Blind Acquisition of Short Burst with Per-Survivor Processing (PSP)

Maruf Mohammad

Thesis submitted to the faculty of the

Virginia Polytechnic Institute & State University

in partial fulfillment of the requirements of the degree

Master of Science

in

Electrical Engineering

Approved:

_____

Dr. W. H. Tranter, Chairman

_____ _____

Dr. Brian D. Woerner Dr. Jeffrey H. Reed

November 26, 2002

Blacksburg, Virginia

Keywords: Per-Survivor Processing (PSP), Blind channel estimation, Viterbi Algorithm, Maximum likelihood (ML) detection, DDFSE, M-algorithm.

Blind Acquisition of Short Burst with Per-Survivor Processing (PSP)

Maruf Mohammad

(Abstract)

This thesis investigates the use of Maximum Likelihood Sequence Estimation (MLSE) in the presence of unknown channel parameters. MLSE is a fundamental problem that is closely related to many modern research areas like Space-Time Coding, Overloaded Array Processing and Multi-User Detection. Per-Survivor Processing (PSP) is a technique for approximating MLSE for unknown channels by embedding channel estimation into the structure of the Viterbi Algorithm (VA). In the case of successful acquisition, the convergence rate of PSP is comparable to that of the pilot-aided RLS algorithm. However, the performance of PSP degrades when certain sequences are transmitted.

In this thesis, the blind acquisition characteristics of PSP are discussed. The problematic sequences for any joint ML data and channel estimator are discussed from an analytic perspective. Based on the theory of indistinguishable sequences, modifications to conventional PSP are suggested that improve its acquisition performance significantly. The effect of tree search and list-based algorithms on PSP is also discussed. Proposed improvement techniques are compared for different channels. For higher order channels, complexity issues dominate the choice of algorithms, so PSP with state reduction techniques is considered. Typical misacquisition conditions, transients, and initialization issues are reported.

Acknowledgments

I would like to express my gratitude to my advisor, Dr.William Tranter, for his advice, support, and encouragement. I wish to extend special thanks to Dr. Jeffrey Reed and Dr. Brian Woerner for not only serving on my advising committee, but also for being great mentors both in and out of the classroom.

I also wish to express my appreciation to my friends and colleagues at the Mobile and Portable Radio Research Group (MPRG). Special thanks to James Hicks for his constant advice and inspiration. I also thank him for proof reading my thesis. I would like to thank the outstanding staff at MPRG for making an atmosphere that is conducive to great research work.

Finally, on a personal note, I would like to thank my parents and my wife, who have supported me with encouragement and love, and have always been a source of inspiration for me, and to whom I dedicate this thesis.

Contents

Chapter 1: Introduction 1

1.1. Thesis Outline 3

Chapter 2: Equalization and MLSE 5

2.1 System Model 5

2.2 Channel Model 7

2.3 Equalizers 9

2.3.1 Linear Equalizer 13

2.3.2 Non-Linear Equalizer 14

2.3.2.1 Decision Feedback Equalizer (DFE) 15

2.3.2.2 Maximum Likelihood Sequence Estimator (MLSE) 16

2.4 Trellis Structure 19

2.5 Viterbi Algorithm 20

2.5.1 Complexity of Viterbi Algorithm 26

2.6 State-Reduction Techniques 26

2.6.1 Delayed Decision Feedback Sequence Estimation (DDFSE) 27

2.6.1.1 Complexity of DDFSE 29

2.7 Conclusion 31

Chapter 3: Adaptive MLSE……32

3.1 Non-Blind Adaptive Algorithms……32

3.2 Blind Adaptive Algorithms……33

3.2.1 Conventional Adaptive MLSE (CA-MLSE) 35

3.2.1.1 Channel Update Using LMS..... 37

3.2.1.2 Channel Update Using RLS 38

3.2.2 Per-Survivor Processing (PSP) 39

3.2.2.1 Channel update using LMS 41

3.2.2.2 Channel estimate using RLS..... 41

3.2.3 Generalized Per-Survivor Processing (GPSP) 43

3.2.3.1 GVA Based PSP 45

3.2.3.1.1 Multiple-Survivor PSP 46

3.2.3.2 Variable-Survivor PSP 49

3.2.3.3 Variable Step-Size PSP 50

3.2.3.4 M-Algorithm Based PSP 52

3.2.3.5 T-Algorithm Based PSP 55

3.3 Complexity Analysis……..56

3.4 Conclusion…….61

Chapter 4: Blind Acquisition Properties of PSP..... 63

4.1 Distinguishability of Sequence……..65

4.2 Phase Ambiguity……68

4.3 Performance Metric……...69

4.4 Learning Curves…….70

4.5 Shifting Phenomenon…….73

4.6 Modeling of Transients……..74

4.7 Trellis Splitting……..77

4.8 Conclusion…….79

Chapter 5: Improvement Techniques and Results 80

5.1 Exploiting the RLS in a PSP framework……...82

5.2 Initialization Issues……85

5.2.1 Channel Transients and Trellis Splitting……...85

5.2.2 Smart Initialization……86

5.3 Improving PSP's Acquisition with Frame Segmentation……..87

5.3.1 Segmented PSP (Seg-PSP)……89

5.3.2 Iterative Segmented PSP (Iseg-PSP)……91

5.3.3 Acquisition Performance of Seg-PSP and ISeg-PSP……93

5.3.4 SER Performance……..95

5.4 Acquisition Performance of Generalized PSP (GPSP)……96

5.4.1 MA-PSP with Segmentation……100

5.5 Acquisition Performance of Reduced Complexity PSP……...101

5.5.1 Reduced Complexity PSP with Segmentation……102

5.6 Channel Order Mis-Match……...103

5.7 Conclusion……...104

Chapter 6: Conclusions 105

6.1 Summary of Results 105

6.2 Suggestion for Future Work..... 107

Appendix A: Complexity Analysis for RLS and LMS Algorithms……...108

Bibliography……...110

Vita……..117

List of Tables

Table 3.1: Number of real operations (per received symbol) for different algorithms (known channel). 57
Table 3.2: Number of real operations (per received symbol) for different adaptive algorithms. 58
Table 3.3: Number of survivor required in the MA-PSP for equivalent complexity with $M = 4$ 59
Table 3.4: Memory requirement for various algorithms for metric update and data decoding..... 61
Table 5.1: Parameters used in the simulation. 82
Table 5.2: Over-estimation of channel order for h=[.407 .815 .407], SNR=14 dB. 103
Table 5.3: Under-estimation of channel order for h=[.407 .815 .407], SNR=14 dB. 104
Table A. 1: Memory requirement for RLS and LMS algorithms. 108
Table A. 2: Number of Flops required in RLS and LMS algorithms. 109

List of Figures

Figure 2.1: A typical communication system model. 6
Figure 2.2: Simplified Discrete-time, white noise equivalent system. 7
Figure 2.3: Channel model for $L = 2$. (a) Tapped Delay Line (TDL) structure (b) Impulse Response. 8
Figure 2.4: Block diagram of a communication system with equalizer at the receiver. 11
Figure 2.5: Linear transversal equalizer. 14
Figure 2.6: Decision feedback equalizer (DFE). 16
Figure 2.7: Tree diagram for BPSK with $L = 2$. 18
Figure 2.8: Trellis structure and state transitions for BPSK with $L = 2$. 19
Figure 2.9: Viterbi Algorithm keeps path with minimum cost metric at each state. 22
Figure 2.10: Trace back in Viterbi Algorithm. 24
Figure 2.11: An example of VA for QPSK with $L = 2$. 25
Figure 2.12: Exploiting channel memory with BPSK and $L = 4$ (a) Full-State MLSE (b) DDFSE, $\mu = 2$ 30
Figure 3.1: Block diagram of non-blind channel estimator. 33
Figure 3.2: Structure of CA-MLSE. 36
Figure 3.3: Channel estimation in CA-MLSE for $\mathcal{A} = \{1, -1\}$, $L = 2$ 38
Figure 3.4: Structure of Per-Survivor Processing. 40
Figure 3.5: Channel estimation in PSP for $\mathcal{A} = \{1, -1\}$, $L = 2$ 42
Figure 3.6: Design of recursive algorithms with path dependent metric. 44
Figure 3.7: Multiple-Survivor PSP for $S = 2$. 47
Figure 3.8: Channel update in Variable-Survivor PSP for $S = 4, N = 2$ 50

Figure 3.9: Channel update in Variable Step-Size PSP. 52
Figure 3.10: M-algorithm trims the tree-search with $N = 4$ best survivors at each stage; $\mathcal{A} = \{1, -1\}$, $L = 4$. 54
Figure 3.11: Complexity (per symbol) as a function of channel order for BPSK. 60
Figure 3.12: Complexity (per symbol) as a function of channel order for QPSK. 60
Figure 4.1: Slot structure of EDGE system. 64
Figure 4.2: Learning curves for PSP. 70
Figure 4.3: Comparison of learning curves for PSP and RLS algorithms. 71
Figure 4.4: The histogram of symbol errors in PSP. 72
Figure 4.5: Shifting phenomenon is responsible for most of the misacquisitions. 73
Figure 4.6: Acquisition problems in PSP. 74
Figure 4.7: Modeling of transients in simulation. 75
Figure 4.8: Trellis initialization and termination by considering transients for $\mathcal{A} = \{1, -1\}, L = 3$; (a) Beginning transients (b) End transients. 76
Figure 4.9: Trellis splitting in PSP for $\mathcal{A} = \{1, -1\}, L = 2$ 77
Figure 4.10: Trellis splitting for $\mathcal{A} = \{1, -1, j, -j\}, L = 2$ 78
Figure 5.1: Channel A characteristics: Impulse response (left), frequency response (right). 81
Figure 5.2: Effect of forgetting factor on indistinguishable sequence. 83
Figure 5.3: Impact of forgetting factor on acquisition rate. 84
Figure 5.4: Proper initialization can alleviate trellis splitting problem. 86
Figure 5.5: Smart initialization improves PSP's acquisition performance. 87
Figure 5.6: Block diagram of overlapping segments (50% overlap). 89
Figure 5.7: Schematic view of Frame segmentation approach. 90
Figure 5.8: Correct convergence can be predicted by the trajectory of accumulated cost metric. 91
Figure 5.9: Block diagram of *Iterative* PSP with frame segmentation. 92
Figure 5.10: Acquisition performance with segmentation for Channel A. 93
Figure 5.11: Acquisition performance with segmentation for Channel B. 94
Figure 5.12: Acquisition performance with segmentation for Channel C. 94
Figure 5.13: Block diagram used in the simulation of SER performance. 96

Figure 5.14: SER performance of PSP on segmented frame for Channel A. 96
Figure 5.15: Acquisition performance comparison among PSP, MA-PSP, MS-PSP for channel A. 97
Figure 5.16: Acquisition performance comparison among PSP, MA-PSP, MS-PSP for channel B. 98
Figure 5.17: Acquisition performance as a function of complexity for channel A. 99
Figure 5.18: Acquisition performance as a function of complexity for channel B..... 99
Figure 5.19: Performance comparison between VA-PSP and MA-PSP with segmentation. 100
Figure 5.20: Performance comparison between reduced complexity algorithms: DDFSE-PSP and MA-PSP. 101
Figure 5.21: Acquisition performance of reduced complexity algorithms with segmentation. 102
Figure A. 1: Computational requirement in RLS and LMS as a function of channel order. 109

Chapter 1: Introduction

Wireless communication systems are developing rapidly to provide voice, data, and multimedia services. These services require large capacity with high data rate transmission. One of the major obstacles to high data rate transmission is the mutipath channel, which causes Inter-Symbol Interference (ISI) at the receiver. Extensive signal processing expertise is required to combat ISI. Over the years, researchers have made significant progress in the field of high data rate transmission with innovative signal processing techniques.

Equalization is a well-known signal processing scheme used to undo the effect of multipath. The two most common equalizers are the Linear Equalizer (LE) and the Decision Feedback Equalizer (DFE) [5], [6]. Linear equalizers try to compensate for channel induced ISI by doing the inverse operation on the received signal. Even though LE is simple, it has the disadvantage of noise enhancement, in the case of channels with deep spectral nulls. DFE cancels the effect of multipath by utilizing the feedback based on detected symbols. Minimum Mean Squared Error (MMSE) DFE doesn't suffer from noise enhancement and therefore outperforms LE. However, since detected symbols are used as feedback information, the DFE's performance degrades due to error propagation.

The optimum detector in the ISI channel is the Maximum Likelihood Sequence Estimator (MLSE), which tests all possible data sequences and selects the most probable one as the output. Usually a brute force implementation of MLSE is computationally intensive, but the Viterbi Algorithm (VA) provides an efficient recursive implementation [12]. The complexity of a Viterbi Equalizer (VEQ) is proportional to the number of states

maintained in the trellis, which increases exponentially with the channel order. Thus, for large order channels or higher order modulation format, the complexity of VEQ quickly becomes prohibitively intensive. This has motivated many researchers to explore state reduction techniques, which are sub-optimal, but computationally very efficient. In the literature, state reduction techniques have been discussed extensively. One of the most popular methods is Delayed Decision Feedback Sequence Estimation (DDFSE) [21]. In DDFSE the channel states are only partially defined in the trellis. The residual information about the channel state is obtained from the survivor history associated with each state. It has been shown that DDFSE can approach the performance of MLSE for minimum phase channels [21], [22]. Since the performance of DDFSE heavily depends on the correct selection of survivor, it doesn't perform well for non-minimum phase channels.

The VA requires perfect knowledge of the channel parameters (i.e., coefficients, delay spread, order) for optimum sequence estimation. However, in practical systems channel information may not be available, so these parameters must be estimated before equalization is performed. In the Conventional Adaptive MLSE (CA-MLSE), data-aided estimation of the channel parameters are used [29]-[33]. The data-aided sequence is obtained by delayed tentative decisions from the survivor history. On the other hand, in Per-Survivor Processing (PSP), channel parameters are estimated by incorporating data-aided estimation techniques into the structure of the VA [35], [36]. PSP is superior to CA-MLSE in the *tracking* mode for fast time-varying channels, since the per-survivor estimator has no delay. Also, it has the potential of blind acquisition[1] since many hypothetical data sequences are considered, as opposed to a single estimator maintained in CA-MLSE. There could be many conceivable applications of PSP, i.e. tracking of a fading channel, state reduction technique, joint Trellis Coded Modulation (TCM) and phase synchronization, blind acquisition of channel parameters, etc.

In PSP, the VA is no longer optimal since the channel is unknown, so the transition metrics are path dependent and the trellis search algorithms become sub-optimal. For path dependent transitions, the optimal algorithm is the tree search. The Generalized

[1] "Blind" acquisition means acquisition without a training sequence.

Viterbi Algorithm (GVA) based PSP incorporates some forms of tree search into the structure of PSP. Multiple-Survivor PSP is an example of GVA-based PSP, which retains more than one survivor per state [55].

Since trellis-based algorithms are not optimal for unknown channels, list-based algorithms are important to consider. One of the most popular list-based algorithms is the M-Algorithm, which keeps an arbitrary number of survivors at each stage of the algorithm [44]. Another example of a cost efficient list-based algorithm is the T-Algorithm, in which the complexity is a user-defined tunable parameter [66].

The question might arise: why focus on blind acquisition when most of the standards use training sequences to estimate the channel? With the implementation of blind acquisition, system throughput and capacity will increase significantly. Also, other related problems like Multi-User Detection (MUD) can benefit from the results of blind acquisition studies. Surprisingly, in the literature, very little has been done in the field of blind acquisition [67]. The objective of this thesis is to explore PSP and its ability to identify the channel. The problems of blind channel estimation have been discussed in general. Modifications to conventional PSP have been suggested to improve its acquisition performance. An innovative technique called “Frame Segmentation” is proposed, which identifies the most conducive portion of the received frame for blind channel identification. The performances of both trellis-based and list-based algorithms have been contrasted in the PSP framework. Also, state reduction techniques have been considered in the case of long channels. This thesis expands upon previous work reported in [67].

1.1 Thesis Outline

The thesis is organized as follows: In Chapter 2, the system model is described. The performance of different equalizers is also presented. The MLSE is derived, and the VA is discussed as an efficient way to implement MLSE in practical systems. Later in the

chapter, DDFSE is described as a method of reduced complexity VEQ. In all of these algorithms, the channel is assumed known.

Chapter 3 provides a detailed analysis of adaptive MLSE for an unknown channel with emphasis on Per-Survivor Processing (PSP). Conventional Adaptive MLSE (CA-MLSE) and PSP have been contrasted from a channel estimator's perspective. The Generalized Viterbi Algorithm (GVA) based PSP is then discussed. List-based algorithms, i.e. the M-algorithm and the T-algorithm, are also presented. The last portion of the chapter deals with the complexity issue of various algorithms.

Chapter 4 starts with an analytical description of joint blind channel and data estimation as described by Chugg [67]. Distinguishability of sequences is defined in terms of their matrix representation. The acquisition problems of PSP are summarized. Transient related problems and trellis initialization issues are also reported.

Chapter 5 presents and analyzes simulation results for various algorithms discussed in previous chapters. Improvement techniques to better PSP's acquisition performance are suggested.

Finally, Chapter 6 concludes the thesis with a summary and a brief discussion on the possible future possible extension of this work.

Chapter 2: Equalization and MLSE

The demand for high data rates has always been a major driving force for communication engineers. The main obstacle to high speed data transmission is the multipath channel (echoes) which impairs the signal observed by a receiver. Equalization is a signal processing technique to combat distortion due to mutipath. In this chapter we will first present the system model. Then performance of different equalizers will be discussed. Later in the chapter we will derive Maximum Likelihood Sequence Estimation (MLSE) and show how the Viterbi Algorithm (VA) can be used to implement MLSE. Also, different state reduction strategies will be analyzed so that MLSE can be practically implemented.

2.1 System Model

A simple but typical communication system model is shown in Figure 2.1. A source symbol, $s[k]$, drawn from an M -ary alphabet,[1] $\mathcal{A}$, is transmitted every T seconds. An encoded complex symbol, $a[k]$, is sent over the complex channel (FIR filter) characterized by the impulse response, $\tilde{h}(t)$. This filter represents the cascade of the transmitter filter and the physical channel. The received signal at the input of the matched filter is given by

[1] For QPSK, $M = 4$, and alphabet, $\mathcal{A} = \{1, -1, j, -j\}$.

$$r(t) = \sum_{k=-\infty}^{\infty} a[k] \cdot \tilde{h}(t - kT) + n(t) \tag{2.1}$$

where the complex noise process is white, Gaussian and independent of the transmitted symbols.

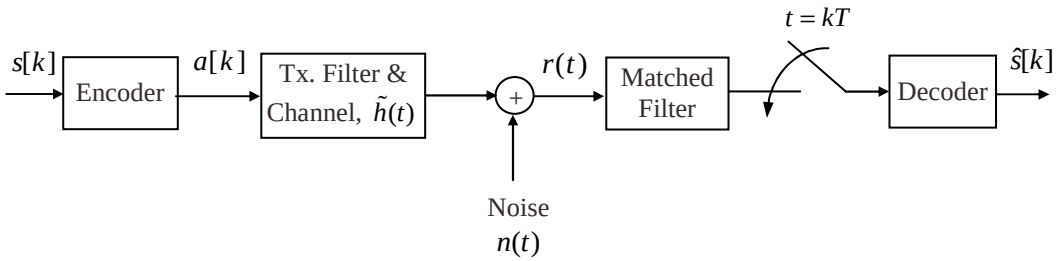


Figure 2.1: A typical communication system model.

Assuming the channel is known, the optimum receiver consists of a filter matched to the impulse of $\tilde{h}(t)$, followed by a sampler sampling at the symbol rate, and a Maximum Likelihood Sequence Estimation (MLSE) decoder. In [1], Forney shows that the matched filter in Figure 1.1 can be replaced by a Whitened Matched Filter (WMF), comprised of both the matched filter and a noise whitening filter, so that the sampled noise sequence is white, and a standard Viterbi decoder, which minimizes the *Euclidean distance metric*, can be used to implement MLSE. We assume that a discrete-time white-noise equivalent of the overall transmitter, channel, and WMF cascade completely characterizes the system that precedes the Viterbi decoder. This simplified model is shown in Figure 2.2. If the impulse response of this equivalent channel is given by $\{h[l]\}_{l=0}^{l=L}$, the received signal at the input of the decoder is

$$r[k] = \sum_{l=0}^{L} h[l]s[k-l] + z[k] \tag{2.2}$$

where L is the memory of the channel[1] and discrete-time noise process, $z[k]$, is zero-mean Gaussian. The WMF can be designed such that the resulting discrete-time

[1] In this thesis, the memory of the channel, L, is assumed finite, since most of the practical channels consist of a finite number of multipath components.

equivalent channel response is minimum phase and shifts the signal energy toward the earliest samples of the channel response. The performance of the Reduced-state Sequence Estimator (RSSE) benefits from this approach [2], [3]. However, MLSE doesn't depend on this condition [1], [4].

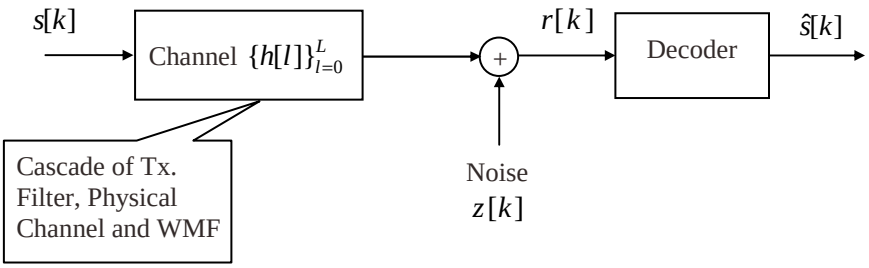


Figure 2.2: Simplified Discrete-time, white noise equivalent system.

2.2 Channel Model

A mobile channel is dynamic due to multipath fading and Doppler spread. Multipath causes different portions of the transmitted signal to arrive at the receiver with different delays, which may introduce Inter-Symbol Interference (ISI) at the receiver. Whether a channel causes significant ISI is dependent on the rms delay spread of the channel impulse response and the symbol rate of the system. The rms delay spread is a multipath channel parameter, which can be determined from a power delay profile [5]. If rms delay spread is very small compared to the symbol period, then the channel impulse response is close to a dirac-delta function. Consequently this type of channel causes little ISI. However, if rms delay is larger than the symbol period, the channel impulse response can be expressed in terms of multiple impulses with different delays. This type of channel causes ISI and is known as a *frequency selective* channel[2] since all the frequency components in the received signal spectrum don't have the same gain. If a channel is resolved into multipath components (Figure 2.3(b)), then the channel is said to have

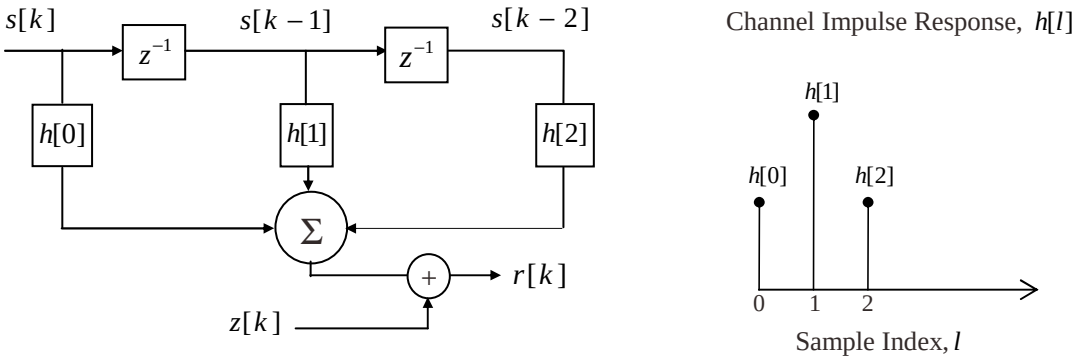
[2] A frequency selective channel is also referred as a time dispersive channel.

memory, with the implication that a given received signal is a combination of present and previously transmitted symbols.

Our system is assumed to operate over a frequency selective multipath channel. We assume a discrete-time, symbol-spaced channel that can be represented with the structure of a Finite Impulse Response (FIR) filter, as shown in Figure 2.3(a). The filter is fed with a transmitted sequence, $\{s[k]\}$, consisting of independent identically distributed (i.i.d.) and equiprobable symbols drawn from the alphabet, $\mathcal{A}$, at time instant k. The channel taps $\{h[l]\}_{l=0}^{l=L}$ are assumed stationary, where the parameter L represents the channel memory. The output of the channel is a linear combination of the present and past symbols (channel states). The received sample at epoch k is given by

$$\begin{aligned} r[k] &= y[k] + z[k] \\ &= \sum_{l=0}^{L} h[l]s[k-l] + z[k] \end{aligned} \tag{2.3}$$

where $y[k]$ is the noiseless component. The additive noise, $z[k]$, is assumed to be discrete-time, complex Gaussian, white, with zero-mean and variance, $\sigma_z^2 = E\left[\left|z[k]\right|^2\right]$.



(a) (b)

Figure 2.3: Channel model for L = 2. (a) Tapped Delay Line (TDL) structure. (b) Impulse Response.

Denoting $\mathbf{h} = \left[h[L], h[L-1], \ldots, h[0]\right]^T$ as the vector of the channel impulse response and $\mathbf{s}_k = \left[s[k-L], s[k-L+1], \ldots, s[k]\right]$ as the portion of transmitted sequence affected by the

channel memory, the received signal at epoch k can be written as

$$r[k] = \mathbf{s}_k \cdot \mathbf{h} + z[k] \tag{2.4}$$

where $[\cdot]^T$ denotes the vector or matrix transposition. The entire transmitted sequence is given by $\mathbf{s} = [s[-L], s[-L+1], \ldots, s[k]]$.

The channel convolution can be written in matrix form as well. A vector of received samples, $\mathbf{r}_k$ can be expressed in terms of the data matrix,[3] $\mathcal{S}_k$, according to (2.5)

$$\underbrace{\begin{bmatrix} r[k] \\ r[k-1] \\ \vdots \\ \vdots \\ r[0] \end{bmatrix}}_{\mathbf{r}_k} = \underbrace{\begin{bmatrix} s[k-L] & s[k-L+1] & \cdots & \cdots & s[k] \\ s[k-L-1] & s[k-L] & \cdots & \cdots & s[k-1] \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s[-L] & s[-L+1] & \cdots & \cdots & s[0] \end{bmatrix}}_{\mathcal{S}_k} \underbrace{\begin{bmatrix} h[L] \\ h[L-1] \\ \vdots \\ \vdots \\ h[0] \end{bmatrix}}_{\mathbf{h}} + \underbrace{\begin{bmatrix} z[k] \\ z[k-1] \\ \vdots \\ \vdots \\ z[0] \end{bmatrix}}_{\mathbf{z}_k} \tag{2.5}$$

where $\mathbf{z}_k$ is a vector of the noise process, $z[k]$, and the toeplitz data matrix, $\mathcal{S}_k$, is given by

$$\mathcal{S}_k = [\mathbf{s}_k, \mathbf{s}_{k-1}, \ldots, \mathbf{s}_0]^T \tag{2.6}$$

2.3 Equalizers

Equalizers are filters used to compensate for frequency selective channels. There are two broad groups of equalization; one that "inverts" the channel at the receiver, and the other that tries to "identify" the channel itself. In the first case, the receiver attempts to compensate for the channel-induced ISI by introducing a filter whose frequency spectrum is inversely proportional to the original channel. The least mean squares linear equalizers and non-linear Decision Feedback equalizers, which are to be presented next, are examples of this type of equalizer. The second class of equalizers estimate the data

[3] In chapter 4, it will be shown that certain sequences are problematic for the joint maximum likelihood data and channel estimation, and these sequences can be defined by their data matrix properties.

sequence by convolving the hypothesized data sequence with some arbitrary channel estimates, comparing it to the received sequence, and choosing the one with the lowest error. In this case, the receiver attempts to exploit the knowledge of the signal format to identify the channel. Maximum Likelihood Sequence Estimation (MLSE) and its variants belong to this class.

Equalizers are designed to work in an unknown and time-varying environment. Its operation includes *tracking* and *training*. Usually a known start-up sequence, which is called a *training sequence*, is used for learning the channel before any decoding [5]-[7]. This is called *training*. The training sequence is designed to permit the equalizer to acquire the proper filter coefficients even in the worst possible channel conditions. Since the mobile fading channel is random and time varying, equalizers must track the time varying characteristics of the channel once it is trained. We need equalizers to be adaptive and to use a recursive algorithm to nullify the effect of the channel. As a consequence, the adaptive equalizer continuously changes its filter characteristics over time. Equalizers require periodic retraining in order to maintain effective ISI cancellation. Time Division Multiple Access (TDMA) systems are well-suited for equalizers since they process data in fixed-length time frames. If the coherence time of the channel is smaller compared to the frame time, the training sequence could be sent at the middle of each frame as well, which is called *mid-amble*.

The time span over which an equalizer converges is a function of the algorithm used, the equalizer structure, and the time rate of change of the multipath radio channel. A wide range of algorithms exists to adapt the channel. Two types of recursive algorithms are widely used for adaptive equalizers. The first one is the Least Mean Square (LMS) algorithm. Though LMS is simple, it converges very slowly. Another algorithm called Recursive Least Square (RLS), converges an order of magnitude faster, but it is computationally intensive. The Kalman RLS algorithm was first proposed by Godard [8] and later others proposed reduced complexity versions [9].

The performance of an equalizer algorithm can be described by the following figures of merits [5], [9]:

Rate of convergence:

The number of iterations required for the algorithm to converge close to the optimum solution in response to stationary inputs.

Tracking ability:

The ability of an equalizer to follow the variation of the channel.

Computations:

The number of arithmetic operations required for one complete iteration of the algorithm.

Error:

A quantitative measure of the deviation of the final value of the mean squared error from the optimal minimum mean squared error.

Numerical stability:

In practical systems, numerical error due to round-off might influence the stability of the algorithm.

Figure 2.4 represents a communication system with an adaptive equalizer. Here the channel represents all the stages between the data input and the equalizer. If $s[k]$ is the original information sequence and $\{h[l]\}_{l=0}^{l=L}$ is the impulse response of the L-order channel, the signal received by the equalizer at epoch k is given by

$$r[k] = \sum_{l=0}^{L} h[l]s[k-l] + z[k] \tag{2.7}$$

where $z[k]$ is the complex AWGN noise at the input of the equalizer.

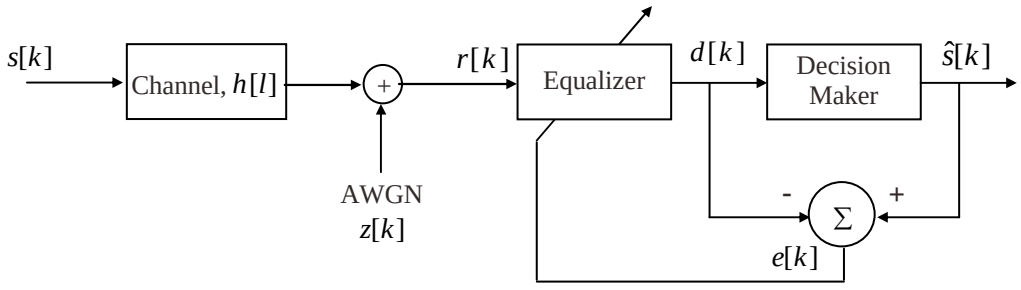


Figure 2.4: Block diagram of a communication system with an equalizer at the receiver.

If the equalizer taps are denoted by $\{f[k]\}_{l=0}^{Leq}$, the output of the equalizer is

$$\begin{aligned} d[k] &= \sum_{l=0}^{Leq} f[l]r[k-l] \\ &= \sum_{l=0}^{L+Leq+1} g[l]s[k-l] + \tilde{z}[k] \end{aligned} \tag{2.8}$$

where $(Leq+1)$ is the length of the equalizer,[4] and $g[k]$ and $\tilde{z}[k]$ are given by (2.9) and (2.10), respectively, as

$$g[k] = \sum_{l=0}^{Leq} f[l]h[k-l] \tag{2.9}$$

$$\tilde{z}[k] = \sum_{l=0}^{Leq} f[l]z[k-l] \tag{2.10}$$

Assuming that $\tilde{z}[k]$ is negligible, in order to make $d[k] = s[k]$ in (2.8), the value of $g[k]$ should be equal to

$$g[k] = \delta[k] \tag{2.11}$$

The relationship in (2.11) can be translated in the frequency domain as

$$F(f)H^{*}(-f) = 1 \tag{2.12}$$

where $F(f)$ and $H(f)$ are the Discrete Fourier transforms of $f[k]$ and $h[k]$, respectively.

Equation (2.12) indicates that an equalizer actually works as an inverse filter of the channel. This is the criteria for a zero-forcing equalizer. If the channel is frequency selective, the equalizer boosts the low points and suppresses the high points in the channel spectrum in order to get a flat frequency response and linear phase response at the output of the equalizer. In doing so, it severely enhances noise at the frequencies where the channel spectrum exhibits high attenuation.

The prediction error of the equalizer is the error between the output of the equalizer before decision-making and the desired output after decision-making. It is given by

$$e[k] = \hat{s}[k] - d[k] \tag{2.13}$$

---

[4] Usually, the equalizer length is greater than that of the channel.

A figure of merit of an equalizer is the expected value of the squared prediction error, and is denoted by $E\left[\left|e[k]\right|^2\right]$. If $e[k]$ is approximately Gaussian, then the smaller the variance, the smaller the chance of the prediction error being larger than the threshold value that could cause a decision error. For sufficiently low SNR, an equalizer that minimizes Mean Squared Error (MSE) usually results in a bit error rate (BER) arbitrarily close to that of a minimum bit error rate equalizer.

Depending on how the output of an adaptive equalizer is used for feedback, equalization techniques can be classified into two general groups, linear and non-linear. In general, the output of the equalizers, $d(t)$, is processed by the hard limiter at the receiver. The hard limiter does a non-linear operation in order to obtain the value of $\hat{s}(t)$. If $\hat{s}(t)$ is not used to adapt the equalizer, the equalization is linear. Otherwise, if $\hat{s}(t)$ is used in the decision-making of the subsequent outputs of the equalizer, the equalization is nonlinear [5].

2.3.1 Linear Equalizer

The most common equalizer structure is a Linear Transversal Equalizer (LTE). It is made of a symbol-spaced (T_S) tapped-delay-line filter and a hard limiter as shown in Figure 2.5. The transfer function of a linear transversal equalizer can be written as a function of delay operators, or z^{-1}. This filter may have many zeros, but poles only at $z=0$, so it is called a Finite Impulse Response (FIR) filter [1]. In such an equalizer, the current and past values of the received signal are linearly weighted by the filter coefficients and summed to produce the output.

The output of the transversal filter before the hard limiter is

$$d_k = \sum_{n=-N_1}^{N_2} c_n^* r_{k-n} \tag{2.14}$$

where c_k's are the complex coefficients or tap-gains of the filter, d_k is the output at time

k, y_i is the input at time $t_0 + iT$, t_0 is the equalizer starting time, and $N = N_1 + N_2 + 1$ is the number of taps. The values N_1 and N_2 are the number of taps used in the forward and reverse portion of the equalizer, respectively. The minimum mean squared error that an LTE can achieve is [6]

$$E\left[\left|e[k]\right|^2\right]_{\min} = \frac{T}{2\pi}\int_{-\pi/T}^{\pi/T}\frac{N_0}{\left|F(e^{j\omega k})\right|^2 + N_0}d\omega \tag{2.15}$$

in which $F(e^{j\omega k})$ is the frequency response of a channel, and N_0 is the noise spectral density.

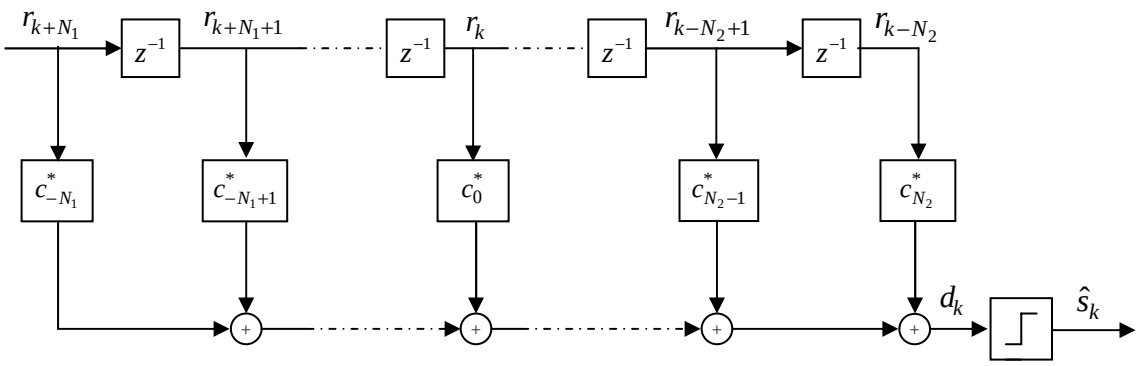


Figure 2.5: Linear transversal equalizer.

The linear equalizer can also be implemented as a lattice filter. Two main advantages of the lattice equalizer are numerical stability and faster convergence [5], [6]. Another feature of this type of equalizer is that the effective length of the lattice equalizer can be dynamically adjusted depending upon the time dispersiveness of the channel. However, all these upsides come at the cost of increased complexity.

2.3.2 Non-Linear Equalizer

Linear equalizers are not very efficient on channels with deep spectral nulls in the pass-band. This is because linear equalizers place high gain near the spectral null in order to

compensate for distortion, and in the process increase the noise present at those frequencies. Non-linear equalizers don't have a noise enhancement problem. The two most common methods of non-linear equalization are Decision Feedback Equalization (DFE) and Maximum Likelihood Sequence Estimation (MLSE).

2.3.2.1 Decision Feedback Equalizer (DFE)

A decision Feedback Equalizer consists of a Feed-Forward Filter (FFF), a Feed-Back Filter (FBF) and a hard limiter. The coefficients of the FBF can be adjusted by the decision on the output of the hard limiter to cancel the ISI on the current symbol from past-detected symbols. Figure 2.6 shows the structure of the DFE. If the equalizer has $N_1 + N_2 + 1$ taps in the feed forward filter and N_3 taps in the feedback filter, its output before the hard limiter is

$$d_k = \sum_{n=-N_1}^{N_2} c_n^* r_{k-n} + \sum_{i=1}^{N_3} F_i^* \hat{s}_{k-i} \tag{2.16}$$

where y_n is the input, c_n is the tap gain of the feed-forward filter, F_i^* is the tap gain of the feedback filter and $\hat{s}_i (i < k)$ is the previous decisions made on the detected signal. $\hat{s}_k$ is hard limited from d_k once it is obtained using equation (2.16). Then $\hat{s}_k$ and previous decisions $\hat{s}_{k-1}, \hat{s}_{k-2}, \ldots$ are fed back to get d_{k+1}, using (2.16). The minimum mean squared error of DFE is [6]

$$E\left[\left|e[k]\right|^2\right]_{\min} = \exp\left(\frac{T}{2\pi}\int_{-\pi/T}^{\pi/T} \frac{N_0}{\left|F(e^{j\omega k})\right|^2 + N_0} d\omega\right) \tag{2.17}$$

It can be shown that the minimum MSE for DFE in (2.17) is always smaller than an LTE in (2.15), unless $\left|F(e^{j\omega k})\right|$ is a constant [5]. However, like linear equalizers, DFE doesn't perform well in the case of non-minimum phase channels, due to the increased probability of error propagation [10]. Even in the absence of error propagation, DFE is

suboptimum, since it can't exploit the signal energy embedded in the ISI terms. Newer versions of DFE based on artificial neural networks exhibit slight performance improvement over conventional DFE's [11]. However, these types of DFE's are computationally intensive.

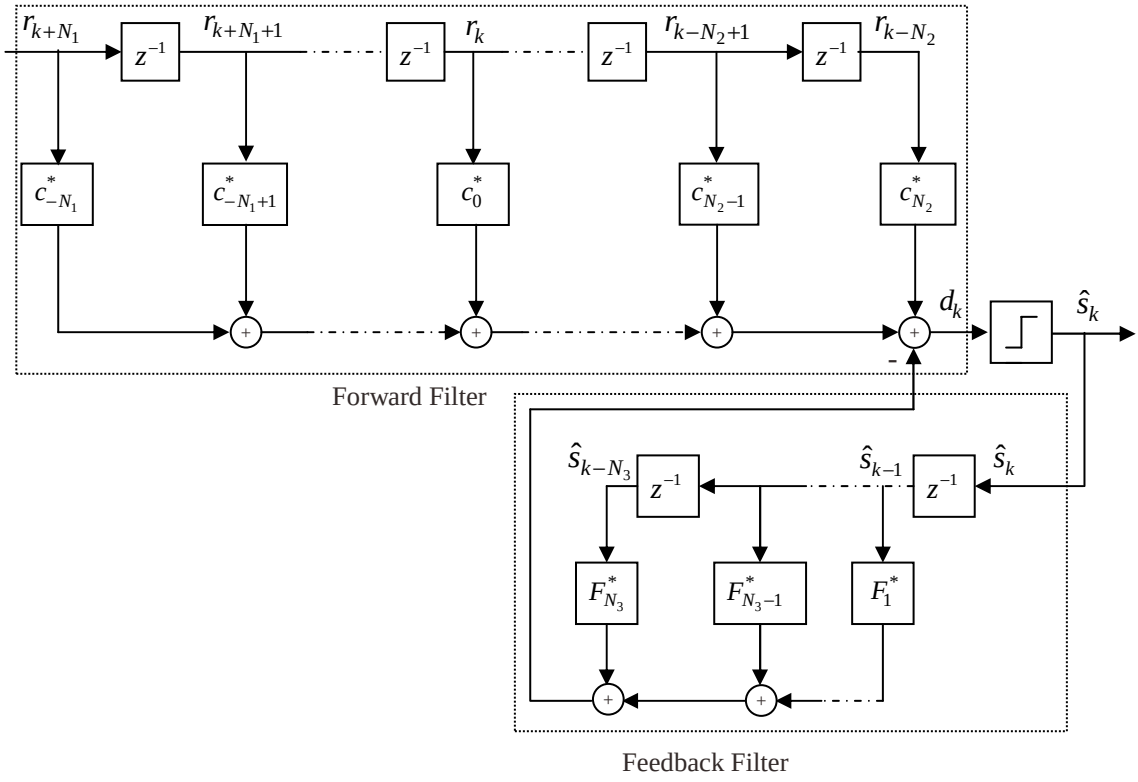


Figure 2.6: Decision feedback equalizer (DFE).

2.3.2.2 Maximum Likelihood Sequence Estimator (MLSE)

Although the DFE performs better than a linear equalizer, it is not the optimum equalizer in a multipath environment due to its symbol-by-symbol detection scheme. In a communication channel with ISI, the optimum detector is a Maximum Likelihood Sequence Estimator (MLSE). Using the known channel impulse response, MLSE tests all possible data sequences drawn from the alphabet and picks the one with the maximum

probability as the output. Usually an MLSE is computationally inefficient, especially if the memory of the channel is very large. Using the MLSE as an equalizer was first proposed by Forney [1] in which a basic MLSE structure was implemented with Viterbi Algorithm (VA) [12]. MLSE for channels with infinite memory has also been addressed [13].

Now we will develop a sequence detector that makes a decision on transmitted signal based on the observation of received signal. For the received signal vertor, $\mathbf{r}$, transmitted sequence, $\mathbf{s}$, and the channel $\mathbf{h}$ (assumed known at this stage), the decision criterion is based on selecting the sequence, $\hat{\mathbf{s}}$, corresponding to the maximum of probability given by (2.18)

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}\in M^N} \{ p(\mathbf{s} \mid \mathbf{r}, \mathbf{h}) \} \tag{2.18}$$

wherein M is the size of the alphabet, $M = |\mathcal{A}|$, N is the frame length, and M^N represents all possible sequences obtained from the alphabet. This decision criterion is called *maximum a posteriori probability* (MAP) criteria. Using Bayes' rule, the posteriori probability can be written as

$$p(\mathbf{s} \mid \mathbf{r}, \mathbf{h}) = \frac{p(\mathbf{r} \mid \mathbf{s}, \mathbf{h})\, p(\mathbf{s})}{p(\mathbf{r}, \mathbf{h})} \tag{2.19}$$

The conditional probability density function (pdf), $p(\mathbf{r} \mid \mathbf{s}, \mathbf{h})$, is called the likelihood function. The decision criterion based on the maximum of $p(\mathbf{r} \mid \mathbf{s}, \mathbf{h})$ over all possible sequences, $\mathbf{s} \in M^N$, is called the *maximum likelihood* (ML) criterion. If all of the sequences are equiprobable, the MAP criterion reduces to ML criterion [7]. The conditional pdf, $p(\mathbf{r} \mid \mathbf{s}, \mathbf{h})$, given by (2.20), is a multi-dimensional Gaussian distribution with the assumption that noise is white and Gaussian.

$$p(\mathbf{r} \mid \mathbf{s}, \mathbf{h}) = \left(\frac{1}{\sqrt{2\pi\sigma_z}} \right)^N \exp(-\left\| \mathbf{r} - \boldsymbol{\mathcal{S}}_N \mathbf{h} \right\|^2 / \sigma_z^2) \tag{2.20}$$

where $\|\cdot\|$ denotes the norm of a vector, and the data matrix, $\boldsymbol{\mathcal{S}}_N$, is given by (2.5).

By taking the logarithm of (2.20) and neglecting the terms that are independent of $\mathbf{r}$, we

find that the ML sequence estimator selects the sequence, $\hat{\mathbf{s}}$, that minimizes the *Euclidean distance metric*

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in M^N} \left\| \mathbf{r} - \boldsymbol{\mathcal{S}}_N \mathbf{h} \right\|^2 \tag{2.21}$$

It is apparent from (2.21) that we must compute the distance for every possible sequence, which is known as a tree-search problem. In tree-search problems, as the frame length increases, the number of hypothesized sequences increases by a factor of M after each symbol. The optimal solution to any tree-search problem is the exhaustive search, which searches through all possible sequences, $\mathbf{s}\in M^N$. However, it is too complex to be implemented in practical systems. Figure 2.7 shows the tree diagram for MLSE for the channel shown in Figure 2.3. We considered BPSK, i.e. $\mathcal{A} = \{1, -1\}$, and assumed the channel is initially in all-one state. At each time epoch, k, the channel states are described by $\sigma[k]$. According to the branching rule, the sequence takes the upper branch if the next transmitted symbol is 1, and the lower if it is -1. Thus we take a particular path through the tree that is determined by the input sequence.

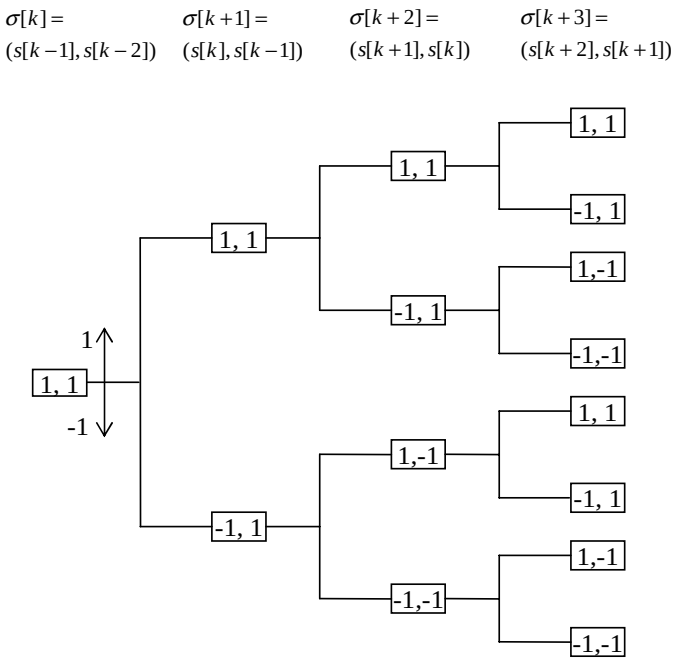


Figure 2.7: Tree diagram for BPSK with L = 2.

2.4 Trellis Structure

A close look at Figure 2.7 reveals that the structure repeats itself after the second stage. This behavior is consistent with the fact that we considered a 2nd-order channel. Due to the finite memory property, the exponentially growing sequence tree repeats itself periodically. This allows the use of a *trellis diagram* in place of a tree. Figure 2.8 shows the equivalent trellis diagram for the tree diagram shown in Figure 2.7 [14].

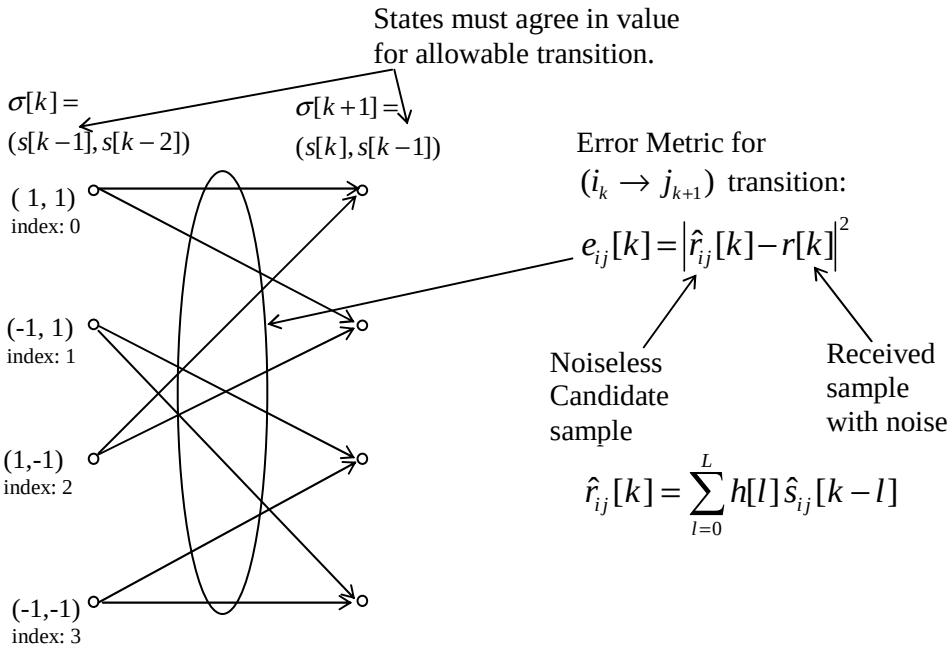


Figure 2.8: Trellis structure and state transitions for BPSK with $L = 2$.

A trellis is a compact way of visualizing all possible sequence of states, where each state at epoch k, $\sigma[k]$, describes all possible values taken on by the L most recently transmitted symbols, with L being the memory of the channel. So each state accounts for the entire channel memory. In a trellis diagram, each node corresponds to a distinct state at a given time, while each line represents a transition to a new state at the next instant of time. A state can be either named by its symbol value, $\sigma[k] = (s[k-1], s[k-2]) = (-1, 1)$, or by its index $\sigma[k] = i = 1$. States are connected only if their values match. For a

transition between state $\sigma[k]$ and $\sigma[k+1]$, the elements $(s[k-1],\ldots,s[k-L+2])$ of state $\sigma[k]$, should be equal to the elements $(s[k-1],\ldots,s[k-L+2])$ of state $\sigma[k+1]$. This should be evident from Figure 2.8. If the alphabet, $\mathcal{A}$, has M set of signal points, then each element in the state vector can take on one of the M values. Therefore the trellis has M^L states, and there are M transitions into and out from each state.

2.5 Viterbi Algorithm

The Viterbi Algorithm (VA) works on the principle of a trellis. It is a recursive optimal approach of estimating the state sequence of a discrete-time finite-state Markov process in AWGN noise [12]. Instead of operating on a symbol-by-symbol basis that yields the least squared error, the VA works with the sequence of states. The Viterbi Algorithm exploits the trellis structure efficiently to reduce the number of computations. It recognizes that the conditional probability of the process in a particular state, given all past states, is equal to the conditional probability of the process in that state, given only the last state, or in equation form

$$p(\sigma[k+1]\,|\,\sigma[0],\sigma[1],\ldots,\sigma[k]) = p(\sigma[k+1]\,|\,\sigma[k]) \tag{2.22}$$

It also takes advantage of the fact that the conditional probability of a noisy received signal, given the entire state sequence is equal to the conditional probability of that received signal, given only the latest state transition. At epoch k

$$p(r[k]\,|\,\boldsymbol{\sigma}) = p(r[k]\,|\,\sigma[k]\rightarrow\sigma[k+1]) \tag{2.23}$$

where $\boldsymbol{\sigma}$ is the vector of entire state sequence and $r[k]$ is the received signal with noise.

The VA can be summarized as follows:

- Most of the problems that can be framed in terms of sequence detection can be solved with the Viterbi Algorithm.
- The most likely sequence of input symbols is the one that results in the lowest metric between the corresponding transmitted signal and the actual received signal.

- For each state, it is possible to calculate the most likely previous state by computing the metrics corresponding to all possible previous states.
- It is possible to determine the most likely sequence of states by recursively computing the most likely previous state.
- Once the state sequence is obtained, it can be translated into a symbol sequence.
- The Viterbi Algorithm performs optimally for full-state MLSE only when the channel is known.

We will now introduce notations that will be useful to delineate the algorithm. Let us denote a transition from the i-th state at time index k to the j-th state at the next time index $(k+1)$ by $(i_k \to j_{k+1})$, and $\hat{r}_{ij}[k]$ be the noiseless candidate signal corresponding to the $(i_k \to j_{k+1})$ transition at epoch k and is given by

$$\hat{r}_{ij}[k] = \sum_{l=0}^{L} h[l]s[k-l] \tag{2.24}$$

where

$$i = \sigma[k] = (s[k-1], s[k-2], \ldots, s[k-L]) \tag{2.25}$$

$$j = \sigma[k+1] = s([k], s[k-1], \ldots, s[k-L+1]) \tag{2.26}$$

The $(i_k \to j_{k+1})$ transition (or "branch") cost at stage k is the error between the received signal and the noiseless candidate signal given as

$$e_{ij}[k] = \left|\hat{r}_{ij}[k] - r[k]\right|^2 \tag{2.27}$$

where $r[k]$ is the received signal with noise. The cumulative cost metric for $(i_k \to j_{k+1})$ transitions is defined by the following equation

$$\mathcal{C}_j[k+1] = \min_{i \in \mathcal{I}_j}\{\mathcal{C}_i[k] + e_{ij}[k]\} \tag{2.28}$$

in which $\mathcal{I}_j$ is the set of all allowable transitions into the j-th state from the i-th states, $\mathcal{C}_j[k+1]$ is the cumulative cost metric at state j of the next stage $(k+1)$, and $\mathcal{C}_i[k]$ is the cumulative cost metric at state i of the present stage k.

If the data sequence is padded with known L header symbols, $(s[-1], s[-2], \ldots, s[-L])$, then the cumulative cost at epoch $k = 0$, $\mathcal{C}_i[0]$, can be initialized as follows

$$\begin{aligned} \mathcal{C}_i[0] &= 0, \quad if \quad \sigma(0) = (s[-1], s[-2], \ldots, s[-L]) \\ &= \infty, \quad otherwise \end{aligned} \tag{2.29}$$

If no header symbols are available, the cumulative cost metrics of all the states are initialized to zero.

$$\mathcal{C}_i[0] = 0, \qquad \forall i = 0,1,\ldots,(N_S - 1) \tag{2.30}$$

where $N_S = M^L$ is the total number of states in the trellis.

The survivor at the j-th state of the $(k+1)$-th stage is defined by

$$I_j[k+1] = state\,index/value(\arg\min_{i \in \mathcal{T}_j}\{\mathcal{C}_i[k] + e_{ij}[k]\}) \tag{2.31}$$

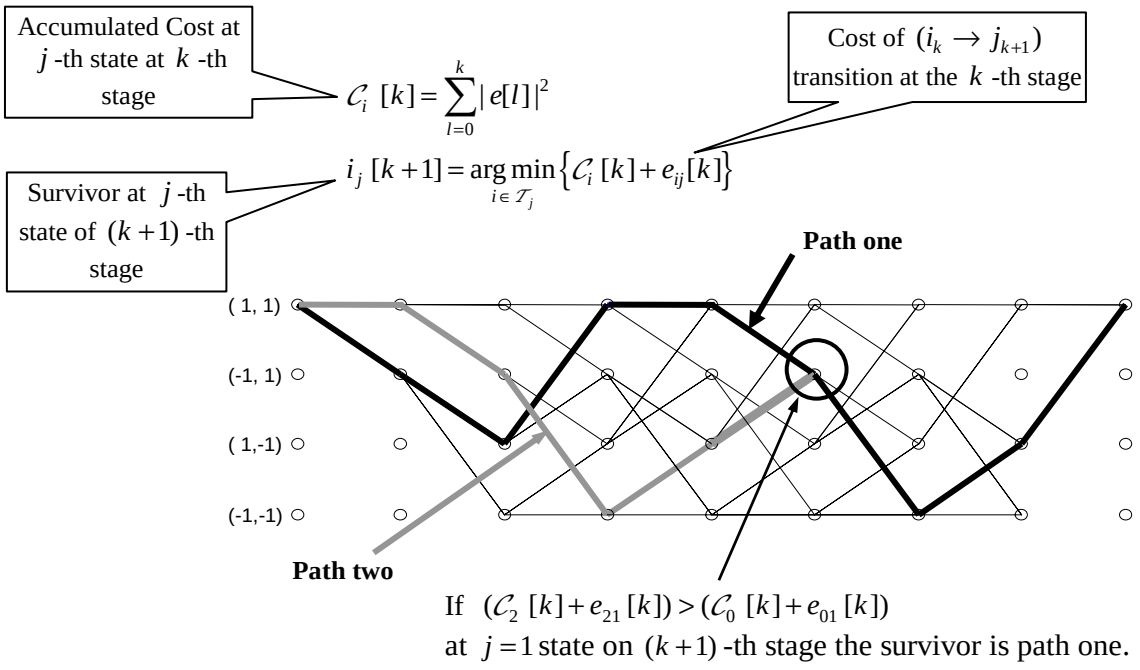


Figure 2.9: The Viterbi Algorithm keeps path with minimum cost metric at each state.

This concept is depicted in Figure 2.9 [14]. If at state $j = 1$ of stage $k+1$, the cumulative cost of path two is greater than that of path one, then the Viterbi Algorithm keeps path one as the survivor of that state.

At any point in the algorithm, the least cost state sequence can be obtained from a list of survivors, $I_j[k]$, which is called *trace back*. Before doing trace back we have to find the state from where the process is started, which we call trace back state. This can be done in two ways. First, by terminating the trellis with known tail symbols. In this case, the frame contains L known tail symbols of $(s[N], s[N+1], \ldots, s[N+L-1])$, where N is the length of the information frame. The least cost path should end with the state $\sigma[N+L] = (s[N+L-1], \ldots, s[N])$. So the trace back starts at state $\sigma[N+L]$, which is now completely defined by the tail symbols. Terminating the trellis with L known tail symbols is called *trellis termination*. However, if the tail symbols are not available, then the trellis is not terminated. The trace back then starts from the state with the minimum cumulative cost metric, $\sigma[N] = \arg\min\{\mathcal{C}_i[N]\},\ \forall i = 0,1,\ldots,(N_S - 1)$. We call this process *Algorithm termination*. In Chapter 4, we will show that if we consider *transients*, we don't need tail symbols to terminate the trellis. The trace back state is defined by

$$\begin{aligned} j_{tr} &= \sigma[N+L] = (s[N+L-1], \ldots, s[N]), && \textit{if trellis terminated} \\ j_{tr} &= \sigma[N] = \arg\min\{\mathcal{C}_i[N]\},\ \forall i = 0,1,\ldots,(N_S-1), && \textit{if algorithm terminated} \end{aligned} \tag{2.32}$$

Once the trace back state is defined, we can construct the least cost state sequence by backward-recursion using (2.33)

$$\begin{aligned} j_{tr} &= I_{j_{tr}}[n] \\ \sigma[n-1] &= j_{tr} \end{aligned} \tag{2.33}$$

where $n = (N+L), (N+L-1), \ldots, 1$, if the trellis is terminated and $n = N, (N-1), \ldots, 1$, if the algorithm is terminated.

Once the least cost state sequence is known, the least cost MLSE symbol sequence is easily obtained from the state definition given in (2.25). In practice, a decision on symbol $s[k-D]$ is made after the algorithm steps at time k, by tracing back on the best survivor at time k. The parameter D is called *trace back depth*. Usually D is chosen such that $7L \geq D \geq 5L$, with little degradation in performance, since the probability of all surviving paths at time k converging together in the trellis at a depth $5L$ is high.

Figure 2.10 explains how trace back is done in the Viterbi Algorithm. The number associated with each state represents the survivor index leading to that state according to

(2.31). In this case we have padded the trellis with all ones to have it terminated at $\sigma[N+2]=(1,1)$ state, so the trace back starts at that state (index 0) according to (2.32). At $\sigma[N+2]=(1,1)$, the stored survivor index is 0, so the algorithm looks back to state $(1,1)$ of the previous stage, i.e. $\sigma[N+1]=(1,1)$. This way the algorithm looks back at each stage for the survivor index and decodes the corresponding symbols. The trace back path is shown by thick lines. For the sake of simplicity, only the best surviving path is shown.

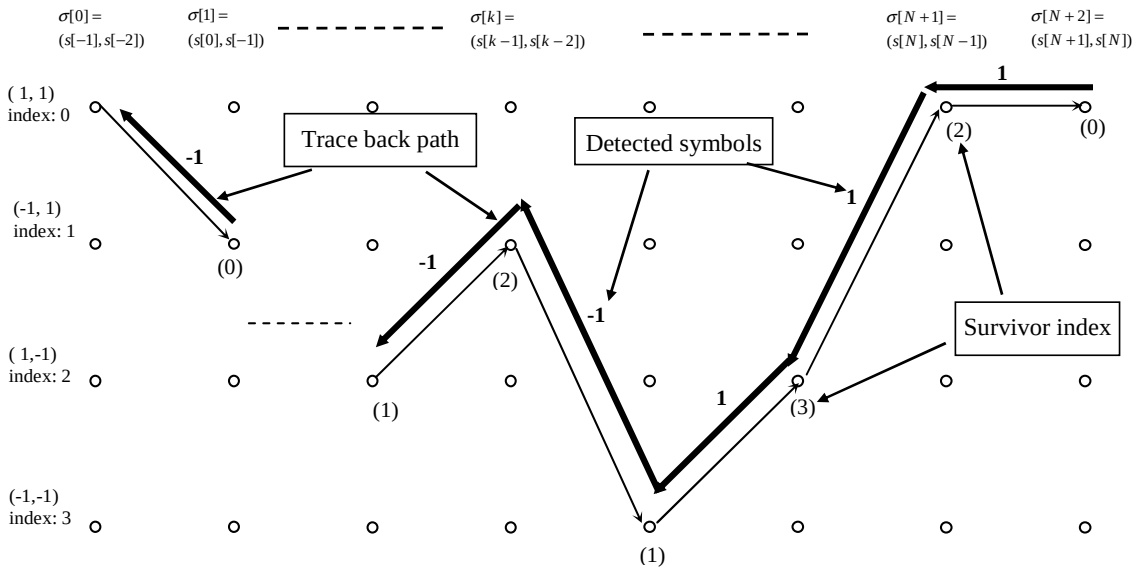


Figure 2.10: Trace back in the Viterbi Algorithm.

A detailed example of Viterbi Equalization (VEQ) is illustrated in Figure 2.11. We have considered a channel with $L=2$ and $\mathcal{A}=\{1,-1,\ j,-j\}$. The frame length is $N=9$, with no header or tail symbols. The cumulative metrics are initialized according to (2.30). Since no tail symbols are available, algorithm termination is used for trace back as per (2.32). As a reference noisy received frame and corresponding decoded symbols are also shown.

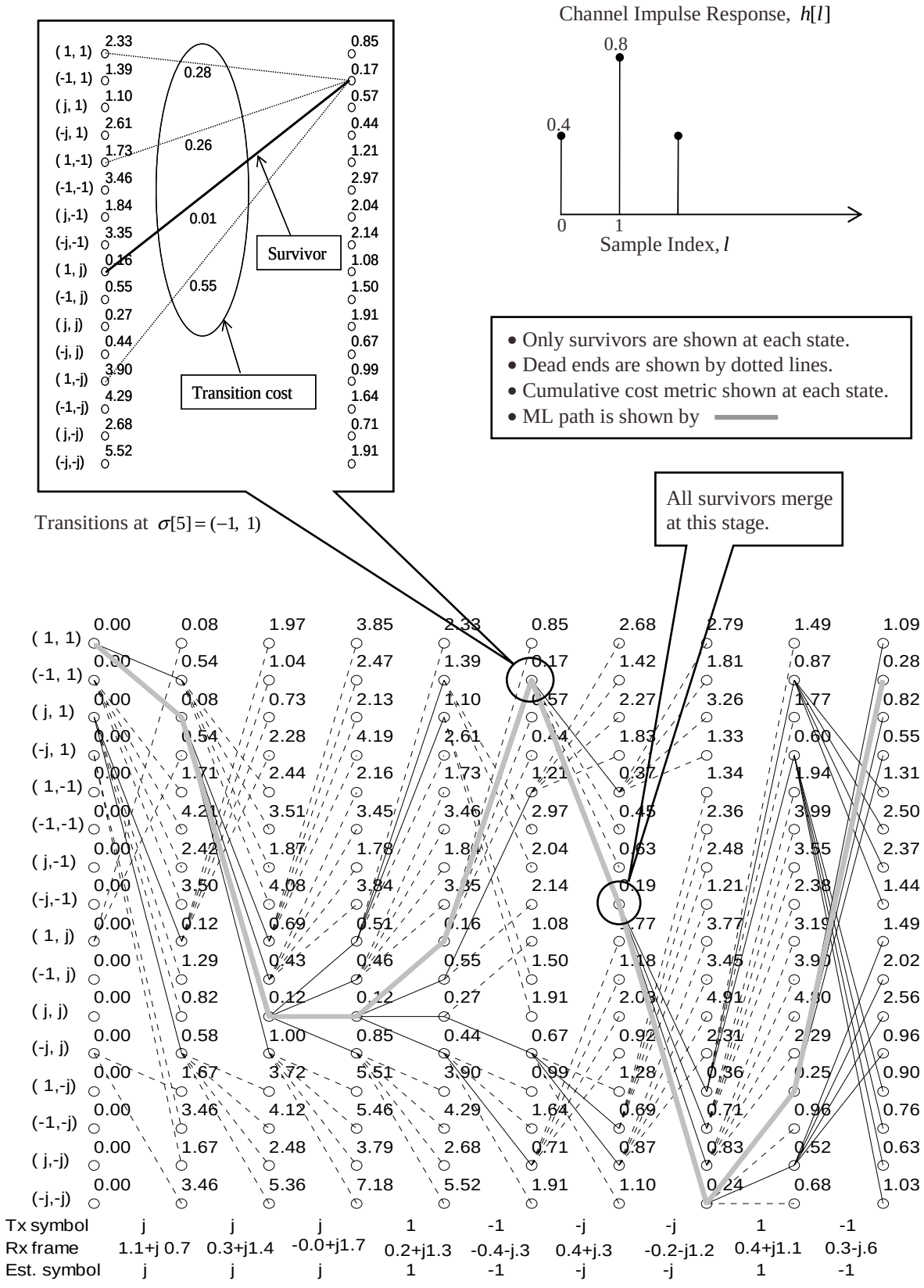


Figure 2.11: An example of VA for QPSK with $L = 2$.

2.5.1 Complexity of the Viterbi Algorithm

The complexity of the Viterbi Algorithm can be expressed in terms of the number of states. However, the number of transitions per each stage is a better yardstick, since different trellises of the same number of states can have a different number of transitions. The trellis size can be measured by the number of states and by the number of transitions per state. If $\mathcal{A}$ is the alphabet and L is the memory of the channel, then the number of states is $|\mathcal{A}|^L$.

2.6 State-Reduction Techniques

VEQ has reasonable complexity for small alphabets and small channel-length. However if the channel length or the alphabet is large, it becomes computationally very complex. This complexity limits the VA's application to very high data rates. However, VEQ's performance in equalizing channels with deep nulls without noise emphasis makes it worth exploring, with some modifications that can reduce the number of states in the trellis. The idea of a reduced state was developed soon after the VA was proposed to implement MLSE [3], [15]-[26]. Different authors have proposed a number of ways to reduce the complexity in the VA trellis. One way to reduce the number of states is by preprocessing the channel to "shorten" the length of the channel. Most of the earlier work concentrated on this approach [15]-[18]. Another method is the direct truncation of the channel impulse response [19]. In this case, the residual ISI terms are not taken care of by the detector. It degrades the performance if the residual ISI terms have significant energy. In [20], Lee and Hill introduced the idea of decision feedback to truncate the channel impulse response. Here, the VA is employed based on partial channel information, while the tentative decisions derived from the temporary best survivor are used to cancel the residual ISI terms.

Recently researchers have promoted the technique to imbed the decision feedback mechanism within the search algorithm itself (e.g. VA), instead of using some external

tentative decisions. In fact, this way of using the search algorithm is the fundamental principle of Per-Survivor Processing (PSP), which will be discussed in detail in Chapter 3. Authors in [21], [22] have incorporated this notion in state reduction techniques. In [21], a reduced-state algorithm, Delayed Decision Feedback Sequence Estimation (DDFSE), was proposed especially for the case of infinite ISI channels. Another algorithm similar to DDFSE, Reduced State Sequence Estimation (RSSE), deals with finite ISI channels and large input alphabets. It incorporates feedback information as well as set partitioning [23], [24] in the decoder. Recently, another reduction technique was proposed, in which pairs of complement states form a group to build the trellis [25]. This trellis has a regular radix-2 structure, which can be exploited for implementation.

We will now present a detailed description of DDFSE.

2.6.1 Delayed Decision Feedback Sequence Estimation (DDFSE)

Delayed Decision Feedback Sequence Estimation (DDFSE) is a reduced-state detection technique that trades off complexity for performance over Inter-Symbol Interference (ISI) channels [21]. In MLSE, each state represents all possible values taken on by L most recently transmitted symbols, with L being the memory of the channel. Unlike Full-State MLSE, in DDFSE, each state describes μ most recently transmitted symbols given by

$$\sigma[k] = (s[k-1], s[k-2], \ldots, s[k-\mu]) \tag{2.34}$$

where $0 \le \mu \le L$. The "reduced memory size" parameter, μ, defines the complexity/performance trade-off. If $\mu = 0$, DDFSE reduces to Decision Feedback Estimation (DFE). On the other hand, with μ approaching L, it performs Full-State MLSE.

According to (2.34), each state in DDFSE trellis provides only partial information about the channel, so it is called *reduced state*. We need more information about the states in order to exploit the channel memory. The required *residual information* about the channel state is provided by the survivor history associated with each state.

Effectively, information symbols from the survivors leading to each state constitute *partial state*. Since this is similar to Decision Feedback Equalization (DFE) it is called *feedback state* as well. Reduced state along with feedback state approximates the complete channel state.

The error metric for the $(i_k \rightarrow j_{k+1})$ transition at epoch k is calculated by

$$e_{ij}[k] = \left| \hat{r}_{ij}[k] - r[k] \right|^2 \tag{2.35}$$

where

$$\hat{r}_{ij}[k] = \sum_{l=0}^{\mu} h[l]\, s[k-l] + \sum_{l=\mu+1}^{L} h[l]\, \hat{s}[k-l] \tag{2.36}$$

in which state i and j are defined by (2.25) and (2.26) respectively, $\{s[l],\ l = k-\mu, k-\mu+1, \ldots, k\}$ denotes the information sequence associated to state transition $(i_k \rightarrow j_{k+1})$, defined by the reduced state trellis; and $\{\hat{s}[l],\ l = k-L, k-L+1, \ldots, k-\mu-1\}$ denotes the information sequence found in the partial state, $\rho[k]$, defined by

$$\rho[k] = (\hat{s}[k-\mu-1], \hat{s}[k-\mu-2], \ldots, \hat{s}[k-L]) \tag{2.37}$$

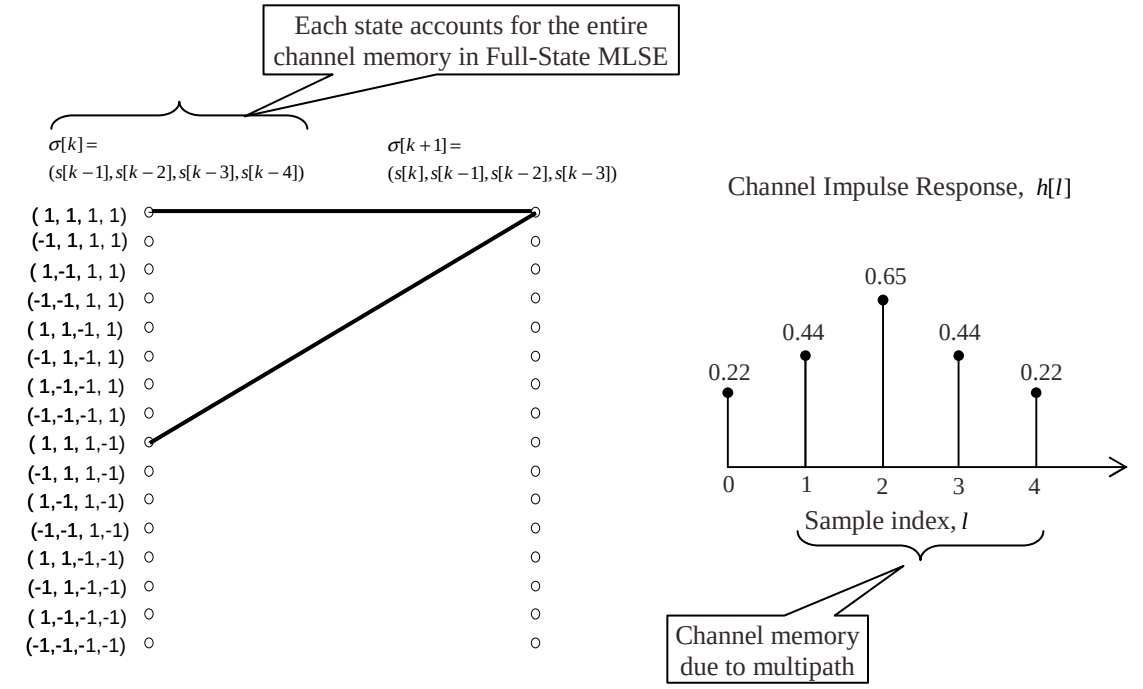
Partial states are formed by symbols extracted from the survivor leading to state i at epoch k. Note that the first summation in (2.36) is the contribution of the reduced state portion of the channel, while the second summation is required to account for the ISI terms not taken care of by the reduced state trellis.

The performance of DDFSE has been reported in [21]. The idea to incorporate the residual state information within the trellis search algorithm itself is strictly dependent on survivors. The more reliable the survivor, the more reliable the residual state estimation associated with it. So it is intuitive to say that, if the channel impulse response has most of the energy in the partial states, the performance of DDFSE depends heavily on the correct selection of survivors. As a result, DDFSE doesn't perform well for non-minimum phase channels. However, if the partial state can be chosen over the impulse response with small energy, DDFSE can closely approximate full-state MLSE.

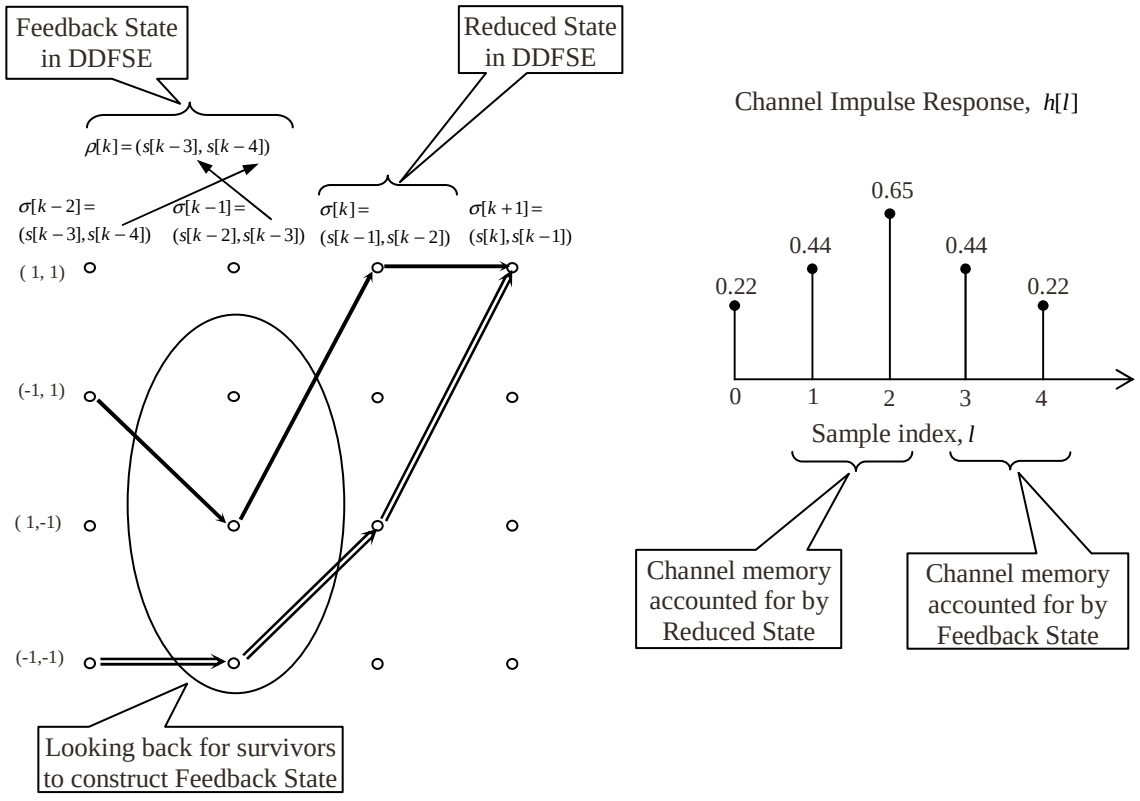
An example trellis for DDFSE is explained in Figure 2.12. The trellis is drawn for BPSK with a channel memory, $L = 4$ and DDFSE parameter, $\mu = 2$. For MLSE, each state in the trellis completely defines the channel state, so there are $|\mathcal{A}|^L = 16$ states. However, in DDFSE, each state, $\sigma[k]$, partially defines the channel state. For $\mu = 2$, each state only accounts for the 1st and 2nd ISI components. The number of states in the reduced-state trellis is $|\mathcal{A}|^\mu = 4$. DDFSE accounts for the 3rd and 4th ISI component by looking back through the trellis for survivors. Let's consider the next state, $\sigma[k+1] = (1, 1)$ in Figure 2.12(b). There are two present states from where a transition can occur to this state. If the present state is $\sigma[k] = (1, 1)$, then the algorithm looks back along its own survivor path to form its partial state. The survivor path for this state is shown by single lines. Similarly, for present state, $\sigma[k] = (1, -1)$, the corresponding survivor path is shown by double lines. For complete partial state the algorithm has to look through two previous stages. This is consistent with the fact that two ISI components were ignored in the reduced-state trellis $(L - \mu = 2)$.

2.6.1.1 Complexity of DDFSE

The complexity of DDFSE is determined by the number of trellis states, which is given by $|\mathcal{A}|^\mu$. This is a substantial reduction from $|\mathcal{A}|^L$ given by Viterbi Algorithm. When $\mu = 0$, the complexity reduces to that of DFE. DDFSE has as many transitions (determined by alphabet) for each state, as in the Viterbi Algorithm, so it doesn't save any computation for a given state. However, since the number of states can be much lower in DDFSE, it can achieve a great computational savings.



(a)



(b)

Figure 2.12: Exploiting channel memory with BPSK and $L = 4$ (a) Full-State MLSE (b) DDFSE, μ = 2.

2.7 Conclusion

In this chapter the basic system model has been presented. Equalizers are required to compensate for the ISI caused by frequency selective channels. Linear equalizers are simple and numerically stable but they suffer from noise enhancement problem. Non-linear equalizers like DFE outperform linear equalizers. However, DFE is prone to error propagation, especially for non-minimum phase channels. MLSE is the optimum detector in the ISI environment. The Viterbi Algorithm implements MLSE for practical systems exploiting the trellis structure. However, the complexity of the VA grows exponentially with the channel order. DDFSE provides a trade-off between complexity and performance by combining the strength of trellis-based sequence estimation and decision feedback equalization. In the next chapter, adaptive MLSE algorithms will be presented which use both VA and DDFSE.

Chapter 3: Adaptive MLSE

The algorithms discussed in the previous chapter assume that the receiver has perfect knowledge of the channel. However, in a practical system, the channel is unknown to the receiver and must be estimated. Most of the channel estimation algorithms can be categorized into two classes; *non-blind* and *blind*, depending on whether a *training sequence* is used or not, respectively. The focus of this chapter is blind estimation of the channel parameters. In the first part of this chapter we discuss blind algorithms that estimate the channel for a Viterbi Equalizer (VEQ). Even though these algorithms can easily be extended to estimate any unknown parameter in the system that affects transition metrics in the VA (e.g. reduced-state MLSE), our focus is on the ISI problem where the channel impulse response is unknown. Later in the chapter, the concept of Per-Survivor Processing (PSP) and Generalized Per-Survivor Processing (GPSP) will be introduced. A detailed complexity analysis of various algorithms is also presented.

3.1 Non-Blind Adaptive Algorithms

In a non-blind adaptive algorithm, a training sequence, $s[k]$, which is known to both the transmitter and receiver, is sent from the transmitter to the receiver. The channel estimator in the receiver uses the information of the training sequence to compute the optimal channel vector. After the training period, information symbols are sent, and the receiver uses the previously estimated channel to process the received signal. Non-blind algorithms are also known as data-aided or decision-directed channel estimation. Least

Mean Square (LMS) and Recursive Least Square (RLS) are the two most popular non-blind adaptive algorithms. Since during the training period data can't be sent over the channel, non-blind algorithms are spectrally inefficient. Most of the non-blind algorithms attempt to minimize the mean squared error between the training sequence, $s[k]$, and the output, $\hat{s}[k]$, which is given by

$$J = E\left[\left|e[k]\right|^2\right] = E\left[\left|r[k]-\hat{r}[k]\right|^2\right] \quad (3.1)$$

Figure 3.1 shows a simple data-aided non-blind channel estimator.

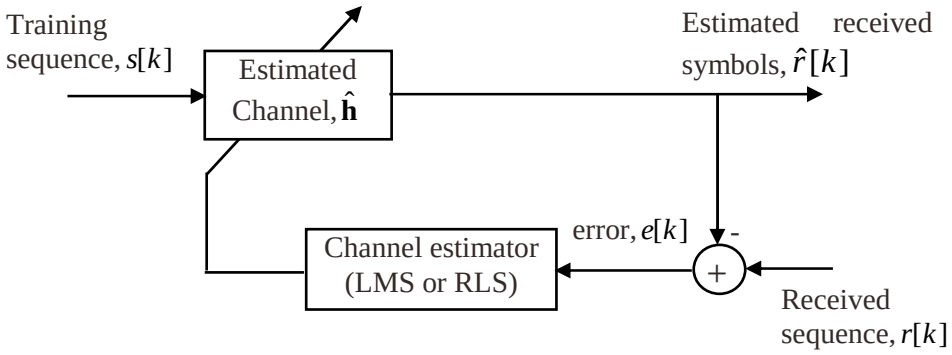


Figure 3.1: Block diagram of non-blind channel estimator.

3.2 Blind Adaptive Algorithms

Blind adaptive algorithms estimate the channel without a training sequence. To overcome the lack of a training sequence, blind algorithms exploit some known properties of the received signal like the constant modulus property, cyclostationary property, or the finite alphabet property. Optimal blind channel estimation is an exhaustive joint Maximum Likelihood (ML) search algorithm that is prohibitively expensive. It is well known that the joint ML estimate of data sequence and channel estimate is obtained by minimizing the metric according to

$$\Lambda\left(\hat{\boldsymbol{\mathcal{S}}}_k, \hat{\mathbf{h}}\right) = \min\left\{\min\left\{\left\|\mathbf{r}_k - \boldsymbol{\mathcal{S}}_k \mathbf{h}\right\|^2 : \mathbf{h} \in \mathbb{C}^{(L+1)}\right\} : \boldsymbol{\mathcal{S}}_k \in \mathbb{T}_{\mathcal{A}}\right\} \quad (3.2)$$

where $\mathbf{r}_k$ is the received vector, and $\hat{\boldsymbol{\mathcal{S}}}_k$ is one of the many minimizing solutions from the set, $\mathbb{T}_{\mathcal{A}}$, of all possible toeplitz data matrices[1] generated by the alphabet, $\mathcal{A}$. This minimization operation can be accomplished first by a minimization over $\mathbf{h}$ for each possible data sequence, followed by a minimization of the result over the set of all possible data sequences. The associated joint ML channel estimate, $\hat{\mathbf{h}}$, is given by

$$\hat{\mathbf{h}} = \hat{\boldsymbol{\mathcal{S}}}_k^{\dagger}\mathbf{r}_k \tag{3.3}$$

where $\hat{\boldsymbol{\mathcal{S}}}_k^{\dagger}$ is the pseudo-inverse[2] of $\hat{\boldsymbol{\mathcal{S}}}_k$.

Using (3.3), the outer minimization of (3.2) can be expressed according to

$$\Lambda\left(\hat{\boldsymbol{\mathcal{S}}}_k\right) = \underset{\hat{\boldsymbol{\mathcal{S}}}_k \in \mathbb{T}_{\mathcal{A}}}{\arg\min} \left\|\hat{\mathbf{Q}}_k \mathbf{r}_k\right\|^2 \tag{3.4}$$

where $\hat{\mathbf{Q}}_k$ is the projection onto the orthogonal complement of the range space of $\hat{\boldsymbol{\mathcal{S}}}_k$

$$\hat{\mathbf{Q}}_k = I - \hat{\boldsymbol{\mathcal{S}}}_k \hat{\boldsymbol{\mathcal{S}}}_k^{\dagger} \tag{3.5}$$

Note that the phase can't be determined in the joint ML channel and data estimation. In particular, if $\left(\hat{\boldsymbol{\mathcal{S}}}_k, \hat{\mathbf{h}}\right)$ is the joint ML channel and data estimates, then for all $\phi \in \angle\mathcal{A}$, $\left(e^{j\phi}\cdot\hat{\boldsymbol{\mathcal{S}}}_k, e^{-j\phi}\cdot\hat{\mathbf{h}}\right)$ also yields the same result.[3] This limitation is common to *all* blind acquisition algorithms, but can be easily overcome with *differential* encoding and decoding.

Existing sub-optimal blind algorithms can be classified into two groups, Block-iterative algorithms and Time-Recursive algorithms. Block-iterative algorithms process a block of data at a time. On each iteration of the same block, information from the previous iteration(s) is used to improve the channel estimates. In contrast, time-recursive algorithms use a new received sample, $r[t+1]$, to update the channel estimate obtained at time t, from all previously received time samples. Recursive algorithms often have a

---

[1] Toeplitz data matrix, $\hat{\boldsymbol{\mathcal{S}}}_k$, is defined in (2.6).

[2] Pseudo-inverse of a full column rank matrix, $\hat{\boldsymbol{\mathcal{S}}}_k$, is given by $\hat{\boldsymbol{\mathcal{S}}}^{\dagger} = \left(\hat{\boldsymbol{\mathcal{S}}}^T\hat{\boldsymbol{\mathcal{S}}}\right)^{-1}\hat{\boldsymbol{\mathcal{S}}}^T$.

[3] The phase of the signal constellation is denoted by $\angle\mathcal{A}$. For QPSK, possible values of $\angle\mathcal{A}$ ={0^0, 90^0, 180^0, 270^0} or {45^0, 135^0, 225^0, 315^0}.

processing delay that is much less than that required by block processing and are suitable for real-time processing.

Two fundamental block-iterative algorithms are the Alternating Maximization (AM) algorithm and the Expectation Maximization (EM) algorithm. The AM algorithm is an approximate joint ML-based sequence and parameter estimation algorithm [26], [27]. The estimation process continues until the difference between successive iterations drops below some threshold. The sequence estimation step is implemented with the Viterbi Algorithm while the channel estimate is implemented by the data-aided ML parameter estimate. The EM Algorithm is a well-known algorithm for maximum likelihood estimation using "incomplete data" [28]. In [26], the performance of the EM algorithm to estimate a channel with an unknown data sequence is discussed extensively.

Since the Viterbi Algorithm is a recursive algorithm, it is not surprising that the variants of this algorithm are also recursive. These algorithms modify the VA to incorporate channel estimation in the case of an unknown channel. Two such algorithms are Conventional Adaptive MLSE (CA-MLSE) and Per-Survivor Processing (PSP). A detailed description of these two time-recursive algorithms will now be presented.

3.2.1 Conventional Adaptive MLSE (CA-MLSE)

The conventional Adaptive MLSE (CA-MLSE) algorithm uses the concept of MLSE along with a single adaptive channel estimator [4], [20], [29]-[33]. In this algorithm, the VA is applied with an estimate of the unknown channel impulse response. The unknown channel is estimated adaptively using data-aided estimation in a decision-directed fashion. The data-aiding sequence is obtained via delayed tentative decisions by tracing back through the survivor history. This is illustrated in Figure 3.2. The estimated sequence has a trace-back depth of D symbols. The tentatively estimated sequence that is used to update the unknown parameter, has a delay of d symbols where, $d << D$. The CA-MLSE has been successfully applied to 2G standards like GSM and IS-136 for the purpose of channel tracking.

Now let us formally describe CA-MLSE in a framework that will help contrast it with other channel estimation schemes. Let the transition from the i-th state value at time index k to the j-th state value at next time index $(k+1)$ be denoted by $(i_k \to j_{k+1})$, i.e., $\sigma_k^i \to \sigma_{k+1}^j = i_k \to j_{k+1}$, where i and j are given by (2.25) and (2.26) respectively. $\hat{\mathbf{h}}_k = \left[\hat{h}[L], \hat{h}[L-1], \ldots, \hat{h}[0]\right]^T$ represents the vector of the estimated channel impulse response coefficients, where the subscript k represents the time dependence, L is the memory of the channel, and $\mathbf{h} = \left[h[L], h[L-1], \ldots, h[0]\right]^T$ is the true channel. The transition cost at epoch k is given by

$$e_{ij}[k] = F(i_k \to j_{k+1}, r[k], \hat{\mathbf{h}}_k) \tag{3.6}$$

where $F(\cdot)$ describes the functional dependence of transition on the received signal, $r[k]$, and the estimated channel vector $\hat{\mathbf{h}}_k$. Channel estimation is dependent on the sequence of tentative decisions $\{\hat{a}[l]\}_{l=-L}^{k-d}$ according to (3.7)

$$\hat{\mathbf{h}}_k = G\left(r[k-d], \{\hat{a}[l]\}_{l=-L}^{k-d}\right) \tag{3.7}$$

where $G(\cdot)$ denotes the functional relationship. The cumulative cost metric for the $(i_k \to j_{k+1})$ transition is given by

$$\mathcal{C}_j[k+1] = \min_{i \in \mathcal{T}_j}\{\mathcal{C}_i[k] + e_{ij}[k]\} \tag{3.8}$$

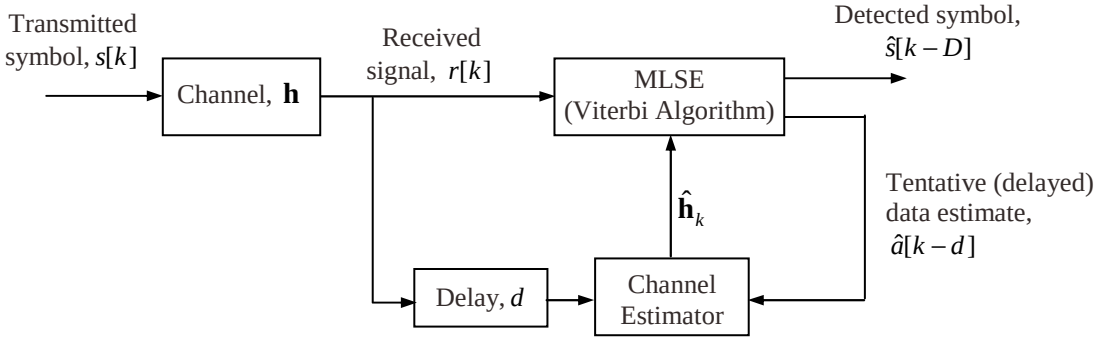


Figure 3.2: Structure of CA-MLSE.

where $\mathcal{T}_j$ is set of all allowable transitions into the j-th state from the i-th state, $\mathcal{C}_j[k+1]$ is the cumulative cost metric at state j of the next stage $(k+1)$, and $\mathcal{C}_i[k]$ is the cumulative cost metric at state i of the present stage k. The survivor at the j-th state of the $(k+1)$-th stage is defined by

$$I_j[k+1] = \arg\min_{i \in \mathcal{T}_j} \{\mathcal{C}_i[k] + e_{ij}[k]\} \tag{3.9}$$

The functional dependency in (3.6) can be written explicitly as

$$e_{ij}[k] = \left| \mathbf{x}_k \cdot \hat{\mathbf{h}}_k - r[k] \right|^2 \tag{3.10}$$

where $\mathbf{x}_k = \left[x[k-L], x[k-L+1\}, \ldots, x[k] \right]$ is the symbol vector associated with the $(i_k \rightarrow j_{k+1})$ transition. The estimation error is given by

$$\varepsilon[k-d] = r[k-d] - \hat{\mathbf{a}}[k-d] \cdot \hat{\mathbf{h}}_k \tag{3.11}$$

where $\hat{\mathbf{a}}[k-d]$ is the vector of tentative decisions with a delay d according to

$$\hat{\mathbf{a}}[k-d] = \left[\hat{a}[k-d-L], \ldots, \hat{a}[k-d] \right] \tag{3.12}$$

We will now discuss two commonly used recursive channel estimation algorithms.

3.2.1.1 Channel Update Using LMS

Once the estimation error is found, the channel vector is updated using any adaptive algorithm. If classical Least Mean Square (LMS) is used, we have [9]

$$\hat{\mathbf{h}}_{k+1} = \hat{\mathbf{h}}_k + \beta \cdot \varepsilon[k-d] \cdot \hat{\mathbf{a}}^H[k-d] \tag{3.13}$$

where β is the step size, $(\cdot)^H$ represents the Hermitian transformation of a matrix, and the estimation error, $\varepsilon[k-d]$, is given by (3.11). The constant β is selected as a trade-off between speed of convergence and stability, as in ordinary stochastic gradient algorithm. The channel updating scheme is illustrated in Figure 3.3. This figure will help distinguish CA-MLSE from other trellis based blind estimators like PSP. The survivor to a particular state is shown by a solid line. Note that all the states have the same channel estimates.

3.2.1.2 Channel Update Using RLS

If the more complex Recursive Least Square (RLS) algorithm is used, at each step, we evaluate the Kalman gain vector, $\mathbf{k}$, update the inverse of the correlation matrix, $\mathbf{P}$, and then use this information to update the channel estimation [9], thus

$$\mathbf{k}_{k+1} = \frac{\mathbf{P}_k \cdot (\hat{\mathbf{a}}_{k-d})^H}{\lambda + \hat{\mathbf{a}}_{k-d} \cdot \mathbf{P}_k \cdot (\hat{\mathbf{a}}_{k-d})^H} \tag{3.14}$$

$$\mathbf{P}_{k+1} = \frac{1}{\lambda}\left[\mathbf{P}_k - \mathbf{k}_{k+1} \cdot \hat{\mathbf{a}}_{k-d} \cdot \mathbf{P}_k\right] \tag{3.15}$$

$$\hat{\mathbf{h}}_{k+1} = \hat{\mathbf{h}}_k + \mathbf{k}_{k+1} \cdot \varepsilon[k-d] \tag{3.16}$$

where λ is the forgetting factor, which limits the effective integration time of the least square estimate $(0 \le \lambda \le 1)$.

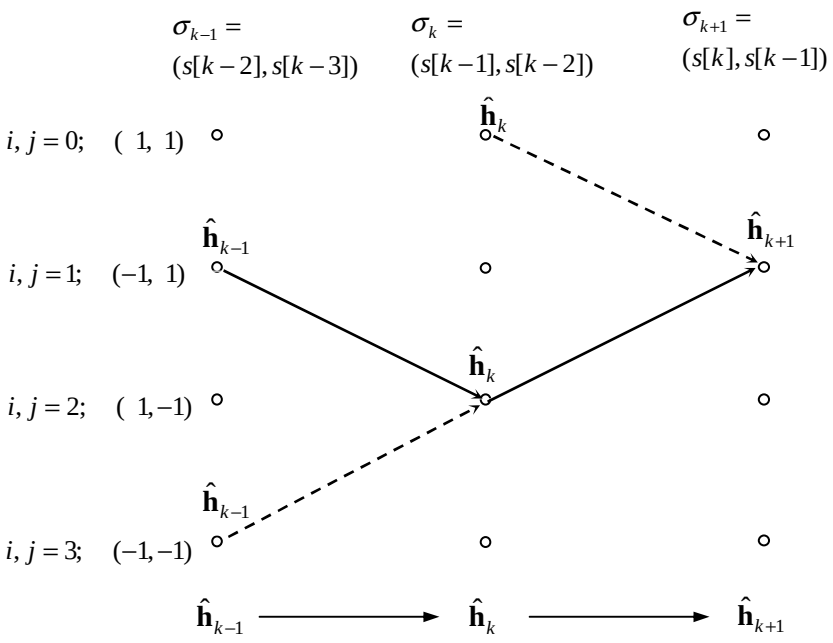


Figure 3.3: Channel estimation in CA-MLSE for $\mathcal{A} = \{1, -1\}$, $L = 2$.

The tentative decision delay parameter, d, is a fundamental design trade-off in CA-MLSE. A large delay is required for reliable decision feedback, while a small delay is desirable to track or acquire the channel efficiently. The choice of d on tracking performance of CA-MLSE is discussed in [34], [35].

3.2.2 Per-Survivor Processing (PSP)

Per-Survivor Processing (PSP) offers a general framework for the approximation of Maximum Likelihood Sequence Estimation (MLSE) algorithms whenever the calculation of the transition metric in the classical Viterbi Algorithm is affected by unknown quantities [35], [36]. A typical example of this unknown quantity is the channel impulse response. The concept of PSP is based on the idea of embedding data-aided estimation techniques into the structure of the VA. The data-aided estimation technique removes or reduces the effect of uncertainty in the transition metrics of the VA.

Now we will explain the basic idea of PSP as described in Figure 3.4. In PSP, the basic idea is that, if we have unknown parameters that prevent exact calculation of the transition metrics, we use estimates of those parameters based on the symbol sequence associated with the surviving path leading to that transition. If any particular survivor is correct, the corresponding estimates are also correct, since they are updated using a correct symbol sequence. Each survivor is extended based on estimates obtained using its associated data sequence, i.e., its associated data history. Note that in PSP the channel estimation is performed in a decentralized manner, as opposed to the global manner of conventional receivers. The per-survivor estimator associated with the best survivor has no delay, thus making PSP very attractive for fast time-varying channels. On the other hand, the performance of CA-MLSE degrades due to the decision delay involved in the VA and the less reliable tentative decisions.

To describe the working principle of PSP let us first define the following quantities. The transition cost for the $(i_k \rightarrow j_{k+1})$ transition is given by

$$e_{ij}[k] = F(i_k \rightarrow j_{k+1}, r[k], \hat{\mathbf{h}}_k^i) \tag{3.17}$$

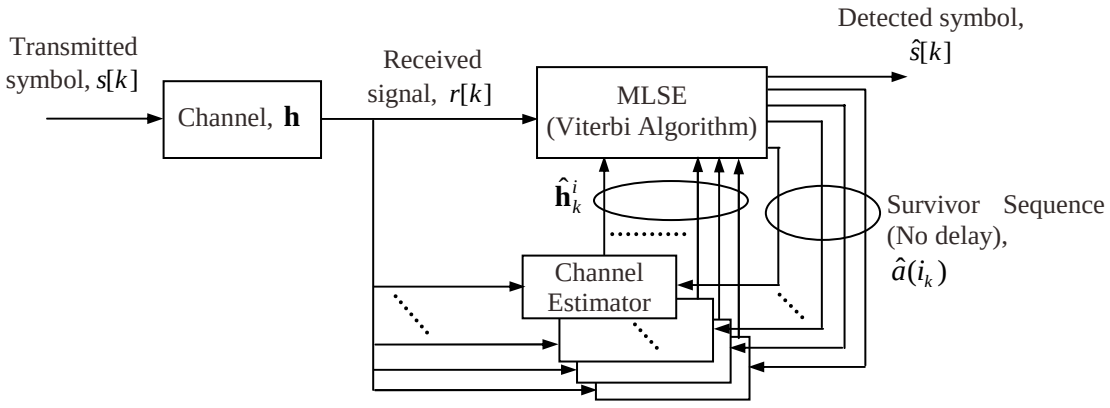


Figure 3.4: Structure of Per-Survivor Processing.

where $F(\cdot)$ denotes the functional dependency of the transition on the received signal, $r[k]$, and $\hat{\mathbf{h}}_k^i$ is the estimated channel vector; superscript i indicates the state dependency, while the subscript k represents time dependency on the channel impulse response estimate. If $\{\hat{a}_l(i_k)\}_{l=-L}^{k}$ denotes the symbol sequence associated with the survivor of state i at epoch k, then the channel estimation of that state is given by

$$\hat{\mathbf{h}}_k^i = G\left(r[k], \{\hat{a}_l(i_k)\}_{l=-L}^{k}\right) \tag{3.18}$$

where the functional dependency of the channel estimation on the received signal and the survivor sequence is described by $G(\cdot)$. The equations for the cumulative cost metric and the survivors for the $(i_k \rightarrow j_{k+1})$ transition are the same as shown for MLSE.

$$\mathcal{C}_j[k+1] = \min_{i \in \mathcal{T}_j}\{\mathcal{C}_i[k] + e_{ij}[k]\} \tag{3.19}$$

$$I_j[k+1] = \arg\min_{i \in \mathcal{T}_j}\{\mathcal{C}_i[k] + e_{ij}[k]\} \tag{3.20}$$

The formation of transition cost metrics in (3.17) in explicit form is given by

$$\begin{aligned} e_{ij}[k] &= \left|e(i_{k\rightarrow} j_{k+1})\right|^2 \\ &= \left|\hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \cdot \hat{\mathbf{h}}_k^i - r[k]\right|^2 \end{aligned} \tag{3.21}$$

where $e(i_k \to j_{k+1})$ is the estimation error, and $\hat{\mathbf{a}}(i_k \to j_{k+1})$ is the symbol sequence associated with the $(i_k \to j_{k+1})$ transition and is partially found in the survivor history according to

$$\hat{\mathbf{a}}(i_k \to j_{k+1}) = \left[\hat{a}_{k-L}(i_k), \ldots, \hat{a}_{k-1}(i_k), \hat{a}_k(j_{k+1})\right] \tag{3.22}$$

Note the difference between (3.10) and (3.21) as far as state dependency is concerned. In PSP, each state has its own channel estimate as opposed to a single channel estimate maintained in CA-MLSE for all states. Also note that there is no delay involved in the survivor sequence, which improves performance under fast-time varying channels. The transition cost is the same as the square of the estimation error.

3.2.2.1 Channel update using LMS

If Least Mean Square (LMS) is employed, the channel vector is updated according to [9]

$$\hat{\mathbf{h}}_{k+1}^{j} = \hat{\mathbf{h}}_k^{i} + \beta \cdot e(i_k \to j_{k+1}) \cdot \hat{\mathbf{a}}^H(i_k \to j_{k+1}) \tag{3.23}$$

where β is the step size, $(\cdot)^H$ represents the Hermitian transformation of a matrix, and the estimation error, $e(i_k \to j_{k+1})$, is given by (3.21). Figure 3.5 gives a pictorial description of the channel update scheme in PSP.

3.2.2.2 Channel estimate using RLS

If Recursive Least Square (RLS) is used, then for each state first we evaluate the Kalman gain vector, $\mathbf{k}$, update the inverse of the correlation matrix, $\mathbf{P}$, and use this information to update the channel estimation [9]

$$\mathbf{k}_{k+1}^{j} = \frac{\mathbf{P}_k^{i} \cdot (\hat{\mathbf{a}}(i_k \to j_{k+1}))^H}{\lambda + \hat{\mathbf{a}}(i_k \to j_{k+1}) \cdot \mathbf{P}_k^{i} \cdot (\hat{\mathbf{a}}(i_k \to j_{k+1}))^H} \tag{3.24}$$

$$\mathbf{P}_{k+1}^{j} = \frac{1}{\lambda}\left[\mathbf{P}_k^{i} - \mathbf{k}_{k+1}^{j} \cdot \hat{\mathbf{a}}(i_k \to j_{k+1}) \cdot \mathbf{P}_k^{i}\right] \tag{3.25}$$

$$\hat{\mathbf{h}}_{k+1}^{j} = \hat{\mathbf{h}}_k^{i} + \mathbf{k}_{k+1}^{j} \cdot e(i_k \to j_{k+1}) \tag{3.26}$$

where λ is the forgetting factor $(0 \le \lambda \le 1)$.

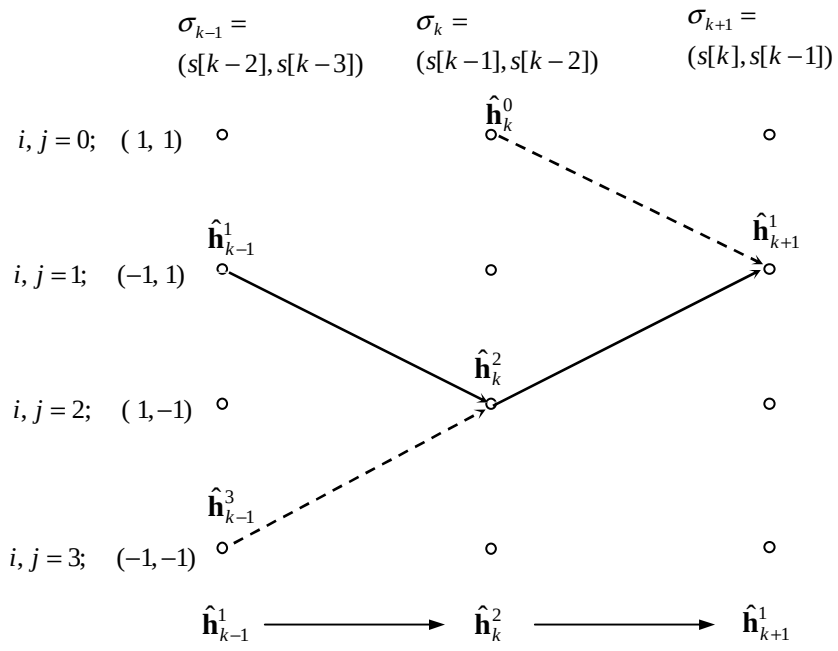


Figure 3.5: Channel estimation in PSP for $\mathcal{A} = \{1, -1\}$, $L = 2$.

PSP was first applied to the cancellation of ISI in the reduced state sequence estimation (RSSE) [3], [32]. Here the residual ISI term forms the unknown parameter to be estimated. Also, the state reduction technique of Delayed Decision Feedback Sequence Estimation (DDFSE) [15] can be viewed as an application of PSP, with the discarded portion of the state information being the unknown parameter. The general concept of PSP was later introduced in [35], [36]. Recently, other authors have proposed PSP for specific applications other than RSSE. PSP has been applied to phase synchronization in [35], [37]-[39]. Multi-User Detection (MUD) is another application of PSP which has been discussed in [40], [41].

3.2.3 Generalized Per-Survivor Processing (GPSP)

A closer look at (3.19) and (3.21) reveals that the transition metric depends not only on the last L transmitted symbols, but on the entire symbol sequence history through the associated channel estimate. Therefore, the use of any dynamic programming like the VA doesn't satisfy [42], [43]. Since the metrics are path-dependent, instead of being state-dependent as in an ordinary (known channel) VA, this channel estimation algorithm could be viewed as a *tree-search* problem. Any forced-folding of the tree-search into a trellis search is sub-optimal. This is because, if a trellis-search algorithm, i.e., standard PSP, is used for path-dependent transitions, then it is possible that the best path at time k is not the path with minimum metric entering the corresponding state at some time $t < k$. As a result the best path might be discarded by PSP at time t. Thus, standard PSP is sub-optimal for path-dependent transitions. The optimal tree-search for path-dependent transition is the *exhaustive search*, which has complexity growing exponentially with the number of observed symbol intervals.

The tree-search algorithms have been classified into three groups as defined in [44]; *depth-first*, *metrics-first*, *breadth-first*. A depth-first algorithm "extends" a single path until it is apparent that the path is no longer good enough to pursue. It then *backtracks* to explore alternate paths. The Fano Algorithm is an example of a depth-first algorithm [45], [46]. In a metrics-first algorithm, all the paths are ranked according to the metric, and extend the path with the current best metric. When the path is no longer the best, it is "stored," and the new best path is extended. If the metric of the current best path exceeds that of the previous best path, the previous best path is extended. The Multiple-Stack algorithm is an example of this type of algorithm [47]. A breadth-first algorithm extends many probable paths at a given time and excludes contending paths according to their metrics. There is no *backtracking* involved in breadth-first algorithms. Both the VA and the PSP are examples of breadth-first algorithms where exactly M^L survivors are extended at each stage.

Figure 3.6 describes the classification of tree search algorithms [48]. Our focus in this dissertation is on the breadth-first algorithms mainly because of implementation

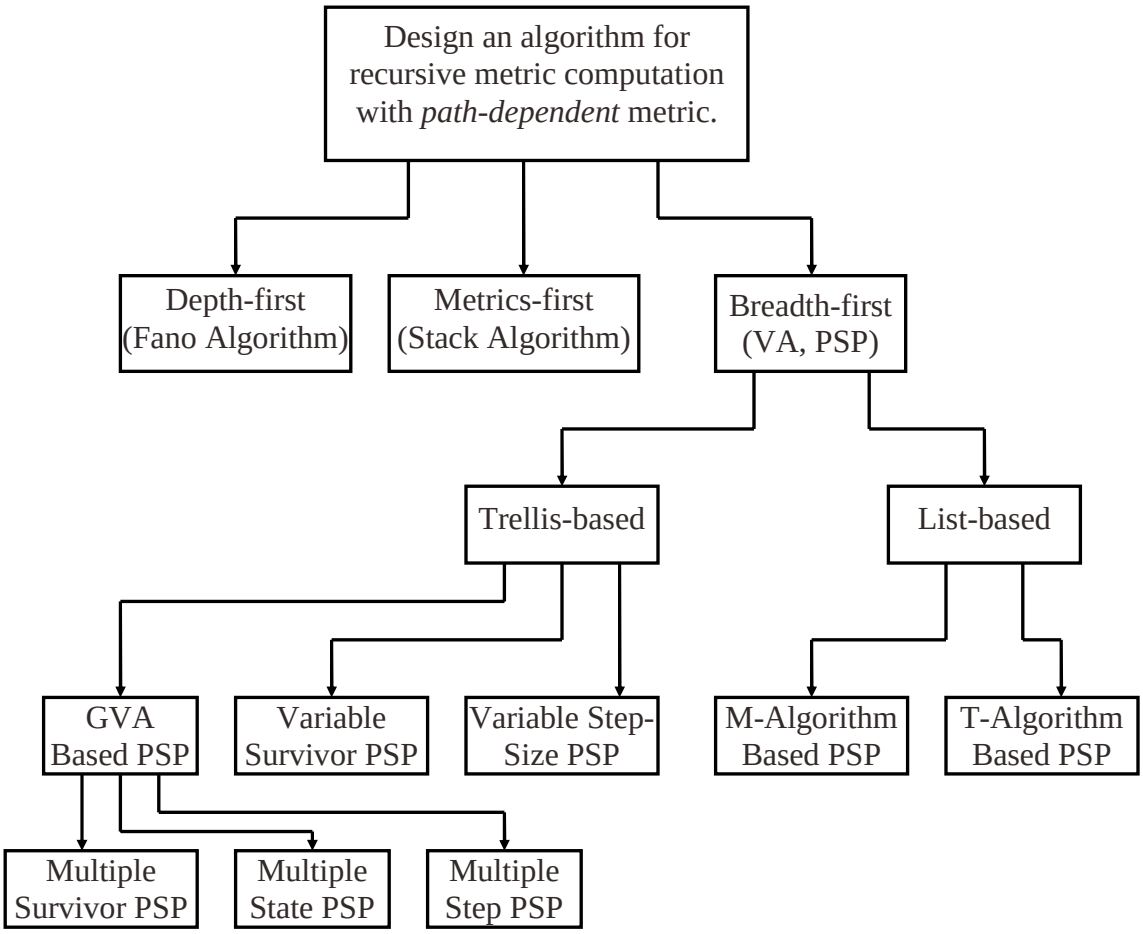


Figure 3.6: Design of recursive algorithms with path dependent metrics.

issues. Usually breadth-first algorithms are synchronous, parallel, and recursive, so that the operations can be pipelined and readily implemented in VLSI. Examples of VLSI implementations that take advantage of parallelism in breadth-first algorithm are given in [49]-[51]. Breadth-first algorithms can be further divided into two categories depending on the trellis structure of the algorithm. In the trellis based algorithm, there is a definite trellis structure which is built on the concept of "state variable." In practice, variations of the Viterbi Algorithm are used to exploit the trellis structure. Here the sequence of states and sequence of symbols have a one-to-one correspondence. The M-algorithm and T-algorithm are classic examples of list-based algorithms. Viterbi algorithms cannot be used since the concept of state doesn't work here. Other application-specific versions of

suboptimal breadth-first algorithms have been proposed in the literature [52], [53], but have been omitted in this discussion for the sake of brevity.

3.2.3.1 GVA-Based PSP

In this algorithm, PSP is applied to the Generalized Viterbi Algorithm (GVA). The notion of GVA was introduced in [54]. A generalized VA is an algorithm that embeds one of many recursive tree search techniques with the standard VA. Note that a known-channel VA is optimal since the transition metrics are only state-dependent. However, in PSP, since the channel is unknown, the VA is no longer optimal because the metrics become path-dependent. In case of path-dependent metrics, the tree search technique is important, since it is possible that the most likely path could be eliminated early in the search. Generalized PSP (GPSP) is a straightforward extension of the GVA, with imbedded per-survivor channel estimation included in the search process. There are many conceivable versions of GVA (GPSP), i.e. multiple-survivor VA (PSP), multiple-state VA (PSP), and multiple-step VA (PSP) [48]. A brief description of GVA algorithms is given here. In the next section, we present a detailed discussion of the multiple-survivor PSP.

Multiple-Survivor VA (PSP):

In this modification, the algorithm maintains multiple survivors, $S \geq 1$, per state. Maintaining more survivors makes it less likely that the best path will be eliminated early in the search process. However, more survivors increase complexity. This algorithm can be viewed as a combination of the VA and M-algorithm. In a known parameter case, Multiple-Survivor VA provides an ordered list of S maximum likelihood paths. The performance of this algorithm has been reported in [55], [56].

Multiple-State VA (PSP):

In this version of GVA, the memory length is over estimated. For example, if the channel memory is L and the additional number of states is N, then there are M^N states

maintained for each of the M^L possible channel states in the VA. In this case the number of hypothetical sequences increases from M^L to M^{L+N}. This concept was introduced in [48], [57].

Multiple-Step VA (PSP):

In this extension of GVA, the VA prolongs the elimination of survivors by taking multiple steps through the trellis. This type of algorithm has been suggested for high-speed implementation of Viterbi decoder exploiting parallel processing [58], [59].
The multiple-survivor PSP will now be discussed in detail.

3.2.3.1.1 Multiple-Survivor PSP

This algorithm retains $S \geq 1$ best estimates of the transmitted data sequence into each state, i_k. However, unlike Multiple-Survivor VA, which produces S globally best estimates of the transmitted sequence, in Multiple-Survivor PSP more survivors can compensate for the insufficient channel knowledge. In the standard VA, at each step and at each state, the algorithm retains the best survivor. The best survivor is picked on the assumption that the channel estimate that is used for metric evaluation is either the correct one or is a good approximation of the true channel. In other words, there is no uncertainty in the evaluation of the transition metric. However, this is not true for a joint data and channel estimation algorithm like PSP. So no path may be deleted in favor of another especially over the first few stages. On the other hand, it is also impossible to keep every path since this would become an unmanageable exhaustive search. Thus the algorithm proposes a finite number ($S \geq 1$) of paths into each state, which translates into a complexity/performance trade-off.

In order to describe the algorithm, let us denote the channel estimates of each survivor into each state, i_k, at epoch k by

$$\hat{\mathbf{h}}^n(i_k) = \left[\hat{h}_L^n(i_k), \ldots, \hat{h}_1^n(i_k), \hat{h}_0^n(i_k) \right]^T \tag{3.27}$$

where superscript $n = 1, 2, \ldots, S$ denotes the survivor index of S survivors retained for each state i_k. Since there are M^L states and M transitions to each state, we have M^{L+1} transitions at each step. Now there are S survivors for each state, so we have a total of $S \cdot M^{L+1}$ transitions for the $(i_k \rightarrow j_{k+1})$ state transition. The cost associated with each of these transitions is given by

$$e^n(i_k \rightarrow j_{k+1}) = \left| \hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \cdot \hat{\mathbf{h}}^n(i_k) - r[k] \right|^2 \tag{3.28}$$

where $\hat{\mathbf{a}}(i_k \rightarrow j_{k+1})$ is the symbol sequence associated with the $(i_k \rightarrow j_{k+1})$ transition and is partially found in the survivor history according to

$$\hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) = \left[\hat{a}_{k-L}(i_k), \ldots, \hat{a}_{k-1}(i_k), \hat{a}_k(j_{k+1}) \right] \tag{3.29}$$

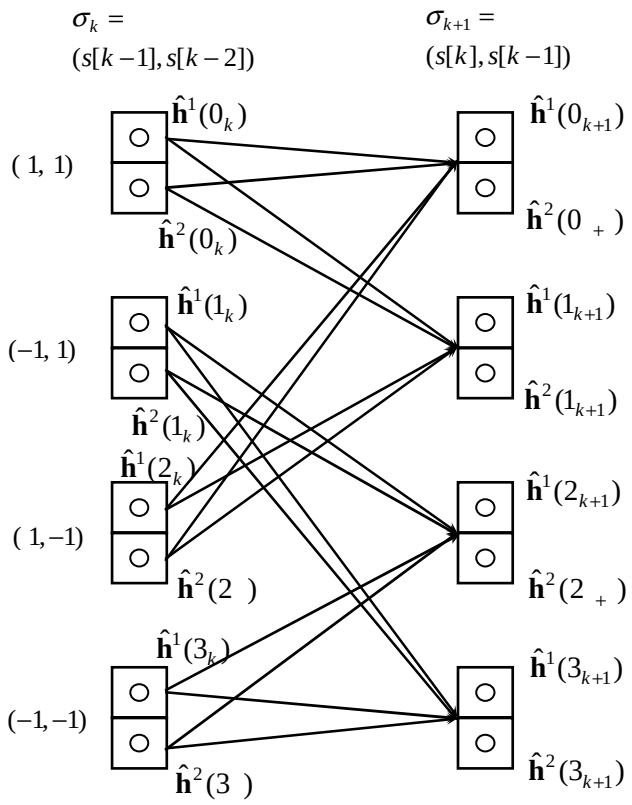


Figure 3.7: Multiple-Survivor PSP for $S = 2$.

Note the difference between a conventional Multiple-Survivor VA and a Multiple-Survivor PSP as far as the number of transitions is concerned. In a Multiple-Survivor VA with a known channel or a unique channel estimate for each survivor of a given state, there are M^{L+1} transition metrics. This results because the survivor history does not affect the transition metric. Let us denote the cumulative metric of the n-th survivor of state i_k by $\mathcal{C}^n(i_k)$. For each successor state j_{k+1}, there are $S \cdot M$ candidate survivors out of which only S are retained according to

$$\mathcal{C}^m[j_{k+1}] = \min_{n;\ i \in \mathcal{T}_j} \{\mathcal{C}^n[i_k] + e^n(i_k \rightarrow j_{k+1})\}; \quad \forall m, n = 1, 2, \ldots, N \tag{3.30}$$

in which $m = 1, 2, \ldots, S$ denotes the survivor index of the successor state j_{k+1}, and $\mathcal{T}_j$ is the set of allowable transitions into the state j_{k+1}. Now the channel estimates of each of the survivors are updated using adaptive algorithms. If LMS is used, the update process is

$$\hat{\mathbf{h}}^m(j_{k+1}) = \hat{\mathbf{h}}^n(i_k) + \beta \cdot e^n(i_k \rightarrow j_{k+1}) \cdot \hat{\mathbf{a}}^H(i_k \rightarrow j_{k+1}) \tag{3.31}$$

where β is the step size.

If the RLS is used, the Kalman gain vectors, $\mathbf{k}$, the inverse of correlation matrices, $\mathbf{P}$, and the channel estimates for each of the survivors, are updated according to

$$\mathbf{k}^m(j_{k+1}) = \frac{\mathbf{P}^n(i_k) \cdot (\hat{\mathbf{a}}(i_k \rightarrow j_{k+1}))^H}{\lambda + \hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \cdot \mathbf{P}^n(i_k) \cdot (\hat{\mathbf{a}}(i_k \rightarrow j_{k+1}))^H} \tag{3.32}$$

$$\mathbf{P}^m(j_{k+1}) = \frac{1}{\lambda}\left[\mathbf{P}^n(i_k) - \mathbf{k}^m(j_{k+1}) \cdot \hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \cdot \mathbf{P}^n(i_k)\right] \tag{3.33}$$

$$\hat{\mathbf{h}}^m(j_{k+1}) = \hat{\mathbf{h}}^n(i_k) + \mathbf{k}^m(j_{k+1}) \cdot e^n(i_k \rightarrow j_{k+1}) \tag{3.34}$$

where λ is the forgetting factor. The tracking performance of Multiple-Survivor PSP has been reported in [60]. A pictorial description of the Multiple-Survivor version of PSP is illustrated in Figure 3.7.

3.2.3.2 Variable-Survivor PSP

Standard PSP uses as many per-survivor channel estimators as the number of surviving paths. On the other hand, conventional adaptive MLSE uses only one channel estimator using the tentative decisions involving some delay. These two extreme scenarios motivate the use of an arbitrary number of channel estimators, which we call Variable-Survivor PSP [61]. This idea of a variable number of survivors has been applied to Trellis coded modulation [62].

Let $S = M^L$ denote the number of survivors retained at each step of the algorithm in a standard PSP. In Variable-Survivor PSP, there is an arbitrary number N, $(1 \le N \le S)$, of channel estimators that are associated with each of the current N best survivors. The number N is a design parameter given a complexity/performance criterion that must be met. The steps of the algorithm are the following.

- Select the best survivor and the N best survivors.
- Extend each of the N best survivors using its own channel estimator.
- Extend each of the remaining $(S - N)$ survivors using the best survivor's channel estimator.
- Update each of the N channel estimators according to their extension.

Figure 3.8 is an example of Variable-Survivor PSP for $S = 4, N = 2$. The numbers associated with each node represent an ordered list of survivors, i.e., 2 is the second best survivor and so on. In the first stage, after extension, the previous best survivor becomes the current best survivor. The channel estimate of the previous best survivor is updated to obtain both the current best and second best channel estimate. For third and fourth best survivor, the best survivor's channel estimate is used. In the second stage, the previous second best survivor becomes the current best survivor. The channel estimate of the previous second best survivor is updated to give the current best channel estimate, and the channel estimate of the previous best survivor is updated to yield the current second best channel estimate.

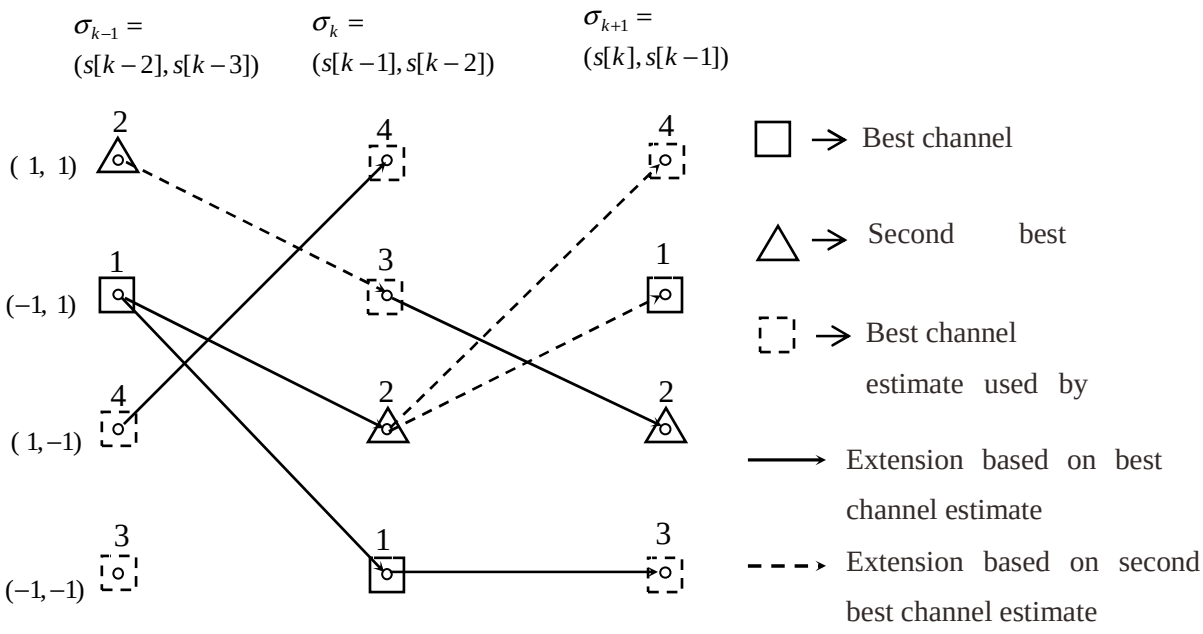


Figure 3.8: Channel update in Variable-Survivor PSP for $S = 4, N = 2$.

3.2.3.3 Variable Step-Size PSP

Channel estimation algorithm can be optimized for estimation error and convergence time. Even though both of these figures of merit are important, they are contradictory, and a trade-off must be made depending on system's requirements. For example, if LMS is used as a channel estimator, the step size parameter, β , dictates this trade-off. With a larger β , the algorithm converges faster, but the estimation error increases[4].

The standard PSP was proposed on a fixed step size for all survivors. However, in a time-varying multipath channel, there may not be a fixed optimum step size that works for all channels. Especially in the acquisition mode, the receiver has no knowledge of how fast the channel is changing. Because of this problem, a variable step-size channel estimator is applied to each of the survivors in the conventional PSP. As explained in Section 3.2.2, the channel update for the $(i_k \to j_{k+1})$ transition using LMS is given by

[4] Here "step" does not refer to the same entity as that for Multiple-Step PSP.

$$\hat{\mathbf{h}}_{k+1}^{j} = \hat{\mathbf{h}}_{k}^{i} + \beta \cdot e_{ij}[k] \cdot \hat{\mathbf{a}}^{H}(i_k \rightarrow j_{k+1}) \tag{3.35}$$

The step size β is fixed and is the same for all survivor paths. In the proposed algorithm, the step size parameters are first decided by the survivor paths. The survivor paths along with the variable step size parameters then estimate the channel coefficients. Therefore, the channel update is

$$\hat{\mathbf{h}}_{k+1}^{j} = \hat{\mathbf{h}}_{k}^{i} + \beta(i_k \rightarrow j_{k+1}) \cdot e_{ij}[k] \cdot \hat{\mathbf{a}}^{H}(i_k \rightarrow j_{k+1}) \tag{3.36}$$

Note that the step size parameter, $\beta(i_k \rightarrow j_{k+1})$, is now dependent on the $(i_k \rightarrow j_{k+1})$ transition and is also adjustable with time, k. The variable step size parameter can be updated using variable step size algorithms. The variable step size LMS algorithm, as described in [63], updates the step size by multiplying (adding) or dividing (subtracting) the previous step size by a factor in the adaptive process. If the step size updating factor, $\alpha > 1$, then

$$\begin{aligned} \beta(i_k \rightarrow j_{k+1}) &= \beta(i_{k-1} \rightarrow j_k) \cdot \alpha, \qquad & if\ \beta(i_k \rightarrow j_{k+1}) < \beta_{\max} \\ \beta(i_k \rightarrow j_{k+1}) &= \beta(i_{k-1} \rightarrow j_k) / \alpha, \qquad & if\ \beta(i_k \rightarrow j_{k+1}) > \beta_{\min} \end{aligned} \tag{3.37}$$

If $0 < \alpha < 1$ then

$$\begin{aligned} \beta(i_k \rightarrow j_{k+1}) &= \beta(i_{k-1} \rightarrow j_k) + \alpha, \qquad & if\ \beta(i_k \rightarrow j_{k+1}) < \beta_{\max} \\ \beta(i_k \rightarrow j_{k+1}) &= \beta(i_{k-1} \rightarrow j_k) - \alpha, \qquad & if\ \beta(i_k \rightarrow j_{k+1}) > \beta_{\min} \end{aligned} \tag{3.38}$$

Even though this approach shows significant improvement over the conventional fixed step size LMS with faster convergence and higher accuracy, it is very sensitive to the selection of α. The algorithm effectively shifts the performance dependency from the step size, β, to the step size updating factor, α. To eliminate this problem, another approach has been proposed in [64], where the variable step size is based on the absolute estimation error. The step size updating process is given by

$$\begin{aligned} \beta(i_k \rightarrow j_{k+1}) &= \beta(i_{k-1} \rightarrow j_k) + \left| e_{ij}[k] \right| / \left| \hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \right|, \quad & if\ \beta(i_k \rightarrow j_{k+1}) < \beta_{\max} \\ \beta(i_k \rightarrow j_{k+1}) &= \beta(i_{k-1} \rightarrow j_k) - \left| e_{ij}[k] \right| / \left| \hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \right|, \quad & if\ \beta(i_k \rightarrow j_{k+1}) > \beta_{\min} \end{aligned} \tag{3.39}$$

In [64] it has been shown that Variable Step-Size PSP can approach the performance of the optimized PSP without the knowledge of the channel and optimum

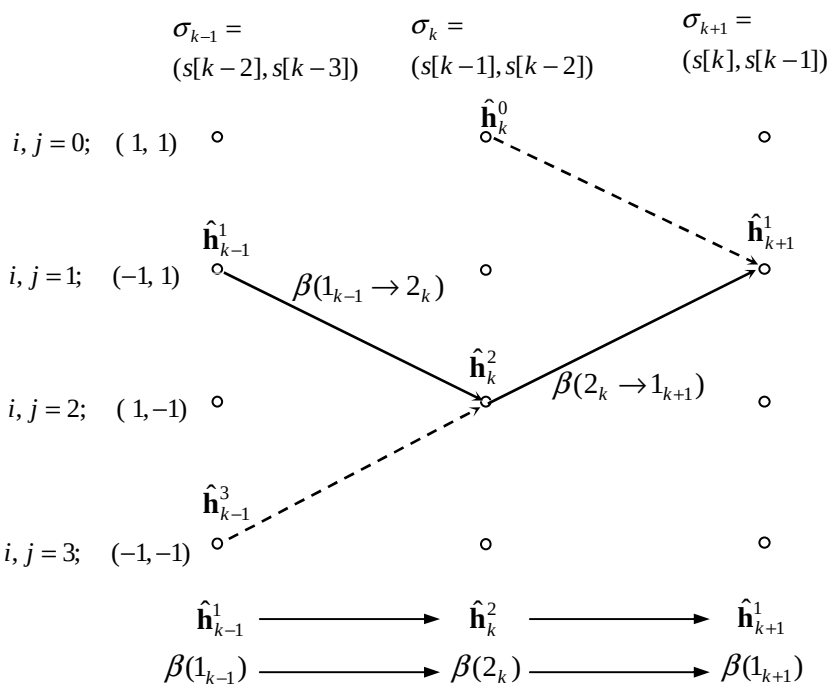


Figure 3.9: Channel update in Variable Step-Size PSP.

step size. The proposed algorithm is also insensitive to the initialization of the step size. Variable Step-Size PSP is illustrated in Figure 3.9. Unlike LMS, the RLS can be viewed as a special case of variable step size LMS [64]. In RLS, the inverse of correlation matrix, $\mathbf{P}$, adapts itself based on its own survivor path. This effectively eliminates dependencies among different paths with one single step size.

3.2.3.4 M-Algorithm Based PSP

In this algorithm, PSP is applied to the well-known M-algorithm [44]. The M-algorithm retains exactly N survivors at any given time.[5] It extends each survivor to M branches to obtain $M \cdot N$ paths, and out of these paths only N paths are maintained, where M is

[5] The number of survivors retained in the M-Algorithm is usually denoted by M. However, we use N to avoid confusion with the size of the alphabet.

the alphabet size, $|\mathcal{A}|$. Now we will describe the algorithm. Let the channel estimates at the k-th epoch be defined by,

$$\hat{\mathbf{h}}_k^s = \left[\hat{h}_k^s[L], \ldots, \hat{h}_k^s[1], \hat{h}_k^s[0]\right]^T \tag{3.40}$$

where $s = 1, 2, \ldots, N$ denotes the survivor index. We define the errors associated with the M possible extensions, η, of each survivor as

$$e_k[s,\eta] = \left|\hat{\mathbf{a}}[s,\eta]\cdot\hat{\mathbf{h}}_k^s - r[k]\right|^2 \tag{3.41}$$

where $\hat{\mathbf{a}}[s,\eta] = \left(\hat{a}_{k-L}^s, \ldots, \hat{a}_{k-1}^s, \hat{a}(\eta)\right)$ denotes the information sequence related to the η extension of the s-th survivor. The M-algorithm maintains the survivors with the N best accumulated metrics, $\mathcal{C}_r[k+1]$, by selecting among all possible $M \cdot N$ candidate extensions according to

$$\left\{\mathcal{C}_1[k+1], \ldots, \mathcal{C}_N[k+1]\right\} = best\ N\left\{\mathcal{C}_s[k] + e_k[s,\eta]\right\}; \quad \forall s = 1, 2, \ldots, N \tag{3.42}$$

in which s is the index of survivors for the k-th epoch. Once the survivors are selected, the channel estimates for each survivor are updated. If LMS is used, the channel update is

$$\hat{\mathbf{h}}_{k+1}^r = \hat{\mathbf{h}}_k^s + \beta\cdot e_k[s,\eta_{k+1}]\cdot\hat{\mathbf{a}}^H[s,\eta_{k+1}] \tag{3.43}$$

where β is the step size.

If the RLS algorithm is used, then the Kalman gain vectors, $\mathbf{k}$, the inverse of correlation matrices, $\mathbf{P}$, and the channel estimates for each of the survivors are updated according to

$$\mathbf{k}_{k+1}^r = \frac{\mathbf{P}_k^s\cdot(\hat{\mathbf{a}}[s,\eta_{k+1}])^H}{\lambda + \hat{\mathbf{a}}[s,\eta_{k+1}]\cdot\mathbf{P}_k^s\cdot(\hat{\mathbf{a}}[s,\eta_{k+1}])^H} \tag{3.44}$$

$$\mathbf{P}_{k+1}^r = \frac{1}{\lambda}\left[\mathbf{P}_k^s - \mathbf{k}_{k+1}^r\cdot\hat{\mathbf{a}}[s,\eta_{k+1}]\cdot\mathbf{P}_k^s\right] \tag{3.45}$$

$$\hat{\mathbf{h}}_{k+1}^r = \hat{\mathbf{h}}_k^s + \mathbf{k}_{k+1}^r\cdot e_k[s,\eta_{k+1}] \tag{3.46}$$

in which, λ is the forgetting factor.

States are defined as

$$\sigma[k] = (s[k-1], s[k-2], s[k-3], s[k-4])$$

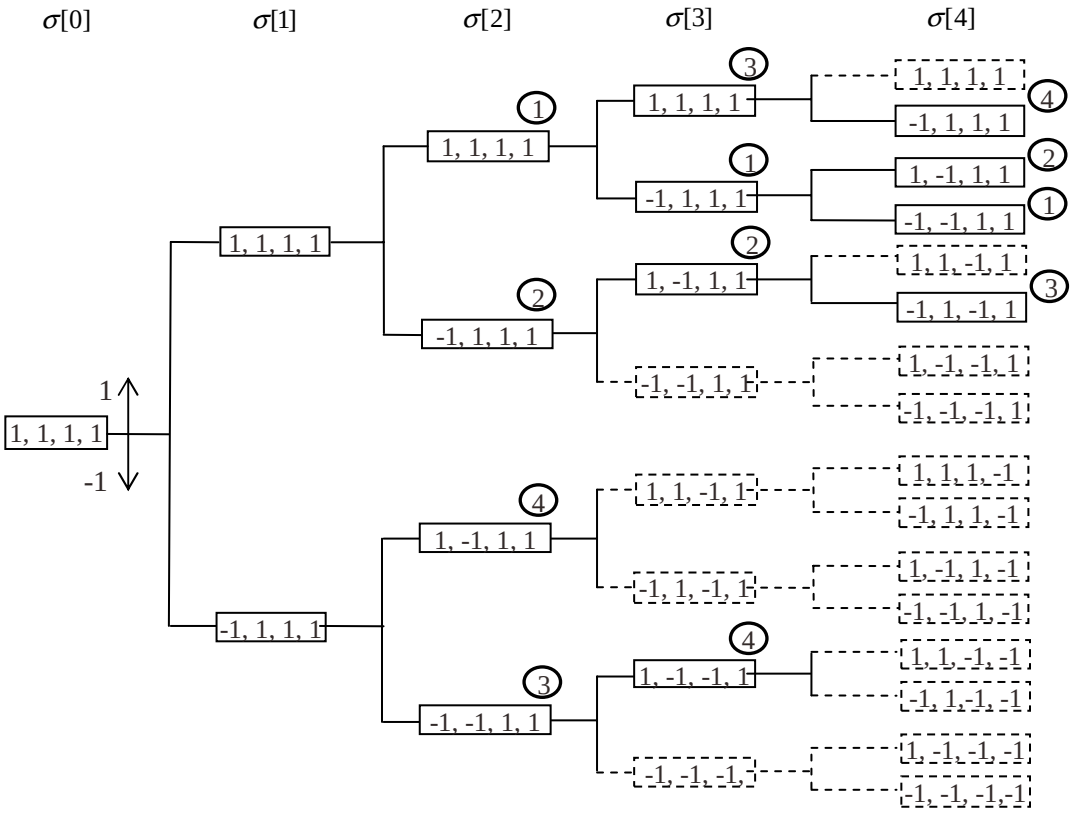


Figure 3.10: The M-algorithm trims the tree-search with $N = 4$ best survivors at each stage; $\mathcal{A} = \{1, -1\}$, $L = 4$.

The tracking performance of the M-algorithm based PSP is discussed in [60], [65]. Note that the M-algorithm is different from the VA even when the number of survivors is the same, $N = M^L$. This is because, in the VA, the survivors are forced to have a different "current state," which corresponds to a unique state in the trellis. However, the M-algorithm is not trellis-based, so there is no constraint on the survivors' recent past. All of the retained survivors are *globally* best, as opposed to the *local* best survivors retained at each state of the VA. Finding the N best survivors at each stage reduces the *exhaustive* search to an M-algorithm based search. This is illustrated in Figure 3.10. Here we have considered BPSK with $N = 4$ and the channel memory, $L = 4$. The initial state is assumed to be all-ones. At each stage, only the four best branches are kept (arbitrarily chosen), which are labeled and indicated by solid lines. Other possible candidate

branches are discarded and shown by dotted lines. While an exhaustive search would have considered all possible sequences with the branches growing exponentially with time, the M-algorithm keeps the size of the tree constant after each extension.

Note the difference between state reduction techniques like DDFSE and the M-algorithm as far as the search process is concerned. Contrast Figure 3.10 with Figure 1.12 (b), where DDFSE has been described for $L = 4, \mu = 2$. Both algorithms retain the same number of survivors; in the DDFSE, $M^{\mu} = 4$ state, four surviving paths are maintained while in the M-algorithm with $N = 4$, the four best branches are maintained at each stage. In DDFSE, the channel states are only partially defined by the trellis, and it uses the surviving sequence to complete the state description. So it doesn't perform well under non-minimum phase channels, where most of the energy lies in the later samples of the channel impulse response. In contrast, the M-algorithm completely defines the channel states and its performance is not limited to minimum-phase channels. In PSP, the M-algorithm has the potential to outperform DDFSE because it retains the N best sequences, which under normal operating conditions should contain the ML path.

3.2.3.5 T-Algorithm Based PSP

The T-Algorithm [66] is another cost-efficient non-trellis based breadth-first algorithm. The Viterbi algorithm, the Generalized Viterbi Algorithm and the M-algorithm, as discussed in previous sections, perform a "fixed" number of computations at each step. In other words, the number of survivors retained in the algorithm remains constant. Even for low-complexity suboptimal algorithms like in the M-algorithm, the number of survivors should be large enough to meet the worst case channel condition. So, in a time varying environment, the algorithm cannot capitalize on periods when the channel is "flat" and the equalization is much easier. The T-algorithm recognizes this fact and has an inherent adaptation ability which reduces the average computational complexity.

An optimal ML decoder chooses the path through the trellis with the minimum (best) metric. In the case of the VA, the search for the best path is done by retaining one survivor to each state in the trellis at each step. The survivor is the one with the minimum

metric over all paths that reach that state. Since the paths with high metric are unlikely to be the ML path, the "exhaustive" nature of this search can be avoided if these paths are discarded. Path metrics are accumulated over time, so selection of paths by comparing with a fixed threshold will not work. Normalizing each path by subtracting the best metric could address this problem. The steps of the T-algorithm are the following:

- Extend each survivor at the k-th epoch to the $(k+1)$-th epoch for all of M possible extensions.
- Update the accumulated metric of each extension.
- Find the extension having the best metric, $\mathcal{C}_b$.
- Subtract $\mathcal{C}_b$ from the metric of each extension and compare it to a threshold, T.
- Delete those paths that produce a metric difference higher than T.

In [66], Simmons shows that the error-rate versus computational complexity behavior for the T-algorithm is far better than that of the Viterbi algorithm. The value of the threshold, T, is a designer's choice given the operating condition (SNR, channel etc.), cost, and the performance criteria to meet.

3.3 Complexity Analysis

In this section a detailed complexity analysis of the various algorithms that have been discussed in the previous sections is presented. All the algorithms are characterized by the following parameters; alphabet size, M, channel order, L, *trace-back depth*, D. Also as before, the parameters N, μ, and S denote the number of survivors retained in M-Algorithm, the reduced memory size in DDFSE and the number of survivors per state in Multiple-Survivor PSP (MS-PSP), respectively.

The computational requirements for practical implementation have been presented by three types of floating point operations (flops): real additions, real multiplications, and

comparisons. Table 3.1 shows the number of operations required per transmitted symbol.[6] In this case, the channel is known and only the metric update is required. All transitions coming from a given current state have common terms that need to be calculated only once. Also with a known channel, the VA can exploit the fact that the candidate signal component doesn't depend on the survivor history, which reduces the computations significantly for long frames. In the M-Algorithm, the N best survivors are sorted out of $M \cdot N$ candidate survivors, which can be implemented using the "Quicksort" algorithm requiring $M.N\log_2(MN)$ comparisons.

Operation	**VA**	**DDFSE**	**M-Algorithm**
# of real addition	$M^L(4M+(4L+2)^*)$	$M^\mu\left[(4\mu+2)^*+4(L+M-\mu)\right]$	$2N(3M+2L\text{-}1)+(2M)^*$
# of real multiplication	$2M^L(M+(2L+2)^*)$	$M^\mu\left[(4\mu+4)^*+2(2L+M-2\mu)\right]$	$2N(M+2L)+(4M)^*$
# of comparison	$M^L(M\text{-}1)$	$M^\mu(M\text{-}1)$	$M.N\log_2(MN)$

Table 3.1: Number of real operations (per received symbol) for different algorithms (known channel).

The computational requirement (per transmitted symbol) for PSP based algorithms are listed in Table 3.2, where the metric and channel update functions are separately calculated. For the channel update we considered the RLS algorithm mainly because of its faster convergence. A detailed computational comparison between the RLS and LMS is given in Appendix A. The number of states, which is determined by the alphabet size, M, and the channel order, L, directly influence the computation. Figure 3.11 and Figure 3.12Figure 3.12 illustrate the effect of modulation order and channel length on complexity. For both BPSK and QPSK, the complexity of MS-PSP and VA-PSP becomes unmanageable with a channel order more than two.

[6] In Table 3.1 the terms marked with $(\,)^*$ do not depend on the received signal and should be calculated only once.

Parameters:

L = Channel Order
M = Size of Alphabet, $|\mathcal{A}|$
μ = "Reduced" Channel Memory Size (DDFSE)
N = Number of Survivors (M-Algorithm)
S = Number of Survivors/State (Multiple-Survivor PSP)

Function	Operations	VA-PSP	DDFSE-PSP	MA-PSP	MS-PSP
Metric Update	# of real addition	$2M^{L}(4M+2L-1)$	$2M^{\mu}(4M+2L-1)$	$2N(4M+2L-1)$	$2S\cdot M^{L}(4M+2L-1)$
	# of real multiplication	$2M^{L}(3M+2L)$	$2M^{\mu}(3M+2L)$	$2N(3M+2L)$	$2S\cdot M^{L}(3M+2L)$
	# of comparison	$M^{L}(M-1)$	$M^{\mu}(M-1)$	$M.N\log_2(MN)$	$M^{L}\cdot S\cdot M\log_2(SM)$
Channel Update (RLS)	# of real addition	$M^{L}(6L^2+20L+13)$	$M^{\mu}(6L^2+20L+13)$	$N(6L^2+20L+13)$	$S\cdot M^{L}(6L^2+20L+13)$
	# of real multiplication	$M^{L}(7L^2+27L+20)$	$M^{\mu}(7L^2+27L+20)$	$N(7L^2+27L+20)$	$S\cdot M^{L}(7L^2+27L+20)$

Table 3.2: Number of real operations (per received symbol) for different adaptive algorithms.

A close observation of Table 3.2 reveals that the MA-PSP requires more computations than the VA-PSP or DDFSE-PSP even when $N = M^L$ or $N = M^\mu$, respectively. This is mainly because of the brute-force nature of the comparison that the MA-PSP does among all possible extensions. On the other hand, the VA-PSP and DDFSE-PSP exploit the trellis structure to optimize the number of comparisons. In Table 3.3 we present the number of survivors in MA-PSP, N, for an equivalent complexity for QPSK.

Channel Order, L	**PSP based Algorithms**	**Flops and Comparisons (per symbol)**	**Number of survivors (N) for MA-PSP for equivalent complexity**
2	MS-PSP, S = 3	12,640	45
	MS-PSP, S = 2	8,352	30
	VA-PSP	4,032	15
	DDFSE-PSP, $\mu = 1$	1,008	4
3	MS-PSP, S = 3	73,600	181
	MS-PSP, S = 2	48,768	120
	VA-PSP	23,808	59
	DDFSE-PSP, $\mu = 2$	5,952	15
4	MS-PSP, S = 3	4,06,530	725
	MS-PSP, S = 2	2,69,824	483
	VA-PSP	1,32,608	239
	DDFSE-PSP, $\mu = 3$	33,152	61
	DDFSE-PSP, $\mu = 2$	8,288	15

Table 3.3: Number of survivors required in the MA-PSP for equivalent complexity with $M = 4$.

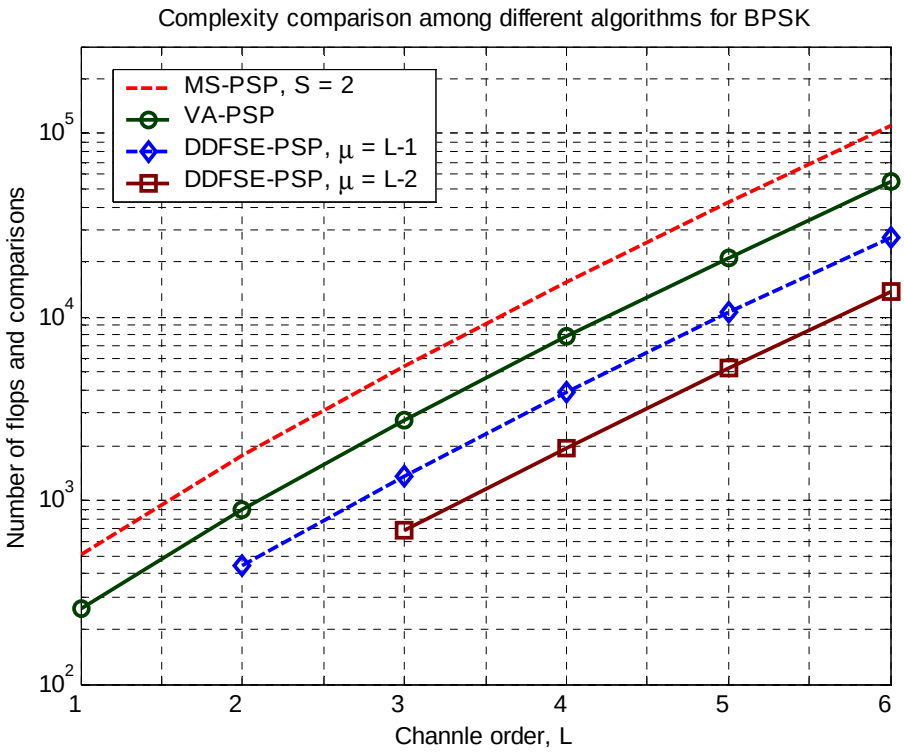


Figure 3.11: Complexity (per symbol) as a function of channel order for BPSK.

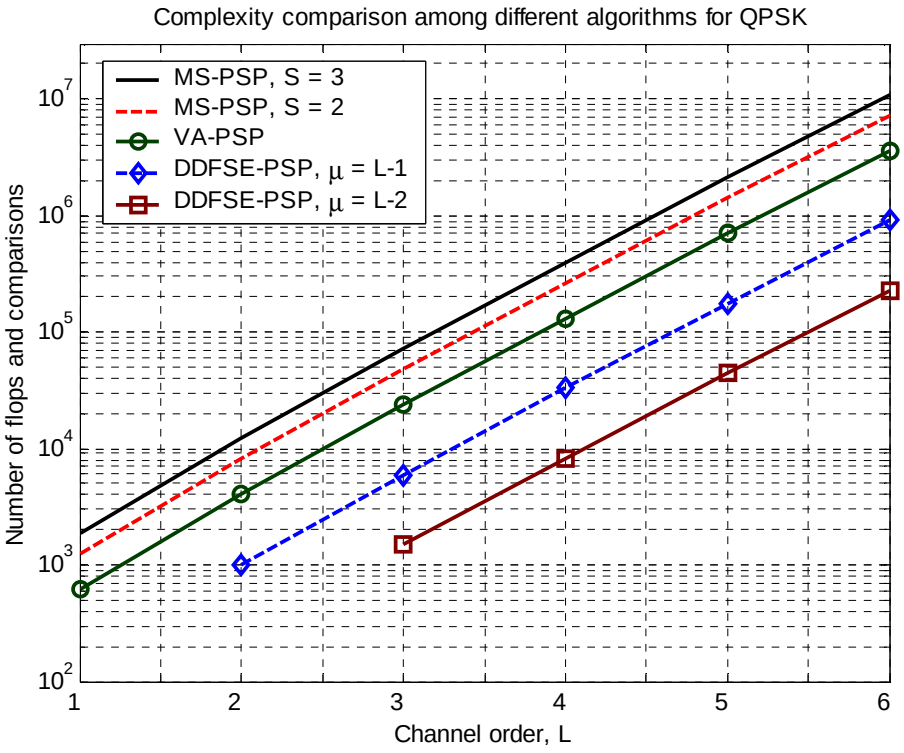


Figure 3.12: Complexity (per symbol) as a function of channel order for QPSK.

Table 3.4: shows the required memory size for various algorithms for metric update and data decoding. Note that the amount of memory is mainly a function of the number of states or surviving paths retained in the respective algorithm. PSP based algorithms require more memory compared to their known channel versions (VA, DDFSE, etc.) to store per-state channel estimates.

Stored Variables	**VA**	**DDFSE**	**M-Algorithm**	**VA-PSP**	**DDFSE-PSP**	**MA-PSP**
Estimated Symbols (Survivors)	$M^L \cdot D$	$M^\mu \cdot D$	$N \cdot D$	$M^L \cdot D$	$M^\mu \cdot D$	$N \cdot D$
Channel Estimates	$L+1$	$L+1$	$L+1$	$M^L(L+1)$	$M^\mu(L+1)$	$N(L+1)$
Cost Metrics	$2M^L$	$2M^\mu$	$N+N \cdot M$	$2M^L$	$2M^\mu$	$N+N \cdot M$
Candidate Symbols (State table)	$M^L \cdot L$	$M^\mu \cdot \mu + 2M^\mu(L-\mu)$	$2N \cdot L+M$	$M^L \cdot L$	$M^\mu \cdot \mu + 2M^\mu(L-\mu)$	$2N \cdot L+M$
Candidate Signal Component	$2M^L$	$3M^\mu$	$N+M$	$2M^L$	$3M^\mu$	$N+M$

Table 3.4: Memory requirement for various algorithms for metric update and data decoding.

3.4 Conclusion

In this Chapter, blind adaptive algorithms have been discussed with emphasis on Per-Survivor Processing (PSP). Since the unknown channel Viterbi Algorithm (VA) is not optimal, several modifications of the conventional VA have been introduced. More complex tree search algorithms like Multiple-Survivor PSP have been presented. Also list-based algorithms, i.e. the M-Algorithm and the T-Algorithm based PSP, have been

discussed. The last section presents a detailed computational analysis. Since Generalized VA (GVA) is too complex for long channels, reduced-state algorithm like DDFSE-based PSP has been considered. The next chapter presents sequence properties and their effect on the acquisition performance of PSP.

Chapter 4: Blind Acquisition Properties of PSP

There are two distinct modes of operation for a joint data and channel estimator, *acquisition* and *tracking***.** The acquisition mode is when the channel is completely unknown or the receiver has no information about either the channel or the data sequence. In this case, it is meaningful to speak in terms of a time-dependent Symbol Error Rate (SER). The process of obtaining the initial estimate of the unknown channel parameters (i.e., channel coefficients, tap delays) is called *acquisition*. However, in the *tracking* mode, the estimates of channel parameters maintained by the receiver are close to the true channel. Unlike acquisition, the algorithm is in a steady state and the symbol error rate is not necessarily time dependent. To summarize, we have two scenarios of interest:

- The acquisition mode with a relatively stationary channel and no apriori information about the channel.
- The tracking mode with a time-varying channel and a reasonable initial estimate of the channel.

Most of the authors in the literature have concentrated on the tracking mode. This may be due to the existing packet formats of GSM, EDGE, and IS-54/136. All of these standards contain an adequate *training sequence* which is used to obtain the initial estimates of the channel (Figure 4.1). The tracking performance of PSP has been discussed in [34]-[36]. Some authors have already reported the tracking performance of other joint data and channel estimation algorithms [26], [55]. However, to the author's knowledge, very little has been done for completely blind channel acquisition [67]. The

potential of PSP and other recursive algorithms, in terms of blind channel acquisition has been discussed in [48], [55]. In most cases, the bit error rate (BER) was computed after some "convergence period." In [67], the blind acquisition performance of PSP with no convergence period has been noted.

In this chapter we consider the *blind short-term* acquisition performance of PSP. By "blind" we mean that there are no training bursts or header/tail symbols available. "Short-term" refers to the time period during the acquisition mode, with no convergence period allowed for the algorithm to settle to a steady state. In the case of *completely blind acquisition*, the receiver does not have any knowledge of the modulation format or length of the channel. Even though completely blind acquisition may be useful only for limited applications (i.e. interception, eavesdropping), we concentrate on this mode to test the joint data and channel estimation in the most severe conditions. Any additional information (i.e. length/main tap of the channel) could improve the performance. In this research, we assume that only the signal type and the length of the channel[1] are known at the receiver.

Blind acquisition and data detection have several advantages over conventional techniques where the channel coefficients are estimated using a training sequence. For example, in GSM, a training sequence is sent with each data packet consisting of 148 bits. Of these bits, 6 are header/tail bits, 26 are used as training sequence, and the rest of the 116 bits are "information" bits. Therefore, the training sequence occupies valuable bandwidth. In the case of blind acquisition, we can shorten each packet by more than 21%, which should be enough to accommodate an extra user in an eight user TDMA frame. Apart from increased throughput, blind acquisition also yields insight into other more complex problems such as Multi-User Detection (MUD) [68]. Also, blind technique

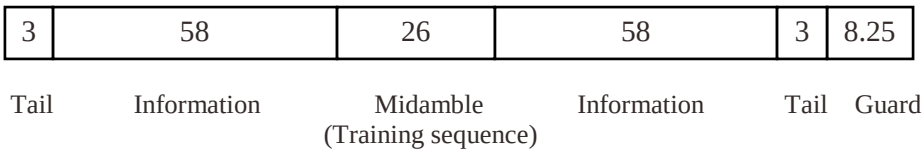


Figure 4.1: Slot structure of EDGE system.

[1] The knowledge of the length of the channel may be relaxed, and has been discussed in section 5.7.

helps prevent *eavesdropping*.

In this chapter, first the *distinguishability* of sequences will be defined. Then the impact of certain sequences on the performance of the joint Maximum Likelihood (ML) channel and data estimator will be presented. Such sequences will be analytically characterized. Also, the acquisition performance and problems of PSP will be discussed. Later in the chapter, the modeling of transients and other relevant problems will be addressed.

4.1 Distinguishability of Sequence

As already explained in Section 3.2, the joint Maximum Likelihood (ML) estimate of the data and channel is obtained by minimizing the metric according to

$$\Lambda\left(\hat{\boldsymbol{\mathcal{S}}}_k\right) = \underset{\hat{\boldsymbol{\mathcal{S}}}_k \in \mathrm{T}_{\mathcal{A}}}{\arg\min} \left\| \hat{\mathbf{Q}}_k \mathbf{r}_k \right\|^2 \tag{3.47}$$

where $\mathbf{r}_k$ is the received vector, $\hat{\boldsymbol{\mathcal{S}}}_k$ is the toeplitz data matrix[2] drawn from the set of all possible matrices, $\mathbb{T}_{\mathcal{A}}$, and generated by the alphabet, $\mathcal{A}$, and $\hat{\mathbf{Q}}_k$ is the projection onto the orthogonal complement of the range space of $\hat{\boldsymbol{\mathcal{S}}}_k$.

In PSP, the transition metric depends on the symbol sequence history through the associated channel estimates. Intuitively, the range space of the toeplitz data matrix $\boldsymbol{\mathcal{S}}_k$, $\mathcal{R}\left(\boldsymbol{\mathcal{S}}_k\right)$, determines the transition metric. In [67], Chugg defined a few properties of this data matrix, which we will now discuss to illustrate how certain sequences affect the short-term acquisition performance of PSP. These definitions will help us diagnose and improve upon some of PSP's limitations. A remarkable result is the fact that many sequences pose problems for PSP regardless of the channel.

---

[2] Toeplitz data matrix, $\hat{\boldsymbol{\mathcal{S}}}_k$, is defined in (2.6).

Definition 1: Degree of Distinguishability (DoD)

Since the joint ML estimate is characterized by the toeplitz data matrix, we define the Degree of Distinguishability (DoD) in terms of $\hat{\boldsymbol{\mathcal{S}}}_k$ by the following relation [67]

$$D(\hat{\boldsymbol{\mathcal{S}}}_k^{(1)}, \hat{\boldsymbol{\mathcal{S}}}_k^{(2)}) = \min\left[rank\left(\hat{\boldsymbol{\mathcal{S}}}_k^{(1)}\right), rank\left(\hat{\boldsymbol{\mathcal{S}}}_k^{(2)}\right)\right] - \dim\left[\mathcal{R}\left(\hat{\boldsymbol{\mathcal{S}}}_k^{(1)}\right) \bigcap \mathcal{R}\left(\hat{\boldsymbol{\mathcal{S}}}_k^{(2)}\right)\right] \quad (3.48)$$

Note that DoD is independent of the channel. Therefore, the measure of distinguishability is important since it helps us isolate sequences which are problematic for *all* channels.

Definition 2: Completely Distinguishable Sequence

Two sequences, $\boldsymbol{\mathcal{S}}_k^{(1)}$ and $\boldsymbol{\mathcal{S}}_k^{(2)}$, are completely *distinguishable*, if for *all* channels

$$D(\hat{\boldsymbol{\mathcal{S}}}_k^{(1)}, \hat{\boldsymbol{\mathcal{S}}}_k^{(2)}) = L + 1 \quad (3.49)$$

where L is the channel order.

Complete *distinguishability* is a strict condition, and the sequences which are rank deficient or have common range space won't qualify for that.

Definition 3: Completely Indistinguishable Sequence

Two sequences, $\boldsymbol{\mathcal{S}}_k^{(1)}$ and $\boldsymbol{\mathcal{S}}_k^{(1)}$, are completely *indistinguishable* if

$$D(\hat{\boldsymbol{\mathcal{S}}}_k^{(1)}, \hat{\boldsymbol{\mathcal{S}}}_k^{(2)}) = 0 \quad (3.50)$$

For example, if $\mathcal{R}\left(\boldsymbol{\mathcal{S}}_k^{(1)}\right) \subset \mathcal{R}\left(\boldsymbol{\mathcal{S}}_k^{(2)}\right)$ or $\mathcal{R}\left(\boldsymbol{\mathcal{S}}_k^{(2)}\right) \subset \mathcal{R}\left(\boldsymbol{\mathcal{S}}_k^{(1)}\right)$, the sequences are completely *indistinguishable*. Also if the received signal, $\mathbf{r}_k$, is in the common range space of two sequences, $\mathbf{r}_k \in \mathcal{R}\left(\hat{\boldsymbol{\mathcal{S}}}_k^{(1)}\right) \bigcap \mathcal{R}\left(\hat{\boldsymbol{\mathcal{S}}}_k^{(2)}\right)$, the sequences are completely *indistinguishable*. Most common completely *indistinguishable* sequences have non-trivial *equivalence classes* which will be explained in the next section. This type of *indistinguishability* is global, i.e., it does not depend on the channel. Even with an

exhaustive search, the receiver cannot distinguish between two completely indistinguishable sequences.

Definition 4: Nearly Indistinguishable (NI) Sequence

Two sequences, $\boldsymbol{\mathcal{S}}_k^{(1)}$ and $\boldsymbol{\mathcal{S}}_k^{(2)}$, are *nearly indistinguishable* if

$$0 < D(\hat{\boldsymbol{\mathcal{S}}}_k^{(1)}, \hat{\boldsymbol{\mathcal{S}}}_k^{(2)}) << (L+1) \tag{3.51}$$

Often *nearly indistinguishable* sequences are sequences whose time-shifted versions have a low degree of distinguishability. We call them *Shift-Near Indistinguishable* (SNI) sequences. These sequences cause most of the problems for PSP.

Definition 4: Channel Dependent Indistinguishable Sequence

The previous definition characterizes sequences that are problematic for any channel. We will now define indistinguishability depending on the channel. A sequence $\mathbf{s}_1$ is *indistinguishable* from $\mathbf{s}_2$ under channel $\mathbf{h}$, if

$$\hat{\mathbf{Q}}_k^{(2)} \cdot \boldsymbol{\mathcal{S}}_k^{(1)} \cdot \mathbf{h} = 0 \tag{3.52}$$

If under channel $\mathbf{h}$, $\mathbf{s}_1$ is *indistinguishable* from $\mathbf{s}_2$, and $\mathbf{s}_2$ is *indistinguishable* from $\mathbf{s}_1$, then the two sequences are *mutually indistinguishable* under $\mathbf{h}$.

Definition 5: Equivalent Sequence

Two sequences $\mathbf{s}_1$ and $\mathbf{s}_2$ are *equivalent* ($\mathbf{s}_1 \equiv \mathbf{s}_2$), if their toeplitz matrix representations have the same range space.

$$\mathcal{R}(\boldsymbol{\mathcal{S}}_k^{(1)}) = \mathcal{R}(\boldsymbol{\mathcal{S}}_k^{(2)}) \tag{3.53}$$

In other words, equivalent sequences are those which have the same joint maximum likelihood metric. For example, if the sequence $\mathbf{s}_1$ is transmitted over the

channel $\mathbf{h}_1$, then the ML receiver cannot distinguish between $(\mathbf{s}_1, \mathbf{h}_1)$ and the sequence $\mathbf{s}_2$ with its associated ML channel estimate $\mathbf{h}_2$, $(\mathbf{s}_2, \mathbf{h}_2)$, because

$$\mathbf{s}_1 \otimes \mathbf{h}_1 = \mathbf{s}_2 \otimes \mathbf{h}_2 \tag{3.54}$$

where $\otimes$ denotes the convolution operation.

Since $\mathcal{S}_k^{(1)}$ and $\mathcal{S}_k^{(2)}$ both have the same range space, there is always some $\mathbf{h}_2$ for which (3.54) is valid. The *equivalence class* of $\mathcal{S}_k^{(1)}$ is the set of all sequences which are equivalent to $\mathcal{S}_k^{(1)}$. The *equivalence classes* can be further divided into *memoryless equivalence* and *memory equivalence classes* [67], [69]. A *memoryless equivalence class* results from the symbol-wise phase rotation of the signal constellation. For example, $\mathcal{S}_k^{(1)}$ and $\mathcal{S}_k^{(2)}$ are *memoryless equivalent* if

$$\mathcal{S}_k^{(2)} = e^{j\phi} \cdot \mathcal{S}_k^{(1)}; \qquad \phi \in \angle \mathcal{A} \tag{3.55}$$

This type of equivalence is called *trivial equivalence*. On the other hand, members of *memory equivalence classes* share more complicated characteristics. Sequences of an *equivalent class* are always completely *indistinguishable*.

Definition 6: Singular Sequence

If the matrix representation of a sequence is rank-deficient, then the sequence is called *singular* [67]. *Singular* sequences cannot be completely *distinguishable*, since the DoD is always less than $L+1$.

4.2 Phase Ambiguity

It is well known that blind algorithms cannot distinguish the actual phase. If the channel is estimated with an unresolved phase, then the channel is called *identifiable*. If the order-L true channel impulse response is given by $\mathbf{h} = [h_0, h_1, \ldots, h_L]^T$, the PSP-identified

channel impulse response can take the form $\hat{\mathbf{h}} = \underset{\phi \in \angle \mathcal{A}}{e^{j\phi}} \cdot \left[\hat{h}_0, \hat{h}_1, \ldots, \hat{h}_L\right]^T$, where $\mathbf{h}$ and $\hat{\mathbf{h}}$ are very close under normal operating conditions. The term $e^{j\phi}$ models a possible constellational rotation, where ϕ represents the allowable angle of the alphabet constellation. We call this problem the *phase ambiguity*. In the ideal case, $\phi = 0$, so there is no phase ambiguity. Phase ambiguity is also common in other blind algorithms. This fact is intuitive in the Constant Modulus Algorithm (CMA), where the algorithm uses the amplitude but not phase information. However, in PSP although the algorithm uses both phase and amplitude information, the phase ambiguity flares up due to some characteristics of the transmitted sequence. In the language of *equivalence class*, PSP cannot distinguish the actual transmitted sequence from any other member of its *memoryless equivalence class*. With the *memoryless equivalent* sequence transmitted, the estimated sequence and the associated channel estimates are symbol-wise phase rotated versions of the original quantities. Fortunately, the effect of *memoryless equivalent* sequences, i.e. the phase ambiguity, can be eliminated by differential encoding/decoding.

4.3 Performance Metric

Usually for channel estimation problems the squared error between the actual channel and the estimated channel impulse response is used as the performance metric. Therefore,

$$\varepsilon = \left\| \hat{\mathbf{h}} - \mathbf{h} \right\|^2 \tag{3.56}$$

where $\mathbf{h}$ is the true channel and $\hat{\mathbf{h}}$ is the estimated channel vector.

However, the performance metric of (3.56) is not applicable to PSP, because it cannot resolve the phase of the channel. To avoid the phase ambiguity in the identified channel, a new performance metric is proposed, which we call Squared Error With respect to Alphabet (SEWA) and is given by

$$\varepsilon = \min_{\phi \in \angle \mathcal{A}} \left\{ \left\| \hat{\mathbf{h}} - e^{j\phi}\mathbf{h} \right\|^2 \right\} \tag{3.57}$$

where $\mathbf{h}$ is the true channel impulse response vector, $\hat{\mathbf{h}}$ is the identified channel vector, $\mathcal{A}$ is the alphabet, and ϕ is the allowable phase of the constellation.

4.4 Learning Curves

Figure 4.2 shows two examples of PSP's acquisition performance for an arbitrary channel. Here the QPSK alphabet, $\mathcal{A} = \{1, -1, j, -j\}$ was used. The initial channel estimates were set to zero. No header/tail symbols were appended to the frame. Figure 4.2 reveals two common cases that can occur in the blind acquisition mode of operation. With the first frame, acquisition occurs and it takes PSP only about 20 symbols to converge. However, given this particular channel, there are sequences for which PSP does not converge. Not only that, but in [67] it is shown that the PSP is unlikely to converge for any channel if the transmitted frame starts with a *nearly indistinguishable*

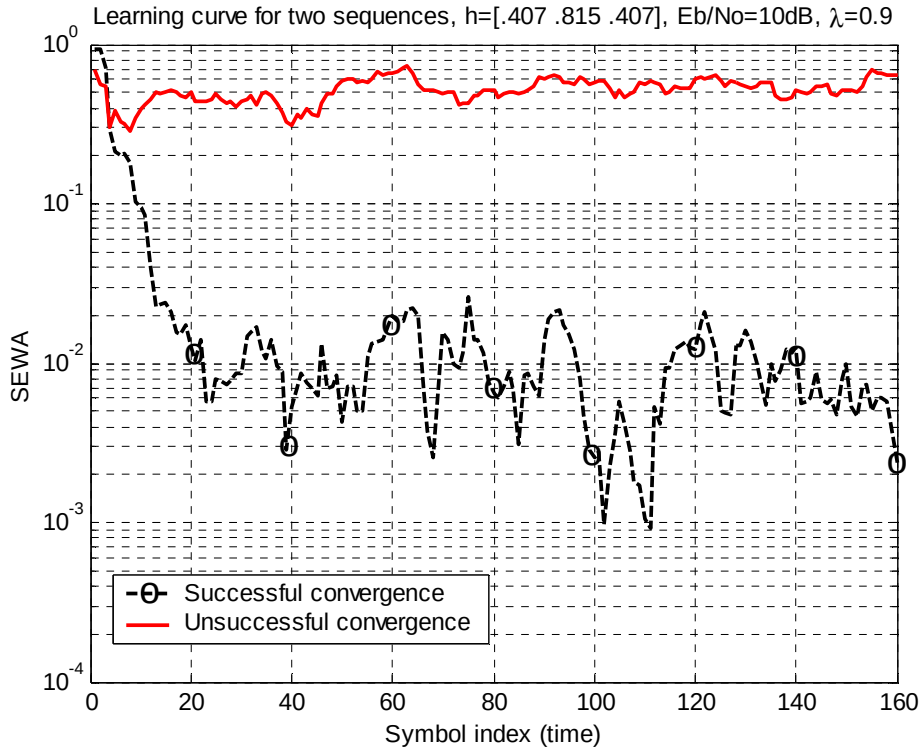


Figure 4.2: Learning curves for PSP.

sequence. The second frame represents the learning curve for such a sequence (solid line). The most common *nearly indistinguishable* sequences are *Shift-Near Indistinguishable* (SNI) sequences. When these types of sequences are transmitted, they tend to drive PSP to lock onto a shifted version of the true channel, which accounts for most of the PSP *misacquisitions*. This is known as the *shifting phenomenon*, which will be discussed in the next section.

Figure 4.3 depicts a comparison between PSP and RLS algorithms. The curves were generated by running the algorithms over 100 frames each consisting of 40 symbols, then the SEWA was averaged over all the 100 frames according to

$$\bar{\varepsilon} = \frac{1}{J}\sum_{j=1}^{J}\left[\min_{\phi \in \angle \mathcal{A}} \left\| \hat{\mathbf{h}} - e^{j\phi}\mathbf{h} \right\|^2\right] \tag{3.58}$$

where J is the number of frames used in the simulation. As before, the channel estimates were initialized to zero for each frame.

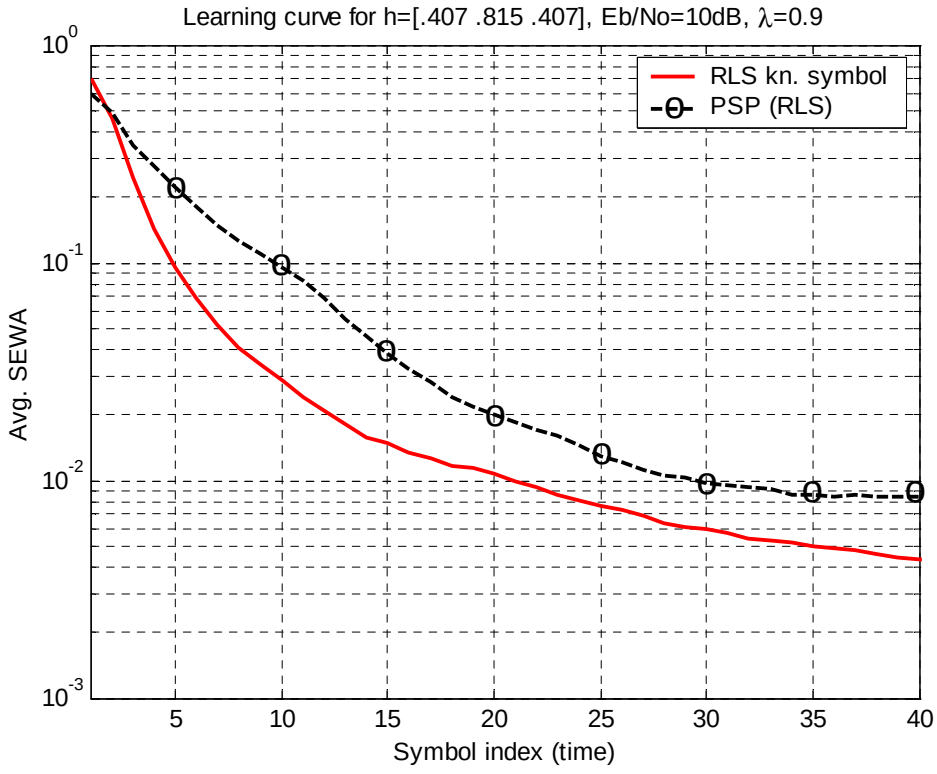


Figure 4.3: Comparison of learning curves for PSP and RLS algorithms.

Note that in the case of successful acquisition,[3] the convergence rate is comparable to that of RLS algorithm with known symbols. This is relatively fast acquisition, considering other blind techniques such as Higher Order Statistics may well take thousand symbol intervals to converge, even at high SNR.

Figure 4.4 illustrates the impact of *misacquisition* on the number of symbol errors in PSP. Note that the histogram has a multimodal distribution, i.e. symbol errors due to *misacquisition* dominate the performance. Another important point to note is that, acquisition performance averaging over many frames could hide misacquisition. Channel estimation performance conditioned on successful/unsuccessful acquisition provides the most insight into PSP's performance.

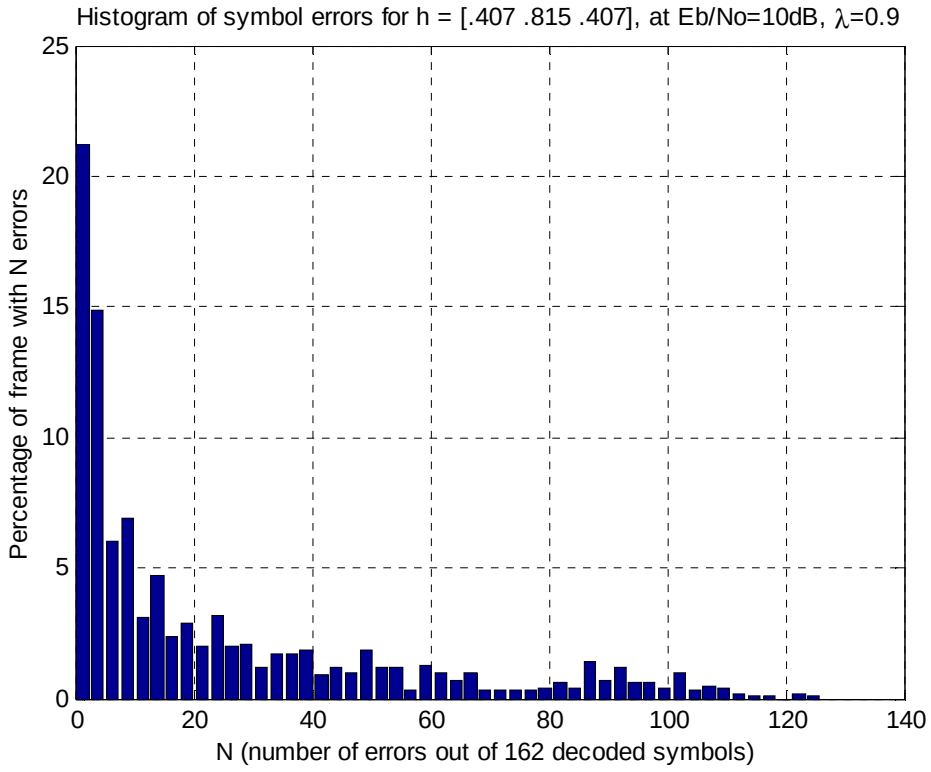


Figure 4.4: The histogram of symbol errors in PSP.

[3] In Figure 4.3, the frames that didn't yield convergence were discarded and weren't included in the calculation of average error for PSP.

4.5 Shifting Phenomenon

PSP is known to exhibit the *shifting phenomenon*, in which the channel estimate resembles a shifted version of the original channel. If the true channel impulse response of order-L is $\mathbf{h}=[h_0,h_1,\ldots,h_L]^T$, the PSP identified channel impulse response could be either right-shift, $\hat{\mathbf{h}}=\underset{\phi\in\angle\mathcal{A}}{e^{j\phi}}\cdot\left[\delta,\hat{h}_0,\ldots,\hat{h}_{L-1}\right]^T$, or left-shift, $\hat{\mathbf{h}}=\underset{\phi\in\angle\mathcal{A}}{e^{j\phi}}\cdot\left[\hat{h}_1,\hat{h}_2,\ldots,\hat{h}_L,\delta\right]^T$, where δ is a very small value and $h_0,h_1,\ldots,h_L$ are very close to $\hat{h}_0,\hat{h}_1\ldots,\hat{h}_L$. The associated sequence estimate is also a shifted version of the correct sequence. The shifting phenomenon occurs when the transmitted sequence is *Shift-Near Indistinguishable* (SNI). In PSP, this fact accounts for most of the misacquisitions. At high SNR, shifting is responsible for about 75% of the misacquisitions (Figure 4.5). The shifting phenomenon has been reported before in [55], [67]. The shifting problem could

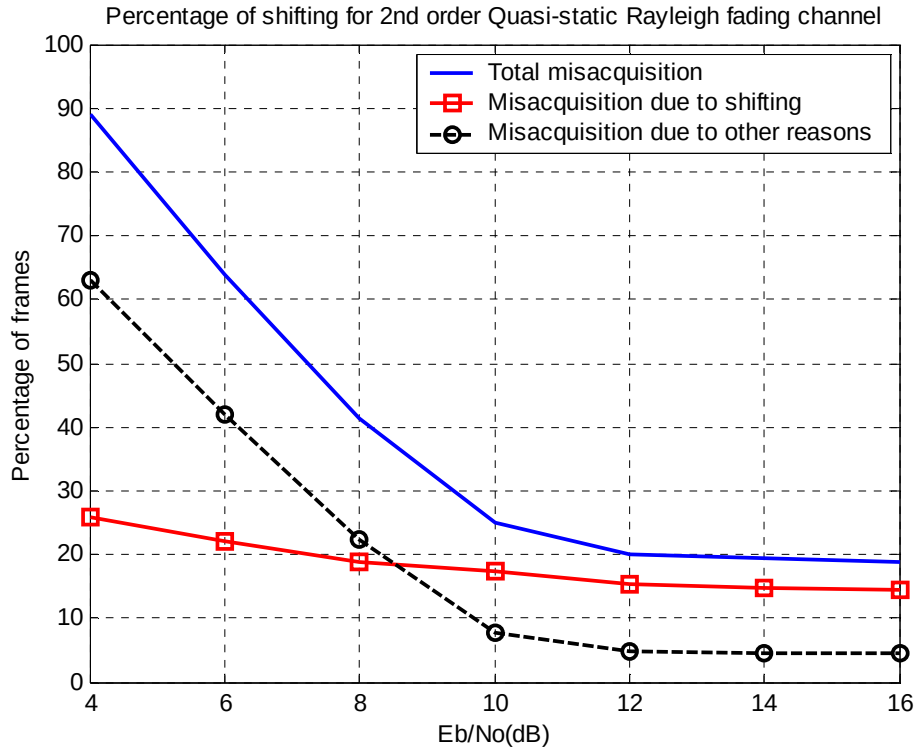


Figure 4.5: Shifting phenomenon is responsible for most of the misacquisitions.

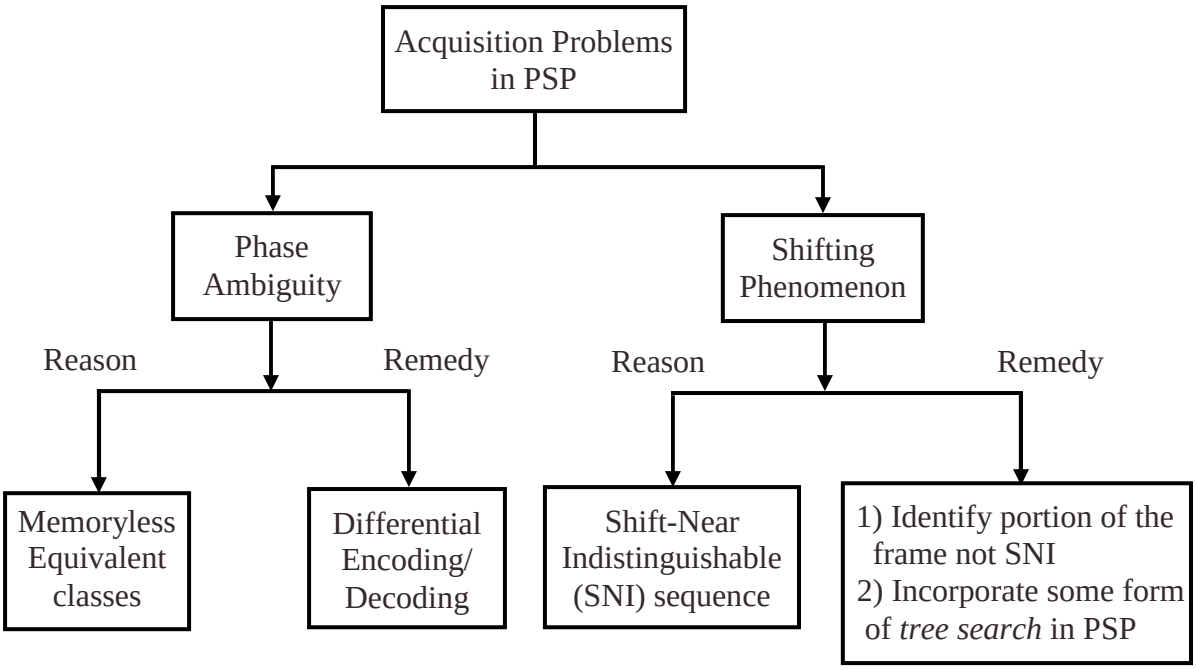


Figure 4.6: Acquisition problems in PSP.

be eliminated either by identifying a portion of the transmitted frame which is not SNI, or by incorporating some form of *tree search algorithm* in the structure of PSP, i.e., Multiple-Survivor PSP. We have discussed some of these techniques in Chapter 3. However, in some applications, i.e., Multi-User Detection (MUD), shifting is not a major concern. In MUD, the auto-correlation of different users is important and not the delay [70]. The acquisition problems in PSP are summarized in Figure 4.6.

4.6 Modeling of Transients

In the Time Division Multiple Access (TDMA) structure, there is some guard time between two adjacent transmitted frames. This guard time accounts for the fact that the channel states are “empty” at both the beginning and the end of each frame. We call the time period over which the channel states are partially or completely empty as *transients*.

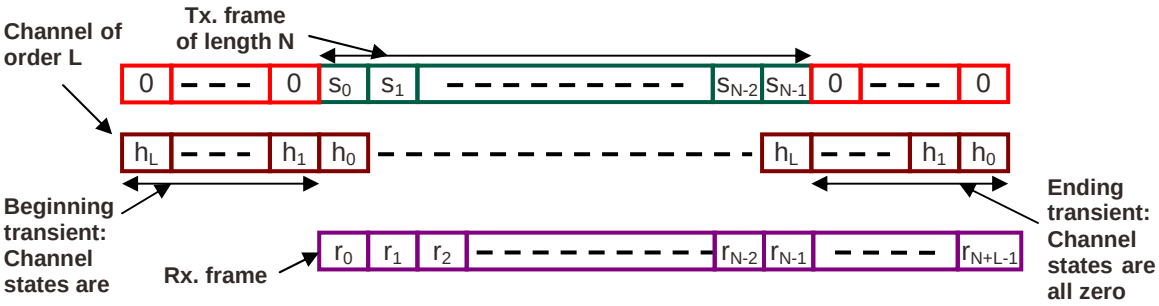
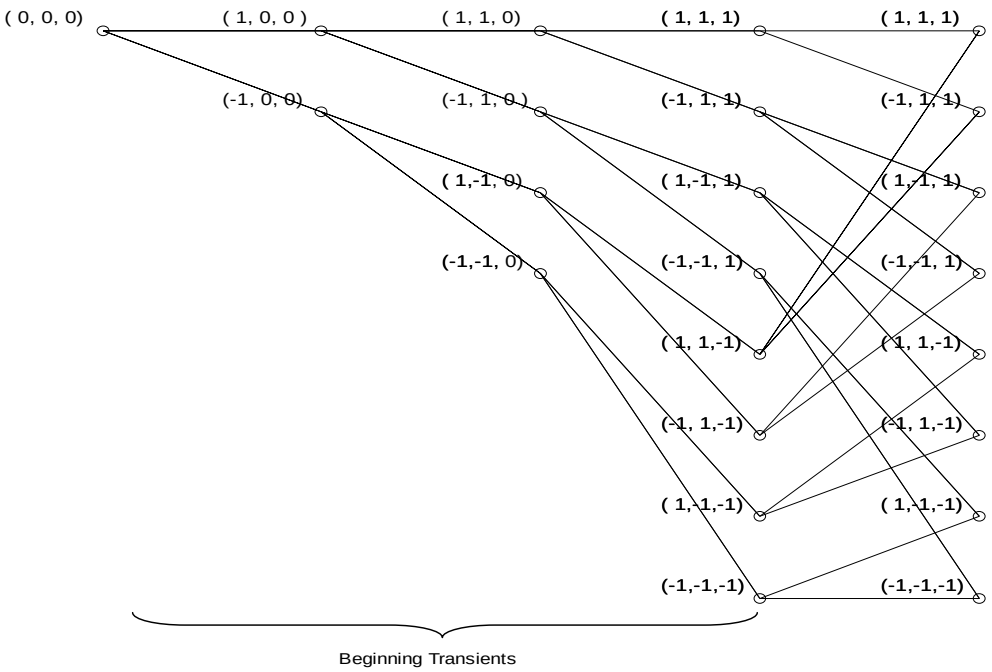


Figure 4.7: Modeling of transients in simulation.

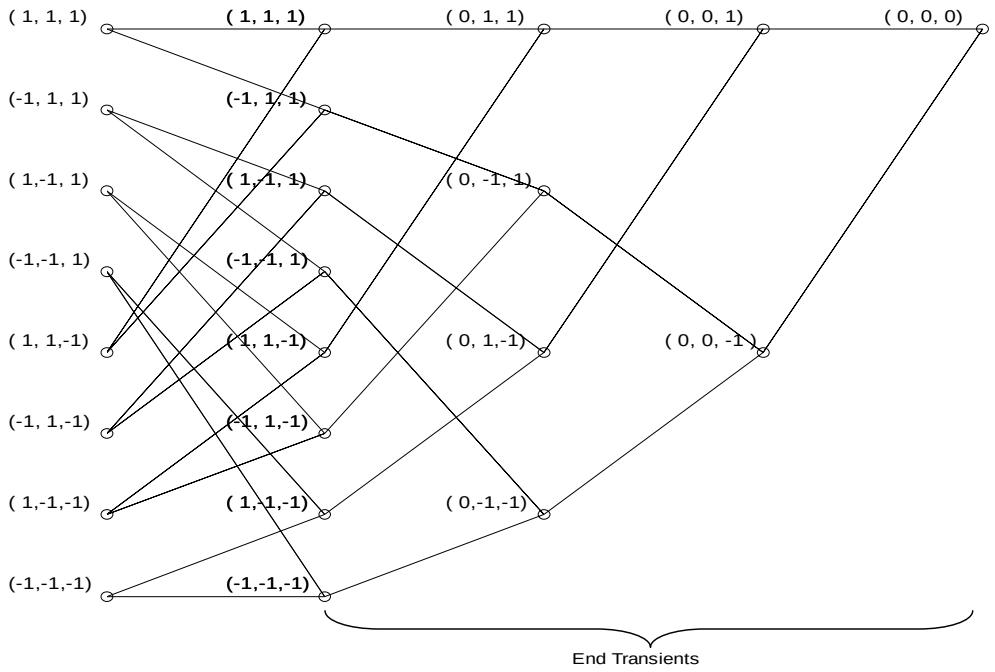
Figure 4.7 illustrates the modeling of transients in our simulation. Filtering is a convolution operation, which is shown here by sliding the channel taps over the transmitted frame. For a channel of order L, we need to append L zeros on either side of the frame to extract information from either edge of the frame. This fact ensures that when the first symbol s_0 fills up the first channel tap, h_0, the other channel taps, $h_1,\ldots,h_L$, are empty. This is also true when the last symbol, s_{N-1}, fills the last tap of the channel, h_L.

Note that the length of the received frame is $(N+L)$ for a length-N transmitted frame if we consider all of the L transient symbols. Transient consideration is important since it contains important information about the received signal. However, with the initial channel taps having no energy (set to zero), we cannot fully exploit this fact in blind PSP.

In blind acquisition, we do not have the luxury of header and tail symbols, appended to the frame for proper initialization and termination of the trellis. Without a termination of the trellis, there is significant performance degradation in the error probability for the last few symbols [71]. Since PSP is VA based, one would expect that PSP would benefit from trellis termination as well. However, if we consider transients, the trellis is terminated even without header and tail symbols. Therefore, the detection of symbols at an end-point is improved. Figure 4.8 describes how a trellis is initialized and terminated to the “all zeros state” when transients are considered. However, one should be careful when considering transients since it could add other problems for PSP if care is



(a)



(b)

Figure 4.8: Trellis initialization and termination by considering transients for $\mathcal{A} = \{1, -1\}, L = 3$; (a) Beginning transients (b) End transients.

not taken in initializing channel estimates. We now discuss transient related problems in the next section.

4.7 Trellis Splitting

Trellis splitting is a phenomenon that occurs in PSP-based algorithms when each state, and all its symbol-wise phase rotated states, in the trellis have an identical metric as well as channel estimates that are a phase rotated version of one another. So effectively, the PSP algorithm splits the trellis by maintaining M sets of survivors and carrying redundant information, where M is the size of the alphabet. As a result, the number of states maintaining unique information becomes $\frac{N_s}{M}$, and the remaining $\left(1-\frac{1}{M}\right)N_s$ states carry redundant information. The redundancy of this search is undesirable because it means that, for the same complexity, PSP will not perform as well as other blind channel estimation schemes.

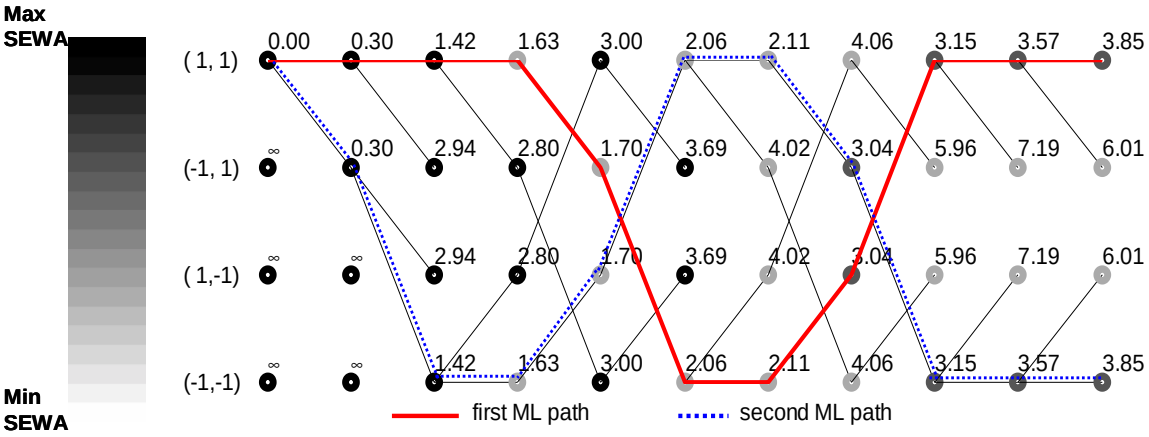


Figure 4.9: Trellis splitting in PSP for $\mathcal{A}=\{1,-1\}, L=2$.

As an example, Trellis splitting occurs in the PSP algorithm, when we consider transients, with all initial channel estimates set to zero. Figure 4.9 illustrates such a

scenario. Here the number and the shading associated with each state represent the corresponding cost metric and channel estimation error, respectively. Note that at each step, each state and its phase rotated states (for example (-1,1) and (1,-1) for a phase of rotation of 180^0 in BPSK) have *exactly* the same cost metric and shading, i.e. SEWA. Also note that, in this example, the state with the least metric has the worst channel estimate, which is not unlikely in PSP if the transmitted sequence is *indistinguishable*. Two possible ML paths exist which are symmetric about the trellis. These are shown by thick solid and dotted lines. The decoded sequences along with these two paths are symbol-wise phase rotated versions of one another. With differential encoding employed at the transmitter, the second ML path becomes redundant.

Since the initial channel estimates are set to zero, each state (transition) has an identical cost metric, which is only a function of the received signal. During the transient, channel states are filled one symbol at a time. These two facts directly lead to the split condition where all of the updated channel estimates for the next step are a phase rotated version of one another. Since these estimates are used for the next step's transitions,

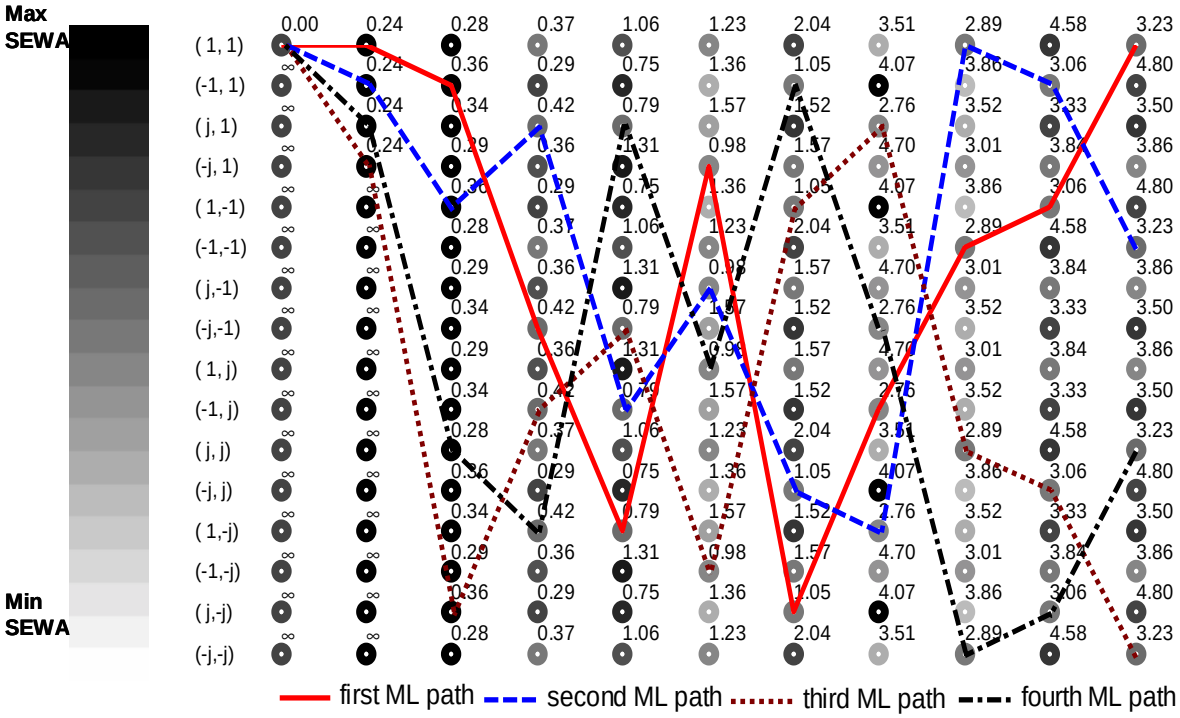


Figure 4.10: Trellis splitting for $\mathcal{A} = \{1, -1, j, -j\}$, $L = 2$.

trellis splitting by induction becomes a cumulative effect. However, careful initialization of channel estimates can avoid this split condition. If we initialize channel taps to a small value rather than all zero, the cost metric at the very beginning of the trellis becomes a function of the state as well as the received signal. This fact destroys the symmetry across the trellis which in turn eliminates trellis splitting.

Trellis splitting has been observed before as an idiosyncrasy of Multiple-State PSP [67].

Figure 4.10 depicts another example of trellis splitting for QPSK. Since $M = 4$ in QPSK, the trellis is split among four surviving paths. For simplicity, only these surviving paths are shown in the trellis.

4.8 Conclusion

In this chapter, joint channel and data estimation algorithm have been described in terms of the toeplitz data matrix. Distinguishability of transmitted sequences has been defined using certain properties of the data matrix. If the transmitted sequence is completely indistinguishable, then it cannot be detected even with the exhaustive search algorithm. The acquisition problems of PSP have been summarized. Trellis initialization and termination using transients have been illustrated. Proper initialization for PSP is important while considering transients, since the all zeros initialization leads to the trellis splitting phenomenon. The next chapter presents proposed techniques to improve on PSP's acquisition performance.

Chapter 5: Improvement Techniques and Results

In the previous Chapter, the performance of PSP in the acquisition mode was demonstrated. In this chapter, modifications to conventional PSP are proposed in order to improve the acquisition rate. The performance of each improvement technique is described along with the simulation results. In the simulation, we assume that the channel is constant over the entire frame, i.e., the coherence time[1] of the channel is greater than the frame time. The signaling format is also assumed to be known at the receiver.

If not mentioned otherwise, the signal to noise ratio (SNR) is assumed to be the ratio of E_b / N_0 experienced at the receiver according to [72]

$$E_b / N_0 (dB) = 10 \log_{10} \left\{ \frac{E\left[|s_k|^2 \right] \sum_{l=0}^{L} |h_l|^2}{2\sigma^2 (\log_2 M)} \right\} \tag{3.59}$$

where σ^2 is the per component variance of noise, s_k is the symbol amplitude, M is the alphabet size, and h is the channel coefficient of order L.

The data symbols used in the simulation are assumed to be independent and identically distributed and drawn from the alphabet, $\mathcal{A} = \{1, -1, j, -j\}$. The modulation format used is Differential QPSK (DQPSK) to encounter the phase ambiguity problem that we discussed in section 4.2. This signal type is close to $\pi/4$-DQPSK, the modulation used by IS-136, which is a 2G US Digital Cellular (USDC) standard. This

[1] Coherence time is the duration of time over which the channel appears "static."

system has a symbol rate of 24.3 ksps and the frame length of each user slot is 162 symbols [5].

The discrete channel impulse response that has been used is the channel in [6, pp. 616, Fig. 10-2-5b]. We refer to this channel as channel A, which exhibits amplitude distortion with linear phase characteristics. It has a spectral null at the band edge (Figure 5.1). Linear equalization of this channel has been shown to be particularly difficult. Many authors in the literature have reported the performance of PSP benchmarked against this canonical channel. Therefore, we included it in our simulation. The second channel that has been considered is a second order "Quasi-static Frequency selective Rayleigh fading channel" referred to as channel B. By "Quasi-static Rayleigh fading" we mean that the channel taps are constant over the entire frame length but are randomly generated (complex number) for each and every frame. This fact has the underlying assumption that the frame time is less than the *coherence time* of the channel, which is valid for many wireless communication systems. The third channel is a fourth order Quasi- static Rayleigh fading channel, which we refer to as channel C.

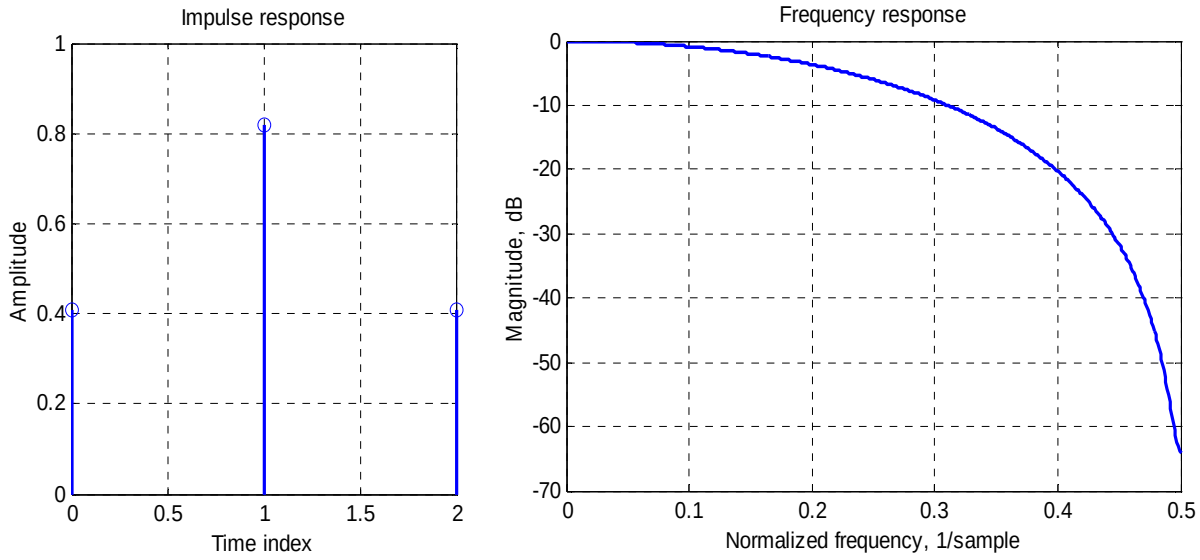


Figure 5.1: Channel A characteristics: Impulse response (left), frequency response (right).

The most important figure of merit for PSP is the acquisition rate. To estimate acquisition rate, the simulations were run as follows:

- Each frame consists of 162 symbols.
- One transmitted frame per trial.
- A trial is considered successful if SEWA < 2%.

For a particular SNR, all simulations were run until either 200 unsuccessful acquisitions were observed or 2,000 trials were simulated, whichever occurred first.

Unless otherwise mentioned, parameters used in the simulation are as follows:

Simulation Parameters	**Values**
Modulation	DQPSK
Frame Length	162 Symbols (IS-136)
Header/Tail Symbols	None
Adaptive Algorithm	Recursive Least Square (RLS)
Initial Channel Estimates	All zero or very small values
Initial Estimate of Correlation Matrix inverse	Identity Matrix
Transients	Yes
Successful Acquisition	SEWA < 0.02

Table 5.1: Parameters used in the simulation.

5.1 Exploiting the RLS in a PSP framework

In the LMS algorithm, the convergence properties are controlled by the step size parameter, β, which must be chosen as a compromise between the stability of the algorithm and the convergence time. In contrast, in the more complex RLS algorithm, the adaptation of each channel coefficient is controlled by one of the elements of the L

dimensional Kalman gain vector, so each tap can be updated independently. This is why RLS often exhibits an order of magnitude faster convergence, as well as better acquisition performance compared to LMS. We used RLS in our simulation to exploit these characteristics.

The trade-off parameter in the RLS algorithm is the forgetting factor, λ, which in turn controls the Kalman gain vector. The forgetting factor $(0 \le \lambda \le 1)$ has the function of limiting the memory of the algorithm. At moderate SNRs, small λ yields weighted Least-Square (LS) estimates with a short effective integration time. In the PSP algorithm, the *Near Indistinguishable* (NI) sequences cause most of the misacquisitions. The transition metric in the conventional PSP ($\lambda = 1$) depends on the entire transmitted sequence through the recursive channel update. Thus if PSP encounters a *near indistinguishable* sequence early in its adaptation, because of its infinite memory, it may never recover, and misacquisition will occur. By varying the forgetting factor, we can control the length of the transmitted sequence over which the successful acquisition depends, and the acquisition

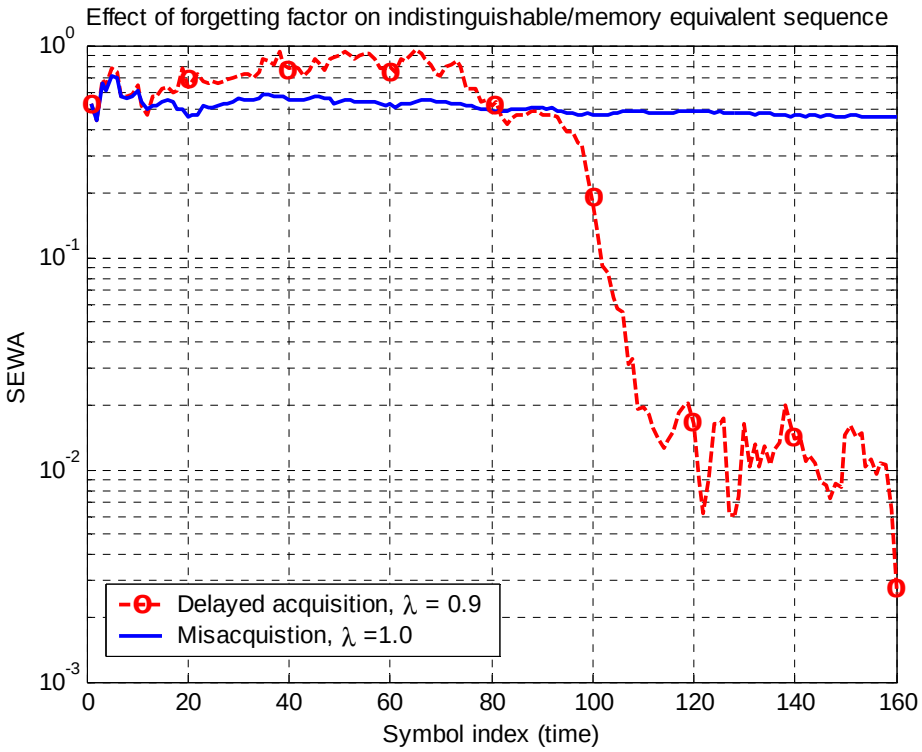


Figure 5.2: Effect of forgetting factor on indistinguishable sequence.

performance can be improved. In other words, $\lambda < 1$ helps the algorithm forget "bad memory," thus enabling it to acquire the channel even after initial misacquisition.

Figure 5.2 illustrates the effect of the forgetting factor on channel estimation error for channel A. With $\lambda = 1.0$, the algorithm never converges (solid line), since the transmitted sequence is an *indistinguishable* sequence. By "reducing" the memory of the algorithm ($\lambda = 0.9$), however, the algorithm was able to acquire the channel after initial misacquisition for the same indistinguishable transmitted sequence. This application is much different than that of pilot-aided channel estimation, where $\lambda = 1.0$ is suggested for stationary channels [9].

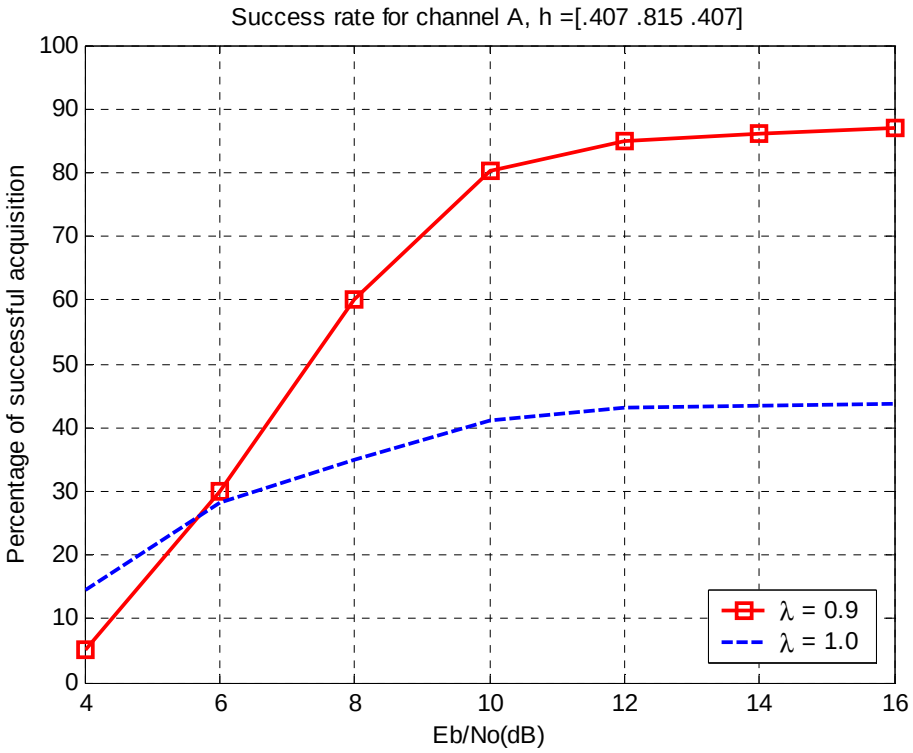


Figure 5.3: Impact of the forgetting factor on acquisition rate.

The effect of the forgetting factor on the acquisition rate is further depicted in Figure 5.3. Recall that successful acquisition is declared if SEWA is less than 0.02. At an SNR of 12 dB, decreasing λ doubles the acquisition rate. Obviously, there is a lower limit on the forgetting factor as far as the stability of the algorithm is concerned.

Heuristically it was found that $\lambda = 0.9$ is the best compromise between convergence rate and stability for a variety of channels. The acquisition rate saturates after 12dB, meaning that the performance is limited by the near *indistinguishable* sequences and not by noise. Also note that, at low SNR values, higher forgetting factors perform better.[1]

5.2 Initialization Issues

Initialization of channel estimates affects PSP's performance significantly. In this section we will discuss how different initializations can help us improve convergence rate. In the first case, *transient* related problems will be presented. The second part will address the instances where we have partial knowledge about the channel.

5.2.1 Channel Transients and Trellis Splitting

The *Trellis Splitting* phenomenon flares up when we consider received *transient* samples, and all channel estimates are initially set to all zeros (see Section 4.7). As a result, the acquisition performance of PSP degrades since the hypothesized sequences retained in the trellis structure maintain redundant information. The question therefore arises as to whether or not it is better to disregard received samples containing transients or modify PSP to avoid trellis splitting. In order to avoid trellis splitting, the initial channel estimates should be set to a small value instead of all zeros. The acquisition performance of these two cases is compared in Figure 5.4. The case, in which no *transients* are considered, is also shown as a reference.

One would think that if trellis splitting is eliminated transient samples would be better, since it is well-known that MLSE with a known channel benefits from the inclusion of transients. However, we have found that from the standpoint of acquisition rate, the case of transients with proper initialization yields about the same performance as

[1] This result is somewhat analogous to the fact that at low values of SNR, LMS (slow convergence) outperforms RLS (fast convergence) in the tracking mode [60], [70].

the case of no transients. The results can be explained as follows; for the case of a transient with channel estimates initialized to small values, the cost metric evolution may be thought of as a small perturbation of the case where no transients are used.

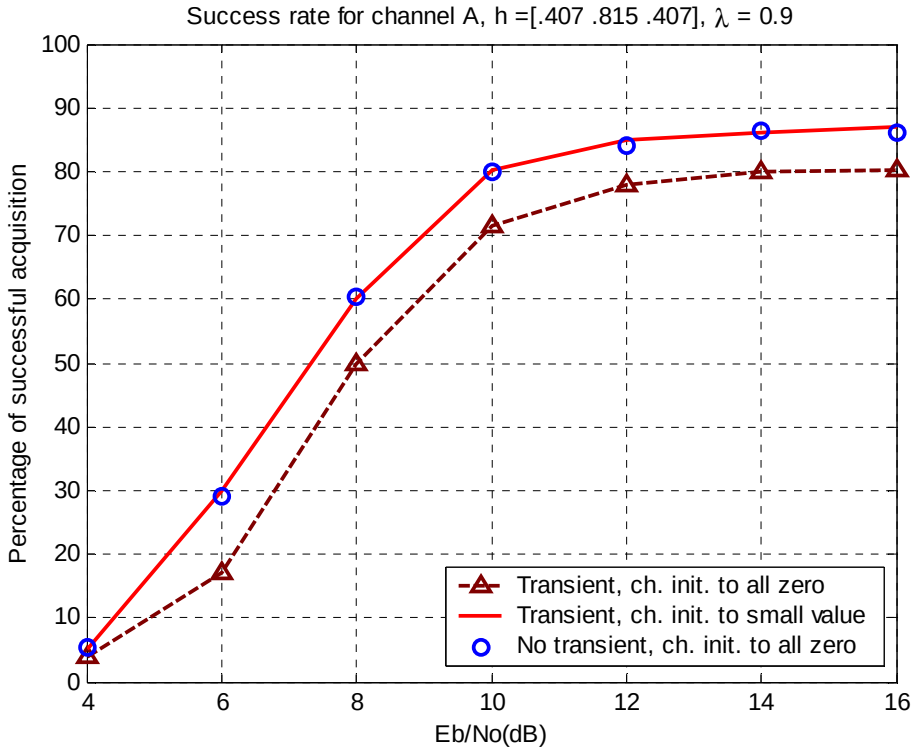


Figure 5.4: Proper initialization can alleviate the trellis splitting problem.

5.2.2 Smart Initialization

So far we have considered only "blind" initialization in our simulations, which means that all the channel estimates are initially set to zero or a small value (to prevent trellis splitting). In this section we will present results which demonstrate that acquisition performance can be improved drastically if we have some limited *apriori* knowledge about the relative phase of the channel. For example, if the channel coefficients are $h = [0.407 \;\; 0.815 \;\; 0.407]$, it is reasonable to initialize the channel estimates as $\hat{h} = [0.0 \;\; 1.0 \;\; 0.0]$. We call this "smart" initialization. With smart initialization, the correct detection of the first few symbols can exploit most of the channel energy.

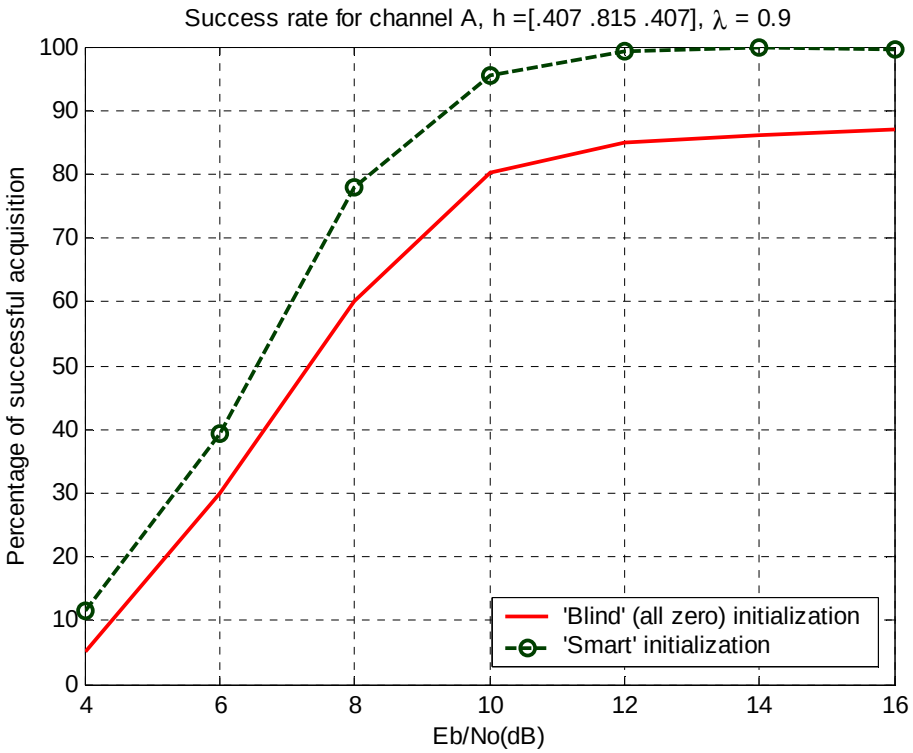


Figure 5.5: Smart initialization improves PSP's acquisition performance.

Therefore, the surviving path has a lower transition cost which easily "isolates" it from other candidate paths. Thus the chance of extracting the potential correct path in favor of a candidate path is greatly reduced.

However, with no initial channel energy (all taps set to zero), the transition cost metrics of all the candidate paths (including the correct survivor) are close, and in a noisy environment the correct survivor could be dropped in favor of other candidates. Figure 5.5 describes the acquisition performance of PSP with smart initialization.

5.3 Improving PSP's Acquisition with Frame Segmentation

In previous sections it was shown that acquisition performance of PSP can be improved by changing the forgetting factor or by exploiting the apriori knowledge of the channel. However, only "smart" initialization achieves an acceptable acquisition rate for

commercial applications. Indeed, in most applications smart initialization is not possible. If apriori information of the channel is not available, other ways to improve PSP's blind acquisition performance should be considered.

As discussed in section 4.5, the main hindrance to the convergence of PSP is the possible presence of *Near Indistinguishable* (NI) sequences. So we attempt to "identify" a portion of the transmitted frame that is conducive to PSP's convergence, even though the frame as a whole may contain multiple NI sequences. In this regard, we segment each received frame into equally smaller portions (segments). The motivation behind the "segmentation" idea is that the transmitted sequence can be *indistinguishable* from other candidate sequences for any given segment of the received frame, but it is highly unlikely that this will happen for all of the segments. Since only one segment is required to estimate the channel, if a segment without an NI sequence can be reliably identified, PSP's acquisition rate can be drastically improved.

Since the cost metric is the sum of errors made at all of the previous stages, the evolution of the cost metric can be used to predict successful acquisition of PSP. Cost metrics are a function of the transmitted sequence, noise, and the length of the frame. Since all the segments are of equal length and the noise is random, it is very much likely that the segment containing a sequence with the highest degree of distinguishability will result in the lowest cost metric. To identify the segment that yields the channel estimates with minimum SEWA, a selection procedure is proposed here. For each segment using *trace-back*, the cost metric trajectory of the ML path is obtained[2]. From the surviving path's cost metric evolution, the growth rate of the cost metric is determined over the last few symbols[3]. We call this the Surviving Metric Estimated Growth Rate (SMEGR). The segment with the minimum SMEGR is the one that converges better than the rest and consequently yields the channel estimate with the minimum SEWA.

---

[2] Like the Survivor table, the cost metric of each state at each stage is stored.

[3] Growth rate is calculated using a two-point slope.

5.3.1 Segmented PSP (Seg-PSP)

The steps of the proposed segmented PSP (Seg-PSP) scheme are summarized as follows:

- Divide a received frame into smaller segments of equal length.
- Run PSP over each segment with channel estimates initialized to small values and zeros for the first segment and the subsequent segments, respectively.
- Calculate the Surviving Metric Estimate Growth Rate (SMEGR) for each segment and store them.
- Select the segment with the minimum SMEGR. We define this segment as the *best segment* and the channel estimates of this segment as the *best channel estimate.* The *best channel estimate* is declared as the final channel estimate.

An important aspect of this scheme is the segment size. The segments should be long enough to accommodate the convergence of the adaptive algorithm, but short enough to cut down on complexity and potential divergence due to NI sequences. After trying with several segment sizes we found that a size of 40 symbols is the best choice. The segments could be overlapping if it is required to derive more segments out of a given received frame (Figure 5.6). Also note that if 0% overlapping is used, segmentation doesn't increase the complexity of PSP. In case of 50% overlapping, complexity is approximately doubled.

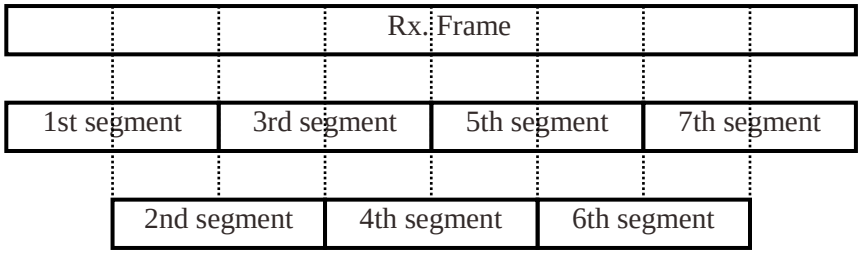


Figure 5.6: Block diagram of overlapping segments (50% overlap).

The initial channel estimates for the first segment should be set to small values to prevent trellis splitting. However, the channel estimates of the subsequent segments can

be set to all zeros since the channel states are already filled with the previously received segment's samples and there are no transients. Also note that one should calculate the SMEGR of the ML path over the last few symbols. In the simulation, SMEGR was determined over the last 15 symbols. This would give the cost metric trajectory enough time to converge or diverge before a decision is made.

Figure 5.7 illustrates the steps taken to implement the segmentation approach. Note that PSP[4] can be run in parallel on all the segments leading to a pipelined approach that makes it attractive in VLSI implementations. In this example, the received frame is divided into four non-overlapping segments each consisting of 40 symbols. After running PSP on each of the segments, the second segment is found to yield the min SMEGR. This is explained in Figure 5.8, where the cost metric trajectory of the second segment has the lowest SMEGR over the last 15 symbols. Also note that the trajectory of the fourth segment's cost metric starts well but suddenly diverges after 10 symbols, which emphasizes the fact that the segments should be long enough to assure that the algorithm will make a reliable choice.

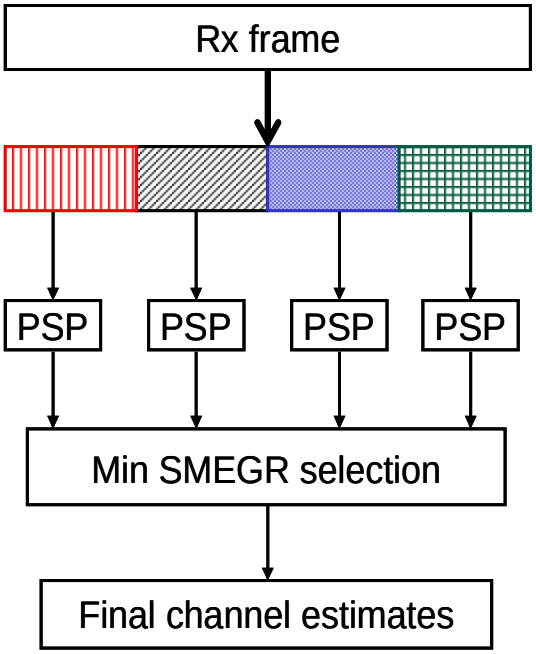


Figure 5.7: Schematic view of Frame segmentation approach.

[4] Note that in this Chapter, PSP and VA-PSP has been used synonymously. Whenever other algorithms are used with the PSP they are mentioned explicitly.

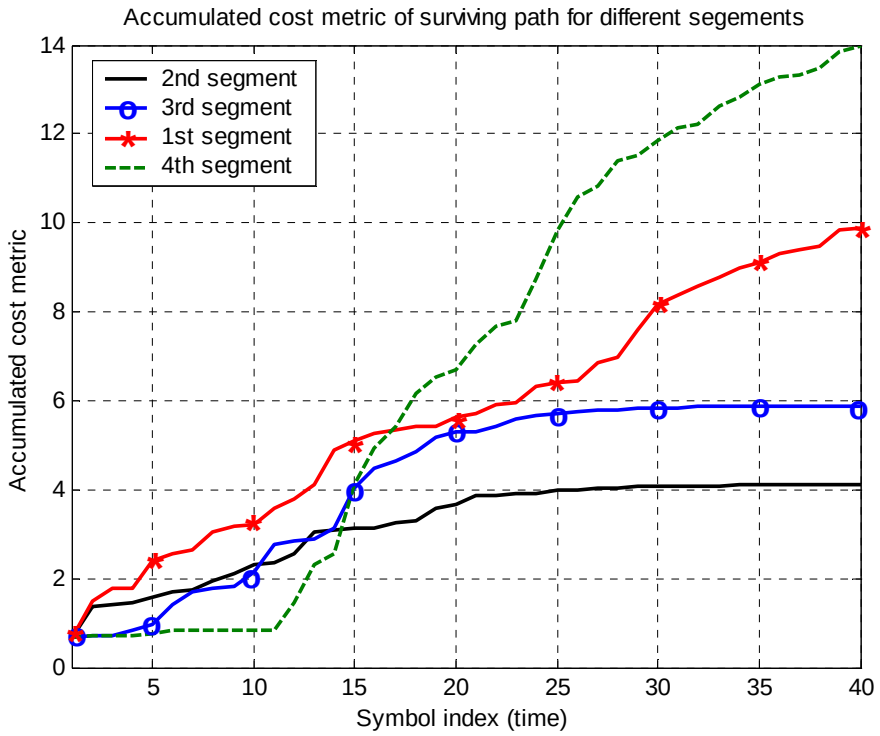


Figure 5.8: Correct convergence can be predicted by the trajectory of the accumulated cost metric.

5.3.2 Iterative Segmented PSP (ISeg-PSP)

Another improvement step that might be appended to the segmentation scheme is the *Iterative* PSP (Figure 5.9). In this approach, once the *best segment* is found, PSP is run over it again with the channel estimates initialized to the *best channel estimate*. *Iterative* PSP might improve performance for two reasons. Firstly, since the length of the *distinguishable* sequence is short (40 symbols), we may not have reliable acquisition if a *delayed* acquisition occurs. Therefore, another run over the same segment will help it clamp onto the true channel. Secondly, since we have a reasonable initial estimate of the channel after the first run, we can use those estimates for the second run for even better acquisition.

However, there are a couple of important factors in the iteration mode which impacts performance. In the iteration process, PSP should not run over the entire frame. In the simulation we encountered cases where, even with the perfect initialization, *Near*

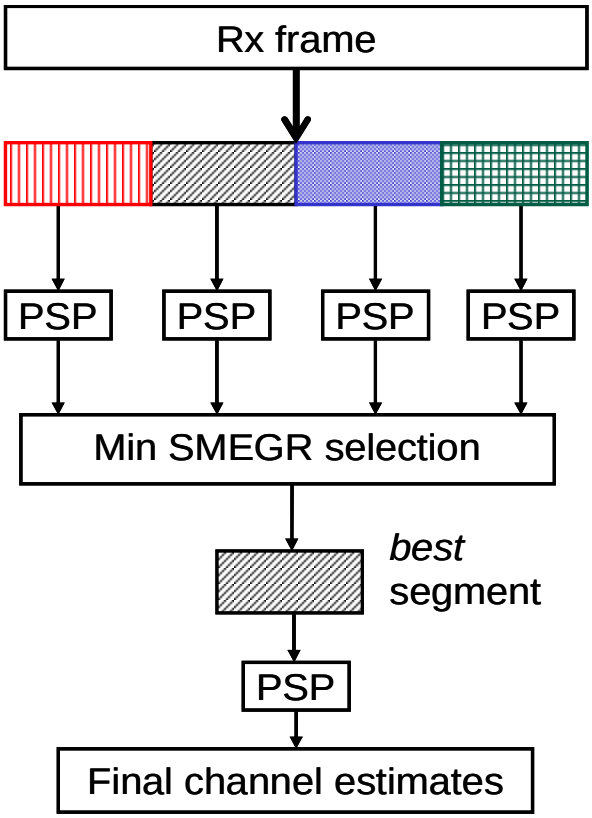


Figure 5.9: Block diagram of *Iterative* PSP with frame segmentation.

Indistinguishable sequences can throw off the acquisition process even in high SNR environments. Therefore, only the *best* (hopefully *distinguishable*) segment should be used for the second iteration. Also, the forgetting factor should be close to one since we already have reasonable initial estimates and don't need large variations in the estimation process. In the simulation we used $\lambda = 0.99$ for the second iteration. In iterative PSP there is practically no performance improvement after the second iteration.

The chief advantage of the segmentation approach is that it can also be used in the *tracking* mode for time-varying channels. Since the main focus of this research is to analyze the performance in the *acquisition* mode, we won't discuss that here. PSP's performance in the tracking mode with time-varying channels have been reported in [34]-[36], [60].

5.3.3 Acquisition Performance of Seg-PSP and ISeg-PSP

The acquisition performance of segmented PSP (Seg-PSP) and *Iterative* Segmented PSP (ISeg-PSP) for Channel A is shown in Figure 5.10. In the simulation, transient symbols were considered, with channel estimates initially set to a small value and a forgetting factor of $\lambda = 0.9$ was used. The segment size was fixed at 40 symbols/segment. Note that there is substantial improvement with the segmentation approach compared to the whole frame. As we use more segments there are more chances that we might be able to find a *distinguishable* sequence, and hence the acquisition performance improves. However, when SNR < 8dB, even after using seven segments, the acquisition rate is less than 80%, meaning that the acquisition performance is dominated by noise and not by the presence of *indistinguishable* sequences. The acquisition performance of *Iterative* PSP is also shown as a reference. Note that, at low SNRs, the performance can be improved drastically with *Iterative* Segmented PSP. However, at high SNRs there is practically no improvement over the segmentation approach.

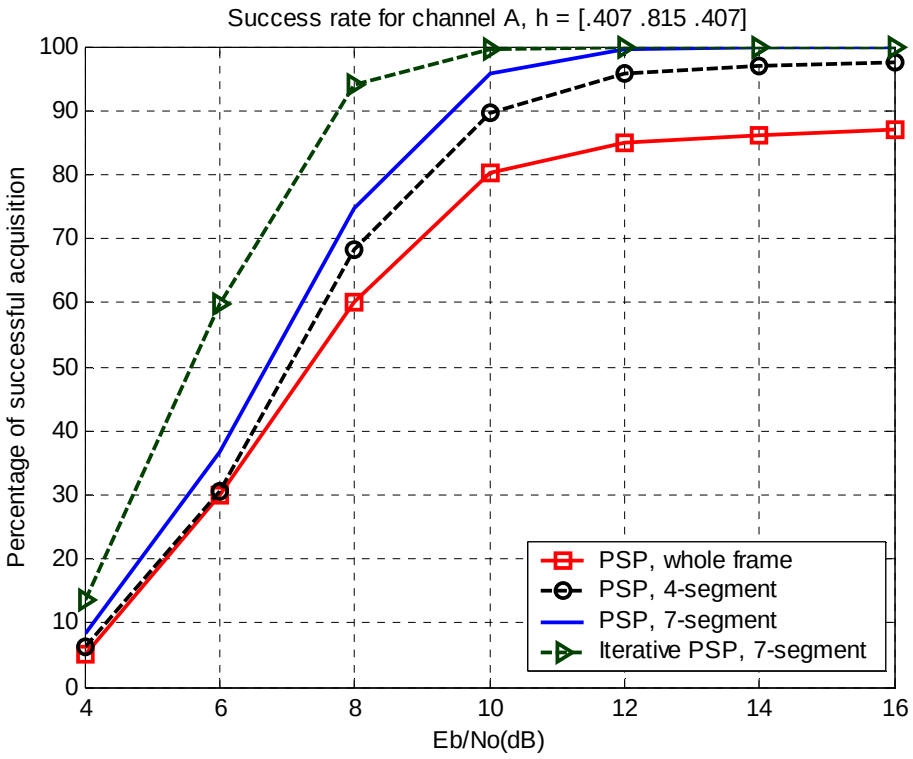


Figure 5.10: Acquisition performance with segmentation for Channel A.

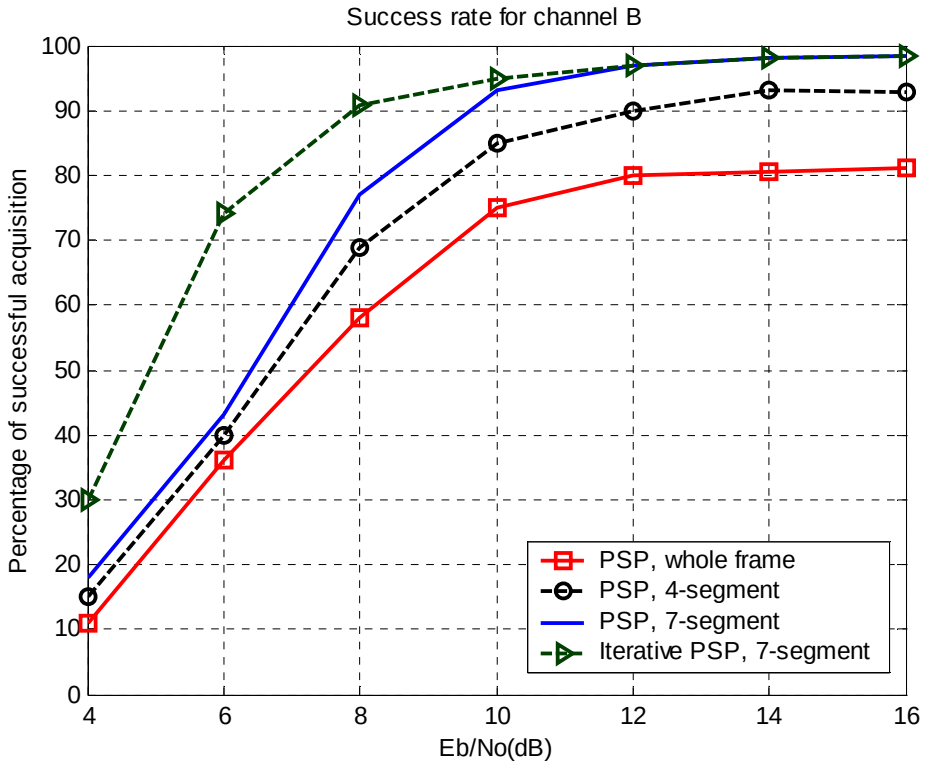


Figure 5.11: Acquisition performance with segmentation for Channel B.

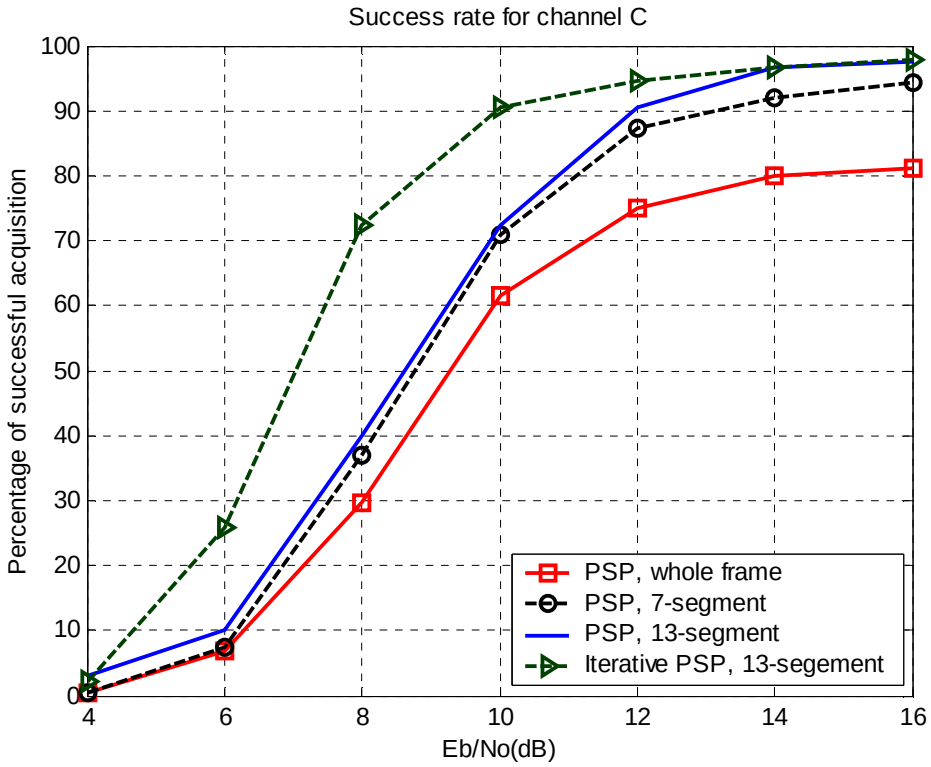


Figure 5.12: Acquisition performance with segmentation for Channel C.

The acquisition performance for the second order Quasi-static Rayleigh fading channel (Channel B) is shown in Figure 5.11. The trend is the same as that for Channel A. The segmentation technique was also tested for higher order channels. Figure 5.12 shows results for fourth order Rayleigh fading channels. Note that the results are worse compared to the second order channels. As the channel order gets higher, the transmitted sequence becomes more likely to be *indistinguishable*, resulting in worse acquisition performance. Therefore, we need more segments for the same acquisition performance. Once again, *iterative* PSP along with segmentation can improve performance at low SNR.

5.3.4 SER Performance

To obtain the Symbol Error Rate (SER) performance, first the received frame was divided into seven segments and each one was processed through the PSP block. The *best segment* and the *best channel estimate* are obtained according to the segmentation scheme described in section 5.3. Then the Viterbi Algorithm (VA) was used to decode the entire received frame with *the best channel estimate* (Figure 5.13).

Figure 5.14 describes the SER performance of PSP with our segmentation approach for Channel A. Note that the numbers associated with the PSP curve is the failure rate at different SNR values, which are derived from the corresponding acquisition rate (Figure 5.10). SER performance for the VA with a known channel is also shown as a reference. Note that at high SNR, the SER curve for PSP is about 1dB off the known channel VA, even though the failure rate is 0%.[5] This apparent anomaly is due to the fact that there is some irreducible error floor in the PSP's convergence. As shown in Figure 4.2, this is around 10^{-2}. Any blind channel estimation algorithm will produce some residual numerical error no matter how well they converge, resulting in a few symbol errors even at high SNR. Also the SER performance is contingent on the successful acquisition, which is declared if SEWA < 0.02.

[5] Note 0% means that no failure was experienced.

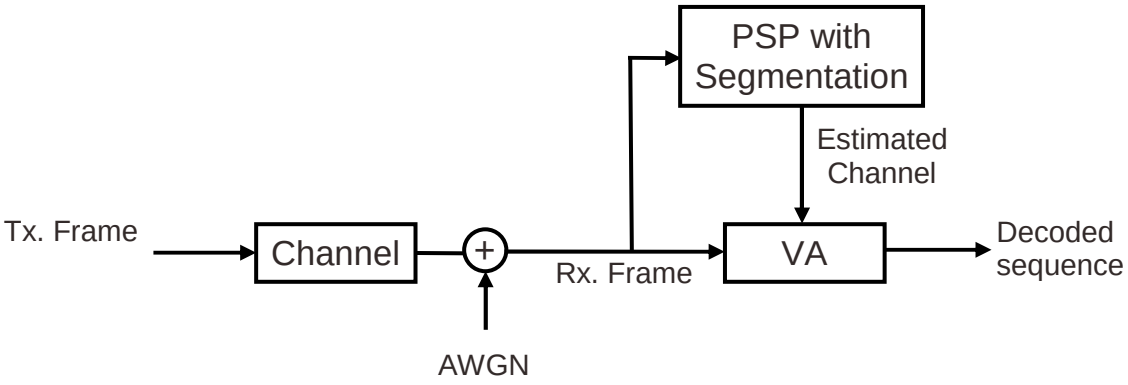


Figure 5.13: Block diagram used in the simulation of SER performance.

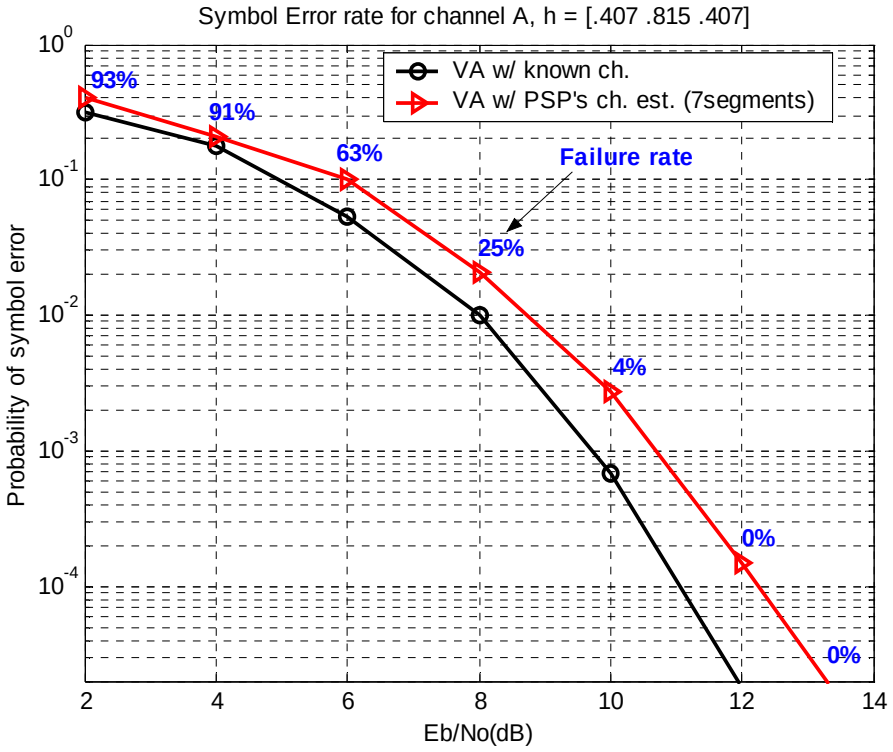


Figure 5.14: SER performance of PSP on the segmented frame for Channel A.

5.4 Acquisition Performance of Generalized PSP (GPSP)

Acquisition performance of PSP can be improved by incorporating some form of *tree search* into the PSP algorithm itself. The idea stems from the fact that in PSP, VA is no longer optimal, and the most likely path may be eliminated early in the search. The

results of the two Generalized PSP (GPSP) algorithms that we discuss here are M-algorithm based PSP (MA-PSP) and Multiple-Survivor PSP (MS-PSP). The simulation results reported in Figure 5.15 and Figure 5.16 are for frames each consisting of 40 symbols. We purposely selected a frame size of 40 symbols so that the results would be directly applicable to our "segmentation" idea. As before, all simulations were run until at least 200 unsuccessful acquisitions were observed at each SNR value or 2,000 total frames were detected. Our simulation results show that the VA-PSP is outperformed by both MS-PSP and MA-PSP for all the cases. Note that for a fair comparison, the number of survivors used in the MA-PSP was picked from Table 3.3. We also observe that for equivalent complexity, the MA-PSP performs better than MS-PSP. This result is expected since the M-algorithm retains the surviving paths which are *globally* best among all candidate paths, as opposed to trellis-based algorithms, i.e. VA, PSP or Multiple-Survivor PSP where only the *local* best paths are retained at each state. Therefore, in

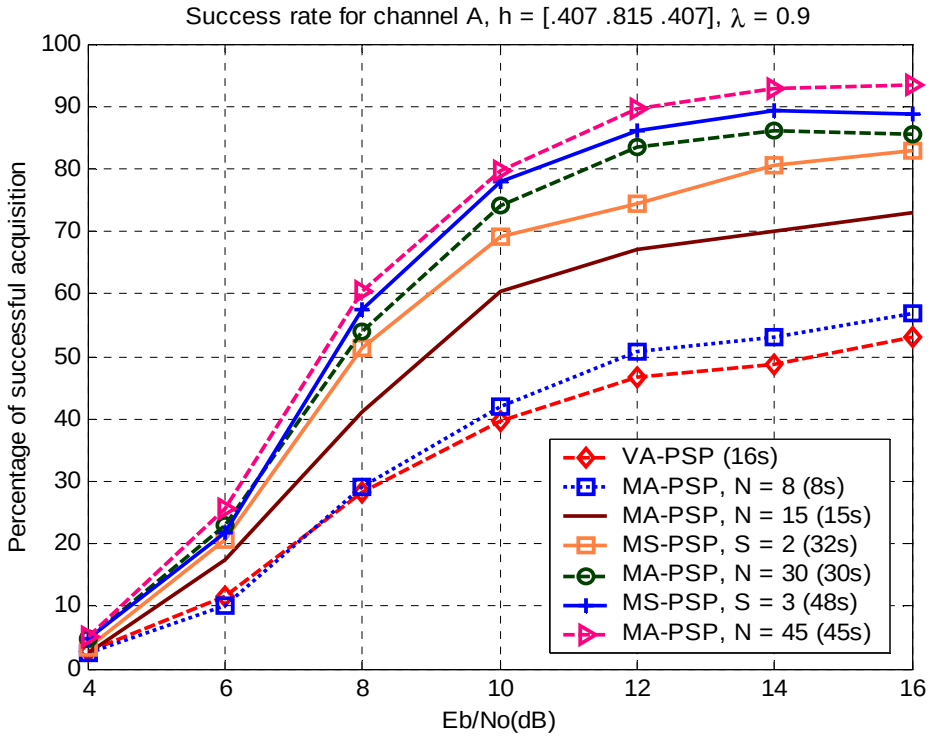


Figure 5.15: Acquisition performance comparison among PSP, MA-PSP, and MS-PSP for channel A.[6]

[6] In Figure 5.15 and Figure 5.16 the total numbers of survivors maintained at each stage are given in the bracket. For example, MS-PSP with S = 2 has 32 survivors per stage, i.e. 2 survivors/state, 16 states.

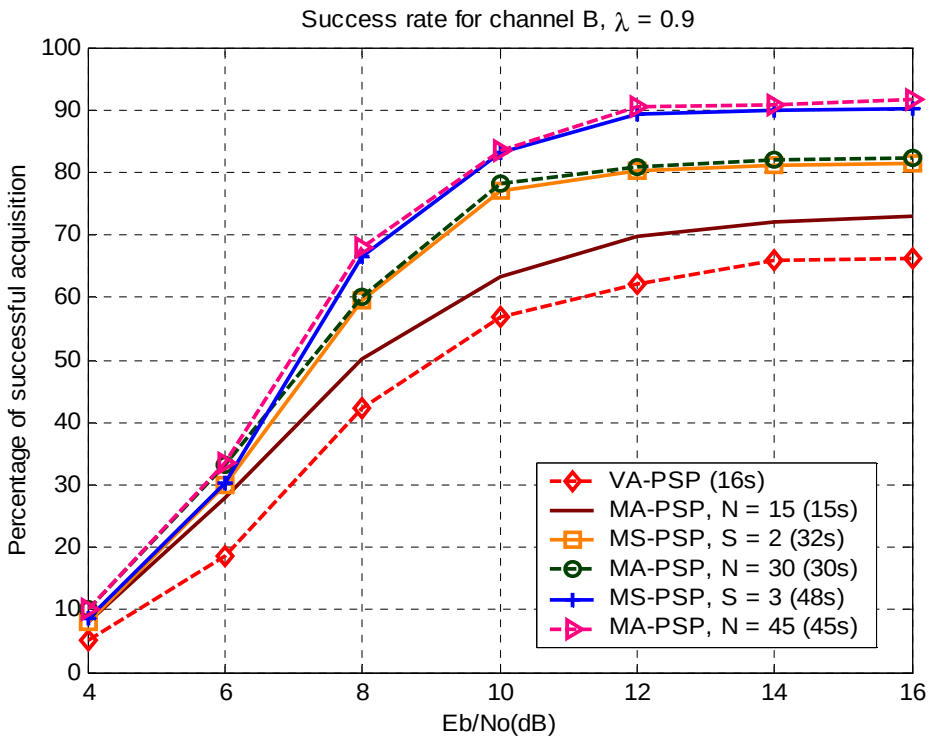


Figure 5.16: Acquisition performance comparison among PSP, MA-PSP, and MS-PSP for channel B.

MS-PSP, some potential surviving paths could be dropped just because it is not the best for any given state, even though it is better than paths retained at other states.

Figure 5.15 describes the acquisition performance for channel A. A better performance of MA-PSP compared to VA-PSP is obvious though both have the same complexity. Even an 8-survivor MA-PSP performs better than VA-PSP with almost half the complexity. A 24-survivor MA-PSP outperforms its counterpart, MS-PSP with S = 2, by a decent margin. With 48-survivor, MS-PSP's performance is effectively that of an *exhaustive search* and no further improvement should be expected by more complex search algorithms. Note that, even with more complex search algorithms, at 16dB SNR the acquisition rate is only about 90%. This fact underlines the impact of completely *indistinguishable* sequences on any joint data and channel estimation algorithm.[7] We also conclude that increasing the number of survivor paths doesn't proportionally improve the

[7] The reader is referred to section 4.1 for a detail analysis of indistinguishable sequences and their properties.

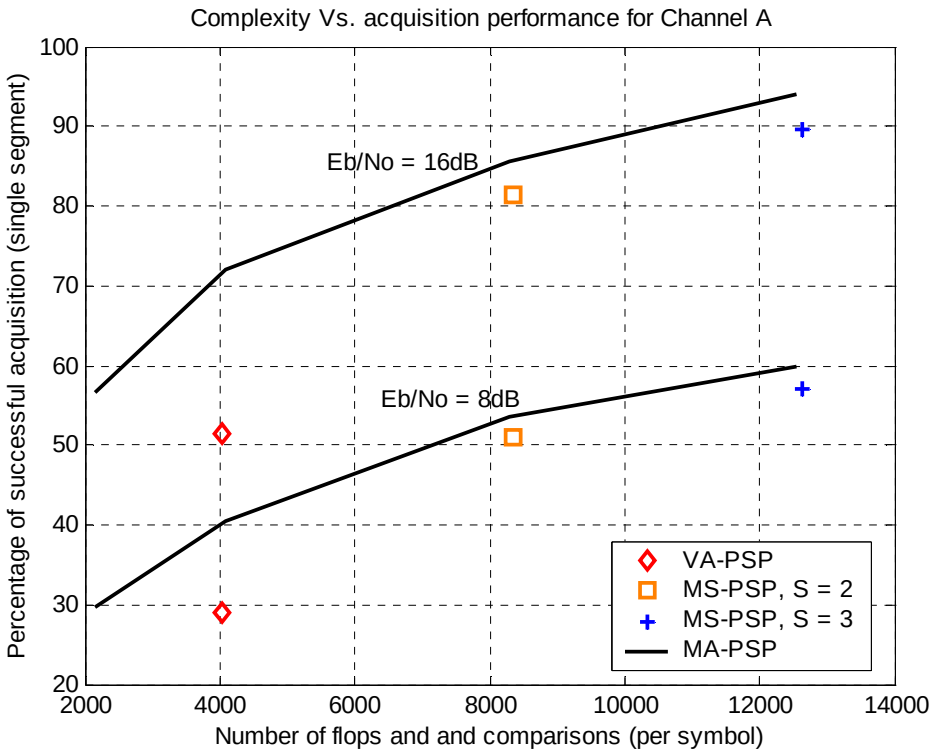


Figure 5.17: Acquisition performance as a function of complexity for channel A.

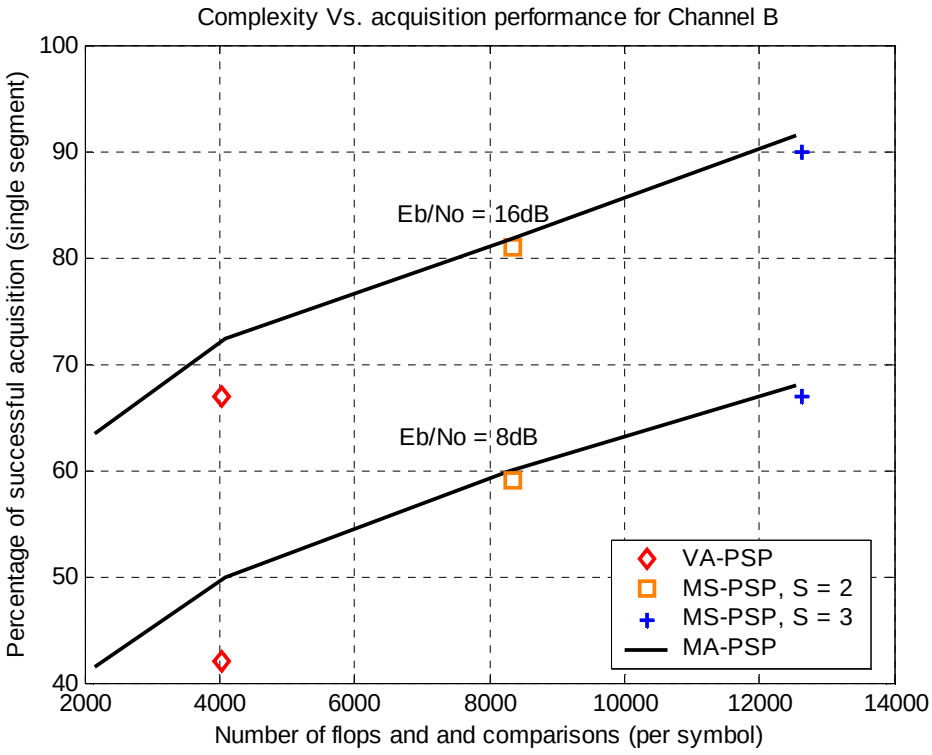


Figure 5.18: Acquisition performance as a function of complexity for channel B.

performance, despite the relative increase in the computational requirement (Figure 3.11 and Figure 3.12). The results for Channel B follow the same trend (Figure 5.16). The results are consistent with the tracking performance of MA-PSP and MS-PSP already reported in [60]. MA-PSP's superiority in the acquisition/complexity trade-off is further illustrated in Figure 5.17 and Figure 5.18 for channel A and channel B, respectively.

5.4.1 MA-PSP with Segmentation

In the previous section we already found that MA-PSP outperforms VA-PSP with an equivalent complexity. This fact motivates a segmented MA-PSP. We found that fewer segments (complexity) are required for MA-PSP for the same acquisition performance. Figure 5.19 shows that for Channel A, a 4-segment MA-PSP performs as good as a 7-segment VA-PSP.

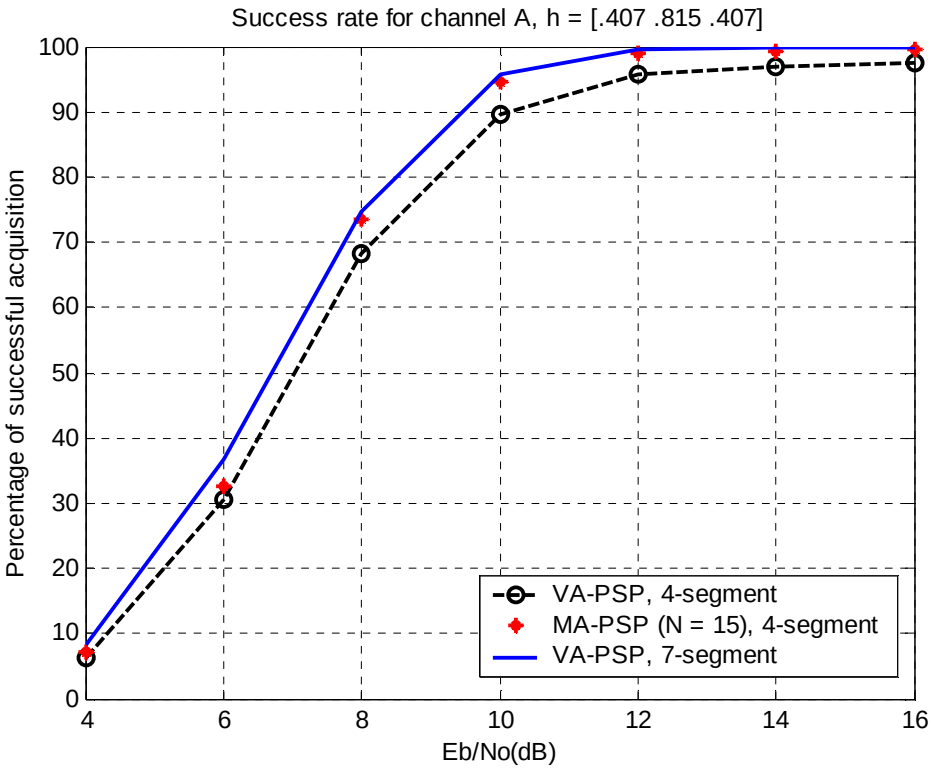


Figure 5.19: Performance comparison between VA-PSP and MA-PSP with segmentation.

5.5 Acquisition Performance of Reduced Complexity PSP

We have already the reported acquisition performance of higher order channels with VA-PSP. However, with the fourth order channel, the trellis has $4^4 = 256$ states which require 1,32,608 flops and comparison per symbol. This is huge a computation even for state-of-the-art computers. This motivated us to embed some state reduction techniques like DDFSE and the M-Algorithm in PSP. In DDFSE, with $\mu = 3$, the number of states is reduced by 75%, so only 33,152 flops and comparisons are required per symbol. For an equivalent complexity, the M-Algorithm requires $N = 61$ survivors (Table 3.3). With $\mu = 2$, the complexity could be further reduced. The acquisition performance with the reduced complexity algorithms are shown in Figure 5.20. It is obvious that the DDFSE-PSP is outperformed by the MA-PSP for equivalent complexity. Again the results are as expected and there are two reasons for this. The first reason has already been mentioned when we said that the M-algorithm outperforms any trellis-based algorithm of equal complexity since it retains the *globally* best paths. Moreover in DDFSE, the states are

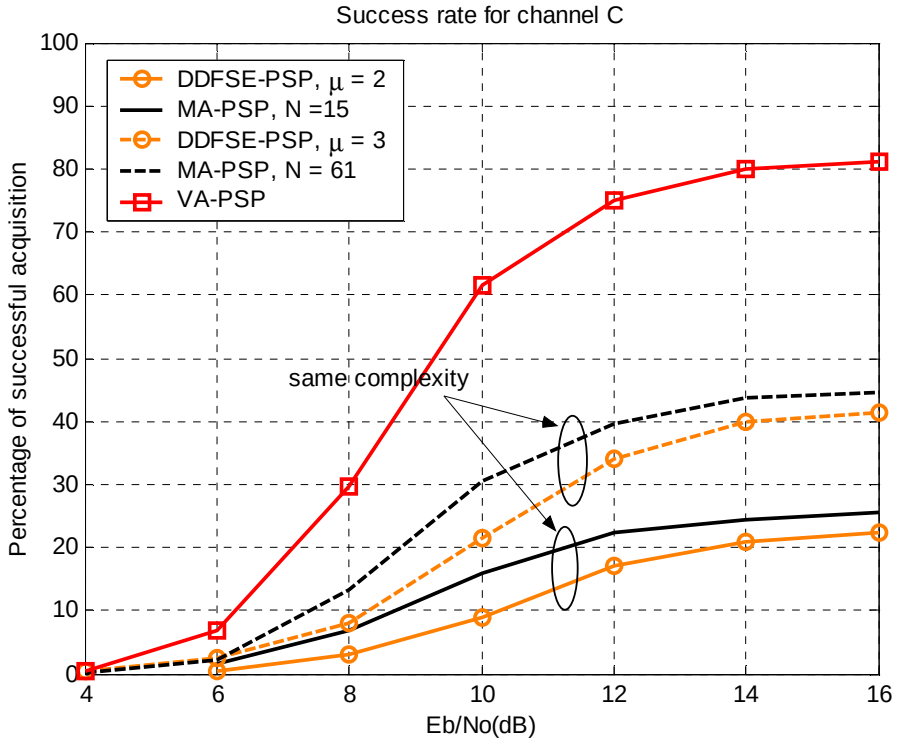


Figure 5.20: Performance comparison between reduced complexity algorithms: DDFSE-PSP and MA-PSP.

partially defined in the trellis and it doesn't perform particularly well for non-minimum phase channels. However, MA-PSP doesn't have these problems since the channel states are completely defined in the sub-optimal tree search (Figure 3.10). Also note that the performance of DDFSE-PSP approaches that of MA-PSP as the SNR improves. The performance of full-state (256-state) VA-PSP is also shown as a reference, which is significantly better than both state reduction techniques. We conclude that in the acquisition mode, any state reduction technique will eliminate potential candidate sequences resulting in performance degradation. However, in the tracking mode DDFSE-PSP's performance has been shown to be satisfactory [73].

5.5.1 Reduced Complexity PSP with Segmentation

In this section, we present results which prove that even state reduction techniques with segmentation can outperform full state VA-PSP. Figure 5.21 shows results for a fourth

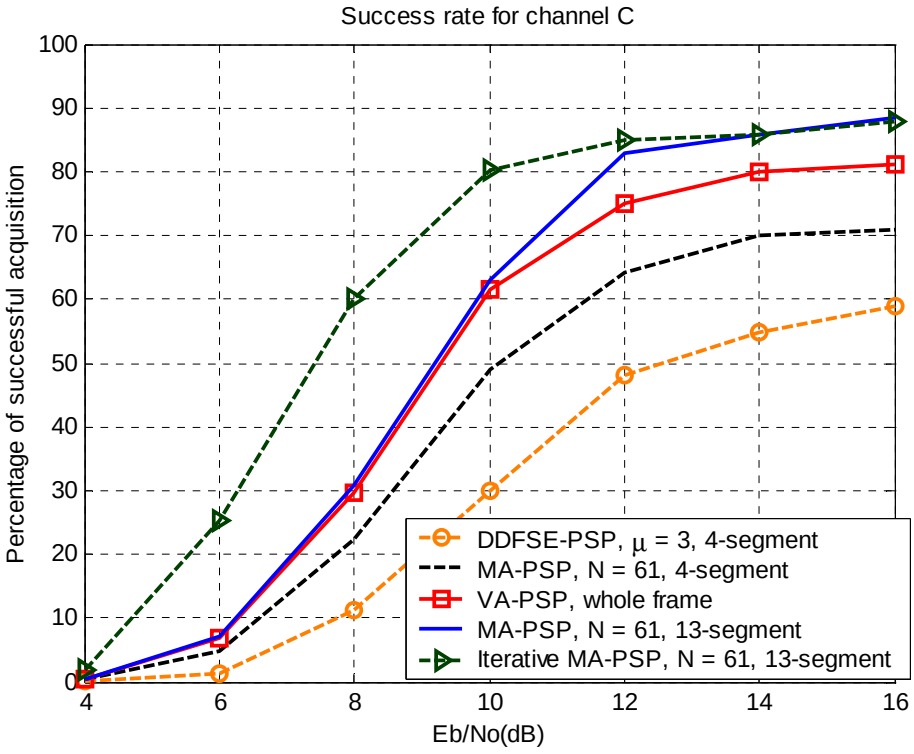


Figure 5.21: Acquisition performance of reduced complexity algorithms with segmentation.

order Rayleigh fading channel. MA-PSP with 13-segment performs better that VA-PSP. Once again, *iterative* MA-PSP acquisition performance at low SNR could be improved. Results also prove that, even with segmentation, DDFSE-PSP doesn't perform as well as MA-PSP.

5.6 Channel Order Mis-Match

In the previous simulation results we have assumed that the length of the channel impulse response is known. However, in practice this information may not be available beforehand. To address this problem, let's consider a case when there is an order mis-match. Table 5.2 shows the result of the estimated channel for five independent frames. For the sake of simplicity, only the magnitudes are shown and the phase rotations are suppressed. Note that for most of the cases, the last impulse response is insignificant meaning that the order of the channel is over-estimated. Therefore, the order could be set to two for an accurate channel order match for the next iterations. Also note the shifting phenomenon for the fourth frame, in which the first impulse response is insignificant.

Frame #	$\lvert\hat{h}[0]\rvert$	$\lvert\hat{h}[1]\rvert$	$\lvert\hat{h}[2]\rvert$	$\lvert\hat{h}[3]\rvert$
1	0.3746	0.8209	0.3818	0.0125
2	0.3993	0.8061	0.4108	0.0434
3	0.4043	0.8195	0.4268	0.0170
4	0.0391	0.4018	0.8062	0.3649
5	0.4319	0.8524	0.3903	0.0267

Table 5.2: Over-estimation of channel order for h=[.407 .815 .407], SNR=14 dB.

Table 5.3 shows the results where simulations are run with the assumption that the channel is first order. The estimated channel impulse response is typically the main tap of

the impulse response (may be shifted version). However, to check whether the order was under-estimated, these truncated estimates could be padded with zeros and used as initial estimates for the next frames. If the estimates after the second run have significant energy in place of added zeros, then the channel order was under-estimated and we need to pad more zeros to try the process once more.

Frame #	$\lvert\hat{h}[0]\rvert$	$\lvert\hat{h}[1]\rvert$
1	0.7127	0.4483
2	0.6660	0.5304
3	0.7380	0.4135
4	0.6544	0.3192
5	0.5652	0.4825

Table 5.3: Under-estimation of channel order for h=[.407 .815 .407], SNR=14 dB.

5.7 Conclusion

In this chapter, different techniques to improve PSP's acquisition performance have been presented. Frame segmentation at the receiver can increase the acquisition rate drastically. Iterative PSP with segmentation can further enhance performance. The Viterbi algorithm is not an optimal solution for unknown channels. It has been shown that more complex search algorithms can improve acquisition performance, however there is an upper-limit of the acquisition performance even at high SNRs. List-based algorithms like MA-PSP outperform trellis-based algorithms for equivalent complexity. Thus, MA-PSP requires fewer segments for the same performance. For higher order channels, the complexity increases, since the receiver needs more segments. Also with segmentation, reduced complexity MA-PSP outperforms conventional VA based PSP. PSP can also be used for channel order estimation in an iterative fashion.

Chapter 6: Conclusions

This chapter concludes the thesis by summarizing the work presented so far and providing suggestions for future research.

In Chapter 2, a brief description of the system model is given. An overview of the MLSE and Viterbi Algorithm is discussed, along with state reduction techniques. Chapter 3 provides a detailed analysis of adaptive MLSE for an unknown channel with emphasis on Per-Survivor Processing (PSP). More complex Generalized Viterbi Algorithm (GVA) based PSP and the list-based M-algorithm are also introduced. Chapter 4 presents an analytical description of distinguishability of sequences in terms of their matrix properties. A detailed description of the PSP's acquisition problems is presented. Transient related problems are highlighted and their solutions are also proposed. In Chapter 5, simulation results for various algorithms, discussed in previous chapters, are presented. Modifications to conventional PSP were suggested, which significantly improves acquisition rate.

6.1 Summary of Results

A class of algorithms based on adaptive MLSE for blind channel identification has been introduced. Analytical results indicate that the Viterbi Algorithm (VA) is an optimal sequence estimator *only* when the channel is known. However, in PSP, due to the imperfect knowledge of the channel parameters, the transition metrics become path-

dependent. If the metrics are path-dependent, then tree-search algorithms outperform trellis-search algorithms like VA.

PSP has the potential for blind channel acquisition. In the case of successful acquisition, the convergence rate of PSP is comparable to that of the pilot-aided RLS algorithm, which is phenomenally fast compared to other blind acquisition techniques. However, PSP is unlikely to converge when certain sequences are transmitted even when noise is not present. For example, *nearly indistinguishable* (NI) sequences are problematic for any joint data and channel estimator. Even with an *exhaustive* search, the ML receiver can't reliably detect such sequences. Also like other blind algorithms, PSP has the *phase ambiguity* problem. When *memoryless equivalent* sequences are transmitted, PSP can't detect the phase correctly. However, this problem can easily be solved using *differential* encoding and decoding scheme. Most of the PSP's misacquisitions occur when *near indistinguishable* (NI) sequences are transmitted. If the transmitted sequence is *shift near indistinguishable* (SNI), PSP usually exhibits a *shifting phenomenon*, where the identified channel and detected sequence are shifted (either right or left) versions of the original quantities.

More complex tree-search algorithms improve PSP's acquisition performance. Multiple-Survivor PSP (MS-PSP) outperforms VA-PSP at the cost of increased complexity. However, there is an upper limit on the performance, since even *exhaustive* search algorithms can't detect completely indistinguishable sequences. M-Algorithm based PSP (MA-PSP) outperforms trellis based PSP. For equivalent complexity, MA-PSP performs better than MS-PSP, VA-PSP, and DDFSE-PSP.

An innovative approach, called "frame *segmentation*," has been proposed in this thesis. Simulation results indicate that this technique improves convergence rate significantly. With non-overlapping segments, segmented PSP (Seg-PSP) has the same complexity as conventional PSP. However, the convergence rate of the former is much higher. *Iterative* PSP with segmentation (ISeg-PSP) can be used to further improve performance at low SNR values. For a given transmitted frame, higher order channels require more segments for the same performance. With segmentation, MA-PSP requires

fewer segments compared to VA-PSP for the same performance. Even reduced complexity MA-PSP with segmentation outperforms full-state conventional VA-PSP.

6.2 Suggestion for Future Work

Even though this thesis provides all of the results listed above, it is far from being comprehensive for all conceivable situations. Some suggestions for future research follows:

1) In the system model studied in this research, no pulse shaping is used. Therefore, the only possible source of ISI is the channel. The pulse shaping filter can be appended to the system and a more complex trellis has to take care of the composite filter consisting of the channel and the pulse-shaping filter.

2) The segmentation idea can be extended to the tracking mode in the case of time varying channels. However, since the channel parameters are no longer constant over the entire frame in a time varying environment, a forward-backward PSP may have to be employed once the *best* segment is found.

3) Completely *indistinguishable* sequences are the most problematic sequences, since no blind algorithm can detect them. Thus, a code might be developed which guarantees that the transmitted sequence is never *indistinguishable*. This facilitates convergence. Such a code property might be built into a linear space-time block code, for instance.

4) In this thesis, the signaling format is assumed known at the receiver. However, there are many scenarios where the modulation format is unknown or partially known to the receiver. For example, in a Multi-User Detection (MUD) environment, different users might transmit different signaling formats. With modifications to the existing algorithms, this case might be explored in detail.

Appendix A

Complexity Analysis for RLS and LMS Algorithms

Two adaptive algorithms that we consider for channel acquisition are Recursive Least Square (RLS) and Least Mean Square (LMS). The convergence rate of RLS is typically an order of magnitude faster than the simple LMS algorithm. However, this improvement in performance is achieved at the cost of increased computational complexity.

The required memory and the number of floating point operations (flops) for both of the algorithms are shown in Table B.1 and Table B.2, respectively, where L denotes the channel order. Note that the LMS has a complexity of $O(8L)$, while it is $O(13L^2)$ for the RLS. The correlation matrix is symmetric and in the calculation of the number of flops for the estimated inverse of the correlation matrix, $\mathbf{P}_k^i$, this fact has been exploited. This is also due to the symmetry of $\mathbf{P}_k^i$, $\mathbf{P}_k^i \cdot \hat{\mathbf{a}}^H = \hat{\mathbf{a}} \cdot \mathbf{P}_k^i$, and should be calculated only once.

Stored Variables	**RLS**	**LMS**
Survivor Sequence	$L+1$	$L+1$
Est. Inverse of auto-correlation matrix, $\mathbf{P}_{k+1}^j$	$(L+1)(L+1)$	…
Estimated channel vector, $\hat{\mathbf{h}}_{k+1}^j$	$L+1$	$L+1$
Total	L^2+4L+3	$2(L+1)$

Table A. 1: Memory requirement for the RLS and LMS algorithms.

Adaptive Algorithms	Functions	# of real additions	# of real multiplications
RLS	$\mathbf{k}_{k+1}^{j} = \frac{\mathbf{P}_k^i \cdot (\hat{\mathbf{a}}(i_k \rightarrow j_{k+1}))^H}{\lambda + \hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \cdot \mathbf{P}_k^i \cdot (\hat{\mathbf{a}}(i_k \rightarrow j_{k+1}))^H}$	$4L^2 + 10L + 5$	$4L^2 + 14L + 10$
	$\mathbf{P}_{k+1}^{j} = \frac{1}{\lambda}\left[\mathbf{P}_k^i - \mathbf{k}_{k+1}^{j} \cdot \hat{\mathbf{a}}(i_k \rightarrow j_{k+1}) \cdot \mathbf{P}_k^i\right]$	$2L^2 + 6L + 4$	$3L^2 + 9L + 6$
	$\hat{\mathbf{h}}_{k+1}^{j} = \hat{\mathbf{h}}_k^i + \mathbf{k}_{k+1}^{j} \cdot e(i_k \rightarrow j_{k+1})$	$4(L+1)$	$4(L+1)$
	Total	$6L^2 + 20L + 13$	$7L^2 + 27L + 20$
LMS	$\hat{\mathbf{h}}_{k+1}^{j} = \hat{\mathbf{h}}_k^i + \beta \cdot e(i_k \rightarrow j_{k+1}) \cdot \hat{\mathbf{a}}^H(i_k \rightarrow j_{k+1})$	$4(L+1)$	$4L + 6$

Table A. 2: Number of Flops required in the RLS and LMS algorithms.

The complexity increase in RLS compared to LMS is readily observed from Figure B.1.

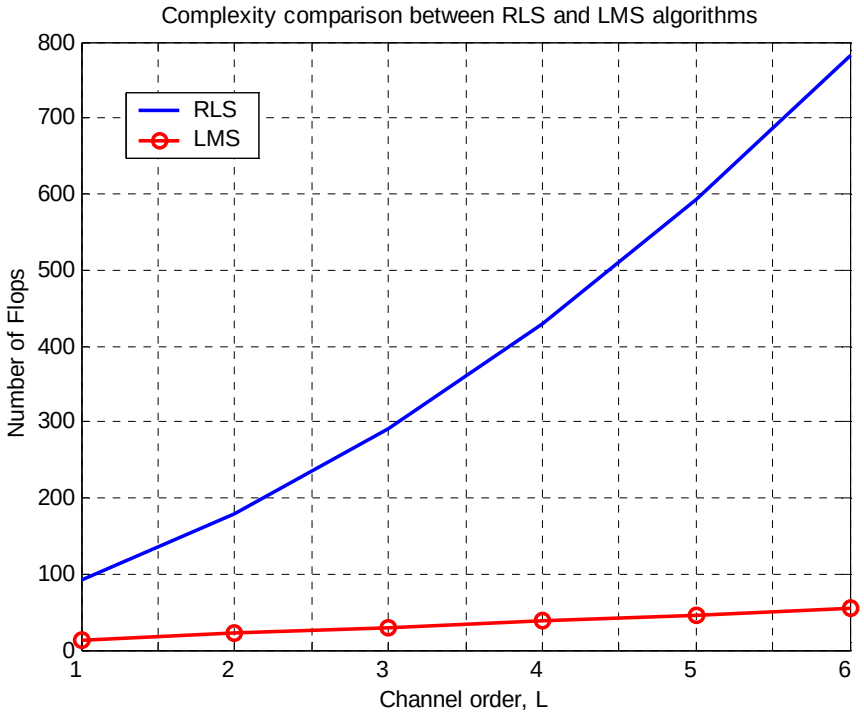


Figure A. 1: Computational requirement in RLS and LMS as a function of the channel order.

Bibliography

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

[2] J. M. W. Bergmans, S. A. Rajput, and F. A. M. Van De Laar, “On the use of decision feedback for simplifying the Viterbi decoder,” *Phillip Journal of Research*, vol. 42, pp. 399-428, no. 4, 1987.

[3] M. V. Eyuboglu and S. U. H. Quereshi, “Reduced-state sequence estimation for coded modulation on intersymbol-interference channels,” *IEEE J. Select. Areas Commun.*, vol. JSAC-7, pp. 989-995, Aug. 1989.

[4] G. Ungerboeck, “Adaptive maximum-likelihood receiver for carrier-modulated data-transmission systems,” *IEEE Trans. Commun.*, vol. COM-22, pp. 624-636, May 1974.

[5] Theodore S. Rappaport, “Wireless Communications principles & practice,” *Prentice Hall PTR*, 1996.

[6] John G. Proakis, “Digital Communications,” *McGraw-Hill*, 1995.

[7] S. U. H. Qureshi, “Adaptive equalization,” *Proceedings of the IEEE*, vol. 73, pp. 1349-1387, Sept 1985.

[8] D. Goddard, “Channel Equalization Using a Kalman Filter for Fast Data Transmission,” *IBM Journal of Research and Development*, pp.267-273, May 1974.

[9] Simon Haykin, “Adaptive Filter Theory,” *Prentice Hall*, 1991.

[10] F. Ling and J. G. Proakis, “Adaptive lattice decision-feedback equalizer – their performance and applications to time-variant multipath channels,” *IEEE Trans. Commun.*, vol. COM-33, pp. 348-356, April 1985.

[11] D. Williamson, R. A. Kennedy, and G. W. Pulford, “Block decision feedback equalization,” *IEEE Trans. on Commun.*, vol. COM-40, pp. 255-264, Feb. 1992.

[12] G. D. Forney, Jr., “The Viterbi Algorithm,” *Proceedings of the IEEE*, Vol. 61, pp. 268-278, March 1973.

[13] A. Polydoros and D. Kazakos, “Maximum likelihood sequence estimation in the presence of infinite inter-symbol interference,” *Conf, Rec.*, *ICC’79*, pp. 25.2.1-25.2.5, June 1979.

[14] James Hicks, “Overloaded Array Processing with Spatially Reduced Search Joint Detection”, *M.sc. thesis, Virginia Polytechnic Institute and State University*, Blacksburg, Virginia, 2000.

[15] F. L. Vermeulen and M. E. Hellman, “Reduced state Viterbi decoders for channels with inter-symbol interference,” *IEEE Conf. on Commun*, pp. 37B1-4, June 1974.

[16] S. U. Qureshi and E. E. Newhall, “An adaptive receiver for data transmission over time-dispersive channels,” *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 448-457, July 1973.

[17] D. D. Falconer and F. R. Magee, Jr., “Adaptive channel memory truncation for maximum-likelihood sequence estimation,” *Bell Syst. Tech. J.*, vol. 52, pp. 1541-1562, Nov. 1973.

[18] G. J. Foschini, “A reduced-state variant of maximum-likelihood sequence detection attaining optimum performance for high signal-to-noise ratio performance,” *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 505-509, Sept, 1977.

[19] P. J. McLane, “A residual interference error bound for truncated-state Viterbi Algorithm,” *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 549-553, Sept 1980.

[20] W. U. Lee and F. S. Hill, Jr. “A maximum likelihood sequence estimator with decision-feedback equalization,” *IEEE Trans. Commun.*, vol. COM-25, pp. 971-980, Sept. 1977.

[21] A. Duel and C. Heegard, “Delayed Decision Feedback Sequence Estimation,” *IEEE Transactions on Communication*, Vol. 37, no. 5, pp.428-436, May, 1989.

[22] M. V. Eyuboglu and S. U. Qureshi, “Reduced-state sequence estimation with set partitioning and decision feedback,” *IEEE Trans. Commun.*, vol. 36, no. 1, pp. 13-20, Jan. 1988.

[23] G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Quereshi, “Efficient modulation for band-limited channels,” *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 632-647, Sept. 1984.

[24] G. Ungerboeck, “Channel coding with multilevel/phase signals,” *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55-67, Jan. 1982.

[25] Runsheng He, J. R. Cruz, and Hongxin Song, “A reduced state sequence estimation technique and its applications,” *IEEE Trans. Magnetics*, vol. 35, no. 5, pp. 2331-2333, Sept. 1999.

[26] M. Ghosh and C. L. Weber, “Maximum-likelihood blind equalization,” *Proc. SPIE*, pp. 181-195, July 1991.

[27] J. Goutsia and J. Mendel, “Optimal simultaneous detection and estimation of filtered discrete semi-Markov chains,” *IEEE Trans. Information Theory*, vol. 34, pp. 551-568, May 1988.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from Incomplete data via the EM algorithm,” *J. Roy. Stat. Soc.*, ser. 39, pp. 1-38, 1977.

[29] F. R. Magee, Jr. and J.G. Proakis, “Adaptive maximum-likelihood sequence estimation for digital signaling in the presence of inter-symbol interference,” *IEEE Trans. Inform. Theory*, pp. 120-125, January, 1973.

[30] J. G. Proakis, “Adaptive equalization for TDMA digital mobile,” *IEEE Trans. Vehicular Technology*, vol. 40, pp. 333-341, May 1991.

[31] R. D'Avella, L. Moreno and M. Sant'Agostino, "An Adaptive MLSE receiver for TDMA digital mobile radio," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 122-129, Jan. 1989.

[32] P. R. Chevillat and E. Eleftheriou, "Decoding of trellis-encoded signals in the presence of intersymbol interference and noise," *IEEE Transactions on Communications*, vol. COM-37, pp. 669-676, July 1989.

[33] E. Dahlman, "New adaptive viterbi decoder for fast fading mobile radio channels," *Electronics letters*, Vol. 26, pp. 1572-1573, 1990.

[34] C.-K. Tzou, "Per-survivor processing: a general approach to MLSE in uncertain environments," *Ph. D. thesis, University of Southern California*, Dec. 1993.

[35] R. Raheli, A. Polydoros, C. Tzou, "Per-Survivor Processing: A General Approach to MLSE in Uncertain Environments," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 354-364, Feb./March/April, 1995.

[36] R. Raheli, A. Polydoros, C. Tzou, "Per-Survivor Processing: A General Approach to approximation and adaptive MLSE," *Proc. GLOBECOM'91*, pp. 1170-1175.

[37] A. J. Macdonald and J. B. Anderson, "PLL synchronization for coded modulation," *Proc. ICC '91*, pp. 52.6.1-52.6.5, June 1991.

[38] A. N. D'Andrea, U. Mengali, and G. M. Vitetta, "Multiple phase synchronization in continuous phase modulation," *Digital Signal Processing – A Review Journal*, Academic press, vol. 3, no. 3, pp. 188-198, July 1993.

[39] A. N. D'Andrea, U. Mengali, and G. M. Vitetta, "Approximate ML decoding of coded PSK with no explicit carrier phase reference," *IEEE Trans. Commun.*, vol. 42, part I, pp. 1033-1039, Feb./March/April, 1994.

[40] R. A. Iltis, "A digital receiver for demodulation of CDMA waveforms with apriori unknown delays and amplitudes," *Proc. MILCOM'91*, pp. 5.3.1-5.3.4.

[41] Z. Xie, C. K. Rushforth, R. T. Short, and T. K. Moon, "Joint signal detection and parameter estimation in multiuser communications," *IEEE Trans. Commun.*, vol. 41, pp. 1208-1216, Aug. 1993.

[42] R. E. Bellman and S. E. Dreyfus, “Applied dynamic programming,” Princeton, NJ, *Princeton University Press*, 1962.

[43] K. M. Chugg, “The condition for applicability of Viterbi Algorithm with implication for fading channel MLSD,” *IEEE Trans. Commun.*, vol. 46, pp. 1112-1116, Sept. 1998.

[44] J. B. Anderson and S. Mohan, “Sequential coding algorithms: A survey and cost analysis,” *IEEE Trans. Commun.*, vol. COM-32, pp. 169-176, Feb. 1984.

[45] R. M. Fano, “A heuristic discussion on probabilistic decoding,” *IEEE Trans. Info. Theory*, vol. IT-9, pp. 64-74, April, 1963.

[46] S. B. Wicker, “Error control systems,” New Jersey, Prentice Hall, 1995.

[47] P. R. Chevillat and D. J. Costello, “A multiple stack algorithm for erasure-free decoding of convolutional codes,” *IEEE Trans. Commun.*, vol. COM-25, pp. 1460-1470, Dec. 1977.

[48] K. M. Chugg, “Sequence estimation in the presence of parametric uncertainty,” Ph.D. dissertation, University of Southern California, 1995.

[49] P. G. Gulak and E. Shwedyk, “VLSI structures for Viterbi receivers,” *IEEE J. Select. Areas Commun*, vol. SAC-4, pp. 142-154. Jan. 1986.

[50] S. Mohan and A. K. Sood, “A multi-processor architecture for the (M,L)-algorithm suitable for VLSI implementation,” *IEEE Trans. Commun.*, vol. COM-34, pp. 1218-1224, Dec. 1986.

[51] S. Simmons, “A non-sorting VLSI structure for implementing the (M,L) algorithm,” IEEE *J. Select. Areas Commun.*, vol. 6, April 1988.

[52] S. Simmons, “Low complexity decoders for constant envelope digital modulations,” *IEEE Trans. Commun.*, vol. COM-31, pp. 1273-1280, Dec. 1983.

[53] G. J. Foschini, “A reduced-state variant of maximum likelihood sequence detection attaining optimal performance for high signal-to-noise ratios,” *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 605-609, Sept. 1977.

[54] N. Seshadri and C. E. Sundberg, “Generalized Viterbi algorithm for error detection with convolutional codes,” *IEEE GLOBECOM’89 Conf. Record*, pp. 1534-1537, 1989.

[55] N. Seshadri, “Joint data and channel equalization using fast blind trellis search techniques,” *Proc. GLOBECOM’ 90*, pp. 1659-1663, Dec. 1990.

[56] N. Seshadri, “Joint channel and data estimation using blind trellis search techniques,” *IEEE Trans. Commun.*, vol. 42, pp. 1000-1016, Feb./March/April, 1994.

[57] X. Yu and S. Pasupathy, “Innovation-based MLSE for Rayleigh fading channels,” *IEEE Trans. Commun.*, vol. 43, pp. 1534-1544, Feb./Mar./April, 1995.

[58] K-A. Wen, T-S. Wen, and J-F. Wang, “A new transform algorithm for Viterbi decoding,” *IEEE Trans. Commun.*, vol. 38, pp. 764-772, June 1990.

[59] G. Fettweis and H. Meyr, “Parallel Viterbi algorithm implementation: breaking the acs-bottleneck,” *IEEE Trans. Commun.*, vol. 37, pp. 785-790, Aug. 1989.

[60] P. Castoldi, R. Raheli, and G. Marino, “Efficient trellis search algorithms for adaptive MLSE on fast Rayleigh fading channels,” in *Proc. CTMC-GLOBECOM*, pp. 196-200, Nov. 1994.

[61] R. Raheli, G. Marino, and P. Castoldi, “Per-survivor processing and tentative decision: what is in between,” *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 127-129, Feb. 1996.

[62] G. Marino, R. Raheli, and G. Picchi, “Tunable complexity RSSE decoding of TCM signals,” in *Proc. Int. Symp. Inform. Theory (ISIT’94)*, pp. 175, June 1994.

[63] J. B. Evans and B. Liu, “Variable step size methods for the LMS adaptive algorithms,” *IEEE Int. Symp. Circuits and Systems’87*, vol. 2, pp. 422-425, May 1987.

[64] Z. Zhu and H. R. Sadjadpour, “Adaptive per-survivor processing,” *Conference record of the 34th Asilomar conference on Signals, Systems and Computers,’00*, pp. 1796-1800, 2000.

[65] S. A. Shah and B.-P. Paris, “Self-adaptive sequence detection via M-algorithm,” *Proc. ICC ’97*, vol. 3, pp. 1479-1483, June 1997.

[66] S. Simmons, “Breadth-first trellis decoding with adaptive effort,” *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 3-12, Jan. 1990.

[67] K. M. Chugg, “Blind acquisition characteristics of PSP-based sequence detector,” *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1518-1529, Oct. 1998.

[68] Sergio Verdu, “Multiuser Detection,” Cambridge University Press, 1998.

[69] K. M. Chugg and A. Polydoros, “On the existence and uniqueness of joint channel and data estimation,” in *Proc. 1995 IEEE Int. Symp. on Info. Theory*, Whistler, B.C., Canada, pp. 402.

[70] J. Hicks, J. Tsai, J. Reed, W. Tranter, and B. Woerner, “Overloaded array processing with MMSE-SIC,” *IEEE Vehicular technology conference*, pp. 542-546, Mar. 2002.

[71] C-K. Tzou, R. Raheli, and A. Polydoros, “Application of per-survivor processing to mobile digital communications,” *Proc. GLOBECOM*, pp. 77-81, Nov. 1993.

[72] W. Zhang and M. Miller, “Baseband equivalents in digital communication system simulation,” *IEEE Trans. Education.*, vol. 35, no. 4, pp. 376-382, Nov. 1992.

[73] J-T. Chen, J. Kim, and J. Cioffi, “Comparison of multi-channel adaptive MLSE equalizers using different channel tracking strategies,” *Proc. of Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2721-2724, 1999.

Vita

Mohammad H. Maruf was born in Dhaka, Bangladesh, November 1, 1974. He received a Bachelor of Engineering degree in Electrical Engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, 1999. He joined the graduate program at Virginia Polytechnic Institute and State University in January 2000, and has been a member of the Mobile and Portable Radio Research Group since August 2000. His research efforts in MPRG have focused on joint data and channel estimation techniques. He will continue as a Ph.D. student with Dr. William Tranter from spring of 2003.