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\\ &\quad + \mathbf{Qw}\,(\mathbf{Qw})^{\mathbf{T}}\end{aligned}$$

Taking the conditional expectation upon $\mathbf{U}$ gives

$$\begin{aligned} \mathcal{E}\left\{\left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}} \mid \mathbf{U}\right\} &= \left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\mathbf{QUb} - \mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\\ &\quad + \mathbf{QWQ^T}\end{aligned}$$

Again taking the expectation yields

$$\begin{aligned}\mathcal{E}\left\{\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\} &= \mathcal{E}\left\{\mathcal{E}\left\{\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}} \mid \mathbf{U}\right\}\right\} \\ &= \mathcal{E}\left\{\left(\mathbf{QUb}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\mathbf{QUb}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\} \\ &\quad +\mathcal{E}\left\{\mathbf{QWQ}^{\mathbf{T}}\right\} \qquad (2.48)\end{aligned}$$

Substituting (2.46) and (2.43) gives

$$\begin{aligned}\mathcal{E}\left\{\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\} &= \mathcal{E}\left\{\left(\left(\mathbf{U}^{\mathbf{T}}\mathbf{RU}\right)^{-1}\mathbf{U}^{\mathbf{T}}\mathbf{RUb}-\mathbf{b}\right)\left(\left(\mathbf{U}^{\mathbf{T}}\mathbf{RU}\right)^{-1}\mathbf{U}^{\mathbf{T}}\mathbf{RUb}-\mathbf{b}\right)^{\mathbf{T}}\right\} \\ &\quad +\mathcal{E}\left\{\mathbf{QWQ}^{\mathbf{T}}\right\} \qquad (2.49) \\ &= \mathcal{E}\left\{\mathbf{QWQ}^{\mathbf{T}}\right\} \qquad (2.50)\end{aligned}$$

□

Definition 2.22 (Consistency) An estimator $\widehat{\beta}$ is consistent if it converges in probability to the true parameter, i.e.

$$\lim_{n\to+\infty}\Pr\left\{|\widehat{\beta}_n-\mathbf{b}|\geq\epsilon\right\}=0$$

Proposition 2.23 (Consistency of the Markov Estimators) If $\mathbf{R}=\mathbf{W}^{-1}$ where

$$\mathbf{W}=\mathcal{E}\left\{\mathbf{ww}^{\mathbf{T}}\mid\mathbf{U}\right\}$$

then the estimator (2.30) is a Markov estimator. If

$$\mathcal{E}\left\{\mathbf{w}|\mathbf{U}\right\}=\mathbf{0}$$

then the Markov estimator is a consistent estimator.

Proof. By substituting $\mathbf{R}=\mathbf{W}^{-1}$ into (2.45) we get

$$\begin{aligned}\mathbf{cov}\left\{\widehat{\beta}\right\} &= \mathcal{E}\left\{\left(\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\mathbf{U}\right)^{-1}\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\mathbf{W}\left(\left(\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\mathbf{U}\right)^{-1}\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\right)^{\mathbf{T}}\right\} \\ &= \mathcal{E}\left\{\left(\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\mathbf{U}\right)^{-1}\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\mathbf{U}\left(\left(\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\mathbf{U}\right)^{-1}\right)\right\} \\ &= \left(\mathcal{E}\left\{\mathbf{U}^{\mathbf{T}}\mathbf{W}^{-1}\mathbf{U}\right\}\right)^{-1} \qquad (2.51)\end{aligned}$$

Eykhoff shows in ([8], pg. 189) that if $lim_{N\to+\infty}\sum_{n=1}^{N}\mathcal{E}\{u(n)^T u(n)\}=\infty$ then by Chebyshev's inequality, the Markov estimator converges in probability to the true parameter.

Definition 2.24 (Efficiency) An estimator $\widehat{\beta}$ is efficient if for any other linear estimator $\widehat{\theta}$

$$\mathbf{cov}\left\{\widehat{\beta}\right\} \leq \mathbf{cov}\left\{\widehat{\theta}\right\}$$

Proposition 2.25 (Efficiency of the Markov Estimators) If $\mathbf{R} = \mathbf{W}^{-1}$ where

$$\mathbf{W} = \mathcal{E}\left\{\mathbf{w}\mathbf{w}^{\mathbf{T}} \mid \mathbf{U}\right\}$$

then the estimator (2.30) is a Markov estimator. If

$$\mathcal{E}\left\{\mathbf{w}|\mathbf{U}\right\} = \mathbf{0}$$

then the Markov estimator is an efficient estimator.

Proof. Eykhoff shows in ([8], pg. 190) the efficiency of the Markov estimator using Schwarz's inequality for matrices.

Remark 2.26 Notice that the least squares estimate is a special case of the Markov estimate when the noise is white. Hence, the least squares estimate is consistent and efficient in white noise environments. ([8], pg. 190)

2.5.1.2 Decision Directed Estimation

2.5.1.2.1 The System Identification Problem When the receiver does not have knowledge of the transmitted signal, it must use blind estimation. If it uses the detected symbols to reconstruct the transmitted signal, and then uses this signal in place of the original signal, then it is using a subclass of blind estimation called decision directed estimation ([34], pg. 7), [26], [16], [6], [5], [20]. A block diagram of this problem's structure is shown in Figure 2.7.

From Figure 2.7 define the following relationships:

$$y(n) = b_1 u_1(n) + \cdots + b_M u_M(n) + w(n) \tag{2.52}$$

$$\widehat{y}(n) = \beta_1 x_1(n) + \cdots + \beta_M x_M(n) \tag{2.53}$$

$$e(n) = y(n) - \widehat{y}(n) = y(n) - (\beta_1 x_1(n) + \cdots + \beta_M x_M(n)) \tag{2.54}$$

where $u_i(n),\ i = 1, \ldots, M$ are real-valued discrete-time random processes, $x_i(n),\ i = 1, \ldots, M$ are the detected $u_i(n)$, $b_1, \cdots, b_M, \beta_1, \cdots, \beta_M \in \mathcal{R}$, M is a positive integer, and $w(n)$ is a mean

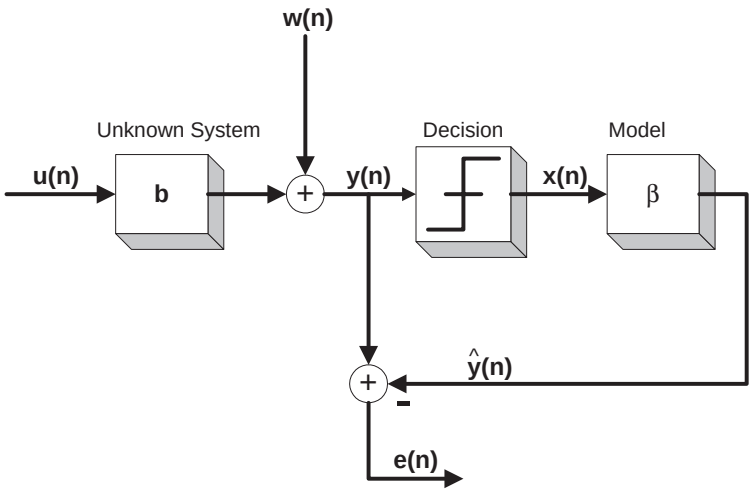

Figure 2.7: A block diagram of the decision directed system identification problem

zero real-valued discrete-time random process. After observing the system for N sample periods we can write the collected data in matrix form as

$$\mathbf{y} = \begin{bmatrix} y(1) & y(2) & \cdots & y(N) \end{bmatrix}^T \tag{2.55}$$

$$\mathbf{w} = \begin{bmatrix} w(1) & w(2) & \cdots & w(N) \end{bmatrix}^T \tag{2.56}$$

$$\mathbf{e} = \begin{bmatrix} e(1) & e(2) & \cdots & e(N) \end{bmatrix}^T \tag{2.57}$$

$$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \cdots & b_M \end{bmatrix}^T \tag{2.58}$$

$$\beta = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_M \end{bmatrix}^T \tag{2.59}$$

$$\mathbf{U} = \begin{bmatrix} u_1(1) & \cdots & u_M(1) \\ \vdots & \ddots & \vdots \\ u_1(N) & \cdots & u_M(N) \end{bmatrix} \tag{2.60}$$

$$\mathbf{X} = \begin{bmatrix} x_1(1) & \cdots & x_M(1) \\ \vdots & \ddots & \vdots \\ x_1(N) & \cdots & x_M(N) \end{bmatrix} \tag{2.61}$$

with

$$\mathbf{y} = \mathbf{U}\mathbf{b} + \mathbf{w} \tag{2.62}$$

$$\mathbf{e} = y - \mathbf{X}\beta \tag{2.63}$$

Define the error or loss function $J(\beta)$ as

$$J(\beta) = \mathbf{e}^{\mathbf{T}}\mathbf{R}\mathbf{e} = (\mathbf{y} - \mathbf{X}\beta)^{\mathbf{T}}\,\mathbf{R}\,(\mathbf{y} - \mathbf{X}\beta) \tag{2.64}$$

where $\mathbf{R}$ is a positive definite symetric N x N matrix of weighting coefficients, possibly dependent on $\mathbf{X}$ and $\mathbf{U}$. Again, the choice of $\mathbf{R}$ depends on the desired application and estimator properties. The *decision weighted* linear estimators are a subclass decision directed linear estimators where $\mathbf{R}$ is a function of the decision quality, which we can measure through mechanisms such as error-detecting codes or soft decisions. We call the non-realizable idealized choice of $\mathbf{R}$ such that $\mathbf{X}^{\mathbf{T}}\mathbf{R}\mathbf{X} = \mathbf{X}^{\mathbf{T}}\mathbf{R}\mathbf{U}$, ideal decision weighted linear estimation.

2.5.1.2.2 Estimator Derivation

Theorem 2.27 (Decision Directed Linear Estimator) The estimator $\widehat{\beta}$ of the system defined by (2.52) to (2.63) that minimizes the loss function (2.64) satisfies the so-called "normal" equation

$$\mathbf{X}^{\mathbf{T}}\mathbf{R}\mathbf{X}\widehat{\beta} = \mathbf{X}^{\mathbf{T}}\mathbf{R}\mathbf{y}$$

and if $\mathbf{X}^{\mathbf{T}}\mathbf{R}\mathbf{X}$ is non-singular then

$$\widehat{\beta} = \left(\mathbf{X}^{\mathbf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{R}\mathbf{y} \tag{2.65}$$

Proof. The proof is identical to the proof of Theorem 2.14 with $\mathbf{U}$ replaced with $\mathbf{X}$.

□

Corollary 2.28 (Decision Directed Recursive Weighted Least Squares Estimator) Let

$$\mathbf{x}(n) = \begin{bmatrix} x_1(n) & x_2(n) & \cdots & x_M(n) \end{bmatrix}^T$$

and

$$\mathbf{R} = \begin{bmatrix} \lambda^{N-1}a_1 & 0 & \cdots & 0 \\ 0 & \lambda^{N-2}a_2 & & \vdots \\ \vdots & \ddots & & \\ 0 & & \lambda a_{N-1} & 0 \\ 0 & \cdots & 0 & a_N \end{bmatrix} \tag{2.66}$$

where $a_i \in \mathcal{R}$ and $0 < \lambda \leq 1$ such that $\mathbf{R}$ is positive definite. If $\mathbf{X^T R X}$ is non-singular, then (2.65) is valid and can be written recursively as

$$\begin{aligned} \widehat{\beta}_n &= \widehat{\beta}_{n-1} + a_n \mathbf{H}_n^{-1} \mathbf{x}(n) e(n) && (2.67) \\ \mathbf{H}_n &= \lambda \mathbf{H}_{n-1} + a_n \mathbf{x}(n) \mathbf{x}^T(n) && (2.68) \\ e(n) &= y(n) - \mathbf{x}(n)^T \widehat{\beta}_{n-1} && (2.69) \end{aligned}$$

Proof. The proof is identical to that of Corollary 2.15 using the detected sequence $\mathbf{x(n)}$ instead of the input sequence $\mathbf{u(n)}$.

□

Definition 2.29 (Ideal Decision Weighted Linear Estimator) The ideal decision weighted linear estimator is defined as the decision directed linear estimator in Theorem 2.27 with $\mathbf{R}$ chosen such that $\mathbf{X^T R X} = \mathbf{X^T R U}$.

2.5.1.2.3 Properties of Decision Directed Linear Estimators

Proposition 2.30 (Linearity of Decision Directed Linear Estimators) Decision directed linear estimators are linear because they fit the form

$$\widehat{\beta} = \mathbf{S y} \tag{2.70}$$

where $\mathbf{S}$ in this case is

$$\mathbf{S} = \left(\mathbf{X^T R X}\right)^{-1} \mathbf{X^T R} \tag{2.71}$$

Proposition 2.31 (Bias of Decision Directed Linear Estimators) If

$$\mathcal{E}\left\{\mathbf{w} | \mathbf{X}, \mathbf{U}\right\} = \mathbf{0} \tag{2.72}$$

then

$$\mathcal{E}\left\{\widehat{\beta}\right\} = \mathcal{E}\left\{\left(\mathbf{X^T R X}\right)^{-1} \mathbf{X^T R U}\right\} \mathbf{b} \tag{2.73}$$

Proof. Substitute (2.62) into (2.65) and simplify to get

$$\begin{aligned}\widehat{\beta} &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R}\left(\mathbf{Ub}+\mathbf{w}\right)\\ &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T RUb}+\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T Rw}\end{aligned}$$

Taking the expectation conditioned upon $\mathbf{X},\mathbf{U}$ gives

$$\begin{aligned}\mathcal{E}\left\{\widehat{\beta}\mid\mathbf{X},\mathbf{U}\right\} &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T RUb}+\mathcal{E}\left\{\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T Rw}\mid\mathbf{X},\mathbf{U}\right\}\\ &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T RUb}+\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R}\mathcal{E}\left\{\mathbf{w}\mid\mathbf{X},\mathbf{U}\right\}\\ &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T RUb}\end{aligned}$$

Again taking the expectation yields

$$\begin{aligned}\mathcal{E}\left\{\widehat{\beta}\right\} &= \mathcal{E}\left\{\mathcal{E}\left\{\widehat{\beta}\mid\mathbf{X},\mathbf{U}\right\}\right\}\\ &= \mathcal{E}\left\{\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T RU}\right\}\mathbf{b}\end{aligned}$$

□

Proposition 2.32 (Unbiasness of Ideal Decision Weighted Linear Estimators) If

$$\mathcal{E}\left\{\mathbf{w}|\mathbf{X},\mathbf{U}\right\}=0 \tag{2.74}$$

and $\mathbf{R}$ is chosen such that $\mathbf{X^T RX}=\mathbf{X^T RU}$ then

$$\mathcal{E}\left\{\widehat{\beta}\right\}=\mathbf{b} \tag{2.75}$$

Proof. This proposition follows from (2.73).

Proposition 2.33 (Covariance of Decision Directed Linear Estimators) If

$$\mathcal{E}\left\{\mathbf{w}|\mathbf{X},\mathbf{U}\right\}=\mathbf{0}$$

Then the covariance of (2.65) is

$$\begin{aligned}\mathbf{cov}\left\{\widehat{\beta}\right\} &= \mathcal{E}\left\{\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\} &(2.76)\\ &= \mathbf{cov}\left\{\mathbf{SUb}\right\}+\mathcal{E}\left\{\mathbf{SVS^T}\right\} &(2.77)\end{aligned}$$

where

$$\begin{aligned}\mathbf{S} &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R} && (2.78)\\ \mathbf{V} &= \mathcal{E}\left\{\mathbf{w w^T} \mid \mathbf{X}, \mathbf{U}\right\} && (2.79)\end{aligned}$$

Proof. Rederive (2.48) by replacing $\mathbf{Q}$ with $\mathbf{S}$ and adding $\mathbf{X}$ to the conditional expectation to get

$$\begin{aligned}\mathcal{E}\left\{\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\} &= \mathcal{E}\left\{\left(\mathbf{SUb}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\mathbf{SUb}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\}\\ &\quad +\mathcal{E}\left\{\mathbf{SVS^T}\right\} && (2.80)\end{aligned}$$

Substituting (2.78) and (2.73) gives

$$\begin{aligned}\mathbf{cov}\left\{\widehat{\beta}\right\} &= \mathcal{E}\left\{\left(\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R U b}-\mathcal{E}\left\{\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R U}\right\}\mathbf{b}\right)\right.\\ &\quad \left.\left(\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R U b}-\mathcal{E}\left\{\left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R U}\right\}\mathbf{b}\right)^{\mathbf{T}}\right\}\\ &\quad +\mathcal{E}\left\{\mathbf{SVS^T}\right\}\\ &= \mathbf{cov}\left\{\mathbf{SUb}\right\}+\mathcal{E}\left\{\mathbf{SVS^T}\right\}\end{aligned}$$

□

Proposition 2.34 (Covariance of Ideal Decision Weighted Linear Estimators) If

$$\mathcal{E}\left\{\mathbf{w}|\mathbf{X}, \mathbf{U}\right\} = \mathbf{0}$$

and $\mathbf{R}$ is chosen such that $\mathbf{X^T R X} = \mathbf{X^T R U}$ then then the covariance of (2.65) is

$$\begin{aligned}\mathbf{cov}\left\{\widehat{\beta}\right\} &= \mathcal{E}\left\{\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)\left(\widehat{\beta}-\mathcal{E}\left\{\widehat{\beta}\right\}\right)^{\mathbf{T}}\right\}\\ &= \mathcal{E}\left\{\mathbf{SVS^T}\right\}\end{aligned}$$

where

$$\begin{aligned}\mathbf{S} &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R}\\ \mathbf{V} &= \mathcal{E}\left\{\mathbf{w w^T} \mid \mathbf{X}, \mathbf{U}\right\}\end{aligned}$$

Proof. From Prop. 2.33

$$\mathbf{cov}\left\{\widehat{\beta}\right\} = \mathbf{cov}\left\{\mathbf{SUb}\right\} + \mathcal{E}\left\{\mathbf{SVS^T}\right\} \tag{2.81}$$

But

$$\begin{aligned}\mathbf{SU} &= \left(\mathbf{X^T R X}\right)^{-1}\mathbf{X^T R U} \\ &= \mathbf{I}\end{aligned}$$

by assumption, where $\mathbf{I}$ is the identity matrix.

□

2.5.1.2.4 More on Ideal Decision Weighted Linear Estimators As mentioned previously, ideal decision weighted linear estimators are a subclass of decision directed linear estimators that have the property $\mathbf{X^T R X} = \mathbf{X^T R U}$. Furthermore, implicit to this assumption is that $\mathbf{X^T R X}$ must be invertible. Are these two conflicting assumptions? Furthermore, what properties of $\mathbf{R}$ are required for $\mathbf{X^T R X} = \mathbf{X^T R U}$? The answer to the first question is no, and we will propose an example of $\mathbf{R}$ that satisfies both conditions. The answer to the second question is answered in the following proposition and theorem.

Proposition 2.35 Let[1] $\mathbf{R} \in \mathcal{R}^{NxN}$ and $\mathbf{X}, \mathbf{U} \in \mathcal{R}^{NxM}$ such that $\mathbf{X} = \mathbf{U}$ except for a single element, say $x_{ij} \neq u_{ij}$ for some row i and column j. Then $\mathbf{X^T R X} = \mathbf{X^T R U}$ if and only if the ith column of $\mathbf{R}$ is orthogonal[2] to each column in $\mathbf{X}$.

[1] $\mathcal{R}^{NxM}$ denotes the vector space of all real N by M matrices; $\mathcal{R}^N$ denotes the vector space of all real N by 1 vectors

[2] Two vectors $\mathbf{x}, \mathbf{y} \in \mathcal{R}^N$ are orthogonal if $\mathbf{x}^T\mathbf{y} = \mathbf{y}^T\mathbf{x} = \sum_{k=1}^{N} x_k y_k = 0$.

Proof. Suppose $x_{ij} - u_{ij} = a_{ij} \neq 0$ for a fixed row i and column j. Then

$$\mathbf{X^T R}\left(\mathbf{X}-\mathbf{U}\right) = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{N1} \\ x_{12} & x_{22} & & \\ \vdots & & \ddots & \vdots \\ x_{1M} & \cdots & & x_{NM} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ r_{21} & r_{22} & & \\ \vdots & & \ddots & \vdots \\ r_{N1} & \cdots & & r_{NN} \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots & & & \vdots \\ 0 & & 0 & 0 & 0 & & 0 \\ 0 & \cdots & 0 & a_{ij} & 0 & \cdots & 0 \\ 0 & & 0 & 0 & 0 & & 0 \\ \vdots & & & \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$= a_{ij} \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{N1} \\ x_{12} & x_{22} & & \\ \vdots & & \ddots & \vdots \\ x_{1M} & \cdots & & x_{NM} \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & r_{1i} & 0 & \cdots & 0 \\ \vdots & & \vdots & r_{2i} & \vdots & & \vdots \\ 0 & & 0 & & 0 & & 0 \\ 0 & \cdots & 0 & \vdots & 0 & \cdots & 0 \\ 0 & & 0 & & 0 & & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & r_{Ni} & 0 & \cdots & 0 \end{bmatrix}$$

$$= a_{ij} \begin{bmatrix} 0 & \cdots & 0 & \sum_{k=1}^{N} r_{ki}x_{k1} & 0 & \cdots & 0 \\ \vdots & & \vdots & \sum_{k=1}^{N} r_{ki}x_{k2} & \vdots & & \vdots \\ 0 & & 0 & & 0 & & 0 \\ 0 & \cdots & 0 & \vdots & 0 & \cdots & 0 \\ 0 & & 0 & & 0 & & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \sum_{k=1}^{N} r_{ki}x_{kM} & 0 & \cdots & 0 \end{bmatrix}$$

The above quantity equals zero if and only if the ith column of $\mathbf{R}$ is orthogonal to each column of $\mathbf{X}$.

□

We now generalize this result in the following theorem.

Theorem 2.36 Let $\mathbf{R} \in \mathcal{R}^{\mathbf{NxN}}$ and $\mathbf{X}, \mathbf{U} \in \mathcal{R}^{\mathbf{NxM}}$ such that $\mathbf{X} = \mathbf{U}$ except for q elements with row / column coordinates $(r_1, c_1), \ldots, (r_q, c_q)$. If the $\{r_1, \ldots, r_q\}$ columns of $\mathbf{R}$ are orthogonal to every column of $\mathbf{X}$, then $\mathbf{X^T RX} = \mathbf{X^T RU}$.

Proof. Let $\mathbf{D} = \mathbf{X^T R}(\mathbf{X} - \mathbf{U})$. Then the ith row, jth column element is

$$d_{ij} = \sum_{z=1}^{N} \sum_{l=1}^{N} x_{li} r_{lz} \left(x_{zj} - u_{zj}\right) \tag{2.82}$$

Now suppose that in the jth column, $k \leq q$ elements with coordinates $(\rho_1, j), \ldots, (\rho_k, j)$ in $\mathbf{X}$ do not equal the corresponding elements in $\mathbf{U}$. Define $R_j = \{\rho_1, \ldots, \rho_k\}$ as the set of row coordinates in the jth column where $\mathbf{X}$ and $\mathbf{U}$ differ. Then (2.82) becomes

$$\begin{aligned} d_{ij} &= \sum_{z \in R_j} \sum_{l=1}^{N} x_{li} r_{lz} \left(x_{zj} - u_{zj}\right) \\ &= \sum_{z \in R_j} \left(x_{zj} - u_{zj}\right) \sum_{l=1}^{N} x_{li} r_{lz} \\ &= 0 \end{aligned}$$

by assumption that the $\{\rho_1, \ldots, \rho_k\} \subseteq \{r_1, \ldots, r_q\}$ columns of $\mathbf{R}$ are orthogonal to every column of $\mathbf{X}$.

□

We will now propose a weighting matrix $\mathbf{R}$ that satisfies $\mathbf{X^T R X} = \mathbf{X^T R U}$, and under certain conditions, $\mathbf{X^T R X}$ non-singular[1]. But first, we will need the following lemmas concerning dimensionality and singularity.

Lemma 2.37 Let B and A be linear transformations from vector spaces V to W and W to Z, respectively. Let F be the field over which V, W, and Z are defined. Then the rank[2] of the composition $A \circ B$ is less than or equal to the minimum of the rank of A and the rank of B. In other words,

$$rank(A \circ B) \leq min\{rank(A), rank(B)\}$$

Proof. Because $B(V) \subset W$, we get $A(B(V)) \subset A(W)$. By ([14], pg. 182), the dimension of $A(B(V))$ is less than or equal to the dimension of $A(W)$, i.e. $rank(A \circ B) \leq rank(A)$. Now let $q = rank(B)$. Then by definition, $B(V)$ has a basis of q elements, say $\mathbf{w}_1, \ldots, \mathbf{w}_q \in W$. Then for $\mathbf{x} \in V$, $B(\mathbf{x}) = x_1 \mathbf{w}_1 + \ldots + x_q \mathbf{w}_q$ for some $x_i \in F, i = 1, \ldots, q$. Hence, $A(B(\mathbf{x})) =$

[1] A non-singular mapping is an invertible mapping.

[2] The rank of linear transformation is the dimension of its range or image ([29], pg. 228). We will denote the image or range of a linear transformation B from vector spaces V to W as $B(V) = \{w \in W | B(v) = w \text{ for some } v \in V\}$. Notice that $B(V) \subset W$, and consequently $rank(B) \leq dim(W)$ ([14], pg. 182).

$x_1 A(\mathbf{w}_1) + \ldots + x_q A(\mathbf{w}_q)$. Therefore, $A(B(V))$ has a basis of at most q elements, implying $rank(A \circ B) \leq rank(B)$.

□

Lemma 2.38 Let B and A be linear transformations from vector spaces V to W and W to Z, respectively. If $rank(B) < dim(Z)$ then $A \circ B$ is singular.

Proof. By Lemma 2.37, $rank(A \circ B) \leq rank(B) < dim(Z)$. Hence, $A \circ B$ is not an onto[1] mapping, and therefore is not invertible.

□

Lemma 2.39 Let B and A be linear transformations from vector spaces V to W and W to Z, respectively. If $dim(W) < dim(Z)$ then $A \circ B$ is singular.

Proof. Because $B(V) \subset W$, we have $rank(B) \leq dim(W) < dim(Z)$. Thus, by Lemma 2.38 $A \circ B$ is singular.

□

Proposition 2.40 Let $\mathbf{X}, \mathbf{U} \in \mathcal{R}^{\mathbf{NxM}}$ such that $\mathbf{X} = \mathbf{U}$ except for q elements with row / column coordinates $(r_1, c_1), \ldots, (r_q, c_q)$. Let $\mathbf{R}$ be an NxN identity matrix with the $\{r_1, \ldots, r_q\}$ columns set to zero. A necessary condition for $\left(\mathbf{X^T R X}\right)^{-1}$ to exist is $N \geq M + Q$, where Q is the number of unique elements of $\{r_1, \ldots, r_q\}$. If $\sum_{n=1}^{N} \delta_n \mathbf{x}_n \mathbf{x}_n^T$ is invertible, where $\delta_n = 0$ if $n \in \{r_1, \ldots, r_q\}$ and 1 otherwise, and $\mathbf{x}_n^T$ is the nth row of $\mathbf{X}$, then $\mathbf{X^T R X}$ is invertible.

Proof. First, if $N < M$, then by Lemma 2.39 $\mathbf{X^T R X}$ is singular because $\mathbf{X^T R}$ is a linear transformation from $\mathcal{R}^N$ to $\mathcal{R}^M$ and $\mathbf{X}$ is a linear transformation from $\mathcal{R}^M$ to $\mathcal{R}^N$. But if $rank(R) = N - Q$, by Lemma 2.38 and 2.37, $N - Q$ must be greater than or equal to M for $\mathbf{X^T R X}$ to be invertible.

We can write $\mathbf{X^T R X}$ as

$$\mathbf{X^T R X} = \sum_{n=1}^{N} \delta_n \mathbf{x}_n \mathbf{x}_n^T$$

where $\delta_n = 0$ if $n \in \{r_1, \ldots, r_q\}$ and 1 otherwise, and $\mathbf{x}_n^T$ is the nth row of $\mathbf{X}$. Hence, if in the rows that decision errors did not occur, the input sequence is rich enough so that this summation is invertible, then $\mathbf{X^T R X}$ is invertible.

□

[1] An onto mapping $f : X \to Y$ has the property that for every $y \in Y$ there exists an $x \in X$ such that $f(x) = y$. If the image of f, or equivalently the rank of f, has a dimension smaller than the dimension of Y, then the mapping is not onto. A mapping must be one-to-one and onto to be invertible ([14], pg. 15)

2.5.1.3 Extension to Complex Signals and Systems

Up to this point we have only considered adaptive algorithms for real signals and systems. These previous results extend nicely to complex signals and systems by representing complex numbers as elements of $\mathcal{R}^2$. For example, from Figure 2.6 we redefine the following relationships:

$$y(n) = b_1 u_1(n) + \cdots + b_M u_M(n) + w(n) \tag{2.83}$$

$$\widehat{y}(n) = \beta_1 u_1(n) + \cdots + \beta_M u_M(n) \tag{2.84}$$

$$e(n) = y(n) - \widehat{y}(n) = y(n) - (\beta_1 u_1(n) + \cdots + \beta_M u_M(n)) \tag{2.85}$$

where $u_i(n),\ i = 1, \ldots, M$ are complex-valued discrete-time random processes, $b_1, \cdots, b_M, \beta_1, \cdots, \beta_M \in \mathcal{C}$ where $\mathcal{C}$ is the set of complex numbers, M is a positive integer, and $w(n)$ is a mean zero complex-valued discrete-time random process. We can then write the real part of $y(n)$ as

$$\begin{aligned}
Re\{y(n)\} &= Re\{b_1 u_1(n)\} + \cdots + Re\{b_M u_M(n)\} + Re\{w(n)\} \\
&= Re\{b_1\} Re\{u_1(n)\} - Im\{b_1\} Im\{u_1(n)\} + \cdots \\
&\quad + Re\{b_M\} Re\{u_M(n)\} - Im\{b_M\} Im\{u_M(n)\} + Re\{w(n)\} \\
&= \left[\begin{array}{ccc} Re\{u_1(n)\} & \cdots & Re\{u_M(n)\} \end{array} \right] \left[\begin{array}{c} Re\{b_1\} \\ \vdots \\ Re\{b_M\} \end{array} \right] \\
&\quad - \left[\begin{array}{ccc} Im\{u_1(n)\} & \cdots & Im\{u_M(n)\} \end{array} \right] \left[\begin{array}{c} Im\{b_1\} \\ \cdots \\ Im\{b_M\} \end{array} \right] + Re\{w(n)\}
\end{aligned}$$

which further reduces to

$$Re\{y(n)\} = \left[\begin{array}{cccccc} Re\{u_1(n)\} & \cdots & Re\{u_M(n)\} & -Im\{u_1(n)\} & \cdots & -Im\{u_M(n)\} \end{array} \right] \left[\begin{array}{c} Re\{b_1\} \\ \vdots \\ Re\{b_M\} \\ Im\{b_1\} \\ \cdots \\ Im\{b_M\} \end{array} \right] + Re\{w(n)\}$$

Similarly, the imaginary part of $y(n)$ is

$$
\begin{aligned}
Im\{y(n)\} &= Im\{b_1 u_1(n)\} + \cdots + Im\{b_M u_M(n)\} + Im\{w(n)\} \\
&= Re\{b_1\} Im\{u_1(n)\} + Im\{b_1\} Re\{u_1(n)\} + \cdots \\
&\quad + Re\{b_M\} Im\{u_M(n)\} + Im\{b_M\} Re\{u_M(n)\} + Im\{w(n)\} \\
&= \left[\begin{array}{ccc} Im\{u_1(n)\} & \cdots & Im\{u_M(n)\} \end{array} \right] + \left[\begin{array}{c} Re\{b_1\} \\ \vdots \\ Re\{b_M\} \end{array} \right] \\
&\quad \left[\begin{array}{ccc} Re\{u_1(n)\} & \cdots & Re\{u_M(n)\} \end{array} \right] \left[\begin{array}{c} Im\{b_1\} \\ \cdots \\ Im\{b_M\} \end{array} \right] + Im\{w(n)\} \\
&= \left[\begin{array}{cccccc} Im\{u_1(n)\} & \cdots & Im\{u_M(n)\} & Re\{u_1(n)\} & \cdots & Re\{u_M(n)\} \end{array} \right] \left[\begin{array}{c} Re\{b_1\} \\ \vdots \\ Re\{b_M\} \\ Im\{b_1\} \\ \cdots \\ Im\{b_M\} \end{array} \right] \\
&\quad + Im\{w(n)\}
\end{aligned}
$$

Writing $y(n)$ as an element of $\mathcal{R}^2$ gives

$$
\left[\begin{array}{c} Re\{y(n)\} \\ Im\{y(n)\} \end{array} \right] = \left[\begin{array}{cccccc} Re\{u_1(n)\} & \cdots & Re\{u_M(n)\} & -Im\{u_1(n)\} & \cdots & -Im\{u_M(n)\} \\ Im\{u_1(n)\} & \cdots & Im\{u_M(n)\} & Re\{u_1(n)\} & \cdots & Re\{u_M(n)\} \end{array} \right] \left[\begin{array}{c} Re\{b_1\} \\ \vdots \\ Re\{b_M\} \\ Im\{b_1\} \\ \cdots \\ Im\{b_M\} \end{array} \right] + \left[\begin{array}{c} Re\{w(n)\} \\ Im\{w(n)\} \end{array} \right] \tag{2.86}
$$

We can now extend this result to a collection of data.

$$\begin{bmatrix} Re\{y(1)\} \\ \vdots \\ Re\{y(N)\} \\ Im\{y(1)\} \\ \vdots \\ Im\{y(N)\} \end{bmatrix} = \begin{bmatrix} Re\{u_1(1)\} & \cdots & Re\{u_M(1)\} & -Im\{u_1(1)\} & \cdots & -Im\{u_M(1)\} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Re\{u_1(N)\} & \cdots & Re\{u_M(N)\} & -Im\{u_1(N)\} & \cdots & -Im\{u_M(N)\} \\ Im\{u_1(1)\} & \cdots & Im\{u_M(1)\} & Re\{u_1(1)\} & \cdots & Re\{u_M(1)\} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Im\{u_1(N)\} & \cdots & Im\{u_M(N)\} & Re\{u_1(N)\} & \cdots & Re\{u_M(N)\} \end{bmatrix} \begin{bmatrix} Re\{b_1\} \\ \vdots \\ Re\{b_M\} \\ Im\{b_1\} \\ \vdots \\ Im\{b_M\} \end{bmatrix} + \begin{bmatrix} Re\{w(1)\} \\ \vdots \\ Re\{w(N)\} \\ Im\{w(1)\} \\ \vdots \\ Im\{w(N)\} \end{bmatrix} \tag{2.87}$$

If once again we let

$$\begin{aligned} \mathbf{y} &= \begin{bmatrix} y(1) & y(2) & \cdots & y(N) \end{bmatrix}^T \\ \mathbf{w} &= \begin{bmatrix} w(1) & w(2) & \cdots & w(N) \end{bmatrix}^T \\ \mathbf{e} &= \begin{bmatrix} e(1) & e(2) & \cdots & e(N) \end{bmatrix}^T \\ \mathbf{b} &= \begin{bmatrix} b_1 & b_2 & \cdots & b_M \end{bmatrix}^T \\ \beta &= \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_M \end{bmatrix}^T \\ \mathbf{U} &= \begin{bmatrix} u_1(1) & \cdots & u_M(1) \\ \vdots & \ddots & \vdots \\ u_1(N) & \cdots & u_M(N) \end{bmatrix} \\ \mathbf{X} &= \begin{bmatrix} x_1(1) & \cdots & x_M(1) \\ \vdots & \ddots & \vdots \\ x_1(N) & \cdots & x_M(N) \end{bmatrix} \end{aligned}$$

then we can write (2.87) as

$$\begin{bmatrix} Re\{\mathbf{y}\} \\ Im\{\mathbf{y}\} \end{bmatrix} = \begin{bmatrix} Re\{U\} & -Im\{U\} \\ Im\{U\} & Re\{U\} \end{bmatrix} \begin{bmatrix} Re\{b\} \\ Im\{b\} \end{bmatrix} + \begin{bmatrix} Re\{w\} \\ Im\{w\} \end{bmatrix} \tag{2.88}$$

We extend this result in the following proposition to rephrase the training sequence and blind estimation problems.

Proposition 2.41 (The Complex Estimation Problem) Define[1]

$$\begin{aligned}
\widetilde{\mathbf{y}} &= \begin{bmatrix} Re\{\mathbf{y}\} \\ Im\{\mathbf{y}\} \end{bmatrix} \\
\widetilde{\mathbf{w}} &= \begin{bmatrix} Re\{\mathbf{w}\} \\ Im\{\mathbf{w}\} \end{bmatrix} \\
\widetilde{\mathbf{e}} &= \begin{bmatrix} Re\{\mathbf{e}\} \\ Im\{\mathbf{e}\} \end{bmatrix} \\
\widetilde{\mathbf{b}} &= \begin{bmatrix} Re\{\mathbf{b}\} \\ Im\{\mathbf{b}\} \end{bmatrix} \\
\widetilde{\beta} &= \begin{bmatrix} Re\{\beta\} \\ Im\{\beta\} \end{bmatrix} \\
\widetilde{\mathbf{U}} &= \begin{bmatrix} Re\{\mathbf{U}\} & -Im\{\mathbf{U}\} \\ Im\{\mathbf{U}\} & Re\{\mathbf{U}\} \end{bmatrix} \\
\widetilde{\mathbf{X}} &= \begin{bmatrix} Re\{\mathbf{X}\} & -Im\{\mathbf{X}\} \\ Im\{\mathbf{X}\} & Re\{\mathbf{X}\} \end{bmatrix}
\end{aligned}$$

with

$$\widetilde{\mathbf{y}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{b}} + \widetilde{\mathbf{w}} \tag{2.89}$$

The error for the training sequence estimation problem is

$$\widetilde{\mathbf{e}} = \widetilde{\mathbf{y}} - \widetilde{\mathbf{U}}\widetilde{\beta} \tag{2.90}$$

While that for the decision directed estimation is

$$\widetilde{\mathbf{e}} = \widetilde{\mathbf{y}} - \widetilde{\mathbf{X}}\widetilde{\beta} \tag{2.91}$$

The redefined error or loss function is then

$$\widetilde{\mathbf{J}}(\widetilde{\beta}) = \widetilde{\mathbf{e}}^{\mathbf{T}}\widetilde{\mathbf{R}}\widetilde{\mathbf{e}} \tag{2.92}$$

[1]The tilde notation in this proposition does not denote the complex envelope as in previous sections

where $\widetilde{\mathbf{R}}$ is a $2N$ x $2N$ matrix of weighting coefficients. Using these modified vectors and matrices, the previous lemmas and propositions on linear estimators apply.

2.5.2 Least Mean Squares (LMS) Estimators

We now explore an alternative to the linear estimators presented in the previous section. This alternative is the stochastic gradient or least mean squares (LMS) method ([34],pg. 77), [15], [38], ([8], pp. 155-157, pg. 241). The training sequence and decision directed problem statements are identical to their linear estimator counterparts except only the most recent data point is of interest.

2.5.2.1 Training Sequence Estimation

2.5.2.1.1 The System Identification Problem From Figure 2.6 define the following relationships:

$$y(n) = b_1 u_1(n) + \cdots + b_M u_M(n) + w(n) \tag{2.93}$$

$$\widehat{y}(n) = \beta_1 u_1(n) + \cdots + \beta_M u_M(n) \tag{2.94}$$

$$e(n) = y(n) - \widehat{y}(n) = y(n) - (\beta_1 u_1(n) + \cdots + \beta_M u_M(n)) \tag{2.95}$$

where $u_i(n),\ i = 1, \ldots, M$ are a real-valued discrete-time random processes, $b_1, \cdots, b_M, \beta_1, \cdots, \beta_M \in \mathcal{R}$, M is a positive integer, and $w(n)$ is a mean zero real-valued discrete-time random process. We also define

$$\mathbf{b} = \left[\begin{array}{cccc} b_1 & b_2 & \cdots & b_M \end{array} \right]^T \tag{2.96}$$

$$\beta = \left[\begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \right]^T \tag{2.97}$$

$$\mathbf{u}(n) = \left[\begin{array}{cccc} u_1(n) & u_2(n) & \cdots & u_M(n) \end{array} \right]^T \tag{2.98}$$

$$\tag{2.99}$$

Thus,

$$y(n) = \mathbf{u}(n)^T \mathbf{b} + w(n) \tag{2.100}$$

$$\widehat{y}(n) = \mathbf{u}(n)^T \beta \tag{2.101}$$

$$e(n) = y(n) - \widehat{y}(n) = y(n) - \mathbf{u}(n)^T \beta \tag{2.102}$$

The error or loss function is defined as

$$J(\beta) = \rho_n e(n)^T e(n) \tag{2.103}$$

where $\rho_n \in \mathcal{R}$ is a weighting sequence. Although $e(n)$ is a scalar, we will treat it as a vector as it will be when we extend these results to complex signals and systems.

2.5.2.1.2 Estimator Derivation The LMS algorithm is a steepest decent algorithm, meaning it adjusts $\widehat{\beta}$ in the direction of maximum change of $J(\beta)$. The algorithm has the form

$$\widehat{\beta}_n = \widehat{\beta}_{n-1} - \frac{1}{2}\mu \left. \frac{\partial J}{\partial \beta} \right|_{\beta=\widehat{\beta}_{n-1}} \tag{2.104}$$

We present the final form of the LMS algorithm in the following theorem.

Theorem 2.42 The LMS algorithm satisfying (2.104) is

$$\widehat{\beta}_n = \widehat{\beta}_{n-1} + \mu\rho_n \mathbf{u}(n)e(n) \tag{2.105}$$

$$e(n) = y(n) - \mathbf{u}(n)^T \widehat{\beta}_{n-1} \tag{2.106}$$

Proof. The gradient was previously found in (2.28) where here $\mathbf{U} = \mathbf{u}(n)^T$. Substituting this into (2.104) and factoring out $\rho_n \mathbf{u}(n)$ gives the desired result.

□

2.5.2.1.3 Properties of Training Sequence LMS Estimators

Definition 2.43 (Asymptotic Unbiasness) An estimator $\widehat{\beta}_n$ is asymptotically unbiased if

$$\lim_{n\to+\infty} \mathcal{E}\left\{\widehat{\beta}_n\right\} = \mathbf{b} \tag{2.107}$$

where $\mathbf{b}$ is the true parameter.

Proposition 2.44 (Asymptotic Unbiasness of the LMS Estimator) Let $\mathbf{R}_\mathbf{u} = \mathcal{E}\left\{\mathbf{u}(n)\mathbf{u}(n)^T\right\}$ be the auto-correlation matrix of the $\mathbf{u}(n)$. Assume that $\mathbf{u}(n)$ is white and ρ_n does not depend on $\mathbf{u}(n)$. Restrict μ and λ_{min} to the ranges

$$0 < \lambda_{min} \tag{2.108}$$

$$\mu < 2/\lambda_{max} \tag{2.109}$$

where λ_{min} and λ_{max} are the minimum and maximum eigenvalues of $\mathbf{R_u}$, respectively. Also restrict ρ_n to

$$0 \leq \rho_n \leq 1 \tag{2.110}$$

for all n with ρ_n not converging to zero. If $\{u(n)\}$ and $\{w(n)\}$ are statistically independent, with $\mathcal{E}\{w(n)\} = 0$, then the LMS estimate in (2.105) is asymptotically unbiased.

Proof. Substituting (2.100) into the LMS algorithm gives

$$\begin{aligned}
\widehat{\beta}_n &= \widehat{\beta}_{n-1} + \mu\rho_n \mathbf{u}(n)\left(\mathbf{u}(n)^T\mathbf{b} + w(n) - \mathbf{u}(n)^T\widehat{\beta}_{n-1}\right) \\
&= \widehat{\beta}_{n-1} - \mu\rho_n \mathbf{u}(n)\mathbf{u}(n)^T\left(\widehat{\beta}_{n-1} - \mathbf{b}\right) + \mu\rho_n \mathbf{u}(n)w(n)
\end{aligned}$$

Subtracting $\mathbf{b}$ from both sides gives

$$\left(\widehat{\beta}_n - \mathbf{b}\right) = \left(\widehat{\beta}_{n-1} - \mathbf{b}\right) - \mu\rho_n \mathbf{u}(n)\mathbf{u}(n)^T\left(\widehat{\beta}_{n-1} - \mathbf{b}\right) + \mu\rho_n \mathbf{u}(n)w(n) \tag{2.111}$$

Take the expectation of both sides conditioned upon $\{\mathbf{u}(k), w(k), k < n\}$ to get

$$\begin{aligned}
\mathcal{E}\left\{\widehat{\beta}_n - \mathbf{b} \mid \mathbf{u}(k), w(k), k < n\right\} &= \left(\widehat{\beta}_{n-1} - \mathbf{b}\right) - \mu\rho_n \mathcal{E}\left\{\mathbf{u}(n)\mathbf{u}(n)^T \mid \mathbf{u}(k), w(k), k < n\right\}\left(\widehat{\beta}_{n-1} - \mathbf{b}\right) \\
&\quad + \mu\rho_n \mathcal{E}\left\{\mathbf{u}(n)w(n) \mid \mathbf{u}(k), w(k), k < n\right\} \\
&= \left(\widehat{\beta}_{n-1} - \mathbf{b}\right) - \mu\rho_n \mathbf{R_u}\left(\widehat{\beta}_{n-1} - \mathbf{b}\right)
\end{aligned}$$

Take the expectation again to get

$$\mathcal{E}\left\{\widehat{\beta}_n - \mathbf{b}\right\} = \mathcal{E}\left\{\widehat{\beta}_{n-1} - \mathbf{b}\right\} - \mu\rho_n \mathbf{R_u}\mathcal{E}\left\{\widehat{\beta}_{n-1} - \mathbf{b}\right\} \tag{2.112}$$

$$\mathbf{C}_n = \mathbf{C}_{n-1} - \mu\rho_n \mathbf{R_u}\mathbf{C}_{n-1} \tag{2.113}$$

where we define $\mathbf{C}_n = \mathcal{E}\left\{\widehat{\beta}_n - \mathbf{b}\right\}$. Then the convergence of $\widehat{\beta}_n$ to $\mathbf{b}$ in expectation is equivalent to the convergence of $\mathbf{C}_n$ to zero. We will show this convergence by spectrally decomposing $\mathbf{R_u}$ then examining each component of $\mathbf{C}_n$.

For simplicity[1], let $\mathbf{R_u}$ have distinct eigenvalues and corresponding unit eigenvectors $\lambda_1, \ldots, \lambda_M$ and $\mathbf{v_1}, \ldots, \mathbf{v_M}$, respectively. Define $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_M] \in \mathcal{R}^{MxM}$ and $\Lambda = diag(\lambda_1, \ldots, \lambda_M) \in \mathcal{R}^{MxM}$. Because $\mathbf{R_u}$ is symmetric, it has an eigenvalue or spectral decomposition $\mathbf{R_u} = \mathbf{V}\Lambda\mathbf{V}^T =$

[1]While it may be true that $\mathbf{R_u}$ is not guaranteed to have distinct eigenvalues, the result of the proposition is still true [36], [37], [9].

$\sum_{k=1}^{M} \lambda_k \mathbf{v}_k \mathbf{v}_k^T$ ([34], pg. 285).

$$\mathbf{C}_n = \mathbf{C}_{n-1} - \mu\rho_n \sum_{k=1}^{M} \lambda_k \mathbf{v}_k \mathbf{v}_k^T \mathbf{C}_{n-1} \tag{2.114}$$

Because the $\lambda_1, \ldots, \lambda_M$ are distinct, $\mathbf{v_1}, \ldots, \mathbf{v_M}$ are orthogonal and form a basis for $\mathcal{R}^M$ ([21], pg. 410), ([34], pg. 285). Define $D_{r,n} = \mathbf{v}_r^T \mathbf{C}_n$ to be the component of $\mathbf{C}_n$ in the direction of $\mathbf{v}_r$, $r = 1, \ldots, M$. Then

$$\begin{aligned}
\mathbf{v}_r^T \mathbf{C}_n &= \mathbf{v}_r^T \mathbf{C}_{n-1} - \mu\rho_n \sum_{k=1}^{M} \lambda_k \mathbf{v}_r^T \mathbf{v}_k \mathbf{v}_k^T \mathbf{C}_{n-1} \\
&= \mathbf{v}_r^T \mathbf{C}_{n-1} - \mu\rho_n \lambda_r \mathbf{v}_r^T \mathbf{C}_{n-1} \\
D_{r,n} &= D_{r,n-1} - \mu\rho_n \lambda_r D_{r,n-1} \\
&= (1 - \mu\rho_n \lambda_r)\, D_{r,n-1}
\end{aligned}$$

Writing this iteratively gives

$$\begin{aligned}
D_{r,2} &= (1 - \mu\rho_2 \lambda_r)\, D_{r,1} \\
D_{r,3} &= (1 - \mu\rho_3 \lambda_r)\, D_{r,2} = (1 - \mu\rho_3 \lambda_r)\,(1 - \mu\rho_2 \lambda_r)\, D_{r,1} \\
\vdots &= \vdots \\
D_{r,n} &= (1 - \mu\rho_n \lambda_r) \cdots (1 - \mu\rho_2 \lambda_r)\, D_{r,1}
\end{aligned}$$

A sufficient condition for the convergence of $D_{r,n}$ to zero is that for all n, $|1 - \mu\rho_n\lambda_r| < 1$, and $|1 - \mu\rho_n\lambda_r|$ does not converge to 1. This condition is met by requiring $0 < \lambda_{min}$, $\mu < 2/\lambda_{max}$, $0 \leq \rho_n \leq 1$, and ρ_n not converging to zero. Under these conditions each component of $\mathbf{C}_n$ converges to zero.

□

Remark 2.45 Other properties of the LMS algorithm, such as covariance and tightness, are derived in [34].

2.5.2.2 Decision Directed Estimation

The decision directed LMS algorithm is like other decision directed algorithms in the respect that it uses detected symbols to reconstruct the transmitted sequence. Its form is identical to that of (2.105) with the exception that $\mathbf{u}(n)$ is replaced with its reconstructed version $\mathbf{x}(n)$.

Theorem 2.46 (Decision Directed LMS) The decision directed LMS algorithm is

$$\widehat{\beta}_n = \widehat{\beta}_{n-1} + \mu\rho_n \mathbf{x}(n)e(n) \tag{2.115}$$

$$e(n) = y(n) - \mathbf{x}(n)^T \widehat{\beta}_{n-1} \tag{2.116}$$

2.5.2.3 Extension to Complex Signals and Systems

Similar to the development in Section 2.5.1.3, the LMS can be extended to complex signals and systems by representing complex numbers as two-dimensional vectors. For an alternate derivation of the complex LMS algorithm see [38]. Using matrix and vector notation, the LMS algorithms extend to complex signals and systems as described in the following proposition.

Proposition 2.47 (The Complex Estimation Problem) Define[1]

$$\begin{aligned}
\widetilde{\mathbf{y}}(\mathbf{n}) &= \begin{bmatrix} Re\{\mathbf{y}(n)\} \\ Im\{\mathbf{y}(n)\} \end{bmatrix} \\
\widetilde{\mathbf{w}}(\mathbf{n}) &= \begin{bmatrix} Re\{\mathbf{w}(n)\} \\ Im\{\mathbf{w}(n)\} \end{bmatrix} \\
\widetilde{\mathbf{e}}(\mathbf{n}) &= \begin{bmatrix} Re\{\mathbf{e}(n)\} \\ Im\{\mathbf{e}(n)\} \end{bmatrix} \\
\widetilde{\mathbf{b}} &= \begin{bmatrix} Re\{\mathbf{b}\} \\ Im\{\mathbf{b}\} \end{bmatrix} \\
\widetilde{\beta}_{\mathbf{n}} &= \begin{bmatrix} Re\{\beta_n\} \\ Im\{\beta_n\} \end{bmatrix} \\
\widetilde{\mathbf{u}}(\mathbf{n}) &= \begin{bmatrix} Re\{\mathbf{u}(n)\} & Im\{\mathbf{u}(n)\} \\ -Im\{\mathbf{u}(n)\} & Re\{\mathbf{u}(n)\} \end{bmatrix} \\
\widetilde{\mathbf{x}}(\mathbf{n}) &= \begin{bmatrix} Re\{\mathbf{x}(n)\} & Im\{\mathbf{x}(n)\} \\ -Im\{\mathbf{x}(n)\} & Re\{\mathbf{x}(n)\} \end{bmatrix}
\end{aligned}$$

Then

$$\widetilde{\mathbf{y}}(\mathbf{n}) = \widetilde{\mathbf{u}}(\mathbf{n})^{\mathbf{T}}\widetilde{\mathbf{b}} + \widetilde{\mathbf{w}}(\mathbf{n}) \tag{2.117}$$

[1] The tilde notation in this proposition does not denote the complex envelope as in previous sections

The error for the training sequence estimation problem is

$$\widetilde{\mathbf{e}}(\mathbf{n}) = \widetilde{\mathbf{y}}(\mathbf{n}) - \widetilde{\mathbf{u}}(\mathbf{n})^{\mathbf{T}}\widetilde{\beta} \tag{2.118}$$

while that for the decision directed estimation is

$$\widetilde{\mathbf{e}}(\mathbf{n}) = \widetilde{\mathbf{y}}(\mathbf{n}) - \widetilde{\mathbf{x}}(\mathbf{n})^{\mathbf{T}}\widetilde{\beta} \tag{2.119}$$

The redefined error or loss function is then

$$\widetilde{\mathbf{J}}(\beta) = \rho_n \widetilde{\mathbf{e}}(\mathbf{n})^{\mathbf{T}}\widetilde{\mathbf{e}}(\mathbf{n}) \tag{2.120}$$

where $\rho_n \in \mathcal{R}$ is a weighting sequence. Using these modified vectors and matrices, the lemmas and propositions of this section on LMS estimators apply.

Proof. The proof follows from the development in Section 2.5.1.3 with $\mathbf{U}$ replaced with $\mathbf{u}(n)^T$.

□

2.5.3 Soft Decision Weighted Algorithms

The conditions for choosing the weights of the ideal decision weighted linear estimator is an idealization because it assumes perfect knowledge of the decision errors. We will use this idealization, however, as motivation for using functions of the receiver soft decisions as weights, reasoning that the receiver soft decisions reflect, in a crude sense, the accuracy of these decisions. This soft decision weighting scheme can then be used to weight the decision directed LMS and RLS algorithms, making them less susceptible to both fluctuations due to noise and decision errors (See Figures 4.1 to 4.9 for the effects of a BPSK decision error and noise on channel estimates). We describe in the following remark one possible way to derive these weights for an MPSK receiver.

Remark 2.48 Recall from Section 2.2.2 that the MPSK demodulation process produces an angle θ_i that is then compared to a set of known possible signal angle coordinates. The soft decision in this case is θ_i, while the hard decision (in terms of angle) is the angle ϕ_i closest to θ_i. For this decision, we define the weight p_i as the distance between ϕ_i and θ_i, normalized to the range $[0, 1]$,

$$p_i = 1 - \frac{|\phi_i - \theta_i|}{\pi/S} \tag{2.121}$$

where S is the number of possible symbol choices.

In Section 2.3 wireless communication channels were modeled as tapped delay line or transversal filters. Hence, for linear and LMS estimators, $\mathbf{u}(n)$ and $\mathbf{x}(n)$ take the form

$$\mathbf{u}(n) = \left[\begin{array}{cccc} u(n) & u(n-1) & \cdots & u(n-M+1) \end{array} \right]^T \tag{2.122}$$

$$\mathbf{x}(n) = \left[\begin{array}{cccc} x(n) & x(n-1) & \cdots & x(n-M+1) \end{array} \right]^T \tag{2.123}$$

Consequently, the algorithm weight at time n needs to reflect the accuracy of the past M decisions. One possible choice for this weight is

$$a_n = p_n p_{n-1} \cdots p_{n-M+1} \tag{2.124}$$

Soft decision recursive least squares estimators use (2.124) in (2.66), while soft decision weighted LMS algorithms use (2.124) in place of ρ_n in (2.115). We will call these weighting schemes for decision directed estimation, soft decision weighted estimation.

Chapter 3

Simulation Methodology

In this chapter we present a description of the simulation methodology used to verify the performance of decision weighted algorithms. We begin with a summary of the algorithms simulated, then proceed by describing the wireless channel models. We then revisit MPSK demodulation, but from a simulation point of view. Next, we describe the performance criteria to evaluate the algorithms. Finally, we outline the details of the simulation procedure.

3.1 Summary of Algorithms

We evaluated the performance of both blind and non-blind algorithms. These ten algorithms are described as follows:

- **Training Sequence LMS (TLMS):** Uses (2.105) - (2.106) with $\rho_n = 1$
- **Blind LMS (BLMS):** Uses (2.115) - (2.116) with $\rho_n = 1$
- **Soft Decision Weighted LMS (SDWLMS):** Uses (2.115) - (2.116) with $\rho_n = a_n$ from (2.124)
- **Ideal Decision Weighted LMS (IDWLMS):** Uses (2.115) - (2.116) with $\rho_n = 1$ if $\mathbf{x(n)} = \mathbf{u(n)}$ and zero otherwise
- **Training Sequence RLS (TRLS):** Uses Corollary 2.15 with $a_n = 1$
- **Blind RLS (BRLS):** Uses Corollary 2.28 with $a_n = 1$
- **Soft Decision Weighted RLS (SDWRLS):** Uses Corollary 2.28 with a_n from (2.124)

- **Ideal Decision Weighted RLS (IDWRLS):** Uses Corollary 2.28 with $a_n = 1$ if $\mathbf{x(n)} = \mathbf{u(n)}$ and zero otherwise (see also Proposition 2.40)

- **Modified[1] Soft Decision Weighted RLS (MSDWRLS):** Uses Corollary 2.28 with a_n from (2.124); however, we modify the matrix update $\mathbf{H}_n$ by removing the weight a_n, resulting in $\mathbf{H}_n = \lambda \mathbf{H}_{n-1} + \mathbf{x}(n)\mathbf{x}^T(n)$.

- **Modified Ideal Decision Weighted RLS (MSDWRLS):** Uses Corollary 2.28 with $a_n = 1$ if $\mathbf{x(n)} = \mathbf{u(n)}$ and zero otherwise; however, we modify the matrix update $\mathbf{H}_n$ by removing the weight a_n, resulting in $\mathbf{H}_n = \lambda \mathbf{H}_{n-1} + \mathbf{x}(n)\mathbf{x}^T(n)$.

3.2 Wireless Channel Models

We simulate two different channel environments, a fixed LOS microwave link and a mobile radio channel as described in ([18], pp. 379-384). In general, the number of multipath components for an HF channel are on the order of 2 to 8 [40], [24], [1].

3.2.1 The Radio Relay Three-Path (Rummler) Model

The Rummler model [30] is a popular model used for LOS links in the 6 GHz frequency band. Although it is a three path model, the first two paths are assumed to be very close in time delay. Hence, the model reduces to two paths, the LOS and one reflected path. The model is described probabilistically in ([18], pg. 379-382). The Matlab code to implement it is in Appendix A.1.1.

3.2.2 The Mobile Radio Channel Model

Unlike the Rummler model, which has an impulse scattering function (i.e. the channel does not vary with time), each tap of the mobile radio channel has a scattering function approximated by a single pole low-pass filter response ([18],pg. 382), [17], ([15], pg. 110)

$$S(v) = A\left(1 - (v/f_m)^2\right)^{-1/2} \tag{3.1}$$

where A is the attenuation of the tap and $f_m = s/\lambda$ where s is the speed of the mobile and λ is the wavelength of the transmitted carrier. This 3 dB frequency of this response, f_m, is

[1] After preliminary simulations, we observed that SDWRLS performed poorly. We examined the update structure of (2.68) and observed that for small $\mathbf{H}_{n-1}$, the weight a_n cancels itself in the estimator update equation (2.67). This motivated removing the weight a_n from (2.68) to produce the "modified" decision weighted RLS algorithms.

sometimes referred to as the Doppler frequency. The above equation was derived by Jakes [17] under assumptions that the mobile is moving at a constant velocity receiving infinitely many reflected waves from all directions uniformly. We implemented each tap of the mobile channel model by passing mean zero circularly symmetric complex white Gaussian noise through a filter with response (3.1). The impulse response derived from (2.8) consists of a direct LOS path ($A = 0$ dB attenuation), a path delayed by 1 symbol interval ($A = -20$ dB attenuation), and a path delayed by 2 symbol intervals ($A = -25$ dB attenuation). The code for implementing the mobile radio channel is in Appendix A.1.2.

3.3 Multiple Phase Shift Keying (MPSK) Receiver Implementation

Synchronization is a difficult challenge in implementing a coherent MPSK receiver. In our simulations, synchronization issues arose in two areas: synchronizing to the channel-induced distortion of the constellation, and symbol delays. A common method to perform synchronization is with phase lock loops (PLLs) [13], ([10], pg. 444). For simulation purposes, however, we can emulate the behavior of PLLs in a coherent receiver by using knowledge of the channel to our advantage as described in the next two sections.

3.3.1 Carrier (Phase) Synchronization

Because the channel is complex, the resulting MPSK constellation is a distorted version of the original constellation. Figure 3.1 illustrates the effect of a Rummler channel on a QPSK constellation. Although the original constellation has only 4 elements, the constellation after the channel has 16. The reason for this increase is the sequence dependent intersymbol interference (ISI). A Rummler channel consists of a LOS and a reflected path; therefore, the possible number of permutations that arise from 4 symbols combined from 2 paths is $4^2 = 16$. A maximum likelihood sequence detector would use a channel estimate to calculate these 16 permutations and pick the point closest to the received point [25].

Although we did not implement a maximum likelihood sequence detector, we did use the true channel response to align the angle decision boundaries in the detector to compensate for a channel induced constellation rotation with no ISI. This emulates a simple PLL adjusting the phase of its quadrature sinusoidal generator to achieve phase synchronization [13]. For the mobile radio channel, the decision boundaries are adjusted at the beginning of each symbol in-

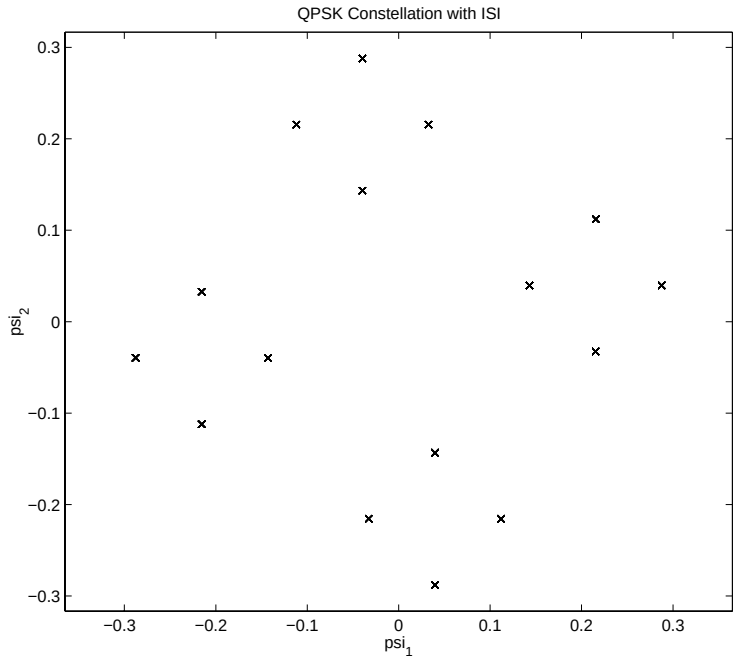

Figure 3.1: Effect of a Rummler channel on a QPSK constellation

terval, similar to a PLL tracking the phase response of the channel. The program that calculates these decision boundaries is listed in Appendix A.4.6.

3.3.2 Symbol Synchronization

The problem of symbol synchronization is to identify when a symbol interval begins and ends. To make the solution to this problem easier, we restrict the taps of the channel impulse responses to be spaced integer multiples of the symbol interval length ([35], pg. 31). The problem now reduces to finding the number of symbol intervals the channel shifted the transmitted sequence. To identify this shift we perform a binary cross correlation between the transmitted symbols and the detected symbols. A binary cross correlation means to take an ordinary cross correlation but replace the multiplication with a binary AND function, resulting in a 1 when two symbols align and a 0 when they do not. The maximum number of shifts to test is the length of the channel's impulse response. The function that performs this symbol synchronization is listed in Appendix A.4.7.

3.4 Performance Criteria: Squared Distance from True Response

To compare the performance of the estimator algorithms we use the average squared absolute difference between the true impulse response and their estimate. In other words, the estimation error for a given simulation is

$$\epsilon = \frac{1}{N - N_0 - 1} \sum_{n=N_0}^{N} \left| \beta - \widehat{\beta}_n \right|^T \left| \beta - \widehat{\beta}_n \right| \tag{3.2}$$

where N_0 is an integer greater than the initial transient response of the estimator. To evaluate each estimator's performance across several simulations, we use the median of estimation errors from each simulation to reduce the effect of the best and worst simulations.

3.5 Delay Spread, SNR, and Doppler Frequency Performance Test Methodology

3.5.1 General Methods

Common to all the performance tests are the parameters summarized below:

- **LMS Gain:** $\mu = 0.3$
- **RLS Forgetting Factor:** $\lambda = 0.99$
- **Sampling Rate:** 1 sample per second
- **Symbol Interval:** 4 samples per symbol
- **Modulation:** QPSK
- **Total Number of Symbols per Individual Simulation:** 300 symbols
- **Number of Individual Simulations to Perform per Iteration:** 20 simulations; we used the median estimation error over these 20 simulations to rank the estimators for a particular value of delay spread, SNR, or Doppler frequency.
- **Maximum Symbol Error Rate (SER):** 0.2 ; we repeated individual simulations if their SER was higher than this threshold
- **Number of Symbols to Skip Before Calculating Estimation Error (N_0)**: 100 symbols; we skip the initial symbols to observe the algorithms' steady state behaviors

- **Initial Estimate:** the true response; we used the true response as the initial estimate to facilitate the simulation of steady state behavior

We now describe each performance test in more detail.

3.5.2 Performance Versus Delay Spread

In this performance test we held the SNR at the receiver fixed at 10 dB while varying the delay spread of a Rummler channel from one to five symbol intervals in one interval increments. The results of this simulation are shown in Figures 4.27 and 4.28. The code for this simulation is listed in Appendix A.3.2.

3.5.3 Performance Versus SNR

This test fixed the delay spread of a Rummler channel to 1 symbol interval, then varied the SNR entering the receiver from 0 to 30 dB in 3 dB increments. The results of this simulation are shown in Figures 4.29 and 4.30. The code for this test is listed in Appendix A.3.3.

3.5.4 Performance Versus Normalized Doppler Frequency

The purpose of this test was to show the disadvantages of using soft decision weighted algorithms. The soft decision weights act as an adaptive gain for the LMS algorithm. This adaptive gain is always less than or equal to the gain of the fixed gain LMS for equal values of $/mu$. As a result, the SDWLMS has a slower learning curve. When the channel is time-varying, we should see poorer performance in the soft decision weighted algorithms for large Doppler frequencies.

In this performance test we held the SNR fixed at 10 dB and varied the Doppler frequency (normalized to the sampling frequency) of a mobile radio channel at 15 points logarithmically spaced between 10^{-16} to 10^{-2}. Figures 4.31 and 4.32 show the results of this test. We then repeated the test for 15 points logarithmically spaced between 10^{-10} to 10^{-2} for better resolution. Figures 4.33 and 4.34 show these results. The code for this performance test is listed in Appendix A.3.4.

Chapter 4

Simulation Results

4.1 Illustrating the Effect of a Decision Error

To illustrate the effect of a decision error on decision directed algorithms, we manually inserted a symbol error at approximately 900 seconds as marked by a gray bar in Figures 4.1 through 4.11. In this BPSK simulation, the sampling frequency was 1 Hz with 4 samples per symbol interval and a 10 dB SNR before entering the receiver. The channel model was a Rummler channel with a delay spread of 1 symbol interval. The training sequence versions of each algorithm are shown for comparison purposes.[1] Appendix A.3.1 lists the Matlab code for this simulation.

4.2 Illustrating the Behavior of the Adaptive Algorithms

To illustrate the general behavior of the adaptive algorithms, we performed a single simulation with parameters identical to those in the previous section with the exception of using QPSK modulation and a delay spread of 2 symbol intervals. The LMS gain and RLS forgetting factor were $\mu = 0.3$ and $\lambda = 0.99$, respectively. The eye and constellation diagrams for this simulation are shown in Figure 4.12. The estimates from each algorithm are shown in Figures 4.13 through 4.22. We marked the symbol errors in the graphs by vertical gray bars. The symbol error rate for this simulation was 0.04. The average squared distance under the estimation error curves (Figures 4.23 and 4.24) in steady-state (after 100 symbol intervals) was

- TLMS - 0.87

[1] We assume that the symbol error does not cause improper recognition of the training sequence.

- BLMS - 1.27
- SDWLMS - 0.12
- IDWLMS - 0.86
- TRLS - 0.36
- BRLS - 0.48
- SDWRLS - 2.47
- IDWRLS - 0.41
- MSDWRLS - 0.02
- MIDWRLS - 0.34

The weights p_i and a_n in (2.121) and (2.124) are shown in Figures 4.25 and 4.26, respectively.

4.3 Performance Tests

Figures 4.27 through 4.34 show the simulated performance of the LMS and RLS algorithms as a function of delay spread, SNR, and normalized Doppler frequency.

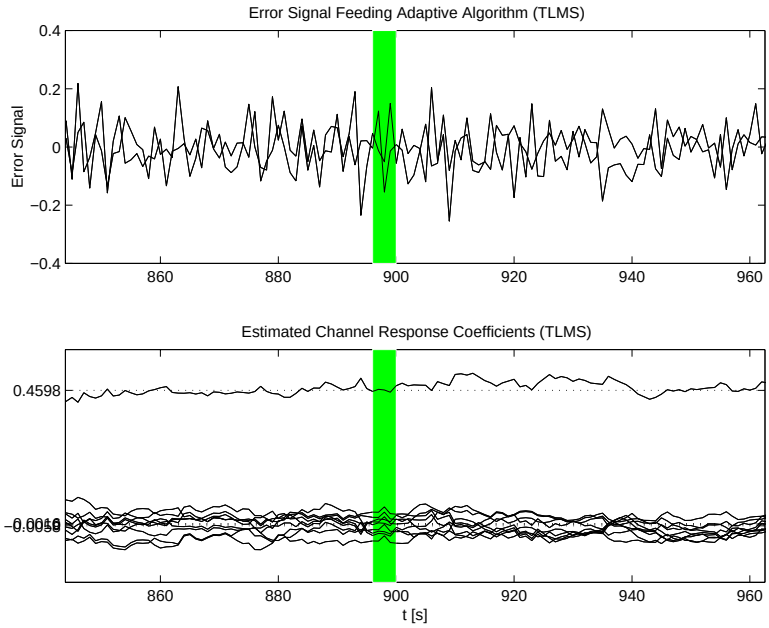

Figure 4.1: Effect of a decision error on the training sequence LMS (TLMS) algorithm ($\mu = 0.3$)

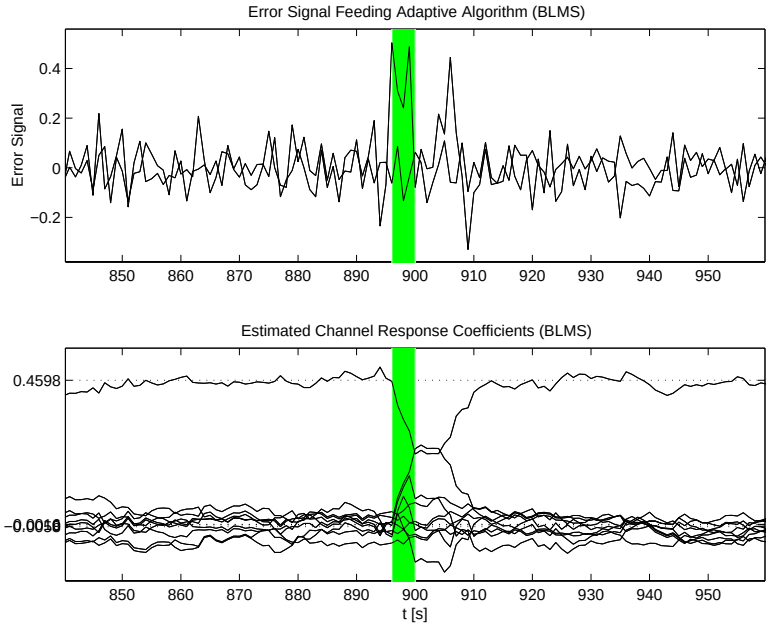

Figure 4.2: Effect of a decision error on the decision directed (blind) LMS (BLMS) algorithm ($\mu = 0.3$)

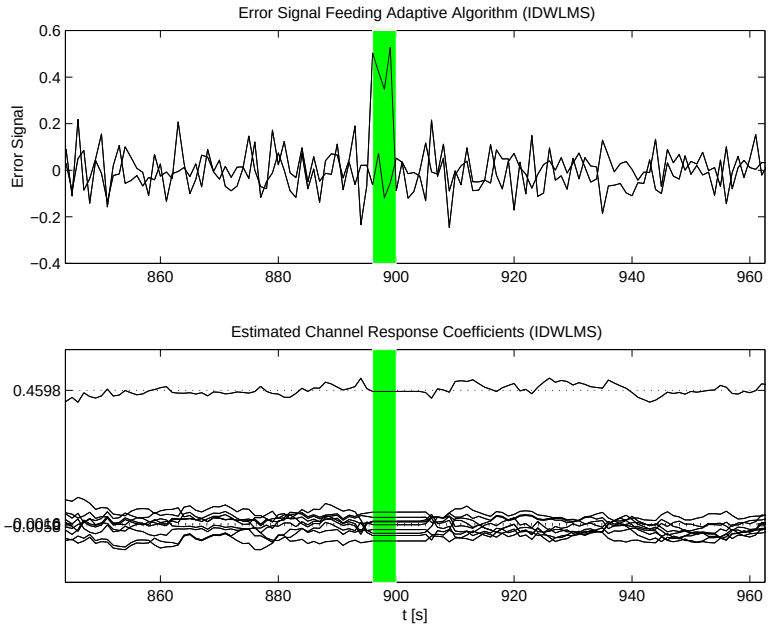

Figure 4.3: Effect of a decision error on the ideal decision weighted LMS (IDWLMS) algorithm ($\mu = 0.3$)

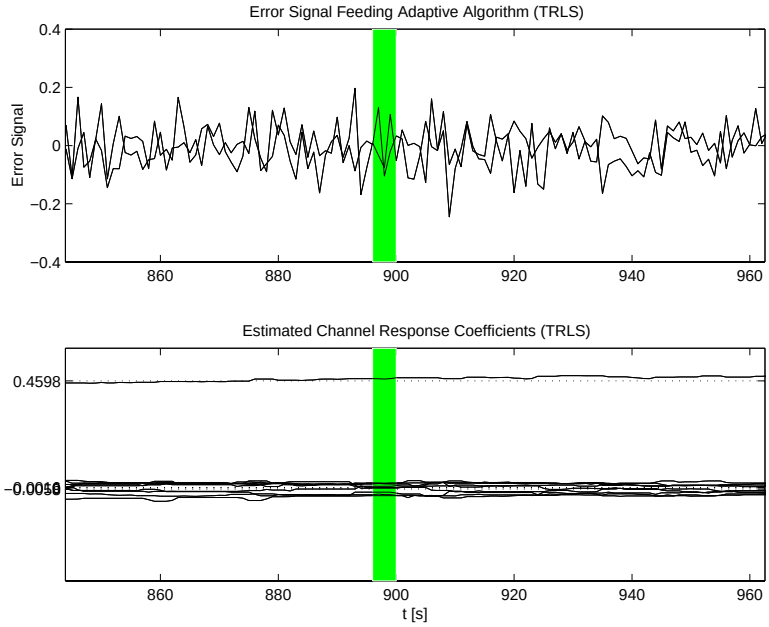

Figure 4.4: Effect of a decision error on the training sequence RLS (TRLS) algorithm ($\lambda = 0.99$)

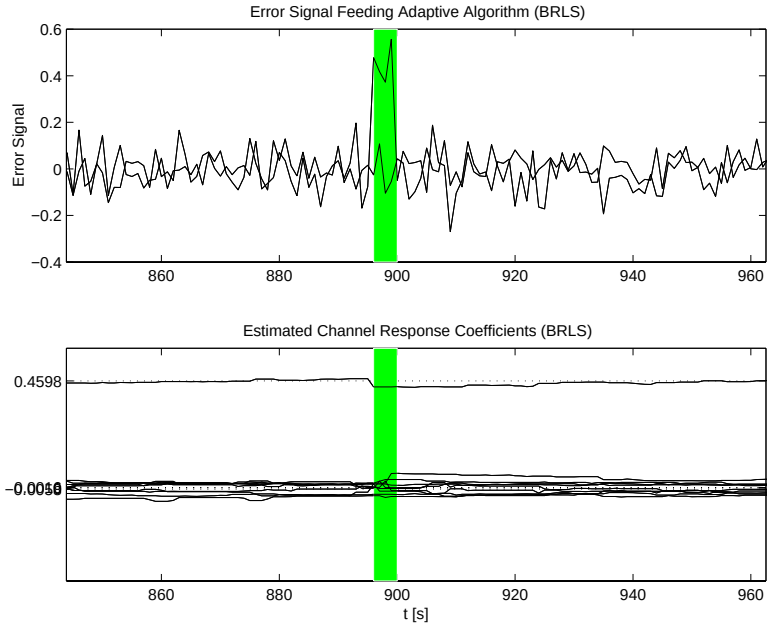

Figure 4.5: Effect of a decision error on the decision directed (blind) RLS (BRLS) algorithm ($\lambda = 0.99$)

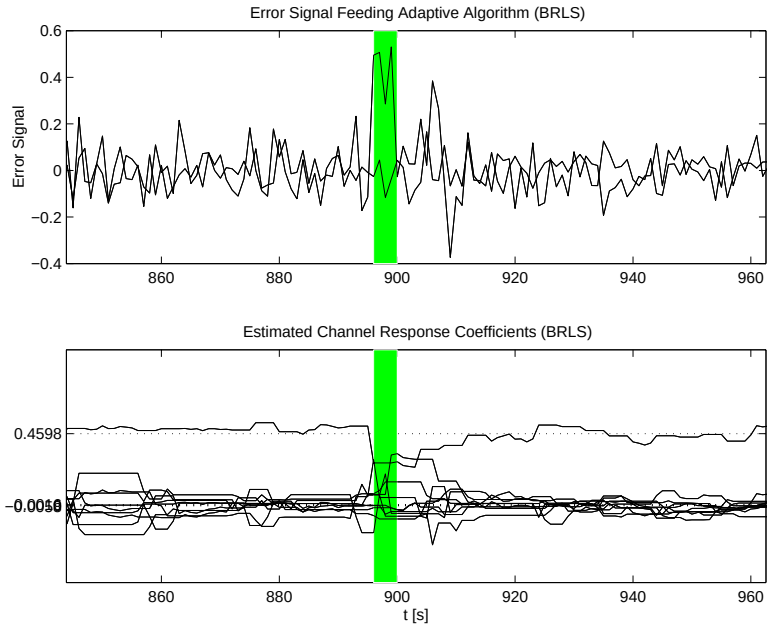

Figure 4.6: Effect of a decision error on the decision directed (blind) RLS (BRLS) algorithm ($\lambda = 0.9$)

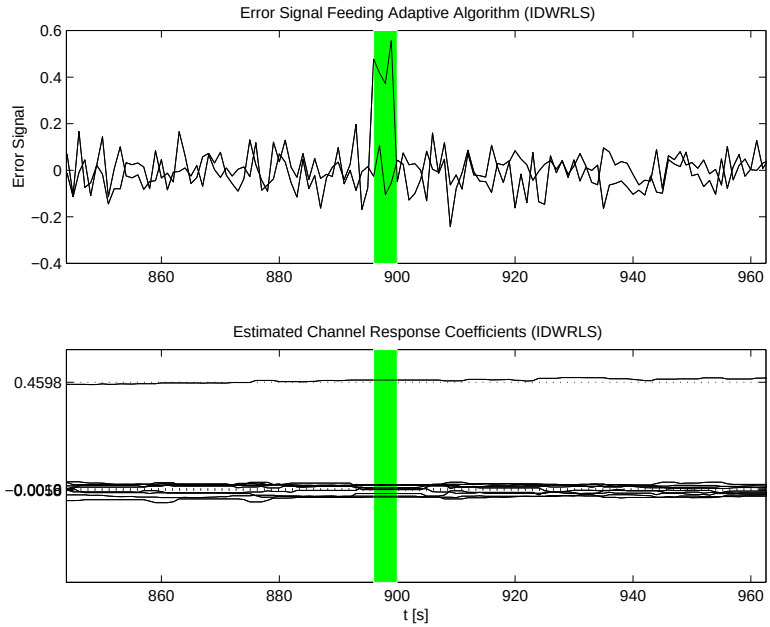

Figure 4.7: Effect of a decision error on the ideal decision weighted RLS (IDWRLS) algorithm ($\lambda = 0.99$)

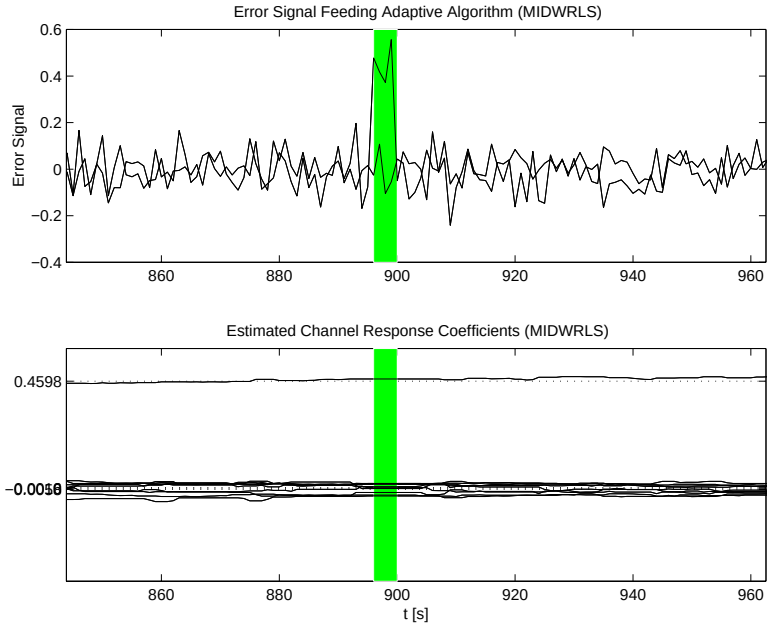

Figure 4.8: Effect of a decision error on the modified ideal decision weighted RLS (MIDWRLS) algorithm ($\lambda = 0.99$)

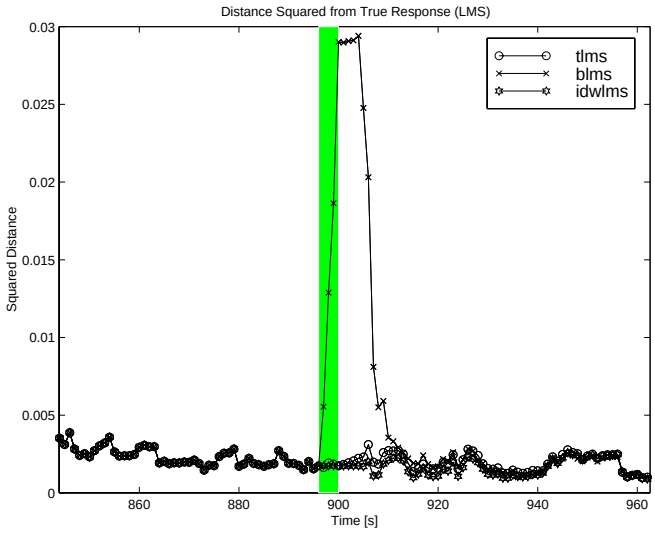

Figure 4.9: Comparison of a decision error's effect on the LMS algorithms ($\mu = 0.3$)

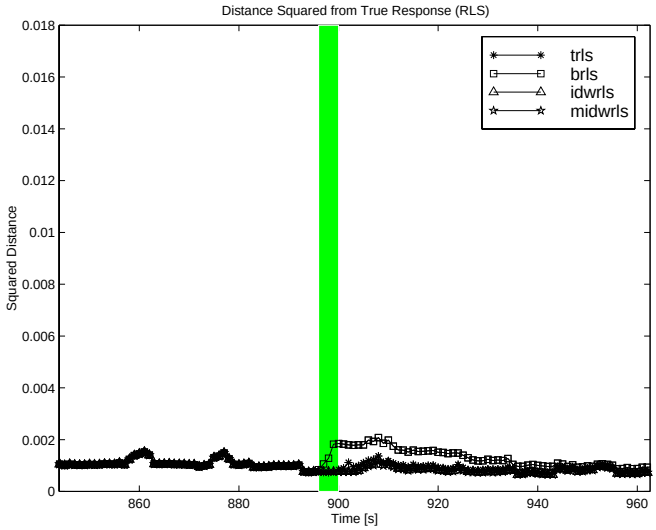

Figure 4.10: Comparison of a decision error's effect on the RLS algorithms ($\lambda = 0.99$)

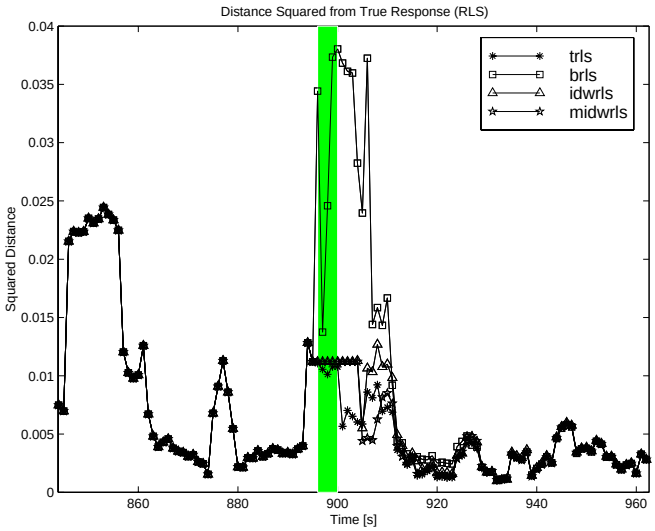

Figure 4.11: Comparison of a decision error's effect on the RLS algorithms ($\lambda = 0.9$)

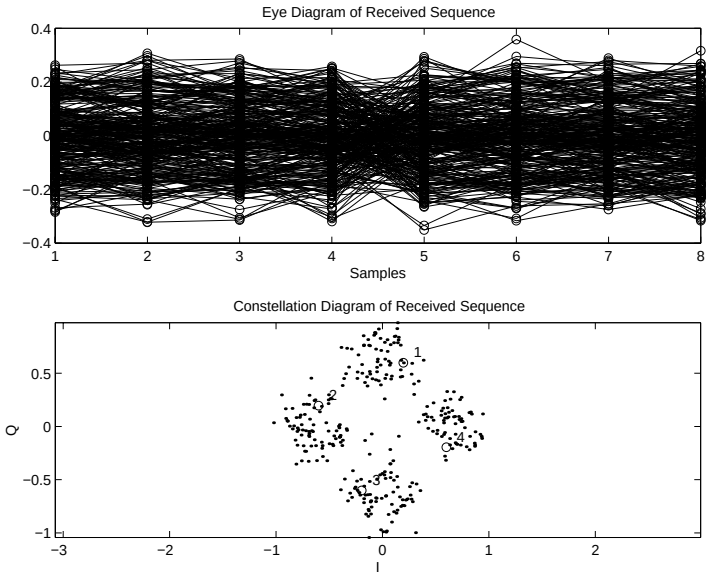

Figure 4.12: Eye and constellation diagram from a QPSK estimation example with a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

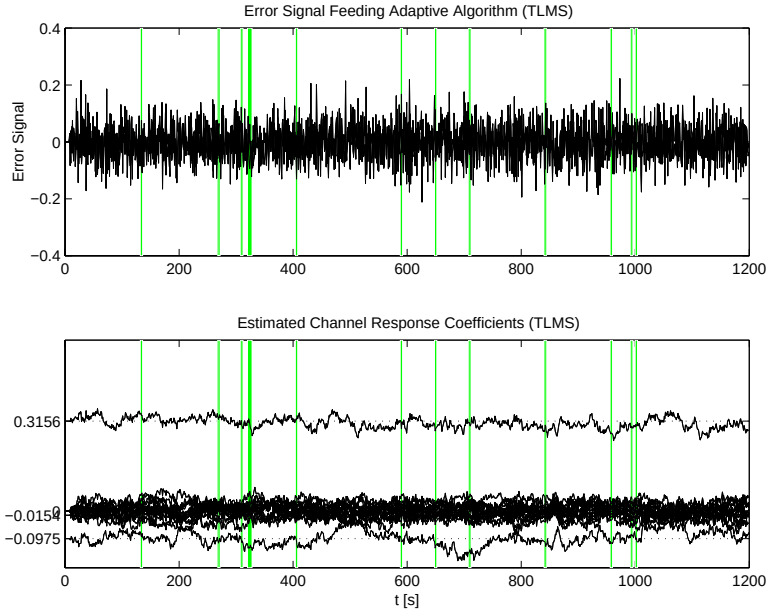

Figure 4.13: Example of TLMS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

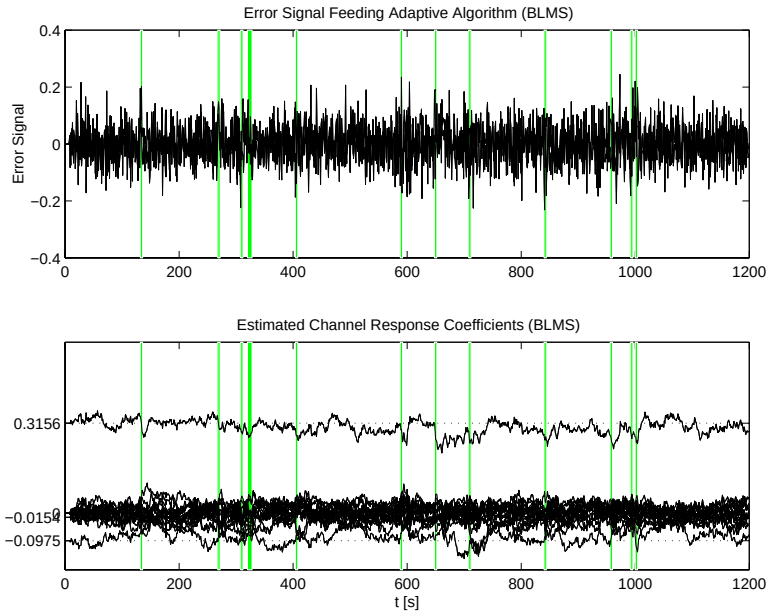

Figure 4.14: Example of BLMS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

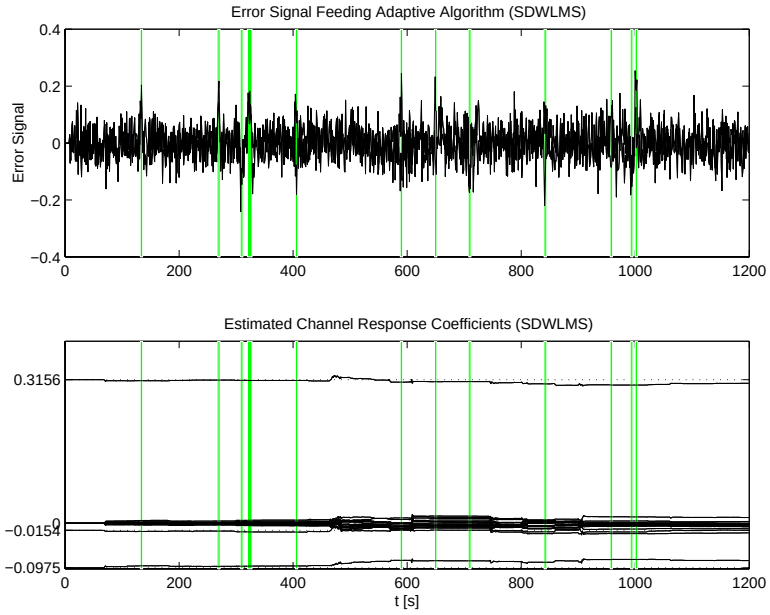

Figure 4.15: Example of SDWLMS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

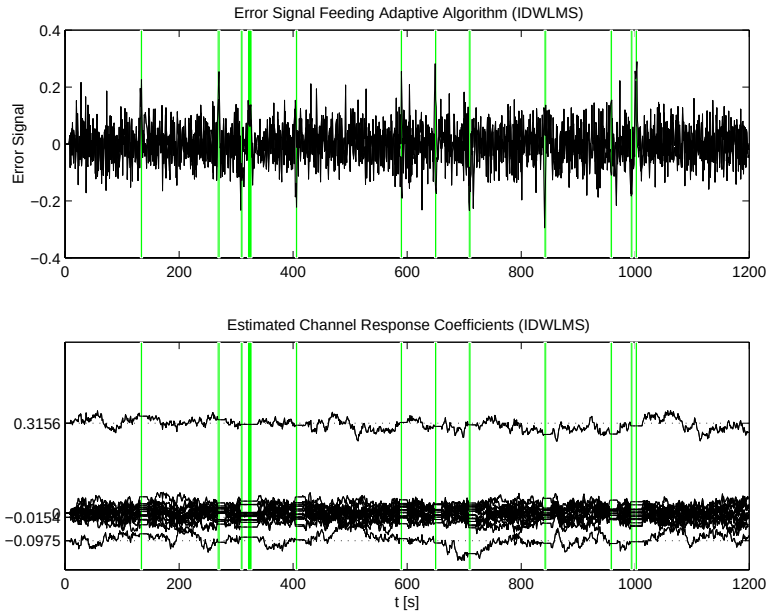

Figure 4.16: Example of IDWLMS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

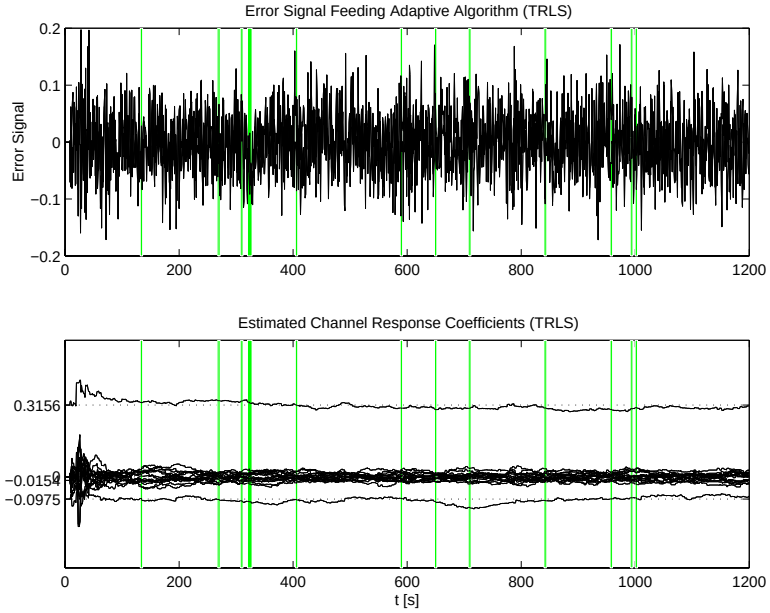

Figure 4.17: Example of TRLS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

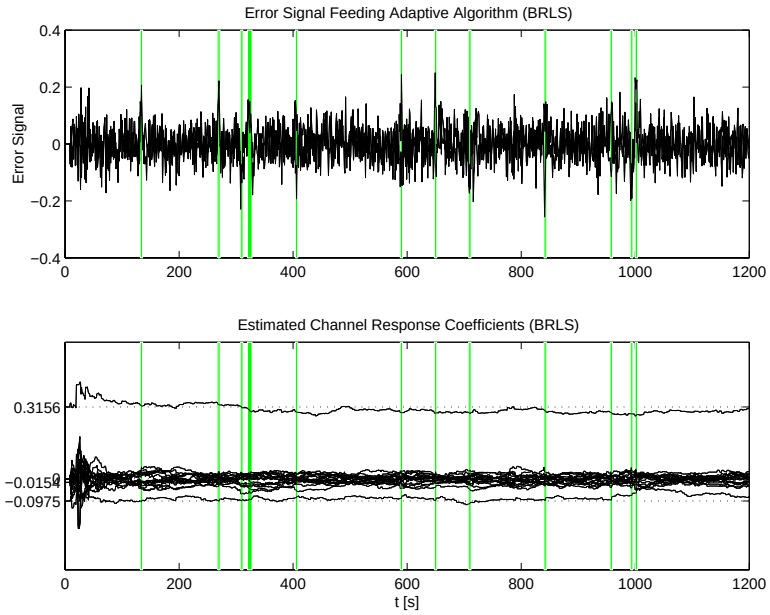

Figure 4.18: Example of BRLS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

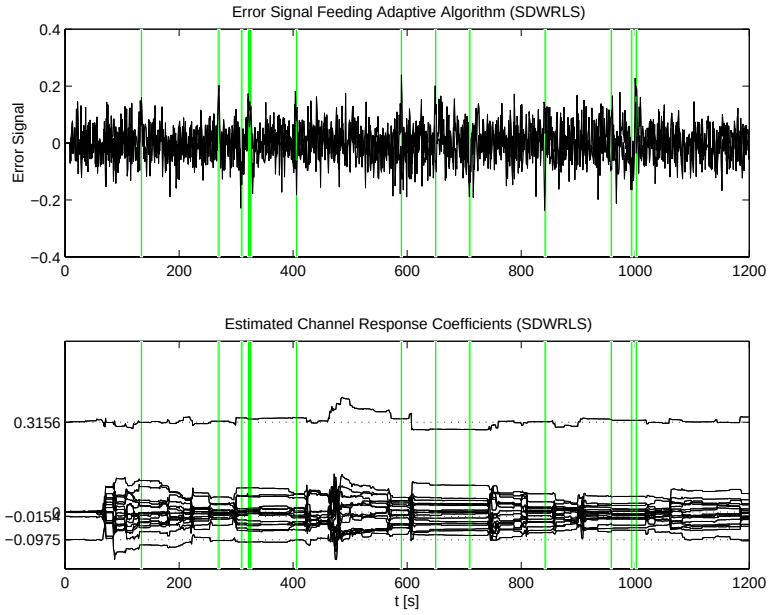

Figure 4.19: Example of SDWRLS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

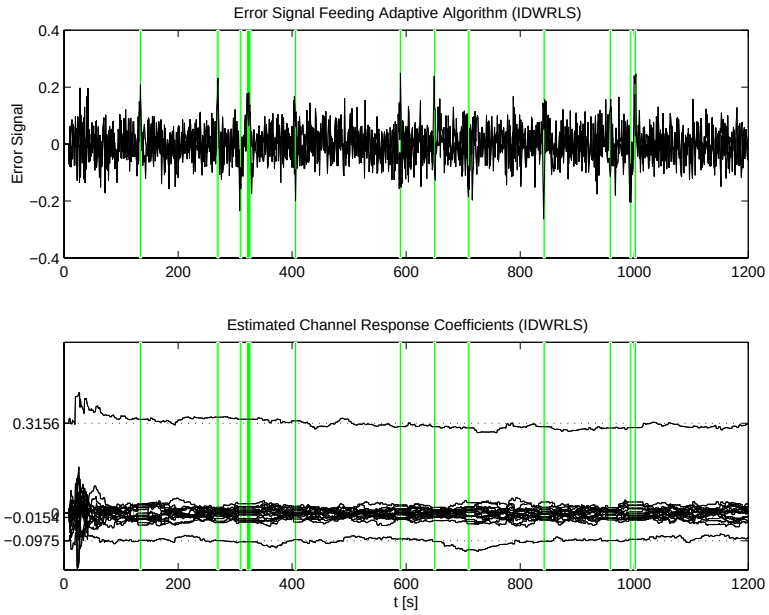

Figure 4.20: Example of IDWRLS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

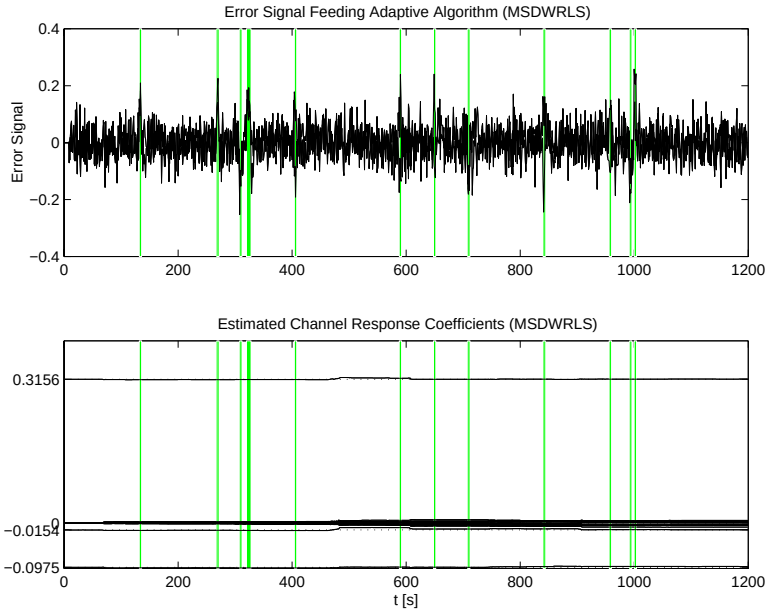

Figure 4.21: Example of MSDWRLS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

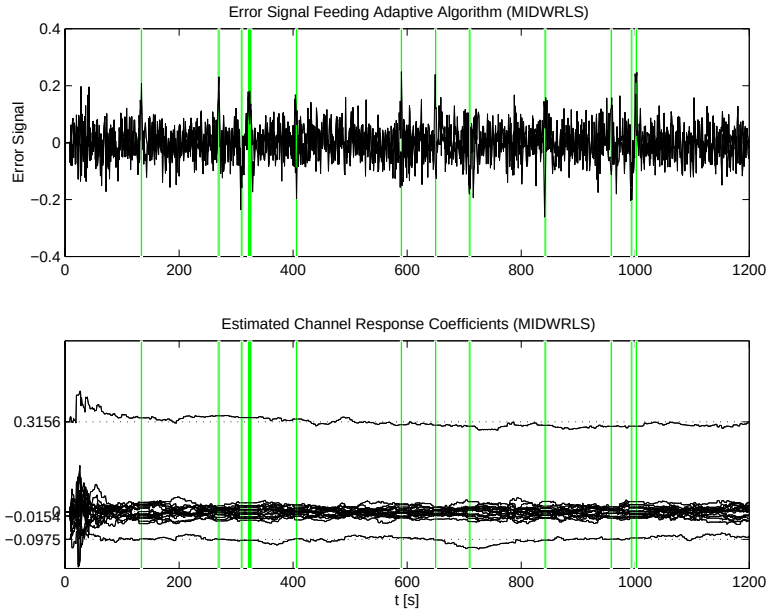

Figure 4.22: Example of MIDWRLS estimates of a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

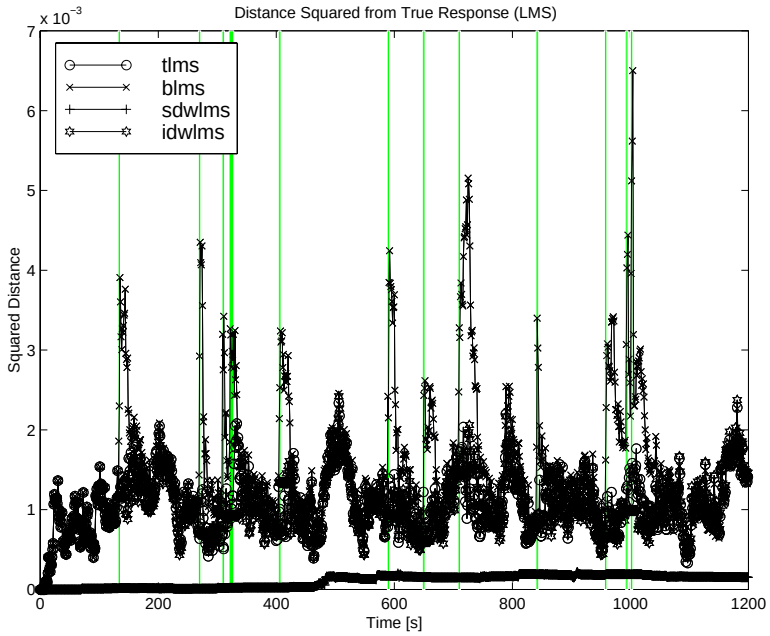

Figure 4.23: Example of LMS estimation error with a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

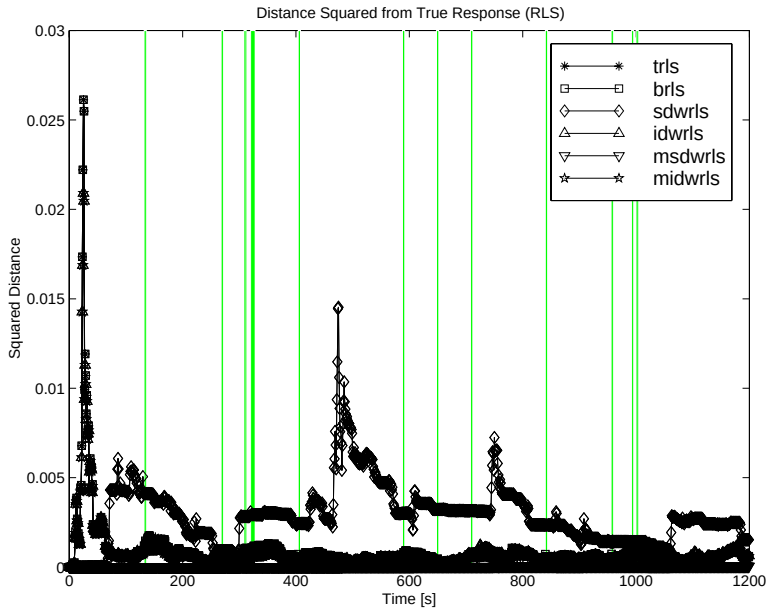

Figure 4.24: Example of RLS estimation error with a Rummler Channel with a delay spread of 2 symbol intervals and a 10 dB SNR

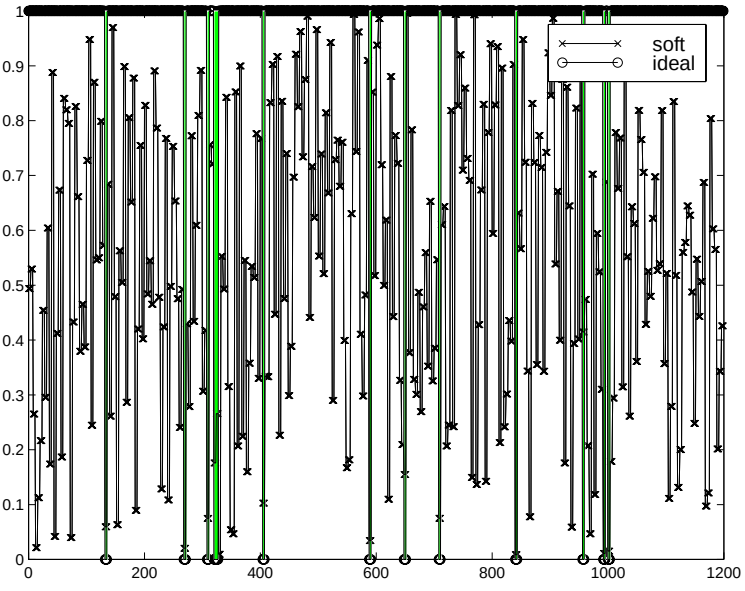

Figure 4.25: The decision weights for each hard decision (p_i in (2.121))

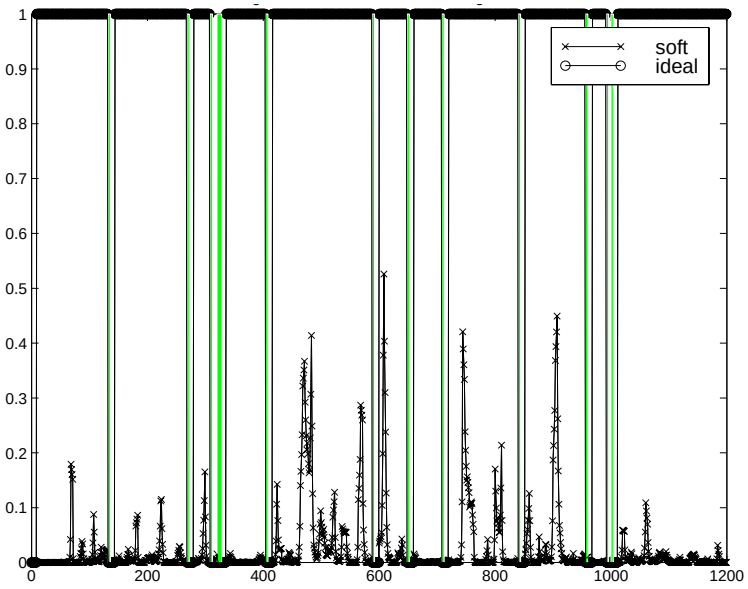

Figure 4.26: The algorithm soft decision weights (a_n in (2.124)) for the previous SDWLMS, IDWLMS, SDWRLS, IDWRLS, MSDWRLS, and MIDWRLS examples

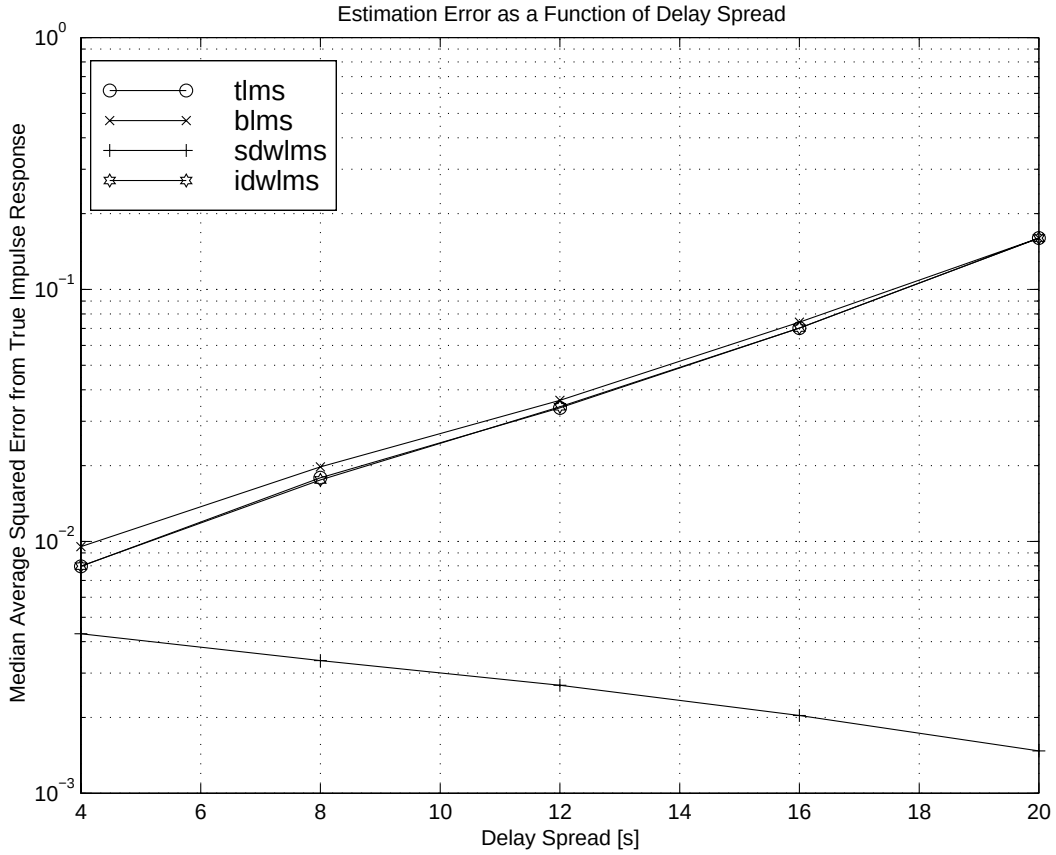

Figure 4.27: Median of the average squared error of LMS algorithms as a function of delay spread (SNR = 10 dB)

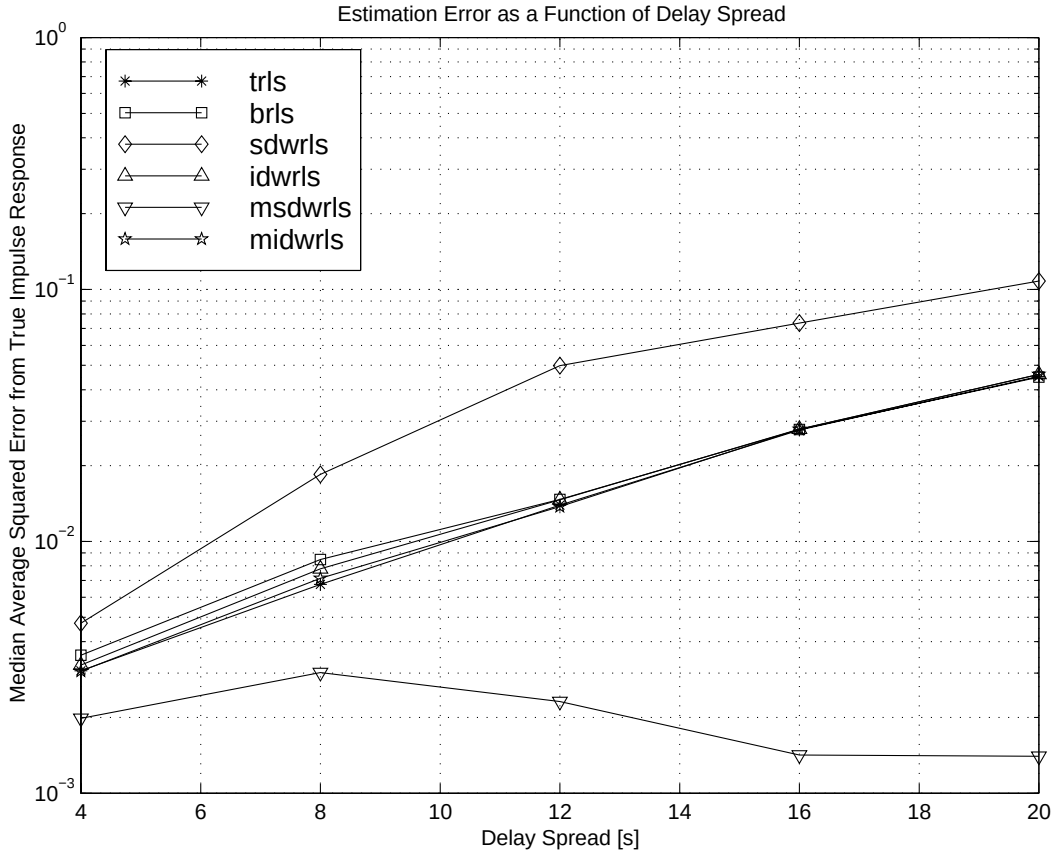

Figure 4.28: Median of the average squared error of RLS algorithms as a function of delay spread (SNR = 10 dB)

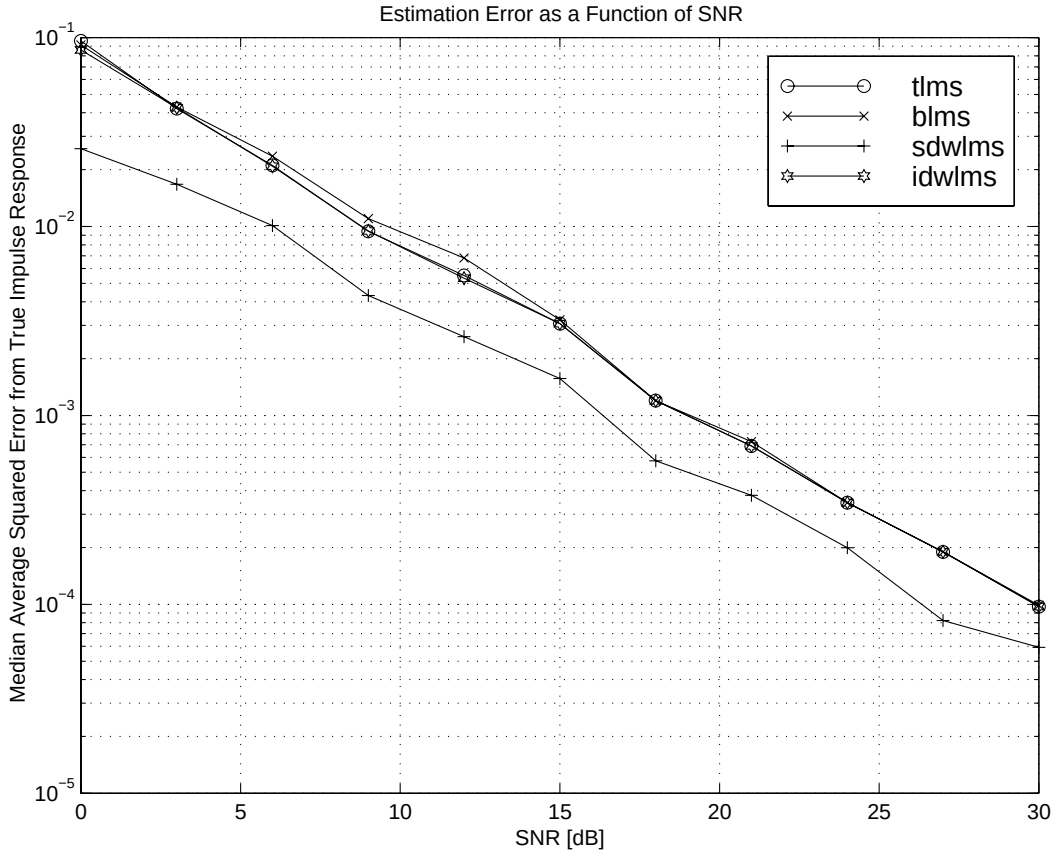

Figure 4.29: Median of the average squared error of LMS algorithms as a function of SNR (Delay Spread = 1 symbol interval)

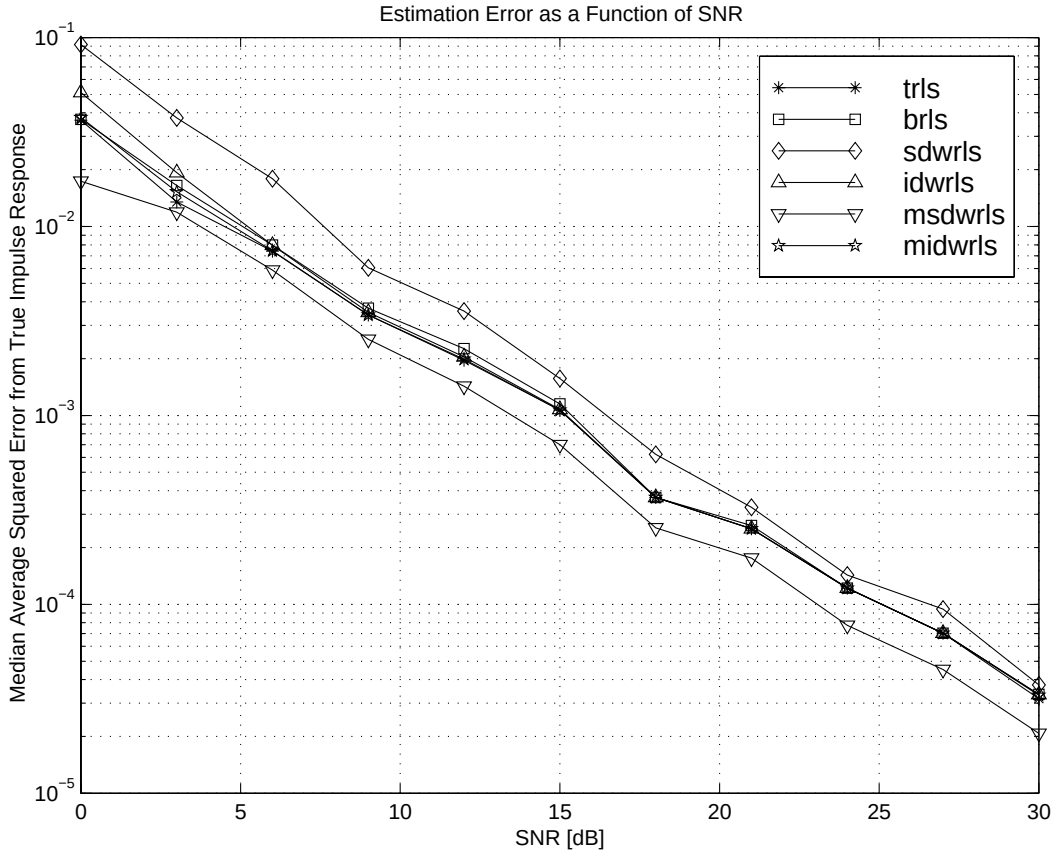

Figure 4.30: Median of the average squared error of RLS algorithms as a function of SNR (Delay Spread = 1 symbol interval)

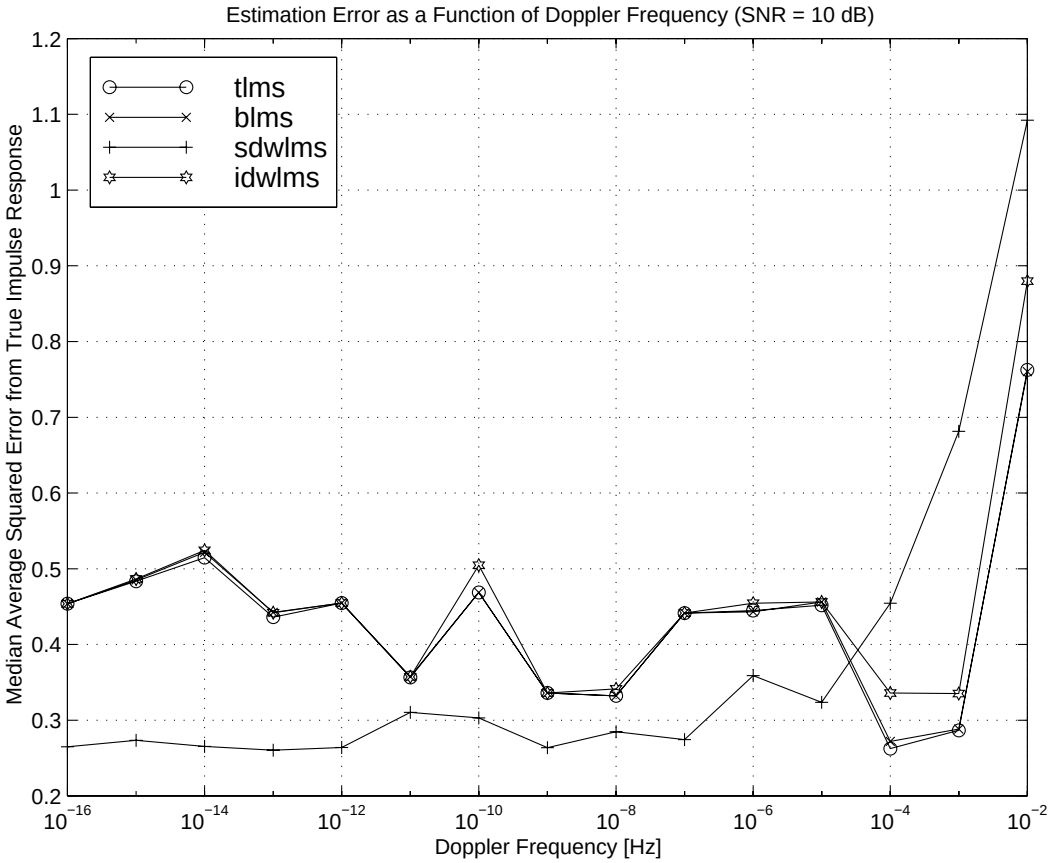

Figure 4.31: Median of the average squared error of LMS algorithms as a function of Doppler frequency (SNR = 10 dB)

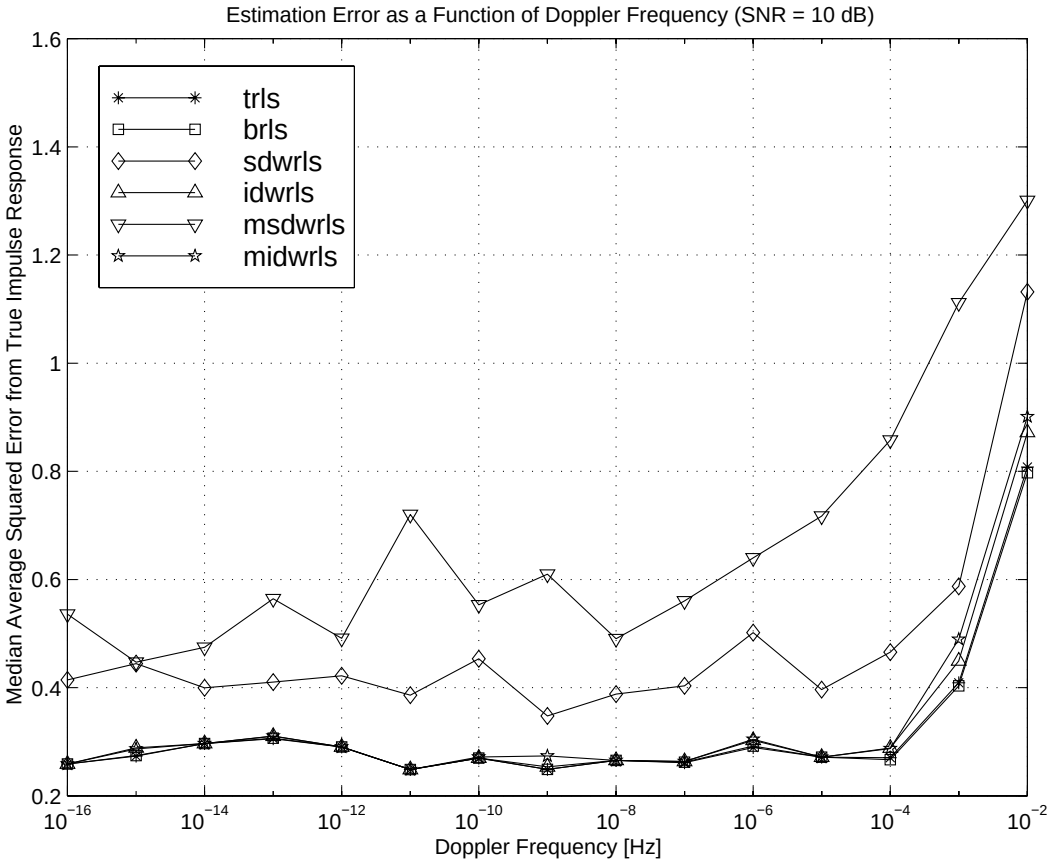

Figure 4.32: Median of the average squared error of RLS algorithms as a function of Doppler frequency (SNR = 10 dB)

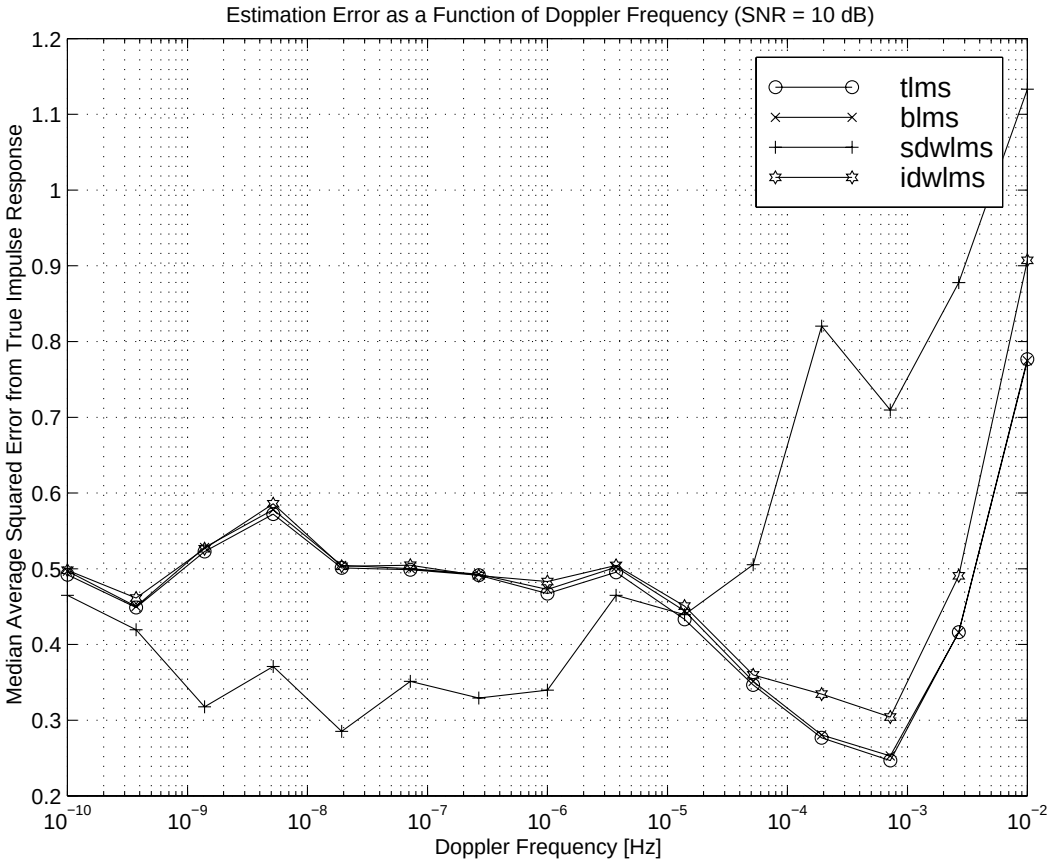

Figure 4.33: Median of the average squared error of LMS algorithms as a function of Doppler frequency (SNR = 10 dB)

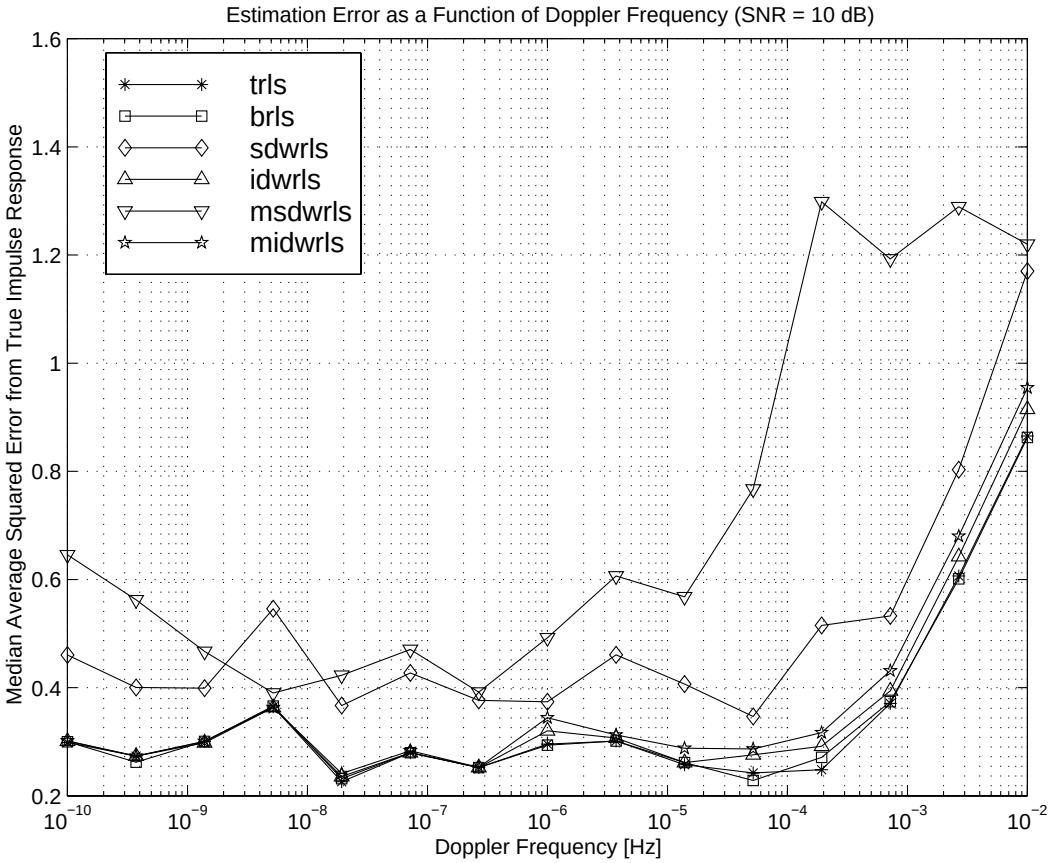

Figure 4.34: Median of the average squared error of RLS algorithms as a function of Doppler frequency (SNR = 10 dB)

Chapter 5

Conclusions

5.1 Performance of Decision Weighted Algorithms

Figures 4.27 through 4.34 illustrate the performance of the decision weighted algorithms as a function of delay spread, SNR, and Doppler frequency in reference to their standard training sequence and decision directed forms. From these graphs we can draw the following general conclusions:

- The soft decision weighted LMS (SDWLMS) performed *better* than the other LMS algorithms in the delay spread (by a factor of 2 to 100, Figure 4.27) and SNR (by a factor of 2, Figure 4.29) tests.
- The soft decision weighted RLS (SDWRLS) performed *worse* than the other RLS algorithms in the delay spread (by a factor of 2, Figure 4.28) and SNR (by a factor of 3, Figure 4.30) tests
- The modified soft decision weighted RLS (MSDWRLS) performed *better* than the other RLS algorithms in the delay spread (by a factor of 2 to 20, Figure 4.28) and SNR (by a factor of 2, Figure 4.28) tests.
- The SDWLMS performed *better* at normalized Doppler frequencies less than 10^{-5} and *worse* at higher Doppler frequencies than the other LMS algorithms (Figures 4.31 and 4.33).
- The SDWRLS and MSDWRLS performed *worse* over all Doppler frequencies than the other RLS algorithms (Figures 4.32 and 4.34).

- The ideal decision weighted LMS and RLS (IDWLMS and IDWRLS) performed similar to their training sequence versions in all the tests, and generally better than their ordinary decision-directed counterparts (Figures 4.27 through 4.34).

The reason the SDWLMS algorithm performed better than the other LMS algorithms in high SNR and large delay spread situations is because in these environments the soft decision weights are small (see Figures 4.25 and 4.26 for an example of these weights), making the SDWLMS sluggish; therefore, it is less susceptible to quick transients caused by noise and decision errors. The SDWLMS, however, does not track the time varying channels as well as the other LMS algorithms for normalized Doppler frequencies greater than 10^{-5} Hz (Figures 4.31 and 4.33) for this same reason. The MSDWRLS exhibits behavior similar to the SDWLMS because of its similar estimate update weighting structure.

While the ideal decision weighted algorithms (IDWLMS, IDWRLS, and MIDWRLS) exhibited robustness to decision errors, they did not protect the estimate from noise as did the SDWLMS and MSDWRLS. We expected this because these ideal algorithms only react to bad decisions and not to noise. On the other hand, the soft decision algorithms respond to the quality of the decision, which degrades in the presense of noise. We do not fully understand why the SDWRLS failed to perform as well as the blind RLS in any of the tests. This failure prompted the creation of its "modified" version, which outperformed the other RLS algorithms in SNR and delay spread tests. This modified estimator, however, is no longer a least squares estimator, i.e. it does not minimize (2.27). It does exhibit nice properties, though, and warrants further study.

In summary, this thesis demonstrated how RLS and LMS adaptive algorithms can estimate the impulse response of a wireless communications channel using either training sequences or decision feedback. It also proposed novel modifications to these algorithms to reduce their susceptibility to decision errors and noise. Of these modifications, the soft decision weighted LMS (SDWLMS) has advantages over the standard decision directed and training sequence LMS if the channel is slowly time-varying. The soft decision weighted RLS (SDWRLS) algorithm did not perform as well as the other RLS algorithms in any of our tests; however, we proposed and tested modifications to the SDWRLS that led to good performances similar to that of the SDWLMS.

5.2 Unanswered Questions/Future Work

The work presented in this thesis could be extended in several ways, including the following:

- *Actual Channel Measurements:* The radios of the Phase 2 Rapidly Deployable Radio Network (RDRN) are capable of measuring the signals required to test the algorithms presented in this thesis. These measurements would allow further study of how practical radio considerations such as automatic gain control (AGC), coherent PLL demodulation, and quantization affect algorithm performance.

- *Decision Weighted Algorithms in Equalizer and Maximum Likelihood Sequence Detector Applications:* Decision directed channel estimation is an easier problem to analyze than applications where the decision process is dependent on the adaptive algorithm. In decision directed adaptive equalizers, for example, a decision error will cause inaccuracies in the equalizer, which in turn could cause more decision errors. These decision directed applications experience rapid degradation as decision errors increase. We suspect that decision weighted algorithms would help reduce this degradation.

- *Further Analysis of Soft Decision Weighted Algorithms:* We motivated soft decision weighted algorithm development through analysis of ideal decision weighted algorithms. We have seen, however, that SDWLMS and MSDWRLS algorithms perform better than their ideal counterparts. The reason for this performance improvement is due to the SDWLMS and MSDWRLS estimator update structure. In low SNR environments, the soft decision weights are small because of the poor decision quality. As a result, the SDWLMS and MSDWRLS algorithms clamp their current estimate close to their previous estimate. We recommend further analysis of this property to quantify the estimation error of these algorithms.

- *Further Analysis of the Modified Decision Weighted RLS Algorithms:* We modified the decision weighted RLS algorithms with intent to improve the performance of the SDWRLS algorithm. We succeeded, but at the same time developed an algorithms that are no longer a least squares estimators (i.e. it does not minimize (2.27)). Further analysis of these modified algorithms could clarify why they work better than the other RLS algorithms.

Bibliography

[1] GSM Recommendation 05.05. *Radio Transmission and Reception*. Version 3.11.0, 1990.

[2] P.A. Bello. Characterization of randomly time-variant linear channels. *IEEE Transactions on Communication Systems*, CS-11:360–393, 1963.

[3] M. Benthin and K.D. Kammeyer. Influence of channel estimation on the performance of a coherent ds-cdma system. *IEEE Transactions on Vehicular Technology*, 46(2):262–268, May 1997.

[4] B. Bercu. Weighted estimation and tracking for armax models. *SIAM Journal on Control and Optimization*, 33(1):89–106, January 1995.

[5] D. Boss and K. Kammeyer. Blind identification of mixed-phase fir systems with application to mobile communication channels. *Proceedings of ICASP-97*, 5:3589–3592, April 1997.

[6] D. Boss, T. Petermann, and K. Kammeyer. Impact of blind versus non-blind channel estimation on the ber performance of gsm receivers. *Proceedings of IEEE Signal Processing Workshop on Higher-Order Statistics*, pages 62–66, July 1997.

[7] Z. Ding. Blind equalization based on joint minimum mse criteria. *IEEE Transactions on Communications*, 42(2 / 3 / 4):648–654, February / March / April 1994.

[8] P. Eykhoff. *System Identification*. John Wiley & Sons Ltd., 1974.

[9] A. Feuer and E. Weinstein. Convergence analysis of lms filters with uncorrelated gaussian data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(1):222–230, February 1985.

[10] M. E. Frerking. *Digital Signal Processing in Communication Systems*. Chapman & Hall, 1994.

[11] D. J. Goodman. Trends in cellular & cordless communications. *IEEE Communications Magazine*, pages 31–40, June 1991.

[12] L. Guo. Self-convergence of weighted least-squares with applications to stochastic adaptive control. *IEEE Transactions on Automatic Control*, 41(1):79–89, January 1996.

[13] Harris Semiconductor. *HSP50210: Digital Costas Loop*, January 1997.

[14] I.N. Herstein. *Topics in Algebra*. John Wiley & Sons Ltd., 1975.

[15] M. L. Honig and V. Poor. Adaptive interference suppression. In V. Poor and G. Wornell, editors, *Wireless Communications: Signal Processing Perspectives*, pages 64–128. Prentice Hall, 1998.

[16] F.M. Hsu. Data directed estimation techniques for single-tone hf modems. *IEEE Proceedings of MILCOM'85*, pages 12.4.1–12.4.10, 1985.

[17] W.C. Jakes. *Microwave Mobile Communications*. John Wiley & Sons Ltd., 1974.

[18] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan. *Simulation of Communication Systems*. Plenum Press, 1992.

[19] T. Kailath. Channel characterization: Time-variant dispersive channels. In E.J. Baghdady, editor, *Lectures on Communication System Theory*. McGraw-Hill, 1961.

[20] K.D. Kammeyer and B. Jelonnek. A new fast algorithm for blind ma-system identification based on higher order cumulants. *SPIE Advanced Signal Processing: Algorithms, Architectures & Implementations V*, July 1994.

[21] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons Ltd., 7 edition, 1975.

[22] P.R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, 1986.

[23] Y. Li and Z. Ding. Convergence analysis of finite length blind adaptive equalizers. *IEEE Transactions on Signal Processing*, 43(9):2120–2129, September 1995.

[24] R.W. Lorenz, J. de Weck, and P. Merki. Power delay profiles measured in mountainous terrain. *Proceedings of Vehicular Technology Conference*, 88:105–112, September 1988.

[25] H. C. Papadopoulos. Equalization of multiuser channels. In V. Poor and G. Wornell, editors, *Wireless Communications: Signal Processing Perspectives*, pages 129–179. Prentice Hall, 1998.

[26] A. J. Paulraj, C. B. Papadias, V. U. Reddy, and A. J. van der Veen. Blind space-time signal processing. In V. Poor and G. Wornell, editors, *Wireless Communications: Signal Processing Perspectives*, pages 179–209. Prentice Hall, 1998.

[27] R. Price and P.E. Green. A communication technique for multipath channels. *Proceedings of the IRE*, 46:555–570, March 1958.

[28] J. G. Proakis and M. Salehi. *Contemporary Communication Systems Using Matlab*. PWS Bookware Companion Series, 1997.

[29] W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, Inc., 3 edition, 1976.

[30] W. D. Rummler. A new selective fading model: Application to propagation data. *AT & T Technical Journal*, 58(5):1037–1071, 1979.

[31] W. D. Rummler, R.P. Coutts, and M. Liniger. Multipath fading channel models for microwave digital radio. *IEEE Communications Magazine*, 24(11):30–42, 1986.

[32] W. D. Rummler and M. Liniger. Multipath fading channel models for microwave digital radio. In *Terrestrial Digial Microwave Communication*. Artech House, 1989.

[33] B. Skylar. *Digital Communications: Fundamentals and Applications*. P T R Prentice Hall, 1988.

[34] V. Solo and X. Kong. *Adaptive Signal Processing Algorithms: Stability and Convergence*. Prentice Hall, 1995.

[35] E. K. Wesel. *Wireless Multimedia Communications: Networking Video, Voice, and Data*. Addison-Wesley, 1997.

[36] B. Widrow. Adaptive filters. In R. E. Kalman and N. DeClaris, editors, *Aspects of Network and System Theory*, pages 563–587. New York: Holt, Rinehart and Winston, 1970.

[37] B. Widrow, P.E. Mantey, L. J. Griffiths, and B.B. Goode. Adaptive antenna systems. *Proceedings of the IEEE*, pages 2143–2159, 1967.

[38] B. Widrow, J. McCool, and M. Ball. The complex lms algorithm. *Proceedings of the IEEE*, pages 719–720, April 1975.

[39] O. A. Williams. System impulse response from digital crosscorrelation of additive input noise and output signal. Master's thesis, Georgia Institute of Technology, April 1969.

[40] Q. Zhang, J.Q. Bao, and L. Tong. Applications of blind channel estimation to hf communication. *Proceedings of IEEE Milcom'97*, 3:1077–1081, November 1997.

Appendix A

Matlab Simulation Code

A.1 Channel Models

A.1.1 Rummler Model

```
function h = rummler(tau,Fs)

%   desc: generates the impulse response of a radio relay three-path model (Rummler model)
%         as described in Simulation of Communication Systems, Jeruchim (pg. 379-81).  The
%         model is popular for 6-GHz frequency band LOS microwave links.
%
%  syntax: h = rummler(tau,Fs)
%
%  inputs: tau = differential delay (delay spread) [s]
%          Fs = sampling frequency [Hz]
% outputs: h = sampled impulse response
%
%    prog: Shane M. Haas -- 1999

% generate B, an exponentially distributed RV with mean of 3.8 dB
B = expgen(3.8);

% generate A, a Gaussian distributed RV with mean 24.6*(B^4 + 500)/(B^4 + 800) dB and
% variance of 5 dB;
A = sqrt(5)*randn + 24.6*(B^4 + 500)/(B^4 + 800);

% generate the notch frequency f_o = theta/(2*pi*tau) where theta is a RV
% with a range [-pi pi] with constant densities on each section abs(theta) > pi/2
% and abs(theta) < pi/2 with P{abs(theta) < pi/2} = 5*P{abs(theta) > pi/2}

if rand > 5/6           % theta is uniform on abs(theta) > pi/2
   if rand > 0.5        % theta is uniform on theta > pi/2
      theta = rand*pi/2 + pi/2;
   else                 % theta is uniform on theta < -pi/2
      theta = rand*pi/2 - pi;
   end
else                    % theta is uniform on abs(theta) < pi/2
   theta = rand*pi - pi/2;
end
f_o = theta/(2*pi*tau);

% the channel is equally likely to be minimum or non-minimum phase
min_phase = rand;
if min_phase > 0.5   % minimum phase channel
   a = exp(-A/20);
   b = 1 - exp(-B/20);
   if round(tau*Fs) - 1 >= 0
      h = a*[1 zeros(1,round(tau*Fs) - 1) -b*exp(j*2*pi*f_o*tau)];
   else
      h = a;
   end
else                % non-minimum phase channel
   ab = exp(-A/20);
   b = 1/(1 - exp(-B/10));
   if round(tau*Fs) - 1 >= 0
      h = ab*[-1/b zeros(1,round(tau*Fs) - 1) exp(j*2*pi*f_o*tau)];
   else
      h = -ab/b;
   end
end
```

A.1.2 Mobile Radio Channel

```
function h = mob_rad_chan(doppler_freq,Fs,Tb,num_imp)

%  desc:  generates impulse responses for a mobile radio channel model;
%         the model includes a direct path
%         a path delayed by Tb, and a path delayed by 2*Tb;  the paths
%         have attenuations 0 dB, -20 dB, and -25 dB respectively;  the model
%         assumes a constant velocity mobile with doppler_freq = v/lambda
%         where v = velocity of mobile, lambda = wavelength of carrier
%         (ref: Jeruchim, Simulation of Communication Systems, pg. 382)
%
% syntax: h = mob_rad_chan(doppler_freq,Fs,Tb,num_imp)
%
% inputs: doppler_freq = (velocity of mobile)/(wavelength of carrier);
%            the doppler frequency determines the rate at which the channel changes
%         Fs = sampling frequency [1/s]
%         Tb = bit duration [s] (Fs*Tb must be an integer)
%         num_imp = number of impulse responses to generate
%outputs: h = matrix of column-wise impulse responses; h(:,k) = response at time k
%
%   prog: Shane M. Haas -- 1999

if round(Fs*Tb) ~= Fs*Tb
   error('ERROR in mob_rad_chan:  Fs*Tb must ben an integer')
end

% parameters
n_ss = 2000;       %number of initial points to throw away to simulate steady state

% path attenuations on a linear scale
A0 = 10^(0/10);
A1 = 10^(-20/10);
A2 = 10^(-25/10);
A = [A0 A1 A2];

% path delays in samples
%d1 = round(Tb*Fs/2);
%d2 = round(Tb*Fs);
d1 = round(Tb*Fs);
d2 = round(2*Tb*Fs);

% the time variation in the coefficients is modeled by passing complex
% Gaussian noise through a single-pole Butterworth filter

[b,a] = butter(1,doppler_freq/Fs*2);
h = [randn(num_imp + n_ss,1) + j*randn(num_imp + n_ss,1) , ...
     randn(num_imp + n_ss,1) + j*randn(num_imp + n_ss,1) ,...
     randn(num_imp + n_ss,1) + j*randn(num_imp + n_ss,1)];
h = filter(b,a,h);

% set variance on each tap
pwr = std(h);
h = h./pwr(ones(num_imp + n_ss,1),:).*sqrt(A(ones(num_imp + n_ss,1),:));

% set tap delays
h = [h(n_ss+1:n_ss + num_imp,1) zeros(num_imp,d1 - 1), ...
     h(n_ss+1:n_ss + num_imp,2) zeros(num_imp,d2 - d1 - 1) ,...
     h(n_ss+1:n_ss + num_imp,3)].';
```

A.2 Adaptive Algorithms

A.2.1 Recursive Least Squares (RLS)

A.2.1.1 Ordinary (Training Sequence) Recursive Least Squares

```
function [e,w,y_hat] = crls_offline(x,y,w_o,ff)
%   desc: given a complex primary signal y (signal + noise) and a
%         complex reference signal x (noise) this function
%         adapts the initial weight vector w_o to minimize the
%         mean squared difference between y and the filtered x using the
%         exponentially weighted recursive least squares estimator
%
% syntax: [e,w,y_hat] = crls_offline(x,y,w_o,ff)
%
%  input: x = complex vector of reference signal data
%           = x_i + j*x_q
%         y = complex vector of primary signal data
%           = y_i + j*y_q
%         w_o = initial FIR filter weights
%         ff = the forgetting factor (0 < ff < 1)
% output: e = vector of error signal = primary signal - filtered reference signal
%         w = matrix of adaptive weights (column k contains the set of filter weights at step k; row 1 = w(1,k))
%         y_hat = estimator of primary signal data = filtered reference signal
```

```
%
%  notes: the weights w(:,k) are not updated for the first length(w_o) iterations; hence,
%         this implementation is not suitable for measuring the initial transient responses of
%         the estimator
%
%   prog: Shane Haas 1999
%

N_data = length(x);
if length(y) ~= N_data
     error('ERROR in clms_offline: x and y must be the same length')
end
N_FIR = length(w_o);
if N_FIR ~= round(N_FIR)
   error('ERROR in clms_offline: w_o must have an even number of elements')
end

% note: the rls estimator is implemented using 2x1 vectors to describe complex
%       numbers

x = x(:);
y = y(:);
w = zeros(2*N_FIR,N_data);
e = zeros(2,N_data);
y_hat = zeros(2,N_data);

w0 = [real(w_o(:)); imag(w_o(:))];
w(:,1:N_FIR) = w0(:,ones(1,N_FIR));

H = .0001*eye(2*N_FIR);
for k = N_FIR+1:N_data

  x_ik = flipud(real(x(k-N_FIR+1:k)));        % x_k = [x(k) ... x(k-N_FIR+1)]
  x_qk = flipud(imag(x(k-N_FIR+1:k)));
  x_k = [x_ik x_qk;-x_qk x_ik];

  y_hat(:,k) = x_k'*w(:,k-1);
  e(:,k) = [real(y(k)); imag(y(k))] - y_hat(:,k);

  H = ff*H + x_k*x_k';

  w(:,k) = w(:,k-1) + inv(H)*x_k*e(:,k);

end

% convert back to complex notation
y_hat = y_hat(1,:) + j*y_hat(2,:);
e = e(1,:) + j*e(2,:);
w = w(1:N_FIR,:) + j*w(N_FIR+1:end,:);
```

A.2.1.2 Decision Weighted Recursive Least Squares

```
function [e,w,y_hat,dw] = mpsk_dwcrls_offline(x_hat,dw_hard,y,w_o,ff)
%   desc: performs the decision weighted complex recursive least squares estimate
%         of a communications channel given an estimate of the transmitted sequence
%         x_hat, x_hat's decision weights (0 <= dw <= 1), and the output of the channel y.
%
% syntax: [e,w,y_hat,dw] = mpsk_dwcrls_offline(x_hat,dw_hard,y,w_o,ff)
%
%  input: x_hat = complex vector of reference signal data
%               = x_i_hat + j*x_q_hat
%         dw_hard  = decision weights for each hard decision in x_hat;
%               (= 1 - abs(phase error in decision)/(pi/M) where M = number of symbols for soft decision weighting, e.g.)
%         y = complex vector of primary signal data
%           = y_i + j*y_q
%         w_o = initial FIR filter weights
%         ff = forgetting factor (0 < ff <= 1); setting ff closer to 0 emphasizes
%              more recent measurements in the update of the weights
% output: e = vector of error signal; each row is y(k) - x_hat(k) filtered by w(:,k)
%         w = matrix of adaptive weights (column k contains the set of filter weights at step k; row 1 = w(1,k))
%         y_hat = estimator of primary signal data = filtered reference signal
%         dw = decision weights calculated for each step of the WRLS algorithm
%
%   prog: Shane Haas 1999
%

N_data = length(x_hat);
if length(y) ~= N_data
     error('ERROR in dwcrls_offline: x and y must be the same length')
end
N_FIR = length(w_o);
if N_FIR ~= round(N_FIR)
   error('ERROR in dwcrls_offline: w_o must have an even number of elements')
end

% note: the rls estimator is implemented using 2x1 vectors to describe complex
%       numbers

x = x_hat(:);
y = y(:);
w = zeros(2*N_FIR,N_data);
e = zeros(2,N_data);
y_hat = zeros(2,N_data);

w0 = [real(w_o(:)); imag(w_o(:))];
w(:,1:N_FIR) = w0(:,ones(1,N_FIR));
```

```
H = .01*eye(2*N_FIR);

for k = N_FIR+1:N_data

  x_ik = flipud(real(x(k-N_FIR+1:k)));        % x_k = [x(k) ... x(k-N_FIR+1)]
  x_qk = flipud(imag(x(k-N_FIR+1:k)));
  x_k = [x_ik x_qk;-x_qk x_ik];
  dw_hard_k = flipud(dw_hard(k-N_FIR+1:k));

  dw(k) = prod(dw_hard_k);

  y_hat(:,k) = x_k'*w(:,k-1);
  e(:,k) = [real(y(k)); imag(y(k))] - y_hat(:,k);

  H = ff*H + dw(k)*x_k*x_k';

  w(:,k) = w(:,k-1) + dw(k).*(H\x_k*e(:,k));

end

% convert back to complex notation
y_hat = y_hat(1,:) + j*y_hat(2,:);
e = e(1,:) + j*e(2,:);
w = w(1:N_FIR,:) + j*w(N_FIR+1:end,:);
```

A.2.1.3 Modified Decision Weighted Recursive Least Squares

```
function [e,w,y_hat,dw] = mpsk_mdwcrls_offline(x_hat,dw_hard,y,w_o,ff)
%   desc: performs the modified decision weighted complex recursive least squares estimate
%         of a communications channel given an estimate of the transmitted sequence
%         x_hat, x_hat's decision weights (0 <= dw <= 1), and the output of the channel y.
%
% syntax: [e,w,y_hat,dw] = mpsk_mdwcrls_offline(x_hat,dw_hard,y,w_o,ff)
%
%  input: x_hat = complex vector of reference signal data
%               = x_i_hat + j*x_q_hat
%         dw_hard  = decision weights for each hard decision in x_hat;
%               (= 1 - abs(phase error in decision)/(pi/M) where M = number of symbols for soft decision weighting, e.g.)
%         y = complex vector of primary signal data
%           = y_i + j*y_q
%         w_o = initial FIR filter weights
%         ff = forgetting factor (0 < ff <= 1); setting ff closer to 0 emphasizes
%              more recent measurements in the update of the weights
% output: e = vector of error signal; each row is y(k) - x_hat(k) filtered by w(:,k)
%         w = matrix of adaptive weights (column k contains the set of filter weights at step k; row 1 = w(1,k))
%         y_hat = estimator of primary signal data = filtered reference signal
%         dw = decision weights calculated for each step of the WRLS algorithm
%
%   prog: Shane Haas 1999
%

N_data = length(x_hat);
if length(y) ~= N_data
    error('ERROR in dwcrls_offline: x and y must be the same length')
end
N_FIR = length(w_o);
if N_FIR ~= round(N_FIR)
   error('ERROR in dwcrls_offline: w_o must have an even number of elements')
end

% note: the rls estimator is implemented using 2x1 vectors to describe complex
%       numbers

x = x_hat(:);
y = y(:);
w = zeros(2*N_FIR,N_data);
e = zeros(2,N_data);
y_hat = zeros(2,N_data);

w0 = [real(w_o(:)); imag(w_o(:))];
w(:,1:N_FIR) = w0(:,ones(1,N_FIR));

H = .01*eye(2*N_FIR);

for k = N_FIR+1:N_data

  x_ik = flipud(real(x(k-N_FIR+1:k)));        % x_k = [x(k) ... x(k-N_FIR+1)]
  x_qk = flipud(imag(x(k-N_FIR+1:k)));
  x_k = [x_ik x_qk;-x_qk x_ik];
  dw_hard_k = flipud(dw_hard(k-N_FIR+1:k));

  dw(k) = prod(dw_hard_k);

  y_hat(:,k) = x_k'*w(:,k-1);
  e(:,k) = [real(y(k)); imag(y(k))] - y_hat(:,k);

  H = ff*H + x_k*x_k';

  w(:,k) = w(:,k-1) + dw(k).*(H\x_k*e(:,k));

end

% convert back to complex notation
y_hat = y_hat(1,:) + j*y_hat(2,:);
e = e(1,:) + j*e(2,:);
```

```
w = w(1:N_FIR,:) + j*w(N_FIR+1:end,:);
```

A.2.2 Least Mean Squares (LMS)

A.2.2.1 Ordinary (Training Sequence) Least Mean Squares

A.2.2.1.1 Using Complex Numbers

```
function [e,w,y_hat] = clms_offline(x,y,w_o,gain)
%   desc: given a complex primary signal y (signal + noise) and a
%         complex reference signal x (noise) this function
%         adapts the initial weight vector w_o to minimize the
%         mean squared difference between y and the filtered x
%
% syntax: [e,w,y_hat] = clms_offline(x,y,w_o,gain)
%
%  input: x = complex vector of reference signal data
%           = x_i + j*x_q
%         y = complex vector of primary signal data
%           = y_i + j*y_q
%         w_o = initial FIR filter weights
% output: e = vector of error signal = primary signal - filtered reference signal
%         w = matrix of adaptive weights (column k contains the set of filter weights at step k; row 1 = w(1,k))
%         y_hat = estimator of primary signal data = filtered reference signal
%
%   prog: Shane Haas 1999
%

N_data = length(x);
if length(y) ~= N_data
     error('ERROR in clms_offline2: x and y must be the same length')
end
N_FIR = length(w_o);

x = x(:);
y = y(:);
w = zeros(N_FIR,N_data);
e = zeros(1,N_data);
y_hat = zeros(1,N_data);

w_o = w_o(:);
w(:,1:N_FIR) = w_o(:,ones(1,N_FIR));

for k = N_FIR:N_data

  x_k = flipud(x(k-N_FIR+1:k));

  y_hat(k) = w(:,k).'*x_k;
  e(k) = y(k) - y_hat(k);
  if k ~= N_data
     w(:,k+1) = w(:,k) + gain*(e(k))*conj(x_k);
  end
end
```

A.2.2.1.2 Using Vectors to Represent Complex Numbers

```
function [e,w,y_hat] = clms_offline3(x,y,w_o,gain)
%   desc: given a complex primary signal y (signal + noise) and a
%         complex reference signal x (noise) this function
%         adapts the initial weight vector w_o to minimize the
%         mean squared difference between y and the filtered x
%
% syntax: [e,w,y_hat] = clms_offline3(x,y,w_o,gain)
%
%  input: x = complex vector of reference signal data
%           = x_i + j*x_q
%         y = complex vector of primary signal data
%           = y_i + j*y_q
%         w_o = initial FIR filter weights
% output: e = vector of error signal = primary signal - filtered reference signal
%         w = matrix of adaptive weights (column k contains the set of filter weights at step k; row 1 = w(1,k))
%         y_hat = estimator of primary signal data = filtered reference signal
%
%   prog: Shane Haas
%

N_data = length(x);
if length(y) ~= N_data
     error('ERROR in clms_offline: x and y must be the same length')
end
N_FIR = length(w_o);
if N_FIR ~= round(N_FIR)
   error('ERROR in clms_offline: w_o must have an even number of elements')
end
x = x(:);
y = y(:);
w = zeros(2*N_FIR,N_data);
e = zeros(2,N_data);
y_hat = zeros(2,N_data);

w(:,1) = [real(w_o(:)); imag(w_o(:))];
```

```
for k = N_FIR:N_data

  x_ik = flipud(real(x(k-N_FIR+1:k)));        % x_k = [x(k) ... x(k-N_FIR+1)]
  x_qk = flipud(imag(x(k-N_FIR+1:k)));
  x_k = [x_ik x_qk;-x_qk x_ik];

  y_hat(:,k) = x_k'*w(:,k);
  e(:,k) = [real(y(k)); imag(y(k))] - y_hat(:,k);
  if k ~= N_data
    w(:,k+1) = w(:,k) + gain*x_k*e(:,k);
  end
end

y_hat = y_hat(1,:) + j*y_hat(2,:);
e = e(1,:) + j*e(2,:);
w = w(1:N_FIR,:) + j*w(N_FIR+1:end,:);
```

A.2.2.2 Decision Weighted Recursive Least Mean Squares

```
function [e,w,y_hat,dw] = mpsk_dwclms_offline(x_hat,dw_hard,y,w_o,gain)
%   desc: performs the decision weighted complex least mean squares estimate
%         of a communications channel given an estimate of the transmitted sequence
%         x_hat, x_hat's decision weights dw_hard for each hard decision, and the output of the channel y.
%
% syntax: [e,w,y_hat,dw] = mpsk_pwclms_offline(x_hat,dw_hard,y,w_o,gain)
%
%  input: x_hat = complex vector of reference signal data
%               = x_i_hat + j*x_q_hat
%         dw_hard = decision weight for each hard decision in x_hat;
%               (= 1 - abs(phase error in decision)/(pi/M) where M = number of symbols for MPSK, e.g.)
%         y = complex vector of primary signal data
%           = y_i + j*y_q
%         w_o = initial FIR filter weights
% output: e = vector of error signal; each row is y(k) - x_hat(k) filtered by w(:,k)
%         w = matrix of adaptive weights (column k contains the set of filter weights at step k; row 1 = w(1,k))
%         y_hat = estimator of primary signal data = filtered reference signal
%         dw = vector of calculated decision weights for each step of the lms algorithm
%
%   prog: Shane Haas 1999
%

N_data = length(x_hat);
if length(y) ~= N_data
     error('ERROR in mpsk_dwclms_offline: x_hat and y must be the same length')
end
N_FIR = length(w_o);

x_hat = x_hat(:);
y = y(:);
w = zeros(N_FIR,N_data);
e = zeros(1,N_data);
y_hat = zeros(1,N_data);

w_o = w_o(:);
w(:,1:N_FIR) = w_o(:,ones(1,N_FIR));

for k = N_FIR:N_data

  x_k = flipud(x_hat(k-N_FIR+1:k));
  dw_hard_k = flipud(dw_hard(k-N_FIR+1:k));

  y_hat(k) = w(:,k).'*x_k;
  e(k) = y(k) - y_hat(k);

  dw(k) = prod(dw_hard_k);

  if k ~= N_data
     w(:,k+1) = w(:,k) + dw(k)*gain*(e(k))*conj(x_k);
  end
end
```

A.3 Algorithm Comparison

A.3.1 Adaptive Channel Estimation Example

```
% Script for Blind Channel Estimation Example
% Shane M. Haas (1999)

%------------------------
% Simulation Parameters
%------------------------

% random number generator state vectors
```

```
if exist('LOCKSTATE') == 1
   rand('state',ustate);
   randn('state',nstate);
else
   rand('state',sum(100*clock));
   ustate = rand('state');
   randn('state',sum(101*clock));
   nstate = randn('state');
end

% noise parameters
ISNR = 1000; %dB       %SNR of noise to add to I channel
QSNR = 1000; %dB       %SNR of noise to add to Q channel

% estimator parameters
lms_gain = .3;       %LMS algorithm gain
ff = 0.99;           %RLS algorithm forgetting factor

% simulation run parameters
N_data = 300;        %number of symbols to generate in each simulation run
Fs = 1;              %normalized sampling frequency
Tb = 4;              %normalized symbol interval

% modulation parameters
M = 4;               %number of symbols
Eb = 1;              %energy per bit

% channel parameters
tau = 8;             %delay spread of Rummler channel

% performance parameters
n_skip = 100*Tb*Fs;        %number of initial points to skip to measure steady-state

% insert symbol error?
INSERT_ERROR = 0;          % 1 = insert symbol error 3/4 way through simulation; 0 = don't insert error

%------------------------
% Memory Allocation
%------------------------

err = zeros(6,round(Fs*Tb)*N_data);

%------------------------
% Simulation
%------------------------

% generate channel
h = rummler(tau,Fs);
h_o = h;
%h_o = zeros(size(h));

% synchronize receiver to channel's rotation of constellation
figure(15); subplot(2,1,2)
mpsk_ang = mpsk_synch(M,Eb,h,Fs,Tb,1);

% generate signals
s = sym_gen(M,N_data);
[x,t] = mpsk_gen(s,Eb,M,Fs,Tb);

% generate output of channel
z = filter(h,1,x);

% estimate signal power entering receiver
s_pwr = (std(z)).^2;
n_i = sqrt(s_pwr/(10^(ISNR/10)))*randn(size(x));
n_q = sqrt(s_pwr/(10^(QSNR/10)))*randn(size(x));
y = z + n_i + j*n_q;

% generated detected sequence
figure(15);hold on
[s_hat,phi_err] = mpsk_rcvr(y,mpsk_ang,Fs,Tb,1); hold off;
[s_hat,end_skip] = sym_synch(s,s_hat,ceil(length(h)/Tb));

% insert error (optional -- for testing purposes)
if INSERT_ERROR
   if s(round(0.75*length(s))) == 1
      s_hat(round(0.75*length(s))) = 2;
   else
      s_hat(round(0.75*length(s))) = 1;
   end
end

% reconstruct transmitted signal
x_hat = mpsk_gen(s_hat,Eb,M,Fs,Tb);
t2 = t(1:end-end_skip*Fs*Tb);

% calculate soft decisions
x_soft = 1 - phi_err/(pi/M);
x_soft = x_soft(ones(1,round(Fs*Tb)),:); x_soft = x_soft(:);
dw_ideal = abs(x_hat - x) < 4*eps;

% calculate SER
n_sym = 1:N_data-end_skip;
n_smpl = 1:length(x) - end_skip*Fs*Tb;
se_loc = find(s(n_sym) ~= s_hat(n_sym));             %symbol error locations
num_bad_det = length(se_loc);
SER = num_bad_det/(N_data - end_skip);
fprintf('Avg SER = %1.4e (ds = %1.2e)\n',SER,tau);

% call channel estimator algorithms

[e_tlms,w_tlms,y_hat_tlms] = clms_offline2(x(n_smpl),y(n_smpl),h_o,lms_gain);                                    %training/probe sim
```

```
[e_blms,w_blms,y_hat_blms] = clms_offline2(x_hat(n_smpl),y(n_smpl),h_o,lms_gain);                                    %std blind sim
[e_sdwlms,w_sdwlms,y_hat_sdwlms,dw_sdwlms] = mpsk_dwclms_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,lms_gain);    %soft dw blind sim
[e_idwlms,w_idwlms,y_hat_idwlms,dw_idwlms] = mpsk_dwclms_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,lms_gain);  %ideal dw blind sim
[e_trls,w_trls,y_hat_trls] = crls_offline(x(n_smpl),y(n_smpl),h_o,ff);                                               %training/probe sim
[e_brls,w_brls,y_hat_brls] = crls_offline(x_hat(n_smpl),y(n_smpl),h_o,ff);                                           %std blind sim
[e_sdwrls,w_sdwrls,y_hat_sdwrls,dw_sdwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);       %soft dw blind sim
[e_idwrls,w_idwrls,y_hat_idwrls,dw_idwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);     %ideal dw blind sim
[e_msdwrls,w_msdwrls,y_hat_msdwrls,dw_msdwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);       %modified soft dw blind sim
[e_midwrls,w_midwrls,y_hat_midwrls,dw_midwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);     %modified ideal dw blind sim

% calculate average distance from true channel response
h_true = h(:);

dist_tlms = mean((abs(h_true(:,ones(1,size(w_tlms,2)))-w_tlms)).^2);
dist_blms = mean((abs(h_true(:,ones(1,size(w_blms,2)))-w_blms)).^2);
dist_sdwlms = mean((abs(h_true(:,ones(1,size(w_sdwlms,2)))-w_sdwlms)).^2);
dist_idwlms = mean((abs(h_true(:,ones(1,size(w_idwlms,2)))-w_idwlms)).^2);
dist_trls = mean((abs(h_true(:,ones(1,size(w_trls,2)))-w_trls)).^2);
dist_brls = mean((abs(h_true(:,ones(1,size(w_brls,2)))-w_brls)).^2);
dist_sdwrls = mean((abs(h_true(:,ones(1,size(w_sdwrls,2)))-w_sdwrls)).^2);
dist_idwrls = mean((abs(h_true(:,ones(1,size(w_idwrls,2)))-w_idwrls)).^2);
dist_msdwrls = mean((abs(h_true(:,ones(1,size(w_msdwrls,2)))-w_msdwrls)).^2);
dist_midwrls =  mean((abs(h_true(:,ones(1,size(w_midwrls,2)))-w_midwrls)).^2);


fprintf('Simulations Complete\n');

r = n_skip:(N_data - end_skip)*Fs*Tb;          %indices of steady-state measurements
fprintf('LMS - Training Cumulative Average Squared Distance from True Response = %1.2f\n',sum(dist_tlms(r)));
fprintf('LMS - Blind Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_blms(r)));
fprintf('SDWLMS Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_sdwlms(r)));
fprintf('IDWLMS Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_idwlms(r)));
fprintf('RLS - Training Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_trls(r)));
fprintf('RLS - Blind Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_brls(r)));
fprintf('SDWRLS - Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_sdwrls(r)));
fprintf('IDWRLS - Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_idwrls(r)));
fprintf('MSDWRLS - Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_msdwrls(r)));
fprintf('MIDWRLS - Cumulative Average Squared Error from True Response = %1.2f\n',sum(dist_midwrls(r)));

%----------------------
%  Display the results
%----------------------

N_plot = 40;

y_imin = min(real(y));
y_imax = max(real(y));
y_qmin = min(imag(y));
y_qmax = max(imag(y));

htick = union(sort([real(h) imag(h)]),sort([real(h) imag(h)]));

figure(1)
subplot(6,1,1)
   plot(t,real(y));
   nudgeaxis(1.1);
   mark_error2(t,dw_ideal,'g')
   title('Output Signal')
   ylabel('y_i')
   %set(gca,'xlim',[0 N_plot*Tb],'ylim',[y_imin-0.5 y_imax+0.5]);
   set(gca,'xticklabel',[])

subplot(6,1,2)
   plot(t,imag(y));
   nudgeaxis(1.1);
   mark_error2(t,dw_ideal,'g');
   ylabel('y_q')
   %set(gca,'xlim',[0 N_plot*Tb],'ylim',[y_qmin-0.5 y_qmax+0.5]);
   set(gca,'xticklabel',[])

subplot(6,1,3)
   stairs(t,real(x_hat));
   nudgeaxis(1.1);
   mark_error2(t,dw_ideal,'g');
   title('Detected Signal')
   ylabel('x_ihat')
   %set(gca,'xlim',[0 N_plot*Tb],'ylim',[min([s0(:);s1(:)])-0.5 max([s0(:);s1(:)])+0.5]);
   set(gca,'xticklabel',[])

subplot(6,1,4)
   stairs(t,imag(x_hat));
   nudgeaxis(1.1);
   mark_error2(t,dw_ideal,'g');
   xlabel('Time [s]')
   ylabel('x_qhat')
   %set(gca,'xlim',[0 N_plot*Tb],'ylim',[min([s0(:);s1(:)])-0.5 max([s0(:);s1(:)])+0.5]);
   set(gca,'xticklabel',[])

subplot(6,1,5)
   stairs(t,real(x))
   nudgeaxis(1.1);
   mark_error2(t,dw_ideal,'g');
   title('Input Signal')
   ylabel('x_i')
   %set(gca,'xlim',[0 N_plot*Tb],'ylim',[min([s0(:);s1(:)])-0.5 max([s0(:);s1(:)])+0.5]);
   set(gca,'xticklabel',[])

subplot(6,1,6)
   stairs(t,imag(x));
   nudgeaxis(1.1);
   mark_error2(t,dw_ideal,'g');
   xlabel('Time [s]')
```

```
   ylabel('x_q')
   %set(gca,'xlim',[0 N_plot*Tb],'ylim',[min([s0(:);s1(:)])-0.5 max([s0(:);s1(:)])+0.5]);

figure(2); clf
set(gcf,'name','TLMS')
subplot(2,1,1)
   plot(t2,real(e_tlms),'b',t2,imag(e_tlms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_tlms),'b',t2,imag(e_tlms),'r'); hold off
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (TLMS)')
subplot(2,1,2)
   plot(t2,real(w_tlms),'b',t2,imag(w_tlms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_tlms),'b',t2,imag(w_tlms),'r'); hold off;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (TLMS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(3); clf
set(gcf,'name','BLMS')
subplot(2,1,1)
   plot(t2,real(e_blms),'b',t2,imag(e_blms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_blms),'b',t2,imag(e_blms),'r'); hold off
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (BLMS)')
subplot(2,1,2)
   plot(t2,real(w_blms),'b',t2,imag(w_blms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_blms),'b',t2,imag(w_blms),'r'); hold off
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (BLMS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(4); clf
set(gcf,'name','SDWLMS')
subplot(2,1,1)
   plot(t2,real(e_sdwlms),'b',t2,imag(e_sdwlms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_sdwlms),'b',t2,imag(e_sdwlms),'r'); hold off;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (SDWLMS)')
subplot(2,1,2)
   plot(t2,real(w_sdwlms),'b',t2,imag(w_sdwlms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_sdwlms),'b',t2,imag(w_sdwlms),'r'); hold off
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (SDWLMS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(5); clf
set(gcf,'name','IDWLMS')
subplot(2,1,1)
   plot(t2,real(e_idwlms),'b',t2,imag(e_idwlms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_idwlms),'b',t2,imag(e_idwlms),'r'); hold off ;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (IDWLMS)')
subplot(2,1,2)
   plot(t2,real(w_idwlms),'b',t2,imag(w_idwlms),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_idwlms),'b',t2,imag(w_idwlms),'r'); hold off ;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (IDWLMS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(6); clf
set(gcf,'name','TRLS')
subplot(2,1,1)
   plot(t2,real(e_trls),'b',t2,imag(e_trls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_trls),'b',t2,imag(e_trls),'r'); hold off ;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (TRLS)')
subplot(2,1,2)
   plot(t2,real(w_trls),'b',t2,imag(w_trls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_trls),'b',t2,imag(w_trls),'r'); hold off ;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (TRLS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(7); clf
set(gcf,'name','BRLS')
subplot(2,1,1)
   plot(t2,real(e_brls),'b',t2,imag(e_brls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_brls),'b',t2,imag(e_brls),'r'); hold off ;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (BRLS)')
subplot(2,1,2)
```

```
   plot(t2,real(w_brls),'b',t2,imag(w_brls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_brls),'b',t2,imag(w_brls),'r'); hold off ;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (BRLS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(8); clf
set(gcf,'name','SDWRLS')
subplot(2,1,1)
   plot(t2,real(e_sdwrls),'b',t2,imag(e_sdwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_sdwrls),'b',t2,imag(e_sdwrls),'r'); hold off ;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (SDWRLS)')
subplot(2,1,2)
   plot(t2,real(w_sdwrls),'b',t2,imag(w_sdwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_sdwrls),'b',t2,imag(w_sdwrls),'r'); hold off ;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (SDWRLS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(9); clf
set(gcf,'name','IDWRLS')
subplot(2,1,1)
   plot(t2,real(e_idwrls),'b',t2,imag(e_idwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_idwrls),'b',t2,imag(e_idwrls),'r'); hold off ;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (IDWRLS)')
subplot(2,1,2)
   plot(t2,real(w_idwrls),'b',t2,imag(w_idwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_idwrls),'b',t2,imag(w_idwrls),'r'); hold off ;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (IDWRLS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(10); clf
set(gcf,'name','MSDWRLS')
subplot(2,1,1)
   plot(t2,real(e_msdwrls),'b',t2,imag(e_msdwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_msdwrls),'b',t2,imag(e_msdwrls),'r'); hold off ;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (MSDWRLS)')
subplot(2,1,2)
   plot(t2,real(w_msdwrls),'b',t2,imag(w_msdwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_msdwrls),'b',t2,imag(w_msdwrls),'r'); hold off ;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (MSDWRLS)')
   set(gca,'ytick',htick,'ygrid','on');

figure(11); clf
set(gcf,'name','MIDWRLS')
subplot(2,1,1)
   plot(t2,real(e_midwrls),'b',t2,imag(e_midwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(e_midwrls),'b',t2,imag(e_midwrls),'r'); hold off ;
   %set(gca,'xlim',[0 N_plot*Tb])
   ylabel('Error Signal')
   title('Error Signal Feeding Adaptive Algorithm (MIDWRLS)')
subplot(2,1,2)
   plot(t2,real(w_midwrls),'b',t2,imag(w_midwrls),'r')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,real(w_midwrls),'b',t2,imag(w_midwrls),'r'); hold off ;
   xlabel('t [s]')
   %set(gca,'xlim',[0 N_plot*Tb])
   title('Estimated Channel Response Coefficients (MIDWRLS)')
   set(gca,'ytick',htick,'ygrid','on');


figure(12); clf
   set(gcf,'name','Squared Distance (LMS)')
   plot(t2,dist_tlms,'-o',t2,dist_blms,'-x',t2,dist_sdwlms,'-+',t2,dist_idwlms,'-h')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,dist_tlms,'-o',t2,dist_blms,'-x',t2,dist_sdwlms,'-+',t2,dist_idwlms,'-h')
   title('Distance Squared from True Response (LMS)')
   xlabel('Time [s]')
   ylabel('Squared Distance')
   legend('tlms','blms','sdwlms','idwlms')

figure(13); clf
   set(gcf,'name','Squared Distance (RLS)')
   plot(t2,dist_trls,'-*',t2,dist_brls,'-s',t2,dist_sdwrls,'-d',t2,dist_idwrls,'-^',t2,dist_msdwrls,'-v',t2,dist_midwrls,'-p')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,dist_trls,'-*',t2,dist_brls,'-s',t2,dist_sdwrls,'-d',t2,dist_idwrls,'-^',t2,dist_msdwrls,'-v',t2,dist_midwrls,'-p')
   title('Distance Squared from True Response (RLS)')
   xlabel('Time [s]')
   ylabel('Squared Distance')
   legend('trls','brls','sdwrls','idwrls','msdwrls','midwrls')

if INSERT_ERROR
```

```
 figure(12); clf
   set(gcf,'name','Squared Distance (LMS)')
   plot(t2,dist_tlms,'-o',t2,dist_blms,'-x',t2,dist_idwlms,'-h')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,dist_tlms,'-o',t2,dist_blms,'-x',t2,dist_idwlms,'-h')
   title('Distance Squared from True Response (LMS)')
   xlabel('Time [s]')
   ylabel('Squared Distance')
   legend('tlms','blms','idwlms')

 figure(13); clf
   set(gcf,'name','Squared Distance (RLS)')
   plot(t2,dist_trls,'-*',t2,dist_brls,'-s',t2,dist_idwrls,'-^',t2,dist_midwrls,'-p')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,dist_trls,'-*',t2,dist_brls,'-s',t2,dist_idwrls,'-^',t2,dist_midwrls,'-p')
   title('Distance Squared from True Response (RLS)')
   xlabel('Time [s]')
   ylabel('Squared Distance')
   legend('trls','brls','idwrls','midwrls')

end

figure(14); clf
set(gcf,'name','Weights')
subplot(2,1,1)
   plot(t,x_soft,'-x',t,dw_ideal,'-o')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t,x_soft,t,dw_ideal); hold off;
   title('Weights of for Each Hard Decision')
   xlabel('Time [s]')
   ylabel('Weight')
   legend('soft','ideal')
subplot(2,1,2)
   plot(t2,dw_sdwrls,'-x',t2,dw_idwrls,'-o')
   mark_error2(t,dw_ideal,'g'); hold on;
   plot(t2,dw_sdwrls,t2,dw_idwrls); hold off
   title('Weights for DWLMS and DWRLS Algorithms')
   xlabel('Time [s]')
   ylabel('Weight')
   legend('soft','ideal')
```

A.3.2 Performance Versus Delay Spread

```
% Script for Comparing Estimator Error vs Delay Spread
% MPSK Modulation
% Shane M. Haas -- 1999

%------------------------
% Simulation Parameters
%------------------------

% random number generator state vectors
if exist('LOCKSTATE') == 1
   rand('state',ustate);
   randn('state',nstate);
else
   rand('state',sum(100*clock));
   ustate = rand('state');
   randn('state',sum(101*clock));
   nstate = randn('state');
end

% noise parameters
ISNR = 10; %dB       %SNR of noise to add to I channel
QSNR = 10; %dB       %SNR of noise to add to Q channel

% estimator parameters
lms_gain = .3;       %LMS algorithm gain
ff = 0.99;            %RLS algorithm forgetting factor

% individual simulation run parameters
N_data = 300;         %number of symbols to generate in each simulation run
Fs = 1;              %normalized sampling frequency
Tb = 4;              %normalized symbol interval

% modulation parameters
M = 4;                        %number of symbols
Eb = 1;                       %energy per bit

% start and stop simulation parameters
ds_start = Tb;                 %start delay spread
ds_step =  Tb;                 %delay spread increment (must be a multiple of Tb for proper receiver synch)
ds_stop = 5*Tb;                %stop delay spread
ds = ds_start:ds_step:ds_stop; %delay spreads to simulate

% other parameters
n_avg = 20;                        %number of simulation to average for each del spread
n_skip = 100*Fs*Tb;                %number of initial points to skip to measure steady-state
                                   %    n_skip must be larger than 2 or 3 times the length
                                   %    of the channel impulse response

SER_thres = 2e-1;                  %repeat simulations with SER higher than this threshold
SER_count_thres = 10;              %number of simulations to repeat with SER higher than SER_thres before proceeding

%------------------------
```

```
% Memory Allocation
%------------------------
ds_iter = 0;
SER = zeros(n_avg,length(ds));
dist_tlms = zeros(n_avg,length(ds));
dist_blms = zeros(n_avg,length(ds));
dist_sdwlms = zeros(n_avg,length(ds));
dist_idwlms = zeros(n_avg,length(ds));
dist_trls = zeros(n_avg,length(ds));
dist_brls = zeros(n_avg,length(ds));
dist_sdwrls = zeros(n_avg,length(ds));
dist_idwrls = zeros(n_avg,length(ds));
dist_msdwrls = zeros(n_avg,length(ds));
dist_midwrls = zeros(n_avg,length(ds));

%------------------------
% Simulation
%------------------------


for tau = ds;

   ds_iter = ds_iter + 1;
   for sim_iter = 1:n_avg

      SER_inst = inf;
      SER_count = 0;
      while (SER_inst > SER_thres) & (SER_count < SER_count_thres)

         % generate channel
         h = rummler(tau,Fs);
         h_o = h;

         % synchronize receiver to channel's rotation of constellation
         mpsk_ang = mpsk_synch(M,Eb,h,Fs,Tb,0);

         % generate signals
         s = sym_gen(M,N_data);
         [x,t] = mpsk_gen(s,Eb,M,Fs,Tb);

         % generate output of channel
         z = filter(h,1,x);

         % estimate signal power entering receiver
         s_pwr = (std(z)).^2;
         n_i = sqrt(s_pwr/(10^(ISNR/10)))*randn(size(x));
         n_q = sqrt(s_pwr/(10^(QSNR/10)))*randn(size(x));
         y = z + n_i + j*n_q;

         % generated detected sequence
         [s_hat,phi_err] = mpsk_rcvr(y,mpsk_ang,Fs,Tb,0);
         [s_hat,end_skip] = sym_synch(s,s_hat,ceil(length(h)/Tb));

         % calculate SER
         n_sym = 1:N_data-end_skip;
         n_smpl = 1:length(x) - end_skip*Fs*Tb;
         se_loc = find(s(n_sym) ~= s_hat(n_sym));             %symbol error locations
         num_bad_det = length(se_loc);
         SER_inst = num_bad_det/(N_data - end_skip);
         SER(sim_iter,ds_iter) = SER_inst;
         fprintf('Inst SER = %1.4e; Avg SER = %1.4e (ds = %1.2e)\n',SER_inst,mean(SER(1:sim_iter,ds_iter)),tau);

         if (SER_inst >= SER_thres) & (SER_count < SER_count_thres)
            fprintf('WARNING: High SER -- repeating simulation\n')
         elseif (SER_inst >= SER_thres) & (SER_count >= SER_count_thres)
            fprintf('WARNING: High SER but SER_count_thres exceeded; proceeding with simulation\n')
         end

      end

      % reconstructed transmitted signal
      x_hat = mpsk_gen(s_hat,Eb,M,Fs,Tb);
      t2 = t(1:end-end_skip*Fs*Tb);

      % calculate soft decisions
      x_soft = 1 - phi_err/(pi/M);
      x_soft = x_soft(ones(1,round(Fs*Tb)),:); x_soft = x_soft(:);
      dw_ideal = abs(x_hat - x) < 4*eps;

      % call channel estimator algorithms

      [e_tlms,w_tlms,y_tlms] = clms_offline2(x(n_smpl),y(n_smpl),h_o,lms_gain);                          %training/probe sim
      [e_blms,w_blms,y_hat_blms] = clms_offline2(x_hat(n_smpl),y(n_smpl),h_o,lms_gain);                  %std blind sim
      [e_sdwlms,w_sdwlms,y_hat_sdwlms,pw_sdwlms] = mpsk_dwclms_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,lms_gain);  %soft dw blind sim
      [e_idwlms,w_idwlms,y_hat_idwlms,pw_idwlms] = mpsk_dwclms_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,lms_gain);  %ideal dw blind sim

      [e_trls,w_trls,y_hat_trls] = crls_offline(x(n_smpl),y(n_smpl),h_o,ff);                             %training/probe sim
      [e_brls,w_brls,y_hat_brls] = crls_offline(x_hat(n_smpl),y(n_smpl),h_o,ff);                         %std blind sim
      [e_sdwrls,w_sdwrls,y_hat_sdwrls,pw_sdwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);       %soft dw blind sim
      [e_idwrls,w_idwrls,y_hat_idwrls,pw_idwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);       %ideal dw blind sim
      [e_msdwrls,w_msdwrls,y_hat_msdwrls,pw_msdwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);   %modified soft dw blind sim
      [e_midwrls,w_midwrls,y_hat_midwrls,pw_midwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);   %modified ideal dw blind sim


      % calculate average error from true channel response
      h_true = h(:);
      r = n_skip:(N_data - end_skip)*Fs*Tb;           %indices of steady-state measurements
      dist_tlms(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_tlms(:,r),2)))-w_tlms(:,r))).^2));
      dist_blms(sim_iter,ds_iter) =  mean(sum((abs(h_true(:,ones(1,size(w_blms(:,r),2)))-w_blms(:,r))).^2));
      dist_sdwlms(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_sdwlms(:,r),2)))-w_sdwlms(:,r))).^2));
      dist_idwlms(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_idwlms(:,r),2)))-w_idwlms(:,r))).^2));
      dist_trls(sim_iter,ds_iter) =  mean(sum((abs(h_true(:,ones(1,size(w_trls(:,r),2)))-w_trls(:,r))).^2));
```

```
      dist_brls(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_brls(:,r),2)))-w_brls(:,r))).^2));
      dist_sdwrls(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_sdwrls(:,r),2)))-w_sdwrls(:,r))).^2));
      dist_idwrls(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_idwrls(:,r),2)))-w_idwrls(:,r))).^2));
      dist_msdwrls(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_msdwrls(:,r),2)))-w_msdwrls(:,r))).^2));
      dist_midwrls(sim_iter,ds_iter) = mean(sum((abs(h_true(:,ones(1,size(w_midwrls(:,r),2)))-w_midwrls(:,r))).^2));

   end
   fprintf('Simulations for delay spread  = %1.2e complete (%1.0f percent done)\n',tau,100*ds_iter/length(ds));

end


%-------------------------
% Display the Results
%-------------------------

figure(1);clf
plot(ds,median(dist_tlms),'-o',ds,median(dist_blms),'-x',ds,median(dist_sdwlms),'-+',ds,median(dist_idwlms),'-h')
%plot(ds,mean(dist_tlms),'-o',ds,mean(dist_blms),'-x',ds,mean(dist_sdwlms),'-+',ds,mean(dist_idwlms),'-h')
title('Estimation Error as a Function of Delay Spread')
xlabel('Delay Spread [s]')
ylabel('Median Average Squared Error from True Impulse Response')
legend('tlms','blms','sdwlms','idwlms')
grid on

figure(2);clf
plot(ds,median(dist_trls),...
     '-*',ds,median(dist_brls),'-s',ds,median(dist_sdwrls),'-d',ds,median(dist_idwrls),'-^',ds,median(dist_msdwrls),'-v',ds,median(dist_midwrls),'-p')
%plot(ds,mean(dist_trls),...
     %'-*',ds,mean(dist_brls),'-s',ds,mean(dist_sdwrls),'-d',ds,mean(dist_idwrls),'-^',ds,mean(dist_msdwrls),'-v',ds,mean(dist_midwrls),'-p')
title('Estimation Error as a Function of Delay Spread')
xlabel('Delay Spread [s]')
ylabel('Median Average Squared Error from True Impulse Response')
legend('trls','brls','sdwrls','idwrls','msdwrls','midwrls')
grid on
```

A.3.3 Performance Versus SNR

```
% Script for Comparing Estimator Error vs SNR
% MPSK Modulation
% Shane M. Haas -- 1999

%-------------------------
% Simulation Parameters
%-------------------------

% random number generator state vectors
if exist('LOCKSTATE') == 1
   rand('state',ustate);
   randn('state',nstate);
else
   rand('state',sum(100*clock));
   ustate = rand('state');
   randn('state',sum(101*clock));
   nstate = randn('state');
end


% estimator parameters
lms_gain = .3;       %LMS algorithm gain
ff = 0.99;            %RLS algorithm forgetting factor

% individual simulation run parameters
N_data = 300;         %number of symbols to generate in each simulation run
Fs = 1;              %normalized sampling frequency
Tb = 4;              %normalized symbol interval (Fs*Tb must be an integer)

% channel parameters
ds = 1*round(Tb*Fs);

% modulation parameters
M = 4;                        %number of symbols
Eb = 1;                       %energy per bit

% start and stop simulation parameters
snr_start = 0;                      %start SNR
snr_step = 3;                       %SNR increment
snr_stop = 30;                      %stop SNR
snr = snr_start:snr_step:snr_stop;  %SNR to simulate

% other parameters
n_avg = 20;                         %number of simulation to average for each del spread
n_skip = 100*round(Fs*Tb);          %number of initial points to skip to measure steady-state
SER_thres = 2e-1;                   %repeat simulations with SER higher than this threshold
SER_count_thres = 10;               %number of simulations to redo with SER higher than SER_thres before proceeding

%-------------------------
% Memory Allocation
%-------------------------
snr_iter = 0;
SER = zeros(n_avg,length(snr));
dist_tlms = zeros(n_avg,length(snr));
dist_blms = zeros(n_avg,length(snr));
dist_sdwlms = zeros(n_avg,length(snr));
dist_idwlms = zeros(n_avg,length(snr));
```

```
dist_trls = zeros(n_avg,length(snr));
dist_brls = zeros(n_avg,length(snr));
dist_sdwrls = zeros(n_avg,length(snr));
dist_idwrls = zeros(n_avg,length(snr));
dist_msdwrls = zeros(n_avg,length(snr));
dist_midwrls = zeros(n_avg,length(snr));

%------------------------
% Simulation
%------------------------


for IQSNR = snr;

   snr_iter = snr_iter + 1;
   for sim_iter = 1:n_avg

      SER_inst = inf;
      SER_count = 0;
      while (SER_inst > SER_thres) & (SER_count < SER_count_thres)
         SER_count = SER_count + 1;

         % generate channel
         h = rummler(ds,Fs);
         h_o = h;

         % synchronize receiver to channel's rotation of constellation
         mpsk_ang = mpsk_synch(M,Eb,h,Fs,Tb,0);

         % generate signals
         s = sym_gen(M,N_data);
         [x,t] = mpsk_gen(s,Eb,M,Fs,Tb);

         % generate output of channel
         z = filter(h,1,x);

         % estimate signal power entering receiver
         s_pwr = (std(z)).^2;
         n_i = sqrt(s_pwr/(10^(IQSNR/10)))*randn(size(x));
         n_q = sqrt(s_pwr/(10^(IQSNR/10)))*randn(size(x));
         y = z + n_i + j*n_q;

         % generated detected sequence
         [s_hat0,phi_err] = mpsk_rcvr(y,mpsk_ang,Fs,Tb,0);
         [s_hat,end_skip] = sym_synch(s,s_hat0,ceil(length(h)/Tb));

         % calculate SER
         n_sym = 1:N_data-end_skip;
         n_smpl = 1:length(x) - end_skip*Fs*Tb;
         se_loc = find(s(n_sym) ~= s_hat(n_sym));             %symbol error locations
         num_bad_det = length(se_loc);
         SER_inst = num_bad_det/(N_data - end_skip);
         SER(sim_iter,snr_iter) = SER_inst;
         fprintf('Inst SER = %1.4e; Avg SER = %1.4e (IQSNR = %1.2e)\n',SER_inst,mean(SER(1:sim_iter,snr_iter)),IQSNR);

         if (SER_inst >= SER_thres) & (SER_count < SER_count_thres)
            fprintf('WARNING: High SER -- repeating simulation\n')
         elseif (SER_inst >= SER_thres) & (SER_count >= SER_count_thres)
            fprintf('WARNING: High SER but SER_count_thres exceeded; proceeding with simulation\n')
         end

      end

      % reconstruct transmitted signal
      x_hat = mpsk_gen(s_hat,Eb,M,Fs,Tb);
      t2 = t(1:end-end_skip*Fs*Tb);

      % calculate weights
      x_soft = 1 - phi_err/(pi/M);
      x_soft = x_soft(ones(1,round(Fs*Tb)),:); x_soft = x_soft(:);
      dw_ideal = abs(x_hat - x) < 4*eps;


      % call channel estimator algorithms
      [e_tlms,w_tlms,y_tlms] = clms_offline2(x(n_smpl),y(n_smpl),h_o,lms_gain);                        %training/probe sim
      [e_blms,w_blms,y_hat_blms] = clms_offline2(x_hat(n_smpl),y(n_smpl),h_o,lms_gain);                %std blind sim
      [e_sdwlms,w_sdwlms,y_hat_sdwlms,pw_sdwlms] = mpsk_dwclms_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,lms_gain);  %soft dw blind sim
      [e_idwlms,w_idwlms,y_hat_idwlms,pw_idwlms] = mpsk_dwclms_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,lms_gain);  %ideal dw blind sim
      [e_trls,w_trls,y_hat_trls] = crls_offline(x(n_smpl),y(n_smpl),h_o,ff);                           %training/probe sim
      [e_brls,w_brls,y_hat_brls] = crls_offline(x_hat(n_smpl),y(n_smpl),h_o,ff);                       %std blind sim
      [e_sdwrls,w_sdwrls,y_hat_sdwrls,pw_sdwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);        %soft dw blind sim
      [e_idwrls,w_idwrls,y_hat_idwrls,pw_idwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);       %ideal dw blind sim
      [e_msdwrls,w_msdwrls,y_hat_msdwrls,pw_msdwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);   %modified soft dw blind sim
      [e_midwrls,w_midwrls,y_hat_midwrls,pw_midwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);   %modified ideal dw blind sim


      % calculate average error from true channel response
      h_true = h(:);
      r = n_skip:(N_data - end_skip)*Fs*Tb;           %indices of steady-state measurements

      dist_tlms(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_tlms(:,r),2)))-w_tlms(:,r))).^2));
      dist_blms(sim_iter,snr_iter) =  mean(sum((abs(h_true(:,ones(1,size(w_blms(:,r),2)))-w_blms(:,r))).^2));
      dist_sdwlms(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_sdwlms(:,r),2)))-w_sdwlms(:,r))).^2));
      dist_idwlms(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_idwlms(:,r),2)))-w_idwlms(:,r))).^2));
      dist_trls(sim_iter,snr_iter) =  mean(sum((abs(h_true(:,ones(1,size(w_trls(:,r),2)))-w_trls(:,r))).^2));
      dist_brls(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_brls(:,r),2)))-w_brls(:,r))).^2));
      dist_sdwrls(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_sdwrls(:,r),2)))-w_sdwrls(:,r))).^2));
      dist_idwrls(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_idwrls(:,r),2)))-w_idwrls(:,r))).^2));
      dist_msdwrls(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_msdwrls(:,r),2)))-w_msdwrls(:,r))).^2));
      dist_midwrls(sim_iter,snr_iter) = mean(sum((abs(h_true(:,ones(1,size(w_midwrls(:,r),2)))-w_midwrls(:,r))).^2));

   end
```

```
   fprintf('Simulations for SNR = %1.2f [dB] complete (%1.0f percent done)\n',IQSNR,100*snr_iter/length(snr));

end


%-------------------------
% Display the Results
%-------------------------

figure(1);clf
plot(snr,median(dist_tlms),'-o',snr,median(dist_blms),'-x',snr,median(dist_sdwlms),'-+',snr,median(dist_idwlms),'-h')
%plot(snr,mean(dist_tlms),'-o',snr,mean(dist_blms),'-x',snr,mean(dist_sdwlms),'-+',snr,mean(dist_idwlms),'-h',
title('Estimation Error as a Function of SNR')
xlabel('SNR [dB]')
ylabel('Median Average Squared Error from True Impulse Response')
set(gca,'ysca','log')
legend('tlms','blms','sdwlms','idwlms')
grid on

figure(2);clf
plot(snr,median(dist_trls),...
     '-*',snr,median(dist_brls),'-s',snr,median(dist_sdwrls),'-d',snr,median(dist_idwrls),'-^',snr,median(dist_msdwrls),'-v',snr,median(dist_midwrls),'-p')
%plot(snr,mean(dist_trls),...
%     '-*',snr,mean(dist_brls),'-s',snr,mean(dist_sdwrls),'-d',snr,mean(dist_idwrls),'-^',snr,mean(dist_msdwrls),'-v',snr,mean(dist_midwrls),'-p')
title('Estimation Error as a Function of SNR')
xlabel('SNR [dB]')
ylabel('Median Average Squared Error from True Impulse Response')
set(gca,'ysca','log')
legend('trls','brls','sdwrls','idwrls','msdwrls','midwrls')
grid on
```

A.3.4 Performance Versus Doppler Frequency

```
% Script for Comparing Estimator Error vs Doppler Frequency
% MPSK Modulation
% Shane M. Haas -- 1999

%-------------------------
% Simulation Parameters
%-------------------------

% random number generator state vectors
if exist('LOCKSTATE') == 1
   rand('state',ustate);
   randn('state',nstate);
else
   rand('state',sum(100*clock));
   ustate = rand('state');
   randn('state',sum(101*clock));
   nstate = randn('state');
end

% noise parameters
ISNR = 10; %dB       %SNR of noise to add to I channel
QSNR = 10; %dB       %SNR of noise to add to Q channel

% estimator parameters
lms_gain = .3;        %LMS algorithm gain
ff = 0.99;            %RLS algorithm forgetting factor

% individual simulation run parameters
N_data = 300;         %number of symbols to generate in each simulation run
Fs = 1;               %normalized sampling frequency
Tb = 4;               %normalized symbol interval

% modulation parameters
M = 4;                        %number of symbols
Eb = 1;                       %energy per bit

% start and stop simulation parameters
df = logspace(-15,-3,10);  % doppler frequencies to simulate

% other parameters
n_avg = 20;                           %number of simulation to average for each del spread
n_skip = 100*Fs*Tb;                   %number of initial points to skip to measure steady-state
                                      %    n_skip must be larger than 2 or 3 times the length
                                      %    of the channel impulse response

SER_thres = 2e-1;                     %repeat simulations with SER higher than this threshold
SER_count_thres = 10;                 %number of simulations to repeat with SER higher than SER_thres before proceeding


%-------------------------
% Memory Allocation
%-------------------------
df_iter = 0;
SER = zeros(n_avg,length(df));
z = zeros(1,N_data*Fs*Tb);
y = zeros(1,N_data*Fs*Tb);
s_hat = zeros(1,N_data*Fs*Tb);
phi_err = zeros(1,N_data*Fs*Tb);
mpsk_ang = zeros(M,N_data*Fs*Tb);

dist_tlms = zeros(n_avg,length(df));
dist_blms = zeros(n_avg,length(df));
```

```
dist_sdwlms = zeros(n_avg,length(df));
dist_idwlms = zeros(n_avg,length(df));
dist_trls = zeros(n_avg,length(df));
dist_brls = zeros(n_avg,length(df));
dist_sdwrls = zeros(n_avg,length(df));
dist_idwrls = zeros(n_avg,length(df));
dist_msdwrls = zeros(n_avg,length(df));
dist_midwrls = zeros(n_avg,length(df));

%------------------------
% Simulation
%------------------------


for dop_freq = df;

   df_iter = df_iter + 1;
   for sim_iter = 1:n_avg

      SER_inst = inf;
      SER_count = 0;
      while (SER_inst > SER_thres) & (SER_count < SER_count_thres)

         % generate channel
         h_tot = mob_rad_chan(dop_freq,Fs,Tb,N_data*Fs*Tb);
         h_o = h_tot(:,1);
         lh = length(h_o);

         % generate signals
         s = sym_gen(M,N_data);
         [x,t] = mpsk_gen(s,Eb,M,Fs,Tb);

         for k = lh:size(h_tot,2)
            h = h_tot(:,k);

            % generate output of channel
            z(k) = h.'*fliplr(x(k - lh +1:k)).';

            % synchronize receiver to channel's rotation of constellation
            %figure(1); hold on
            mpsk_ang(:,k) = mpsk_synch(M,Eb,h,Fs,Tb,0).';
            %drawnow

            % estimate signal power entering receiver and generate noise
            s_pwr = (std(z(lh:k))).^2;
            n_i = sqrt(s_pwr/(10^(ISNR/10)))*randn;
            n_q = sqrt(s_pwr/(10^(QSNR/10)))*randn;
            y(k) = z(k) + n_i + j*n_q;

         end

         % generated detected sequence
         [s_hat0,phi_err] = mpsk_rcvr(y,mpsk_ang(:,[1:Tb*Fs:N_data*Tb*Fs]),Fs,Tb,0);

         % synchronize symbols
         [s_hat,end_skip] = sym_synch(s,s_hat0,ceil(length(h_o)/Tb));

         % calculate SER
         n_sym = ceil(lh/Fs/Tb):N_data-end_skip;
         n_smpl = n_sym(1)*Fs*Tb+1:n_sym(end)*Fs*Tb;
         se_loc = find(s(n_sym) ~= s_hat(n_sym));             %symbol error locations
         num_bad_det = length(se_loc);
         SER_inst = num_bad_det/length(n_sym);
         SER(sim_iter,df_iter) = SER_inst;
         fprintf('Inst SER = %1.4e; Avg SER = %1.4e (df = %1.2e)\n',SER_inst,mean(SER(1:sim_iter,df_iter)),dop_freq);

         if (SER_inst >= SER_thres) & (SER_count < SER_count_thres)
            fprintf('WARNING: High SER -- repeating simulation\n')
         elseif (SER_inst >= SER_thres) & (SER_count >= SER_count_thres)
            fprintf('WARNING: High SER but SER_count_thres exceeded; proceeding with simulation\n')
         end

      end

      % reconstruct transmitted sequence
      x_hat = mpsk_gen(s_hat,Eb,M,Fs,Tb);
      t2 = t(1:end-end_skip*Fs*Tb);

      % calculate soft decisions
      x_soft = 1 - phi_err/(pi/M);
      x_soft = x_soft(ones(1,round(Fs*Tb)),:); x_soft = x_soft(:);
      dw_ideal = abs(x_hat - x) < 4*eps;


      % call channel estimator algorithms
      [e_tlms,w_tlms,y_tlms] = clms_offline2(x(n_smpl),y(n_smpl),h_o,lms_gain);                        %training/probe sim
      [e_blms,w_blms,y_hat_blms] = clms_offline2(x_hat(n_smpl),y(n_smpl),h_o,lms_gain);                %std blind sim
      [e_sdwlms,w_sdwlms,y_hat_sdwlms,pw_sdwlms] = mpsk_dwclms_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,lms_gain);  %soft dw blind sim
      [e_idwlms,w_idwlms,y_hat_idwlms,pw_idwlms] = mpsk_dwclms_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,lms_gain);  %ideal dw blind sim
      [e_trls,w_trls,y_hat_trls] = crls_offline(x(n_smpl),y(n_smpl),h_o,ff);                           %training/probe sim
      [e_brls,w_brls,y_hat_brls] = crls_offline(x_hat(n_smpl),y(n_smpl),h_o,ff);                       %std blind sim
      [e_sdwrls,w_sdwrls,y_hat_sdwrls,pw_sdwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);       %soft dw blind sim
      [e_idwrls,w_idwrls,y_hat_idwrls,pw_idwrls] = mpsk_dwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);       %ideal dw blind sim
      [e_msdwrls,w_msdwrls,y_hat_msdwrls,pw_msdwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),x_soft(n_smpl),y(n_smpl),h_o,ff);   %modified soft dw blind sim
      [e_midwrls,w_midwrls,y_hat_midwrls,pw_midwrls] = mpsk_mdwcrls_offline(x_hat(n_smpl),dw_ideal(n_smpl),y(n_smpl),h_o,ff);   %modified ideal dw blind sim


      % calculate average error from true channel response
      r = n_skip:length(n_smpl);          %indices of steady-state measurements
      dist_tlms(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_tlms(:,r))).^2));
      dist_blms(sim_iter,df_iter) =  mean(sum((abs(h_tot(:,r)-w_blms(:,r))).^2));
      dist_sdwlms(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_sdwlms(:,r))).^2));
```

```
      dist_idwlms(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_idwlms(:,r))).^2));
      dist_trls(sim_iter,df_iter) =  mean(sum((abs(h_tot(:,r)-w_trls(:,r))).^2));
      dist_brls(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_brls(:,r))).^2));
      dist_sdwrls(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_sdwrls(:,r))).^2));
      dist_idwrls(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_idwrls(:,r))).^2));
      dist_msdwrls(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_msdwrls(:,r))).^2));
      dist_midwrls(sim_iter,df_iter) = mean(sum((abs(h_tot(:,r)-w_midwrls(:,r))).^2));


   end
   fprintf('Simulations for doppler frequency  = %1.2e complete (%1.0f percent done)\n',dop_freq,100*df_iter/length(df));

end


%-------------------------
% Display the Results
%-------------------------

figure(1)
%plot(df,median(dist_tlms),'-o',df,median(dist_blms),'-x',df,median(dist_sdwlms),'-+',df,median(dist_idwlms),'-h')
plot(df,mean(dist_tlms),'-o',df,mean(dist_blms),'-x',df,mean(dist_sdwlms),'-+',df,mean(dist_idwlms),'-h')
set(gca,'xsca','log')
title(sprintf('Estimation Error as a Function of Doppler Frequency (SNR = %1.f dB)',ISNR))
xlabel('Doppler Frequency [Hz]')
ylabel('Median Average Squared Error from True Impulse Response')
legend('tlms','blms','sdwlms','idwlms')
grid on

figure(2)
%plot(df,median(dist_trls),...
%     '-*',df,median(dist_brls),'-s',df,median(dist_sdwrls),'-d',df,median(dist_idwrls),'-^',df,median(dist_msdwrls),'-v',df,median(dist_midwrls),'-p')
plot(df,mean(dist_trls),...
     '-*',df,mean(dist_brls),'-s',df,mean(dist_sdwrls),'-d',df,mean(dist_idwrls),'-^',df,mean(dist_msdwrls),'-v',df,mean(dist_midwrls),'-p')
set(gca,'xsca','log')
title(sprintf('Estimation Error as a Function of Doppler Frequency (SNR = %1.f dB)',ISNR))
xlabel('Doppler Frequency [Hz]')
ylabel('Median Average Squared Error from True Impulse Response')
legend('trls','brls','sdwrls','idwrls','msdwrls','midwrls')
grid on
```

A.4 Data Generation, Modulation, and Demodulation

A.4.1 Binary Data Generator

```
function data = bin_data_gen(data_length)

%   desc:  generates a binary (1 or 0) random data sequence
%
% syntax:  data = bin_data_gen(data_length)
%
% inputs:  data_length = length of random data sequence
% output:  data = real sequence of binary (1 or 0) digits
%
%   prog:  Shane M. Haas 1999

data = (sign(randn(1,data_length))+1)/2;
```

A.4.2 Symbol Generator

```
function y = sym_gen(M,seq_len)

%    desc: generates seq_len symbols (integers from 1 to M) randomly
%
%  syntax: y = sym_gen(M,seq_len)
%
%  inputs: M = number of symbols to choose from
%          seq_len = number of symbols to generate
% outputs: y = sequence of symbols
%
%    prog: Shane M. Haas

y = ceil(rand(1,seq_len)*M);
```

A.4.3 MPSK Baseband Modulator

```
function [y,t] = mpsk_gen(sym_seq,Eb,M,Fs,Tb)

%    desc:  Multiple phase shift keying (mpsk) complex waveform generator
%
%  syntax:  [y,t] = mpsk_gen(sym_seq,Eb,M,Fs,Tb)
%
%  inputs:  sym_seq = symbol sequence to modulate (integers ranging from 1 to M)
%           Eb = signal energy per symbol interval = sum(abs(y(1 symbol interval).^2))
%           M = number of symbols
%           Fs = sampling frequency
%           Tb = bit duration (note: num of samples per sym interval =
%                Fs*Tb must be an integer)
% outputs:  y = sampled complex baseband mpsk waveform
%           t = time vector associated with y
%
%    prog:  Shane M. Haas -- 1999

num_sps = round(Fs*Tb);
if num_sps ~= Fs*Tb
   error('ERROR in mpsk_gen: number of samples per symbol interval must be an integer');
end

y = sym_seq(ones(1,num_sps),:);
for sym = 1:M
   y(find(y == sym)) = sqrt(Eb/num_sps)*cos(2*pi*sym/M) + j*sqrt(Eb/num_sps)*sin(2*pi*sym/M);
end
y = y(:).';

t = 0:1/Fs:Tb*length(sym_seq)-1/Fs;
```

A.4.4 MPSK Baseband Demodulator

```
function [s,phi_err] = mpsk_rcvr(r,mpsk_ang,Fs,Tb,plot_flag)

%    desc:  Multiple phase shift keying (mpsk) complex baseband demodulator
%
%  syntax:  [s,phi_err] = mpsk_rcvr(r,mpsk_ang,Fs,Tb,plot_flag);
%
%  inputs:  r = complex received signal
%           mpsk_ang = angles of mpsk signal constellation (radians from -pi to pi)
%           Fs = sampling frequency
%           Tb = bit duration (note: num of samples per sym interval =
%                Fs*Tb must be an integer)
%           plot_flag = flag to plot the constellation
%                and eye diagram (1 = plot, 0 = no plot)
% outputs:  s = recovered symbols
%           phi_err = abs error in radians for each symbol angle as specified in mpsk_ang
%
%    prog:  Shane M. Haas -- 1999

num_sps = Fs*Tb;
if num_sps ~= round(num_sps)
   error('ERROR in mpsk_rcvr: number of samples per symbol interval must be an integer');
end

num_sym = length(r)/num_sps;
if num_sym ~= round(num_sym)
   error('ERROR in mpsk_rcvr: r must contain an integer number of symbol intervals');
end

% reshape receive vector into symbol intervals
sym_matrix = reshape(r,num_sps,num_sym);

% integrate over symbol intervals
z = sum(sym_matrix);

% compute angle of resultant sum
phi_hat = angle(z);

% find minimum distance of each angle to true symbol angle

if size(mpsk_ang,2) == num_sym
   ang_mat = mpsk_ang;
   phi_mat = phi_hat(ones(1,size(mpsk_ang,1)),:);
else
   phi_mat = phi_hat(ones(1,length(mpsk_ang)),:);
   mpsk_ang = mpsk_ang(:);
   ang_mat = mpsk_ang(:,ones(1,num_sym));
end

dist = ang_dist(phi_mat,ang_mat);
[phi_err,s] = min(dist);

% plot the eye diagram and constellation
if plot_flag
   subplot(2,1,1)
   num_sym_eye = min(factor(num_sym));
   plot(real(reshape(r,num_sym_eye*num_sps,num_sym/num_sym_eye)),'-ob');hold on
   plot(imag(reshape(r,num_sym_eye*num_sps,num_sym/num_sym_eye)),'-or');hold off
```

```
   title('Eye Diagram of Received Sequence')
   xlabel('Samples')
   subplot(2,1,2)
   plot(z,'.')
   title('Constellation Diagram of Received Sequence')
   xlabel('I')
   ylabel('Q')
   axis equal;
end
```

A.4.5 Angular Distance Function

```
function d = ang_dist(x,y)

%    desc:  finds the angular distance element-wise of x and y, two matrices containing
%           containing angles from -pi to pi in radians
%
%  syntax:  d = ang_dist(x,y)
%
%  inputs: x,y = matrices with elements from -pi to pi in radians
% outputs: d = anglular distance between elements of x and y
%
%    prog: Shane M. Haas -- 1999

if size(x) ~= size(y)
   error('ERROR in ang_dist: size(x) must equal size(y)')
end

if any(abs(x) > pi | abs(y) > pi)
   error('ERROR in ang_dist: all elements of x and y must be between -pi and pi')
end

d = zeros(size(x));

% case #1: (x > 0 and y > 0) or (x < 0 and y < 0)
c1 = (x >= 0 & y >= 0) | (x <= 0 & y <= 0);
d(c1) = abs(x(c1) - y(c1));

% case #2: x > 0 and y < 0
c2 = x >= 0 & y <= 0;
d(c2) = min(abs(x(c2) - y(c2) - 2*pi),abs(x(c2) - y(c2)));

% case #3: x < 0 and y > 0
c3 = x <= 0 & y >= 0;
d(c3) = min(abs(y(c3) - x(c3) - 2*pi),abs(y(c3) - x(c3)));
```

A.4.6 MPSK Decision Threshold Generator

```
function mpsk_ang = mpsk_synch(M,Eb,h_chan,Fs,Tb,plot_flag)

%    desc:  multiple phase shift keying (mpsk) phase synchronizer
%           for complex baseband demodulator mpsk_rcvr; passes each symbol
%           through the channel to determine how the channel rotates the
%           signal constellation
%
%  syntax:  mpsk_ang = mpsk_synch(M,Eb,h_chan,Fs,Tb,plot_flag)
%
%  inputs:  M = number of symbols
%           Eb = energy of signalling waveform in each symbol interval
%           h_chan = channel impulse response
%           Fs = sampling frequency
%           Tb = bit duration (note: num of samples per sym interval =
%                Fs*Tb must be an integer)
%           plot_flag = flag to plot the constellation (1 = plot, 0 = no plot)
%
% outputs:  mpsk_ang = angles of mpsk signal constellation (radians from -pi to pi)
%
%    prog:  Shane M. Haas -- 1999

num_sps = Fs*Tb;
if num_sps ~= round(num_sps)
   error('ERROR in mpsk_synch: number of samples per symbol interval must be an integer');
end

% generate each symbol
r = mpsk_gen(1:M,Eb,M,Fs,Tb);

% reshape receive vector into symbol intervals
sym_matrix = reshape(r,num_sps,M);

% pass sym_matrix through the channel
sym_chan = filter(h_chan,1,[sym_matrix; zeros(length(h_chan)-1,M)]);

% integrate over symbol intervals
z = sum(sym_chan);

% compute angle of resultant sum
mpsk_ang = angle(z);
```

```
% plot the constellation
if plot_flag
   plot(z,'o')
   title('Constellation Diagram of Symbols after the Channel')
   xlabel('I')
   ylabel('Q')
   for sym = 1:M
      text(real(z(sym))+0.1,imag(z(sym))+0.1,num2str(sym))
   end
end
```

A.4.7 Symbol Synchronizing Function

```
function [s_hat_out,sym_shift] = sym_synch(s,s_hat_in,max_shift)

%    desc:  symbol synchronizer; shifts s_hat_in to produce an s_hat_out that
%           most closely aligns with s
%
%  syntax: [s_hat_out,sym_shift] = sym_synch(s,s_hat_in);
%
%  inputs: s = true symbol sequence
%          s_hat_in = estimated symbol sequence (possibly delayed (right shifted) version of s)
%          max_shift = maximum number of symbol intervals to shift in search
%             (max_shift < size(s_hat_in)
% outputs: s_hat_out = shifted s_hat_in aligning with s; s_hat_out is padded with a sym_shift
%             number of NaN's to have the same number of elements as s
%          sym_shift = number of symbols s_hat_in was shifted to the left to align it
%             with s
%
%    prog: Shane M. Haas

if max_shift > length(s_hat_in) - 1
   error('ERROR in sym_synch: max_shift must be less than the number of elements in s_hat_in')
end
num_align = zeros(1,max_shift+1);

s_hat_in = s_hat_in(:).';

for i = 0:max_shift
   num_align(i+1) = length(find([s_hat_in(i+1:end) NaN*ones(1,i)] == s));
end
[max_corr,sym_shift] = max(num_align);

s_hat_out = [s_hat_in(sym_shift:end) NaN*ones(1,sym_shift-1)];
sym_shift = sym_shift - 1;
```

A.5 Miscellaneous Functions

A.5.1 Exponential Random Variable Generator

```
function z = expgen(mean)

%    desc: generates a matrix of size size(mean) of independent exponentially distributed
%          random variables with given means
%
%  syntax: z = expgen(mean)
%
%  inputs: mean = mean of rv's (each element is the mean of the corresponding element in z)
% outputs: z = matrix of exponentially distributed RVs
%
%    prog: Shane M. Haas -- 1999
%
%    refr: Leon-Garcia, Probability and Random Processes for Electrical Engineering

u = rand(size(mean));
z = -log(u).*mean;
```

A.5.2 Decision Error Plotter

```
function mark_error2(x,err_loc,color)

%    desc: plots patches that mark the errors designated by err_loc (does not draw top line like mark_error.m does)
%
%  syntax:  mark_error2(err_loc,color)
%
```

```
%  inputs:  x = xaxis vector of plot
%           err_loc = vector of 1's and 0's where err_loc(n) = 1 means no error and err_loc(n) = 0 means an error
%           color = color to make patches
%
%    prog:  Shane M. Haas -- 1999


V = axis;
ymin = V(3);
ymax = V(4);

x = [x(:)' x(end) + mean(diff(x))];
err_loc = err_loc(:)';

err_start = find(diff(err_loc) < 0) + 1;
err_stop  = find(diff(err_loc) > 0) ;


if (err_loc(1) ~= 0) & (err_loc(end) == 0)
   err_stop = [err_stop length(err_loc)];
end
if (err_loc(1) == 0) & (err_loc(end) ~= 0)
   err_start = [1 err_start];
end
if (err_loc(1) == 0) & (err_loc(end) == 0)
   err_start = [1 err_start];
   err_stop = [err_stop length(err_loc)];
end

bg = get(gca,'Color');
for k = 1:length(err_start)
   h =  patch([x(err_start(k)) x(err_stop(k)+1) x(err_stop(k)+1) x(err_start(k))],[ymin ymin ymax ymax],color);
   set(h,'EdgeColor',bg);
end
```

A.5.3 Axis Nudger

```
function axis_out = nudgeaxis(frac)

%    desc:   nudges the vertical axices up and down by a given fraction
%
%  syntax:   axis_out = nudgeaxis(frac)
%
%  inputs:   frac = percent to expand vertical axis (1 = keep same)
% outputs:   axis_out = resulting axis

if frac <= 0
   error('ERROR in nudgeaxis: frac must be strictly greater than zero')
end

v = axis;

ymid = 0.5*(v(3)+v(4));
yrng = v(4)-v(3);

axis_out = [v(1) v(2) ymid-yrng*frac/2 ymid+yrng*frac/2];
axis(axis_out);
```