Near Maximum-Likelihood Sequential-Search Decoding Algorithms for Binary Convolutional Codes

Prepared by Shin-Lin Shieh

Directed by Prof. Po-Ning Chen

In Partial Fulfillment of the Requirements

for the Degree of
Doctor of Philosophy

Department of Communications Engineering

National Chiao Tung University

Hsinchu, Taiwan 300, R.O.C.

June 29, 2008

Abstract

In this work, the maximum-likelihood sequential-search decoding algorithm proposed in [17] is revisited. By replacing the conventional Fano metric with one that is derived based on the Wagner rule, the sequential-search decoding in [17] guarantees the maximum-likelihood (ML) performance, and was therefore named the *maximum-likelihood sequential decoding algorithm* (MLSDA). It was then concluded by simulations that when the MLSDA is operated over the convolutional code trellis, its software computational complexity is in general considerably smaller than that of the Viterbi algorithm.

A common problem on sequential-type decoding is that at the signal-to-noise ratio (SNR) below the one corresponding to the cutoff rate, the average decoding complexity and the required stack size grow rapidly with the information length [25]. This problem, to some extent, prevent the practical use of sequential-type decoding from codes with long information sequence. In order to alleviate the problem in the MLSDA, we propose to directly eliminate the top path whose end node is Δ-trellis-level prior to the farthest one among all nodes that have been expanded thus far by the sequential search, which we termed the *early elimination*. We then analyze the early-elimination window Δ that results in negligible performance degradation for the MLSDA. Our asymptotic-based analytical result indicates that the required early elimination window for negligible performance degradation is around three times (resp. 2.2-fold) of the constraint length for rate one-half convolutional codes under additive white Gaussian (resp. binary symmetric) channel. For rate one-third convo-

lutional codes, the required early-elimination window reduces to two times (resp. 1.2-fold) of the constraint length for the same channel. The theoretical level thresholds almost coincide with the simulation results.

As a consequence of small early elimination window required for near maximum-likelihood performance, the MLSDA with early elimination modification rules out considerable computational burdens, as well as memory requirement, by directly eliminating a big number of the top paths. This makes the MLSDA with early elimination suitable for applications that dictate a low-complexity software implementation with near maximum-likelihood performance. The upper bounds of decoding complexity of both the MLSDAs with and without early elimination are subsequently derived by utilizing the Berry-Esseen inequality. Both the upper bound and the simulated complexity indicate that the average decoding complexity per output bit for the MLSDA with early elimination is almost irrelevant to the memory order, as well as the message length, for medium to high SNRs.

Contents

List of Figures

Chapter 1

Introduction

The convolutional code, as invented by Elias [5] in 1955, is perhaps the most famous error correcting code in the history of communication industry. Right after its invention, Wozencraft and Reiffen [39] proposed a sequential decoding algorithm to effectively decode convolutional codes with large constraint lengths. Thereafter, Fano [6] developed the sequential decoding algorithm with extreme efficiency. These works further inspired Zigangirov [40], and independently, Jelinek [21] for the invention and development of the *stack algorithm*.

Unfortunately, the sequential decoding algorithm has received little attention in the past 30 years due to its sub-optimum performance and lack of efficient and cost-effective hardware implementation. It is however specially suitable for the decoding of convolutional codes with large memory order because its decoding complexity is irrelevant to the code constraint length. For this reason, the sequential decoding algorithm has recently been proposed to be used in the decoding of the so-called "super-code" that considers the joint effect of multi-path channels and convolutional codes [19].

Another commonly used decoding algorithm for convolutional codes is the Viterbi algorithm. It operates on a convolutional code trellis, and has been shown to be a maximum-likelihood decoder [25]. Since its decoding complexity grows exponentially with the code

constraint length, the Viterbi algorithm is usually applied only for convolutional codes with short constraint lengths.

In 2002, a variant of the sequential decoding algorithm has been established. The new variant uses a novel metric derived based on the Wagner rule, and was proved to result in maximum-likelihood performance [17]. The new sequential-type decoding algorithm was therefore termed the *maximum-likelihood sequential decoding algorithm* (MLSDA). By simulations, the authors in [17] observed that from pure software implementation standpoint, the average decoding complexity of the MLSDA is in general considerably smaller than the Viterbi algorithm when the signal-to-noise ratio (SNR) of the additive white Gaussian noise (AWGN) channel is larger than 2 dB.

When the information sequence is long, path truncation was suggested for a practical implementation of the Viterbi decoder [25]. Instead of keeping all trellis branches of the survivor paths in the decoder memory, only a certain number of the most recent trellis branches is retained, and a decision is forced on the oldest trellis branch whenever a new data arrives in the decoder. In literature, three strategies have been proposed on the forceful decision: (1) majority-vote strategy that traces back from all states, and outputs the decision that occurs most often; (2) best state strategy that only traces back from the state with the best metric, and outputs the information bits corresponding to the path being traced; (3) random state strategy that randomly traces back from one state, and outputs the information bits corresponding to the path being traced. Although none of the three forceful strategies guarantees maximum-likelihood, their performance degradation can be made negligible as long as the traceback window or truncation window is sufficiently large.

In [9], Forney proved by random coding argument that a truncation window of 5.8-fold of the code constraint length suffices to provide negligible performance degradation for the best state strategy. Hemmati and Costello [20] later derived an upper performance bound

as a function of the truncation window for a specific convolutional encoder, and obtained a similar conclusion for the best state strategy. McEliece and Onyszchuk [28] studied the tradeoff between length of the truncation window and performance loss for the random state strategy, and concluded that the truncation window for the random state strategy should be about twice as large as that for the best state strategy.

Similar to the Viterbi algorithm, the decoding burden of the sequential decoding algorithm, both in memory consumption and in computational complexity, grows as the length of the information sequence increases. Yet, in order to compensate the SNR loss due to the additional zeros at the end of the information sequence, a long information sequence is often preferred in practice. One solution to reduce the decoding burden as a result of a practically long information sequence is to introduce the path truncation concept of the Viterbi algorithm to the sequential decoding algorithm. As an example, Zigangirov considered the situation, in which the decoder traces back the top path in stack to force the decisions of the symbols at those levels prior to a backsearch limit, and derived an error probability upper bound for the sequential decoding with backsearch limit [41]. In case the channel critical rate is smaller than $(\kappa - 1)/\kappa$ of the computational cutoff rate, where κ is the ratio of the backsearch limit against code constraint length, Zigangirov's bound was shown to reduce to the Yudkin-Viterbi bound [11] for infinite backsearch limit at low to medium rates, and coincide with the random coding bound at high rate [41].

In this dissertation, an alternative approach to lower the decoding complexity of the new variant of the sequential decoding algorithm, i.e., the MLSDA, is examined. Instead of tracing back the top path in stack to force the decision of the symbols beyond the backsearch limit, we propose to directly eliminate the top path whose end node is Δ-level-prior to the farthest node among all that have been expanded thus far by the sequential search, which is so named the *early elimination*.

In the analysis of sufficiently large Δ such that the performance degradation is negligible, two attempts based on different techniques are made. The first one follows similarly the random coding argument used by Forney [9], while the second one elaborates the code generator polynomial specifically for the convolutional code adopted. The random coding argument then indicates that under binary symmetric channels (BSCs), the required early elimination window for negligible performance degradation is just 2.2-fold of the constraint length for rate one-half convolutional codes, and for rate one-third convolutional codes, the required early-elimination window even reduces to 1.2-fold of the constraint length. With the knowledge of code generator polynomial, additional error rate due to early elimination can be formulated under additive white Gaussian noise (AWGN) channels, which is accordingly used to determine the sufficient large early elimination window for near optimal performance. Simulations are henceforth performed, and confirm the accuracy of these analytical results.

As a consequence of small early-elimination window required for near maximum-likelihood performance, the MLSDA with early-elimination modification rules out considerable computational burden, as well as memory requirement, by directly eliminating a large number of the top paths. It can also be implemented together with the backsearch scheme to provide timely decision of fixed delay to further reduce the decoding complexity. This suggests the potential and suitability of the MLSDA with early elimination for applications that dictate a low-complexity software implementation with near maximum-likelihood performance.

In the analysis of the decoding complexity of the MLSDA, as well as the complexity reduction due to early elimination, upper bounds that utilize the Berry-Esseen inequality [7, Sec. XVI. 5] are established. Both the analytical and simulation results substantiate that the early elimination modification can significantly reduce the decoding computational complexity. Also shown from these results is that the average decoding complexity per information bit for the MLSDA with early elimination does not grow with the message

length, which makes it specially suitable for the timely decoding of codes with long message lengths.

The rest of the dissertation is organized as follows. The channel model, convolutional coding, and the conventional sequential decoding algorithm as well as the Fano metric are briefed in Section 2. The MLSDA algorithm and its variation with early elimination are presented in Section 3. The analyses of the sufficient early elimination window for near-maximum-likelihood performance under BSC and AWGN channels are given in Sections 4 and 5, respectively. Complexity upper bounds for both the MLSDAs with and without early elimination scheme are presented in Section 6. The concluding remarks and future work are summarized in Section 7.

Chapter 2

Convolutional Codes, Channel Models and Sequential Decoding algorithms

In this chapter, the convolutional coding and channel model considered are introduced in Sections 2.1 and 2.2, respectively. Then, the conventional decoding algorithm as well as the Fano metric is briefed in Section 2.3.

2.1 Convolutional code and its graphical representation

A binary convolutional encoder is conveniently structured as a mechanism with shift registers and modulo-2 adders, where the encoder output bits are given by modulo-2 additions of selective shift register contents and input bits at present. Let $\mathscr{C}$ denote a binary (n, k, m) convolutional code, in which the encoder outputs a block of n bits whenever a block of k information bits are inputted. The value m designates the maximum number of previous k-bit blocks that have to be memorized in the encoder (i.e., if the number of stages of the jth shift register is K_j, then $m = \max_{1 \le j \le k} K_j$). The initial values of shift registers are all zeros,[1] and the current n output bits are linear combination of the present k input bits and

[1] One exception is the tail-biting convolutional code

the previous $m \times k$ input bits. In this work, we assume that the input sequence contains $k \times L$ bits that come from k input sequences, each of length L bits. In addition, m zeros will be attached at the end of each input sequence in order to reset the encoder shift registers. Consequently, these $k(L+m)$ input bits jointly induce $n(L+m)$ output bits.

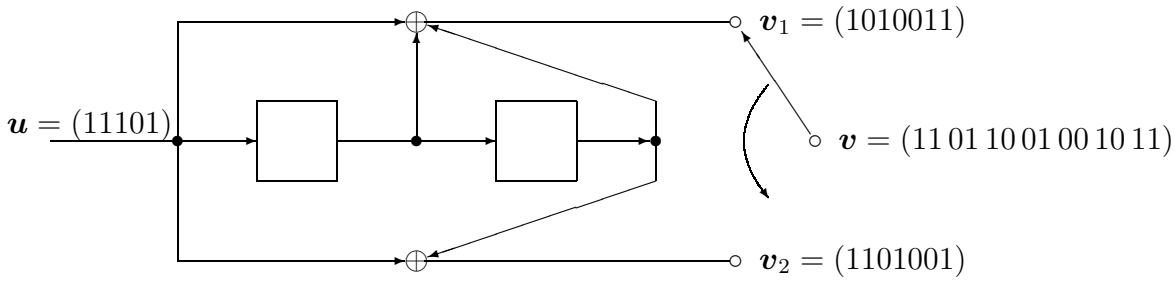

Figure 2.1: Encoder for the binary $(2, 1, 2)$ convolutional code with generators $g_1 = 7$ (octal) and $g_2 = 5$ (octal), where g_i is the generator polynomial characterizing the ith output.

Figures 2.1 and 2.2 exemplify the encoders of binary $(2, 1, 2)$ and $(3, 2, 2)$ convolutional codes, respectively. As illustrated in Fig. 2.1, the encoder of the $(2, 1, 2)$ convolutional code emits two output sequences,

$$\boldsymbol{v}_1 = (v_{1,0}, v_{1,1}, v_{1,2}, \ldots, v_{1,6}) = (1010011)$$

and

$$\boldsymbol{v}_2 = (v_{2,0}, v_{2,1}, v_{2,2}, \ldots, v_{2,6}) = (1101001)$$

due to the single input sequence $\boldsymbol{u} = (u_0, u_1, u_2, u_3, u_4) = (11101)$ of length $L = 5$, where u_0 is fed in the encoder first. The encoder then interleaves $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ to yield the codeword

$$\boldsymbol{v} = (v_{1,0}, v_{2,0}, v_{1,1}, v_{2,1}, \ldots, v_{1,6}, v_{2,6}) = (11\,01\,10\,01\,00\,10\,11)$$

of which the length is $2(5 + 2) = 14$. On the other hand, the encoder of the $(3, 2, 2)$

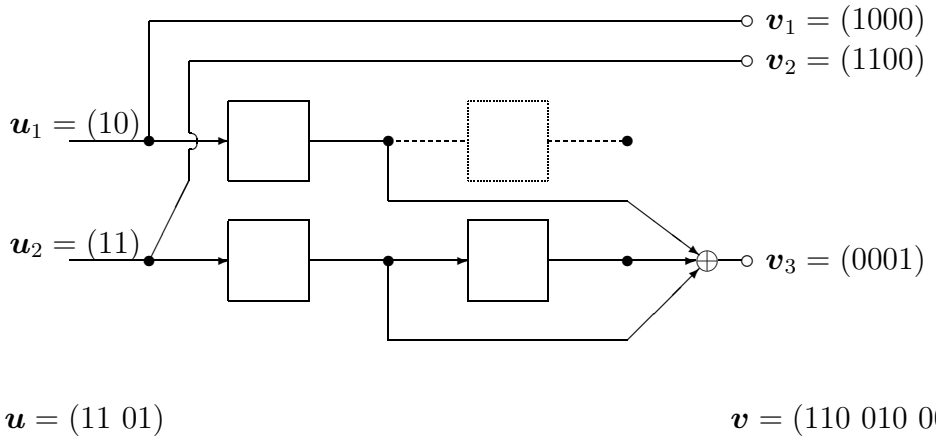

Figure 2.2: Encoder for the binary $(3, 2, 2)$ systematic convolutional code with generators $g_1^{(1)} = 4$ (octal), $g_1^{(2)} = 0$ (octal), $g_2^{(1)} = 0$ (octal), $g_2^{(2)} = 4$ (octal), $g_3^{(1)} = 2$ (octal) and $g_3^{(2)} = 3$ (octal), where $g_i^{(j)}$ is the generator polynomial characterizing the ith output according to the jth input. The dashed box is redundant and can actually be removed from this encoder; its presence here is only to help demonstrating the derivation of generator polynomials. Thus as far as the number of stages of the jth shift register is concerned, $K_1 = 1$ and $K_2 = 2$.

convolutional code in Fig. 2.2 generates the output sequences of

$$\boldsymbol{v}_1 = (v_{1,0}, v_{1,1}, v_{1,2}, v_{1,3}) = (1000),$$

$$\boldsymbol{v}_2 = (v_{2,0}, v_{2,1}, v_{2,2}, v_{2,3}) = (1100)$$

and

$$\boldsymbol{v}_3 = (v_{3,0}, v_{3,1}, v_{3,2}, v_{3,3}) = (0001)$$

due to the two input sequences $\boldsymbol{u}_1 = (u_{1,0}, u_{1,1}) = (10)$ and $\boldsymbol{u}_2 = (u_{2,0}, u_{2,1}) = (11)$ of length $L = 2$, which in turn generates the interleaved output sequence

$$\boldsymbol{v} = (v_{1,0}, v_{2,0}, v_{3,0}, v_{1,1}, v_{2,1}, v_{3,1}, v_{1,2}, v_{2,2}, v_{3,2}, v_{1,3}, v_{2,3}, v_{3,3}) = (110\,010\,000\,001)$$

of length $3(2 + 2) = 12$. In terminology, the interleaved output $\boldsymbol{v}$ is called the convolutional *codeword* corresponding to the combined input sequence $\boldsymbol{u}$.

One representation that characterizes the relation between the encoder inputs and encoder outputs is the generator polynomials. For example, $g_1(x) = 1+x+x^2$ and $g_2(x) = 1+x^2$ can be used to identify $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ induced by $\boldsymbol{u}$ in Fig. 2.1, where the appearance of x^i indicates that a physical connection is applied in Fig. 2.1 at the $(i+1)$th dot position, counted from the left. To be specific, putting $\boldsymbol{u}$ and $\boldsymbol{v}_i$ in polynomial form as $\boldsymbol{u}(x) = u_0+u_1x+u_2x^2+\cdots$ and $\boldsymbol{v}_i(x) = v_{i,0}+v_{i,1}x+v_{i,2}x^2+\cdots$ yields that $\boldsymbol{v}_i(x) = \boldsymbol{u}(x)g_i(x)$ for $i = 1, 2$, where addition of coefficients is based on modulo-2 operation.

Similarly, the relation between the inputs and the outputs can also be characterized by matrix operation. For example, the relation in Fig. 2.2 can be formulated as

$$\begin{bmatrix}\boldsymbol{v}_1(x) & \boldsymbol{v}_2(x) & \boldsymbol{v}_3(x)\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}_1(x) & \boldsymbol{u}_2(x)\end{bmatrix} \begin{bmatrix} g_1^{(1)}(x) & g_2^{(1)}(x) & g_3^{(1)}(x) \\ g_1^{(2)}(x) & g_2^{(2)}(x) & g_3^{(2)}(x)\end{bmatrix},$$

where $\boldsymbol{v}_i(x) = v_{i,0} + v_{i,1}x + v_{i,2}x^2 + \cdots$ and $\boldsymbol{u}_j(x) = u_{j,0} + u_{j,1}x + u_{j,2}x^2 + \cdots$ define the ith output sequence and the jth input sequence, respectively, and the generator polynomial $g_i^{(j)}(x)$ characterizes the relation between the ith output and the jth input sequences. For simplicity, generator polynomials are sometimes abbreviated by their coefficients in octal number format. Continuing the example in Fig. 2.1, the generator polynomials in octal format are $g_1 = 7$ (octal) and $g_2 = 5$ (octal).

A finite-length (n, k, m) convolutional code can be transformed to an equivalent linear block code with *effective code rate*[2] $R_{\text{effective}} = kL/[n(L+m)]$, where L is the length of the information input sequences. Usually, the *code rate* of the (n, k, m) convolutional code is referred to as $R = k/n$, which can be viewed as the effective code rate at L approaching infinity.

The *constraint length* of an (n, k, m) convolutional code has two different definitions in literature: $n_A = m+1$ [38] and $n_A = n(m+1)$ [25]. In this dissertation, the former definition

[2]The effective code rate is defined as the average number of input bits carried by an output bit [25].

is adopted, because it is more extensively used in industrial publications.

Let $\boldsymbol{v}_{(a,b)} = (v_a, v_{a+1}, \ldots, v_b)$ denote a portion of codeword $\boldsymbol{v}$, and abbreviate $\boldsymbol{v}_{(0,b)}$ by $\boldsymbol{v}_{(b)}$. Define the Hamming distance between the first rn bits of codewords $\boldsymbol{v}$ and $\boldsymbol{z}$ by:

$$d_H\left(\boldsymbol{v}_{(rn-1)}, \boldsymbol{z}_{(rn-1)}\right) = \sum_{i=0}^{rn-1} v_i \oplus z_i,$$

where "$\oplus$" denotes modulo-2 addition. The Hamming weight of the first rn bits of codeword $\boldsymbol{v}$ thus can be represented by $d_H(\boldsymbol{v}_{(rn-1)}, \mathbf{0}_{(rn-1)})$, where $\mathbf{0}$ represents the all-zero codeword. Furthermore, define the *column distance function* (CDF) $d_c(r)$ of a binary (n, k, m) convolutional code as the minimum Hamming distance between the first rn bits of any two codewords whose first n bits are distinct, i.e.,

$$d_c(r) = \min\left\{d_H(\boldsymbol{v}_{(rn-1)}, \boldsymbol{z}_{(rn-1)}) : \boldsymbol{v}_{(n-1)} \neq \boldsymbol{z}_{(n-1)} \text{ for } \boldsymbol{v}, \boldsymbol{z} \in \mathscr{C}\right\},$$

where $\mathscr{C}$ is the set of all codewords. Clearly, $d_c(r)$ is nondecreasing in r. Two cases of CDFs are of specific interest: $r = m + 1$ and $r = \infty$. In $r = \infty$ case, the Hamming distance should be calculated with infinite-length input sequences. However, $d_c(r)$ for an (n, k, m) convolutional code reaches its largest value $d_c(\infty)$ when r is a little beyond $5 \times m$ in most cases. This property facilitates the determination of $d_c(\infty)$. The value $d_c(\infty)$, or d_{free} in general, is called the *free distance*, whereas $d_c(m + 1)$ is called the *minimum distance* of the convolutional code.

The operational meanings of the minimum distance, the free distance and the CDF of a convolutional code are as follows. When a maximum-likelihood decoder is employed onto a received codeword with sufficiently large length, the error correcting performance is mainly characterized by d_{free} [36]. On the other hand, if a decoder figures the transmitted bits only based on the first $n(m + 1)$ received bits (as in, for example, the majority-logic decoding [26]), $d_c(m + 1)$ can be used instead to characterize the error correcting capability. Finally, the column distance function characterizes the decoding computational complexity, defined

as the number of metric computations performed for the sequential decoding algorithm. Usually, the sequential decoding algorithm requires a rapid initial growth of CDF in order to have a small decoding complexity.

Next, we introduce two graphical representations, *code tree* and *trellis*, of convolutional codewords. A *code tree* of a binary (n, k, m) convolutional code presents every codeword as a path on a tree. For input sequences of length L bits, the code tree consists of $(L + m + 1)$ levels. The single leftmost node at level 0 is called the *origin node*. At the first L levels, there are exactly 2^k branches leaving each node. For those nodes located at levels L through $(L + m)$, only one branch remains. The 2^{kL} rightmost nodes at level $(L + m)$ are called the *terminal nodes*. As expected, a path from the single origin node to a terminal node represents a codeword; therefore, it is named the *code path* corresponding to the codeword. Figure 2.3 illustrates the code tree for the encoder in Fig. 2.1 with a single input sequence of length 5.

In contrast to a code tree, a code *trellis* as termed by Forney [8] is a structure obtained from a code tree by merging those nodes in the same *state*. The *state* associated with a node is determined by the associated shift-register contents. For a binary (n, k, m) convolutional code, the number of states at levels m through L is 2^K, where $K = \sum_{j=1}^{k} K_j$ and K_j is the length of the jth shift register in the encoder; hence, there are 2^K nodes on these levels. Due to node merging, only one terminal node remains in a trellis. Analogous to a code tree, a path from the single origin node to the single terminal node in a trellis also mirrors a codeword. Figure 2.4 exemplifies the trellis of the $(3, 1, 2)$ convolutional code.

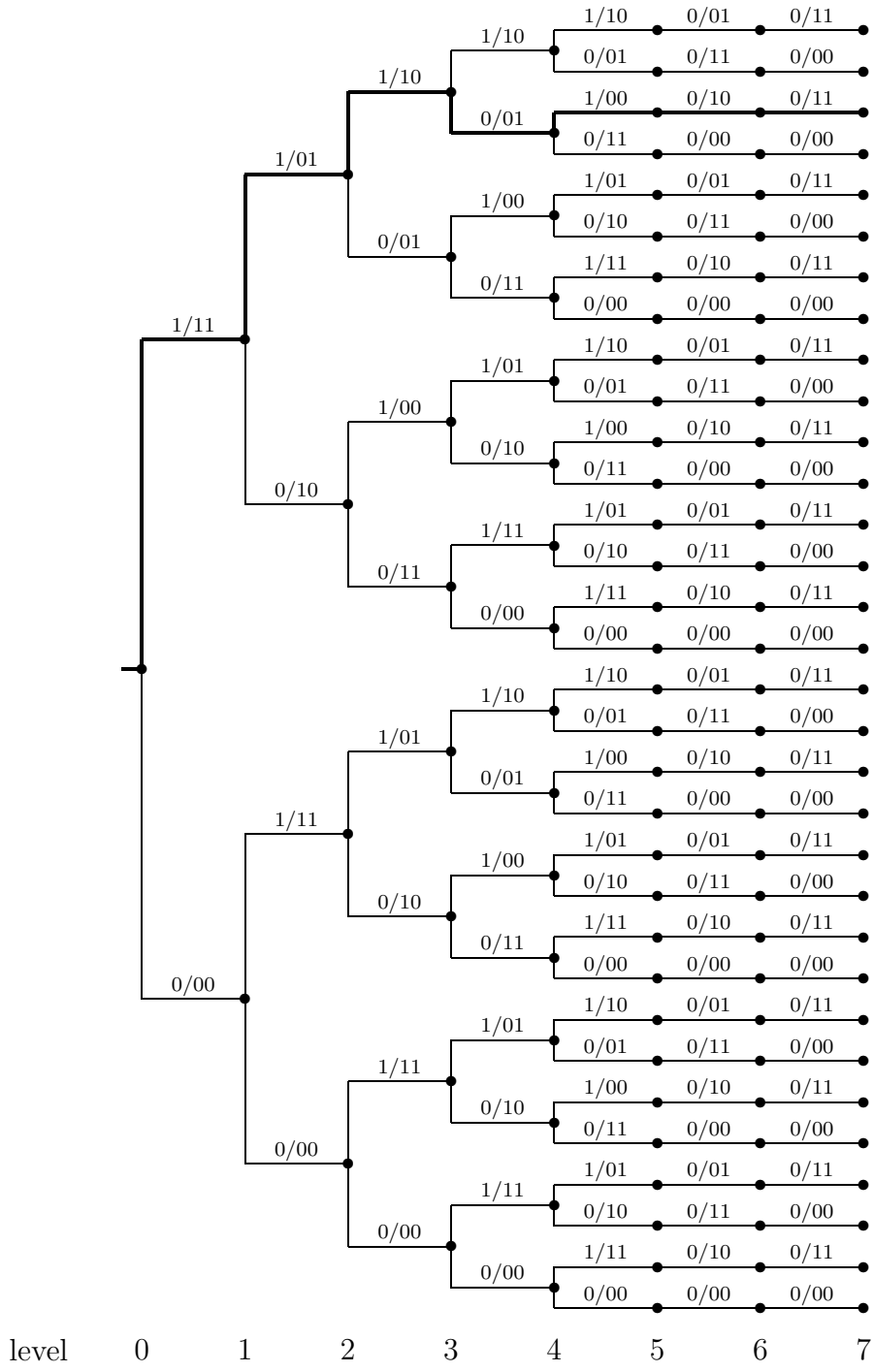

Figure 2.3: Code tree for the binary $(2, 1, 2)$ convolutional code in Fig. 2.1 with single input sequence of length 5. Each branch is labeled by its respective "input bit/output code bits". The code path indicated by the thick line is labeled in sequence by code bits 11, 01, 10, 01, 00, 10 and 11, and its corresponding codeword is $\boldsymbol{v} = (11\,01\,10\,01\,00\,10\,11)$.

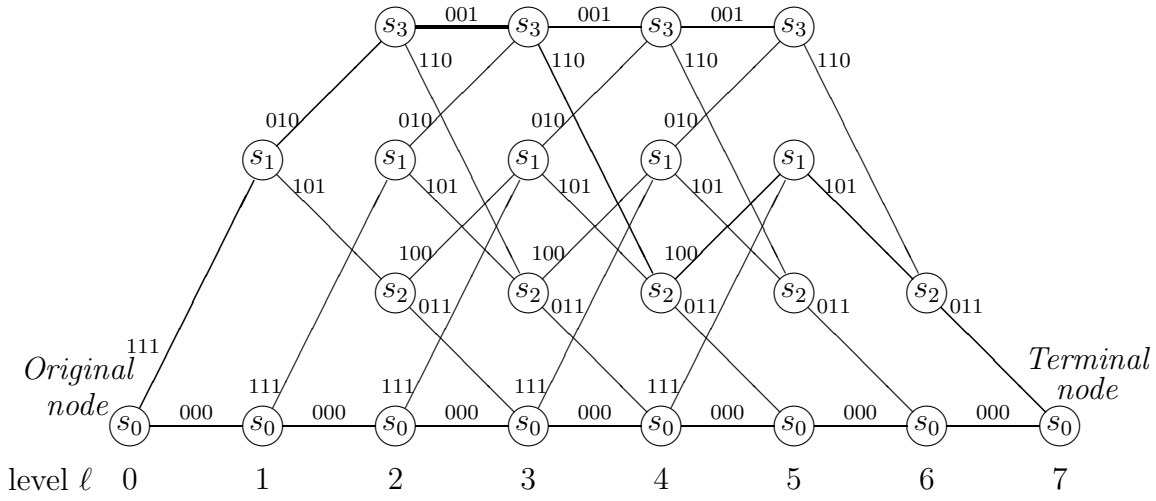

Figure 2.4: Trellis for a $(3, 1, 2)$ binary convolutional code with information length $L = 5$. In this case, the code rate $R = 1/3$ and the codeword length $N = 3(5 + 2) = 21$. The code path indicated by the thick line is labeled by 111, 010, 001, 110, 100, 101 and 011, thus its corresponding codeword is $\boldsymbol{v} = (111010001110100101011)$.

2.2 Channel models for hard-decision and soft-decision decoders

When the $n(L+m)$ convolutional code bits, encoded from kL input bits, are modulated into respective waveform for transmission over a physical medium, the received waveform is garbled by attenuation, distortion, interference, noise, etc. The demodulator then transforms the received waveform into discrete signals for use by the decoder to determine the original transmitted sequences. If the discrete signals are of two values, usually denoted by $\{0, 1\}$, then the demodulator is termed a *hard-decision demodulator*. If the demodulator passes discrete-in-time but continuous-in-value analog outputs to the decoder, then it is classified as a *soft-decision demodulator*. Terminologically, if a soft-decision demodulator is employed, then the subsequent decoder is also classified as a *soft-decision decoder*. In situation in which the decoder receives inputs from a hard-decision demodulator, the decoder is called a *hard-decision decoder*. In general, the soft-decision decoder provides better error correcting performance than the hard-decision decoder.

The decoder should determine the original information sequences based on the $n(L+m)$ demodulator decision outputs according to some criterion. The criterion that most frequently applies is the *maximum-likelihood decoding* (MLD) rule. It is well-known that the MLD minimizes the codeword error probability under the premiss that the transmitted codewords are equiprobable.

In this dissertation, we focus on two typical channel types — the *binary symmetric channel* (BSC) and the *additive white Gaussian noise* (AWGN) *channel*. The former is a typical channel model for the performance evaluation of hard-decision decoders, while the latter is widely used in examining the error rate of soft-decision decoders. It should be mentioned that for a coding system, a *channel* is simply a signal passage that *aggregates* all the intermediate effects onto the signal, including modulation, upconversion, signal distortion, downconver-

sion, demodulation, thermal noise and others. The demodulator in concept incorporates these effects into a widely adopted additive channel model as

$$r = s + n,$$

where r is the demodulator output, s is the transmitted signal, and n represents the aggregated signal distortion, simply termed *noise*.

The aggregated signal distortions for every transmitted and received bits are further assumed to be independent and identically distributed with common marginal distribution, which is termed *memoryless*. The extension to multiple independent channel usages is given by

$$r_j = s_j + n_j,$$

for $0 \leq j \leq N-1$, where all $\{n_j\}_{j=0}^{N-1}$ share the same probability distribution. In situation where the power spectrum of the noise samples is a constant, which can be interpreted as the noise contributing equal power at all frequencies, the noise is dubbed *white*. Therefore, the AWGN channel for a time-discrete coding system specifically indicates a memoryless noise sequence with a Gaussian distributed marginal.

As it turns out, the decoder inputs $r_0, r_1, \ldots, r_{N-1}$ are independent and Gaussian distributed with means $s_0, s_1, \ldots, s_{N-1}$, respectively, and equal variance $N_0/2$, where $N_0/2$ is *the doubled-sided noise power per hertz*. Assuming an *antipodal* transmission and equal prior on $c_j \in \{0, 1\}$ gives

$$s_j = s_j(c_j) = (-1)^{c_j}\sqrt{E},$$

where $c_j \in \{0, 1\}$ is the jth code bit, and

$$E = E[s_j^2] = \frac{1}{2}\left(\sqrt{E}\right)^2 + \frac{1}{2}\left(-\sqrt{E}\right)^2$$

is the average energy for single code bit transmission.

An index that guides the error performance for AWGN channels is the *signal-to-noise ratio* (SNR). For the time-discrete system considered, it is defined as the average signal energy E divided by N_0. Notably, the SNR ratio is invariable with respect to scaling of the demodulator output. In other words, the SNR ratio remains unchanged by scaling r_j by a multiplicative factor λ, since

$$\lambda \cdot r_j = \lambda \cdot (-1)^{c_j}\sqrt{E} + \lambda \cdot n_j.$$

Accordingly, the performance of the soft-decision decoding algorithm under AWGN channels is often illustrated by error rate against SNR. In order to account for the code redundancy for different code rates, the code bit energy E is further transformed to E_b, the equivalent average transmission energy per information bit. Their relation can be easily characterized by $E_b = E/R_{\text{effective}} = E \times [n(L+m)/(kL)]$ as the energy of $n(L+m)$ code bits should be equally distributed to kL information bits. Thus, a new index, denoted by E_b/N_0, is used instead of SNR$= E/N_0$ in plotting the performance curves.

The channel model can be further simplified to binary channel input and binary channel output, for which the noise sample n and the transmitted signal s are both elements of $\{0, 1\}$. Their modulo-2 addition yields the hard-decision demodulation output r. The binary channel statistics can be defined using two crossover probabilities: $p_1 = \Pr(r = 1|s = 0)$ and $p_2 = \Pr(r = 0|s = 1)$. In this dissertation, we focus on the case that two crossover probabilities are equal $p_1 = p_2 = p$. The binary channel is therefore *symmetric*, and is called the *binary symmetric channel*. The binary symmetric channel can be treated as a quantized simplification of the AWGN channel. Hence, the crossover probability p can be derived from

$$r_j = (-1)^{c_j}\sqrt{E} + n_j$$

as

$$p = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E}{N_0}}\right),$$

where

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-x^2} dx$$

is the complementary error function. This convention is adopted here in presenting the performance figures for BSCs.

Throughout the dissertation, as there exists a one-to-one correspondence between the transmitted signals $\boldsymbol{s} = (s_0, s_1, \ldots, s_{N-1})$ and the code words $\boldsymbol{c} = (c_0, c_1, \ldots, c_{N-1})$,

$$\Pr(\boldsymbol{r}|\boldsymbol{c}) = \prod_{j=0}^{N-1} \Pr(r_j|c_j)$$

and

$$\Pr(\boldsymbol{r}|\boldsymbol{s}) = \prod_{j=0}^{N-1} \Pr(r_j|s_j)$$

will be used interchangeably to represent the channel statistics of receiving $\boldsymbol{r}$ given that $\boldsymbol{s}$ (equivalently, $\boldsymbol{c}$) is transmitted.

2.3 Sequential decoding of convolutional codes

Since its discovery in 1963 [6], the Fano metric has become the most popular path metric in sequential decoding. The Fano metric was originally discovered through massive simulations, and was first used by Fano in his sequential decoding algorithm on code trees [6]. For any path $\boldsymbol{v}_{(\ell n-1)}$ that ends at level ℓ on a code tree, the *Fano metric* is defined as:

$$M\left(\boldsymbol{v}_{(\ell n-1)}|\boldsymbol{r}_{(\ell n-1)}\right) = \sum_{j=0}^{\ell n-1} M(v_j|r_j),$$

where $\boldsymbol{r} = (r_0, r_1, \ldots, r_{N-1})$ is the received vector, and the *bit metric* is defined as

$$M(v_j|r_j) = \log_2 \left(\frac{\Pr(r_j|v_j)}{\Pr(r_j)} \right) - R.$$

In the above bit metric formula, $R = k/n$ is the convolutional code rate, and the calculation of $\Pr(r_j)$ follows the convention that the code bits are transmitted with equal probability,

i.e.,

$$\Pr(r_j) = \sum_{v_j \in \{0,1\}} \Pr(v_j) \Pr(r_j|v_j) = \frac{1}{2} \Pr(r_j|v_j = 0) + \frac{1}{2} \Pr(r_j|v_j = 1).$$

For example, for BSCs with crossover probability p, where $0 < p < 1/2$, the Fano metric for path $\boldsymbol{v}_{(\ell n-1)}$ is given by:

$$M(\boldsymbol{v}_{(\ell n-1)}|\boldsymbol{r}_{(\ell n-1)}) = \sum_{j=0}^{\ell n-1} \log_2 \Pr(r_j|v_j) + \ell n(1-R), \tag{2.1}$$

where

$$\log_2 \Pr(r_j|v_j) = \begin{cases} \log_2(1-p), & \text{for } r_j = v_j; \\ \log_2(p), & \text{for } r_j \neq v_j. \end{cases}$$

In terms of the Hamming distance, (2.1) can be re-written as:

$$M\left(\boldsymbol{v}_{(\ell n-1)}|\boldsymbol{r}_{(\ell n-1)}\right) = -\alpha \cdot d_H(\boldsymbol{r}_{(\ell n-1)}, \boldsymbol{v}_{(\ell n-1)}) + \beta \cdot \ell, \tag{2.2}$$

where $\alpha = -\log_2[p/(1-p)] > 0$, and $\beta = n[1-R+\log_2(1-p)]$. It can be easily observed from (2.2) that a larger Hamming distance between the path labels and the respective portion of the received vector results in a smaller path metric. This property guarantees that when the received vector $\boldsymbol{r}$ is exactly the transmitted codeword, and $R < 1+\log_2(1-p)$ (equivalently, $\beta > 0$), the path metric increases along the correct code path, and the path metric along any incorrect path is smaller than that of the equally long correct path.[3] Such a property is essential for a metric to work properly with sequential decoding.

The ZJ algorithm was discovered by Zigangirov [40] and later independently by Jelinek [21] to search over a code tree for the optimal codeword based on the Fano metric. The algorithm is also called the *stack algorithm* because a stack is required in its implementation. For completeness, the stack algorithm [25] is quoted below.

[3] Without the assumption of error free reception, the code rate margin, below which the Fano-metric-based sequential decoding performs well, is the *channel capacity*. For BSCs with crossover probability P, the channel capacity is equal to $C = 1 + p\log_2(p) + (1-p)\log_2(1-p)$. The condition that $R < 1+\log_2(1-p) = C + p\log_2[(1-p)/p]$, derived from $\beta > 0$, can only justify the subsequent argument under the special case of error free reception. Channel capacity as a well-performed code rate margin for sequential decoding is beyond the scope of this dissertation. Interested readers can refer to [4].

<The Stack (ZJ) Algorithm>

Step 1. Load the stack with the origin node in the tree, whose metric is taken to be zero.

Step 2. Compute the metrics of the successors of the top path in the stack.

Step 3. Delete the top path from the stack.

Step 4. Insert the new paths in the stack and rearrange the paths in the stack in order of decreasing metric values.

Step 5. If the top path in the stack ends at a terminal node in the tree, the algorithm stops. Otherwise, return to Step 2.

A major issue in the implementation of the stack algorithm is the efficient maintenance of the stack. For example, the efficiency in the rearrangement of paths in the stack in Step 4 will greatly affect the time consumed in the sequential search.

Another issue that a practical implementation of the stack algorithm may encounter is that the stack size is finite in practice, and therefore, may be insufficient to accommodate the possible large number of paths examined during the search process. The situation is usually addressed as stack overflow. A straightforward way to deal with the stack overflow problem is to discard the paths with smaller metric values [25], since they are less likely to be the optimal code path. The technical issue remained is the determination of the practical stack size such that the performance degradation due to path discarding is within acceptable region.

Chapter 3

MLSDA and the Proposed Early Elimination Scheme

Assume that the binary codeword in a (N, K) linear block code $\mathscr{C}$ is transmitted over a binary-input time-discrete channel with channel output $\boldsymbol{r} \triangleq (r_0, r_1, \ldots, r_{N-1})$. Define the hard-decision sequence $\boldsymbol{y} \triangleq (y_0, y_1, \ldots, y_{N-1})$ corresponding to $\boldsymbol{r}$ as:

$$y_j \triangleq \begin{cases} 1, & \text{if } \phi_j < 0; \\ 0, & \text{otherwise,} \end{cases}$$

where $\phi_j \triangleq \log[\Pr(r_j|v_j = 0)/\Pr(r_j|v_j = 1)]$, and $\Pr(r_j|v_j)$ is the channel transition probability of r_j given v_j. According to the Wagner rule, the maximum-likelihood decoding output $\hat{\boldsymbol{v}}$ for received vector $\boldsymbol{r}$ is given by

$$\hat{\boldsymbol{v}} = \boldsymbol{y} \oplus \boldsymbol{e}^*, \tag{3.1}$$

where "$\oplus$" is the bit-wise exclusive-or operation, $\boldsymbol{e}^*$ is the one with the smallest $\sum_{j=0}^{N-1} e_j|\phi_j|$ among all error patterns $\boldsymbol{e} \in \{0,1\}^N$ satisfying $\boldsymbol{e}\mathbb{H}^T = \boldsymbol{y}\mathbb{H}^T$, and $\mathbb{H}$ is the parity check matrix of $\mathscr{C}$. Here, superscript "T" is used to denote the matrix transpose operation. Recall that a binary (n, k, m) convolutional code with input sequence of length L can be treated as a (N, K) linear block code with $N = n(L + m)$ and $K = kL$. Based on the observation in (3.1), a new sequential-type decoder can be established by replacing the Fano metric in the

conventional sequential decoding algorithm by a metric defined as:

$$\mu\left(\boldsymbol{x}_{(\ell n-1)}\right) \triangleq \sum_{j=0}^{\ell n-1} \mu(x_j), \tag{3.2}$$

where $\boldsymbol{x}_{(\ell n-1)} = (x_0, x_1, \ldots, x_{\ell n-1}) \in \{0,1\}^{\ell n}$ represents the label of a path ending at level ℓ in the (n, k, m) convolutional code tree, and $\mu(x_j) \triangleq (y_j \oplus x_j)|\phi_j|$. Since the new decoding metric is nondecreasing along the code path, and since finding $\boldsymbol{e}^*$ is equivalent to finding the code path with the smallest metric in the code tree, it was proved in [17] that the new sequential-type decoder can always locate the maximum-likelihood codeword through the priority-first sequential codeword search. For this reason, the new sequential-type decoder is named the *maximum-likelihood sequential decoding algorithm* (MLSDA) [17].

By adding a second stack, the MLSDA can be made to operate on a code trellis instead of a code tree [17]. The two stacks used in the trellis-based MLSDA are referred to as the *Open Stack* and the *Closed Stack*. The *Open Stack* contains all paths that end at the frontier part of the trellis being thus far explored (cf. Fig. 3.4). The Open Stack functions similarly as the single stack in the conventional sequential decoding algorithm. The *Closed Stack* stores the information of the ending states and ending levels of the paths that had been the top paths of the Open Stack. The Closed Stack is used to determine whether two paths intersect in the code trellis during the sequential search. The trellis-based MLSDA [17] is quoted below for completeness.

<Trellis-Based MLSDA>

Step 1. Load the Open Stack with the origin node whose metric is zero.

Step 2. Put into the Closed Stack *both the state and the same level of the end node of the top path in the* Open Stack. *Compute the path metric for each of the successor paths of the top path in the* Open Stack *by adding the branch metric of the extended branch to the path metric of the top path. Delete the top path from the* Open Stack.

Step 3. Discard the successor paths in Step 2, which end at a node that has the same state and level as any entry in the Closed Stack. *If any successor path ends at the same node as a path already in the* Open Stack, *eliminate the path with higher path metric.*[1]

Step 4. Insert the remaining successor paths into the Open Stack *in order of ascending path metrics. If two paths in the Open Stack have equal metric, sort them in order of descending levels. If, in addition, they happen to end at the same level, sort them randomly.*

Step 5. If the top path in the Open Stack *reaches the end of the convolutional code trellis, the algorithm stops; otherwise go to Step 2.*

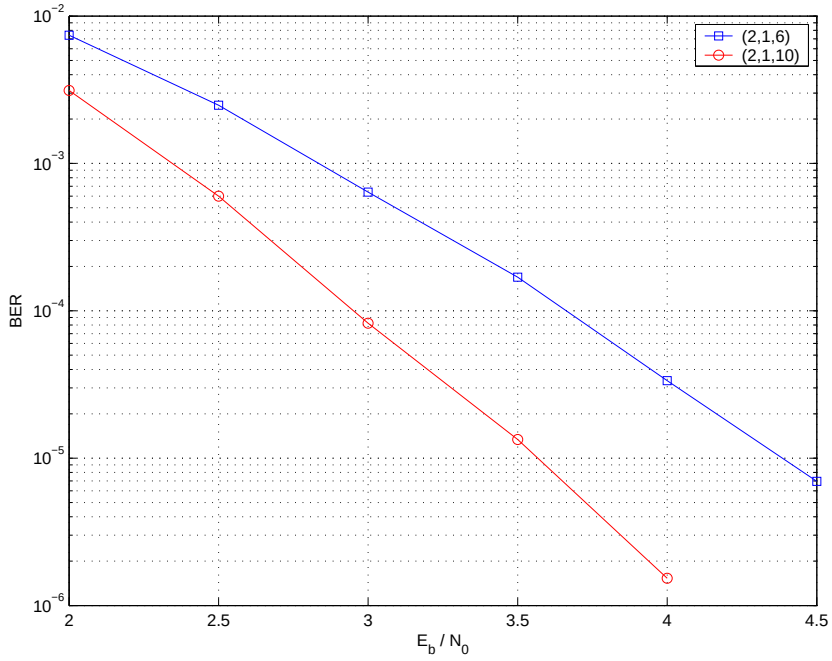

Figure 3.1: Bit error rates of the MLSDA for $(2, 1, 6)$ and $(2, 1, 10)$ convolutional codes with $L = 100$.

[1] For discrete channels, it may occur that the successor path not only ends at the same node as some path already in the Open Stack but also has equal path metric to it. In such case, just randomly eliminate one of them.

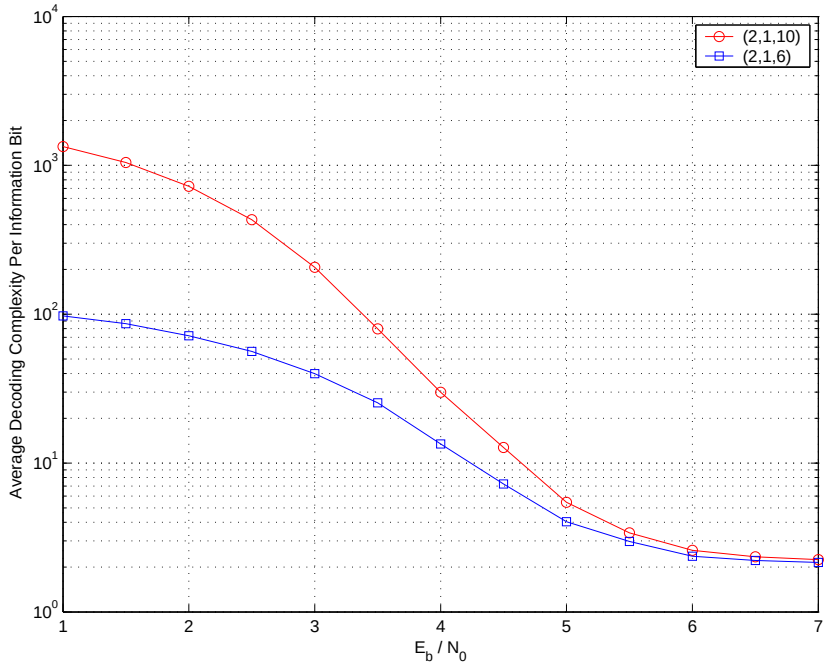

Figure 3.2: Average decoding complexities of the MLSDA for $(2, 1, 6)$ and $(2, 1, 10)$ convolutional codes with $L = 100$.

We next show the simulation results of performance and average decoding complexity for the MLSDA. The bit error rates of the MLSDA for $(2, 1, 6)$ and $(2, 1, 10)$ convolutional codes are summarized in Fig. 3.1, while the decoding complexity as measured by the number of metric computations is depicted in Fig. 3.2. Notably, the computational efforts of sequential-search decoding algorithms, including the MLSDA, are in fact determined not only by the number of metrics computed but also by the cost of searching and inserting of the stack elements. The latter cost however can be made of comparable order to the former by adopting the *double-ended heap* (DEAP) [3] data structure in the stack implementation.[2] This justifies the common usage of number of metric computations as the key determinant of the algorithmic complexity of the sequential-search decoding algorithm.

[2]In practical decoder design, only stacks with finite size are available. When the stack is full, one common strategy is to remove the bottom path, that is, the path with the worst path metric. A double ended heap is thus useful in this regard because it can access the top path as well as the bottom path in case of stack overflow. Throughout this dissertation, we assume an infinite stack size and hence, no stack overflow strategy is required. However, we propose to use DEAP for future practical decoder implementation.

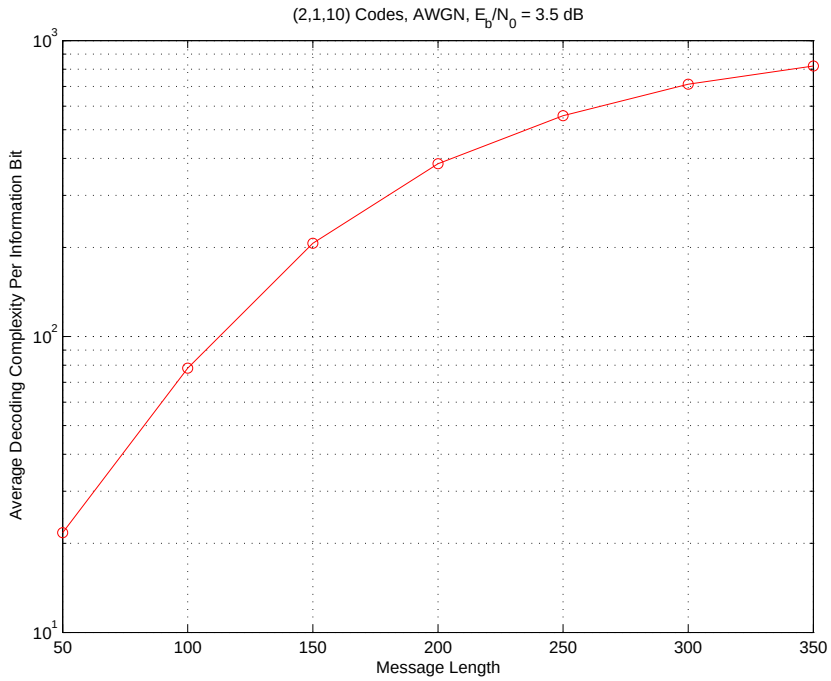

Figure 3.3: Average decoding complexity versus information length for the MLSDA applied to the $(2, 1, 10)$ convolutional code.

It can be observed from Fig. 3.2 that the average decoding complexities for the MLSDA is high for low SNRs. An even more serious problem is that the average decoding complexity per information bit grows as the information length increases as shown in Fig. 3.3. This phenomenon restricts the usage of the MLSDA for long convolutional codes.

We therefore introduce the early elimination modification to alleviate the problem of growing complexity with respect to the information length. The modification is based on the following observation. Suppose that the path ending at node C in Fig. 3.4 is a portion of the final code path to be located at the end of the sequential search, and suppose that the path ending at node D happens to be the current top path. Then, expanding node D until all of its offsprings finally have decoding metrics exceeding those of the successors of the path ending at node C may consume considerable but unnecessary number of computational efforts. This observation hints that by setting a proper level threshold Δ and directly eliminating the top

path whose level is no larger than $(\ell_{\max} - \Delta)$, where $\ell_{\max}$ is the largest level for all nodes that have been expanded thus far by the sequential search, the computational complexity of the MLSDA may be reduced without sacrificing much of the performance.

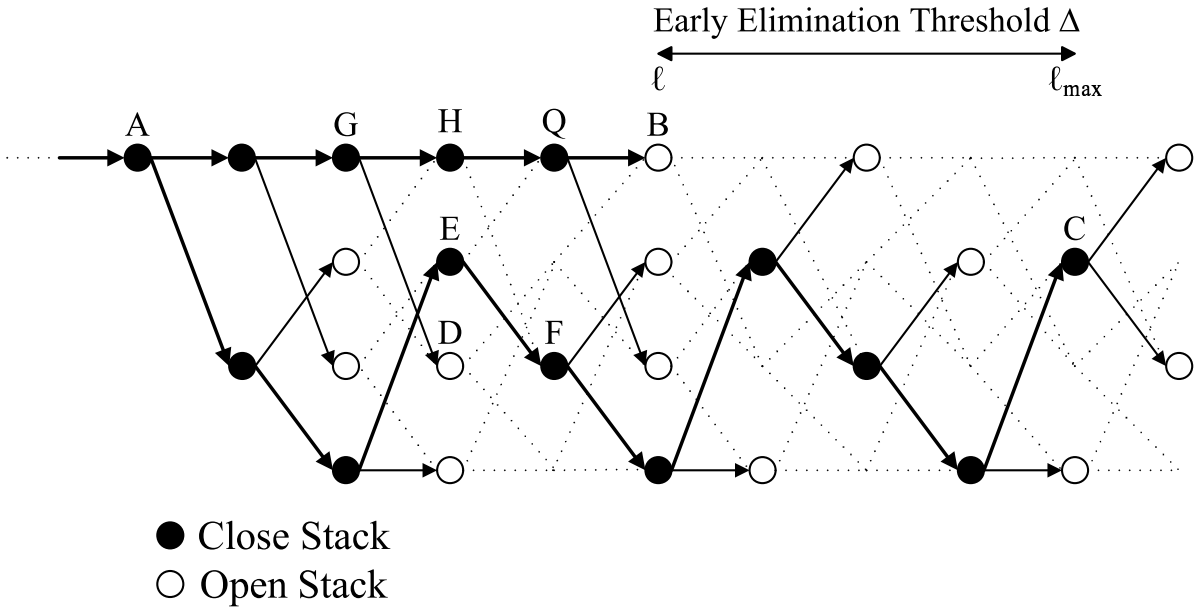

Figure 3.4: Early elimination window Δ in the trellis-based MLSDA.

It should be mentioned that since the decoding metric is monotonically nondecreasing along the path portion to be searched, the path that updates the current $\ell_{\max}$ is always the one with the smallest path metric among all paths ending at the same level [17]. In fact, this is the key to ensure that for the sequential search using the maximum-likelihood metric in (3.2), the first top path that reaches the last level of the code tree or code trellis is exactly the maximum-likelihood code path.

Based on the above observation, we propose to set a level threshold Δ in the trellis-based MLSDA, and directly eliminate the top path whose level is no larger than $(\ell_{\max} - \Delta)$. For this modification, we only need to modify Step 2 in the trellis-based MLSDA as follows.

<Trellis-Based MLSDA with Early Elimination Modification>

Initialization. Set a level threshold Δ. *Assign* $\ell_{\max} = 0$.

Step 2′. Perform the following check before executing the original Step 2 in the trellis-based MLSDA.

- *If the top path in the Open Stack ends at a node whose level is no larger than* $(\ell_{\max} - \Delta)$, *directly eliminate the top path, and go to Step 5; otherwise, update* $\ell_{\max}$ *if it is smaller than the ending level of the current top path.*

The choice of Δ is apparently a tradeoff between complexity and bit error probability. Intuitively, the smaller the Δ, the higher the possibility that the maximum-likelihood path is early eliminated. From simulation results, we found that the performance degradation is almost negligible simply for a small Δ. This encourages us to analyze the least value of Δ to produce near maximum-likelihood performance as well as the complexity reduction due to this early elimination modification.

Chapter 4

Analysis of the Window Size for Negligible Performance Degradation over BSC Channels

This chapter provides detailed derivation on the early elimination window that yields negligible performance degradation for binary symmetric channel (BSC) channels. As the random coding analysis is the main technique used to analyze the window size for the MLSDA, we will first review the random coding technique in the analysis of the truncation window size in Viterbi decoders in Section 4.1. Then, the derivation of the early elimination window for the MLSDA such that the performance degradation is negligible is presented in Section 4.2. Numerical and simulation results will be given in Section 4.3.

4.1 Random Coding Analysis of the Path Truncation Window in Viterbi Decoder

In [10], Gallager considered the discrete memoryless channel with input alphabet size I, output alphabet size J and channel transition probability P_{ji}, and presented the random coding bound for the maximum-likelihood decoding error P_e of the (N, K) block code as:

$$P_e \leq \exp\left\{-N\left[-\rho R + E_0(\rho, \boldsymbol{p})\right]\right\}$$

for all $0 \leq \rho \leq 1$, where $R = \log(I^K)/N = (K/N)\log(I)$ is the code rate measured in nats per symbol, $\boldsymbol{p} = (p_1, p_2, \cdots, p_I)$ is the input distribution adopted for the random selection of codewords, and

$$E_0(\rho, \boldsymbol{p}) \triangleq -\log \sum_{j=1}^{J} \left(\sum_{i=1}^{I} p_i P_{ji}^{1/(1+\rho)} \right)^{1+\rho}. \tag{4.1}$$

Gallager's result leads to the well-known random coding exponent:

$$E_r(R) \triangleq \max_{0 \leq \rho \leq 1} \max_{\boldsymbol{p}} [-\rho R + E_0(\rho, \boldsymbol{p})] = \max_{0 \leq \rho \leq 1} [-\rho R + E_0(\rho)],$$

where $E_0(\rho) \triangleq \max_{\boldsymbol{p}} E_0(\rho, \boldsymbol{p})$ is the Gallager function [42]. Notably, the random coding exponent is a lower bound of the channel reliability function $E(R) \triangleq \lim_{N \to \infty} -(1/N)\log(P_e)$ (provided the limit exists), and is tight for code rates above the cutoff rate.

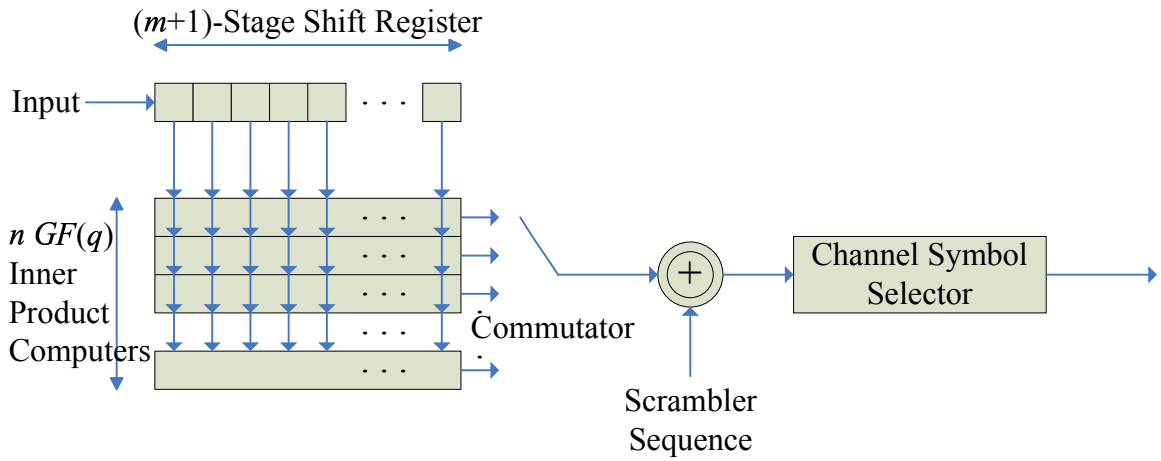

Figure 4.1: Single-input n-output encoder model considered in [36]. All elements are in GF(q), where q is either a prime or a power of a prime.

In [36], Viterbi applied similar random coding argument to the derivation of the decoding error for time-varying convolutional codes. Specifically, he considered a single-input n-output convolutional encoder with one $(m+1)$-stage shift register as shown in Fig. 4.1. The n inner product computers may change with each new input symbol, and hence, a time-varying code trellis is resulted. As all elements are assumed to be in GF(q), each input symbol will induce q branches on the code trellis, and each branch is labelled by n channel symbols.

As a result of the attached m zeros at the end, the encoder will produce $n(L+m)$ output channel symbols in response to the input sequence of L symbols. Under the above system setting, Viterbi showed that the maximum-likelihood decoding error $P_{e,c}$ for time-varying convolutional codes can be upper-bounded by:

$$P_{e,c} \leq \frac{q-1}{1-q^{-\lambda/R}} \exp[-n(m+1)E_0(\rho)] \tag{4.2}$$

for all $0 \leq \rho \leq 1$, where $R \triangleq \log(q)/n$ is the code rate in unit of nats per symbol, and $\lambda \triangleq E_0(\rho) - \rho R$ is a constant. Since λ is required to be positive, it can be concluded that:

$$\liminf_{n\to\infty} -\frac{1}{n} \log P_{e,c} \geq (m+1)E_c(R),$$

where $E_c(R) \triangleq \max_{\{\rho\in[0,1] \,:\, E_0(\rho)>\rho R\}} E_0(\rho)$. For symmetric channels, $E_0(\rho)$ is an increasing and concave function in ρ with $E_0(0) = 0$; therefore, $E_c(R)$ can be reduced to:

$$E_c(R) = \begin{cases} R_0, & \text{if } 0 \leq R < R_0; \\ E_0(\rho^*), & \text{if } R_0 \leq R < C; \\ 0, & \text{if } R \geq C, \end{cases} \tag{4.3}$$

where $R_0 = E_0(1)$ is the cutoff rate, $C = E_0'(0)$ is the channel capacity, and $\rho^* = \rho^*(R)$ is the unique solution of $E_0(\rho) = \rho R$. It is also shown in the same work that $E_c(R)$ is a tight exponent for $R \geq R_0$.

In order to derive the path truncation window with near-optimal performance, Forney [9] treated the truncated convolutional code as a block code, and upper-bounded the additional decoding error $P_{e,T}$ due to path truncation in the Viterbi decoder by means of Gallager's technique as:

$$P_{e,T} \leq \exp[-n\tau E_r(R)], \tag{4.4}$$

where τ is the truncation window size. Forney then noticed that as long as

$$\liminf_{n\to\infty} -\frac{1}{n} \log P_{e,T} > \limsup_{n\to\infty} -\frac{1}{n} \log P_{e,c}, \tag{4.5}$$

the additional error $P_{e,T}$ due to path truncation becomes exponentially negligible with respect to $P_{e,c}$. For $R \geq R_0$, condition (4.5) reduces to

$$\tau E_r(R) > (m+1)E_c(R)$$

by inequality (4.4) and the tightness of $E_c(R)$. A specific case is given in Fig. 4.2 in which the binary symmetric channel (BSC) with crossover probability 0.4 gives that the path truncation window at the cutoff rate $R_0 = 0.0146$ bit/symbol must be larger than $E_c(R_0)/E_r(R_0) \approx 0.0146/0.0025 = 5.84$-fold of the code constraint length. This number parallels the one obtained under the very noisy channels, where 5.8-fold of the code constraint length is suggested for the path truncation window at the cutoff rate [37].

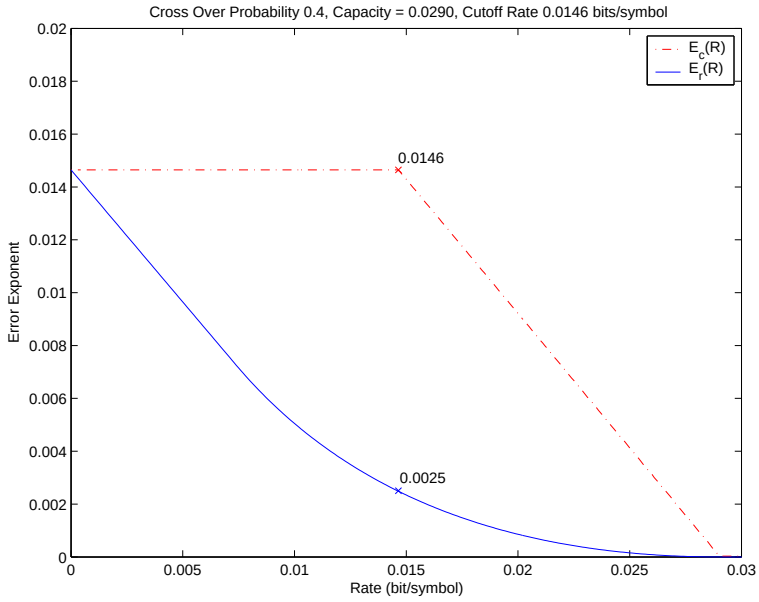

Figure 4.2: Exponent lower bound $E_r(R)$ of the additional error due to path truncation and exponent $E_c(R)$ of the maximum-likelihood decoding error for time-varying convolutional codes (without path truncation) under the BSC with crossover probability 0.4.

4.2 Sufficient Large Window Size for the MLSDA

For simplicity, the analysis in this section is restricted to the simple BSC with crossover probability ϵ. Extension analysis to other discrete channels can be likewise established.

Refer to Fig. 3.4 in our analysis below. Suppose that the path ending at node B at level ℓ is the current top path of the Open Stack, and let the current $\ell_{\max}$ be updated due to the expansion of node C. According to the merging operation at Step 3 of the trellis-based MLSDA, any two paths that survive in the Open stack can be traced back to a common node before which they shares common traces. Hence, we may assume that the path that ends at node B and the path that updates the current $\ell_{\max}$ have common traces before node A, whose level, without loss of generality, can be assumed zero in the below analysis.

Observe that the current top path ending at node B is early-eliminated if, and only if, node C is expanded earlier than node B, provided $\ell \leq \ell_{\max} - \Delta$. Since the decoding metric of the MLSDA is nondecreasing along the path portion to be searched, that node C is expanded prior to node B is equivalent to that

$$\mu\left(\boldsymbol{x}_{(\ell n-1)}\right) \geq \mu\left(\tilde{\boldsymbol{x}}_{(\ell_{\max} n-1)}\right), \tag{4.6}$$

which in turn is equivalent to

$$(1-\epsilon)^{n(\ell_{\max}-\ell)} \cdot \Pr\left(\boldsymbol{r}_{(\ell n-1)} \middle| \boldsymbol{x}_{(\ell n-1)}\right) \leq \Pr\left(\boldsymbol{r}_{(\ell_{\max} n-1)} \middle| \tilde{\boldsymbol{x}}_{(\ell_{\max} n-1)}\right). \tag{4.7}$$

The above statement can be proved as follows. For the BSC with crossover probability $0 < \epsilon < 1/2$,

$$\phi_j \triangleq \log \frac{\Pr(r_j|v_j=0)}{\Pr(r_j|v_j=1)} = \begin{cases} \log[(1-\epsilon)/\epsilon], & \text{if } r_j = 0; \\ \log[\epsilon/(1-\epsilon)], & \text{if } r_j = 1. \end{cases}$$

Hence,

$$r_j = y_j = \begin{cases} 1, & \text{if } \phi_j < 0; \\ 0, & \text{otherwise} \end{cases},$$

and $\mu(x_j) = (y_j \oplus x_j)|\phi_j| = (r_j \oplus x_j)\log[(1-\epsilon)/\epsilon]$. As a result, (4.6) is equivalent to

$$\begin{aligned}
&\mu\left(\boldsymbol{x}_{(\ell n-1)}\right) \geq \mu\left(\tilde{\boldsymbol{x}}_{(\ell_{\max} n-1)}\right) \\
\Leftrightarrow\ &\sum_{j=0}^{\ell n-1} \mu(x_j) \geq \sum_{j=0}^{\ell_{\max} n-1} \mu(\tilde{x}_j) \\
\Leftrightarrow\ &\sum_{j=0}^{\ell n-1} (r_j \oplus x_j) \geq \sum_{j=0}^{\ell_{\max} n-1} (r_j \oplus \tilde{x}_j) \\
\Leftrightarrow\ &\sum_{j=0}^{\ell n-1} [(r_j \oplus \tilde{x}_j) - (r_j \oplus x_j)] + \sum_{j=\ell n}^{\ell_{\max} n-1} (r_j \oplus \tilde{x}_j) \leq 0.
\end{aligned}$$

Similarly, (4.7) is equivalent to

$$\begin{aligned}
&(1-\epsilon)^{n(\ell_{\max}-\ell)} \Pr\left(\boldsymbol{r}_{(\ell n-1)} \middle| \boldsymbol{x}_{(\ell n-1)}\right) \leq \Pr\left(\boldsymbol{r}_{(\ell_{\max} n-1)} \middle| \tilde{\boldsymbol{x}}_{(\ell_{\max} n-1)}\right) \\
\Leftrightarrow\ &\sum_{j=0}^{\ell n-1} \log \Pr(r_j|x_j) + n(\ell_{\max}-\ell)\log(1-\epsilon) \leq \sum_{j=0}^{\ell_{\max} n-1} \log \Pr(r_j|\tilde{x}_j) \\
\Leftrightarrow\ &\sum_{j=0}^{\ell n-1} [(1-r_j \oplus x_j)\log(1-\epsilon) + (r_j \oplus x_j)\log(\epsilon)] + n(\ell_{\max}-\ell)\log(1-\epsilon) \\
&\qquad\qquad \leq \sum_{j=0}^{\ell_{\max} n-1} [(1-r_j \oplus \tilde{x}_j)\log(1-\epsilon) + (r_j \oplus \tilde{x}_j)\log(\epsilon)] \\
\Leftrightarrow\ &\log\frac{(1-\epsilon)}{\epsilon}\left(\sum_{j=0}^{\ell n-1}[(r_j \oplus \tilde{x}_j) - (r_j \oplus x_j)] + \sum_{j=\ell n}^{\ell_{\max} n-1} (r_j \oplus \tilde{x}_j)\right) \leq 0 \\
\Leftrightarrow\ &\sum_{j=0}^{\ell n-1}[(r_j \oplus \tilde{x}_j) - (r_j \oplus x_j)] + \sum_{j=\ell n}^{\ell_{\max} n-1} (r_j \oplus \tilde{x}_j) \leq 0.
\end{aligned}$$

Therefore, the desired equivalence of (4.6) and (4.7) is validated.

By noting that for the MLSDA, the path that updates the current $\ell_{\max}$ is exactly the one with the smallest path metric among all paths ending at the same level [17], condition (4.7) can be equivalently re-written as:

$$(1-\epsilon)^{n(\ell_{\max}-\ell)} \cdot \Pr\left(\boldsymbol{r}_{(\ell n-1)} \middle| \boldsymbol{x}_{(\ell n-1)}\right) \leq \max_{\tilde{\boldsymbol{x}}_{(\ell_{\max} n-1)} \in \mathcal{C}_{\ell_{\max}}} \Pr\left(\boldsymbol{r}_{(\ell_{\max} n-1)} \middle| \tilde{\boldsymbol{x}}_{(\ell_{\max} n-1)}\right), \qquad (4.8)$$

where $\mathcal{C}_{\ell_{\max}}$ is the set of all labels of length $\ell_{\max} n$, whose corresponding paths consist of different branches from path AB after node A. Consequently, additional decoding error may

be introduced by early elimination if (4.8) is valid for some ℓ and $\ell_{\max}$ with $\ell \leq \ell_{\max} - \Delta$, when $\boldsymbol{x}$ is the transmitted codeword.[1]

Continue the derivation by replacing $\ell_{\max}$ by β for notational convenience. The probability $\xi(\ell, \beta)$ that (4.8) occurs is given by:

$$\xi(\ell, \beta) = \sum_{\boldsymbol{r}_{(\beta n-1)} \in \{0,1\}^{\beta n}} \Phi_0\left(\boldsymbol{r}_{(\beta n-1)}\right) \Pr\left(\boldsymbol{r}_{(\beta n-1)} \middle| \boldsymbol{x}_{(\beta n-1)}\right), \tag{4.9}$$

where $\Phi_0\left(\boldsymbol{r}_{(\beta n-1)}\right) = 1$ if (4.8) is valid, and 0, otherwise. From

$$\Phi_0\left(\boldsymbol{r}_{(\beta n-1)}\right) \leq \left[\frac{\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)} \in \mathcal{C}_\beta} \Pr\left(\boldsymbol{r}_{(\beta n-1)} \middle| \tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}}{(1-\epsilon)^{n(\beta-\ell)/(1+\rho)} \Pr\left(\boldsymbol{r}_{(\ell n-1)} \middle| \boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}} \right]^{\rho} \quad \text{for } \rho \geq 0,$$

we obtain:

$$\xi(\ell, \beta) \leq \sum_{\boldsymbol{r}_{(\beta n-1)} \in \{0,1\}^{\beta n}} \left[\frac{\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)} \in \mathcal{C}_\beta} \Pr\left(\boldsymbol{r}_{(\beta n-1)} \middle| \tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}}{(1-\epsilon)^{n(\beta-\ell)/(1+\rho)} \Pr\left(\boldsymbol{r}_{(\ell n-1)} \middle| \boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}} \right]^{\rho} \Pr\left(\boldsymbol{r}_{(\beta n-1)} \middle| \boldsymbol{x}_{(\beta n-1)}\right).$$

Taking expectation of $\xi(\ell, \beta)$ with respect to random selection of codewords of length (βn) according to code bit selection distribution $\boldsymbol{p} = (p_0, p_1)$, where p_0 and p_1 are the probabilities

[1] Since early-elimination of the path with label $\boldsymbol{x}$ is always performed whenever (4.8) is valid, it is clear that additional error is introduced only when the transmitted label $\boldsymbol{x}$ corresponds to the maximum-likelihood code path. In other words, when $\boldsymbol{x}$ does not label the maximum-likelihood code path, the validity of (4.8) or early-elimination of the path with label $\boldsymbol{x}$ will not add a new error to maximum-likelihood decoding. As what we concern is an upper probability bound for the additional error due to early-elimination, it suffices to analyze the probability bound on the occurrence of (4.8).

Notably, when equality holds in (4.8), $\boldsymbol{x}$ will still be early-eliminated according to Step 4 of the algorithm.

respectively for bits 0 and 1, yields that:

$$\begin{aligned}
\overline{\xi(\ell,\beta)} &\leq (1-\epsilon)^{-n(\beta-\ell)\rho/(1+\rho)} \sum_{\boldsymbol{r}_{(\beta n-1)}\in\{0,1\}^{\beta n}} \frac{\overline{\left[\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)}\in\mathcal{C}_\beta} \Pr\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}\right]^{\rho}}}{\times\Pr\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}\Pr\left(\boldsymbol{r}_{(\ell n,\beta n-1)}\middle|\,\boldsymbol{x}_{(\ell n,\beta n-1)}\right)} \quad (4.10)\\
&\leq (1-\epsilon)^{-n(\beta-\ell)\rho/(1+\rho)} \sum_{\boldsymbol{r}_{(\beta n-1)}\in\{0,1\}^{\beta n}} \frac{\left[\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)}\in\mathcal{C}_\beta} \overline{\Pr\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}}\right]^{\rho}}{\times\Pr\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}\Pr\left(\boldsymbol{r}_{(\ell n,\beta n-1)}\middle|\,\boldsymbol{x}_{(\ell n,\beta n-1)}\right)} \quad (4.11)\\
&= |\,\mathcal{C}_\beta|^{\rho}\times(1-\epsilon)^{-n(\beta-\ell)\rho/(1+\rho)} \sum_{\boldsymbol{r}_{(\beta n-1)}\in\{0,1\}^{\beta n}} \frac{\left[\overline{\Pr\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}}\right]^{\rho}}{\Pr\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}\Pr\left(\boldsymbol{r}_{(\ell n,\beta n-1)}\middle|\,\boldsymbol{x}_{(\ell n,\beta n-1)}\right)},
\end{aligned}$$

where (4.10) holds since labels $\boldsymbol{x}_{(\ell n-1)}$ and any labels in $\mathcal{C}_\beta$ are selected independently, and (4.11) is valid due to Jensen's inequality with $\rho \leq 1$. Finally, by noting that $|\mathcal{C}_\beta| \leq 2^{k\beta} = 2^{n\beta R}$, we obtain:

$$\overline{\xi(\ell,\beta)} \leq 2^{-\ell n[-\rho R+E_0(\rho,\boldsymbol{p})]}\cdot 2^{-(\beta-\ell)n[-\rho R+\rho\log_2(1-\epsilon)/(1+\rho)+E_1(\rho,\boldsymbol{p})]}, \quad (4.12)$$

where

$$E_0(\rho,\boldsymbol{p}) \triangleq -\log_2\sum_{j=0}^{1}\left(\sum_{i=0}^{1} p_i\Pr(r=j|v=i)^{1/(1+\rho)}\right)^{1+\rho}$$

and

$$E_1(\rho,\boldsymbol{p}) \triangleq -\log_2\left[\sum_{j=0}^{1}\left(\sum_{i=0}^{1} p_i\Pr(r=j|v=i)\right)\left(\sum_{i=0}^{1} p_i\Pr(r=j|v=i)^{1/(1+\rho)}\right)^{\rho}\right].$$

Inequality (4.12) provides an upper probability bound for a top path ending at level ℓ being early-eliminated. Based on (4.12), we can proceed to derive the bound for the probability $P_{e,E}$ that an incorrect codeword is claimed at the end of the sequential-type search because the correct path is early-eliminated during the decoding process.

Without loss of generality, assume that the all-zero codeword $\mathbf{0}$ is transmitted. Then,

$$\begin{aligned} P_{e,E} = P_{e,E}(\Delta) &\leq \Pr\left(\bigcup_{\ell=1}^{L-\Delta} \mathbf{0}_{n\ell-1} \text{ is early-eliminated}\right) \\ &\leq \sum_{\ell=1}^{L-\Delta} 2^{-\ell n[-\rho R+E_0(\rho)]} 2^{-\Delta n[-\rho R+\rho\log_2(1-\epsilon)/(1+\rho)+E_1(\rho)]}, \end{aligned} \tag{4.13}$$

where the last inequality follows from (4.12) by taking $\boldsymbol{p} = \boldsymbol{p}^*$, and the observations that $\beta - \ell \geq \Delta$ and $[-\rho R + \rho\log_2(1-\epsilon)/(1+\rho) + E_1(\rho)]$ is non-negative subject to $E_0(\rho) > \rho R$. Denoting $\lambda \triangleq E_0(\rho) - \rho R$, we continue the derivation from (4.13):

$$\begin{aligned} P_{e,E} &\leq 2^{-\Delta n[-\rho R+\rho\log_2(1-\epsilon)/(1+\rho)+E_1(\rho)]} \sum_{\ell=1}^{L-\Delta} 2^{-\ell n\lambda} \\ &\leq 2^{-\Delta n[-\rho R+\rho\log_2(1-\epsilon)/(1+\rho)+E_1(\rho)]} \sum_{\ell=1}^{\infty} 2^{-\ell n\lambda} \\ &= K_n \cdot 2^{-\Delta n[-\rho R+\rho\log_2(1-\epsilon)/(1+\rho)+E_1(\rho)]}, \end{aligned} \tag{4.14}$$

where $K_n = 2^{-n\lambda}/(1-2^{-n\lambda})$ is a constant, independent of Δ. Consequently,

$$\begin{aligned} \liminf_{n\to\infty} -\frac{1}{n}\log_2 P_{e,E} &\geq \Delta[-\rho R + \rho\log_2(1-\epsilon)/(1+\rho) + E_1(\rho)] + \lambda \\ &\geq \Delta[-\rho R + \rho\log_2(1-\epsilon)/(1+\rho) + E_1(\rho)], \end{aligned}$$

subject to $E_0(\rho) > \rho R$ with $0 \leq \rho \leq 1$, which immediately implies:

$$\liminf_{n\to\infty} -\frac{1}{n}\log_2 P_{e,E} \geq \Delta \cdot E_{el}(R),$$

where $E_{el}(R) \triangleq \max_{\{\rho\in[0,1]\ :\ E_0(\rho)>\rho R\}}[-\rho R + \rho\log_2(1-\epsilon)/(1+\rho) + E_1(\rho)]$. Following similar argument as [9], we conclude that the additional error due to early elimination in the MLSDA becomes exponentially negligible if

$$\Delta \cdot E_{el}(R) > (m+1)E_c(R), \quad \text{or equivalently,} \quad \Delta/(m+1) > E_c(R)/E_{el}(R) \tag{4.15}$$

for convolutional code rates above the channel cutoff rate R_0, where

$$E_c(R) \triangleq \max_{\{\rho\in[0,1]\ :\ E_0(\rho)>\rho R\}} E_0(\rho),$$

since $(m+1)E_c(R)$ is the exact error exponent of the maximum-likelihood decoding error for convolutional codes for $R \geq R_0$ [36].

4.3 Numerical and Simulation Results

By choosing $\epsilon = 0.045$ and $\epsilon = 0.095$ to approach the desired cutoff rate 1/2 and 1/3, it can be observed from numerical plots in Figs. 4.3 and 4.4 (or directly derived from (4.15)) that the suggested early elimination windows are:

$$\Delta > \frac{0.4996}{0.2271} \times (m+1) \approx 2.1999(m+1) \text{ for rate } 1/2 \text{ codes;} \tag{4.16}$$

$$\Delta > \frac{0.3342}{0.2816} \times (m+1) \approx 1.1868(m+1) \text{ for rate } 1/3 \text{ codes.} \tag{4.17}$$

Condition (4.16) and (4.17) indicate that for (2,1,12) and (3,1,8) convolutional codes, taking $\Delta = 29$ and $\Delta = 11$ respectively should suffice to result in negligible performance degradation at the cutoff rate. It can be observed respectively from the simulations in Figs. 4.5 and 4.6 that the MLSDA with early elimination window $\Delta = 30$ for (2,1,12) convolutional code and $\Delta = 11$ for (3,1,8) convolutional code exhibit negligible performance degradation for all E_b/N_0's simulated, where we take $\epsilon = \frac{1}{2}\text{erfc}(\sqrt{E_b/N_0})$ as a convention, and $\text{erfc}(\cdot)$ is the complementary error function. For comparison, the performance of the stack algorithm with the Fano metric is also illustrated.

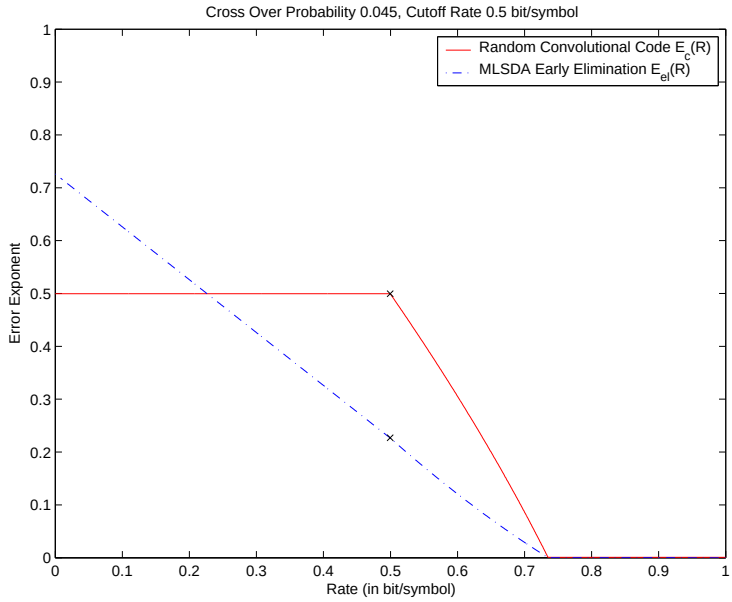

Figure 4.3: Exponent lower bound $E_{el}(R)$ of the additional error due to early elimination and exponent $E_c(R)$ of the maximum-likelihood decoding error for time-varying convolutional codes (without early elimination) under the BSC with crossover probability 0.045.

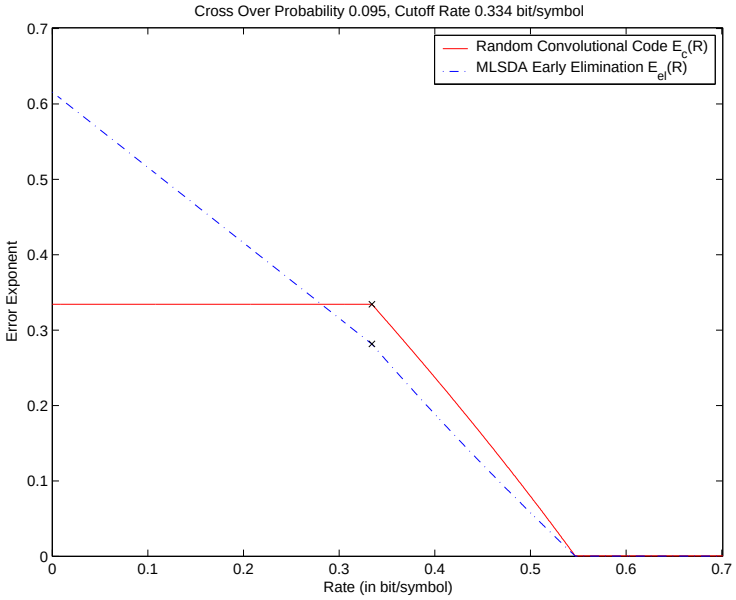

Figure 4.4: Exponent lower bound $E_{el}(R)$ of the additional error due to early elimination and exponent $E_c(R)$ of the maximum-likelihood decoding error for time-varying convolutional codes (without early elimination) under the BSC with crossover probability 0.095.

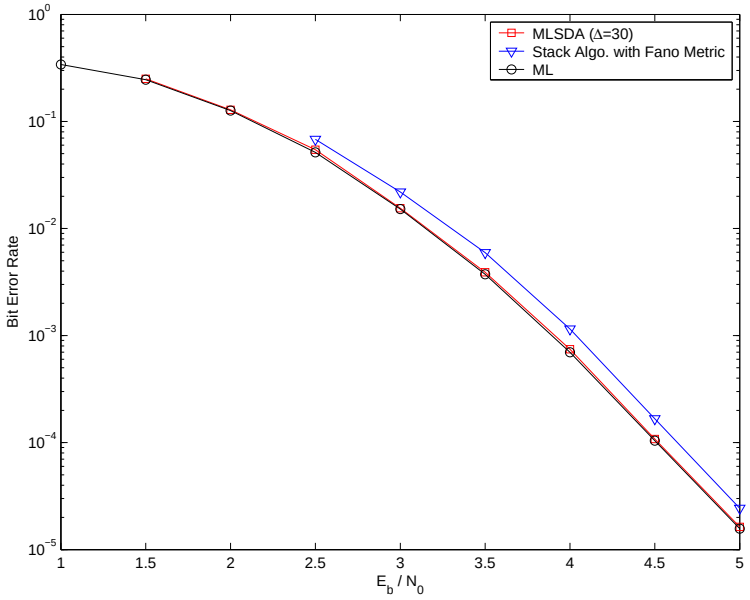

Figure 4.5: Performance for (2,1,12) convolutional codes for maximum-likelihood (ML) decoder, stack algorithm with Fano metric, and MLSDA with early elimination window $\Delta = 30$ under BSC. The generator polynomial of the code is [42554 77304] in octal. The message length $L = 500$.

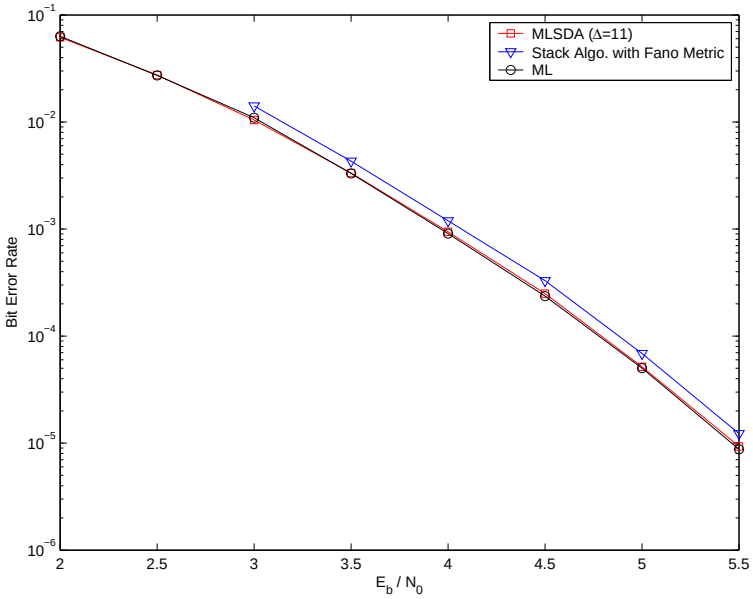

Figure 4.6: Performance for (3,1,8) convolutional codes for maximum-likelihood (ML) decoder, stack algorithm with Fano metric, and MLSDA with early elimination window $\Delta = 11$ under BSC. The generator polynomial of the code is [557 663 711] in octal. The message length $L = 500$.

Chapter 5

Analysis of the Window Size with Negligible Performance Degradation over AWGN Channels

In this chapter, the analysis on the early elimination window that suffices to provide near optimal performance over AWGN channels is provided. The road map is as follows. In Section 5.1, we obtain a union bound of block error rate (BLER) for convolutional codes with finite length. This upper bound is accurate at high SNRs, and hence, can be used as a faithful replacement of the BLERs. In Section 5.2, an upper bound for additional BLER due to the introduction of early elimination is established. In Section 5.3, the suggestive values of early elimination window Δ for negligible performance degradation is obtained based on the analytic bounds in the previous two sections. Simulation results are also shown to validate the suggested values.

5.1 Block Error Rate Analysis for Finite-Length Convolutional Codes with ML Decoder

For a convolutional code with specific code rate and generator polynomial, the weight enumerator function (WEF) $A(x)$ is defined as [25]

$$A(x) = \sum_{d=d_{\text{free}}}^{\infty} A_d x^d,$$

where d_{free} is the free distance and A_d denote the number of codewords with weight d. The WEF can be calculated by Mason's gain formula or by computers. For example, the WEF for the (2,1,6) convolutional code with generator polynomial [554,744] (in octal) is

$$A(x) = 11x^{10} + 38x^{12} + 193x^{14} + 1331x^{16} + 7275x^{18} + 40406x^{20} + \cdots,$$

and it means that there are 11 codewords with weight 10, 38 codewords with weight 12, 193 codewords with weight 14, and so on. For convolution decoding, we say that the a *first event error* is made at time unit t if the correct path is eliminated (in, e.g., Viterbi decoder) for the first time at time unit t. For infinite length convolutional codes, the first event error probability, denoted by P_{ev}, is shown to be independent of t. For binary-input AWGN channels, P_{ev} can be bounded above as follows: (c.f. equation (12.20) and (12.44a) in [25])

$$P_{ev} \leq \sum_{d=d_{free}}^{\infty} A_d Q\left(\sqrt{2dR\frac{E_b}{N_0}}\right), \tag{5.1}$$

where R is the code rate, E_b/N_0 is the signal-to-noise ratio per information bit, and $Q(x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$. Notably, the union upper bound in (5.1) is also valid for the finite-length convolutional code since the competitor paths (that compete with the correct path) of the finite-length convolutional code are only sub-portions of the competitor paths of the corresponding infinite-length convolutional code.

Based on the first event error probability P_{ev}, we can derive a "union-type" upper bound for block error rate (BLER) for convolutional codes with finite information length L as

$$\text{BLER} \leq P_B = L \cdot P_{ev}. \tag{5.2}$$

It should be mentioned that the union-type bounds are generally accurate at high SNRs, which can be substantiated by simulations. Figures 5.1 and 5.2 summarize P_B in (5.2) and the simulation results of BLERs for (2,1,6) and (2,1,10) convolutional codes, respectively, with information length $L = 200$. The two figures show that at SNR higher than 4 dB, P_B is almost indifferent to the true BLER. We hereafter use P_B in (5.2) as an approximate of BLER for finite-length convolutional codes in determining the early-elimination window as what we concern in this work is medium to high SNRs.

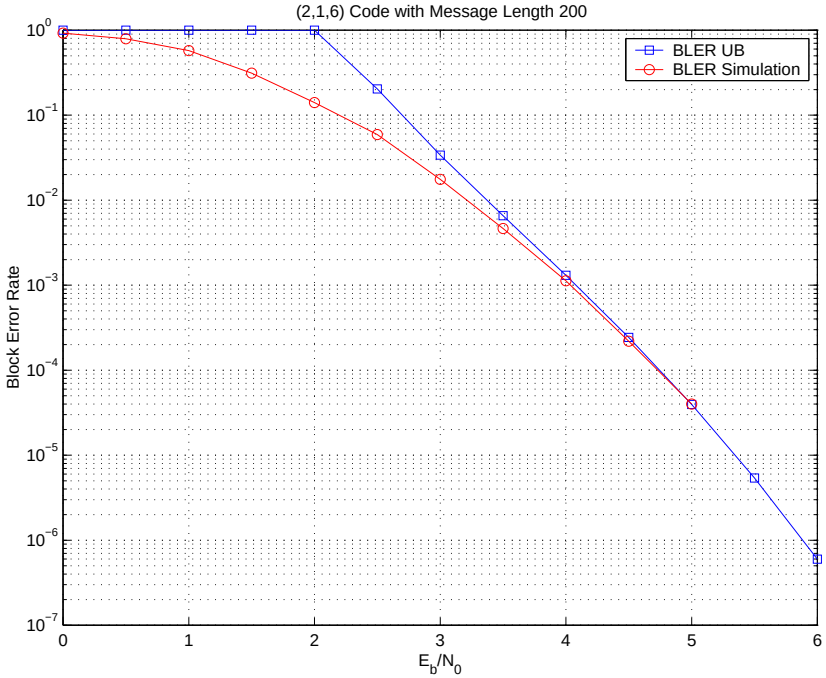

Figure 5.1: Block error rate upper bound (BLER UB) given by (5.2) and simulated BLER for $(2, 1, 6)$ convolutional code under AWGN channels.

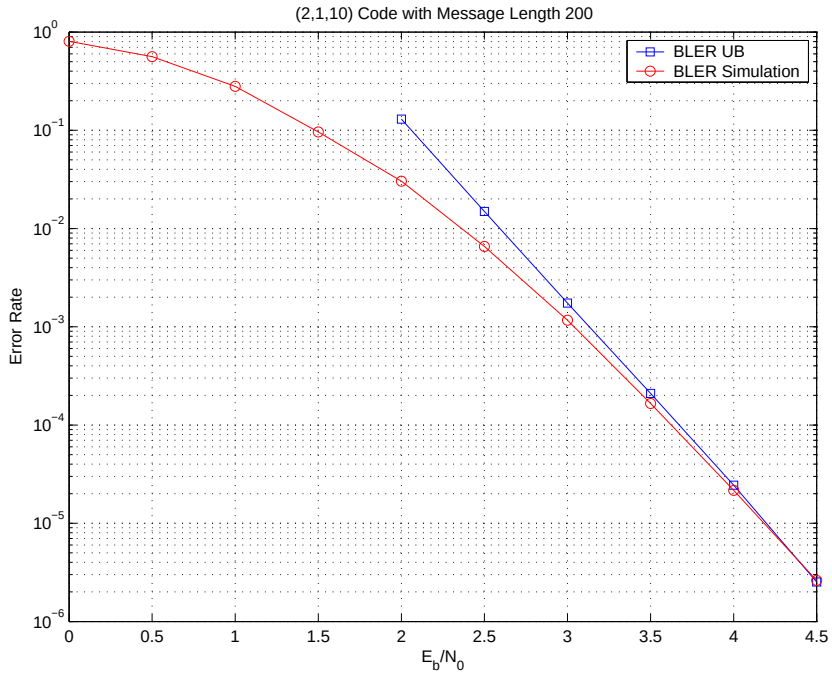

Figure 5.2: Block error rate upper bound (BLER UB) given by (5.2) and simulated BLER for $(2, 1, 10)$ convolutional code under AWGN channels.

5.2 Moment Generating Function Bound of Additional Error Due to Early Elimination

In the derivation of the error rate, we may assume without loss of generality that the all-zero codeword $\mathbf{0}$ is transmitted. Then, the block error occurs when either (i) the all-zero code path is not maximum-likelihood (ML), or (ii) any offspring of the all-zero code path is early eliminated (EE). This observation results in that the BLER for the MLSDA with early

elimination is upper-bounded by:

$$
\begin{aligned}
&\Pr\left([\mathbf{0} \text{ not ML}] \text{ or } [\mathbf{0} \text{ EE}]\right) \\
&= \Pr\left(\mathbf{0} \text{ not ML}\right) + \Pr\left(\mathbf{0} \text{ EE and } \mathbf{0} \text{ ML}\right) \\
&\leq \Pr\left(\mathbf{0} \text{ not ML}\right) + \Pr\left(\bigcup_{\ell_e=1}^{L+m-\Delta} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE and } \mathbf{0} \text{ ML}\right]\right) \\
&= \Pr\left(\mathbf{0} \text{ not ML}\right) + \Pr\left(\bigcup_{\ell_e=1}^{L+m-\Delta} \bigcup_{\ell_b=0}^{\ell_e-1} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(n\ell_b)} \text{ and } \mathbf{0} \text{ ML}\right]\right) \\
&= \Pr\left(\mathbf{0} \text{ not ML}\right) + \Pr\left(\bigcup_{\ell_b=0}^{L+m-\Delta-1} \bigcup_{\ell_e=\ell_b+1}^{L+m-\Delta} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(n\ell_b)} \text{ and } \mathbf{0} \text{ ML}\right]\right) \\
&\leq \Pr\left(\mathbf{0} \text{ not ML}\right) + \sum_{\ell_b=0}^{L+m-\Delta-1} \Pr\left(\bigcup_{\ell_e=\ell_b+1}^{L+m-\Delta} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(n\ell_b)} \text{ and } \mathbf{0} \text{ ML}\right]\right) \\
&\leq \Pr\left(\mathbf{0} \text{ not ML}\right) + \sum_{\ell_b=0}^{L+m-\Delta-1} \Pr\left(\bigcup_{\ell_e=1}^{L+m-\Delta} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(0)} \text{ and } \mathbf{0} \text{ ML}\right]\right) \quad (5.3) \\
&= \Pr\left(\mathbf{0} \text{ not ML}\right) + (L+m-\Delta) \Pr\left(\bigcup_{\ell_e=1}^{L+m-\Delta} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(0)} \text{ and } \mathbf{0} \text{ ML}\right]\right), \quad (5.4)
\end{aligned}
$$

where

$$\tilde{\mathbf{0}}_{(n\ell_b)} = (\underbrace{0, 0, \ldots, 0}_{n\ell_b \text{ of them}}, 1),$$

and (5.3) follows from the fact that the probability of

$$\Pr\left(\bigcup_{\ell_e=\ell_b+1}^{L+m-\Delta} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(n\ell_b)} \text{ and } \mathbf{0} \text{ ML}\right]\right)$$

is non-increasing in ℓ_b as the set of competitor paths that possibly eliminate the all-zero path for a larger ℓ_b is a subset of those for a smaller ℓ_b. Notably, in the above derivation, the event [**0** not ML] includes those cases that the all-zero path has been eliminated due to other equal-metric path even if it has the smallest metric (cf. Footnote 1.) Similarly, [**0** ML] includes those cases that the smallest-metric all-zero path survives the random choice in footnote 1 whenever it merges with some other equal-metric paths.

The first term in (5.4) is the BLER of the maximum-likelihood decision, while the second term can be regarded as a probability upper bound of additional BLER due to early elimination. Observe that the second term is non-increasing in Δ. Therefore, if Δ is chosen to be sufficiently large such that the additional BLER is negligibly less than the BLER of the maximum-likelihood decision, the desired near optimal performance is obtained.

Before calculating the second term in (5.4), we notice that the offspring of $\tilde{\mathbf{0}}_{(n\ell_b)}$, which causes the early-elimination of $\mathbf{0}_{(n\ell_e-1)}$, will never re-visit the zero state after level ℓ_b. This is because if the offspring of $\tilde{\mathbf{0}}_{(n\ell_b)}$ merges with the all-zero path at some level ℓ with $\ell > \ell_b$, and survives after the conduction of Step 3 in the MLSDA (so that it can later cause the early-elimination of $\mathbf{0}_{(n\ell_e-1)}$), its metric up to the merged level ℓ must be smaller than that of $\mathbf{0}_{(n\ell-1)}$, which indicates the violation of event [$\mathbf{0}$ ML] (that includes the situation that $\mathbf{0}$ wins the random pick in footnote 1).

Now, in lie of this observation, we let $\boldsymbol{P}_{\ell_e+\Delta}$ be the set of paths of length $n(\ell_e+\Delta)$, which diverge from $\mathbf{0}_{(n\ell_e-1)}$ at level 0, and never re-visit the zero state. Denote by $\boldsymbol{x}_{(n(\ell_e+\Delta)-1)}$ one particular path in $\boldsymbol{P}_{\ell_e+\Delta}$ with weight d_1 in the first $n\ell_e$ bits and weight d_2 in the last $n\Delta$

bits. Then, The probability that $\mathbf{0}_{(n\ell_e-1)}$ is early eliminated by $\boldsymbol{x}_{(n(\ell_e+\Delta)-1)}$ is

$$\begin{aligned}
&\Pr\left(\mathbf{0}_{(n\ell_e-1)} \text{ EE due to } \boldsymbol{x}_{(n(\ell_e+\Delta)-1)} \text{ and } \mathbf{0} \text{ ML}\right) \\
&\leq \Pr\left(\mathbf{0}_{(n\ell_e-1)} \text{ EE due to } \boldsymbol{x}_{(n(\ell_e+\Delta)-1)}\right) \\
&= \Pr\left(\mu(\boldsymbol{x}_{(n(\ell_e+\Delta)-1)}) \leq \mu(\mathbf{0}_{(n\ell_e-1)})\right) \\
&= \Pr\left(\sum_{j=0}^{n(\ell_e+\Delta)-1} (y_j \oplus x_j)|\phi_j| \leq \sum_{j=0}^{n\ell_e-1} (y_j \oplus 0)|\phi_j|\right) \\
&= \Pr\left(\sum_{j\in\mathcal{J}(\boldsymbol{x}_{(n(\ell_e+\Delta)-1)})} (1-2y_j)|\phi_j| + \sum_{j=n\ell_e}^{n(\ell_e+\Delta)-1} y_j|\phi_j| \leq 0\right) \\
&= \Pr\left(\sum_{j\in\mathcal{J}(\boldsymbol{x}_{(n(\ell_e+\Delta)-1)})} \phi_j + \frac{1}{2}\sum_{j=n\ell_e}^{n(\ell_e+\Delta)-1} (|\phi_j| - \phi_j) \leq 0\right) \\
&= \Pr\left(\sum_{j\in\mathcal{J}(\boldsymbol{x}_{(n(\ell_e+\Delta)-1)})} 2r_j + \sum_{j=n\ell_e}^{n(\ell_e+\Delta)-1} (|r_j| - r_j) \leq 0\right), \qquad (5.5)
\end{aligned}$$

where $\mathcal{J}(\boldsymbol{x}_{(b)}) = \{j : x_j = 1\}$ is the set of indices in $\boldsymbol{x}_{(b)} = (x_0, x_1, \ldots, x_b)$ such that $x_j = 1$, and (5.5) follows from $\phi_j \triangleq \log[\Pr(r_j|v_j = 0)/\Pr(r_j|v_j = 1)] = (4\sqrt{\mathcal{E}}/N_0)r_j$.

Under the assumption that $\boldsymbol{x}_{(n(\ell_e+\Delta)-1)}$ has weight d_1 in the first $n\ell_e$ bits and weight d_2 in the last $n\Delta$ bits, and also the observation that $\{r_j\}_{j=1}^N$ is independent and identically Gaussian distributed with mean $\mathcal{E}$ and variance $N_0/2$ with respect to the transmitted $\mathbf{0}$, the probability in (5.5) is in turn equal to

$$\begin{aligned}
&\Pr\left(\sum_{j=1}^{d_1} 2r_j + \sum_{j=d_1+1}^{d_1+d_2} (|r_j| + r_j) + \sum_{j=d_1+d_2+1}^{d_1+n\Delta} (|r_j| - r_j) \leq 0\right) \\
&= \Pr\left(\sum_{j=1}^{d_1} r_j + \sum_{j=d_1+1}^{d_1+d_2} r_j^+ + \sum_{j=d_1+d_2+1}^{d_1+n\Delta} r_j^- \leq 0\right) \\
&= \Pr\left(\sum_{j=1}^{d_1} (-r_j) + \sum_{j=d_1+1}^{d_1+d_2} \left(-r_j^+\right) + \sum_{j=d_1+d_2+1}^{d_1+n\Delta} \left(-r_j^-\right) \geq 0\right), \qquad (5.6)
\end{aligned}$$

where

$$r_j^+ = \begin{cases} r_j, & \text{if } r_j > 0; \\ 0, & \text{elsewhere,} \end{cases} \qquad (5.7)$$

and

$$r_j^- = \begin{cases} 0, & \text{if } r_j > 0; \\ -r_j, & \text{elsewhere.} \end{cases} \tag{5.8}$$

The moment generating functions $M(t)$, $M_+(t)$ and $M_-(t)$ of $(-r_j)$, $(-r_j^+)$ and $(-r_j^-)$ are respectively given by:

$$\begin{aligned} M(t) &= \int_{-\infty}^{\infty} e^{tx} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(x+\mathcal{E})^2}{N_0}\right) dx \\ &= \exp\left(\frac{N_0 t^2}{4} - \mathcal{E}t\right) \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(x+\mathcal{E}-\frac{N_0 t}{2})^2}{N_0}\right) dx \\ &= \exp\left(\frac{N_0 t^2}{4} - \mathcal{E}t\right), \end{aligned}$$

$$\begin{aligned} M_+(t) &= Q\left(\frac{\mathcal{E}}{\sqrt{N_0/2}}\right) + \int_{-\infty}^{0} e^{tx} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(x+\mathcal{E})^2}{N_0}\right) dx \\ &= Q\left(\frac{\mathcal{E}}{\sqrt{N_0/2}}\right) + \exp\left(\frac{N_0 t^2}{4} - \mathcal{E}t\right) \int_{-\infty}^{\frac{2\mathcal{E}-N_0 t}{\sqrt{2N_0}}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz \\ &= Q\left(\frac{\mathcal{E}}{\sqrt{N_0/2}}\right) + \exp\left(\frac{N_0 t^2}{4} - \mathcal{E}t\right) Q\left(\frac{N_0 t - 2\mathcal{E}}{\sqrt{2N_0}}\right), \end{aligned}$$

and

$$\begin{aligned} M_-(t) &= 1 - Q\left(\frac{\mathcal{E}}{\sqrt{N_0/2}}\right) + \int_{-\infty}^{0} e^{tx} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(x-\mathcal{E})^2}{N_0}\right) dx \\ &= 1 - Q\left(\frac{\mathcal{E}}{\sqrt{N_0/2}}\right) + \exp\left(\frac{N_0 t^2}{4} + \mathcal{E}t\right) \int_{-\infty}^{\frac{-2\mathcal{E}-N_0 t}{\sqrt{2N_0}}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz \\ &= 1 - Q\left(\frac{\mathcal{E}}{\sqrt{N_0/2}}\right) + \exp\left(\frac{N_0 t^2}{4} + \mathcal{E}t\right) Q\left(\frac{N_0 t + 2\mathcal{E}}{\sqrt{2N_0}}\right). \end{aligned}$$

The probability that $\mathbf{0}_{(n\ell_e-1)}$ is early-eliminated by one particular path $\boldsymbol{x}_{(n(\ell_e+\Delta)-1)}$ in $\boldsymbol{P}_{\ell_e+\Delta}$ is therefore upper-bounded by the moment generating bound as:

$$\Pr\left(\sum_{j=1}^{d_1}(-r_j) + \sum_{j=d_1+1}^{d_1+d_2}\left(-r_j^+\right) + \sum_{j=d_1+d_2+1}^{d_1+n\Delta}\left(-r_j^-\right) \geq 0\right) \leq [M(t)]^{d_1} \left[M_+(t)\right]^{d_2} \left[M_-(t)\right]^{n\Delta-d_2}$$

for $t > 0$.

Define the weight distribution function for all paths in $\boldsymbol{P}_{\ell_e+\Delta}$ by

$$W(\ell_e, \Delta) = \sum_{d_1,d_2} A_{d_1,d_2}(\ell_e, \Delta) \cdot w_1^{d_1} \cdot w_2^{d_2}, \tag{5.9}$$

where $A_{d_1,d_2}(\ell_e, \Delta)$ is the number of paths with weight d_1 in the first $n\ell_e$ bits and weight d_2 in the last $n\Delta$ bits. This function can be established by computers. For example, for (2,1,6) convolutional code with generator polynomial [554,744] (in octal), we have:

$$W(2,3) = 2w_1^3w_2 + 3w_1^3w_2^2 + 6w_1^3w_2^3 + 4w_1^3w_2^4 + w_1^3w_2^6;$$
$$W(3,2) = w_1^3w_2 + 2w_1^3w_2^2 + w_1^3w_2^3 + w_1^4 + w_1^4w_2 + 3w_1^4w_2^2 + 3w_1^4w_2^3 + 2w_1^5w_2 + w_1^5w_2^2 + w_1^5w_2^4;$$
$$W(3,5) = 2w_1^3w_2^2 + 2w_1^3w_2^3 + 8w_1^3w_2^4 + \cdots + 2w_1^4w_2 + w_1^4w_2^2 + 8w_1^4w_2^3 + \cdots + 3w_1^5w_2^2 + \cdots .$$

As a consequence, the second term in (5.4) can be bounded above by

$$\begin{aligned}
&\Pr\left(\bigcup_{\ell_e=1}^{L+m-\Delta} \left[\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(0)} \text{ and } \mathbf{0} \text{ ML}\right]\right) \\
&\leq \sum_{\ell_e=1}^{L+m-\Delta} \Pr\left(\mathbf{0}_{(n\ell_e-1)} \text{ EE due to an offspring of } \tilde{\mathbf{0}}_{(0)} \text{ and } \mathbf{0} \text{ ML}\right) \\
&\leq \sum_{\ell_e=1}^{L+m-\Delta} \sum_{d_1,d_2} A_{d_1,d_2}(\ell_e, \Delta) \cdot \min_{t>0} \left[M(t)^{d_1} \cdot M_+(t)^{d_2} \cdot M_-(t)^{n\Delta-d_2}\right].
\end{aligned} \tag{5.10}$$

5.3 Numerical and Simulation Results

In the previous two sections, we obtain that the first term in (5.4) can be well approximated by (5.2), and the second term can be bounded above by the moment generating bound in (5.10). The numerical values of these bounds respectively for $(2,1,6)$, $(2,1,8)$, $(2,1,10)$, $(2,1,12)$ and $(3,1,8)$ convolutional codes are plotted in Figs. 5.3, 5.4, 5.5, 5.6 and 5.7.

Define the sufficiently large window size Δ to be the least integer such that the BLER upper bound of the MLSDA with early elimination (BLER UB MLSDA/EE) is less than 0.1 dB inferior to the BLER upper bound (BLER UB) of the ML decoding at BLER UB=10^{-5}.

The resultant sufficiently large window sizes are summarized in Tables 5.1 and 5.2 for rate one-half and rate one-third convolutional codes, respectively. We found that the window sizes suggested by the upper bounds are close to those obtained by simulations, and therefore can be used as a guide to predict the near-optimal window size (especially when simulations for some codes are not available).

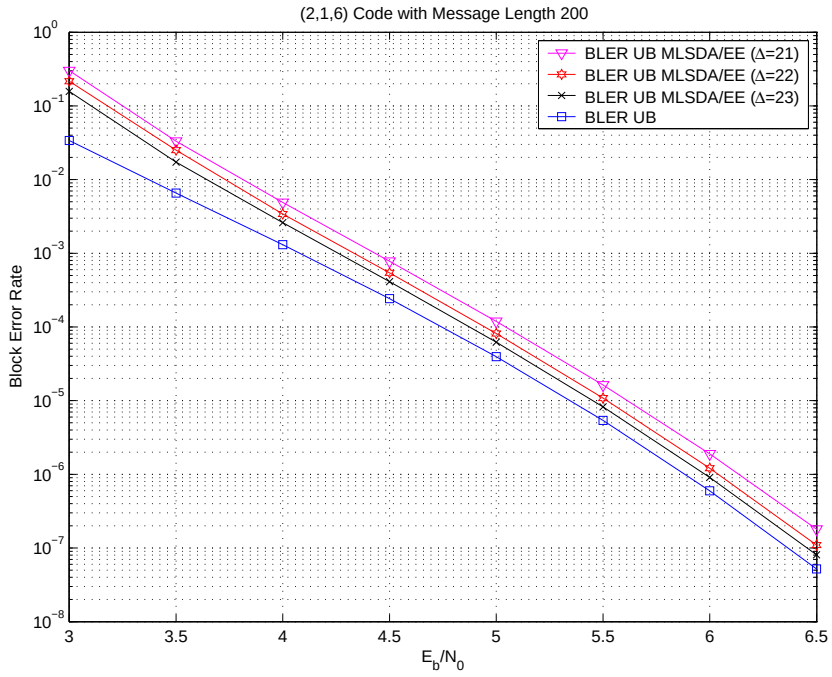

Figure 5.3: Block error rate upper bounds for $(2, 1, 6)$ convolutional codes with $L = 200$.

Table 5.1: Suggested Δ values for rate one-half convolutional codes

Memory Order m	6	8	10	12	14	16
Generator Polynomial	554 744	561 753	4672 7542	42554 77304	56721 61713	716502 514576
$d_{\min}$	10	12	14	16	18	20
Sufficiently Large Δ by UB	23	29	33	38	42	46
Sufficiently Large Δ by Simulations	20	24	28	32	36	40

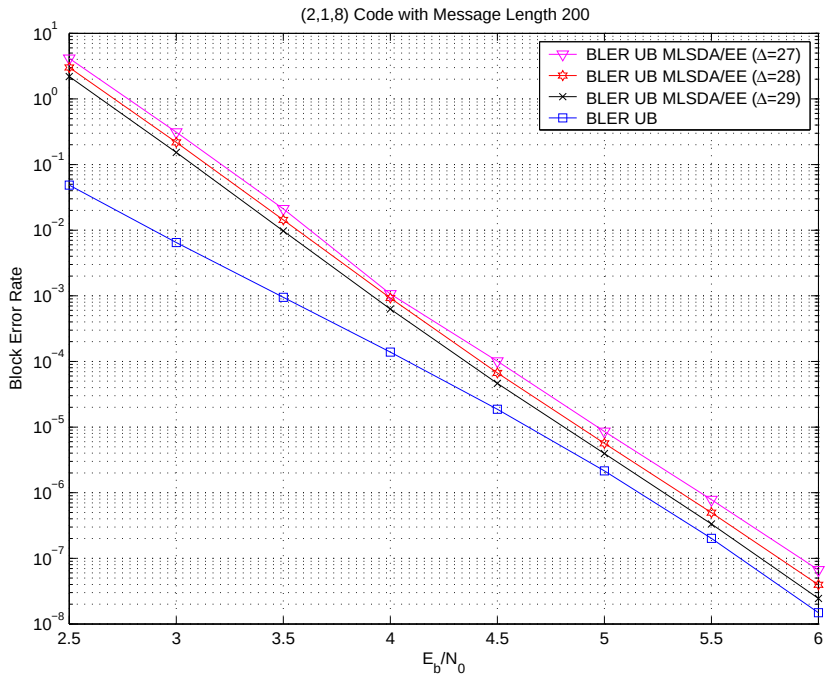

Figure 5.4: Block error rate upper bound for $(2, 1, 8)$ convolutional codes with $L = 200$.

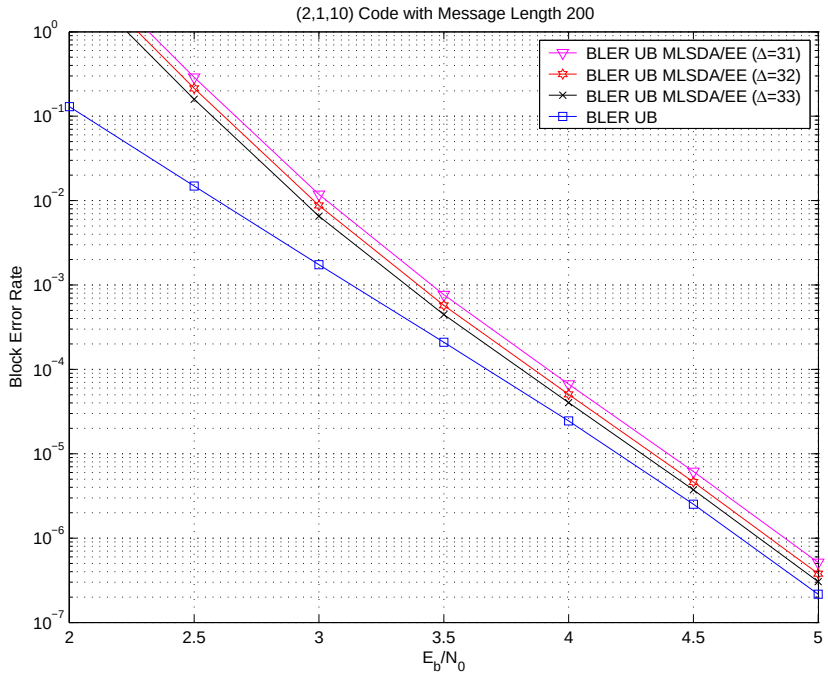

Figure 5.5: Block error rate upper bounds for $(2, 1, 10)$ convolutional codes with $L = 200$.

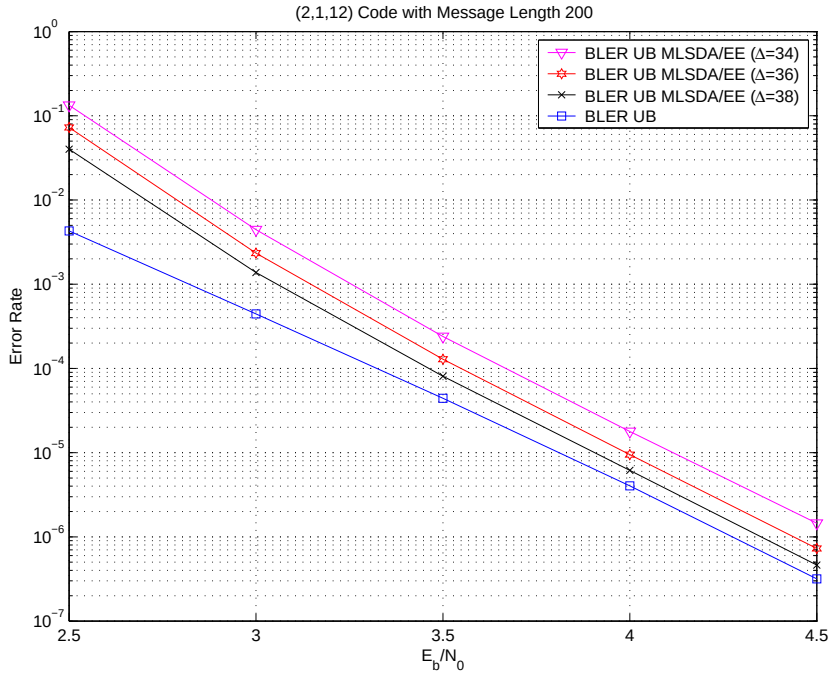

Figure 5.6: Block error rate upper bounds for $(2, 1, 12)$ convolutional codes with $L = 200$.

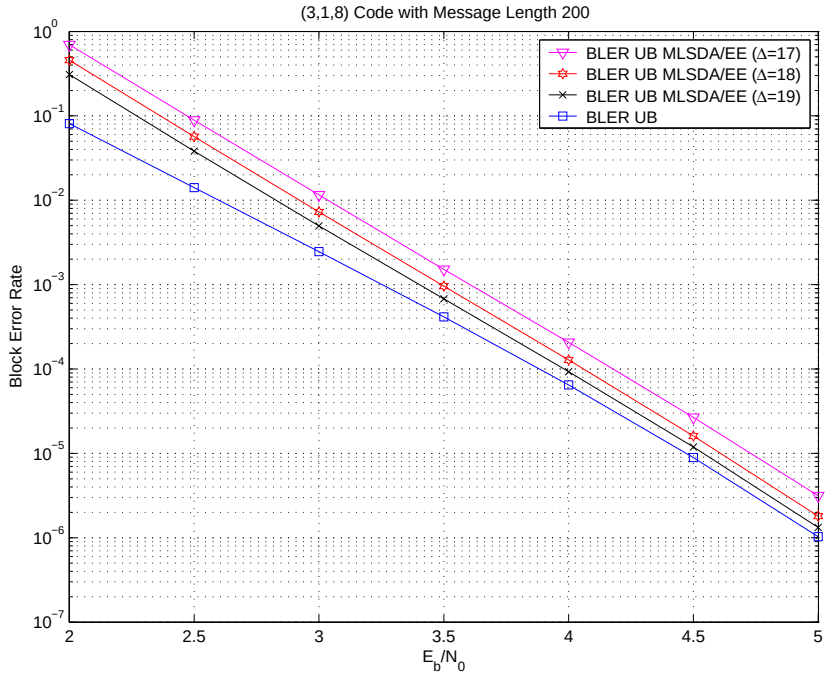

Figure 5.7: Block error rate upper bounds for $(3, 1, 8)$ convolutional codes with $L = 200$.

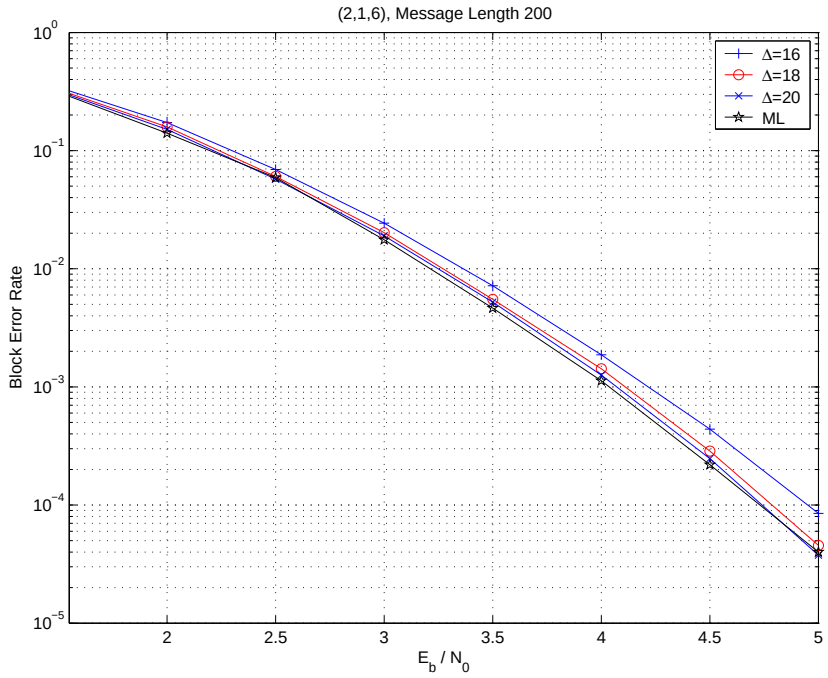

Figure 5.8: Simulated block error rates for $(2, 1, 6)$ convolutional codes with $L = 200$.

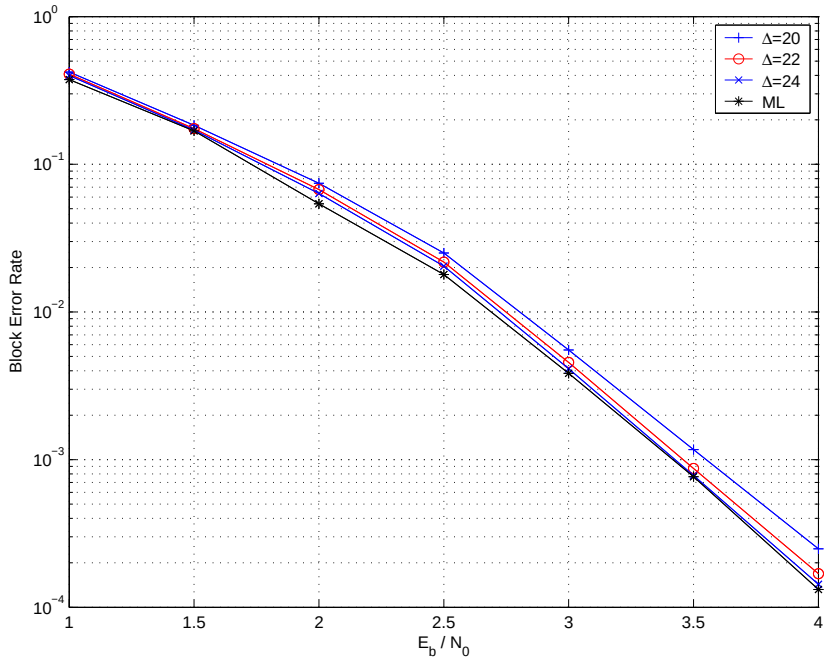

Figure 5.9: Simulated block error rates for $(2, 1, 8)$ convolutional codes with $L = 200$.

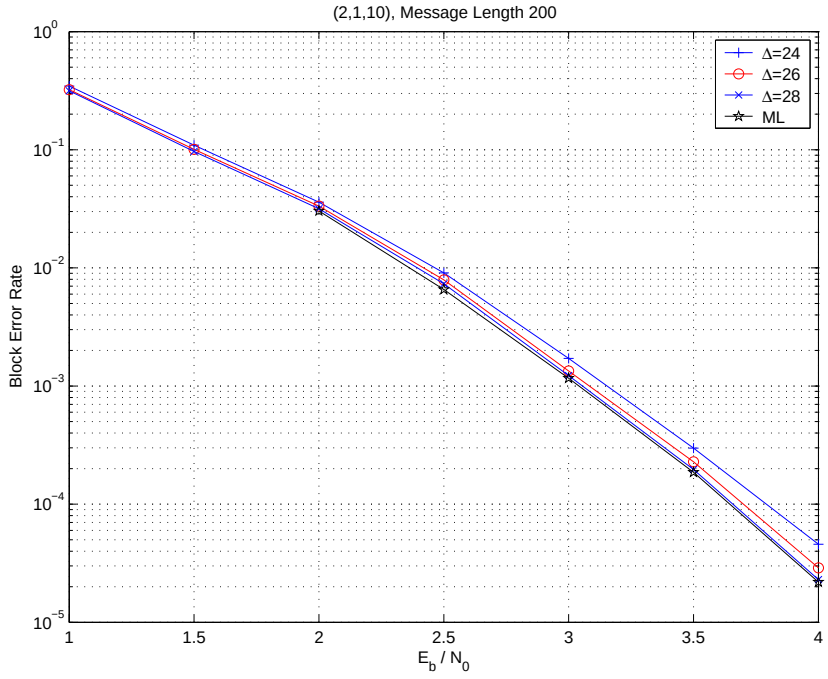

Figure 5.10: Simulated block error rates for $(2, 1, 10)$ convolutional codes for $L = 200$.

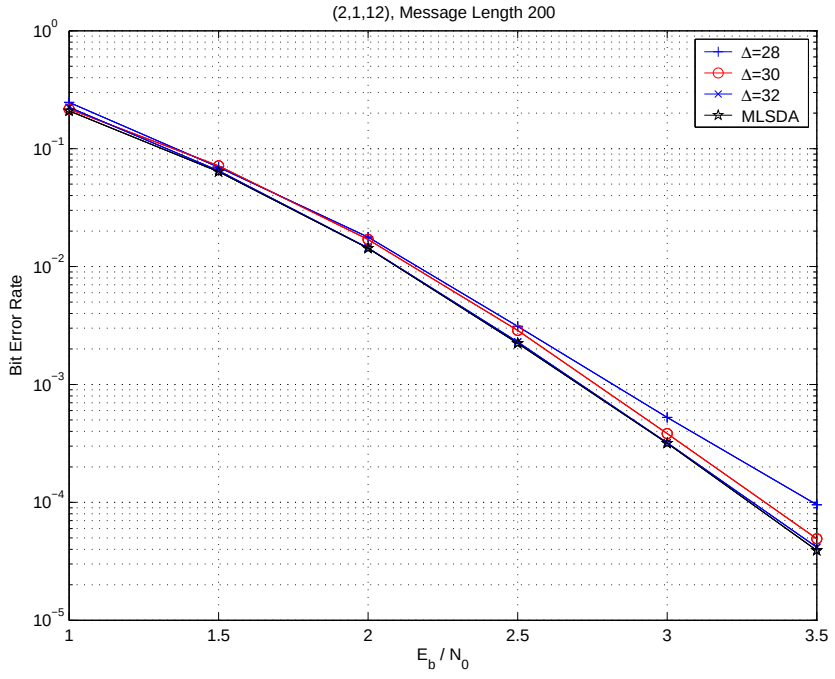

Figure 5.11: Simulated block error rates for $(2, 1, 12)$ convolutional codes for $L = 200$.

Table 5.2: Suggested Δ values for rate one-third convolutional codes

Memory Order m	6	8	10
Generator Polynomial	554 624 764	557 663 711	4726 5562 6372
Minimum Distance $d_{\min}$	15	19	22
Sufficiently Large Δ by UB	14	18	23
Sufficiently Large Δ by Simulation	12	16	20

Chapter 6

Analysis of the Computational Efforts of MLSDA and MLSDA with Early Elimination

In this chapter, the large deviations technique and the Berry-Esseen theorem [7, sec.XVI. 5] are utilized to estimate the computational complexity of the MLSDA with and without early elimination. The large deviations technique is generally applied to compute the *exponent* of an exponentially decaying probability mass. In order to obtain a better computational complexity upper bound, we also apply the Berry-Esseen inequality to evaluate the *subexponential* detail of the concerned probability. Simulations show that the resultant complexity upper bounds are close to the true complexity at both low and high SNRs.

6.1 Berry-Esseen Theorem and Probability Bound

The Berry-Esseen theorem [7, sec.XVI. 5] states that the distribution of the sum of independent zero-mean random variables $\{X_i\}_{i=1}^n$, normalized by the standard deviation of the sum, differs from the unit Gaussian distribution by no more than $C\, r_n/s_n^3$, where s_n^2 and r_n are, respectively, the sums of the marginal variances and the marginal absolute third moments,

and the Berry-Esseen coefficient, C, is an absolute constant. Specifically, for every $a \in \Re$,

$$\left| \Pr\left\{ \frac{1}{s_n} \left(X_1 + \cdots + X_n\right) \leq a \right\} - \Phi(a) \right| \leq C \, \frac{r_n}{s_n^3}, \tag{6.1}$$

where $\Phi(\cdot)$ represents the unit Gaussian cumulative distribution function (cdf). A typical estimate of the absolute constant is $C = 6$ [7, sec.XVI. 5, Thm. 2]. When $\{X_n\}_{i=1}^n$ are identically distributed, in addition to independent, the absolute constant can be reduced to $C = 3$, and has been reported to be improved down to 2.05 [7, sec.XVI. 5, Thm. 1]. Shiganov improved the absolute constant down to 0.7915 for an independent sample sum, and, 0.7655, if these samples are also identically distributed [34]. Shiganov's result is generally considered to be the best result yet obtained thus far [32].

Based on the Berry-Esseen inequality, we first derive an upper probability bound for the sum of independent, but non-Gaussian random variables, and later use this bound to analyze computational efforts.

Lemma 1. *Let $Y_n = \sum_{i=1}^n X_i$ be the sum of i.i.d. random variables whose marginal distribution is $F(\cdot)$. Define the twisted distribution with parameter θ corresponding to $F(\cdot)$ as:*

$$dF^{(\theta)}(x) \triangleq \frac{\exp\{\theta x\}\, dF(x)}{M(\theta)},$$

where $M(\theta) \triangleq E[e^{\theta X_1}]$. Let the random variable with probability distribution $F^{(\theta)}(\cdot)$ be $X^{(\theta)}$. Then, for every $\theta < 0$,

$$\Pr\left\{Y_n \leq -n\alpha\right\} \leq A_n(\theta, \alpha) e^{\theta \alpha n} M^n(\theta),$$

where $A_n(\theta, \alpha) = \min\{B_n(\theta, \alpha), 1\}$,

$$B_n(\theta, \alpha) \triangleq \begin{cases} \dfrac{\sigma(\theta)}{\sqrt{2\pi n}[(\mu(\theta) + \alpha) - \theta\sigma^2(\theta)]} e^{-(\mu(\theta)+\alpha)^2 n/[2\sigma^2(\theta)]} + 2C \dfrac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}}, & \text{if } \alpha > \theta\sigma^2(\theta) - \mu(\theta); \\ e^{\theta[\theta\sigma^2(\theta) - 2(\mu(\theta)+\alpha)]n/2} + 2C \dfrac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}}, & \text{otherwise,} \end{cases}$$

$$\mu(\theta) = E[X^{(\theta)}], \quad \sigma^2(\theta) = E[|X^{(\theta)} - \mu(\theta)|^2], \quad \rho(\theta) = E[|X^{(\theta)} - \mu(\theta)|^3]$$

and $C = 0.7655$.

Proof. Define $F_n^{(\theta)}(y) = \Pr[X_1^{(\theta)} + X_2^{(\theta)} + \cdots + X_n^{(\theta)} \leq y]$, and let the distribution of $[(X_1^{(\theta)} - \mu(\theta)) + \cdots + (X_n^{(\theta)} - \mu(\theta))]/[\sigma(\theta)\sqrt{n}]$ be $H_n(\cdot)$, where in the evaluation of the above two statistics, $\{X_i^{(\theta)}\}_{i=1}^n$ are assumed independent with common marginal distribution $F^{(\theta)}(\cdot)$. Then, by denoting $Y_n^{(\theta)} = X_1^{(\theta)} + X_2^{(\theta)} + \cdots + X_n^{(\theta)}$, we obtain:

$$\begin{aligned}
\Pr\left(Y_n \leq -n\alpha\right) &= \int_{[x_1+\cdots+x_n \leq -n\alpha]} dF(x_1)dF(x_2)\cdots dF(x_n) \\
&= M^n(\theta) \int_{[x_1+\cdots+x_n \leq -n\alpha]} e^{-\theta(x_1+\cdots+x_n)} dF^{(\theta)}(x_1)dF^{(\theta)}(x_2)\cdots dF^{(\theta)}(x_n) \\
&= M^n(\theta) E\left[e^{-\theta(X_1^{(\theta)}+\cdots+X_n^{(\theta)})} \mathbf{1}\{X_1^{(\theta)} + \cdots + X_n^{(\theta)} \leq -n\alpha\}\right] \\
&= M^n(\theta) E\left[e^{-\theta Y_n^{(\theta)}} \mathbf{1}\{Y_n^{(\theta)} \leq -n\alpha\}\right] \\
&= M^n(\theta) \int_{-\infty}^{-n\alpha} e^{-\theta y} dF_n^{(\theta)}(y) \qquad (y \to \sigma(\theta)\sqrt{n}y' + \mu(\theta)n) \\
&= M^n(\theta) \int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} e^{-\theta\sigma(\theta)\sqrt{n}y' - \theta\mu(\theta)n} dH_n(y') \qquad (6.2) \\
&= e^{\theta\alpha n} M^n(\theta) \int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} e^{-\theta\sigma(\theta)\sqrt{n}\,[y' + (\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)]} dH_n(y'), \qquad (6.3)
\end{aligned}$$

where $\mathbf{1}\{\cdot\}$ is the set indicator function, and (6.2) follows from $H_n(y) = F_n^{(\theta)}(\sigma(\theta)\sqrt{n}y + \mu(\theta)n)$.

Integrating by parts on (6.3) with $\lambda(dy) \triangleq -\theta\sigma(\theta)\sqrt{n}\exp\{-\theta\sigma(\theta)\sqrt{n}[y + (\mu(\theta) + \alpha)\sqrt{n}/\sigma(\theta)]\}dy$ defined over $(-\infty, -(\mu(\theta) + \alpha)\sqrt{n}/\sigma(\theta)]$, and then applying equation (6.1)

yields

$$\int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} e^{-\theta\sigma(\theta)\sqrt{n}\,[y+(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)]} dH_n(y) \tag{6.4}$$

$$\begin{aligned}
&= \int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} \left[H_n\left(-\frac{(\mu(\theta)+\alpha)\sqrt{n}}{\sigma(\theta)} \right) - H_n(y) \right] \lambda(dy) \\
&\leq \int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} \left[\Phi\left(-\frac{(\mu(\theta)+\alpha)\sqrt{n}}{\sigma(\theta)} \right) - \Phi(y) + 2C\frac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}} \right] \lambda(dy) \\
&= \int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} \left[\Phi\left(-\frac{(\mu(\theta)+\alpha)\sqrt{n}}{\sigma(\theta)} \right) - \Phi(y) \right] \lambda(dy) + 2C\frac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}} \\
&= \int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} e^{-\theta\sigma(\theta)\sqrt{n}\,[y+(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)]} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy + 2C\frac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}} \qquad (6.5) \\
&= e^{\theta^2\sigma^2(\theta)n/2} e^{-\theta(\mu(\theta)+\alpha)n} \Phi\left(\theta\sigma(\theta)\sqrt{n} - \frac{(\mu(\theta)+\alpha)\sqrt{n}}{\sigma(\theta)} \right) + 2C\frac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}} \\
&\leq \begin{cases} \dfrac{\sigma(\theta)}{\sqrt{2\pi n}[(\mu(\theta)+\alpha)-\theta\sigma^2(\theta)]} e^{-(\mu(\theta)+\alpha)^2 n/[2\sigma^2(\theta)]} + 2C\dfrac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}}, & \text{if } \alpha > \theta\sigma^2(\theta) - \mu(\theta); \\ e^{\theta^2\sigma^2(\theta)n/2} e^{-\theta(\mu(\theta)+\alpha)n} + 2C\dfrac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}}, & \text{otherwise,} \end{cases} \qquad (6.6)
\end{aligned}$$

where (6.5) holds by, again, applying integration by part, and (6.6) follows from

$$\Phi(-u) \leq \frac{1}{\sqrt{2\pi}u} e^{-u^2/2} \quad \text{and} \quad \Phi(u) \leq 1 \quad \text{for } u > 0.$$

It remains to show that

$$\int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} e^{-\theta\sigma(\theta)\sqrt{n}\,[y+(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)]} dH_n(y) \leq 1,$$

which be established by observing that

$$\begin{aligned}
e^{\theta\alpha n} M^n(\theta) \int_{-\infty}^{-(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)} e^{-\theta\sigma(\theta)\sqrt{n}\,[y+(\mu(\theta)+\alpha)\sqrt{n}/\sigma(\theta)]} dH_n(y) &= \Pr\{Y_n \leq -n\alpha\} \qquad (6.7) \\
&= \Pr\left\{ e^{\theta(Y_n+n\alpha)} \geq 1 \right\} \\
&\leq E[e^{\theta(Y_n+n\alpha)}] \\
&= e^{\theta\alpha n} M^n(\theta). \qquad (6.8)
\end{aligned}$$

□

Some remarks are made following Lemma 1 as follows. First, the upper probability bound in Lemma 1 consists of two parts, the exponentially decaying $e^{\theta\alpha n}M^n(\theta)$ and the subexponentially bounded $A_n(\theta,\alpha)$. When $\alpha > \theta\sigma^2(\theta) - \mu(\theta)$ and $\alpha \neq -\mu(\theta)$,

$$B_n(\theta,\alpha) = \frac{\sigma(\theta)}{\sqrt{2\pi n}[(\mu(\theta)+\alpha) - \theta\sigma^2(\theta)]} e^{-(\mu(\theta)+\alpha)^2 n/[2\sigma^2(\theta)]} + 2C\frac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}} \approx 2C\frac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}}$$

since the first term decays exponentially fast, and $B_n(\theta,\alpha)$ reduces to the Berry-Esseen probability bound. However, when θ is taken to satisfy $\mu(\theta) = -\alpha$,

$$B_n(\theta,\alpha) = \frac{1}{\sqrt{2\pi n}|\theta|\sigma(\theta)} + 2C\frac{\rho(\theta)}{\sigma^3(\theta)\sqrt{n}},$$

and a larger bound (than the Berry-Esseen one) is resulted. In either case, $B_n(\theta,\alpha)$ vanishes exactly at the speed of $1/\sqrt{n}$. Secondly, when $A_n(\theta,\alpha) = 1$, the upper probability bound reduces to the simple Chernoff bound $e^{\theta\alpha n}M^n(\theta)$ for which a four-line proof from (6.7) to (6.8) is sufficient [11, Eq. (5.4.9)], and is always valid for every $\theta < 0$, regardless of whether $\alpha > \theta\sigma^2(\theta) - \mu(\theta)$ or not.

The independent samples $\{X_i\}_{i=1}^n$ with which our decoding problems are concerned actually consist of two i.i.d. sequences, one of which is Gaussian distributed and the other is non-Gaussian distributed. One way to bound the desired probability of $\Pr[\sum_{i=1}^n X_i \leq 0]$ is to directly use the Berry-Esseen inequality for independent but non-identical samples (which can be done by following similar proof of Lemma 1). However, in order to manage a better bound, we will apply Lemma 1 only to those non-Gaussian i.i.d. samples, and manipulate the remaining Gaussian samples directly by way of their known probability densities in the below lemma (cf. The derivation in (6.9)).

Lemma 2. *Let $Y_n = \sum_{i=1}^n X_i$ be the sum of independent random variables $\{X_i\}_{i=1}^n$, among which $\{X_i\}_{i=1}^d$ are identically Gaussian distributed with positive mean μ and non-zero variance σ^2, and $\{X_i\}_{i=d+1}^n$ have common marginal distribution as $\min\{X_1, 0\}$. Let $\gamma \triangleq$*

$(1/2)(\mu^2/\sigma^2)$. *Then*

$$\Pr\{Y_n \leq 0\} \leq \mathcal{B}(d, n-d, \gamma),$$

where

$$\mathcal{B}(d,n-d,\gamma) = \begin{cases} \Phi(-\sqrt{2\gamma n}), & \text{if } d = n; \\ \Phi\left(-\frac{(n-d)\hat{\mu}+d\sqrt{2\gamma}}{\sqrt{d}}\right) + \tilde{A}_{n-d}(\lambda) & \\ \quad \times \left[\Phi(-\lambda)e^{-\gamma}e^{\lambda^2/2} + \Phi(\sqrt{2\gamma})\right]^{n-d} & \\ \quad \times e^{d(-\gamma+\lambda^2/2)}\Phi\left(\frac{(n-d)\hat{\mu}+\lambda d}{\sqrt{d}}\right), & \text{if } 1 > \frac{d}{n} \geq 1 - \frac{\sqrt{4\pi\gamma}e^{\gamma}}{1+\sqrt{4\pi\gamma}e^{\gamma}\Phi(\sqrt{2\gamma})}; \\ 1, & \text{otherwise,} \end{cases}$$

$$\begin{aligned} a &\triangleq -\hat{\mu} + (\sqrt{2\gamma} - \lambda)\tilde{\sigma}^2(\lambda) + \tilde{\mu}(\lambda), \\ \tilde{A}_{n-d}(\lambda) &\triangleq \min\left(\mathbf{1}\{a > 0\}\left[\frac{\tilde{\sigma}(\lambda)}{a\sqrt{2\pi(n-d)}} + 2C\frac{\tilde{\rho}(\lambda)}{\tilde{\sigma}^3(\lambda)\sqrt{n-d}}\right] + \mathbf{1}\{a \leq 0\}, 1\right), \\ \hat{\mu} &\triangleq E[X_{d+1}] = -(1/\sqrt{2\pi})e^{-\gamma} + \sqrt{2\gamma}\Phi(-\sqrt{2\gamma}), \\ \tilde{\mu}(\lambda) &= -\frac{d}{n-d}\lambda, \\ \tilde{\sigma}^2(\lambda) &\triangleq -\frac{d}{n-d} - \frac{nd}{(n-d)^2}\lambda^2 + \frac{n}{n-d}\frac{1}{1+\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})}, \\ \tilde{\rho}(\lambda) &\triangleq \frac{n}{(n-d)}\frac{\lambda}{[1+\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})]}\left\{1 - \frac{d(n+d)}{(n-d)^2}\lambda^2\right. \\ &\quad +2\left[\frac{n^2}{(n-d)^2}\lambda^2 + 2\right]e^{-d(2n-d)\lambda^2/[2(n-d)^2]} \\ &\quad -\frac{d}{n-d}\left[\frac{n+d}{n-d}\lambda^2 + 3\right]\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma}) \\ &\quad \left.-\frac{2n}{n-d}\left[\frac{n^2}{(n-d)^2}\lambda^2 + 3\right]\sqrt{2\pi}\lambda e^{\lambda^2/2}\Phi\left(-\frac{n}{n-d}\lambda\right)\right\}, \end{aligned}$$

and λ *is the unique solution (in* $[0, \sqrt{2\gamma})$*) of*

$$\lambda e^{(1/2)\lambda^2}\Phi(-\lambda) = \frac{1}{\sqrt{2\pi}}\left(1 - \frac{d}{n}\right) - \frac{d}{n}e^{\gamma}\Phi(\sqrt{2\gamma})\lambda.$$

Proof. Only the bound for $d < n$ is proved since the case of $d = n$ can be easily substantiated.

Let

$$\tilde{\mu}(\theta) = \frac{E[X_{d+1}^{(\theta)}]}{\sigma}, \quad \tilde{\sigma}(\theta) = \frac{\mathrm{Var}[X_{d+1}^{(\theta)}]}{\sigma^2}, \quad \text{and} \quad \tilde{\rho}(\theta) = \frac{E[|X_{d+1}^{(\theta)} - E[X_{d+1}^{(\theta)}]|^3]}{\sigma^3},$$

and let $\hat{\mu} = E[X_{d+1}]/\sigma$. By noting that $(\mu/\sigma) = \sqrt{2\gamma}$, and for any $\theta < 0$ satisfying that

$$a \triangleq -\hat{\mu} - \sigma\theta\tilde{\sigma}^2(\theta) + \tilde{\mu}(\theta) > 0,$$

$\Pr(Y_n \leq 0)$ can be bounded by

$$\begin{aligned}
&\Pr(Y_n \leq 0) \\
&= \Pr\{X_1 + \cdots + X_d + X_{d+1} + \cdots + X_n \leq 0\} \\
&= \int_{-\infty}^{\infty} \Pr\{X_{d+1} + \cdots + X_n \leq -x\} \frac{1}{\sqrt{2\pi d\sigma^2}} e^{-\frac{(x-d\mu)^2}{2d\sigma^2}} dx, \quad (x \to \sigma x') \\
&= \int_{-\infty}^{\infty} \Pr\{X_{d+1} + \cdots + X_n \leq -\sigma x'\} \frac{1}{\sqrt{2\pi d}} e^{-\frac{(x'-d\sqrt{2\gamma})^2}{2d}} dx', \quad (x' \to (n-d)x'') \\
&= \int_{-\infty}^{\infty} \Pr\{X_{d+1} + \cdots + X_n \leq -\sigma(n-d)x''\} \frac{1}{\sqrt{2\pi d/(n-d)^2}} e^{-\frac{(x''-d\sqrt{2\gamma}/(n-d))^2}{2d/(n-d)^2}} dx'' \\
&= \int_{-\infty}^{\sigma\theta\tilde{\sigma}^2(\theta)-\tilde{\mu}(\theta)+a} \Pr\{X_{d+1} + \cdots + X_n \leq -\sigma(n-d)x''\} \frac{1}{\sqrt{2\pi d/(n-d)^2}} e^{-\frac{(x''-d\sqrt{2\gamma}/(n-d))^2}{2d/(n-d)^2}} dx'' \\
&+ \int_{\sigma\theta\tilde{\sigma}^2(\theta)-\tilde{\mu}(\theta)+a}^{\infty} \Pr\{X_{d+1} + \cdots + X_n \leq -\sigma(n-d)x''\} \frac{1}{\sqrt{2\pi d/(n-d)^2}} e^{-\frac{(x''-d\sqrt{2\gamma}/(n-d))^2}{2d/(n-d)^2}} dx'' \\
&\leq \int_{-\infty}^{\sigma\theta\tilde{\sigma}^2(\theta)-\tilde{\mu}(\theta)+a} \frac{1}{\sqrt{2\pi d/(n-d)^2}} e^{-\frac{(x''-d\sqrt{2\gamma}/(n-d))^2}{2d/(n-d)^2}} dx'' \\
&+ \int_{\sigma\theta\tilde{\sigma}^2(\theta)-\tilde{\mu}(\theta)+a}^{\infty} \min\left(\frac{\tilde{\sigma}(\theta)}{a\sqrt{2\pi(n-d)}} e^{-(\tilde{\mu}(\theta)+x'')^2(n-d)/[2\tilde{\sigma}^2(\theta)]} + 2C\frac{\tilde{\rho}(\theta)}{\tilde{\sigma}^3(\theta)\sqrt{n-d}}, 1 \right) \\
&\quad \times e^{\theta\sigma(n-d)x''} M^{n-d}(\theta) \frac{1}{\sqrt{2\pi d/(n-d)^2}} e^{-\frac{(x''-d\sqrt{2\gamma}/(n-d))^2}{2d/(n-d)^2}} dx'', \qquad (6.9)
\end{aligned}$$

where $M(\theta) = E[e^{\theta X_{d+1}}]$, and the last inequality follows from Lemma 1. Observe that

$$e^{-(\tilde{\mu}(\theta)+x'')^2(n-d)/[2\tilde{\sigma}^2(\theta)]} \leq 1.$$

Thus,

$$\begin{aligned}
\Pr(Y_n \le 0) &\le \int_{-\infty}^{-\hat{\mu}} \frac{1}{\sqrt{2\pi d/(n-d)^2}} e^{-\frac{(x''-d\sqrt{2\gamma}/(n-d))^2}{2d/(n-d)^2}} dx'' \\
&+ \int_{-\hat{\mu}}^{\infty} \min\left(\frac{\tilde{\sigma}(\theta)}{a\sqrt{2\pi(n-d)}} + 2C\frac{\tilde{\rho}(\theta)}{\tilde{\sigma}^3(\theta)\sqrt{n-d}}, 1\right) \\
&\quad \times e^{\theta\sigma(n-d)x''} M^{n-d}(\theta) \frac{1}{\sqrt{2\pi d/(n-d)^2}} e^{-\frac{(x''-d\sqrt{2\gamma}/(n-d))^2}{2d/(n-d)^2}} dx'' \\
&= \Phi\left(-\frac{(n-d)\hat{\mu} + d\sqrt{2\gamma}}{\sqrt{d}}\right) \\
&\quad + \tilde{A}_{n-d}(\theta) M^{n-d}(\theta) e^{d(\theta\sigma\sqrt{2\gamma}+\theta^2\sigma^2/2)} \Phi\left(\frac{(n-d)\hat{\mu} + d\sqrt{2\gamma}}{\sqrt{d}} + \theta\sigma\sqrt{d}\right),
\end{aligned} \tag{6.10}$$

where for $a > 0$,

$$\tilde{A}_{n-d}(\theta) = \min\left(\frac{\tilde{\sigma}(\theta)}{a\sqrt{2\pi(n-d)}} + 2C\frac{\tilde{\rho}(\theta)}{\tilde{\sigma}^3(\theta)\sqrt{n-d}}, 1\right).$$

Now for $\theta < 0$ and $a \le 0$, we can use Chernoff bound in (6.9) instead, in which case the derivation up to (6.10) similarly follows with $\tilde{A}_{n-d}(\theta) = 1$.

We then note that

$$M^{n-d}(\theta) e^{d(\theta\sigma\sqrt{2\gamma}+\theta^2\sigma^2/2)}$$

is exactly the moment generating function of $Y_n = \sum_{i=1}^{n} X_i$; hence, if $E[Y_n] = d\mu + (n-d)\sigma\hat{\mu} > 0$, then the solution θ of $\partial E[e^{\theta Y_n}]/\partial\theta = 0$ is definitely negative.

For notational convenience, we let $\lambda = (\mu/\sigma) + \sigma\theta = \sqrt{2\gamma} + \sigma\theta$, and yield that

$$M(\theta) = \Phi(-\lambda)\, e^{-\gamma} e^{\lambda^2/2} + \Phi(\sqrt{2\gamma}) \quad \text{and} \quad e^{\theta\sigma\sqrt{2\gamma}+\theta^2\sigma^2/2} = e^{-\gamma} e^{\lambda^2/2}.$$

Accordingly, the chosen $\lambda = \sqrt{2\gamma} + \sigma\theta$ should satisfy

$$\frac{\partial\left(\left[\Phi(-\lambda)e^{-\gamma}e^{\lambda^2/2} + \Phi(\sqrt{2\gamma})\right]^{n-d} e^{d(-\gamma+\lambda^2/2)}\right)}{\partial\lambda} = 0,$$

or equivalently,

$$e^{(1/2)\lambda^2}\Phi(-\lambda) = \frac{1}{\sqrt{2\pi}\lambda}\left(1-\frac{d}{n}\right) - \frac{d}{n}e^{\gamma}\Phi(\sqrt{2\gamma}). \tag{6.11}$$

As it turns out, the solution $\lambda = \lambda(\gamma)$ of the above equation depends only on γ. Now, by replacing $e^{(1/2)\lambda^2}\Phi(-\lambda)$ with $(1-d/n)/(\sqrt{2\pi}\lambda) - (d/n)e^{\gamma}\Phi(\sqrt{2\gamma})$, we obtain

$$\tilde{\mu}(\lambda) = \left.\frac{E\left[X_{d+1}^{(\theta)}\right]}{\sigma}\right|_{\theta=(\lambda-\sqrt{2\gamma})/\sigma} = -\frac{d}{n-d}\lambda$$

$$\begin{aligned}\tilde{\sigma}^2(\lambda) &\triangleq \left.\frac{\mathrm{Var}\left[X_{d+1}^{(\theta)}\right]}{\sigma^2}\right|_{\theta=(\lambda-\sqrt{2\gamma})/\sigma} \\ &= -\frac{d}{n-d} - \frac{nd}{(n-d)^2}\lambda^2 + \frac{n}{n-d}\frac{1}{1+\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})},\end{aligned}$$

and

$$\begin{aligned}\tilde{\rho}(\lambda) &\triangleq \left.\frac{E\left[\left|X_{d+1}^{(\theta)}-\hat{\mu}\right|^3\right]}{\sigma^3}\right|_{\theta=(\lambda-\sqrt{2\gamma})/\sigma} \\ &= \frac{n}{(n-d)}\frac{\lambda}{\left[1+\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})\right]}\left\{1-\frac{d(n+d)}{(n-d)^2}\lambda^2\right. \\ &\quad +2\left[\frac{n^2}{(n-d)^2}\lambda^2+2\right]e^{-d(2n-d)\lambda^2/[2(n-d)^2]} \\ &\quad -\frac{d}{n-d}\left[\frac{n+d}{n-d}\lambda^2+3\right]\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma}) \\ &\quad \left.-\frac{2n}{n-d}\left[\frac{n^2}{(n-d)^2}\lambda^2+3\right]\sqrt{2\pi}\lambda e^{\lambda^2/2}\Phi\left(-\frac{n}{n-d}\lambda\right)\right\}\end{aligned}$$

Hence, the previously obtained upper bound for $\Pr(Y_n \leq 0)$ can be reformulated as

$$\begin{aligned}&\Phi\left(-\frac{(n-d)\hat{\mu}+d\sqrt{2\gamma}}{\sqrt{d}}\right) \\ &\quad +\tilde{A}_{n-d}(\lambda)\left[\Phi(-\lambda)e^{-\gamma}e^{\lambda^2/2}+\Phi(\sqrt{2\gamma})\right]^{n-d}e^{d(-\gamma+\lambda^2/2)}\Phi\left(\frac{(n-d)\hat{\mu}+\lambda d}{\sqrt{d}}\right),\end{aligned}$$

where

$$\tilde{A}_{n-d}(\lambda) = \min\left(\mathbf{1}\{a>0\}\left[\frac{\tilde{\sigma}(\lambda)}{a\sqrt{2\pi(n-d)}}+2C\frac{\tilde{\rho}(\lambda)}{\tilde{\sigma}^3(\lambda)\sqrt{n-d}}\right]+\mathbf{1}\{a\leq 0\},1\right).$$

Finally, a simple derivation yields

$$\begin{aligned} E[Y_n] &= dE[X_1] + (n-d)E[X_{d+1}] \\ &= \sigma\left(d\sqrt{2\gamma} + (n-d)\left[-(1/\sqrt{2\pi})e^{-\gamma} + \sqrt{2\gamma}\Phi(-\sqrt{2\gamma})\right]\right), \end{aligned}$$

and hence, the condition of $E[Y_n] > 0$ can be equivalently replaced by

$$\frac{d}{n} \geq 1 - \frac{\sqrt{4\pi\gamma}e^{\gamma}}{1 + \sqrt{4\pi\gamma}e^{\gamma}\Phi(\sqrt{2\gamma})}.$$

□

Again, if the simple Chernoff inequality is used instead in the derivation of (6.9), the bound remains of the same form in Lemma 2 except that $\tilde{A}_{n-d}(\lambda)$ is always equal to one.

Empirical evaluations of $\tilde{A}_{n-d}(\lambda)$ in Figs. 6.1 and 6.2 indicates that when the sample number $n \leq 50$, $\tilde{A}_{n-d}(\lambda)$ will be close to 1, and the subexponential analysis based on the Berry-Esseen inequality does not help improving the upper probability bound. However, for a slightly larger n such as $n = 200$, a visible reduction in the probability bound can be obtained through the introduction of the Berry-Esseen inequality.

One of the main studied subjects in this chapter is to examine whether the introduction of the subexponential analysis can help improving the complexity bound at practical code length. The observation from Figs. 6.1 and 6.2 does coincide with what we obtained in later applications. That is, some visible improvement in complexity bound can really be obtained for a little larger codeword length in the MLSDA (specifically, $N = 2(100+10)$ or $2(100+6)$).

We end this section by presenting the operational meanings of the three arguments in function $\mathcal{B}(\cdot,\cdot,\cdot)$ before their practice in subsequent sections. When in use for sequential-type decoding complexity analysis, the first integer argument is the Hamming distance between the transmitted codeword and the examined codeword up to the level of the currently visited tree (or trellis) node. The second integer argument represents a prediction of the future route,

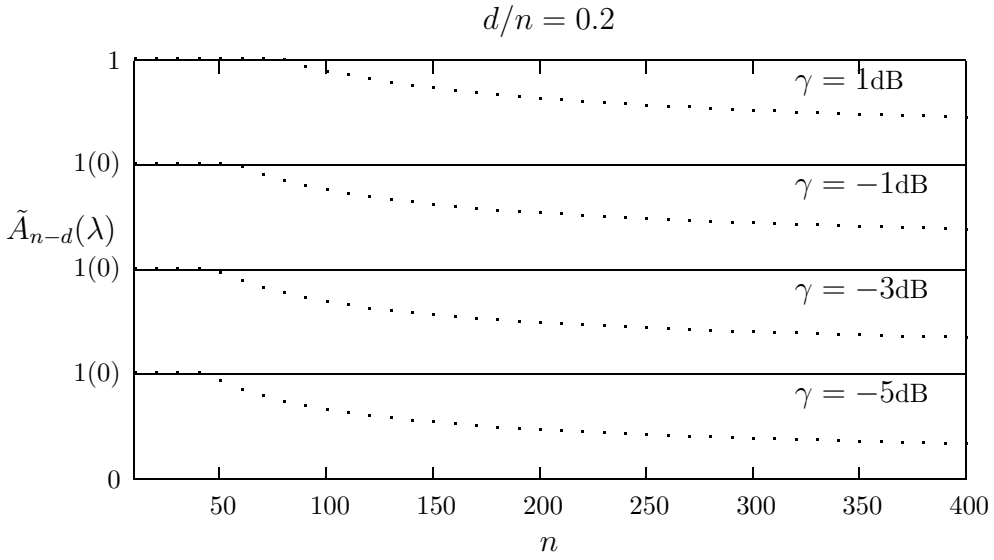

Figure 6.1: $\tilde{A}_{n-d}(\lambda)$ for fixed $d/n = 0.2$ with respect to different γ. Notation "1(0)" represents that the y-tic is either 1 (for the curve below) or 0 (for the curve above).

which is not yet occurred.[1] The third argument is exactly the signal-to-noise ratio for the decoding environment, and is reasonably assumed to be always positive.

6.2 Computation Complexity for MLSDA

Notations that will be used in the next theorem are first introduced. Denote by $s_j(\ell)$ the node that is located at level ℓ and corresponds to state index j. Let $\mathcal{S}_j(\ell)$ be the set of paths that end at node $s_j(\ell)$. Also let $\mathcal{H}_j(\ell)$ be the set of the Hamming weights of the paths in $\mathcal{S}_j(\ell)$. Denote the minimum Hamming weight in $\mathcal{H}_j(\ell)$ by $d_j^*(\ell)$. As an example, $\mathcal{S}_3(3)$ equals $\{111010001, 000111010\}$ in Fig. 2.4, which results in $\mathcal{H}_3(3) = \{5, 4\}$ and $d_3^*(3) = 4$.

[1]The metric for use of sequential-type decoding can be generally divided into two parts, where the first part is determined by the *past* branches traversed thus far, while the second part helps predicting the *future* route to speed up the code search process [15]. For example, by adding a constant term $\sum_{i=1}^{N} \log_2 \Pr(y_i)$ to the accumulant Fano metric $\sum_{i=1}^{q} (\log_2[\Pr(y_j|b_j)/\Pr(y_j)] - R)$ up to level q, it can be seen that $\sum_{i=1}^{q} (\log_2(\Pr(y_j|b_j) - R)$ weights the history, and $\sum_{i=q+1}^{N} \log_2 \Pr(y_j)$ is the expectation of branch metrics to be added for possible future routes. Based on the intuition, the first argument and the second argument respectively realize the *historical* known part and the *future* predictive part of the decoding metric.

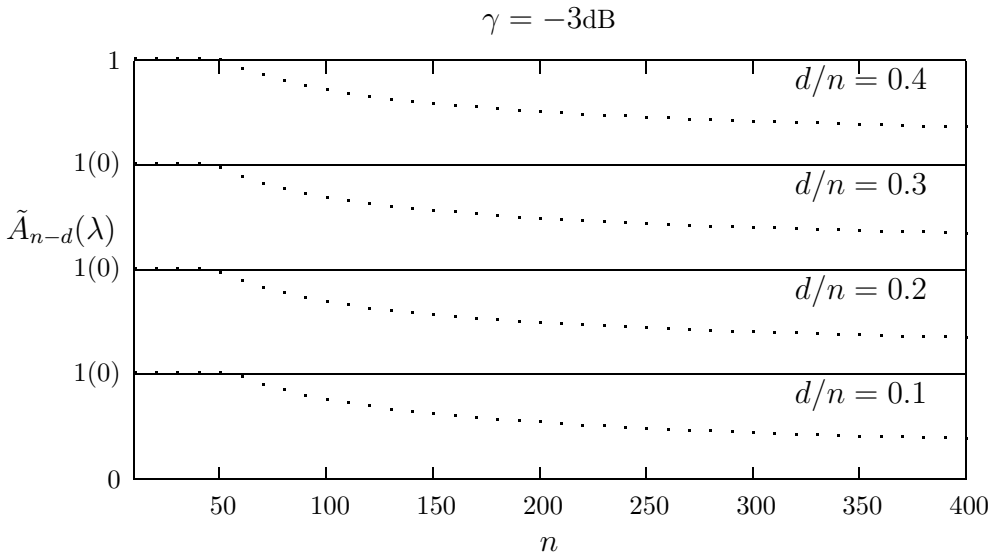

Figure 6.2: $\tilde{A}_{n-d}(\lambda)$ for fixed $\gamma = -3dB$ with respect to different d/n ratios. Notation "1(0)" represents that the y-tic is either 1 (for the curve below) or 0 (for the curve above).

Theorem 1 (Complexity of the MLSDA). *Consider an (n, k, m) binary convolutional code transmitted via an AWGN channel. The average number of branch metric computations evaluated by the MLSDA, denoted by $L_{\mathrm{MLSDA}}(\gamma_b)$, is upper-bounded by*

$$L_{\mathrm{MLSDA}}(\gamma_b) \leq 2^k \sum_{\ell=0}^{L-1} \sum_{j=0}^{2^m-1} \mathcal{B}\left(d_j^*(\ell), N - \ell n, \frac{kL}{N}\gamma_b\right), \tag{6.12}$$

where if $\mathcal{H}_j(\ell)$ is empty, implying the non-existence of state j at level ℓ, then $\mathcal{B}(d_j^*(\ell), N - \ell n, kL\gamma_b/N) = 0$.

Proof. Assume without loss of generality that the all-zero codeword $\mathbf{0}$ is transmitted. First, observe that for any two paths that end at a common node, only one of them will survive in the Open Stack. In other words, one of the two paths will be discarded either due to a larger path metric or because its end node has the same state and level as an entry in the Closed Stack. In the latter case, the surviving path has clearly reached the common end node earlier, and has already been extended by the MLSDA at some previous time (so that

the state and level of its end node has already been stored in the Closed Stack). Accordingly, the branch metric computations that follow these two paths will only be performed once. It therefore suffices to derive the computational complexity of the MLSDA based on the nodes that have been extended rather than the paths that have been extended.

Let $\boldsymbol{x}^*$ label the minimum-metric code path for a given log-likelihood ratio $\boldsymbol{\phi}$. Then we claim that if a node $s_j(\ell)$ is extended by the MLSDA, given that $\boldsymbol{x}_{(\ell n-1)}$ is the only surviving path (in the Open Stack) that ends at this node at the time this node is extended, then

$$\mu(\boldsymbol{x}_{(\ell n-1)}) \leq \mu(\boldsymbol{x}^*). \tag{6.13}$$

The validity of the above claim can be simply proved by contradiction. Suppose

$$\mu(\boldsymbol{x}_{(\ell n-1)}) > \mu(\boldsymbol{x}^*).$$

Then the non-negativity of the individual metric $(y_j \oplus x_j)|\phi_j|$, which implies $\mu(\boldsymbol{x}^*) \geq \mu(\boldsymbol{x}^*_{(b)})$ for every $0 \leq b \leq N-1$, immediately gives $\mu(\boldsymbol{x}_{(\ell n-1)}) > \mu(\boldsymbol{x}^*_{(b)})$ for every $0 \leq b \leq N-1$. Therefore, $\boldsymbol{x}_{(\ell n-1)}$ cannot be on top of the Open Stack (because some $\boldsymbol{x}^*_{(b)}$ always exists in the Open Stack), and hence violates the assumption that $s_j(\ell)$ is extended by the MLSDA.

For notational convenience, denote by $\mathcal{A}(s_j(\ell), \boldsymbol{x}_{(\ell n-1)})$ the *event* that "$\boldsymbol{x}_{(\ell n-1)}$ *is the only path in the intersection of* $\mathcal{S}_j(\ell)$ *and the Open Stack at the time node* $s_j(\ell)$ *is extended.*" Notably,

$$\{\mathcal{A}(s_j(\ell), \boldsymbol{x}_{(\ell n-1)})\}_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)}$$

are disjoint, and

$$\sum_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\mathcal{A}\left(s_j(\ell), \boldsymbol{x}_{(\ell n-1)}\right)\right\} = 1.$$

Then according to the above claim,

$$
\begin{aligned}
&\Pr\{\text{node } s_j(\ell) \text{ is extended by the MLSDA}\} \\
&= \sum_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\mathcal{A}\left(s_j(\ell), \boldsymbol{x}_{(\ell n-1)}\right)\right\} \Pr\left\{ \begin{array}{l} \text{node } s_j(\ell) \text{ is extended} \\ \text{by the MLSDA} \end{array} \,\middle|\, \mathcal{A}\left(s_j(\ell), \boldsymbol{x}_{(\ell n-1)}\right) \right\} \\
&\leq \max_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{ \begin{array}{l} \text{node } s_j(\ell) \text{ is extended} \\ \text{by the MLSDA} \end{array} \,\middle|\, \mathcal{A}\left(s_j(\ell), \boldsymbol{x}_{(\ell n-1)}\right) \right\} \\
&\leq \max_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\mu(\boldsymbol{x}_{(\ell n-1)}) \leq \mu(\boldsymbol{x}^*)\right\} \\
&\leq \max_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\mu(\boldsymbol{x}_{(\ell n-1)}) \leq \mu(\mathbf{0})\right\} \\
&= \max_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\sum_{j=0}^{\ell n-1} (y_j \oplus x_j)|\phi_j| \leq \sum_{j=0}^{N-1} (y_j \oplus 0)|\phi_j|\right\}, \qquad (6.14)
\end{aligned}
$$

where the replacement of $\boldsymbol{x}^*$ by the all-zero codeword $\mathbf{0}$ follows from $\mu(\boldsymbol{x}^*) \leq \mu(\mathbf{0})$. We then observe that for the AWGN channel, $\phi_j = 4\sqrt{E} r_j / N_0$; hence, y_j can be determined by

$$
y_j = \begin{cases} 1, & \text{if } r_j < 0; \\ 0, & \text{otherwise.} \end{cases}
$$

This observation, together with the fact that $2(y_j \oplus x_j)|r_j| = r_j[(-1)^{y_j} - (-1)^{x_j}]$, gives

$$
\begin{aligned}
&\Pr\{\text{node } s_j(\ell) \text{ is extended by the MLSDA}\} \\
&\leq \max_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\sum_{j=0}^{\ell n-1} (y_j \oplus x_j)|r_j| \leq \sum_{j=0}^{N-1} (y_j \oplus 0)|r_j|\right\}, \\
&= \max_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\sum_{j=0}^{\ell n-1} r_j \left[(-1)^{y_j} - (-1)^{x_j}\right] \leq \sum_{j=0}^{N-1} r_j \left[(-1)^{y_j} - (-1)^{0}\right]\right\} \\
&= \max_{\boldsymbol{x}_{(\ell n-1)} \in \mathcal{S}_j(\ell)} \Pr\left\{\sum_{j \in \mathcal{J}(\boldsymbol{x}_{(\ell n-1)})} r_j + \sum_{j=\ell n}^{N-1} \min(r_j, 0) \leq 0\right\},
\end{aligned}
$$

where $\mathcal{J}(\boldsymbol{x}_{(\ell n-1)})$ is the set of index j, where $0 \leq j \leq \ell n - 1$, for which $x_j = 1$. As r_j is Gaussian distributed with mean $\sqrt{E}$ and variance $N_0/2$ due to the transmission of the

all-zero codeword, Proposition 1 (in the Appendix) and Lemma 2 can be applied to obtain

$$\begin{aligned}
&\Pr\{\text{node } s_j(\ell) \text{ is extended by the MLSDA}\} \\
&\leq \max_{d\in\mathcal{H}_j(\ell)} \Pr\left\{r_1+\cdots+r_d+\sum_{j=\ell n}^{N-1}\min(r_j,0)\leq 0\right\} \\
&= \Pr\left\{r_1+\cdots+r_{d_j^*(\ell)}+\sum_{j=\ell n}^{N-1}\min(r_j,0)\leq 0\right\} \\
&\leq \mathcal{B}\left(d_j^*(\ell), N-\ell n, \frac{kL}{N}\gamma_b\right).
\end{aligned}$$

Consequently,

$$L_{\text{MLSDA}}(\gamma_b) \leq 2^k \sum_{\ell=0}^{L-1}\sum_{j=0}^{2^m-1} \mathcal{B}\left(d_j^*(\ell), N-\ell n, \frac{kL}{N}\gamma_b\right),$$

where the multiplication of 2^k is due to the fact that whenever a node is extended, 2^k branch metric computations will follow. □

6.3 Computation Complexity for MLSDA with Early Elimination

Next we analyze the decoding complexity for the MLSDA with early elimination. Note that in this section, we assume again that the all-zero sequence is transmitted.

By referring to Fig. 6.3, let $\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}$, $\boldsymbol{x}^{*}_{((\ell+\Delta)n-1)}$, and $\mathbf{0}_{((\ell+\Delta)n-1)}$ label the first extended path, the minimum metric path, and the all-zero path, respectively, among all paths from $s_0(0)$ to nodes at level $(\ell+\Delta)$ by the MLSDA with early elimination. We claim that a node $s_j(\ell)$ is extended by the MLSDA with early elimination, given that $\boldsymbol{x}_{(\ell n-1)}$ is the only surviving path (in the Open Stack) that ends at this node at the time this node is extended, only if

$$\mu(\boldsymbol{x}_{(\ell n-1)}) \leq \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}). \tag{6.15}$$

The validity of the above claim can be simply proved by contradiction. Suppose

$$\mu(\boldsymbol{x}_{(\ell n-1)}) > \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}).$$

Then $\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}$ will be extended before the expansion of $\boldsymbol{x}_{(\ell n-1)}$, and hence $\boldsymbol{x}_{(\ell n-1)}$ will be early eliminated, which violates the assumption that $s_j(\ell)$ is extended by the MLSDA with early elimination.

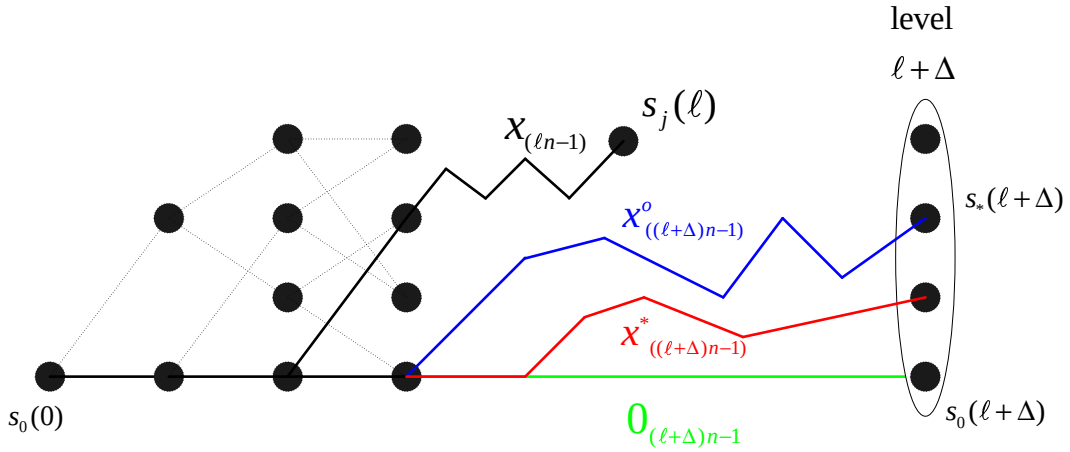

Figure 6.3: Exemplified trellis diagram for the MLSDA with early elimination.

Unlike the MLSDA without early elimination, where the first extended path must be the minimum metric path among all paths ending at the same level, the path $\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}$ may not be the minimum metric path $\boldsymbol{x}^{*}_{((\ell+\Delta)n-1)}$ for the MLSDA with early elimination. An example is illustrated in Fig. 6.4. In this example, the minimum-metric (resp. all-zero) path is path OB (resp. OD) with metric $4+2=6$ (resp. $5+6=11$). The first extended path $\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}$ however is path OC whose path metric $3+10=13$ is larger than the metrics of paths OB and OD. This is because path OC' that has a small accumulated metric 3 at early steps eliminates paths OB' and OD'.

We denote the probability that a node $s_j(\ell)$ is extended by the MLSDA with early

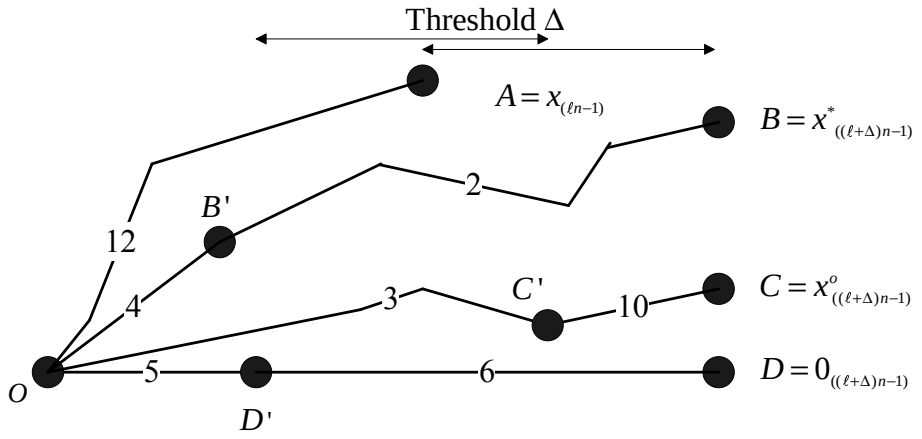

Figure 6.4: Example that the first extended path has a larger metric, when it is compared with the all-zero path for the MLSDA with early elimination.

elimination by $\Pr\{s_j(\ell) \text{ Ext}\}$, and separate it into two cases:

$$\begin{aligned}\Pr\{s_j(\ell) \text{ Ext}\} &= \Pr\left\{s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) > \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right\} \\ &\quad + \Pr\left\{s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) \le \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right\}. \quad (6.16)\end{aligned}$$

Notably, the event $\left[s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) > \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right]$ happens only when path $\mathbf{0}_{((\ell+\Delta)n-1)}$ has smaller metric than that of path $\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}$ but path $\mathbf{0}_{((\ell+\Delta)n-1)}$ is early-eliminated before it reaches level $(\ell+\Delta)$. This event clearly implies the occurrence of block error. Therefore,

$$\Pr\left\{s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) > \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right\} \le P_{\text{EE}}(\gamma_b, \Delta), \quad (6.17)$$

where $P_{\text{EE}}(\gamma_b, \Delta)$ denotes the block error rate of the MLSDA with early elimination window Δ under SNR γ_b, which can be further upper-bounded by the results in Chapter 5.[2] Notably, since Δ is presumably chosen to be large enough to achieve near ML performance, the event $\left[s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) > \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right]$ actually has a much smaller probability than event $\left[s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) \le \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right]$. We therefore call the event

[2]Taking the block error rate upper bound in Chapter 5 as the upper bound on the first term in (6.16) only provides satisfactory result for medium to high SNRs. However, from simulations (not shown in this dissertation), we found the SNR range that the effect of the first term to the overall decoding complexity is negligible can actually be extended to low SNRs.

corresponding to the second term in (6.16) the *normal event*, and simply bound the first term in (6.16) by the block error rate of the MLSDA with early elimination. The next theorem gives the decoding complexity upper bound due to the occurrence of the normal event.

Theorem 2. (Complexity of the MLSDA with early elimination due to the normal event) *Consider an (n, k, m) binary convolutional code transmitted via an AWGN channel. The average number of branch metric computations evaluated by the MLSDA with early elimination given the occurrence of the normal event, denoted by $L_{\mathrm{EE,normal}}(\gamma_b)$, is upper-bounded by*

$$L_{\mathrm{EE,normal}}(\gamma_b) \leq 2^k \sum_{\ell=0}^{L-1} \sum_{j=0}^{2^m-1} \mathcal{B}\left(d_j^*(\ell), \Delta n, \frac{kL}{N}\gamma_b\right), \tag{6.18}$$

where if $\mathcal{H}_j(\ell)$ is empty, implying the non-existence of state j at level ℓ, then

$$\mathcal{B}(d_j^*(\ell), \Delta n, kL\gamma_b/N) = 0.$$

Proof. Given that the normal event occurs,

$$
\begin{aligned}
&\Pr\left\{s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) \le \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right\} \\
&= \sum_{\boldsymbol{x}_{(\ell n-1)}\in\mathcal{S}_j(\ell)} \Pr\left\{\mathcal{A}\left(s_j(\ell), \boldsymbol{x}_{(\ell n-1)}\right)\right\} \\
&\quad \times \Pr\left\{ s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) \le \mu(\mathbf{0}_{((\ell+\Delta)n-1)}) \middle| \mathcal{A}\left(s_j(\ell), \boldsymbol{x}_{(\ell n-1)}\right)\right\} \\
&\le \max_{\boldsymbol{x}_{(\ell n-1)}\in\mathcal{S}_j(\ell)} \Pr\left\{ s_j(\ell) \text{ Ext, and } \mu(\boldsymbol{x}^{\circ}_{((\ell+\Delta)n-1)}) \le \mu(\mathbf{0}_{((\ell+\Delta)n-1)}) \middle| \mathcal{A}\left(s_j(\ell), \boldsymbol{x}_{(\ell n-1)}\right)\right\} \\
&\le \max_{\boldsymbol{x}_{(\ell n-1)}\in\mathcal{S}_j(\ell)} \Pr\left\{\mu(\boldsymbol{x}_{(\ell n-1)}) \le \mu(\mathbf{0}_{((\ell+\Delta)n-1)})\right\} \\
&= \max_{\boldsymbol{x}_{(\ell n-1)}\in\mathcal{S}_j(\ell)} \Pr\left\{\sum_{j=0}^{\ell n-1}(y_j \oplus x_j)|\phi_j| \le \sum_{j=0}^{(\ell+\Delta)n-1}(y_j \oplus 0)|\phi_j|\right\}. \\
&= \max_{\boldsymbol{x}_{(\ell n-1)}\in\mathcal{S}_j(\ell)} \Pr\left\{\sum_{j=0}^{\ell n-1}(y_j \oplus x_j)|r_j| \le \sum_{j=0}^{(\ell+\Delta)n-1}(y_j \oplus 0)|r_j|\right\}, \\
&= \max_{\boldsymbol{x}_{(\ell n-1)}\in\mathcal{S}_j(\ell)} \Pr\left\{\sum_{j=0}^{\ell n-1} r_j\left[(-1)^{y_j} - (-1)^{x_j}\right] \le \sum_{j=0}^{(\ell+\Delta)n-1} r_j\left[(-1)^{y_j} - (-1)^{0}\right]\right\} \\
&= \max_{\boldsymbol{x}_{(\ell n-1)}\in\mathcal{S}_j(\ell)} \Pr\left\{\sum_{j\in\mathcal{J}(\boldsymbol{x}_{(\ell n-1)})} r_j + \sum_{j=\ell n}^{(\ell+\Delta)n-1}\min(r_j, 0) \le 0\right\} \\
&= \max_{d\in\mathcal{H}_j(\ell)} \Pr\left\{r_1 + \cdots + r_d + \sum_{j=\ell n}^{(\ell+\Delta)n-1}\min(r_j, 0) \le 0\right\} \\
&= \Pr\left\{r_1 + \cdots + r_{d_j^*(\ell)} + \sum_{j=\ell n}^{(\ell+\Delta)n-1}\min(r_j, 0) \le 0\right\} \\
&\le \mathcal{B}\left(d_j^*(\ell), \Delta n, \frac{kL}{N}\gamma_b\right).
\end{aligned}
$$

Consequently,

$$
L_{\mathrm{EE,normal}}(\gamma_b) \le 2^k \sum_{\ell=0}^{L-1}\sum_{j=0}^{2^m-1} \mathcal{B}\left(d_j^*(\ell), \Delta n, \frac{kL}{N}\gamma_b\right).
$$

□

We remark that the upper bound for the complexity of the MLSDA with early elimination

due to the occurrence of the normal event is exactly the same as that of the MLSDA without early elimination except the second parameter in function $\mathcal{B}(\cdot,\cdot,\cdot)$. Since the new parameter Δn is not related to the message length, it is anticipated (and later confirmed by numericals) that the resultant upper bound is less relevant to the message length.

Finally, we bound the overall complexity of the MLSDA with early elimination, $L_{\mathrm{EE}}(\gamma_b)$, by combining (6.17) and (6.18), and obtain:

$$\begin{aligned} L_{\mathrm{EE}}(\gamma_b) &\leq 2^k \sum_{\ell=0}^{L-1} \sum_{j=0}^{2^m-1} \left[P_{\mathrm{EE}}(\gamma_b, \Delta) + \mathcal{B}\left(d_j^*(\ell), \Delta n, \frac{kL}{N}\gamma_b \right) \right], \\ &= L \cdot 2^{k+m} \cdot P_{\mathrm{EE}}(\gamma_b, \Delta) + 2^k \sum_{\ell=0}^{L-1} \sum_{j=0}^{2^m-1} \mathcal{B}\left(d_j^*(\ell), \Delta n, \frac{kL}{N}\gamma_b \right). \end{aligned} \tag{6.19}$$

6.4 Numerical and Simulation Results

We examine the complexity bounds in terms of (2,1,10) convolutional code with message length $L = 100$ in this section.

We first note from Fig. 6.5 that the complexity upper bound in (6.19) is dominated by the second term for medium to high SNRs. This figure also shows that the complexity reduction of the early elimination modification is significant especially when SNR $\leq$ 5 dB. Specifically, Fig. 6.5 indicates that for (2,1,10) convolutional code, the average decoding complexities of the MLSDA and the MLSDA with early elimination window $\Delta = 30$ are 79.78 and 14.8 at SNR = 3.5 dB, and therefore, more than 80% of the decoding efforts is saved without visible degradation in performance.

A potential problem of the MLSDA (without early elimination) is that its decoding complexity grows as the message length increases. This complexity dependence of the MLSDA in message length can be observed in Fig. 6.6 in both the simulations and analytical bounds in Theorem 1. To be specific, Fig. 6.6 shows that the average decoding complexity per infor-

mation bit of the MLSDA is 79.78 when $L = 100$ but increases dramatically to 446.15 when $L = 200$.

By adding the early elimination modification, it is shown in Fig. 6.6 that the average decoding complexity of the MLSDA with early elimination becomes almost independent of message length L. For example, the average decoding complexities of the MLSDA with early elimination window $\Delta = 30$ are 14.07 and 12.09 respectively for $L = 100$ and $L = 200$.

Fig. 6.7 depicts the simulated average decoding complexities for codes with different memory order m. Five different rate one-half convolutional codes with memory order 2, 4, 6, 8 and 10 are respectively examined. The early elimination window Δ are thus chosen to be 10, 15, 20, 25 and 30 such that no performance degradations can be observed. The message length is $L = 100$. It can be observed from this figure that the decoding complexities of both the MLSDA with and without early elimination grow as the memory order increases at SNR = 3 dB. However, when we further increase the SNR to 5 dB, the decoding complexity grows at a very slow speed especially for the MLSDA with early elimination. This enhances the potentiality of the MLSDA with early elimination in applications over codes with large constraint length (at medium to high SNRs).

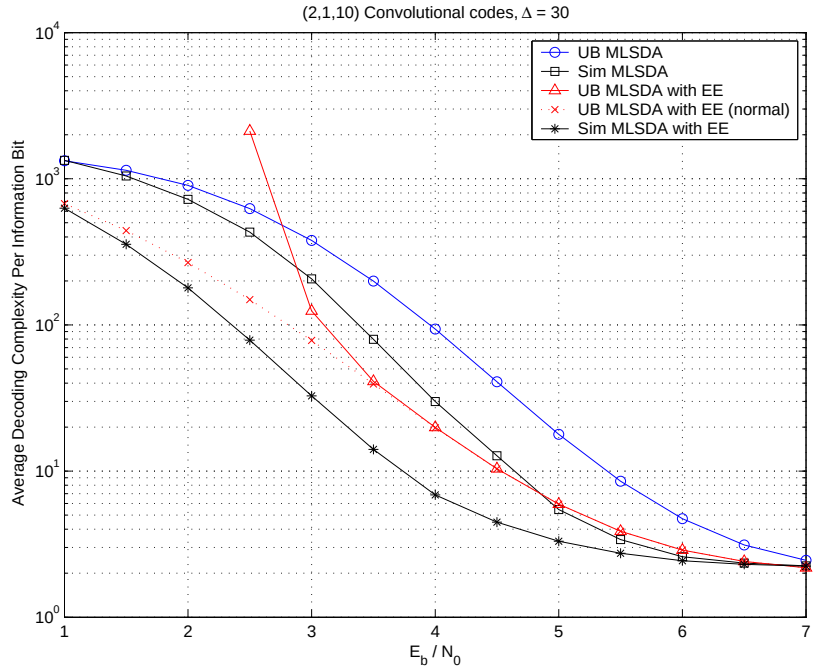

Figure 6.5: Decoding complexity upper bounds and simulations for (2,1,10) convolutional codes. The message length $L = 100$.

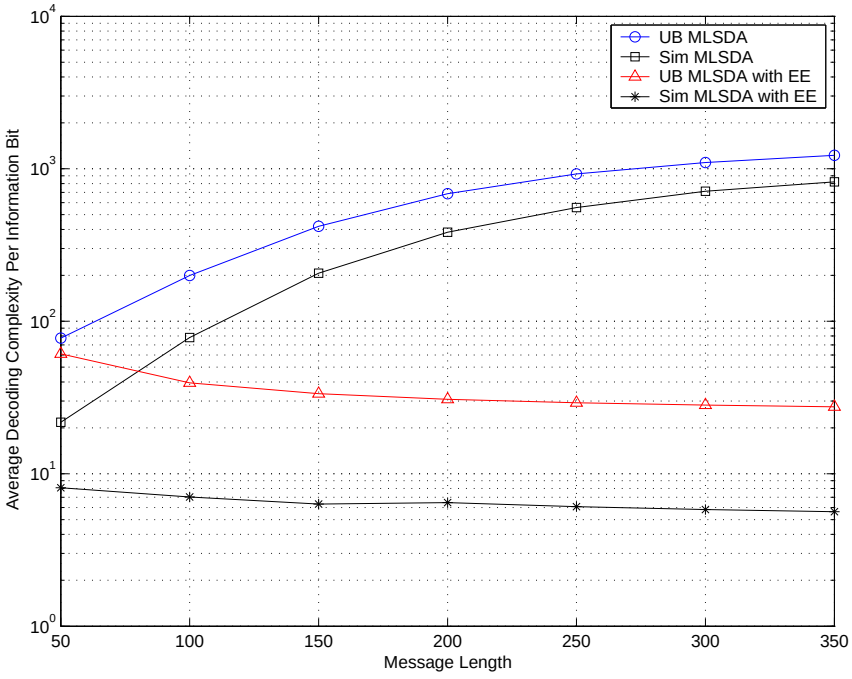

Figure 6.6: Upper bounds and simulation results of the average decoding complexity per information bit versus message length L for (2,1,10) convolutional codes at SNR = 3.5 dB.

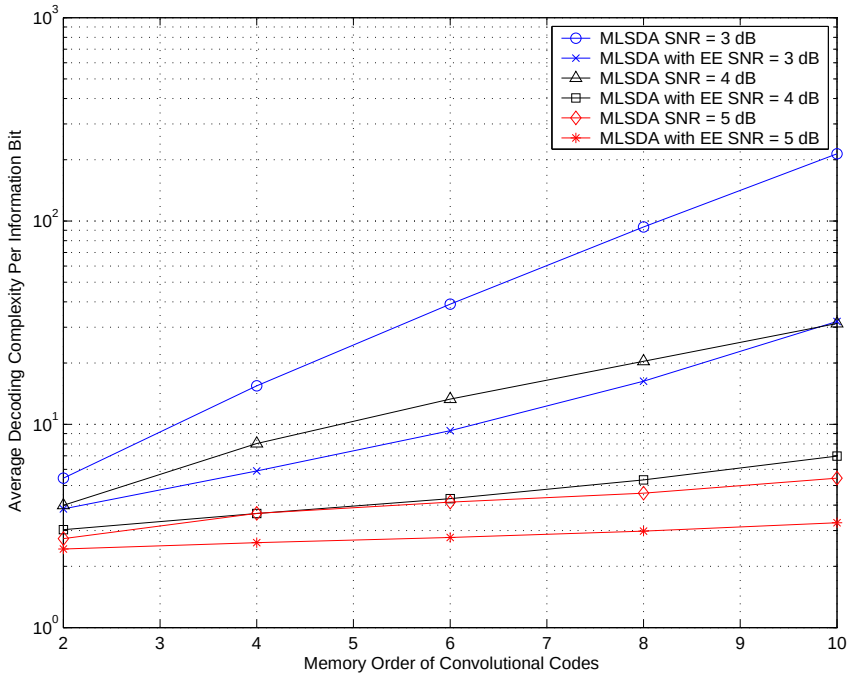

Figure 6.7: Simulation results of the average decoding complexity per information bit versus the memory order m. The message length $L = 100$. The chosen $\Delta = 10, 15, 20, 25, 30$ for $m = 2, 4, 6, 8, 10$.

Chapter 7

Concluding Remarks and Future Work

In this work, we propose to improve the computational complexity and memory requirement of the maximum-likelihood sequential-search decoding algorithm by early elimination. By setting a window and directly eliminating the partial path outside the window, the decoding complexity should be reduced with negligible performance degradation if the window size is large enough. The analysis of the sufficient early elimination window for negligible performance degradation, as well as the subsequent simulations, confirms our anticipated improvement. We also analyze the average decoding complexity for both the MLSDA with and without early elimination. The analytical and simulation results confirm the complexity reduction by introducing early elimination.

Since the MLSDA, after the introduction of early elimination modification, is justified to suit applications that dictate a near-ML software decoder with limited support in computational power and memory, a future work of practical interest will be to apply the MLSDA with early elimination to the "super-code" for joint multi-path channel equalization and convolution decoding [19].

Bibliography

[1] J. B. Anderson and S. Mohan, "Sequential coding algorithms: a survey and cost analysis," *IEEE Trans. Commun.*, vol. COM-32, no. 2, pp. 169–176, February 1984.

[2] M. Bossert, *Channel Coding for Telecommunications*. New York: John Wiley and Sons, 1999.

[3] S. Carlsson, "The DEAP–a double-ended heap to implement double-ended priorify queues," *Information Processing Letters*, vol. 26, pp. 33–36, September 1987.

[4] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York, NY: John Wiley and Sons, 1991.

[5] P. Elias, "Coding for Noisy Channels," *I.R.E. Convention Record Part* 4, pp. 37–47, September 1987.

[6] R. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inform. Theory*, vol. IT-9, no. 2, pp. 64–73, April 1963.

[7] W. Feller, *An Introduction to Probability Theory and its Applications*. New York, NY: John Wiley and Sons, 1970.

[8] G. D. Forney, Jr., "Review of random tree codes," Appendix A. Study of Coding Systems Design for Advanced Solar Missions. NASA Contract NAS2-3637. Codex Corporation. December 1967.

[9] G. D. Forney, Jr., "Convolutional codes II: Maximum likelihood decoding," *Inform. Control*, 25:222–66, July 1974.

[10] R. G. Gallager, "A simple derivation of the coding theorem and some applications," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 3–18, Jan. 1965.

[11] R. G. Gallager, *Infirmatian Theory and Reliable Communication*. New York: Wiley, 1968.

[12] J. M. Geist, "Am empirical comparison of two sequential decoding algorithms," *IEEE Trans. Commun. Technol.*, vol. COM-19, no. 4, pp. 415–419, August 1971.

[13] D. Haccoun and M. J. Ferguson, "Generalized stack algorithms for decoding convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-21, no. 6, pp. 638–651, November 1975.

[14] Y. S. Han, "A new treatment of priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. 44, no. 7, pp. 3091–3096, November 1998.

[15] Y. S. Han and P.-N. Chen, "Sequential decoding of convolutional codes," *The Wiley Encyclopedia of Telecommunications*, edited J. Proakis, John Wiley and Sons, Inc., 2002.

[16] Y. S. Han, P.-N. Chen, and M. P. C. Fossorier, "A generalization of the fano metric and its effect on sequential decoding using a stack," in *IEEE Int. Symp. on Information Theory*, Lausanne, Switzerland, 2002.

[17] Y. S. Han, P.-N. Chen, and H.-B. Wu, "A maximum-likelihood soft-decision sequential decoding algorithm for binary convolutional codes," *IEEE Trans. Commun.*, vol. 50, no. 2, pp. 173–178, February 2002.

[18] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, "Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. 39, no. 5, pp. 1514–1523, September 1993.

[19] C. Heegard, S. Coffey, S. Gummadi, E. J. Rossin, M. B. Shoemake and M. Wilhoyte, "Combined equalization and decoding for IEEE 802.11b devices," *IEEE Journal on Selected Areas in Commun.*, vol. 21, no. 2, pp. 125–138, February 2003.

[20] F. Hemmati and D. J. Costello, Jr., "Truncation error probability in Viterbi decoding," *IEEE Trans. Comm.*, vol. 25, no. 5, pp. 530–532, May 1977.

[21] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. and Dev., 13*, pp. 675–685, November 1969.

[22] R. Johannesson and K. Sh. Zigangirov, *Fundamentals of Convolutional Coding*, Wiley-IEEE Press, 1999.

[23] D. E. Knuth, *The Art of Computer Programming. Volume III: Sorting and Searching*, Reading, MA: Addison-Wesley, 1973.

[24] P. Lavoie, D. Haccoun, and Y. Savaria, "A systolic architecture for fast stack sequential decoders," *IEEE Trans. Commun.*, vol. 42, no. 5, pp. 324–335, May 1994.

[25] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, 2nd ed. Upper Saddle River, NJ: Pearson Prentice-Hall, 2004.

[26] J. L. Massey, *Threshold Decoding*, MIT Press, Cambridge, Mass., 1963.

[27] J. L. Massey, "Variable-length codes and the fano metric," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 1, pp. 196–198, January 1972.

[28] R. J. McEliece and I. M. Onyszchuk, "Truncation effects in Viterbi decoding," in *Proc. MILCOM '89*, October 1989, pp. 29.3.1–29.3.5.

[29] S. Mohan and J. B. Anderson, "Computationally optimal metric-first code tree search algorithms," *IEEE Trans. Commun.*, vol. COM-32, no. 6, pp. 710–717, June 1984.

[30] I. M. Onyszchuk, "Truncation length for Viterbi decoding," *IEEE Trans. Comm.*, vol. 39, no. 7, pp. 1023–1026, July 1991.

[31] B. Reiffen, "A note on very noisy channels," *Inform. Control*, vol. 6, pp. 126–130, 1963.

[32] Senatov and V. Vladimir, *Normal Approximation: New Results, Methods, and Problems.* Utrecht: The Netherlands, 1998.

[33] S. L. Shieh, Y. S. Han, and P.-N. Chen, "Reduction of computational complexity and sufficient stack size of the MLSDA by early elimination," in *IEEE Int. Symp. on Information Theory*, Nice, France, 2007.

[34] I. S. Shiganov, "Refinement of the upper bound of the constant in the central limit theorem," *Journal of Soviet Mathematics* 2545–2550, 1986.

[35] Texas Instruments "Using TMS320C6416 Coprocessors: Viterbi Coprocessor (VCP)," *SPRA750D Texas Instruments Application Report,* Sep. 2003. (Download from: www.ti.com.)

[36] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, no. 2, pp. 260–269, April 1967.

[37] A. J. Viterbi and J. K. Omura, *Priciples of Digital Communications and Coding*, New York: McGraw-Hill, 1979.

[38] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1995.

[39] J. M. Wozencraft and B. Reiffen, *Sequential Decoding*, MIT Press, Cambridge, 1961.

[40] K. Sh. Zigangirov, "Some sequential decoding procedures," *Probl. Peredachi Inf., 2*, pp. 13–25, 1966.

[41] K. Sh. Zigangirov, "On the error probability of the sequential decoding in the BSC," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 1, pp. 199–202, January 1972.

[42] K. Sh. Zigangirov, "New upper bounds for decoding error probability for convolutional codes," *Probl. Peredachi Inform.*, 21:20–31, 1985.

Appendix A

Proposition 1. *For a fixed non-negative integer k, the probability mass of*

$$\Pr\{r_1 + \cdots + r_d + \min(w_1, 0) + \ldots + \min(w_k, 0) \leq 0\}$$

is a decreasing function for non-negative integer d, provided that r_1, r_2, ..., r_d, w_1, w_2, ..., w_k are i.i.d. with a Gaussian marginal distribution of positive mean μ and variance σ^2.

Proof. Assume without loss of generality that $\sigma^2 = 1$. Also, assume $k \geq 1$ since the proposition is trivially valid for $k = 0$.

Let $\Omega_d \triangleq r_1 + \cdots + r_d$. Denote the probability density function of w_1 by $f(\cdot)$. Then putting $\nu \triangleq \Pr\{w_j = 0\}$ yields

$$\begin{aligned}
& \Pr\{\Omega_d + w_1 + w_2 + \cdots + w_k \leq 0\} \\
= & \sum_{j=0}^{k} \Pr\{\text{exactly } (k-j) \text{ zeros in } (w_1, w_2, \ldots, w_k)\} \\
& \qquad \Pr\{\Omega_d + w_1 + w_2 + \cdots + w_k \leq 0 | \text{ exactly } (k-j) \text{ zeros in } (w_1, w_2, \ldots, w_k)\} \\
= & \binom{k}{0}\nu^k \Pr\{\Omega_d \leq 0\} + \binom{k}{1}\nu^{k-1}(1-\nu)\int_{-\infty}^{0} f(x)\Pr\{\Omega_d \leq -x\}dx \\
& + \binom{k}{2}\nu^{k-2}(1-\nu)^2 \int_{-\infty}^{0}\int_{-\infty}^{0} f(x_1)f(x_2)\Pr\{\Omega_d \leq -(x_1+x_2)\}dx_1dx_2 \\
& + \cdots \\
& + \binom{k}{k}(1-\nu)^k \int_{-\infty}^{0}\cdots\int_{-\infty}^{0} f(x_1)\cdots f(x_k)\Pr\{\Omega_d \leq -(x_1+\cdots+x_k)\}dx_1\cdots dx_k.
\end{aligned}$$

Accordingly, if each of the above $(k+1)$ terms is non-increasing in d, so is their sum. Let

$$\begin{aligned} q_j(d) &\triangleq \int_{-\infty}^{0} \cdots \int_{-\infty}^{0} f(x_1) \cdots f(x_j) \Pr\{\Omega_d \le -(x_1 + \cdots + x_j)\} dx_1 \cdots dx_j \\ &= \int_{-\infty}^{0} \cdots \int_{-\infty}^{0} f(x_1) \cdots f(x_j) \Phi\left(-\frac{x_1 + \cdots + x_j}{\sqrt{d}} - \sqrt{d}\mu \right) dx_1 \cdots dx_j. \end{aligned}$$

Then

$$\begin{aligned} \frac{\partial q_j(d)}{\partial\left(\sqrt{d}\right)} &= \int_{-\infty}^{0} \cdots \int_{-\infty}^{0} f(x_1) \cdots f(x_j) \\ &\quad \times \left(\frac{x_1 + \cdots + x_j}{d} - \mu \right) \frac{1}{\sqrt{2\pi}} e^{-(x_1 + \cdots + x_j + d\cdot\mu)^2/(2d)} dx_1 \cdots dx_j \\ &\le -\frac{\mu}{\sqrt{2\pi}} \int_{-\infty}^{0} \cdots \int_{-\infty}^{0} f(x_1) \cdots f(x_j) e^{-(x_1 + \cdots + x_j + d\cdot\mu)^2/(2d)} dx_1 \cdots dx_j \quad (7.1) \\ &< 0, \end{aligned}$$

where (7.1) follows from $x_i \le 0$ (according to the range of integration) for $1 \le i \le j$. Consequently, $q_j(d)$ is decreasing in d for d positive and every $1 \le j \le k$. The proof is completed by noting that the first term, $\Pr\{\Omega_d \le 0\} = \Phi(-\sqrt{d}\mu)$, is also decreasing in d. □