國立交通大學

電信工程學系碩士班

碩 士 論 文

去尾迴旋碼結合解碼前環狀位移之解碼演算法

Decoding the Tail-Biting Convolutional Codes with Pre-Decoding Circular Shift

研 究 生：蘇敬堯

指導教授：陳伯寧　教授

中 華 民 國 九 十 八 年 七 月

去尾迴旋碼結合解碼前環狀位移之解碼演算法

Decoding the Tail-Biting Convolutional Codes with Pre-Decoding Circular Shift

研 究 生：蘇敬堯　　Student：Ching-Yao Su

指導教授：陳伯寧　教授　　Advisor：Po-Ning Chen

國 立 交 通 大 學

電 信 工 程 學 系 碩 士 班

碩 士 論 文

A Thesis

Submitted to Institute of and Communication Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Communication Engineering

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

去尾迴旋碼結合解碼前環狀位移之解碼演算法

研究生：蘇敬堯　　指導教授：陳伯寧　教授

國立交通大學

電信工程學系碩士班

中文摘要

由於去尾迴旋碼的網格具有環狀迴旋不改變結構的特性，因此解碼前作環狀位移(搭配解碼後的環狀位移回覆)並不會影響其解碼結果。初步模擬結果顯示，適當的解碼前環狀位移確實能夠改善解碼效能以及解碼複雜度。因此在本論文中，我們提出位移維特比演算法(shifting Viterbi algorithm)和位移環狀解碼演算法(shifting circular decoding algorithm)搭配「等權重」以及「不等權重」之位移量搜尋法。經由模擬比較不進行解碼前環狀位移之環狀解碼演算法(circular decoding algorithm)，使用位移環狀解碼演算法可以明顯縮減可達到接近最大概率(maximum-likelihood)效能的前訓練窗(forward training window)與後訓練窗(backward training window)的大小。最後，我們提出可達最大概率效能的訓練窗大小的理論預測方法。

Decoding the Tail-Biting Convolutional Codes with Pre-Decoding Circular Shift

Student: Ching-Yao Su Advisor: Prof. Po-Ning Chen

Institute of Communication Engineering

National Chiao Tung University

Abstract

By noting that the convolutional tail-biting code (CTBC) can be represented by a circular-free trellis structure, pre-decoding circular shift (together with the post-decoding shift back) will not change its decoding procedure. Simulations in the literature have already shown that the decoding performance as well as decoding complexity can be apparently improved by a proper pre-decoding circular shift. In this thesis, we proposed the shifting Viterbi algorithm and the shifting circular decoding algorithm using equal-weight and unequal-weight pre-decoding shift methods. We then show empirically that our methods can reduce the forward and backward training window sizes required for near maximum-likelihood performance. We also provide an intuitive analytical approach to determine the training window sizes required for near optimal performance.

Acknowledgements

I would like to express my gratitude to the advisor, Prof. Po-Ning Chen, for his patient guidance and support. My gratitude also goes to Prof. Yunghsiang S. Han for his fruitful discussion during my research. To Shin-Lin Shieh, thanks for your opinions and suggestions in our discussion. You make me understand more about the research topic. To Shih-Wei Wang and Chien Yao, you always listen to my problem, give me much help and more advises, and encourage me when I am depressed. This thesis would not have been possible without their help. I would also like to thank the Department of Communication Engineering, National Chiao Tung University, for providing many resources for study and research. Finally, I would like to thank all NTL labmates for their support and friendship, as well as all those who have helped me in the past two years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In 1955, convolutional codes were first introduced by Elias [2]. Later, Viterbi [12] proposed a maximum likelihood (ML) decoding algorithm for convolutional codes in 1967. Till now, convolutional codes are perhaps the most popular codes in modern digital communications.

In the encoding process of convolutional codes, a certain number of zeros is often required to be appended at the end of the information sequence in order to clear the contents of encoder shift registers. It can be anticipated that the code rate loss introduced by these appended zero tail-bits is negligible for sufficiently long information sequence. However, as long as a short information block length is considered, code rate loss due to zero tail bits become significant.

In the literature, several approaches have been proposed to resolve the code rate loss of the convolutional zero-tail codes (CZTC), such as Direct Truncation, Puncturing [14] and Tail-biting [10, 13]. Among them, the convolutional tail-biting codes (CTBC) avoid the code rate loss by directly "biting" out the zero tail-bits. In comparison with the CZTC, the CTBC always starts from and ends at the same state, but the starting and ending state is not necessary the all-zero state. For this reason, the Viterbi algorithm (VA) in its original

form cannot be applied to obtain the maximum likelihood (ML) solution of the CTBC.

Due to the simplicity in the implementation of the VA, most of the decoding algorithms for the CTBC [10, 13, 9, 8, 1] were modified from the VA. Specifically, Wang *et al.* extended the received codeword by repeating a portion of it, and applied the VA to the extended code trellis. Their algorithm thus provided a tradeoff between the performance and the complexity by varying the length of the repetitive portion [13]. Shao *et al.* proposed to wrap-around several code trellises, and resulted the least average complexity algorithm thus far with near optimal performance [10]. In 2000, by adding forward and backward training windows, W. Sung devised a circular decoding algorithm that can achieve near optimal performance [11]. Very recently, Shankar *et al.* developed an ML two-phase decoding algorithm of practical decoding complexity by first applying the VA, followed by the algorithm A* at the second phase [9].

Because the CTBC code trellis has circular-free property, it is not necessary to decode the received sequence at its original order if the receiver actually starts the decoding process after the reception of the entire received sequence. One can circularly shift the received sequence before the decoding process begins to potentially improve the decoding complexity or performance for suboptimal decoding algorithm. In order to resume the order of transmitted sequence, the decoding output should be reversely shifted back the same number of times. It should be noted that the decoding trellis remains the same for the shifted sequence as it is circular-free. As such, Handlery *et al.* proposed an equal-weighted average method to determine the proper number of shifts applied to the received sequence [5]. Similar idea has also been applied to the two-phase decoding algorithm, which was shown by simulations that significant decoding complexity reduction can be obtained by properly shifting the received sequence before decoding begins [3].

1.2 Contribution of the Research

In this thesis, we proposed a new unequal-weighted average method to determine the number of pre-decoding shifts on the received sequence. We then found by simulations that the forward and backward training window sizes requiring for near ML performance in the circular decoding algorithm of W. Sung can be reduced when the decoding process starts from the positions of the received sequence, which are suggested by weighted average methods. We subsequently analyze the forward and backward training window sizes required for near-optimal performance. Our analytical results conform to those obtained by simulations.

Throughout the thesis, we will use *shifting VA* to refer to the decoding algorithm that shifts the received sequence before the decoding process begins but does not employ the forward and backward training windows. The decoding algorithm that shifts the received sequence before the decoding process begins, and at the same time, adds the forward and backward training windows, will be referred to as *shifting circular decoding algorithm*. Details will be given in Chapter 2.

1.3 Organization of the Thesis

The thesis is organized as follows. In Chapter 2, the decoding algorithms employed in this thesis are introduced, and the proposed method to determine the number of pre-decoding shifts is presented. In Chapter 3, the simulation results are summarized and remarked. In Chapter 4, the forward and backward training window sizes required for near-optimal performance are analyzed for shifting circular decoding algorithm. Chapter 5 concludes the thesis.

Chapter 2

Circular Decoding Algorithm with Dynamic Starting Position

In this chapter, we present the weighted average methods to determine the number of pre-decoding circular shifts on the received sequence. We then examine the effect of using these suggestive pre-decoding shift numbers on the received sequence in *shifting VA* and *shifting circular decoding algorithm*.

2.1 Definitions and Notations

Let $\mathscr{C}$ be an $(n, 1, m)$ CTBC, where n is the number of output bits per information bit, and m is the memory order of the CTBC encoder. Assume that the length of the input sequence is L. Then, the encoding code trellis of $\mathscr{C}$ has $L + 1$ levels. Notably, for shifting VA, the decoding code trellis of $\mathscr{C}$ also has $L + 1$ levels (being indexed from level 0 to level L), but for shifting circular decoding algorithm, the number of levels in the decoding code trellis becomes of $F + L + B + 1$ levels , where F and B are the sizes of the forward and backward windows, respectively. For convenience, the level index will always start at zero; hence, the largest level index in the decoding code trellis of shifting circular decoding algorithm is $F + L + B$. Denote the state space at each level by $\mathcal{S}$. Obviously, $|\mathcal{S}| = 2^m$.

A legitimate tail-biting path for the codeword of $\mathscr{C}$ should constrain on the same initial and final state. By relaxing such constraint, the inclusion of paths that start and end at different states result in a supper code $\mathscr{C}_s$ of $\mathscr{C}$. For notational convenience, we denote the binary input information sequence by $\boldsymbol{z} \triangleq (z_0, z_1, \ldots, z_{L-1}) \in \{0,1\}^L$, the binary codeword of $\mathscr{C}$ by $\boldsymbol{v} \triangleq (v_0, v_1, \ldots, v_{N-1}) \in \{0,1\}^N$, and the original received vector by $\boldsymbol{r} \triangleq (r_0, r_1, \ldots, r_{N-1})$, where $N = nL$.

To facilitate the following description of the decoding process, we denote by $\rho_{t_1,t_2}(s)$ the portion of the survivor path between levels t_1 and t_2, where $t_1 < t_2$, which starts at level t_1, and ends at state s at level t_2. Likewise, we denote by $\beta_{t_1,t_2}(s)$ the state of the survivor path at level t_1, which ends at state s at level t_2. Note that there is only one survivor path ending at state s at level t_2; hence, $\beta_{t_1,t_2}(s)$ is uniquely determined by s, t_1 and t_2. The metric associated with $\rho_{t_1,t_2}(s)$ is denoted by $m_{t_1,t_2}(s)$. By these notations, the best path ρ_{best} in the decoding code trellis of super code $\mathscr{C}_s$ is the one that gives the smallest $m_{0,L}(s)$ (respectively, $m_{0,F+L+B}(s)$) for all $s \in \mathcal{S}$ for shifting VA (respectively, shifting circular decoding algorithm). The path metric m_{best} associated with ρ_{best} is thus given by:

$$m_{\text{best}} \triangleq \begin{cases} \min\limits_{s\in\mathcal{S}} m_{0,L}(s) & \text{for shifting VA;} \\ \min\limits_{s\in\mathcal{S}} m_{0,F+L+B}(s) & \text{for shifting circular decoding algorithm.} \end{cases}$$

As aforementioned, a legitimate code path should starts from and ends at the same state, which is usually termed *tail-biting path* in the literature. Thus, for shifting VA, the best tail-biting path $\rho_{\text{TB,best}}$ is the one that gives the smallest $m_{0,L}(s)$ with $\beta_{0,L}(s) = s$. Likewise, for shifting circular decoding algorithm, the best tail-biting path $\rho_{\text{TB,best}}$ is the one that gives the smallest $m_{0,F+L+B}(s)$ with $\beta_{F,F+L+B}(s) = \beta_{F+L,F+L+B}(s)$.

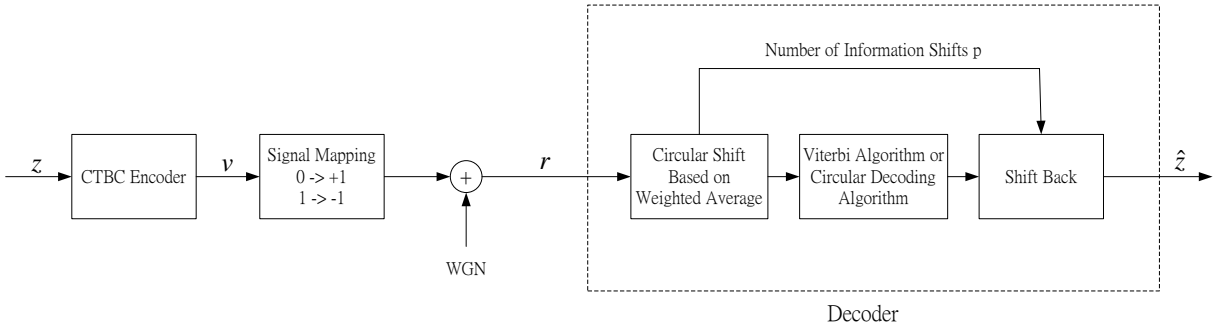


Figure 2.1: System Model in AWGN Channel

2.2 Determination of Number of Pre-Decoding Circular Shift

Figure 2.1 depicts our system model. The encoded sequence is antipodally transmitted, and then is corrupted by the additive white Gaussian noise (AWGN). The receiver will circular shift the received sequence np times before the decoding process begins. It will then shifts it back, and outputs the final decision.

The weighted average method to determine the number of information shifts p is introduced. Since p represents the number of information shifts, the received sequence will in fact be circularly shifted $n \times p$ times.

Step 1. Set the weighted average window W*, and the weighting coefficients* $(w_0, w_1, \ldots, w_{W-1})$.

Step 2. For received sequence $\boldsymbol{r} \triangleq (r_0, r_1, \ldots, r_{N-1})$*, compute for* $0 \le \ell < L$*,*

$$R_\ell = \sum_{j=n\ell}^{n\ell+nW-1} w_{\lfloor (j-n\ell)/n \rfloor} |r_{j \bmod N}|.$$

Step 3. Let $p = \arg\max_\ell R_\ell$.[1]

[1] If the maximum of R_ℓ is achieved by more than one ℓ such as in binary erasure channels, then just

Step 4. Output the shifted received sequence as $(r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0, \ldots, r_{np-1})$.

In the above algorithm, we propose to average the absolute values of nW consecutive received scalars. This is specific for AWGN channels. For a general channel, $|r_j|$ should be replaced by *reliability* $|\phi_j|$, where

$$\phi_j \triangleq \ln \frac{\Pr(r_j|v_j = 0)}{\Pr(r_j|v_j = 1)}. \tag{2.1}$$

It can be shown that the larger reliability ϕ_j is, the higher the probability of correct decision based on the hard-decision

$$y_j \triangleq \begin{cases} 1, & \text{if } \phi_j < 0; \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

Therefore, to begin the decoding process from $n \times W$ most reliable received scalars may possibly speed up the decoding process and improve the performance for suboptimal decoding algorithms. Simulations do confirm our conjecture.

It should be noted that for AWGN channels and antipodal transmission, r_j is proportional to ϕ_j; hence, we can replace ϕ_j by r_j for simplicity. Also, since for AWGN channels and antipodal transmission,

$$\Pr[r_j > \tau | v_j = 1] = \Pr[r_j < -\tau | v_j = 0] \tag{2.3}$$

for any positive τ, the degrees of reliability respectively for positive r_j and for negative r_j are symmetric. Thus, no adjustment based on their signs is necessary when we perform average on nW consecutive $|r_j|$. In case (2.3) is violated, one may need to adjust the absolute value of the reliability based on their signs according to hard-decision error probability.

randomly pick one.

Also, for shifting circular decoding algorithm, we will additional perform $p = p + W/2$ because the position that is intended to located is in the middle of the forward subwindow coefficients $w_0, w_1, \ldots, w_{W/2-1}$ and the backward subwindow coefficients $w_{W/2}, w_{W/2+1}, \ldots, w_{W-1}$. Please refer to the next section for more details.

We close this section by remarking that in its implementation, p can be any integer number between 0 and $L-1$. We however will restrict p to be a multiple of W in our analysis in Chapter 4. By adding this "constraint", the averaged sequence $R_0, R_W, R_{2W}, \ldots$ becomes i.i.d. in statistics, which greatly simplify our subsequence analysis.

2.3 Derivation of Weighting Coefficients

For a given CTBC code structure, the probability of correct path being eliminated owing to the other paths during the decoding process is different at each level. Based on this observation, the *equal-weight* method proposed by Handlery *et al.* may be further improved by considering *unequal weights* that are proportional to, e.g., the correct-path elimination probability at each level. The intuition behind our provision is that a higher erroneous elimination probability at a certain level should be compensated by a higher reliability at the respective level; hence, with a higher weight coefficient at that position, the position that can well compensate the erroneous elimination according to the code structure can be located.

As an example in the $(2, 1, 3)$ CTBC trellis in Fig. 2.2, given that the all-zero codeword is transmitted, the path from state 1 at level 0 to state 0 at level 1 may eliminate the correct

all-zero path at level 1, which will occur with probability[2]

$$\begin{aligned} q_0 &= \Pr\left[(y_0 \oplus 0)|\phi_0| + (y_1 \oplus 0)|\phi_1| > (y_0 \oplus 1)|\phi_0| + (y_1 \oplus 1)|\phi_1| \middle| v_0 = v_1 = 0\right] \\ &= \Pr[r_0 + r_1 < 0 | v_0 = v_1 = 0] \\ &= Q(\sqrt{2}/\sigma), \end{aligned}$$

where σ^2 is the variance of the additive Gaussian noise, and

$$Q(x) \triangleq \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

is the Q-function. At level 2, the correct all-zero path may be eliminated by either the path from state 2 at level 0 to state 0 at level 2 or the path from state 3 at level 0 to state 0 at level 2, for which the probability of occurrence is given by

$$q_1 = \Pr[r_1 + r_2 + r_3 < 0 \text{ or } r_0 + r_2 + r_3 < 0 | v_0 = v_1 = v_2 = v_3 = 0].$$

By repeating this procedure, the correct all-zero path may be eliminated due to four paths ending at level 3, which are respectively the ones from state 4, 5, 6 and 7 at level 0 to state 0 at level 3. As such, we can pre-calculate these probabilities as $q_0, q_1, q_2, \ldots, q_{L-1}$. Notably, the block error probability is actually less than $q_0 + q_1 + q_2 + \cdots + q_{L-1}$ because, for example, the calculation of q_2 does not exclude the event that q_1 concerns, i.e., $r_0 + r_1 > 0$. As a result, our weighting coefficients for shifting VA will be set intuitively as:

$$(w_0, w_1, \ldots, w_{W-1}) = \frac{1}{q_0} (q_0, q_1, q_2, \ldots, q_{W-1})$$

corresponding to a certain operational signal-to-noise ratio such as $E_b/N_0 = 1$ dB. The calculations of erroneous elimination probabilities for $(3, 1, 6)$ and $(2, 1, 6)$ CTBCs are detailed in Appendix A.

[2]The path metric we employed in our decoding is defined as [4]

$$\mu(x_0, x_1, \ldots, x_{\ell n-1}) = \sum_{j=0}^{\ell n-1} (y_j \oplus x_j)|\phi_j|,$$

where $(x_0, x_1, \ldots, x_{\ell n-1})$ is the path label, "$\oplus$" is the module-2 addition operation, and ϕ_j and y_j are respectively defined in (2.1) and (2.2).

For shifting circular decoding algorithm, the weighting coefficients will be divided into two groups: (i) forward subwindow coefficients $w_0, w_1, \ldots, w_{W/2-1}$ and (ii) backward subwindow coefficients $w_{W/2}, w_{W/2+1}, \ldots, w_{W-1}$, where W is taken to be an even number. The backward subwindow coefficients will be defined the same as the coefficients for shifting VA, i.e.,

$$(w_{W/2}, w_{W/2+1}, \ldots, w_{W-1}) = \frac{1}{q_0} \left(q_0, q_1, q_2, \ldots, q_{W/2}\right).$$

This part accounts for the influence from erroneous elimination. The forward subwindow coefficients consider the backward structure in which the accumulated metric of the correct path is larger than those of the other paths. Take Fig. 2.3 as an example. There will be two branches emitting from state 0 at level $L-1$. The one marked with 00 belongs to the correct path. Then, the weight $w_{W/2-1}$ will be proportional to the probability that the metric corresponding to the path from state 0 at level $L-1$ to state 4 at level L is smaller than that of the path from state 0 at level $L-1$ to state 0 at level L. Again, $w_{W/2-2}$ will be proportional to the probability that the metric corresponding to the path from state 0 at level $L-2$ to state 0 at level L is larger than either that of the path from state 0 at level $L-2$ to state 2 at level L or that of the path from state 0 at level $L-2$ to state 6 at level L. Detailed calculations of these window values for $(3,1,6)$ and $(2,1,6)$ CTBCs are given in Appendix A.

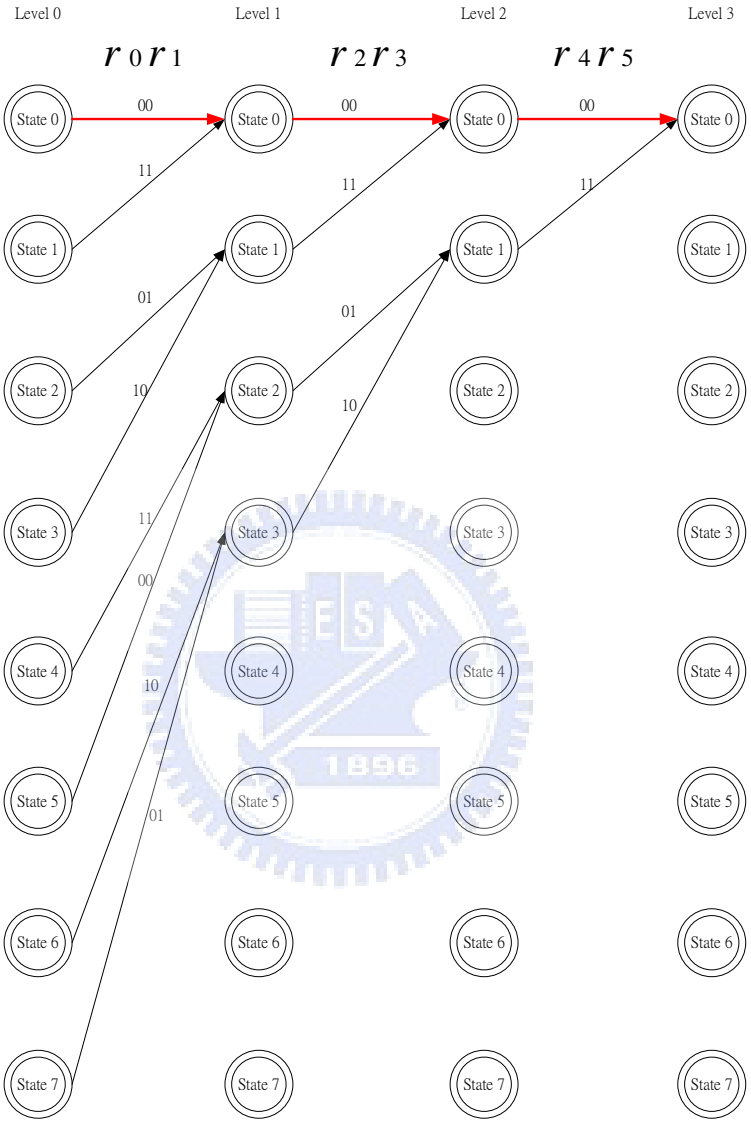


Figure 2.2: The forward trellis of (2,1,3) CTBC

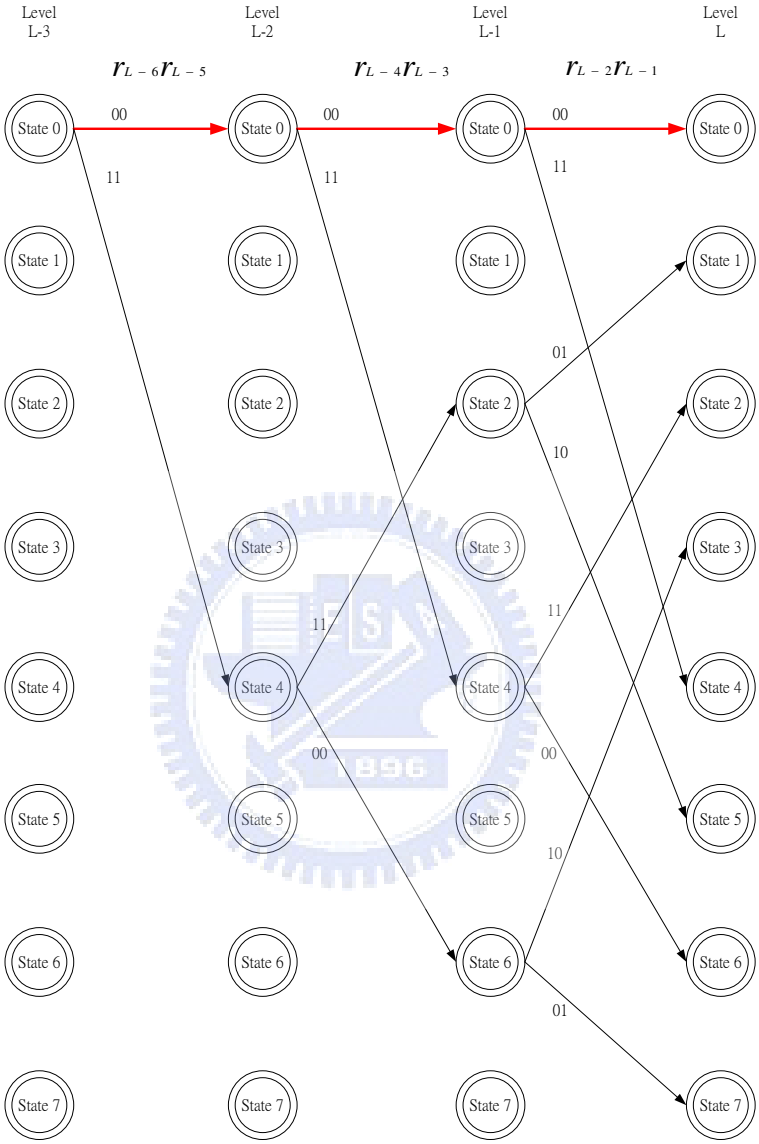


Figure 2.3: The backward trellis of (2,1,3) CTBC

2.4 Shifting Viterbi Algorithm

The shifting VA is simply to apply the VA after circularly shifting the received sequence. For completeness, its procedure is given below.

Step 1. For the shifted received sequence $\boldsymbol{r'}= (r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0, \ldots, r_{np-1})$*, apply the VA with initial metrics being zero for all states.*

Step 2. Output the information sequence corresponding to $\rho_{\text{TB,best}}$ *if it exists. Otherwise, output the information sequence corresponding to* ρ_{best}*.*

As shown in Fig. 2.1, the output information sequence of the above algorithm is the shifted version of the true estimate at the receiver. Hence, the entire decoding process is completed by the last "shift back" operation.

2.5 Shifting Circular Decoding Algorithm

For shifting circular decoding algorithm, in order to account for the effect of the forward and backward training windows denoted respectively by F and B, we will add $W/2$ to p before the decoding process begins as indicated in Footnote 1 (cf. Page 6). Because it requires additional working load to find $\rho_{\text{TB,best}}$, namely, the examination of $\beta_{F,F+L+B}(s) = \beta_{F+L,F+L+B}(s)$ for all $s \in \mathcal{S}$, and because the error performance by merely considering the information sequence corresponding to ρ_{best} is already acceptably good, we will directly output the information sequence corresponding to ρ_{best} without examining $\rho_{\text{TB,best}}$ here. Detailed procedure is summarized in the following.

Step 1. For the shifted received sequence $\boldsymbol{r'}= (r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0, \ldots, r_{np-1})$*, re-index it as*

$$\boldsymbol{r}' = (r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0, \ldots, r_{np-1}) = (r'_0, r'_1, r'_2, \ldots, r'_{N-1}).$$

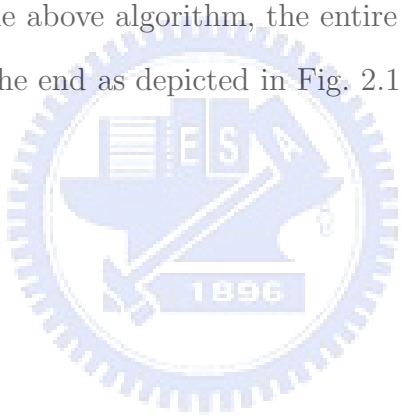
Step 2. Copy the back nF values of $\boldsymbol{r'}$ to the head, and copy the front nB values of $\boldsymbol{r'}$ to the tail. This gives that

$$\boldsymbol{r}'_{TW} = (r'_{N-1-(nF-1)}, \ldots, r'_{N-1}, r'_0, r'_1, \ldots, r'_{N-1}, r'_0, \ldots, r'_{nB-1}).$$

Step 3. For $\boldsymbol{r}'_{TW}$, apply the VA to the super code trellis of $F + L + B + 1$ levels with initial metrics being zero for all states.

Step 4. Retain the information sequence of length $F + L + B + 1$ corresponding to ρ_{best}. Remove the front F bits and back B bits. Output the remaining information sequence of length L.

After the execution of the above algorithm, the entire decoding process is completed by performing "shift back" in the end as depicted in Fig. 2.1.



Chapter 3

Simulation Results

3.1 Performance of Shifting Viterbi Algorithm

In this section, we will compare the performances of the brutal-force maximum-likelihood decoding algorithm, the VA, and the shifting VA. They are respective abbreviated as ML, VA and SVA in the legends. Note that the difference between the VA and the SVA is that no circular shift is performed on the received sequence in the VA. The "unequal-weight" appearing before the SVA indicates that the weighting coefficients derived in Section 2.3 are used.

The CTBCs used in our simulations are respectively the $(3, 1, 6)$ code with generator polynomial (554 744 724) (in octal) and the $(2, 1, 6)$ code with generator polynomial (744 554) (in octal). The weighted average window is taken to be $W = 6$ only. After the binary codeword is mapped onto the antipodal signals, the additive white Gaussian is added. From Figs. 3.1, 3.2 and 3.3, we found that both equal-weight and unequal-weight SVAs improve the performance of the VA. In addition, the proposed unequal-weight SVA performs better than the equal-weight SVA.

From Figs. 3.4, 3.5 and 3.6, we found again that both equal-weight and unequal-weight SVAs improve the performance of the VA. But, it is a little different from those obtained

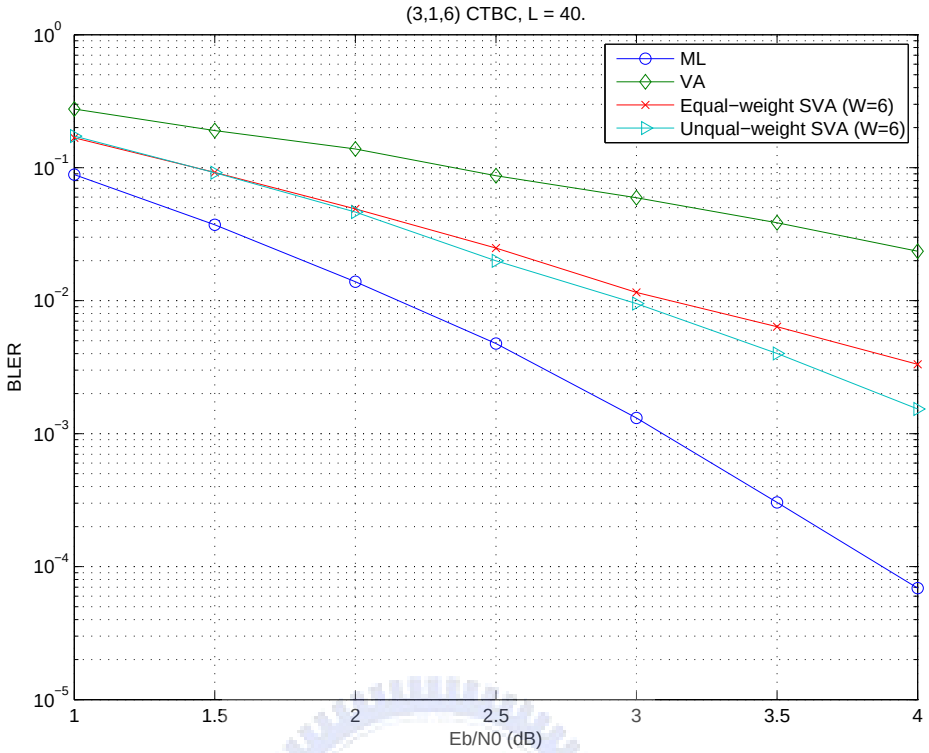


Figure 3.1: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(3, 1, 6)$ CTBC. The length of the information sequence is $L = 40$.

Figs. 3.1, 3.2 and 3.3 that the unequal-weight SVA performs almost the same as the equal-weight SVA. Thus, the code rate could be a factor to result in different behavior between the equal- and unequal-weight SVA. Hence, we further examine the $(3, 1, 4)$ CTBC with generator polynomial (52 66 76) (in octal).

Taking the average window size to be equal to the memory order of the CTBC code (i.e., $W = 4$ for $(3, 1, 4)$ code), we observe from Figs. 3.7, 3.8 and 3.9 that both equal-weight and unequal-weight SVAs outperforms the VA as usual, and unequal-weight SVA performs better than the equal-weight SVA.

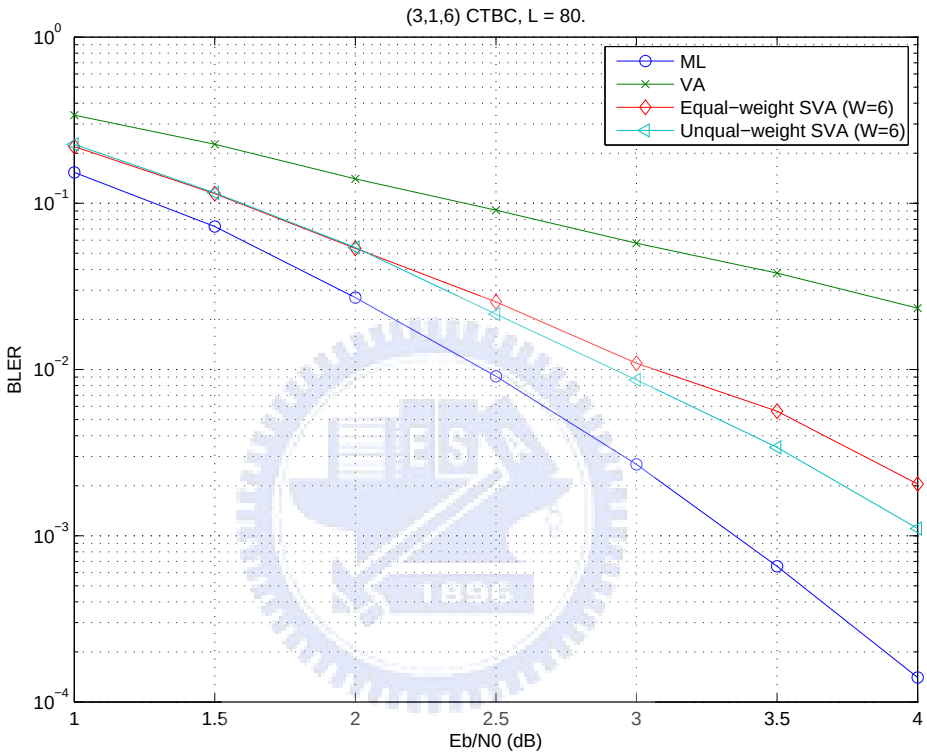


Figure 3.2: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(3, 1, 6)$ CTBC. The length of the information sequence is $L = 80$.

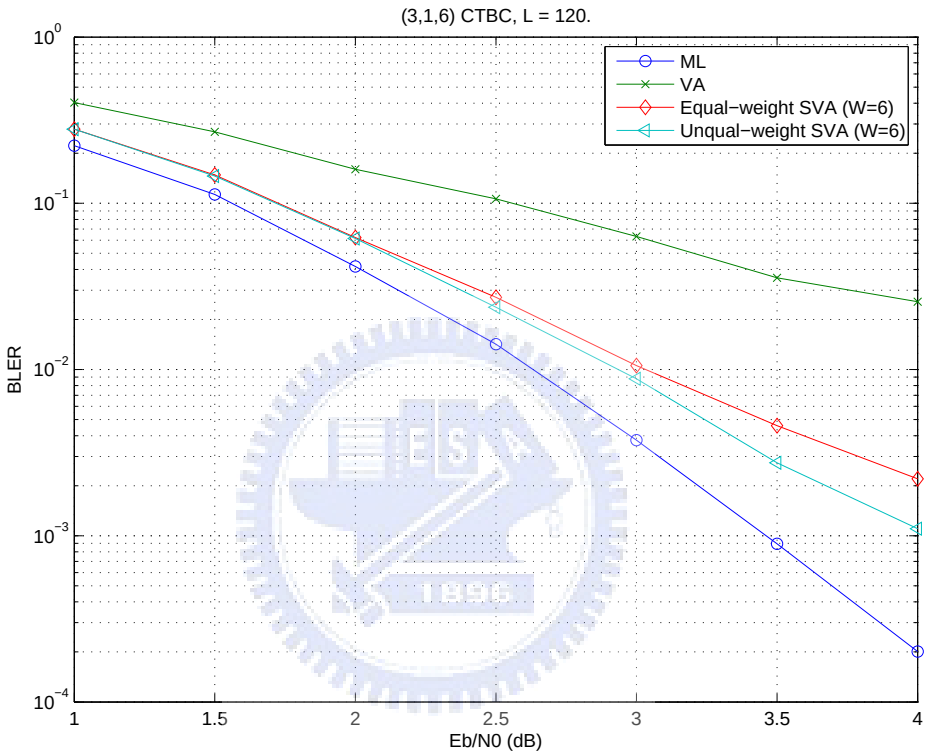


Figure 3.3: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(3, 1, 6)$ CTBC. The length of the information sequence is $L = 120$.

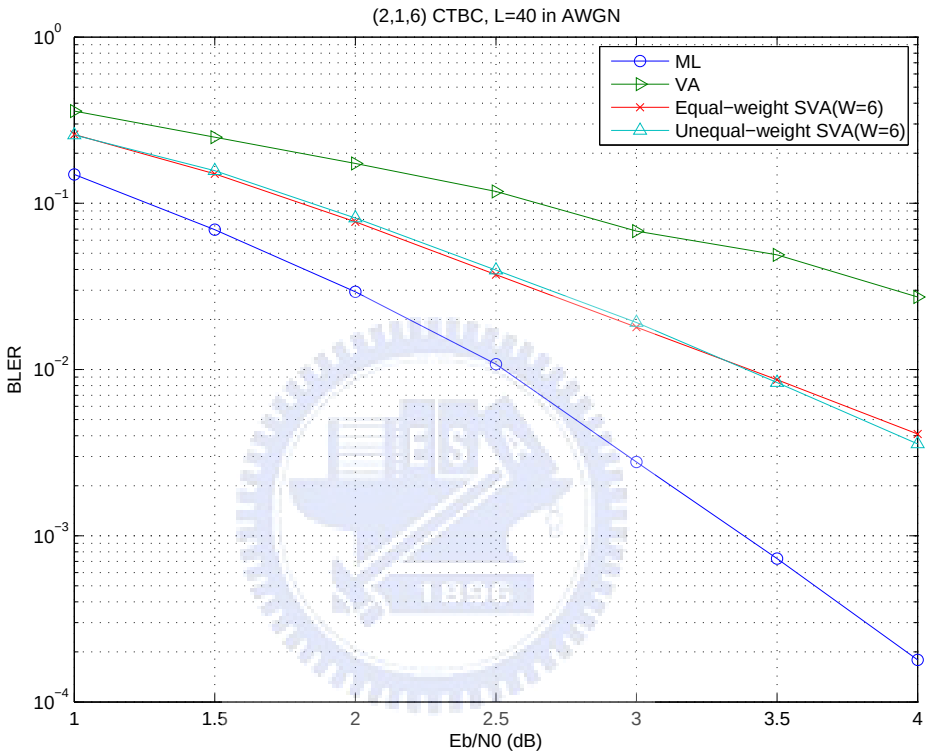


Figure 3.4: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(2, 1, 6)$ CTBC. The length of the information sequence is $L = 40$.

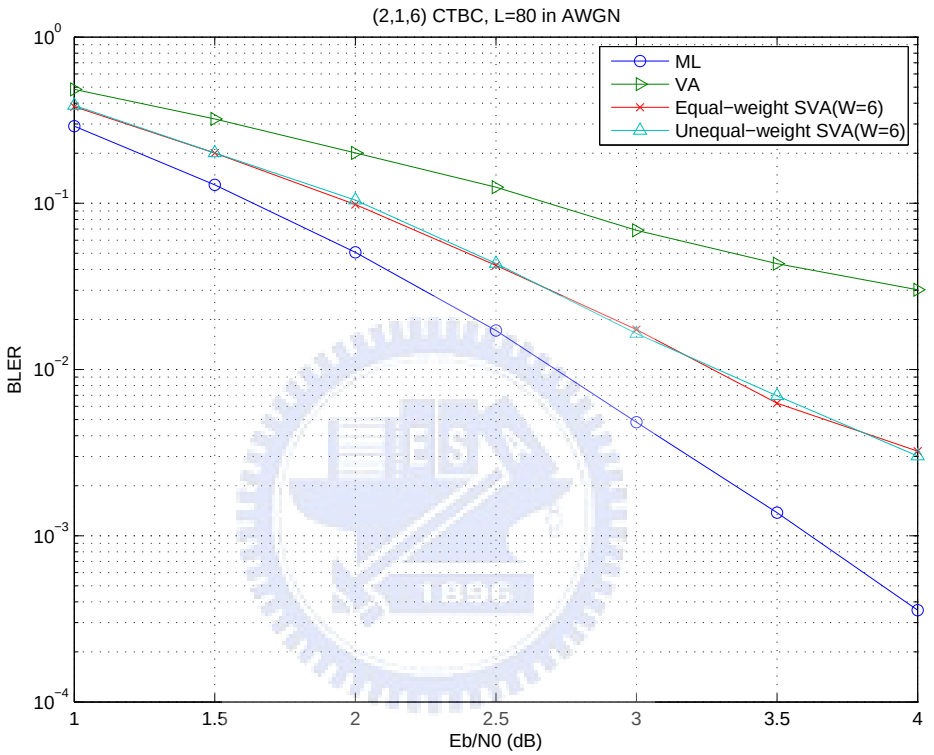


Figure 3.5: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(2, 1, 6)$ CTBC. The length of the information sequence is $L = 80$.

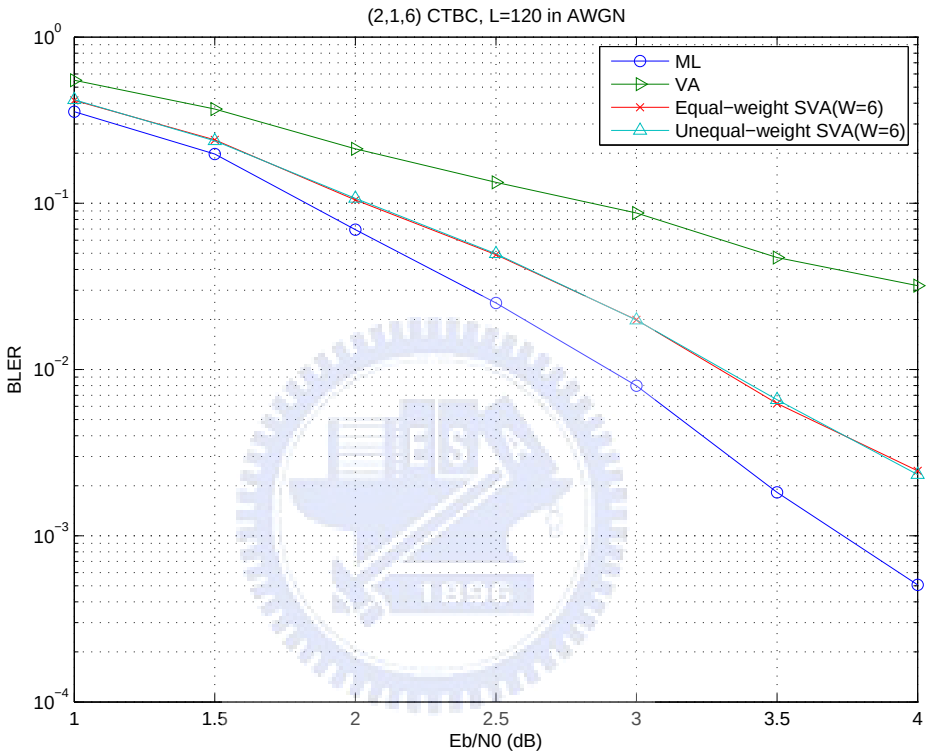


Figure 3.6: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(2, 1, 6)$ CTBC. The length of the information sequence is $L = 120$.

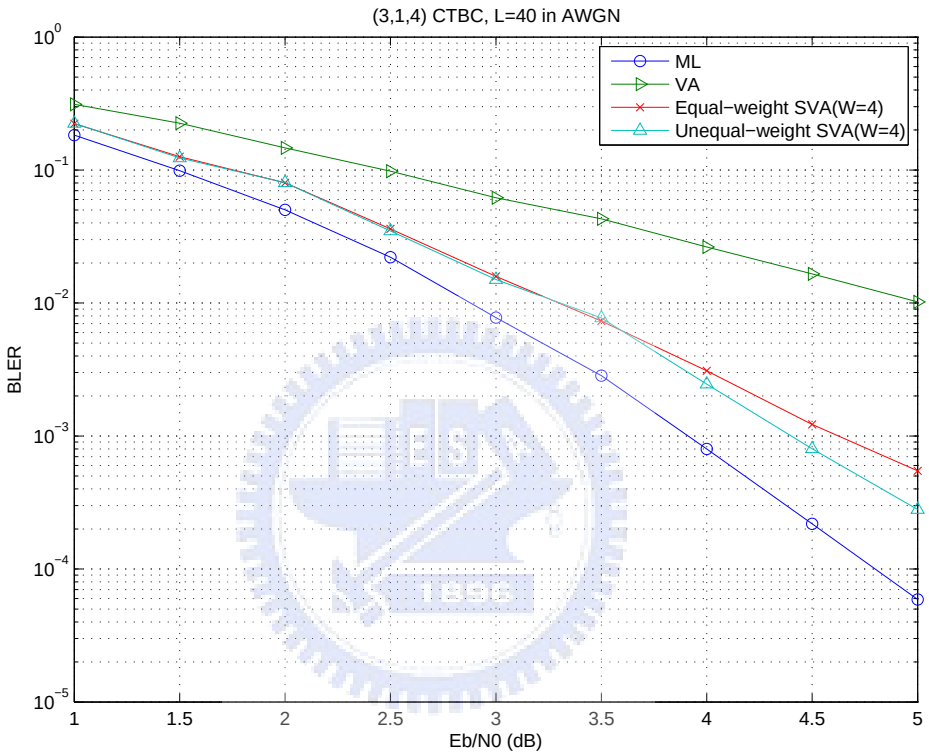


Figure 3.7: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(3, 1, 4)$ CTBC. The length of the information sequence is $L = 40$.

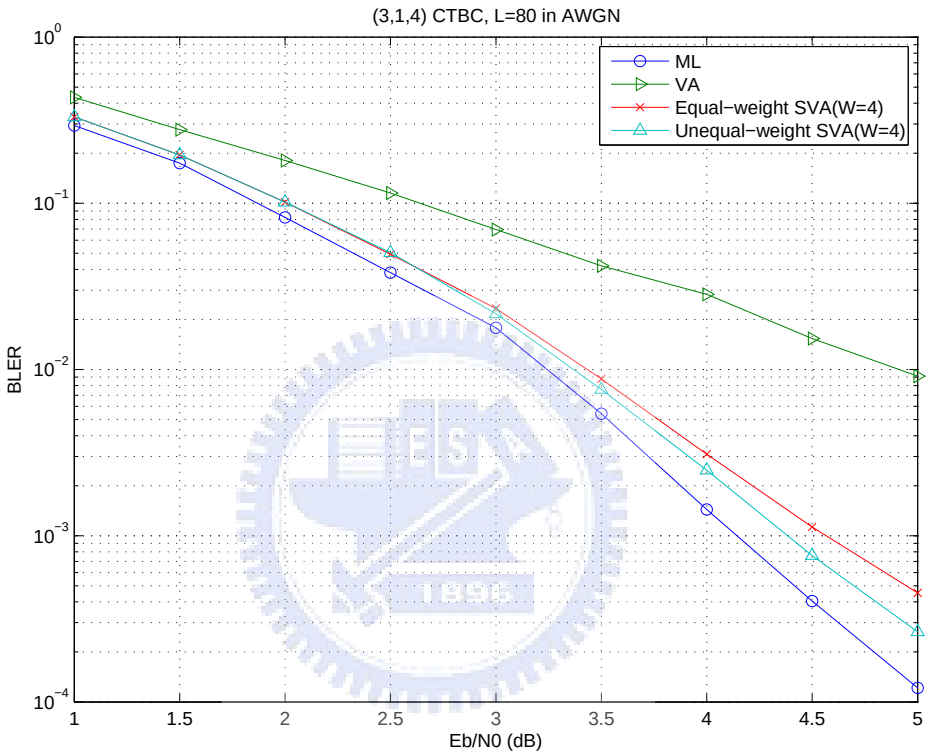


Figure 3.8: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(3, 1, 4)$ CTBC. The length of the information sequence is $L = 80$.

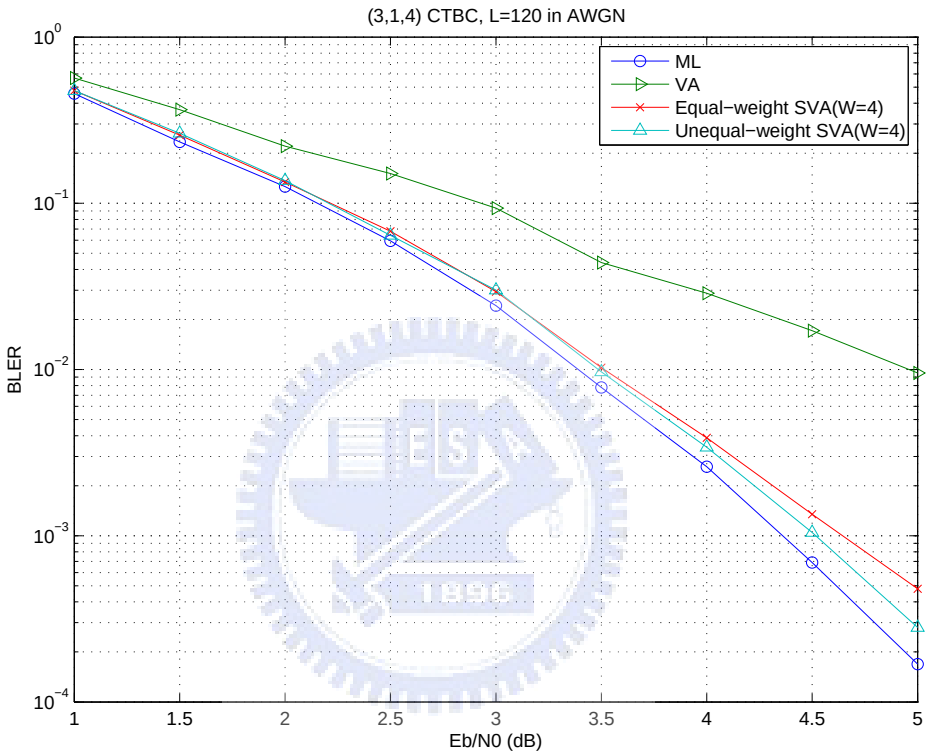


Figure 3.9: Block error rates (BLER) of the maximum-likelihood (ML) decoder, the Viterbi algorithm (VA) and the shifting Viterbi algorithm (SVA) for $(3, 1, 4)$ CTBC. The length of the information sequence is $L = 120$.

3.2 Performance of Shifting Circular Decoding Algorithm

We continue to compare the performances of circular decoding algorithm (CDA) without pre-decoding shifting and shifting circular decoding algorithm (SCDA) for the CTBC codes examined in the previous section. Since we propose to use both forward and backward window, and since it is intuitive to have a window size to cover or accumulate the effect in regard of both the forward window and backward window, the weighted average window size W is taken to be twice of the memory order (instead of one memory-order that is taken for the VA). Both equal-weight and unequal-weight coefficients are examined.

We observe from Figs. 3.10, 3.11 and 3.12 that forward and backward window sizes (F, B) required for near-ML performance are about $(19, 20)$, and are irrelevant to the information length L. This result coincides with that in [11].

Figures 3.13, 3.14 and 3.15 summarize the training window sizes that are sufficient to achieve near-ML performance for equal-weight SCDA. Specifically, Fig. 3.13 shows that the near-ML-performance window sizes can be reduced down to $(F, B) = (12, 18)$ when $L = 40$. Further reduction can be reached down to $(F, B) = (6, 10)$ when L increases to 80. While $L = 120$, the near-ML-performance window sizes reduce to only $(F, B) = (4, 10)$.

By adding the constraint that $(p - W/2)$ must be a multiple of $W = 12$, the near-ML training window sizes become $(F, B) = (12, 18)$ for $L = 40$, $(F, B) = (8, 10)$ for $L = 80$, and $(F, B) = (6, 10)$ when $L = 120$ as indicated in Figs. 3.16, 3.17 and 3.18.

The performances of the unequal-weight shifting circular decoding algorithm are examined in Figs. 3.19, 3.20 and 3.21. These three figures show that the near-ML training window sizes are exactly the same as those obtained for equal-weight shifting circular decoding algorithm except that $(F, B) = (12, 16)$ when $L = 40$.

Next, we turn to the $(2, 1, 6)$ CTBC with generator polynomial (744 554) (in octal) and the weighted average window size $W = 12$. From Figs. 3.22, 3.23 and 3.24, the forward and backward training window sizes (F, B) required for near-ML performance are about $(27, 28)$, and are again irrelevant to the information length L.

For the equal-weight SCDA, Figs. 3.25, 3.26 and 3.27 tell that the near-ML-performance training window sizes are reduced to $(F, B) = (24, 26)$, $(16, 20)$ and $(10, 14)$ respectively for $L = 40$, 80 and 120. These window sizes remains the same even if an additional constraint that $(p - W/2)$ must be a multiple of W is added, except for $(F, B) = (24, 28)$ for $L = 40$ as shown in Figs. 3.28, 3.29 and 3.30.

It remains to examine the performances of the unequal-weight shifting circular decoding algorithm for the $(2, 1, 6)$ CTBC. From Figs. 3.31, 3.32 and 3.33, the near-ML training window sizes are $(F, B) = (20, 26)$ for $L = 40$, $(F, B) = (12, 18)$ for $L = 80$, and $(F, B) = (8, 12)$ for $L = 120$, respectively. This hints that the window sizes required for near-ML performance can be a little reduced by switching from the equal weights to unequal weights.

Now, we consider the last code examined in this section, i.e., the $(3, 1, 4)$ CTBC with generator polynomial (52 66 76) (in octal). From Figs. 3.34, 3.35 and 3.36, the forward and backward training window sizes (F, B) required for near-ML performance are also irrelevant to the information length L, which is $(F, B) = (13, 13)$. From Figs. 3.37, 3.38 and 3.39, we notice that the near-ML-performance window sizes are reduced to almost none as $(F, B) = (8, 8)$, $(2, 4)$ and $(2, 4)$ respectively for $L = 40$, 80 and 120. The constraint that $(p - W/2)$ must be a multiple of $W = 8$ does not change these required (F, B) as indicated in Figs. 3.40, 3.41 and 3.42. In the end, the performances of the unequal-weight SCDA are examined, and are resulted some improvement over equal-weight SCDA. Specifically, from Figs. 3.43, 3.44 and 3.45, the near-ML training window sizes are $(F, B) = (6, 6)$ for $L = 40$, $(F, B) = (2, 4)$ for $L = 80$, and $(F, B) = (2, 2)$ for $L = 120$.

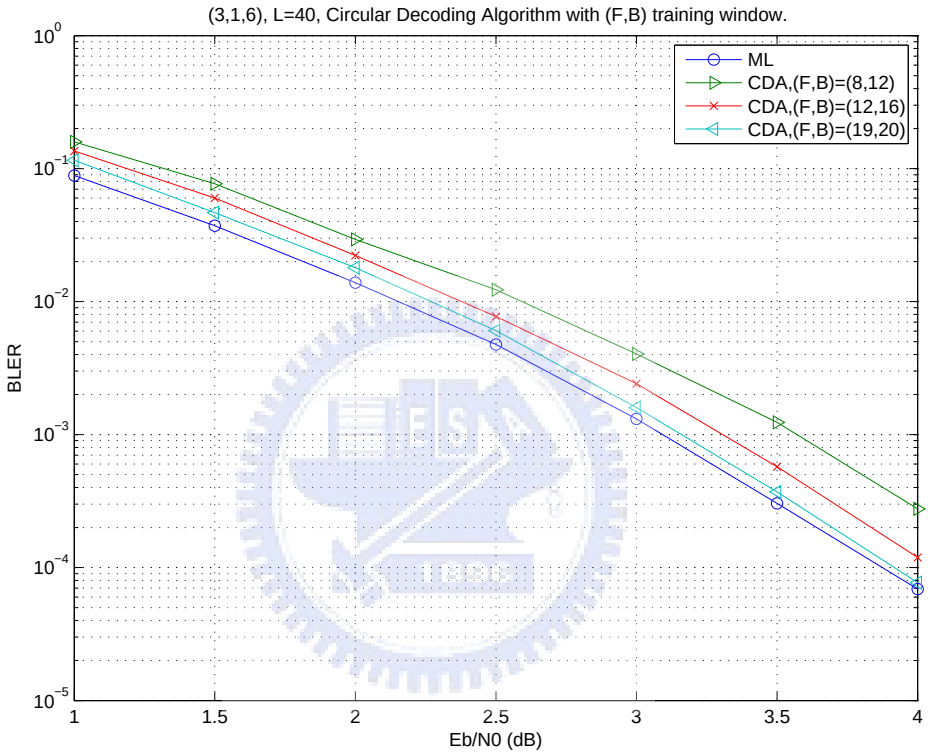


Figure 3.10: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(3, 1, 6)$ CTBC. The length of the information sequence is $L = 40$.

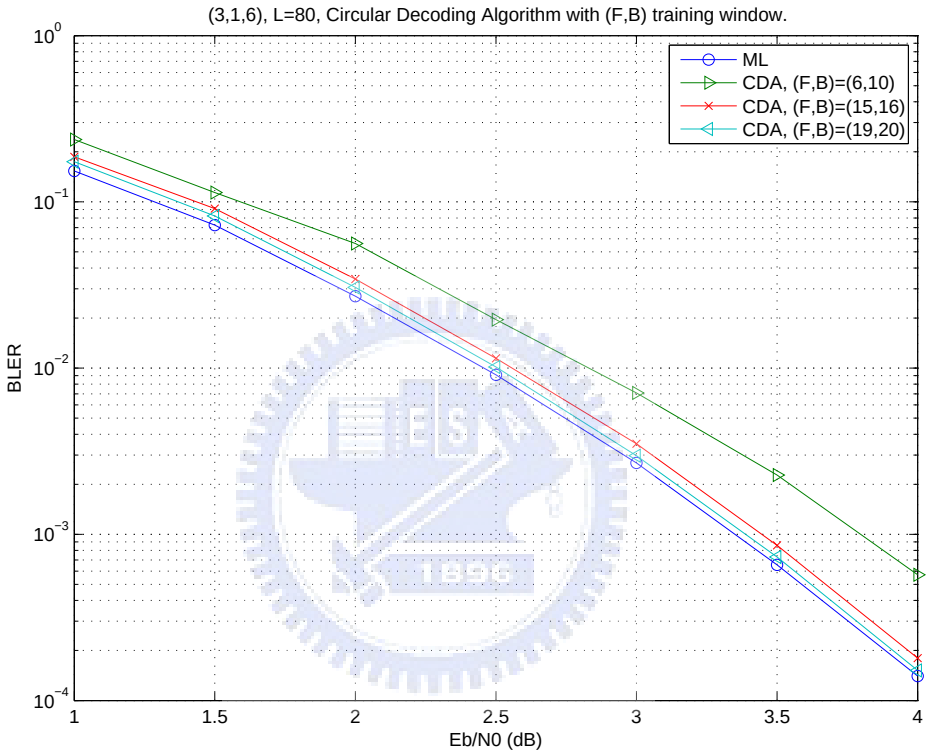


Figure 3.11: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(3, 1, 6)$ CTBC. The length of the information sequence is $L = 80$.

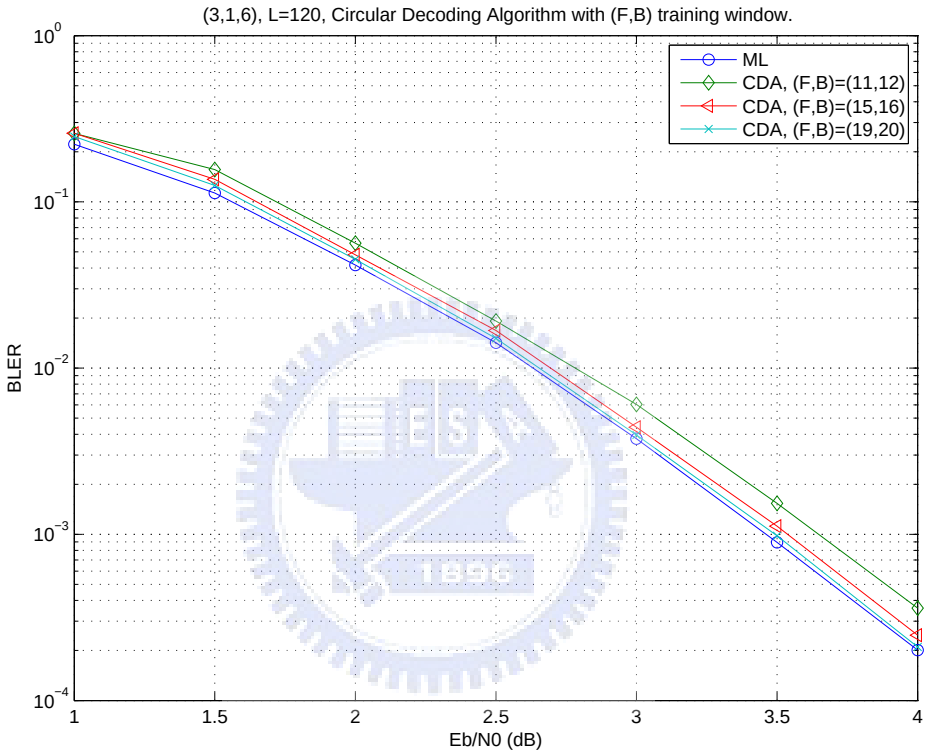


Figure 3.12: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(3, 1, 6)$ CTBC. The length of the information sequence is $L = 120$.

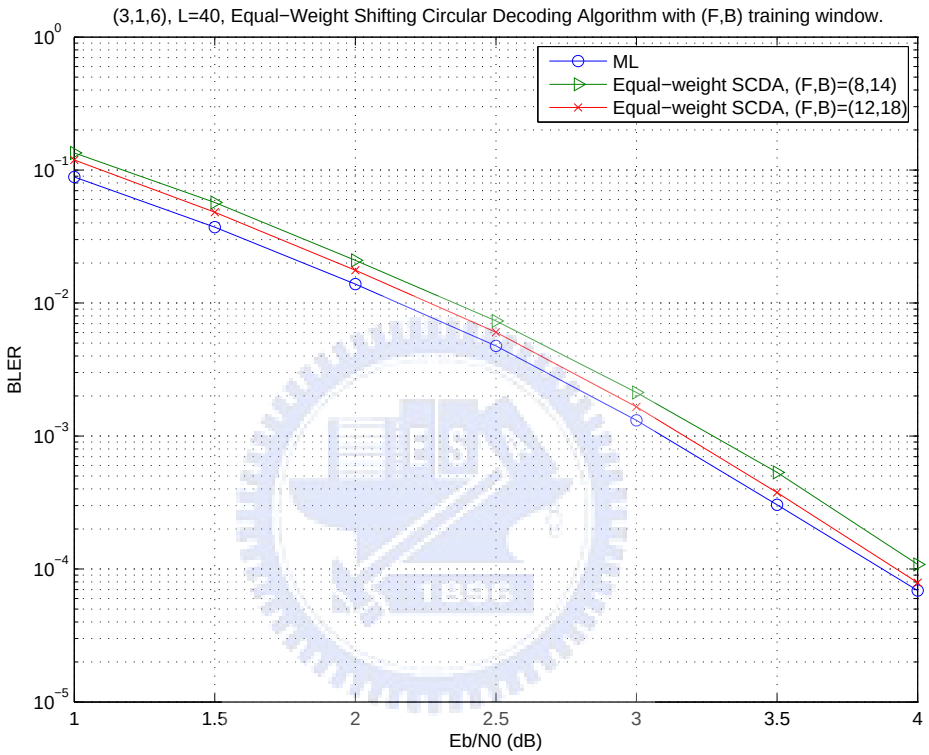


Figure 3.13: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

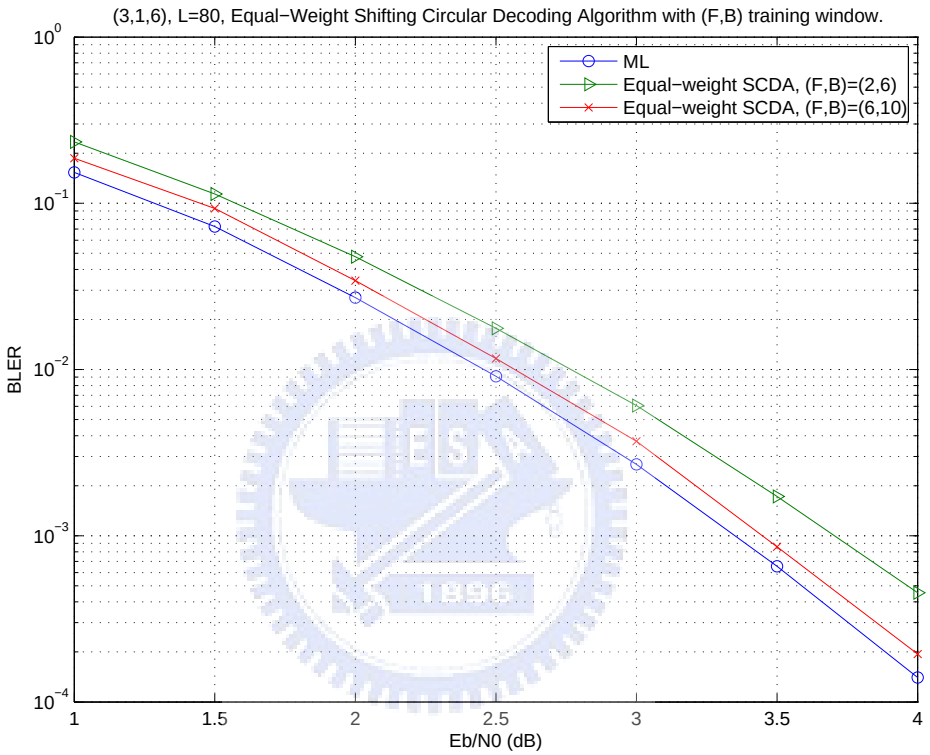


Figure 3.14: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectivley. The length of the information sequence is $L = 80$.

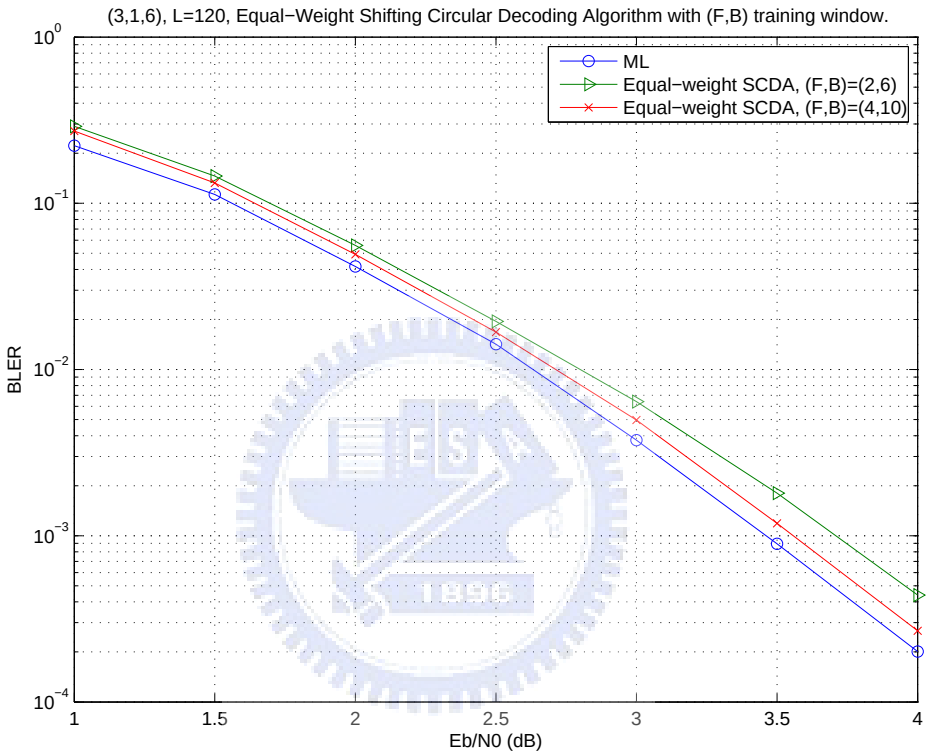


Figure 3.15: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectivley. The length of the information sequence is $L = 120$.

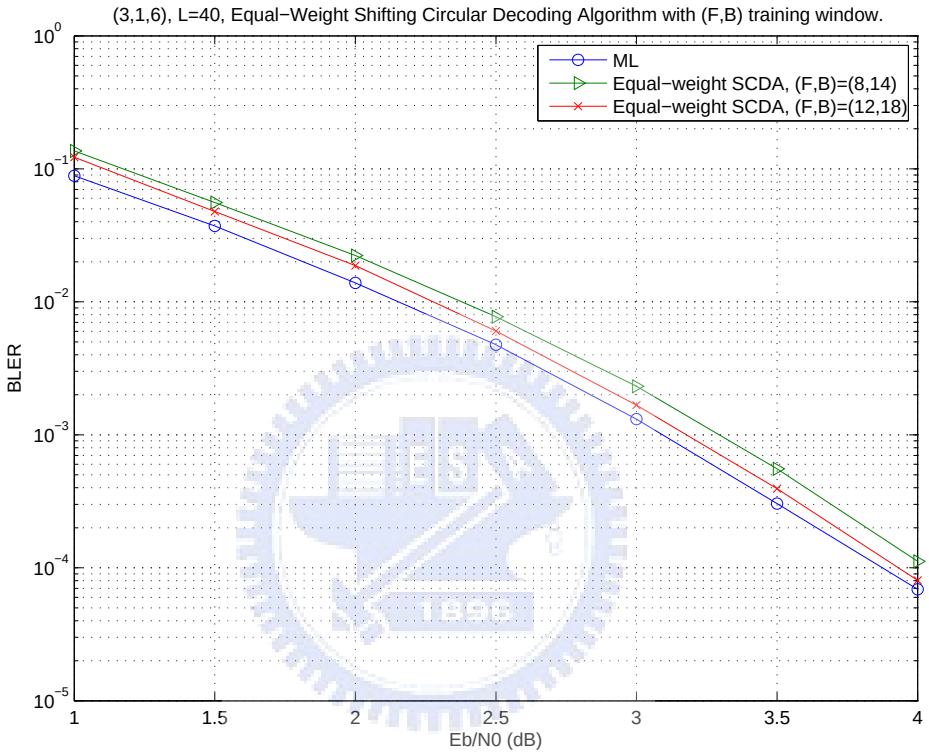


Figure 3.16: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

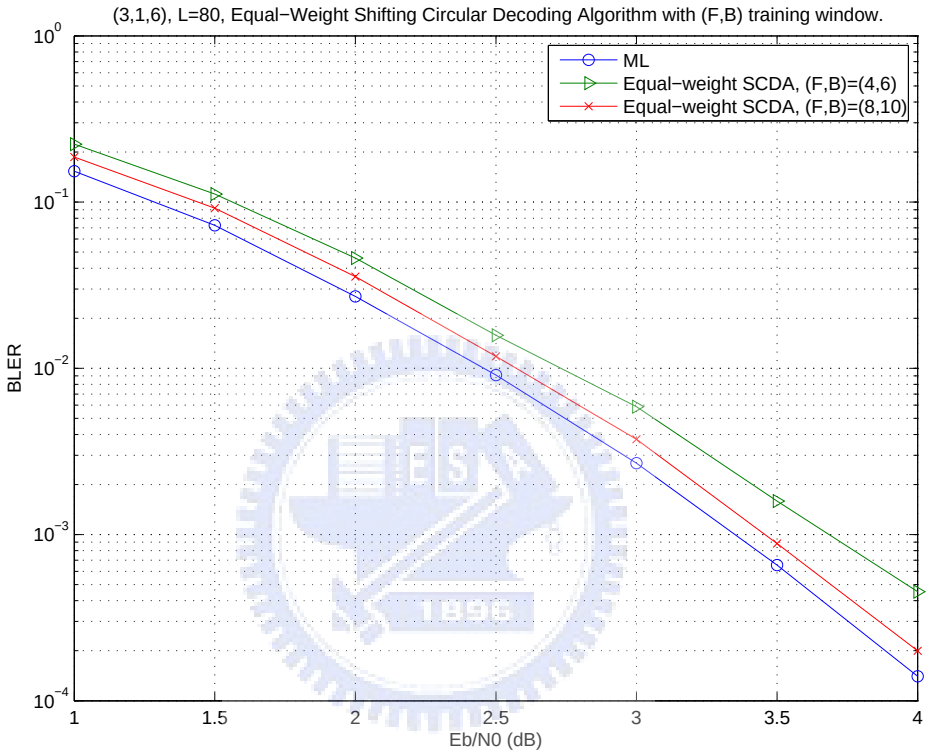


Figure 3.17: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 80$.

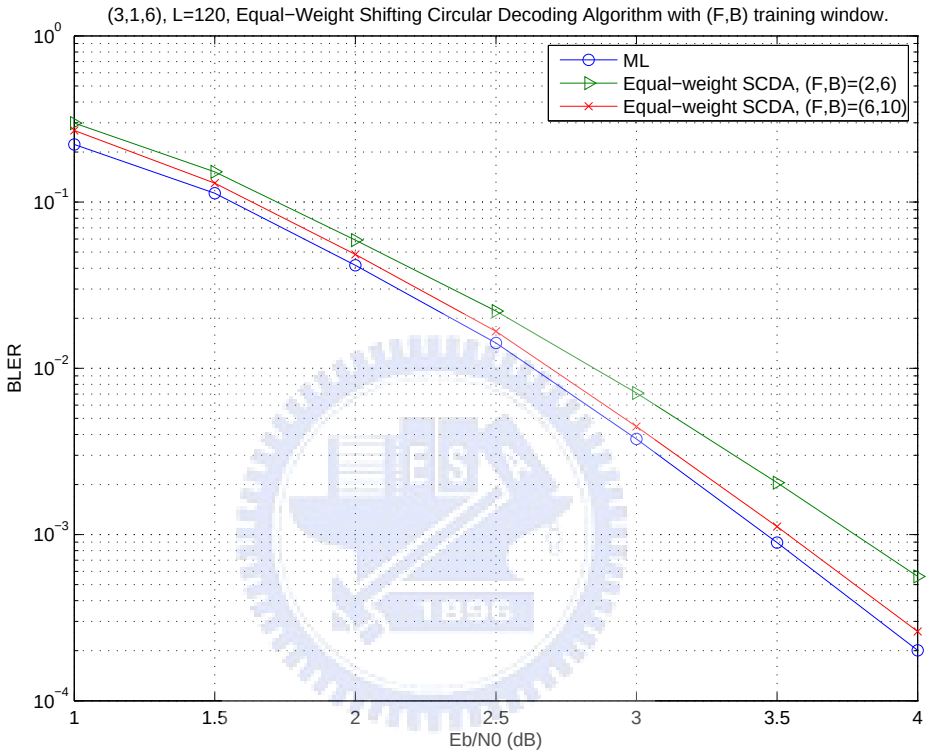


Figure 3.18: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 120$.

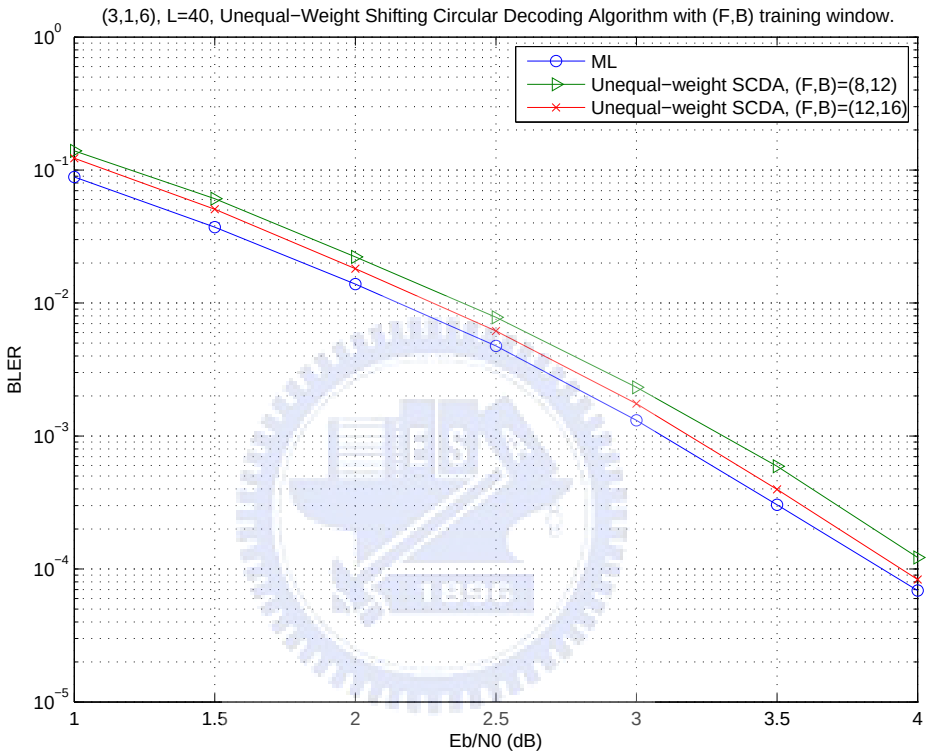


Figure 3.19: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

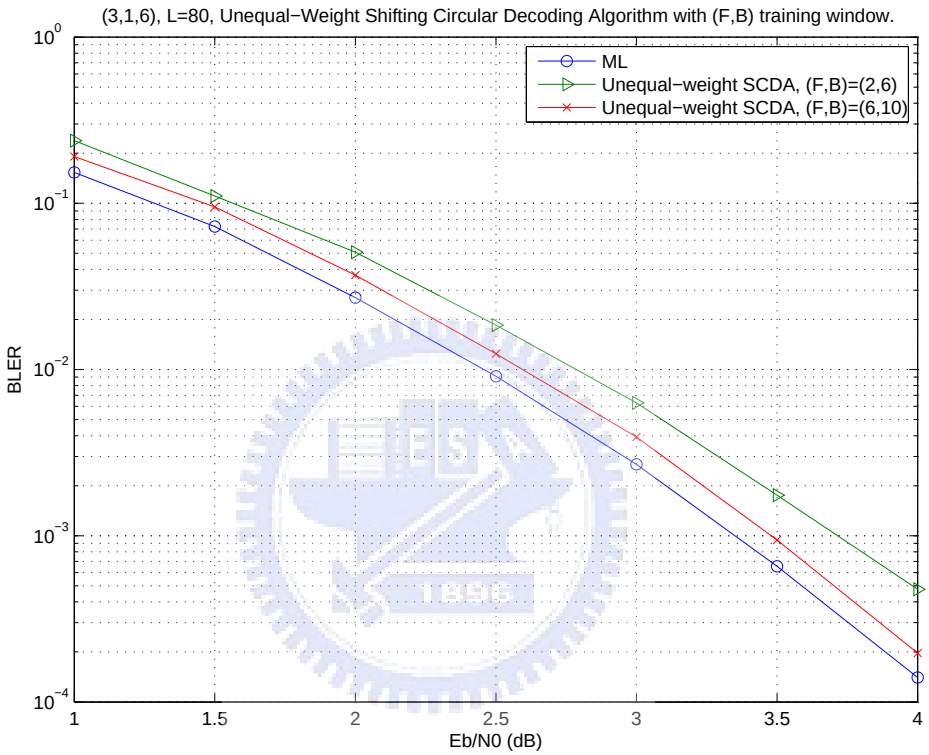


Figure 3.20: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 80$.

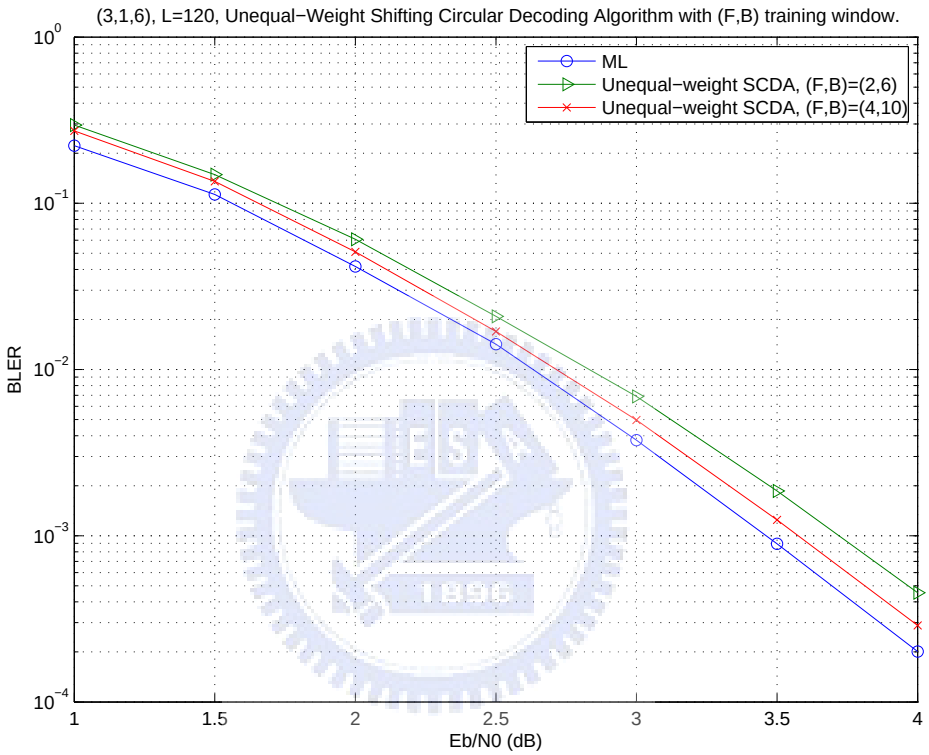


Figure 3.21: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(3, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 120$.

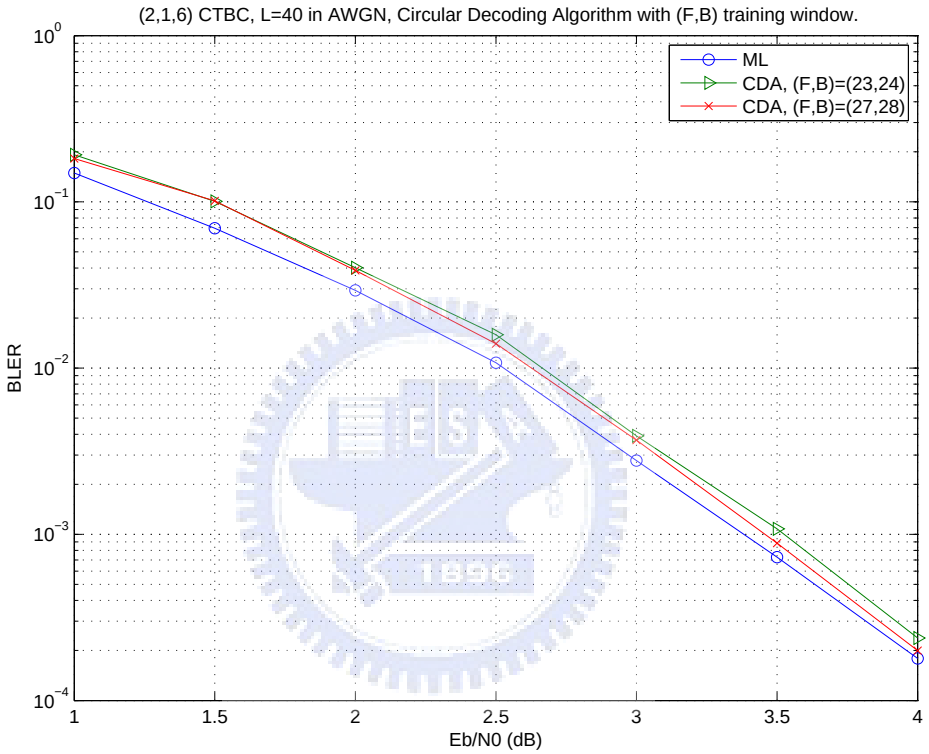


Figure 3.22: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(2, 1, 6)$ CTBC. The length of the information sequence is $L = 40$.

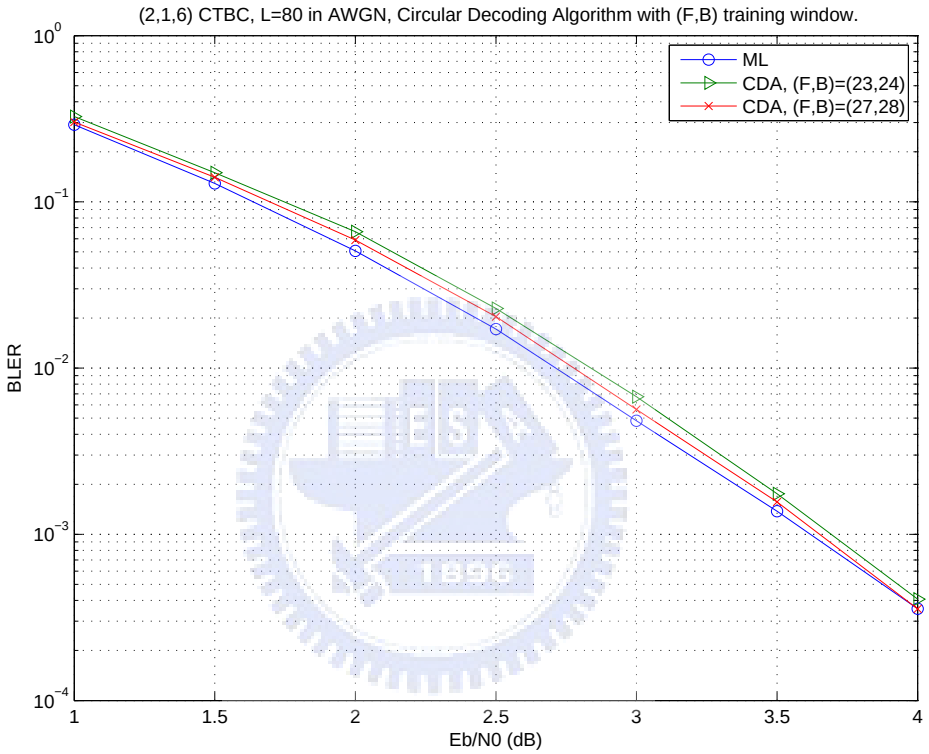


Figure 3.23: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(2, 1, 6)$ CTBC. The length of the information sequence is $L = 80$.

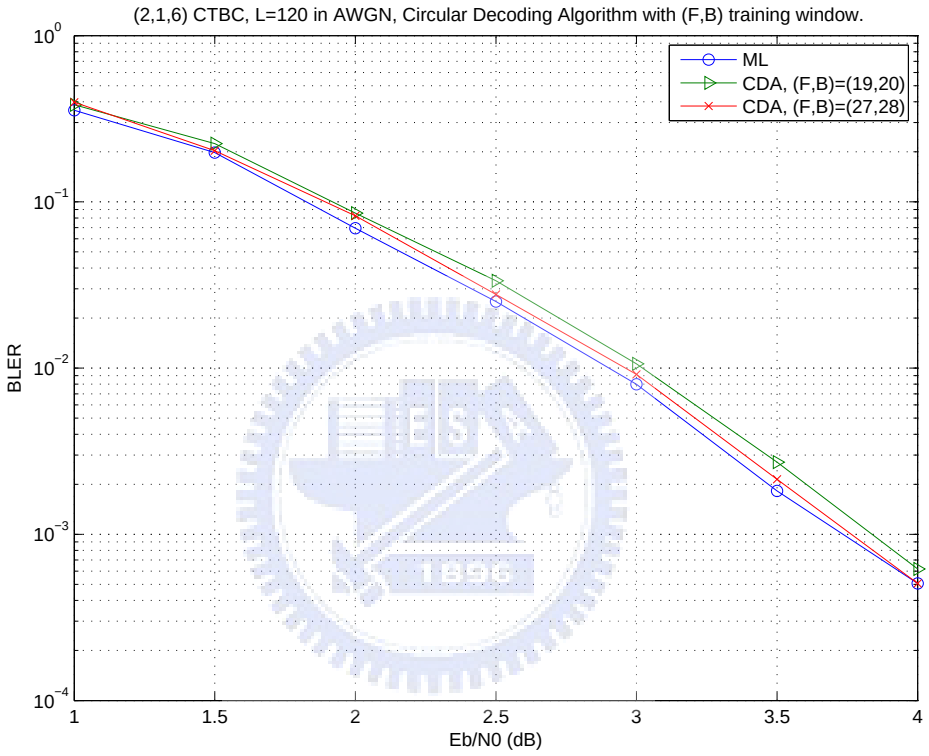


Figure 3.24: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(2, 1, 6)$ CTBC. The length of the information sequence is $L = 120$.

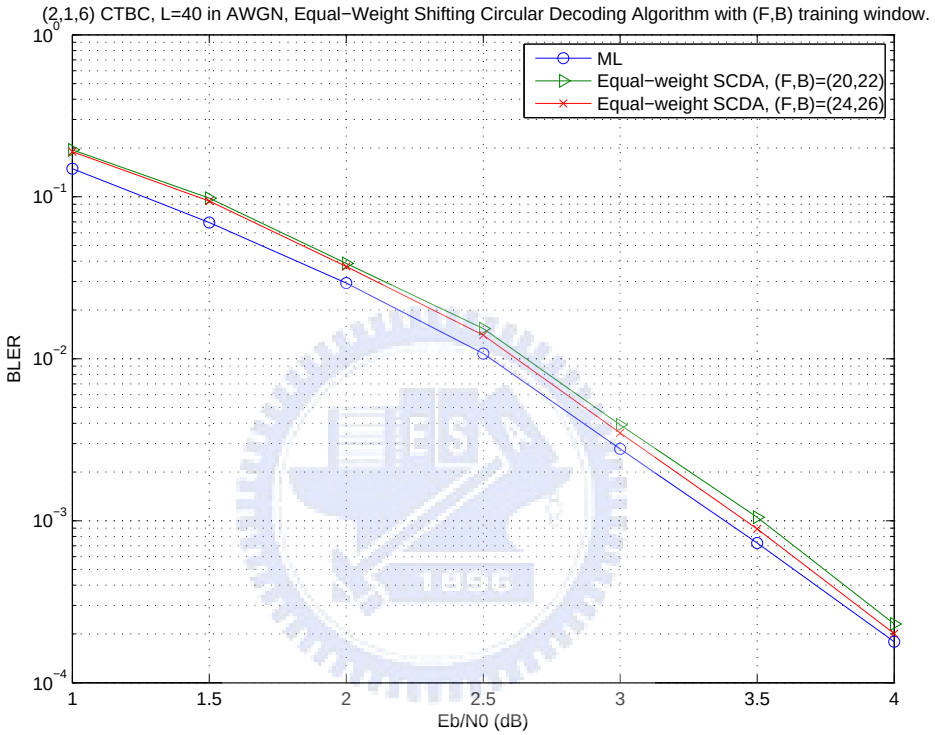


Figure 3.25: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

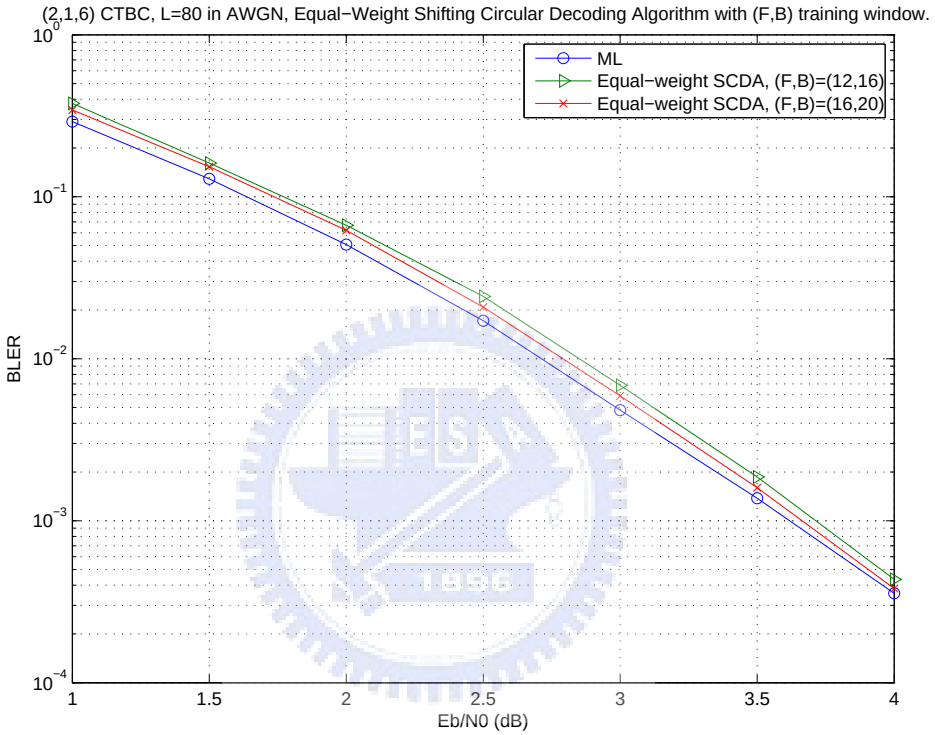


Figure 3.26: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectivley. The length of the information sequence is $L = 80$.

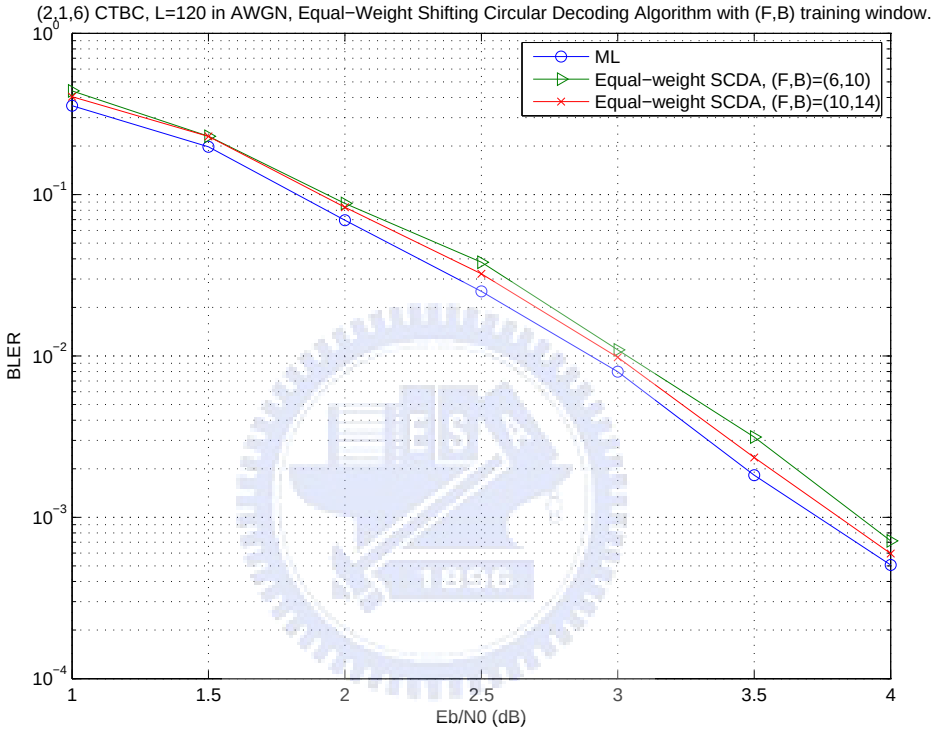


Figure 3.27: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectivley. The length of the information sequence is $L = 120$.

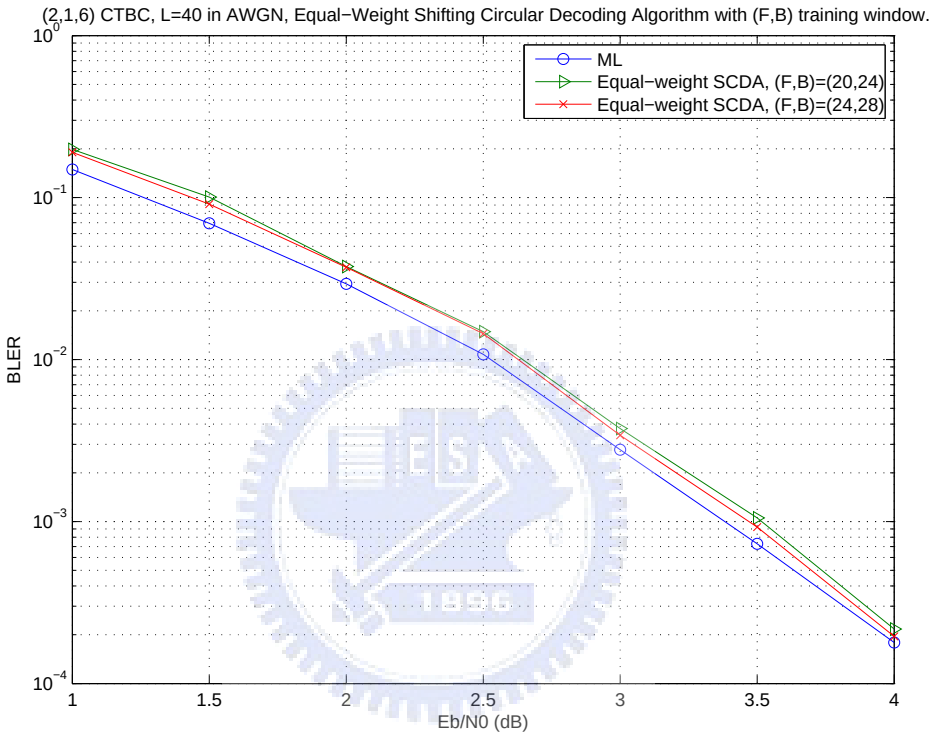


Figure 3.28: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

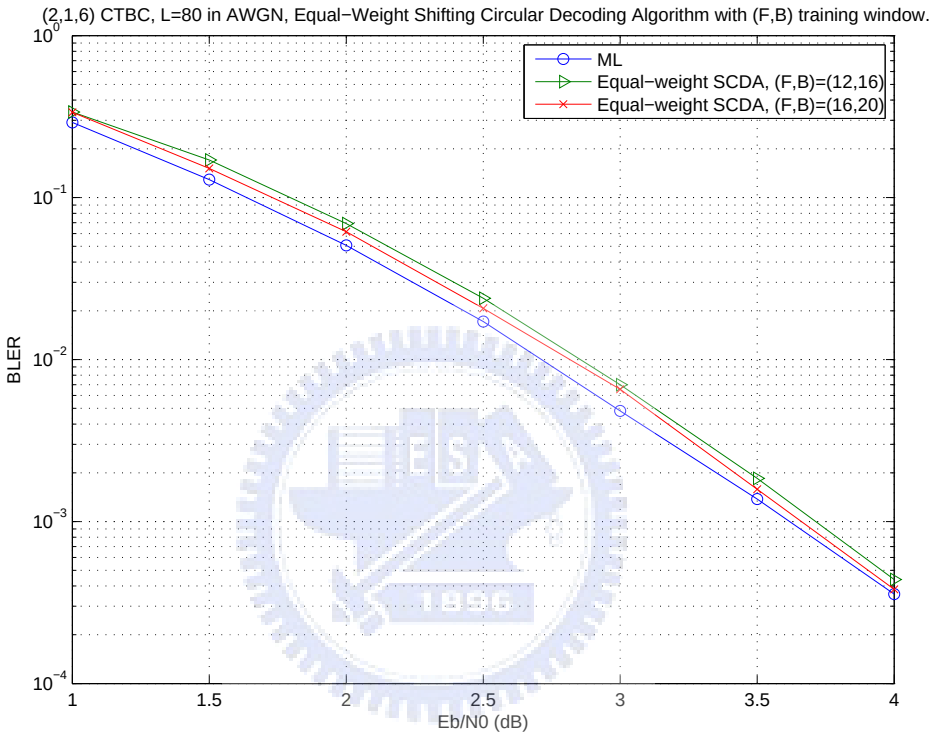


Figure 3.29: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 80$.

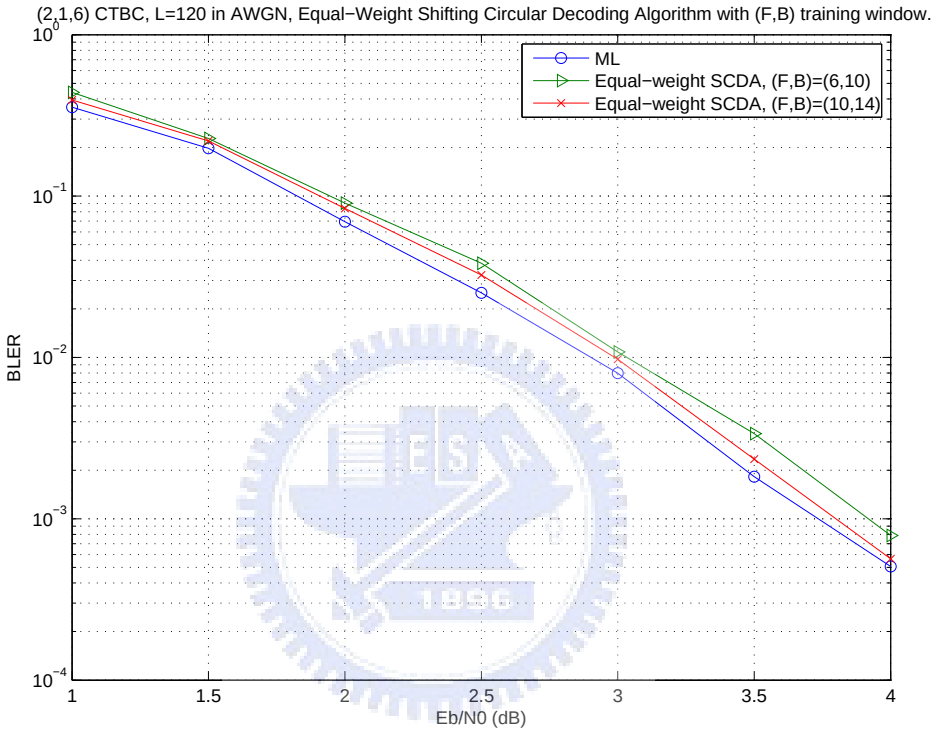


Figure 3.30: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 120$.

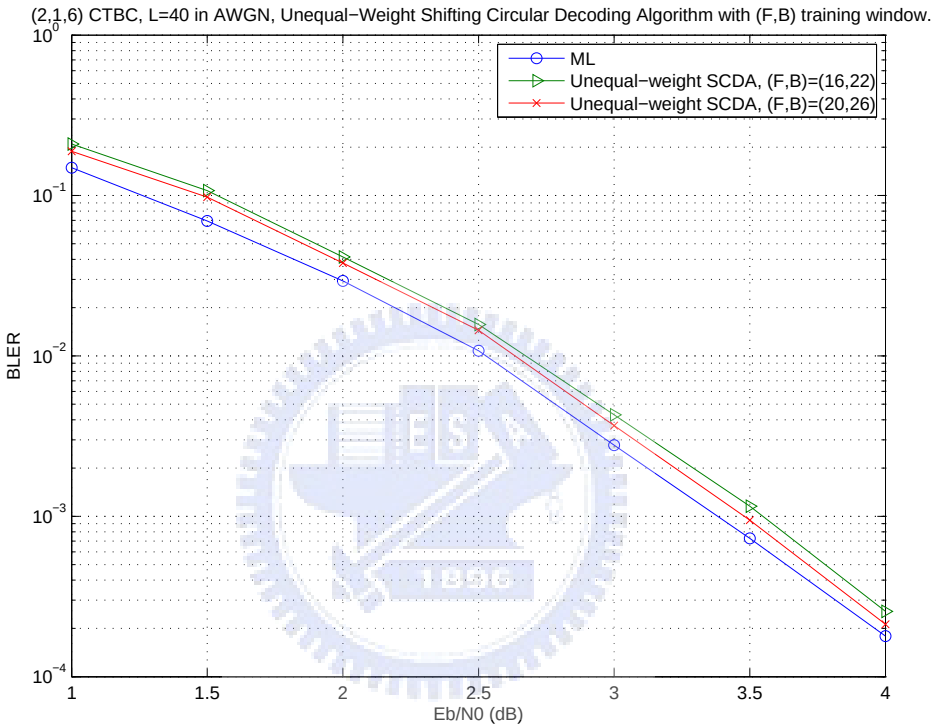


Figure 3.31: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

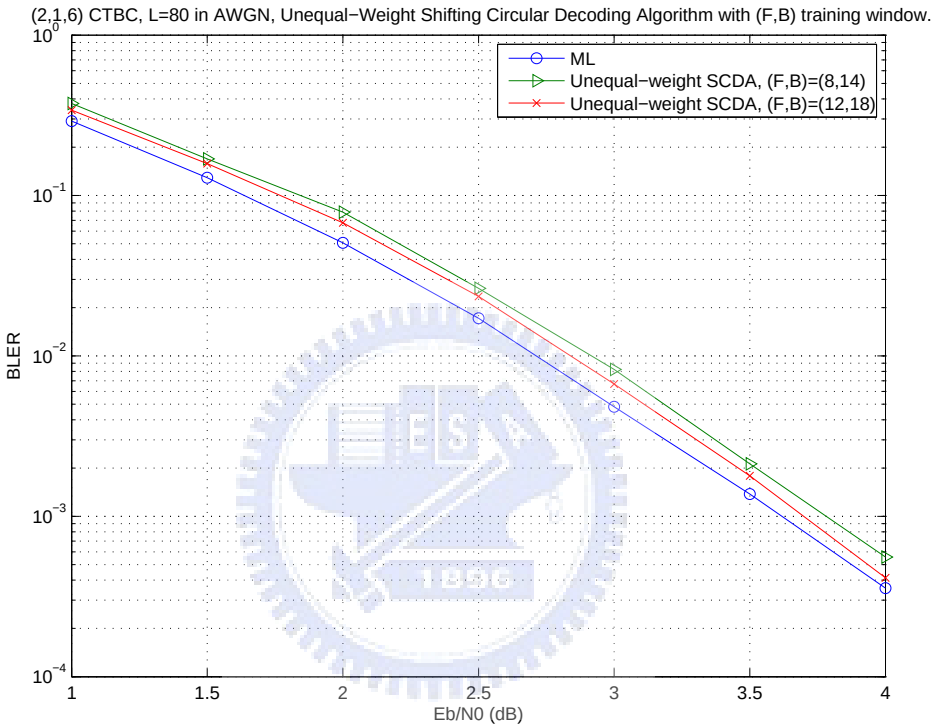


Figure 3.32: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 80$.

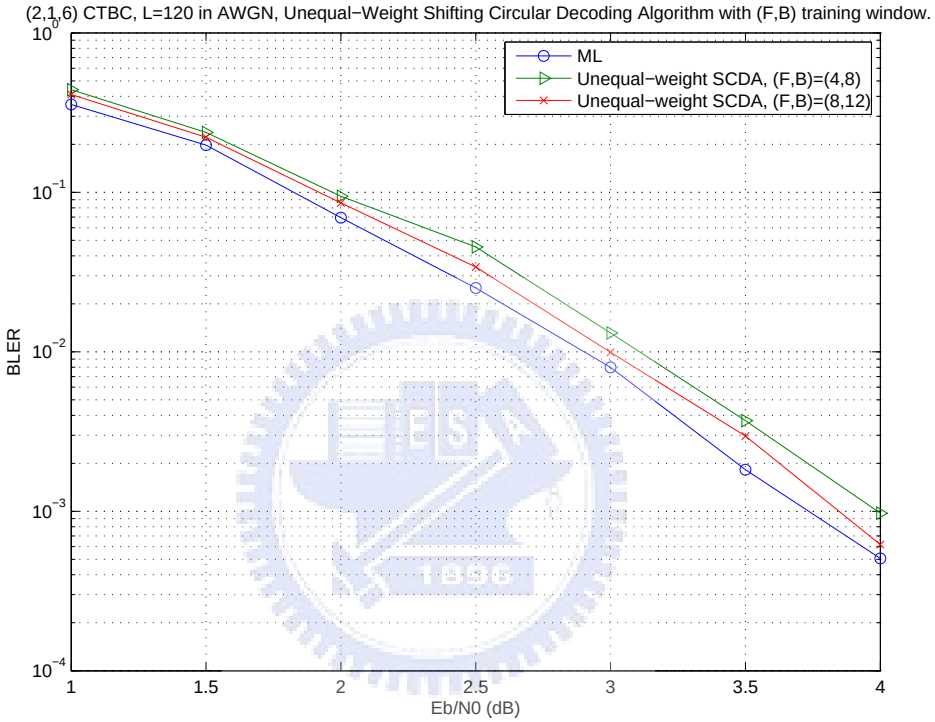


Figure 3.33: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(2, 1, 6)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 120$.

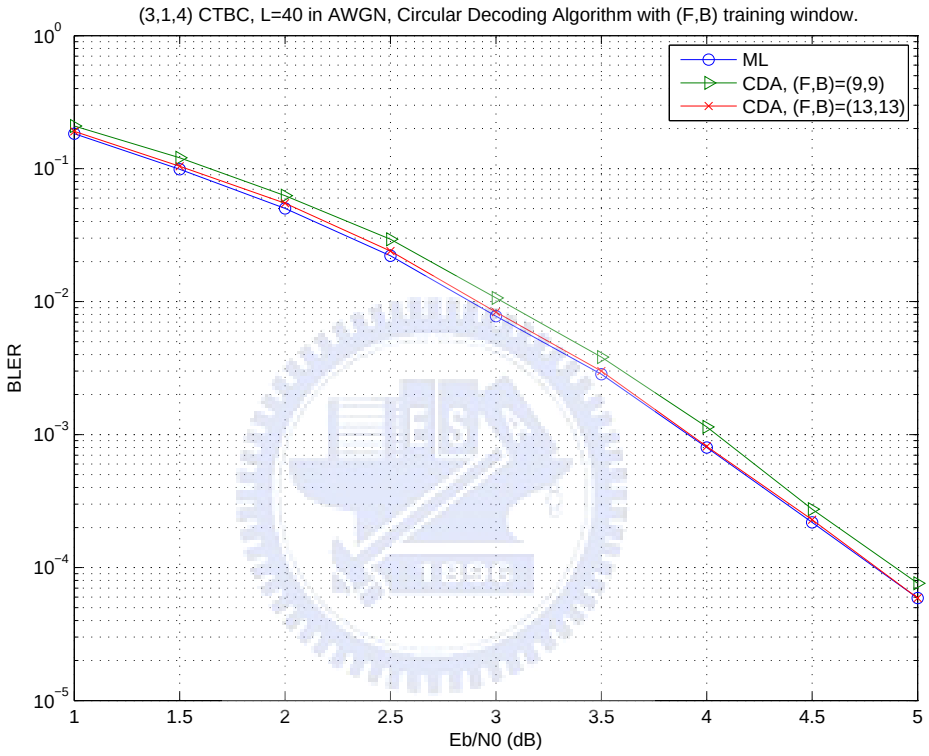


Figure 3.34: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(3, 1, 4)$ CTBC. The length of the information sequence is $L = 40$.

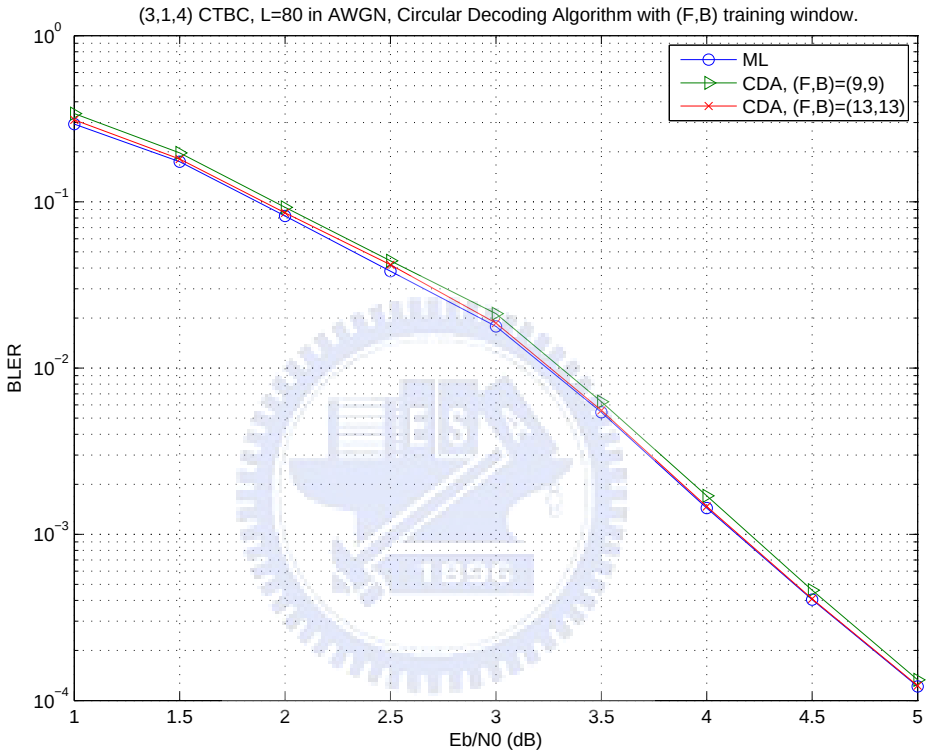


Figure 3.35: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(3, 1, 4)$ CTBC. The length of the information sequence is $L = 80$.

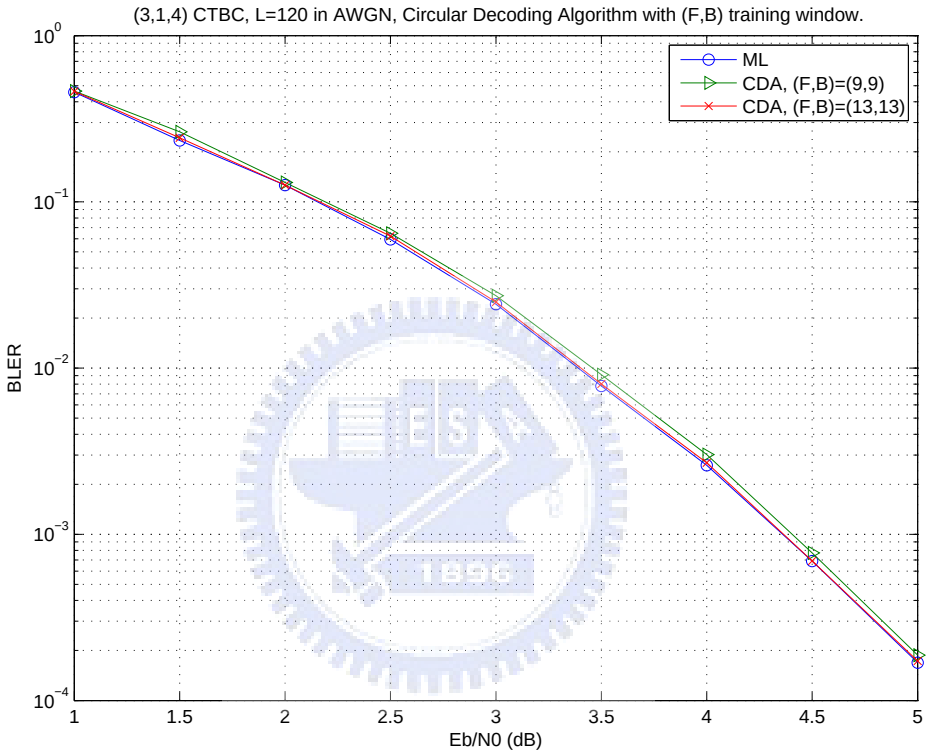


Figure 3.36: Block error rates (BLER) of the circular decoding algorithm without pre-decoding shift for $(3, 1, 4)$ CTBC. The length of the information sequence is $L = 120$.

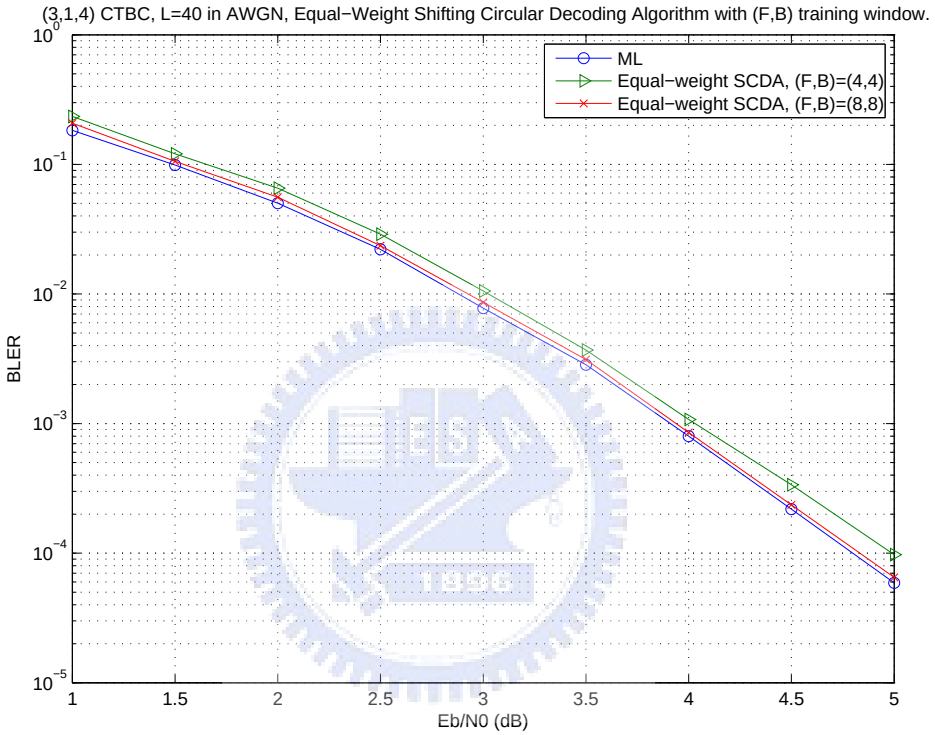


Figure 3.37: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

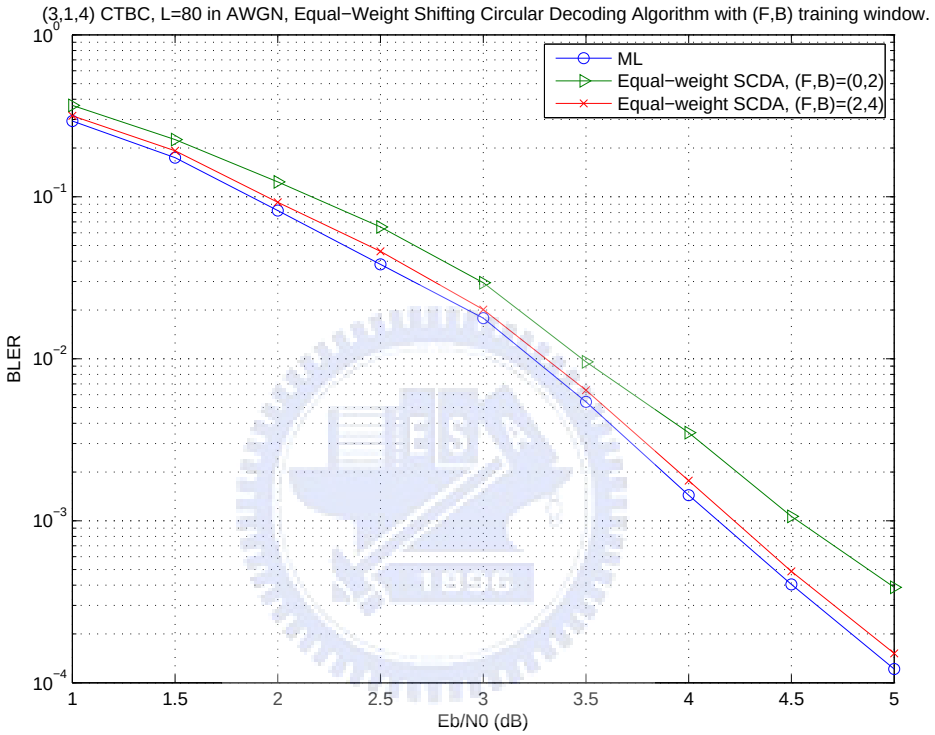


Figure 3.38: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC. F and B denote the forward and backward training window sizes, respectivley. The length of the information sequence is $L = 80$.

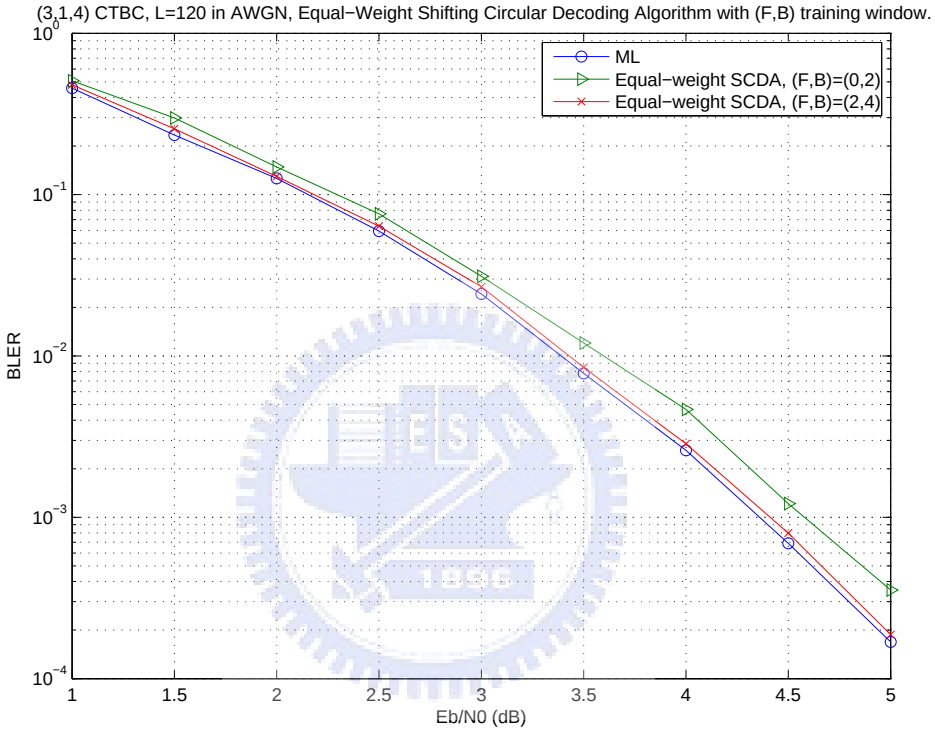


Figure 3.39: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC. F and B denote the forward and backward training window sizes, respectivley. The length of the information sequence is $L = 120$.

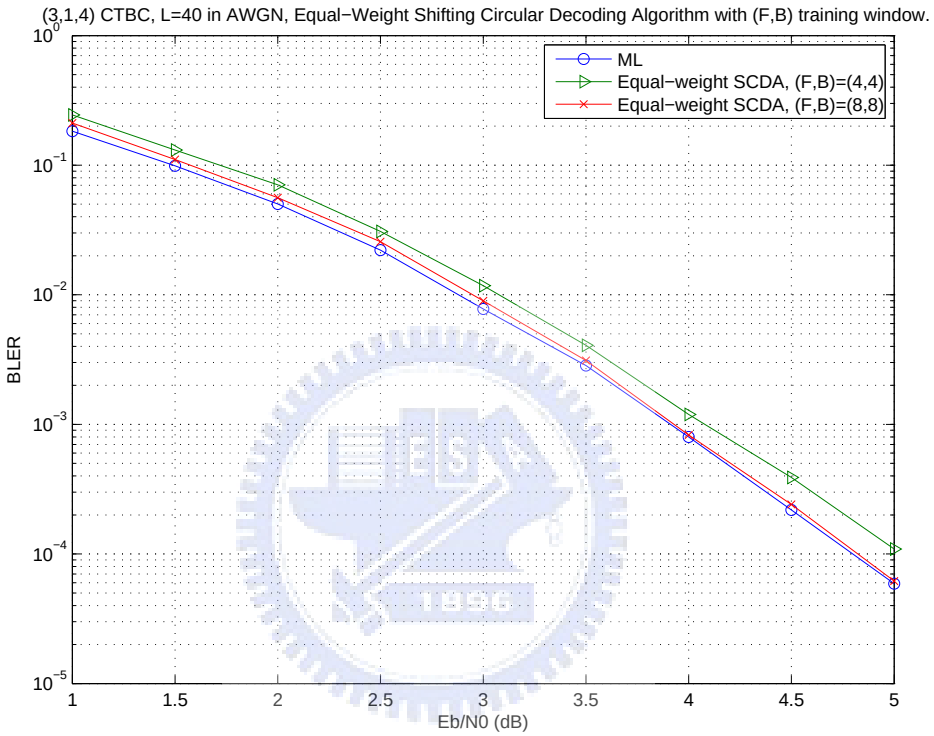


Figure 3.40: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

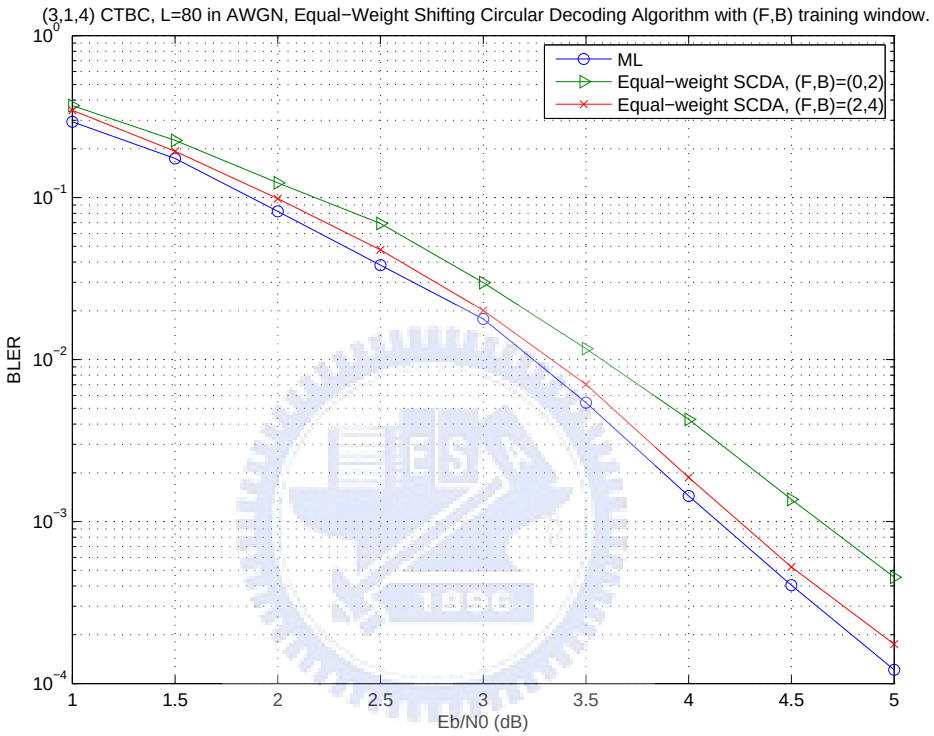


Figure 3.41: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 80$.

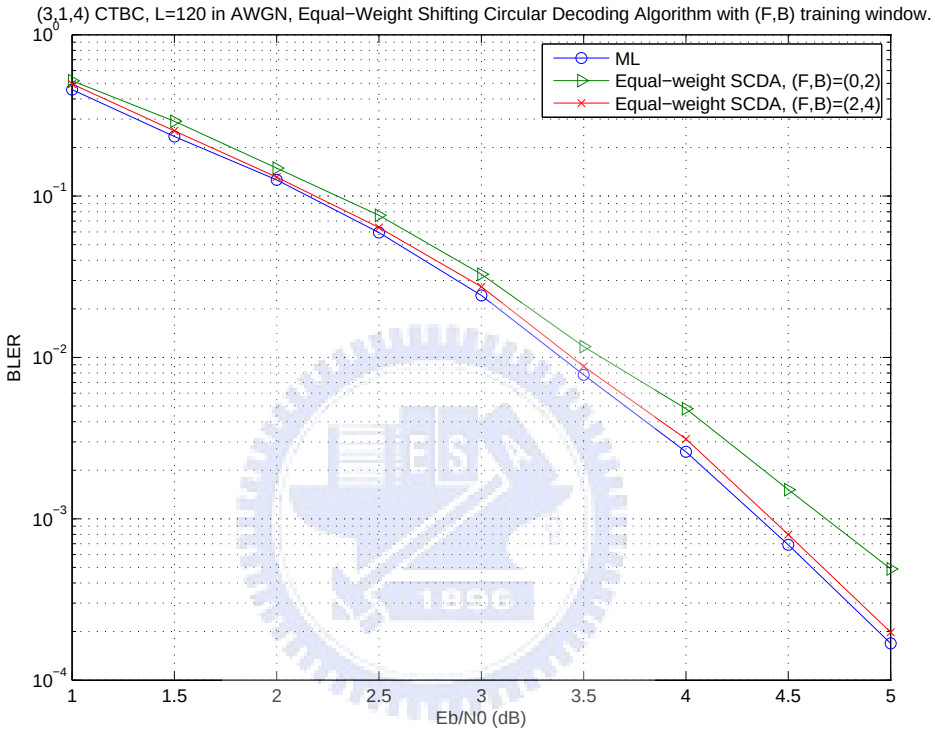


Figure 3.42: Block error rates (BLER) of the equal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC, where $(p - W/2)$ must be a multiple of W. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 120$.

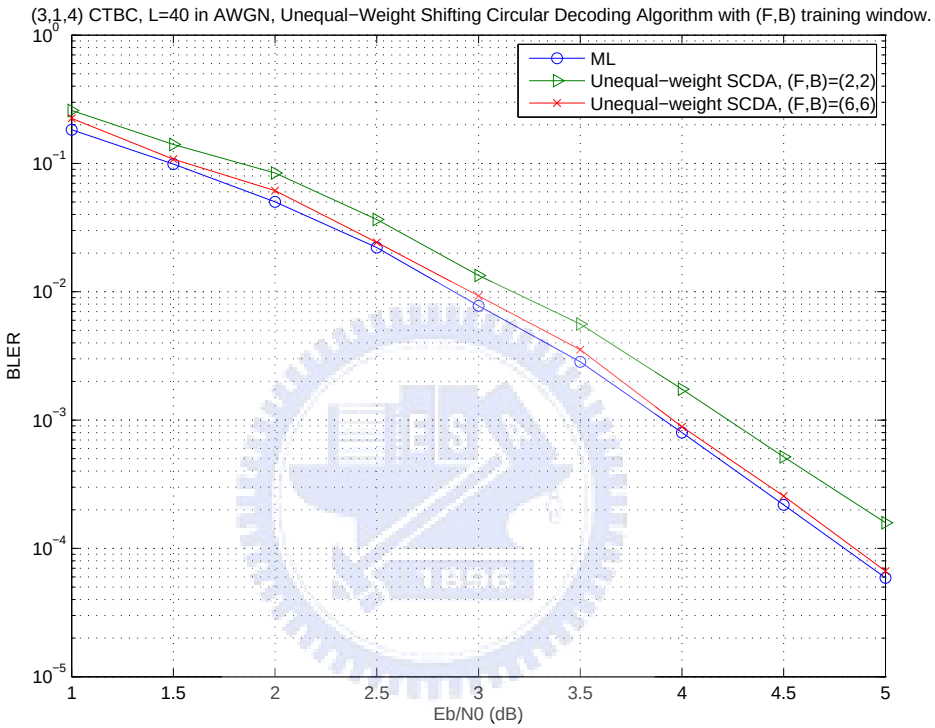


Figure 3.43: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 40$.

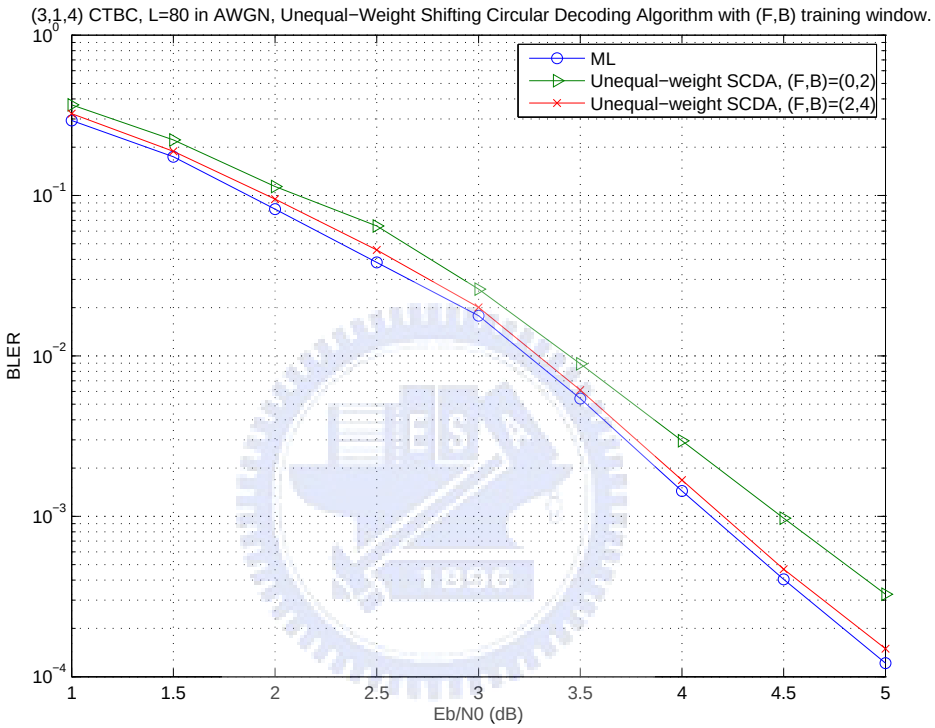


Figure 3.44: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 80$.

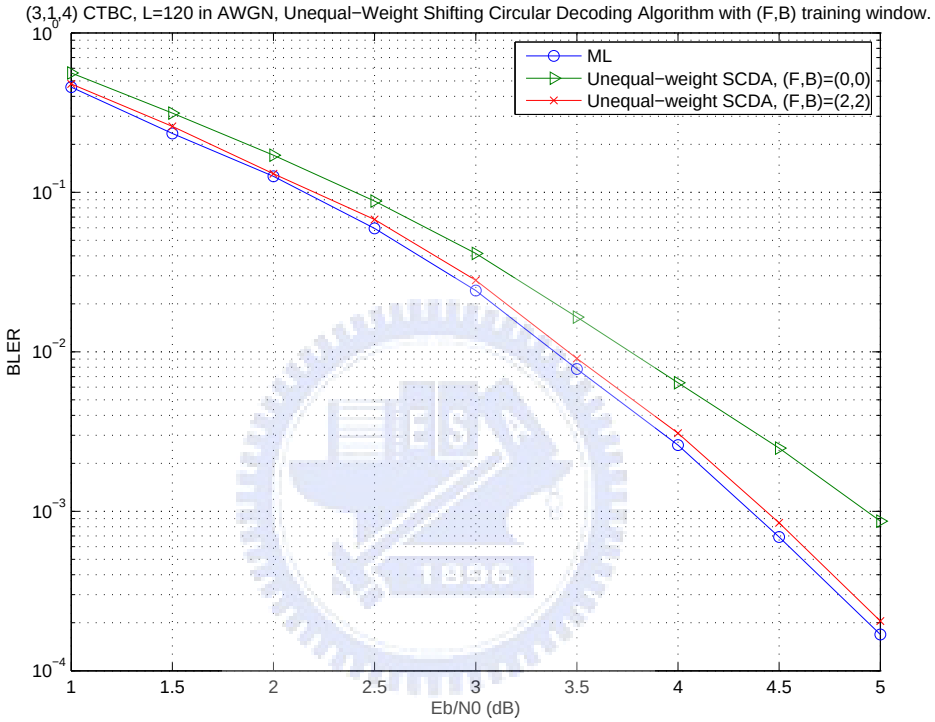


Figure 3.45: Block error rates (BLER) of the unequal-weight shifting circular decoding algorithm for $(3, 1, 4)$ CTBC. F and B denote the forward and backward training window sizes, respectively. The length of the information sequence is $L = 120$.

Chapter 4

Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm

In the previous chapter, we observed from simulations that the forward and backward training window sizes required to achieve near ML performance can be reduced by pre-decoding circular shift of the received sequence. However, these windows were determined by simulations. A natural subsequent work is whether or not we can obtain suggestive training window sizes analytically.

4.1 Error Probability for Truncated Viterbi Decoding

In [6], the authors gave an upper bound on the bit error probability due to path truncation for Viterbi decoding. This bound can then be used to determine the truncation window size such that the additional error probability due to truncation becomes negligible with respect to the ML decoding error.

Assume that the all-zero code word is transmitted, and the Viterbi decoder adopts truncation window of size T. The decoding error of the first information bit due to truncation occurs when one of the paths that diverges from the all-zero state at level 0 and never returns

to the all-zero state within T levels has the best metric among all paths ending at level T (cf. Fig. 4.1). The input-output weight enumerator function (IOWEF) for this set of paths is given by

$$\sum_{i=1}^{2^m-1} A_{0,i}^T(W, X, L) \tag{4.1}$$

where $A_{0,i}^T(W, X, L)$ is the weight enumerator function in the modified state diagram, which connects the all-zero state with ith state, and which is expurgated to include only paths of length T. Here, W is the dummy variable whose exponent represents the weight of information bits, X is the dummy variable whose exponent is the weight of code bits, and the exponent of L is the length of the branch [7, pp. 493].[1]

Using union bound technique, we can bound the truncation error probability for the first information bit on a binary symmetric channel (BSC) as

$$P_{T,1} < \sum_{i=1}^{2^m-1} A_{0,i}^T(W, X, L)\Big|_{W=1, X=2\sqrt{\varepsilon(1-\varepsilon)}, L=1}, \tag{4.2}$$

where ε is the channel crossover probability. By summing the first event error of length less than T and the truncation error for first information bit in (4.2), the overall error probability for the first bit is given by

$$P_{E,1} < \left(A_T(W, X, L) + \sum_{i=1}^{2^m-1} A_{0,i}^T(W, X, L) \right)\Bigg|_{W=1, X=2\sqrt{\varepsilon(1-\varepsilon)}, L=1}, \tag{4.3}$$

where $A_T(W, X, L)$ is the weight enumerator function for the paths that start from the all-zero state and later return to the all-zero state with length T or less.

Next, consider decoding the information bit at an arbitrary level t. Any path that diverges from the all-zero state at or before level t and never returns to the all-zero state through level $t+T$ will cause a truncation error (cf. Fig. 4.2). Hence, we can obtain a bound similar

[1] The authors in [6] use notation $G_{0,i}^T(D, L)$ in their paper, which is exactly our $A_{0,i}^T(W = 1, X = D, L)$. Similar notation conversion applies to the following derivations in this section.

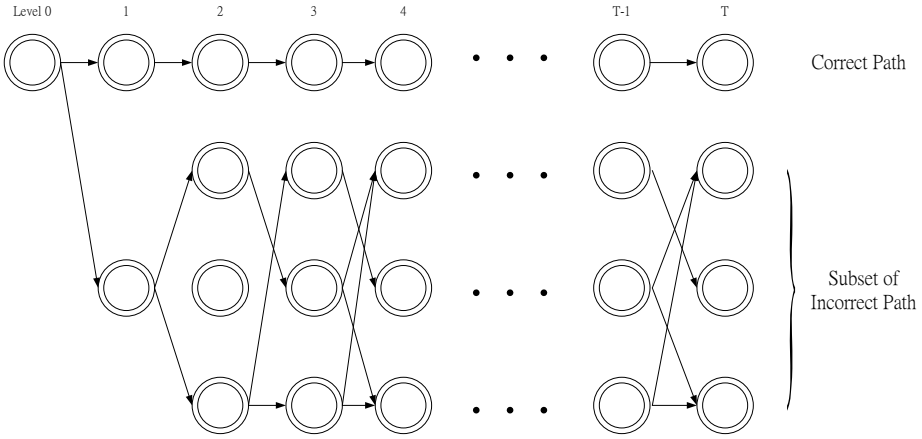


Figure 4.1: The set of paths that diverge from the correct path at level 0 but never return to the zero-state within T level

to (4.2). By extending the bound in terms of the weight enumerator function $A_{0,i}^{T,\infty}(W,X,L)$ in the modified state diagram with length T and greater, we can establish

$$P_T < \sum_{i=1}^{2^m-1} A_{0,i}^{T,\infty}(W,X,L)\Big|_{W=1,X=2\sqrt{\varepsilon(1-\varepsilon)},L=1} . \tag{4.4}$$

This concludes, as analogous to (4.3), that the overall bit error rate is bounded by

$$P_E < \left(\frac{dA(W,X,L)}{dW} + \sum_{i=1}^{2^m-1} A_{0,i}^{T,\infty}(W,X,L)\right)\Bigg|_{W=1,X=2\sqrt{\varepsilon(1-\varepsilon)},L=1} \tag{4.5}$$

where $A(W,X,L)$ is the weight enumerator function of the code.

The first term in (4.5) is the ML bit error probability, and the second term corresponds to the bit error probability bound due to truncation. Both terms are dominated by the component with the smallest power of X. For the ML term, the smallest power of X is the free distance d_{free}. For the term corresponding to truncation error, the smallest power of X is contributed by the minimum Hamming weight path of the incorrect path subset, denoted by d_T. Therefore, if T is chosen such that $d_T > d_{\text{free}}$, then the error will be dominated by the ML term.

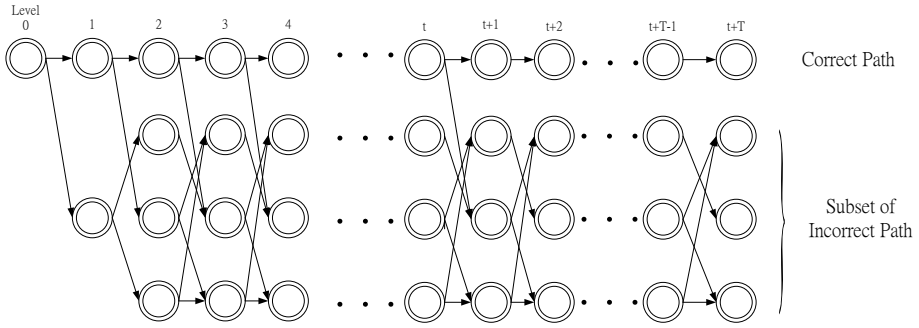


Figure 4.2: The set of paths that diverge from the correct path at level t but never return to the zero-state within $t+T$ level

4.2 Error Bound for Circular Decoding

In [11], the author uses the techniques in Section 4.1 to analyze the error probability bound for circular decoding. He actually regards the forward and backward training windows as two truncation windows.

Following similarly (4.5), we can obtain the BSC error probability bound of circular decoding as

$$P_E < \left(\frac{dA(W,X,L)}{dW} + \sum_{i=1}^{2^m-1} A_{i,0}^{F,\infty}(W,X,L) + \sum_{i=1}^{2^m-1} A_{0,i}^{B,\infty}(W,X,L) \right) \Bigg|_{W=1,X=2\sqrt{\varepsilon(1-\varepsilon)},L=1}, \tag{4.6}$$

where $A_{i,0}^{F,\infty}(W,X,L)$ denotes the weight enumerator function in the modified state diagram for the paths connecting the ith state with the all-zero state with length F or greater for the forward training window, and $A_{0,i}^{B,\infty}(W,X,L)$ denotes the weight enumerator function in the modified state diagram for the paths connecting the all-zero state with the ith state with length B or greater for the backward training window.

For the second and third terms in (4.6), denote respectively by d_F and d_B the dominant weights of their minimal Hamming weight paths of the respective incorrect path subsets for the forward and backward training windows. As a result, if $d_F > d_{\text{free}}$ and $d_B > d_{\text{free}}$, the

BSC error probability will be dominated by the ML term, i.e., the first term in (4.6).

We note in the end that the derivations from (4.2) to (4.6) also hold for the binary erasure channel (BEC) with $X = \delta$ instead of $X = 2\sqrt{\varepsilon(1-\varepsilon)}$, where δ is the erasure probability. They also hold for the additive white Gaussian noise (AWGN) channel with $X = e^{-E_s/N_0}$, where E_s/N_0 is the signal-to-noise ratio.

4.3 Error Bound for Shifting Circular Decoding in Binary Erasure Channel (BEC)

Over binary erasure channels (BEC) as depicted in Fig. 4.3, the erasure probability for the the second and third terms in (4.6) will be changed by adding the pre-decoding circular shifting. It should be pointed out that for BECs and equal prior probability, the statistics of the absolute value of reliability ϕ_j becomes

$$|\phi_j| = \begin{cases} \infty, & \text{with probability } (1-\delta); \\ 0, & \text{with probability } \delta. \end{cases}$$

Hence, with the signal mapping at both the input and output signals as

$$\begin{array}{ccc} 0 & \rightarrow & 1 \\ \text{erasure} & \rightarrow & 0 \\ 1 & \rightarrow & -1 \end{array},$$

the pre-decoding circular shifts on the receive sequence $\boldsymbol{r} \in \{-1, 0, 1\}^N$ is again performed based on the absolute values of the received components. We can then analyze the effective erasure probability δ_e as follows.

To simplify the analysis, we assume that the "unadjusted" pre-decoding information circular shift $\tilde{p} = p - W/2$ can only be a multiple of weighted average window W,[2] information

[2] Again, for shifting circular decoding algorithm, $p = \tilde{p} + W/2$, in which $W/2$ will be added afterwards. Here, we denote $\tilde{p} = \arg\max_\ell R_\ell$. Thus, what we assume in this derivation is that

$$\tilde{p} = \arg \max_{\ell = 0, W, 2W, \ldots, (\lfloor L/W \rfloor - 1)W} R_\ell.$$

See Footnote (1) in page 6.

length L for $(n, 1, m)$ CTBC is also a multiple of W, and equal weights that are all equal to unity are adopted. Define for $0 \leq i < M = L/W$,

$$U_i = |r_{inW}| + |r_{inW+1}| + \cdots + |r_{inW+nW-1}|.$$

Then, we derive that

$$\begin{aligned} \Pr[U_i = u] &= \Pr\left[|r_{inW}| + |r_{inW+1}| + \cdots + |r_{inW+nW-1}| = u\right] \\ &= \binom{nW}{u} \delta^{nW-u}(1-\delta)^u, \end{aligned}$$

where we know that $\{r_j\}_{j=0}^{N-1}$ are i.i.d., and the absolute value of r_j is distributed as

$$|r_j| = \begin{cases} 1, & \text{with probability } 1-\delta; \\ 0, & \text{with probability } \delta. \end{cases}$$

Now, the sorted $U_0, U_1, \ldots, U_{M-1}$ gives that

$$U_{(0)} \leq U_{(1)} \leq U_{(2)} \leq \cdots \leq U_{(M-1)}.$$

In such case, the pre-decoding circular shift $\tilde{p} = Wi$, if $U_{(M-1)} = U_i$, and the unadjusted inputs for the circular decoding algorithm is $r_{\tilde{p}n}, r_{\tilde{p}n+1}, \ldots, r_{N-1}, r_0, r_1, \ldots, r_{\tilde{p}n-1}$.[3] It remains to find the distribution of the unadjusted shifted input sequence.

For $(a_0, a_1, \ldots, a_{N-1}) \in \{-1, 0, 1\}^N$ with $\sum_{\ell=0}^{nW-1} |a_\ell| < \max_{1 \leq i \leq M-1} \sum_{\ell=0}^{nW-1} |a_{inW+\ell}|$,

$$\Pr\left[r_{\tilde{p}n} = a_0 \wedge r_{\tilde{p}n+1} = a_1 \wedge \cdots \wedge r_{\tilde{p}n-1} = a_{N-1}\right] = 0.$$

[3]There is potentially a notational flaw here. For example, $r_{\tilde{p}n-1}$ will equal r_{-1}, which happens when $\tilde{p} = 0$. For notational convenience, r_{-1} is well understood as r_{N-1} since $r_{-1} = r_{(-1) \bmod N} = r_{N-1}$. Similar rule will be applied to, say, $r_{-2}, r_{-3}, \ldots$ whenever they are encountered.

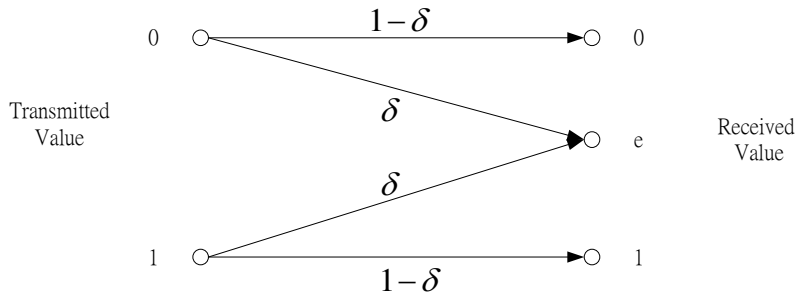


Figure 4.3: Binary erasure channel (BEC) model

For $(a_0, a_1, \ldots, a_{N-1}) \in \{-1, 0, 1\}^N$ with $\sum_{\ell=0}^{nW-1} |a_\ell| > \max_{1 \le i \le M-1} \sum_{\ell=0}^{nW-1} |a_{inW+\ell}|$,[4]

$$\begin{aligned}
&\Pr\left[r_{\tilde{p}n} = a_0 \wedge r_{\tilde{p}n+1} = a_1 \wedge \cdots \wedge r_{\tilde{p}n-1} = a_{N-1}\right] \\
&= \sum_{i=0}^{M-1} \Pr\left[r_{inW} = a_0 \wedge r_{inW+1} = a_1 \wedge \cdots \wedge r_{inW-1} = a_{N-1}\right] \qquad (4.7) \\
&= M \prod_{\ell=0}^{N-1} \Pr\left[r_\ell = a_\ell\right], \qquad (4.8)
\end{aligned}$$

where (4.8) is valid since $\Pr\left[r_{inW} = a_0 \wedge r_{inW+1} = a_1 \wedge \cdots \wedge r_{inW-1} = a_{N-1}\right]$ is the same for every i as $\{r_\ell\}_{\ell=0}^{N-1}$ is i.i.d. For $(a_0, a_1, \ldots, a_{N-1}) \in \{-1, 0, 1\}^N$ with $\sum_{\ell=0}^{nW-1} |a_\ell| =$

[4]Note that we give below a simple example to facilitate the understanding of (4.7). Since $\tilde{p}$ is a random variable, the event of $[r_{\tilde{p}n} = a_0 \wedge r_{\tilde{p}n+1} = a_1 \wedge \cdots \wedge r_{\tilde{p}n-1} = a_{N-1}]$ is seemingly involved in its calculation. This involvement however can be clarified by the following example.

Now, suppose $W = 2$, $L = 6$ and $n = 2$. Then, $M = 3$, and $\tilde{p}$ must be even, and we receive a sequence of length 12. Given that $(a_0, a_1, \ldots, a_{11}) = (1, 1, -1, -1, 0, 0, 1, 1, -1, 0, -1, 1)$, the event that $(r_{2\tilde{p}}, r_{2\tilde{p}+1}, \ldots, r_{2\tilde{p}-1}) = (1, 1, -1, -1, 0, 0, 1, 1, -1, 0, -1, 1)$ occurs only when one of the three cases below occur, which are

$$\begin{cases}
(r_0, r_1, \ldots, r_{11}) = (\underline{1, 1, -1, -1}, 0, 0, 1, 1, -1, 0, -1, 1) \\
(r_0, r_1, \ldots, r_{11}) = (-1, 0, -1, 1, \underline{1, 1, -1, -1}, 0, 0, 1, 1) \\
(r_0, r_1, \ldots, r_{11}) = (0, 0, 1, 1, -1, 0, -1, 1, \underline{1, 1, -1, -1})
\end{cases}$$

This explains why (4.7) holds.

$\max_{1\le i\le M-1}\sum_{\ell=0}^{nW-1}|a_{inW+\ell}|$,[5]

$$\begin{aligned}
&\Pr\left[r_{\tilde{p}n}=a_0\wedge r_{\tilde{p}n+1}=a_1\wedge\cdots\wedge r_{\tilde{p}n-1}=a_{N-1}\right]\\
&= \frac{1}{\sum_{k=0}^{M-1}\mathbf{1}\left\{\sum_{\ell=0}^{nW-1}|a_{knW+\ell}|=\sum_{\ell=0}^{nW-1}|a_\ell|\right\}}\\
&\quad\times\sum_{i=0}^{M-1}\Pr\left[r_{inW}=a_0\wedge r_{inW+1}=a_1\wedge\cdots\wedge r_{inW-1}=a_{N-1}\right]\\
&= \frac{M}{\sum_{k=0}^{M-1}\mathbf{1}\left\{\sum_{\ell=0}^{nW-1}|a_{knW+\ell}|=\sum_{\ell=0}^{nW-1}|a_\ell|\right\}}\prod_{\ell=0}^{N-1}\Pr\left[r_\ell=a_\ell\right],
\end{aligned}$$

where $\mathbf{1}\{\cdot\}$ is the set indicator function. The above derivation concludes:

$$\begin{aligned}
&\Pr\left[r_{\tilde{p}n}=a_0\wedge r_{\tilde{p}n+1}=a_1\wedge\cdots\wedge r_{\tilde{p}n-1}=a_{N-1}\right]\\
&= M\frac{\mathbf{1}\left\{\max\limits_{1\le i\le M-1}\sum\limits_{\ell=0}^{nW-1}|a_{inW+\ell}|\le\sum\limits_{\ell=0}^{nW-1}|a_\ell|\right\}}{\sum\limits_{k=0}^{M-1}\mathbf{1}\left\{\sum\limits_{\ell=0}^{nW-1}|a_{knW+\ell}|=\sum\limits_{\ell=0}^{nW-1}|a_\ell|\right\}}\prod_{\ell=0}^{N-1}\Pr\left[r_\ell=a_\ell\right]\\
&= M\frac{\mathbf{1}\left\{\max\limits_{1\le i\le M-1}\sum\limits_{\ell=0}^{nW-1}|a_{inW+\ell}|\le\sum\limits_{\ell=0}^{nW-1}|a_\ell|\right\}}{\sum\limits_{k=0}^{M-1}\mathbf{1}\left\{\sum\limits_{\ell=0}^{nW-1}|a_{knW+\ell}|=\sum\limits_{\ell=0}^{nW-1}|a_\ell|\right\}}\left(\frac{1-\delta}{2}\right)^{\sum_{\ell=0}^{N-1}|a_\ell|}\delta^{N-\sum_{\ell=0}^{N-1}|a_\ell|}. \qquad (4.9)
\end{aligned}$$

It is clear from (4.9) that the probability only depends on $\sum_{\ell=0}^{nW-1}|a_{inW+\ell}|$ for $0\le i\le M-1$.

[5]In case $\sum_{\ell=0}^{nW-1}|a_\ell| = \max_{1\le i\le M-1}\sum_{\ell=0}^{nW-1}|a_{inW+\ell}|$ such as $(a_0,a_1,\ldots,a_{11}) = (1,1,-1,-1,0,0,1,1,-1,1,-1,1)$ as continuing from the example in Footnote 4, this event occurs when one of the three cases below occur, i.e.,

$$\begin{cases}
(r_0,r_1,\ldots,r_{11})=(\underline{1,1,-1,-1},0,0,1,1,-1,1,-1,1)\\
(r_0,r_1,\ldots,r_{11})=(-1,1,-1,1,\underline{1,1,-1,-1},0,0,1,1)\\
(r_0,r_1,\ldots,r_{11})=(0,0,1,1,-1,1,-1,1,\underline{1,1,-1,-1}).
\end{cases}$$

However, since $|a_0|+|a_1|+|a_2|+|a_3| = |a_8|+|a_9|+|a_{10}|+|a_{11}|$, and since we will randomly pick one of those R_ℓ that achieve maximum, the probability should be cut in half, i.e., to multiply

$$\frac{1}{\sum_{k=0}^{2}\mathbf{1}\left\{\sum_{\ell=0}^{3}|a_{4k+\ell}|=\sum_{\ell=0}^{3}|a_\ell|\right\}}=\frac{1}{\sum_{k=0}^{2}\mathbf{1}\left\{\sum_{\ell=0}^{3}|a_{4k+\ell}|=4\right\}}=\frac{1}{2}.$$

As a result,

$$\begin{aligned}
&\Pr\left[r_{\tilde{p}n} = a_0 \wedge r_{\tilde{p}n+1} = a_1 \wedge \cdots \wedge r_{\tilde{p}n+nW-1} = a_{nW-1}\right] \\
&= \sum_{(a_{nW},\ldots,a_{N-1})\in\{-1,0,1\}^{N-nW}} \Pr\left[r_{\tilde{p}n} = a_0 \wedge r_{\tilde{p}n+1} = a_1 \wedge \cdots \wedge r_{\tilde{p}n-1} = a_{N-1}\right] \\
&= \sum_{(a_{nW},\ldots,a_{N-1})\in\{-1,0,1\}^{N-nW}} M \frac{\mathbf{1}\left\{\max\limits_{1\le i\le M-1} b_i \le b_0\right\}}{\sum\limits_{k=0}^{M-1} \mathbf{1}\left\{b_k = b_0\right\}} \left(\frac{1-\delta}{2}\right)^{\sum_{i'=0}^{M-1} b_{i'}} \delta^{N-\sum_{i'=0}^{M-1} b_{i'}},
\end{aligned}$$

where for notational convenient, we denote

$$b_i = \sum_{\ell=0}^{nW-1} |a_{inW+\ell}|.$$

Continue the derivation with $q = (1-\delta)/\delta$:

$$
\begin{aligned}
&\Pr\left[r_{\tilde{p}n} = a_0 \wedge r_{\tilde{p}n+1} = a_1 \wedge \cdots \wedge r_{\tilde{p}n+nW-1} = a_{nW-1}\right] \\
&= M\delta^N (q/2)^{b_0} \sum_{(b_1,\ldots,b_{M-1})\in\{0,\ldots,b_0\}^{M-1}} \binom{nW}{b_1} 2^{b_1} \binom{nW}{b_2} 2^{b_2} \cdots \binom{nW}{b_{M-1}} 2^{b_{M-1}} \frac{(q/2)^{\sum_{i'=1}^{M-1} b_{i'}}}{1 + \sum_{k=1}^{M-1} \mathbf{1}\left\{b_k = b_0\right\}} \\
&= M\delta^N (q/2)^{b_0} \sum_{(b_1,\ldots,b_{M-1})\in\{0,\ldots,b_0\}^{M-1}} \binom{nW}{b_1} \binom{nW}{b_2} \cdots \binom{nW}{b_{M-1}} \frac{q^{\sum_{i'=1}^{M-1} b_{i'}}}{1 + \sum_{k=1}^{M-1} \mathbf{1}\left\{b_k = b_0\right\}} \\
&= \begin{cases} \delta^N, & b_0 = 0 \\ M\delta^N (q/2)^{b_0} \sum_{k=0}^{M-1} \frac{1}{1+k} \binom{M-1}{k} \binom{nW}{b_0}^k & \\ \quad \sum_{(b_{k+1},\ldots,b_{M-1})\in\{0,\ldots,b_0-1\}^{M-1-k}} \binom{nW}{b_{k+1}} \cdots \binom{nW}{b_{M-1}} q^{kb_0 + \sum_{i'=k+1}^{M-1} b_{i'}}, & b_0 > 0 \end{cases} \\
&= \begin{cases} \delta^N, & b_0 = 0 \\ M\delta^N (q/2)^{b_0} \sum_{k=0}^{M-1} \frac{1}{1+k} \binom{M-1}{k} \binom{nW}{b_0}^k q^{kb_0} \prod_{i'=k+1}^{M-1} \left(\sum_{b_{i'}=0}^{b_0-1} \binom{nW}{b_{i'}} q^{b_{i'}} \right), & b_0 > 0 \end{cases} \\
&= \begin{cases} \delta^N, & b_0 = 0 \\ M\delta^N (q/2)^{b_0} \sum_{k=0}^{M-1} \frac{1}{1+k} \binom{M-1}{k} \left(\binom{nW}{b_0} q^{b_0} \right)^k \left(\sum_{b_{i'}=0}^{b_0-1} \binom{nW}{b_{i'}} q^{b_{i'}} \right)^{M-1-k}, & b_0 > 0 \end{cases} \\
&= \begin{cases} \delta^N, & b_0 = 0 \\ M\delta^N (q/2)^{b_0} \dfrac{\left(\sum_{b_{i'}=0}^{b_0} \binom{nW}{b_{i'}} q^{b_{i'}} \right)^M - \left(\sum_{b_{i'}=0}^{b_0-1} \binom{nW}{b_{i'}} q^{b_{i'}} \right)^M}{M \binom{nW}{b_0} q^{b_0}}, & b_0 > 0 \end{cases} \\
&= \frac{\delta^N}{2^{b_0} \binom{nW}{b_0}} \left[\left(\sum_{b_{i'}=0}^{b_0} \binom{nW}{b_{i'}} q^{b_{i'}} \right)^M - \left(\sum_{b_{i'}=0}^{b_0-1} \binom{nW}{b_{i'}} q^{b_{i'}} \right)^M \right],
\end{aligned}
$$

where in the last equation, we assume for convenience that $\sum_{b_{i'}=0}^{b_0-1} \binom{nW}{b_{i'}} q^{b_{i'}} = 0$ when $b_0 = 0$.

This leads to that

$$\begin{aligned}
\delta_e &= \Pr\left[r_{\tilde{p}n} = 0\right] \\
&= \sum_{(a_1,\ldots,a_{nW-1})\in\{-1,0,1\}^{nW-1}} \Pr\left[r_{\tilde{p}n} = 0 \wedge r_{\tilde{p}n+1} = a_1 \wedge \cdots \wedge r_{\tilde{p}n+nW-1} = a_{nW-1}\right] \\
&= \sum_{(a_1,\ldots,a_{nW-1})\in\{-1,0,1\}^{nW-1}} \frac{\delta^N}{2^{b_0}\binom{nW}{b_0}} \left[\left(\sum_{b_{i'}=0}^{b_0}\binom{nW}{b_{i'}}q^{b_{i'}}\right)^M - \left(\sum_{b_{i'}=0}^{b_0-1}\binom{nW}{b_{i'}}q^{b_{i'}}\right)^M\right] \\
&= \sum_{k=0}^{nW-1}\binom{nW-1}{k}2^k\frac{\delta^N}{2^k\binom{nW}{k}} \left[\left(\sum_{b_{i'}=0}^{k}\binom{nW}{b_{i'}}q^{b_{i'}}\right)^M - \left(\sum_{b_{i'}=0}^{k-1}\binom{nW}{b_{i'}}q^{b_{i'}}\right)^M\right] \\
&= \delta^N \sum_{k=0}^{nW-1}\frac{(nW-k)}{nW} \left[\left(\sum_{b_{i'}=0}^{k}\binom{nW}{b_{i'}}q^{b_{i'}}\right)^M - \left(\sum_{b_{i'}=0}^{k-1}\binom{nW}{b_{i'}}q^{b_{i'}}\right)^M\right].
\end{aligned}$$

This gives the effective marginal erasure probability δ_e based on pre-decoding circular shift we propose. Through several empirical examinations, we observe that taking this new marginal erasure probability δ_e into (4.6) also provides an upper error bound on P_E for the shifting circular decoding algorithm. We however fail to substantiate this observation analytical, and will defer it as our future work.

Based on our "conjectured" observation, we will try to determine (F, B) "analytically". Specifically, considering only the terms with the lowest exponent respectively for the three terms in (4.6), we first solve

$$\delta^{d_{\text{free}}} = \delta_e^d.$$

where we use $\delta^{d_{\text{free}}}$ to approximate the ML error. We then choose F and B such that d_F and d_B equal $\lceil d \rceil$. In the next section, we will compare F and B derived in this section with those obtained by simulations.

Table 4.1: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 2)$ CTBC. The weighted average window is $W = 16$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(9, 9)$	$(9, 9)$	$(9, 9)$
Derived (F, B) of equal-weight SCDA	$(6, 6)$	$(5, 5)$	$(5, 5)$
Simulated (F, B) of equal-weight SCDA	$(2, 4)$	$(2, 2)$	$(2, 2)$

Table 4.2: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 4)$ CTBC. The weighted average window is $W = 24$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(13, 13)$	$(13, 13)$	$(15, 15)$
Derived (F, B) of equal-weight SCDA	$(10, 10)$	$(9, 9)$	$(9, 9)$
Simulated (F, B) of equal-weight SCDA	$(8, 8)$	$(4, 6)$	$(4, 4)$

4.4 Forward and Backward Training Windows in BEC

In this section, we investigate the forward and backward window sizes of the circular decoding algorithm (CDA) and equal-weight shifting circular decoding algorithm (SCDA), requiring for near maximum-likelihood performance in BEC. The weighted average window is taken to be twice of the d_{free}, i.e., $W = 2d_{\text{free}}$. Three CTBC codes are considered, which are the $(3, 1, 2)$, $(3, 1, 4)$ and $(3, 1, 6)$ codes respectively with generator polynomials (5 7 7), (52 66 76) and (554 744 724) (in octal). The results are summarized in Tables 4.1, 4.2 and 4.3. They show that the (F, B) of the CDA required for near ML performance in BEC match those reported in [11]. Also, the derived (F, B) pairs are a little larger than the simulated ones. The performance curves can be found in Figs 4.4–4.21.

Table 4.3: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 6)$ CTBC. The weighted average window is $W = 30$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(19, 20)$	$(19, 20)$	$(19, 21)$
Derived (F, B) of equal-weight SCDA	$(16, 19)$	$(15, 18)$	$(12, 15)$
Simulated (F, B) of equal-weight SCDA	$(14, 14)$	$(10, 12)$	$(8, 10)$

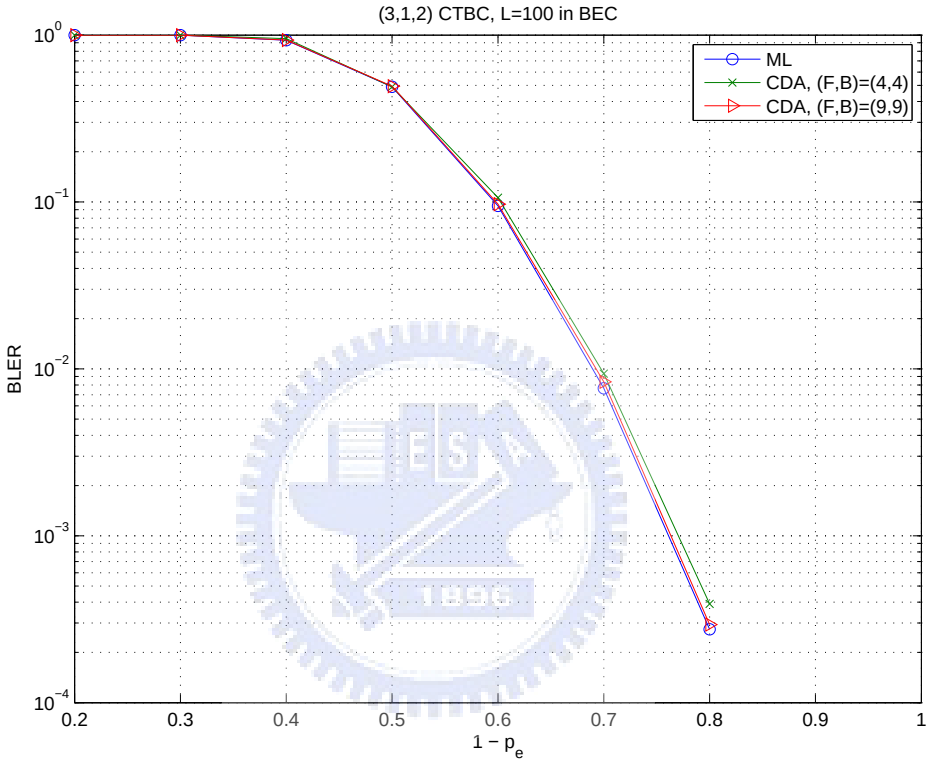


Figure 4.4: Block error rates (BLER) of the CDA for $(3, 1, 2)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$.

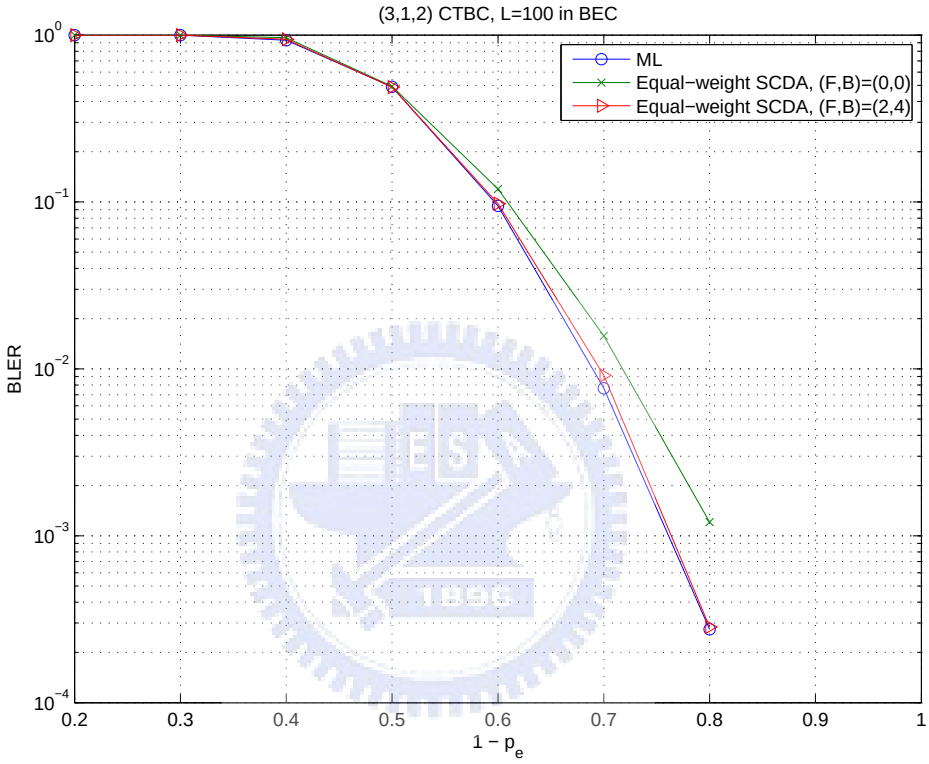


Figure 4.5: Block error rates (BLER) of the SCDA for $(3, 1, 2)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$ and the weighted average window is $W = 16$.

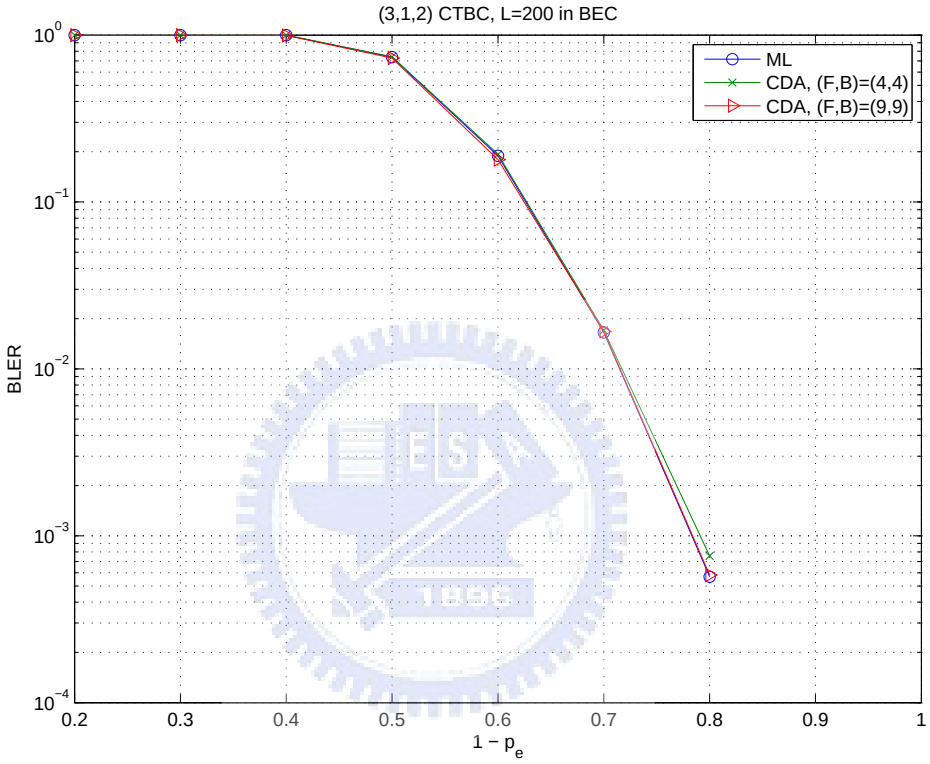


Figure 4.6: Block error rates (BLER) of the CDA for $(3, 1, 2)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$.

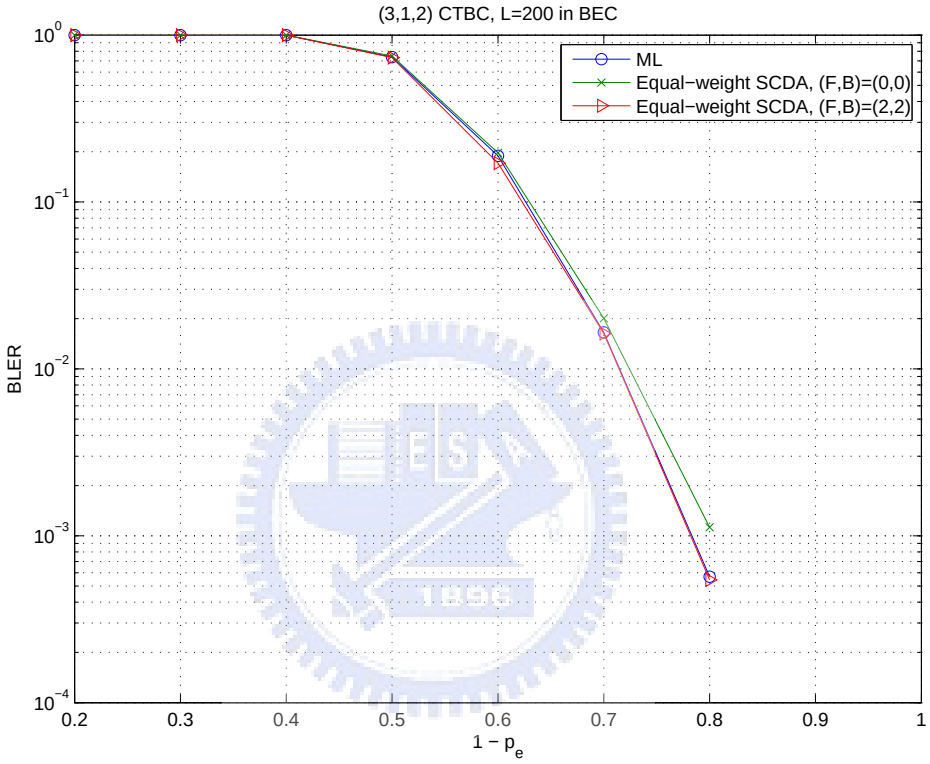


Figure 4.7: Block error rates (BLER) of the SCDA for $(3, 1, 2)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$ and the weighted average window is $W = 16$.

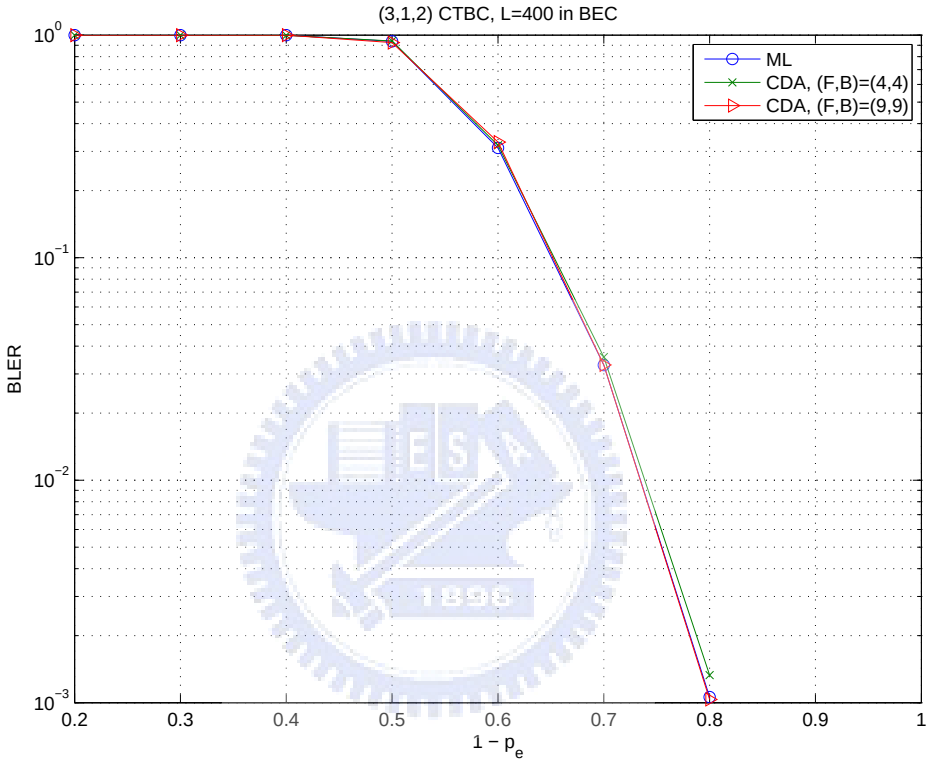


Figure 4.8: Block error rates (BLER) of the CDA for $(3, 1, 2)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$.

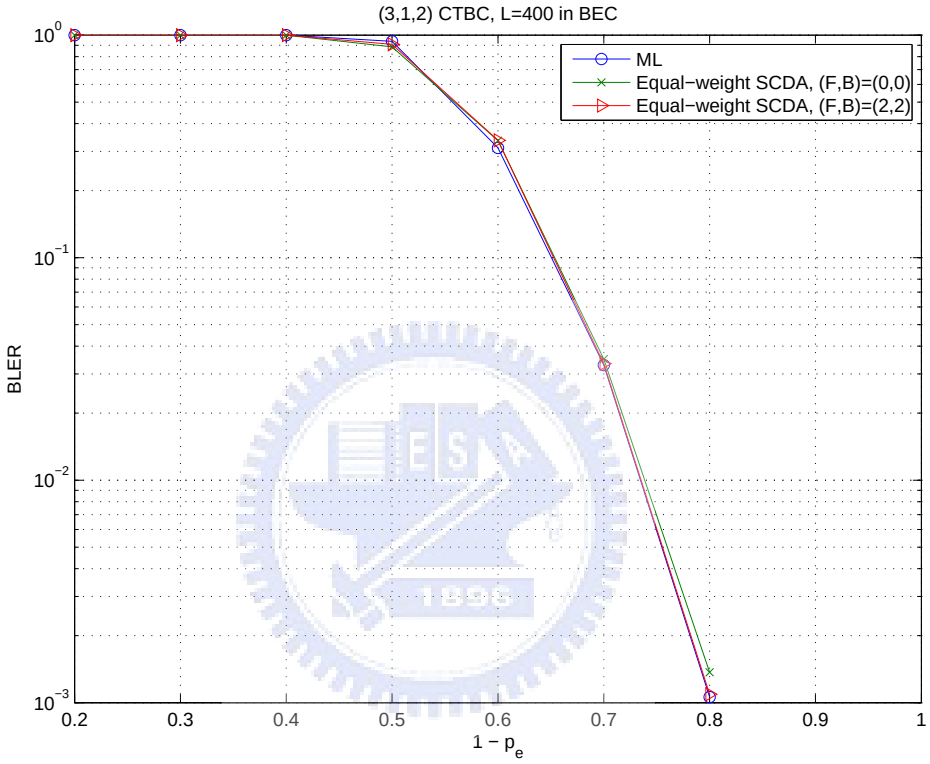


Figure 4.9: Block error rates (BLER) of the SCDA for $(3, 1, 2)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$ and the weighted average window is $W = 16$.

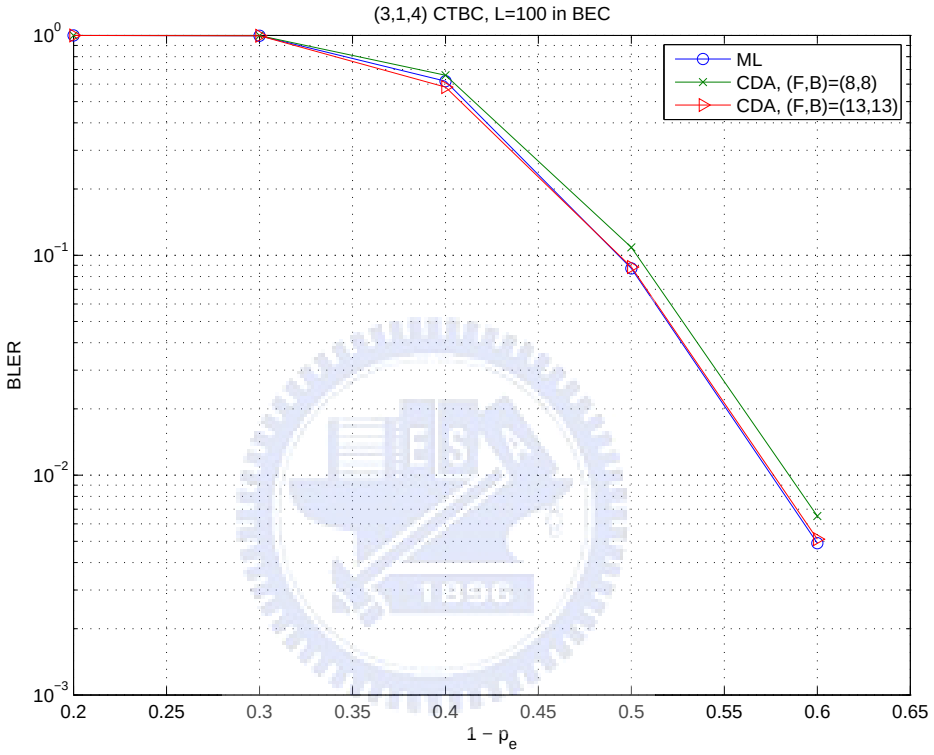


Figure 4.10: Block error rates (BLER) of the CDA for $(3, 1, 4)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$.

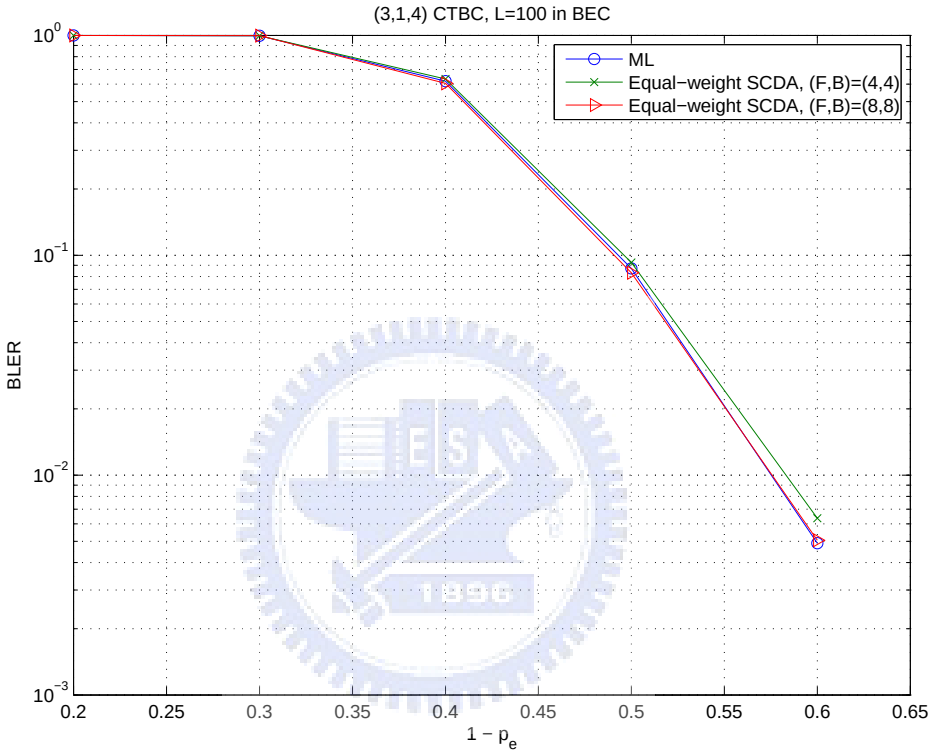


Figure 4.11: Block error rates (BLER) of the SCDA for $(3, 1, 4)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$ and the weighted average window is $W = 24$.

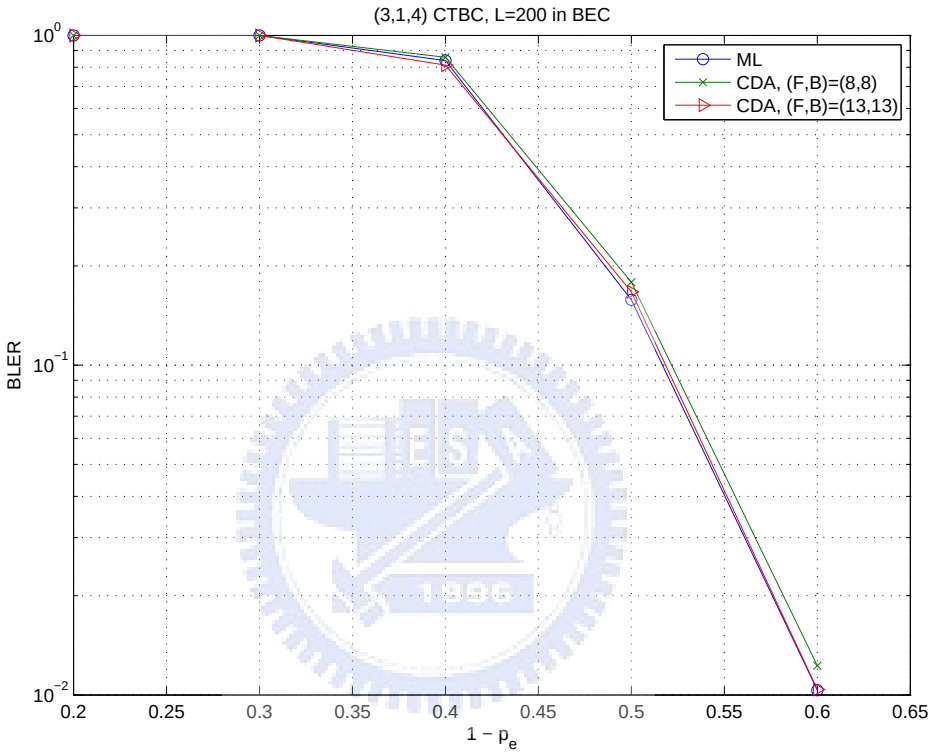


Figure 4.12: Block error rates (BLER) of the CDA for $(3, 1, 4)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$.

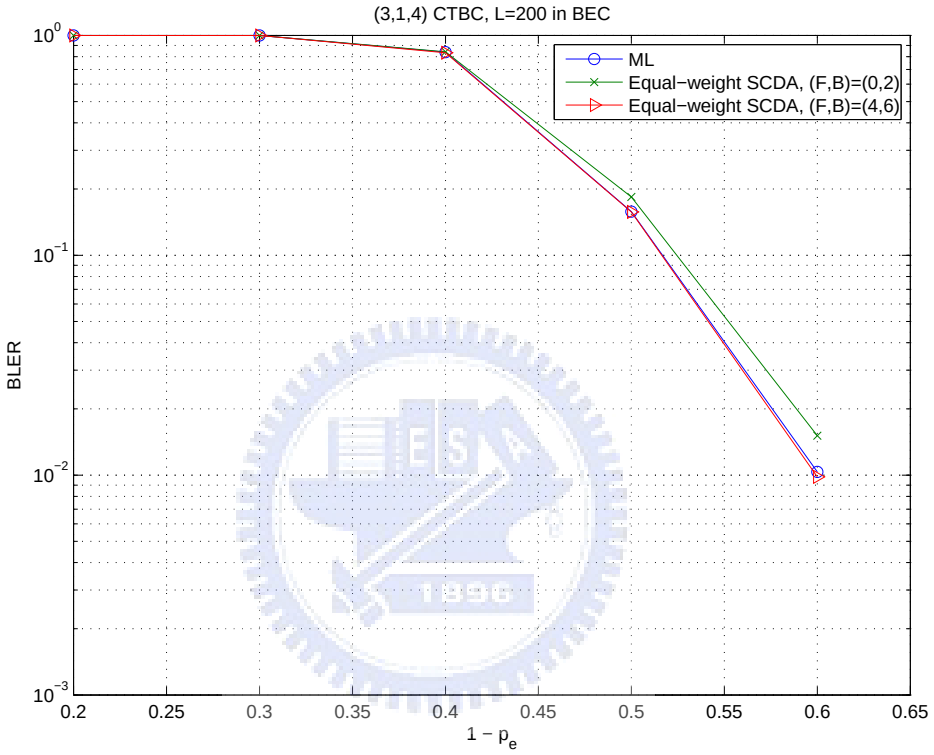


Figure 4.13: Block error rates (BLER) of the SCDA for $(3, 1, 4)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$ and the weighted average window is $W = 24$.

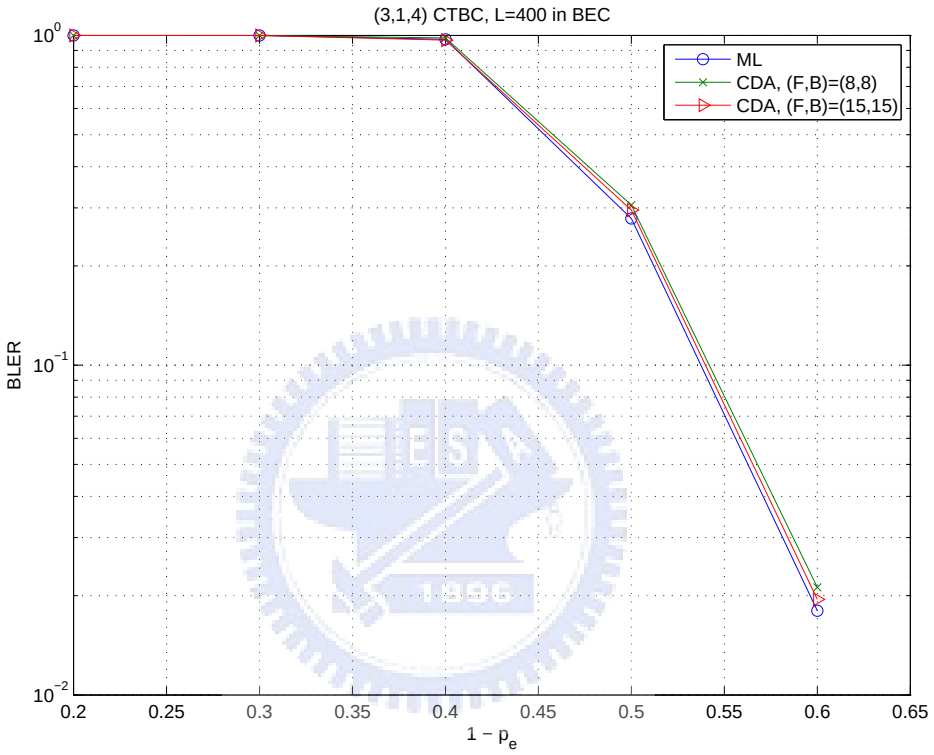


Figure 4.14: Block error rates (BLER) of the CDA for $(3, 1, 4)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$.

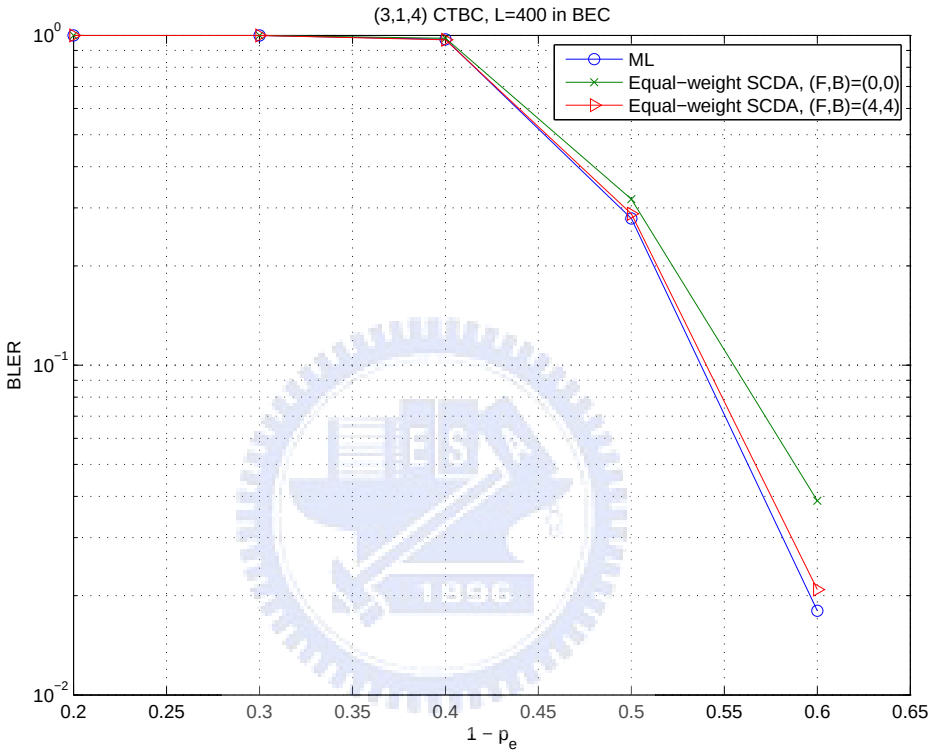


Figure 4.15: Block error rates (BLER) of the SCDA for $(3, 1, 4)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$ and the weighted average window is $W = 24$.

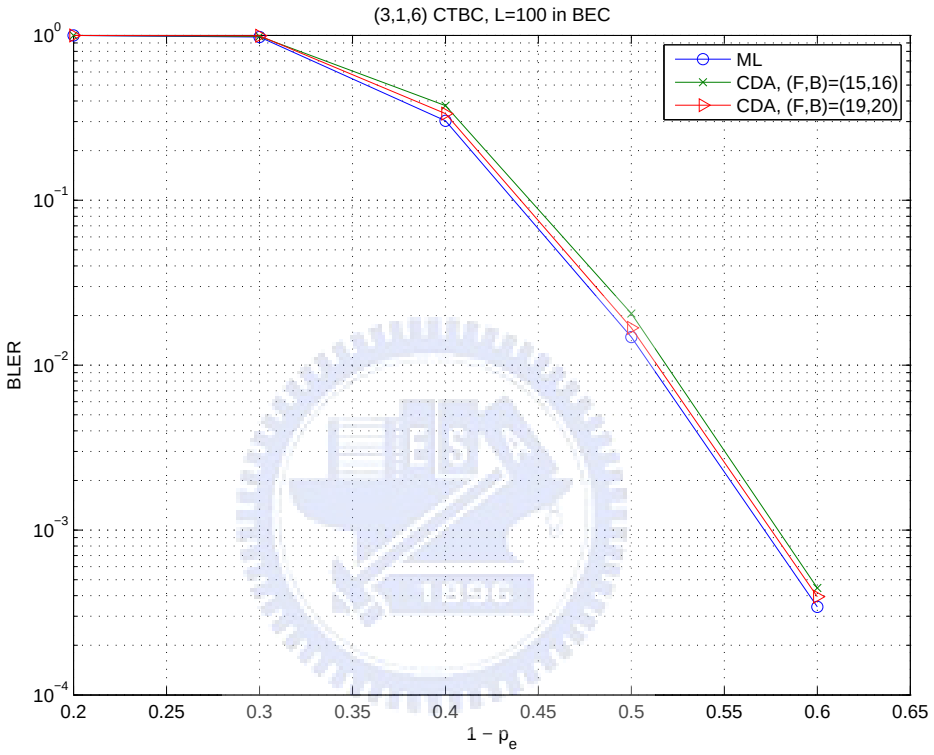


Figure 4.16: Block error rates (BLER) of the CDA for $(3, 1, 6)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$.

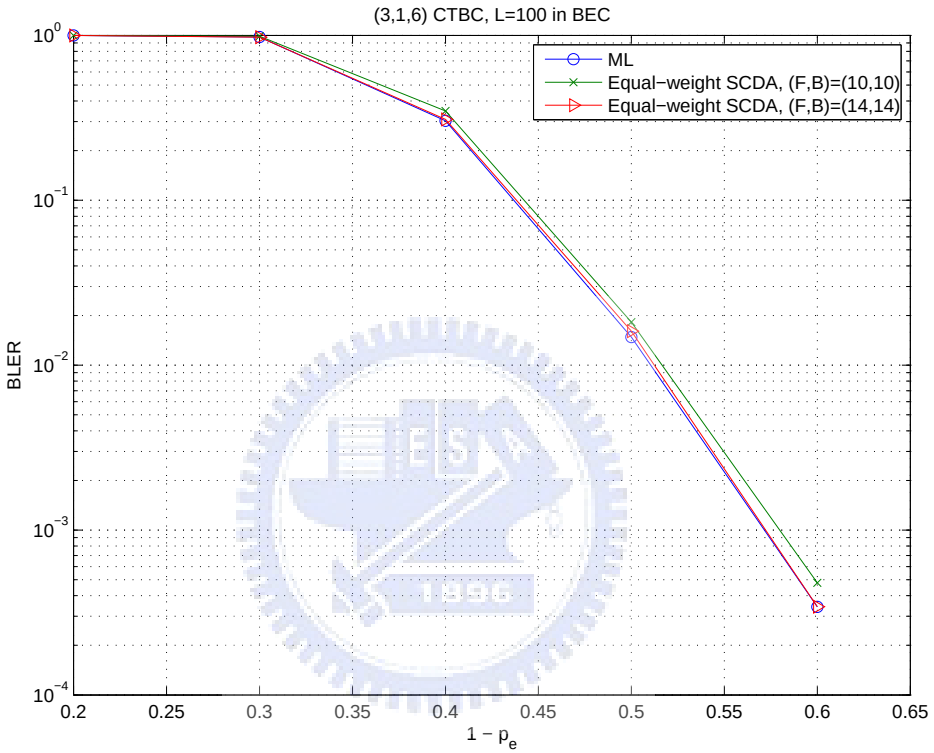


Figure 4.17: Block error rates (BLER) of the SCDA for $(3, 1, 6)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$ and the weighted average window is $W = 30$.

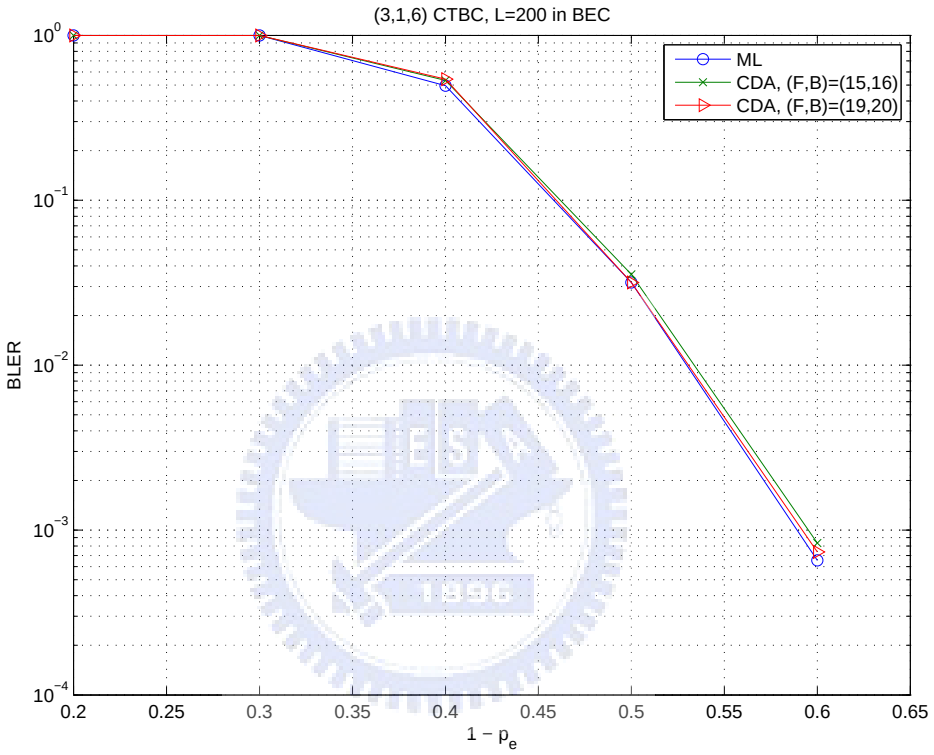


Figure 4.18: Block error rates (BLER) of the CDA for $(3, 1, 6)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$.

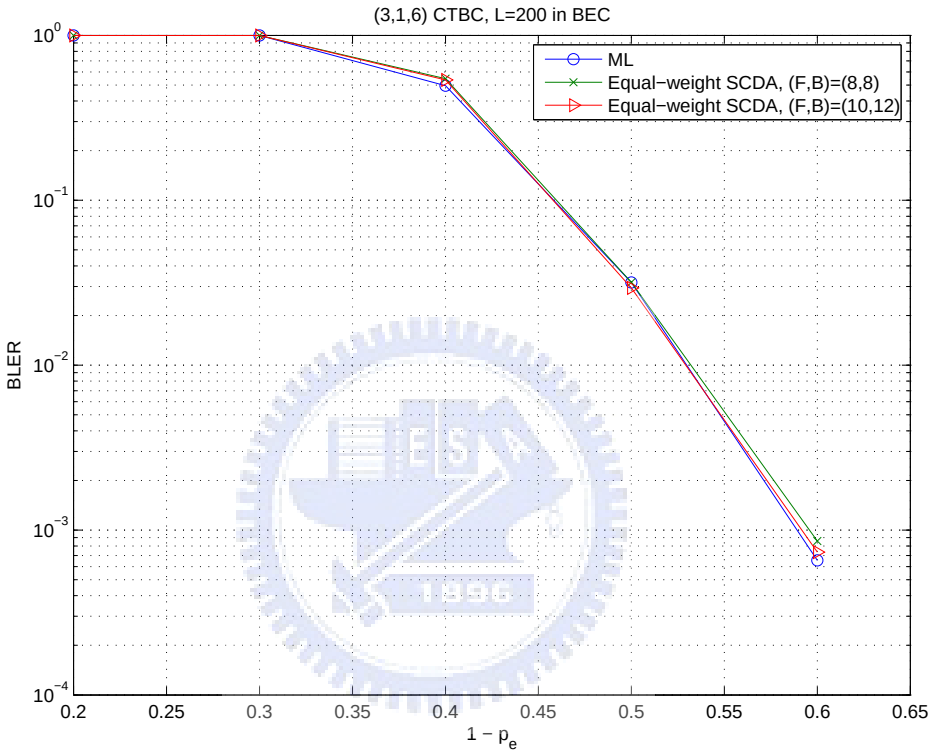


Figure 4.19: Block error rates (BLER) of the SCDA for $(3, 1, 6)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$ and the weighted average window is $W = 30$.

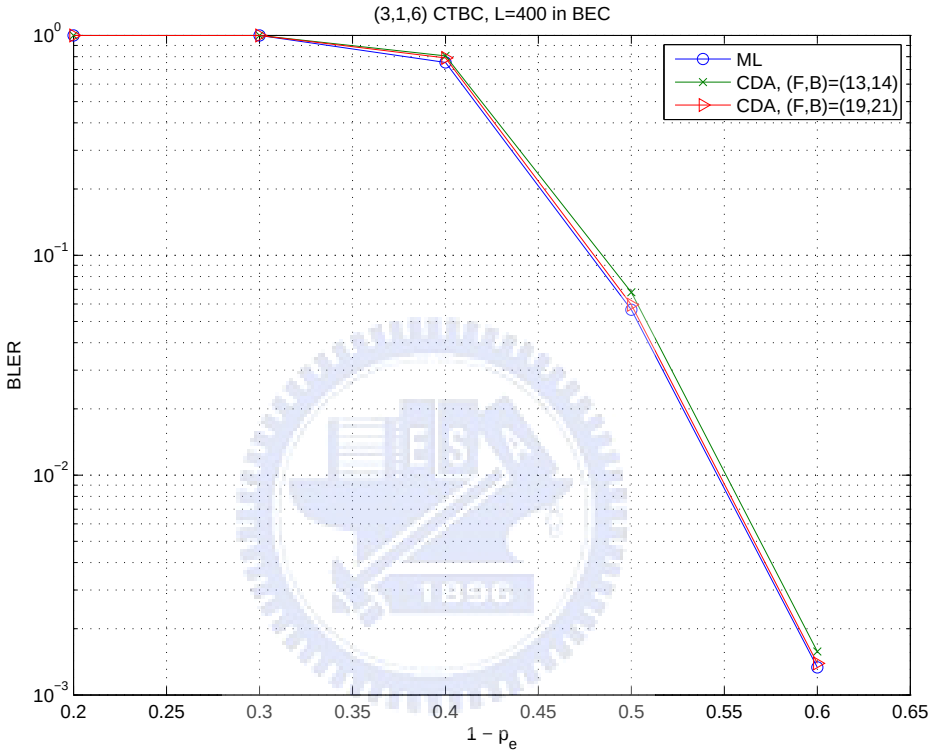


Figure 4.20: Block error rates (BLER) of the CDA for $(3, 1, 6)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$.

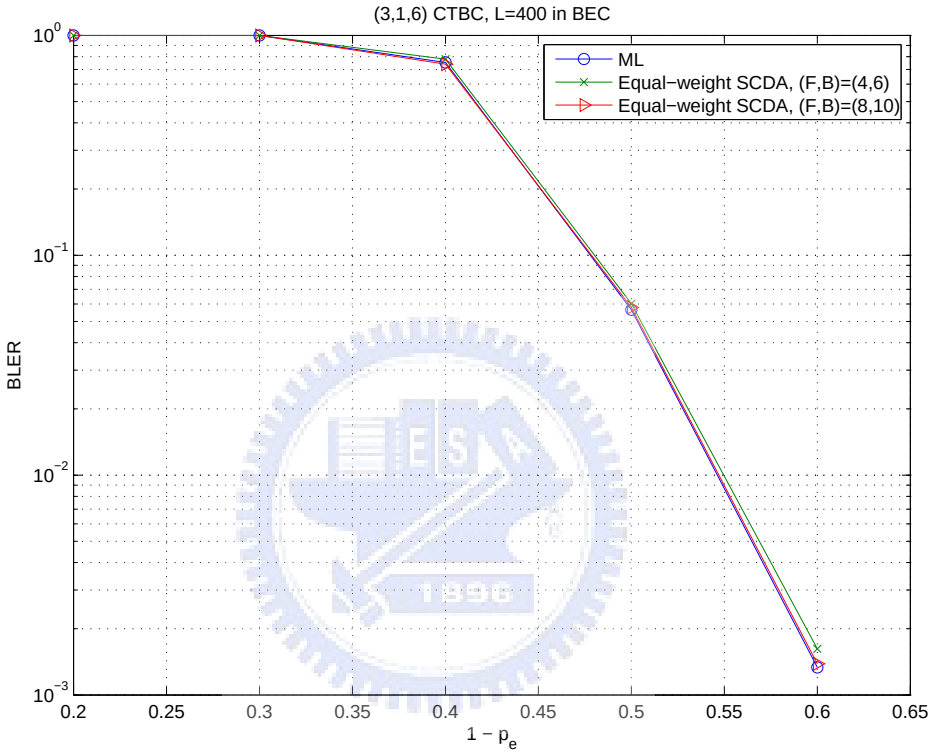


Figure 4.21: Block error rates (BLER) of the SCDA for $(3, 1, 6)$ CTBC in BEC. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$ and the weighted average window is $W = 30$.

4.5 Error Bound for Shifting Circular Decoding in Additive White Gaussian Noise (AWGN) Channel

On AWGN channels, the log-likelihood ratio ϕ_j is proportional to the received scalar r_j. Hence, the pre-decoding circular shifts on the receive sequence $\boldsymbol{r}$ is again performed based on the absolute values of the received components. Since the code we considered is linear, the error rate conditional on the code word transmitted is independent of the code word. We then assume without loss of generality that the transmitted code word is the all-zero one. For notational convenience, we omit the conditional notation in the following derivations.

Similar to the analysis in BEC, in order to simplify the analysis, we assume that the "unadjusted" pre-decoding information circular shift $\tilde{p} = p - W/2$ can only be a multiple of weighted average window W, information length L for $(n, 1, m)$ CTBC is also a multiple of W, and equal weights that are all equal to unity are adopted. Define for $0 \leq i < M = L/W$,

$$U_i = |r_{inW}| + |r_{inW+1}| + \cdots + |r_{inW+nW-1}|.$$

Now, the sorted $U_0, U_1, \ldots, U_{M-1}$ gives that

$$U_{(0)} \leq U_{(1)} \leq U_{(2)} \leq \cdots \leq U_{(M-1)}.$$

In such case, the pre-decoding circular shift $\tilde{p} = Wi$, if $U_{(M-1)} = U_i$, and the unadjusted inputs for the circular decoding algorithm is $r_{\tilde{p}n}, r_{\tilde{p}n+1}, \ldots, r_{N-1}, r_0, r_1, \ldots, r_{\tilde{p}n-1}$. It remains to find the probability density f of the unadjusted shifted input sequence.

It is clear that the density f for $(a_0, a_1, \ldots, a_{N-1}) \in \Re^N$ with

$$\sum_{\ell=0}^{nW-1} |a_\ell| < \max_{1 \leq i \leq M-1} \sum_{\ell=0}^{nW-1} |a_{inW+\ell}|$$

is zero. For $(a_0, a_1, \ldots, a_{N-1}) \in \Re^N$ with $\sum_{\ell=0}^{nW-1} |a_\ell| > \max_{1 \leq i \leq M-1} \sum_{\ell=0}^{nW-1} |a_{inW+\ell}|$,

$$f\left(r_{\tilde{p}n} = a_0, r_{\tilde{p}n+1} = a_1, \cdots, r_{\tilde{p}n-1} = a_{N-1}\right) \;=\; M \prod_{\ell=0}^{N-1} f\left(r_\ell = a_\ell\right), \tag{4.10}$$

where (4.10) is valid since $f\left(r_{inW}=a_0, r_{inW+1}=a_1, \cdots, r_{inW-1}=a_{N-1}\right)$ is the same for every i as $\{r_\ell\}_{\ell=0}^{N-1}$ is i.i.d. The above derivation concludes:

$$\begin{aligned} & f\left(r_{\tilde{p}n}=a_0, r_{\tilde{p}n+1}=a_1, \cdots, r_{\tilde{p}n-1}=a_{N-1}\right) \\ = \; & M\left(\prod_{\ell=0}^{N-1} f\left(r_\ell=a_\ell\right)\right) \mathbf{1}\left\{\max_{1\le i\le M-1} \sum_{\ell=0}^{nW-1} |a_{inW+\ell}| \le \sum_{\ell=0}^{nW-1} |a_\ell|\right\}. \end{aligned} \tag{4.11}$$

Notably,

$$f\left(r_\ell=a_\ell\right)=\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(a_\ell-\mu)^2/(2\sigma^2)},$$

where $\mu=(-1)^0=1$ is the signal transmitted, and σ^2 is the noise variance. It can be observed that (4.11) is not just a function of $\sum_{\ell=0}^{nW-1}|a_{inW+\ell}|$ for $0\le i\le M-1$ in AWGN channel as the the case in BEC. As a result,

$$\begin{aligned} & f\left(r_{\tilde{p}n}=a_0, r_{\tilde{p}n+1}=a_1, \cdots, r_{\tilde{p}n+nW-1}=a_{nW-1}\right) \\ = \; & \int_{(a_{nW},\ldots,a_{N-1})\in\Re^{N-nW}} f\left(r_{\tilde{p}n}=a_0, r_{\tilde{p}n+1}=a_1, \cdots, r_{\tilde{p}n-1}=a_{N-1}\right) da_{nW}\cdots da_{N-1} \\ = \; & M\left(\prod_{\ell=0}^{nW-1} f\left(r_\ell=a_\ell\right)\right) \\ & \quad \times \int_{\{(a_{nW},\ldots,a_{N-1})\in\Re^{N-nW}\,:\,\max_{1\le i\le M-1} b_i\le b_0\}} \left(\prod_{\ell=nW}^{N-1} f\left(r_\ell=a_\ell\right)\right) da_{nW}\cdots da_{N-1} \\ = \; & M\left(\prod_{\ell=0}^{nW-1} f\left(r_\ell=a_\ell\right)\right) \\ & \quad \times \prod_{i=1}^{M-1} \int_{\{(a_{inW},\ldots,a_{inW+nW-1})\in\Re^{nW}\,:\,b_i\le b_0\}} \left(\prod_{\ell=0}^{nW-1} f\left(r_{inW+\ell}=a_{inW+\ell}\right)\right) da_{inW}\cdots da_{inW+nW-1} \\ = \; & M\left(\prod_{\ell=0}^{nW-1} f\left(r_\ell=a_\ell\right)\right) \prod_{i=1}^{M-1} \Pr[U_i\le b_0] \\ = \; & M\left(\prod_{\ell=0}^{nW-1} f\left(r_\ell=a_\ell\right)\right) \left(\Pr[U\le b_0]\right)^{M-1}, \end{aligned}$$

where for notational convenient, we denote $b_i=\sum_{\ell=0}^{nW-1}|a_{inW+\ell}|$, and U is a random variable, independent of r_0, …, r_{nW-1}, of which the probability density is the same as $|r_0|+\cdots+$

$|r_{nW-1}|$. This leads to that

$$\begin{aligned}
& f\left(r_{\tilde{p}n}=a_0\right) \\
&= \int_{(a_1,\ldots,a_{nW-1})\in\Re^{nW-1}} f\left(r_{\tilde{p}n}=a_0, r_{\tilde{p}n+1}=a_1,\cdots,r_{\tilde{p}n+nW-1}=a_{nW-1}\right) da_1\cdots d_{nW-1} \\
&= \int_{(a_1,\ldots,a_{nW-1})\in\Re^{nW-1}} M\left(\prod_{\ell=0}^{nW-1} f\left(r_\ell=a_\ell\right)\right)\left(\Pr[U\le b_0]\right)^{M-1} da_1\cdots d_{nW-1} \\
&= M\cdot f(r_0=a_0)\int_{(a_1,\ldots,a_{nW-1})\in\Re^{nW-1}} \left(\prod_{\ell=1}^{nW-1} f\left(r_\ell=a_\ell\right)\right)\left(\Pr[U\le b_0]\right)^{M-1} da_1\cdots da_{nW-1} \\
&= M\cdot f(r_0=a_0)E_B\left[\left(\Pr[U\le |a_0|+B]\right)^{M-1}\right]
\end{aligned}$$

where random variable B is defined as $B\triangleq\sum_{\ell=1}^{nW-1}|r_\ell|$. By letting f_0 and f_1 be respectively the densities of random variables $|r_0|$ and B, the density of U should be the same as that of $|r_0|+B$ with $|r_0|$ and B being independent. Thus,

$$\begin{aligned}
E_B\left[\left(\Pr[U\le |a_0|+B]\right)^{M-1}\right] &= \int_0^\infty f_1(b)\left(\int_0^{|a_0|+b}\left(\int_0^\infty f_1(\tau)f_0(t-\tau)d\tau\right)dt\right)^{M-1} db \\
&= \int_0^\infty f_1(b)\left(\int_0^\infty f_1(\tau)\int_0^{|a_0|+b} f_0(t-\tau)dtd\tau\right)^{M-1} db \\
&= \int_0^\infty f_1(b)\left(\int_0^{|a_0|+b} f_1(\tau)F_0(|a_0|+b-\tau)d\tau\right)^{M-1} db,
\end{aligned}$$

where F_0 denotes the cumulative distribution function of random variable $|r_0|$.

This gives the effective marginal density based on pre-decoding circular shift we propose. We can then numerically compute the mean and second moment of $r_{\tilde{p}n}$ given that transmitted codeword bit is 0 to obtain the effective mean μ_e and effective noise variance σ_e^2 (and hence, the effective signal-to-noise ratio).[6]

[6]From what we just derived,

$$E[r_{\tilde{p}n}]=M\int_{-\infty}^{\infty}\frac{a_0}{\sqrt{2\pi\sigma^2}}e^{-(a_0-\mu)^2/(2\sigma^2)}\left(\int_0^\infty f_1(b)\left(\int_0^{|a_0|+b} f_1(\tau)F_0(|a_0|+b-\tau)d\tau\right)^{M-1} db\right)da_0$$

Again, through several empirical examinations, we observe that taking this new signal-to-noise ratio into (4.6) with $X = e^{-\mu_e^2/(2\sigma_e^2)}$ also provides an upper error bound on P_E for the shifting circular decoding algorithm. We however fail to substantiate this observation analytical, and will defer it as our future work.

Based on our "conjectured" observation, we will try to determine (F, B) "analytically". Specifically, considering only the terms with the lowest exponent respectively for the three terms in (4.6), we first solve

$$\exp\left\{-d_{\text{free}} \cdot \frac{\mu^2}{2\sigma^2}\right\} = \exp\left\{-d \cdot \frac{\mu_e^2}{2\sigma_e^2}\right\}.$$

We then choose F and B such that d_F and d_B equal $\lceil d \rceil$. In the next section, we will compare F and B derived in this section with those obtained by simulations.

4.6 Forward and Backward Training Windows in AWGN

Now, based on the derivation in the previous section, we investigate the forward and backward window sizes of the CDA and equal-weight SCDA, requiring for near maximum-likelihood performance in AWGN. The weighted average window is taken to be twice of the d_{free}, i.e., $W = 2d_{\text{free}}$. Three CTBC codes are considered, which are the $(3, 1, 4)$, $(3, 1, 6)$ and $(2, 1, 6)$ codes respectively with generator polynomials (52 66 76), (554 744 724) and (744 554) (in octal). In addition, we adopt $E_b/N_0 = 3$ dB as the operation SNR for $(2, 1, 6)$ and $(3, 1, 6)$, and $E_b/N_0 = 4$ dB for $(3, 1, 4)$. The results are summarized in Tables 4.4, 4.5 and 4.6. They show that the (F, B) of the CDA required for near ML performance in

---

and

$$E[r_{\tilde{p}n}^2] = M \int_{-\infty}^{\infty} \frac{a_0^2}{\sqrt{2\pi\sigma^2}} e^{-(a_0-\mu)^2/(2\sigma^2)} \left(\int_0^{\infty} f_1(b) \left(\int_0^{|a_0|+b} f_1(\tau) F_0(|a_0| + b - \tau) d\tau \right)^{M-1} db \right) da_0.$$

We then have: $\mu_e = E[r_{\tilde{p}n}]$ and $\sigma_e^2 = E[r_{\tilde{p}n}^2] - E^2[r_{\tilde{p}n}]$.

Table 4.4: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 4)$ CTBC. The weighted average window is $W = 24$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(13, 13)$	$(13, 13)$	$(13, 13)$
Derived (F, B) of equal-weight SCDA	$(12, 12)$	$(9, 9)$	$(7, 7)$
Simulated (F, B) of equal-weight SCDA	$(6, 10)$	$(4, 8)$	$(4, 6)$

Table 4.5: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 6)$ CTBC. The weighted average window is $W = 30$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(19, 20)$	$(19, 20)$	$(19, 20)$
Derived (F, B) of equal-weight SCDA	$(16, 19)$	$(15, 18)$	$(12, 15)$
Simulated (F, B) of equal-weight SCDA	$(12, 16)$	$(10, 14)$	$(6, 12)$

AWGN match those reported in [11]. Also, the derived (F, B) pairs are a little larger than the simulated ones. The performance curves can be found in Figs 4.22–4.39.

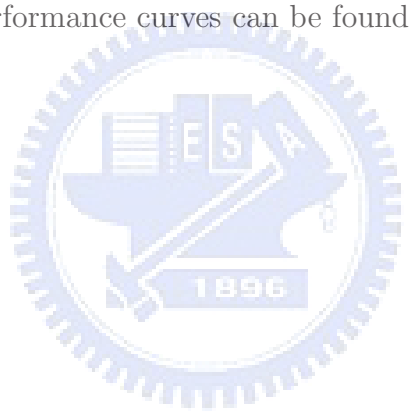


Table 4.6: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(2, 1, 6)$ CTBC. The weighted average window is $W = 20$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(27, 28)$	$(27, 28)$	$(27, 28)$
Derived (F, B) of equal-weight SCDA	$(17, 18)$	$(10, 11-15)$	$(7, 8-10)$
Simulated (F, B) of equal-weight SCDA	$(14, 18)$	$(8, 12)$	$(6, 10)$

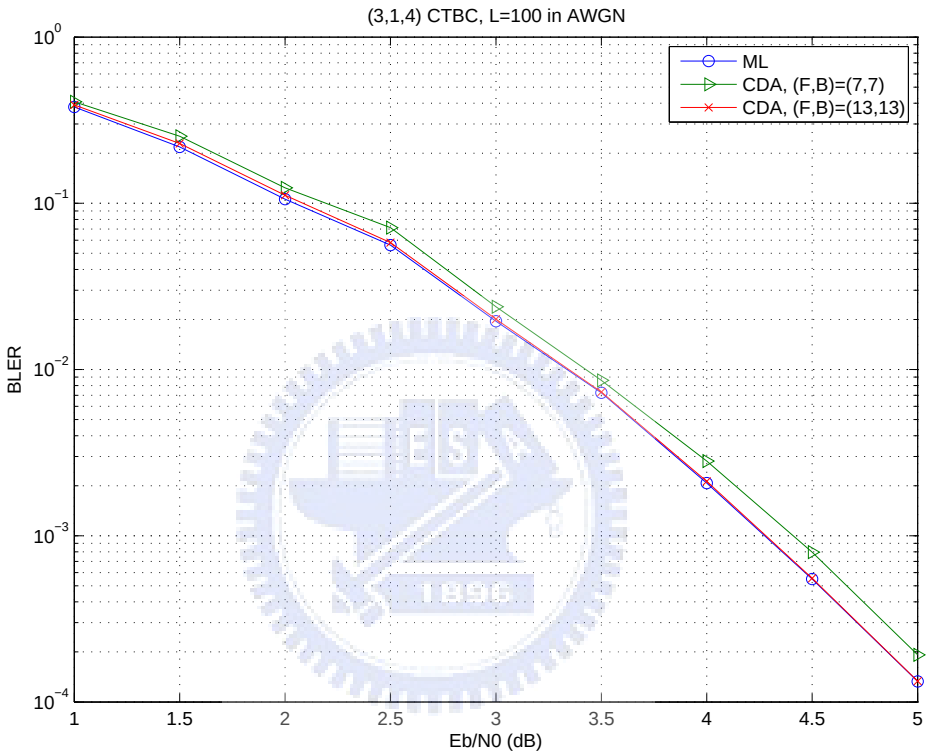


Figure 4.22: Block error rates (BLER) of the CDA for $(3, 1, 4)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$.

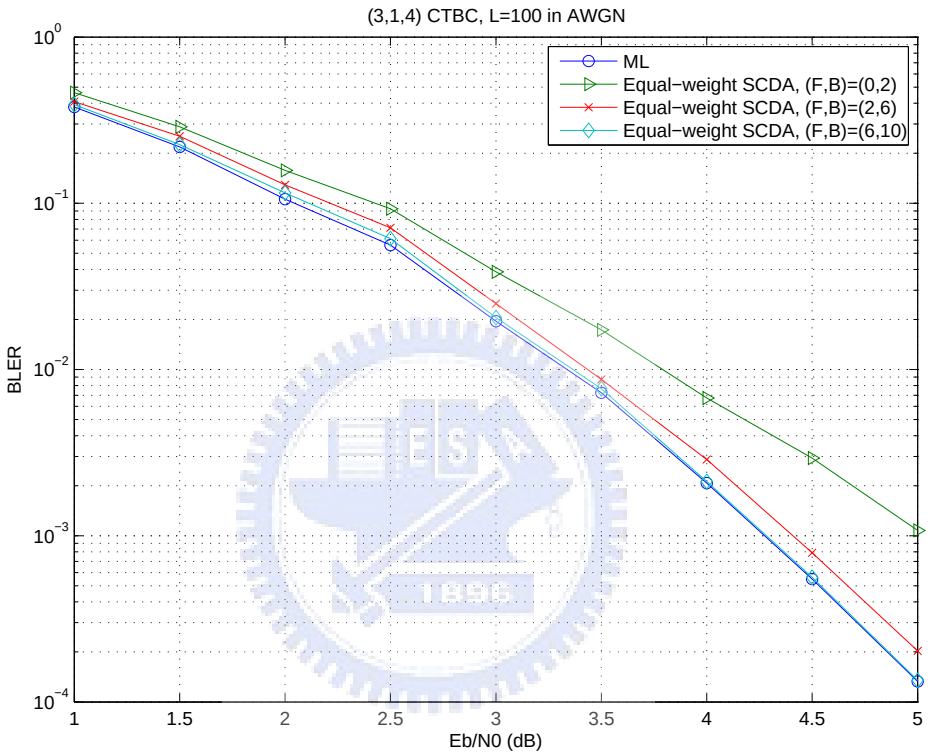


Figure 4.23: Block error rates (BLER) of the SCDA for $(3, 1, 4)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$ and the weighted average window is $W = 24$.

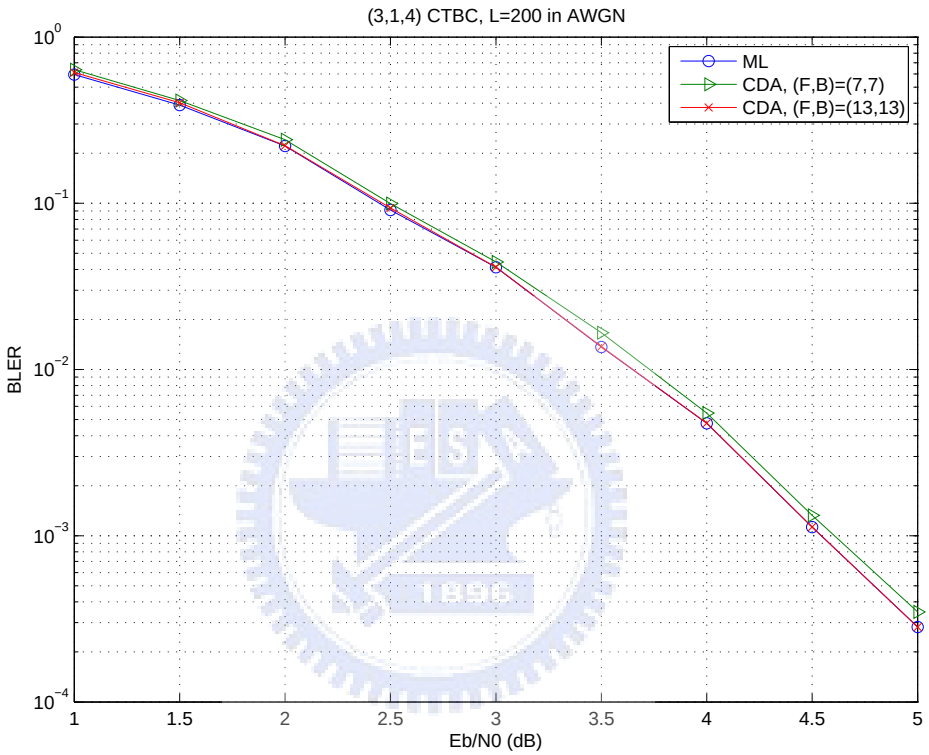


Figure 4.24: Block error rates (BLER) of the CDA for $(3, 1, 4)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$.

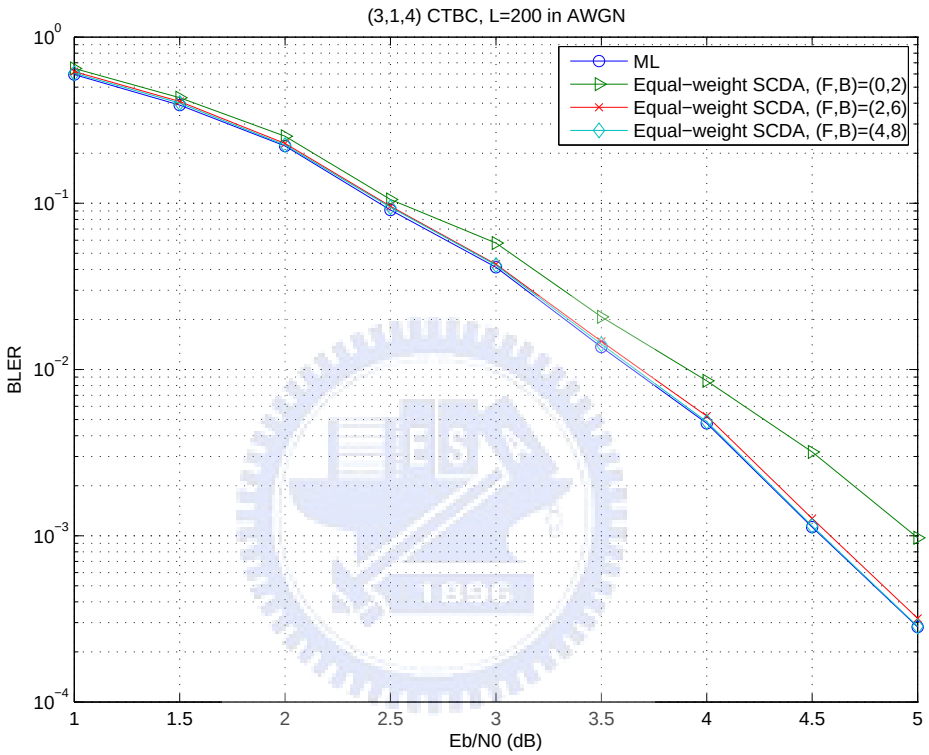


Figure 4.25: Block error rates (BLER) of the SCDA for $(3, 1, 4)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$ and the weighted average window is $W = 24$.

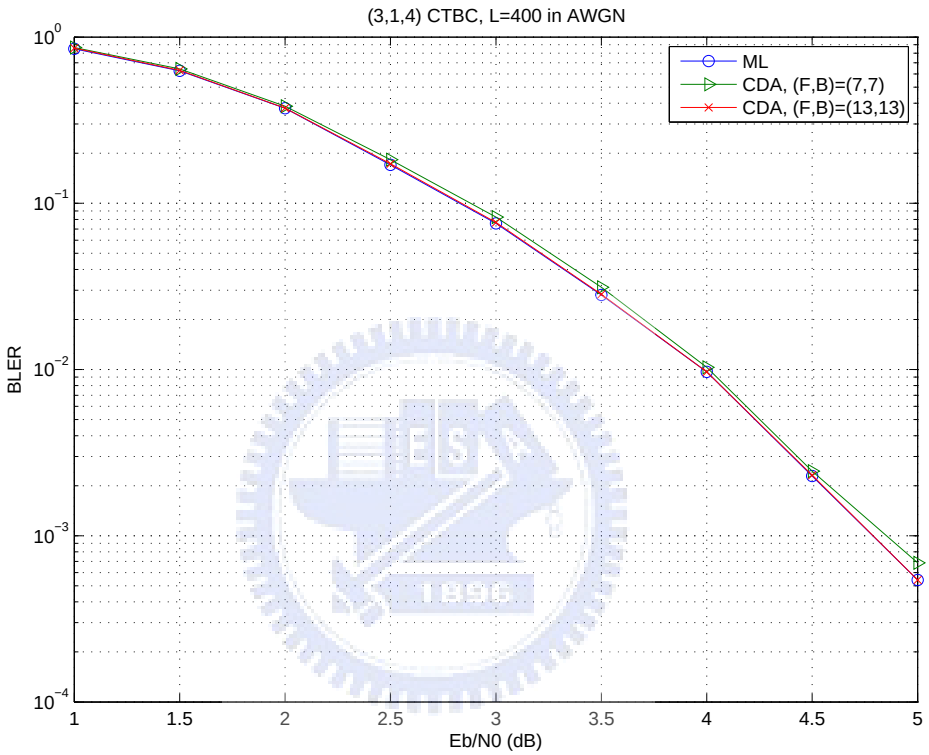


Figure 4.26: Block error rates (BLER) of the CDA for $(3, 1, 4)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$.

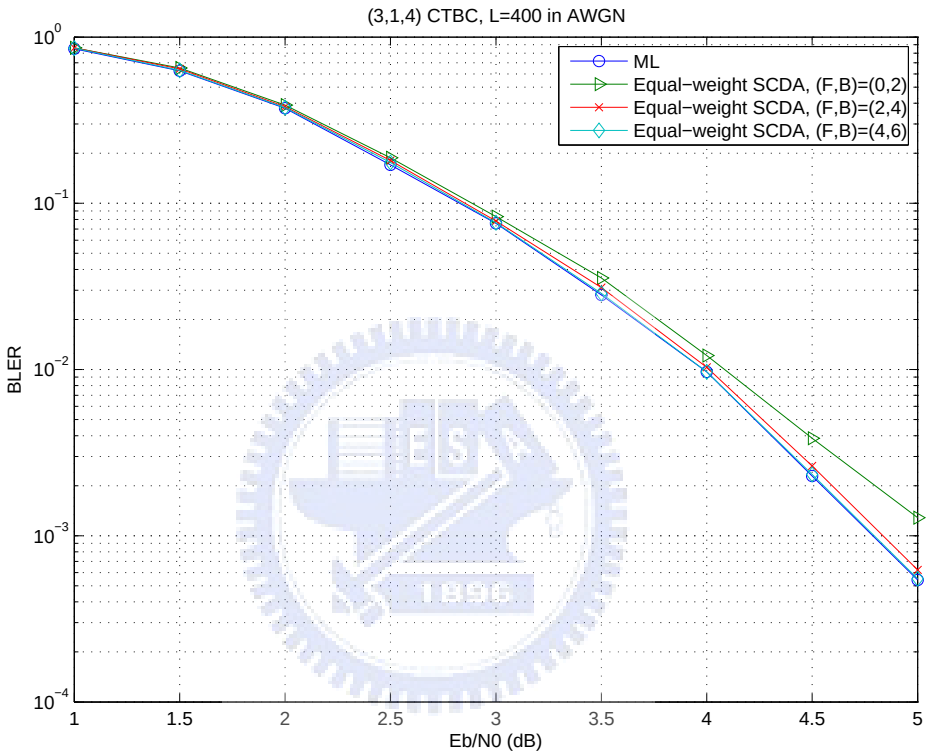


Figure 4.27: Block error rates (BLER) of the SCDA for $(3, 1, 4)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$ and the weighted average window is $W = 24$.

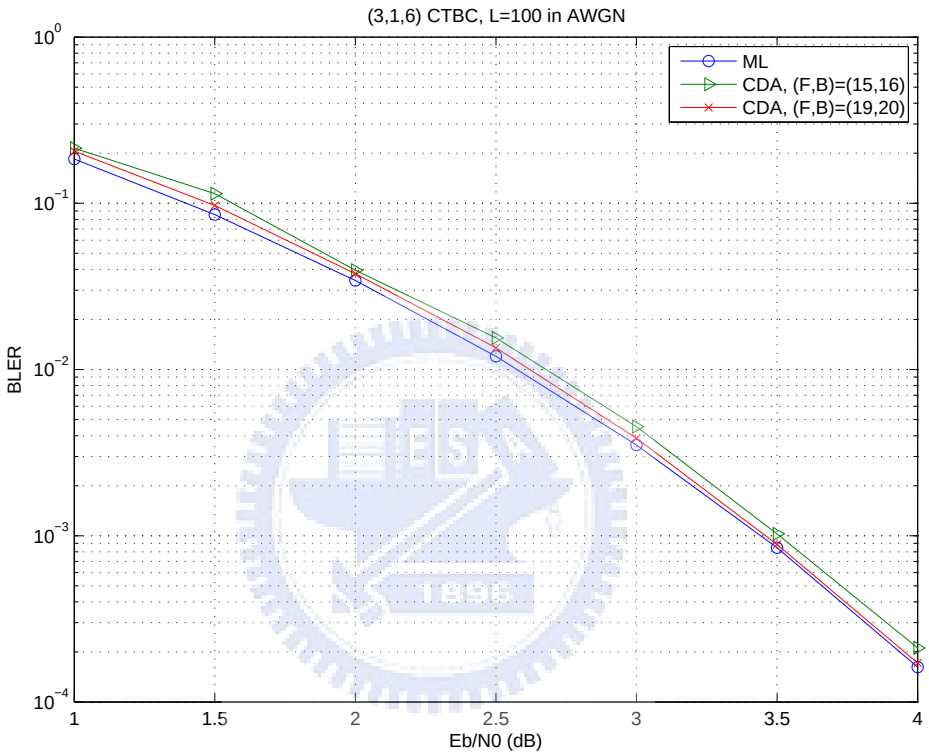


Figure 4.28: Block error rates (BLER) of the CDA for $(3, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$.

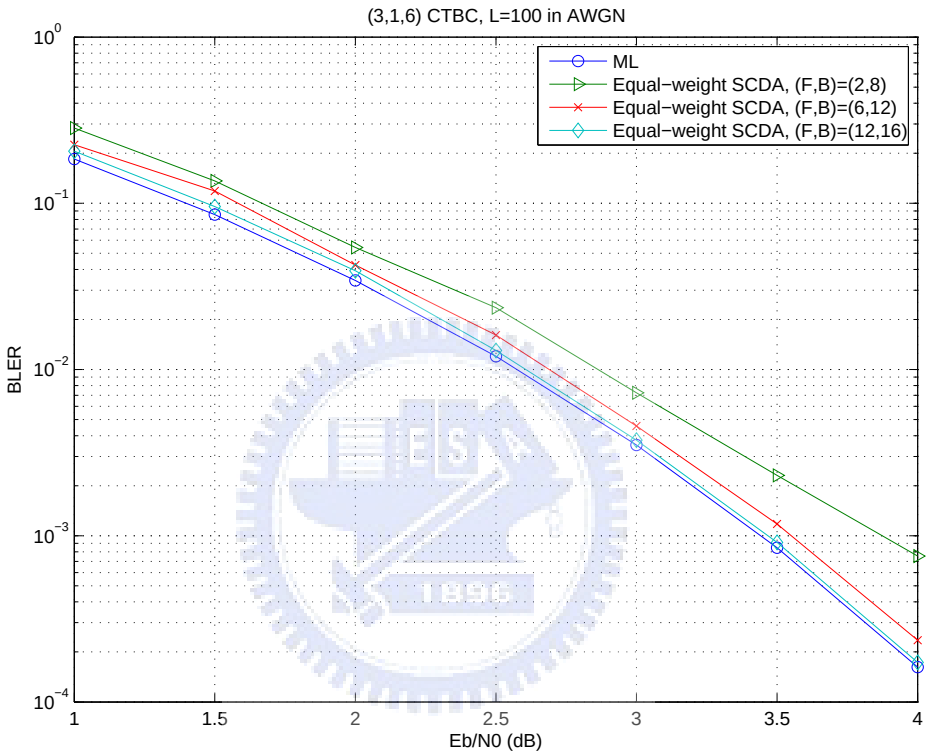


Figure 4.29: Block error rates (BLER) of the SCDA for $(3, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$ and the weighted average window is $W = 30$.

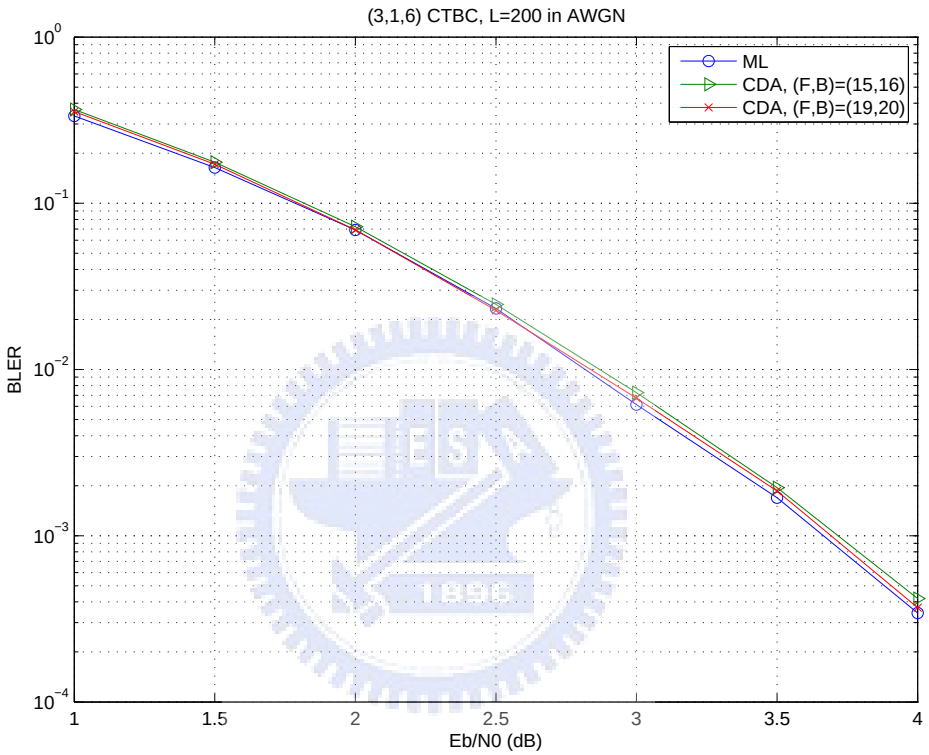


Figure 4.30: Block error rates (BLER) of the CDA for $(3, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$.

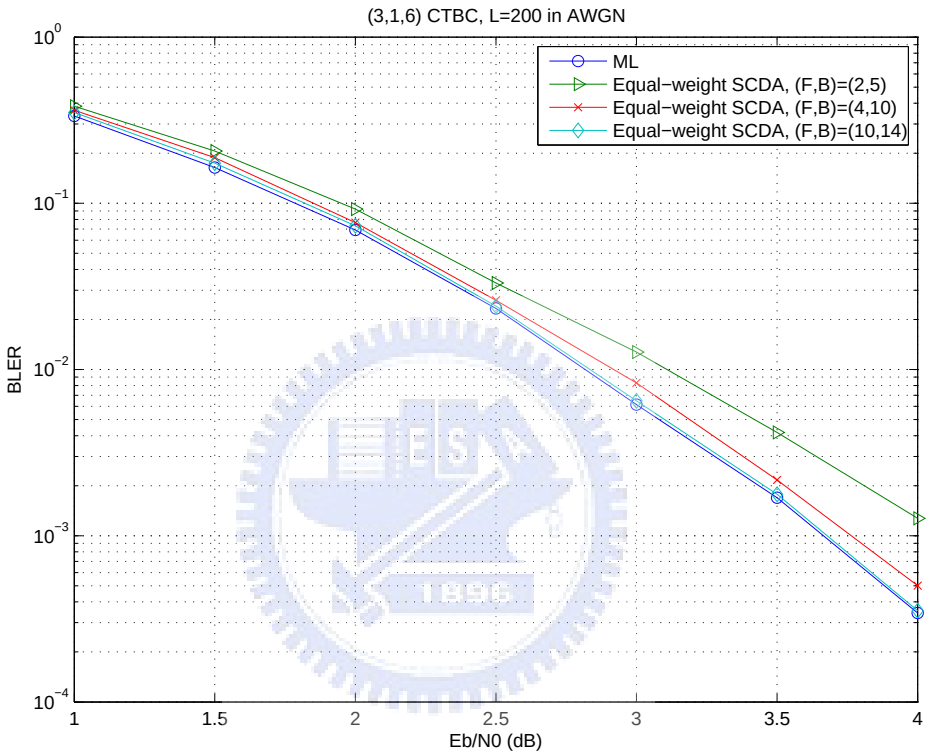


Figure 4.31: Block error rates (BLER) of the SCDA for $(3, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$ and the weighted average window is $W = 30$.

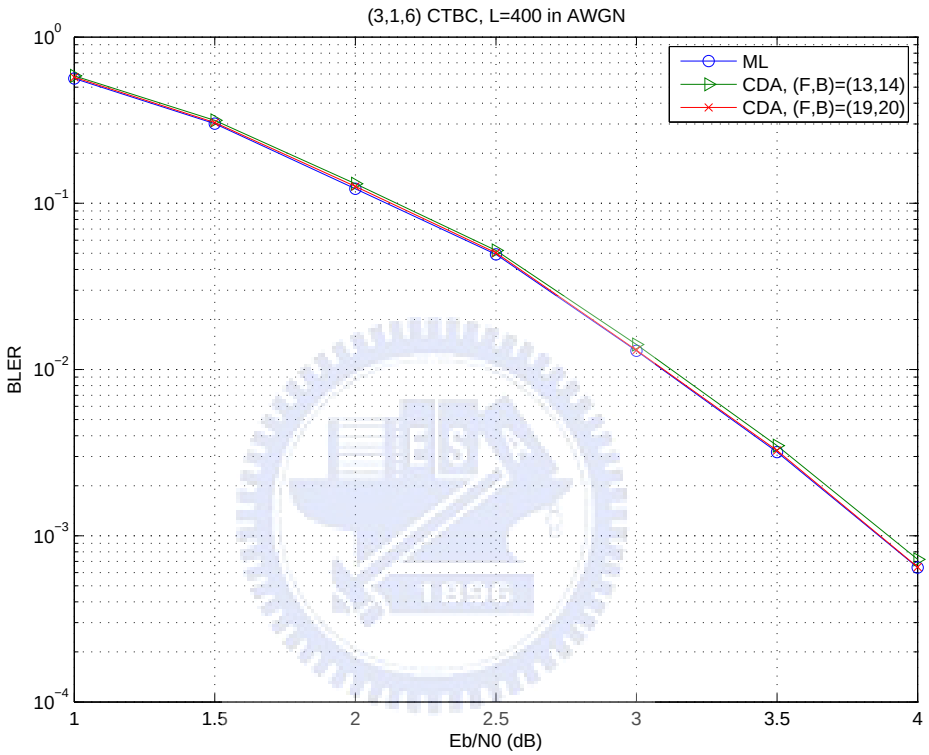


Figure 4.32: Block error rates (BLER) of the CDA for $(3, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$.

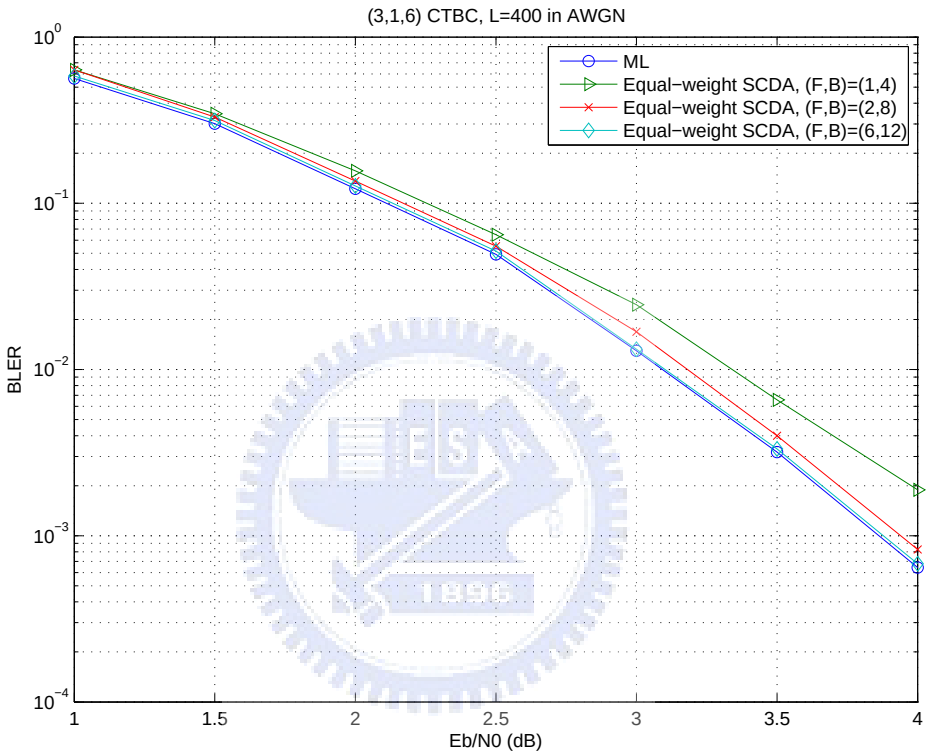


Figure 4.33: Block error rates (BLER) of the SCDA for $(3, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$ and the weighted average window is $W = 30$.

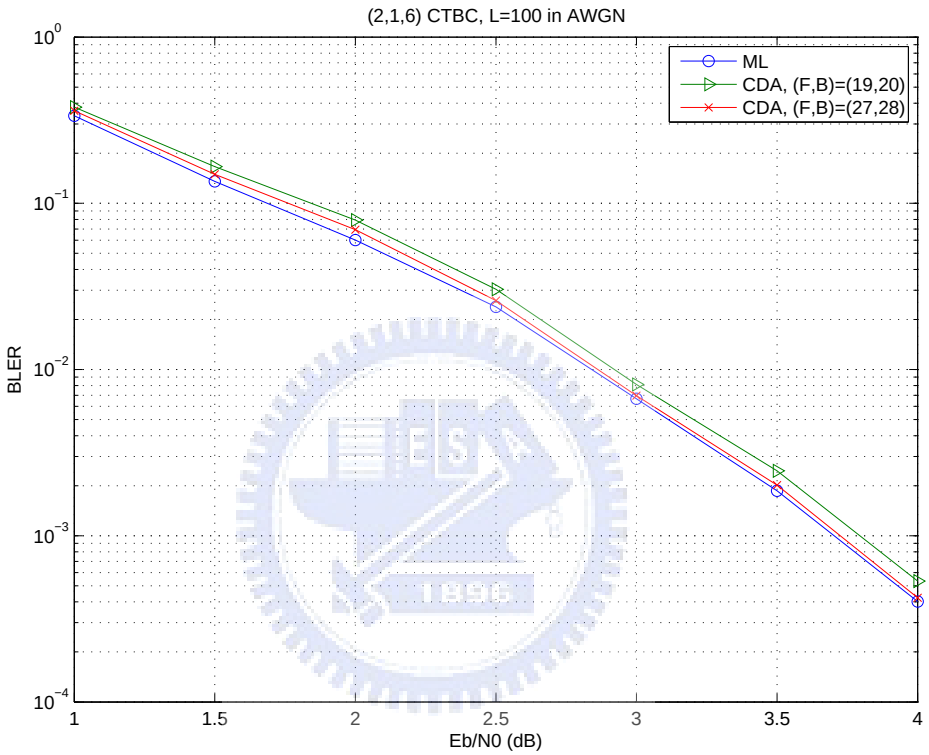


Figure 4.34: Block error rates (BLER) of the CDA for $(2, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$.

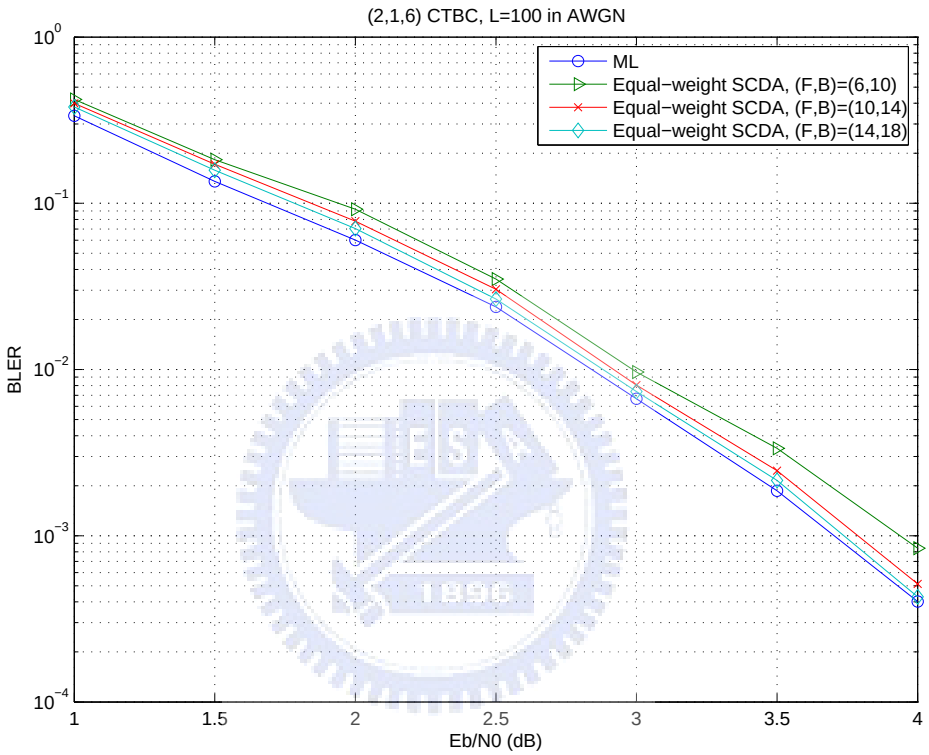


Figure 4.35: Block error rates (BLER) of the SCDA for $(2, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 100$ and the weighted average window is $W = 20$.

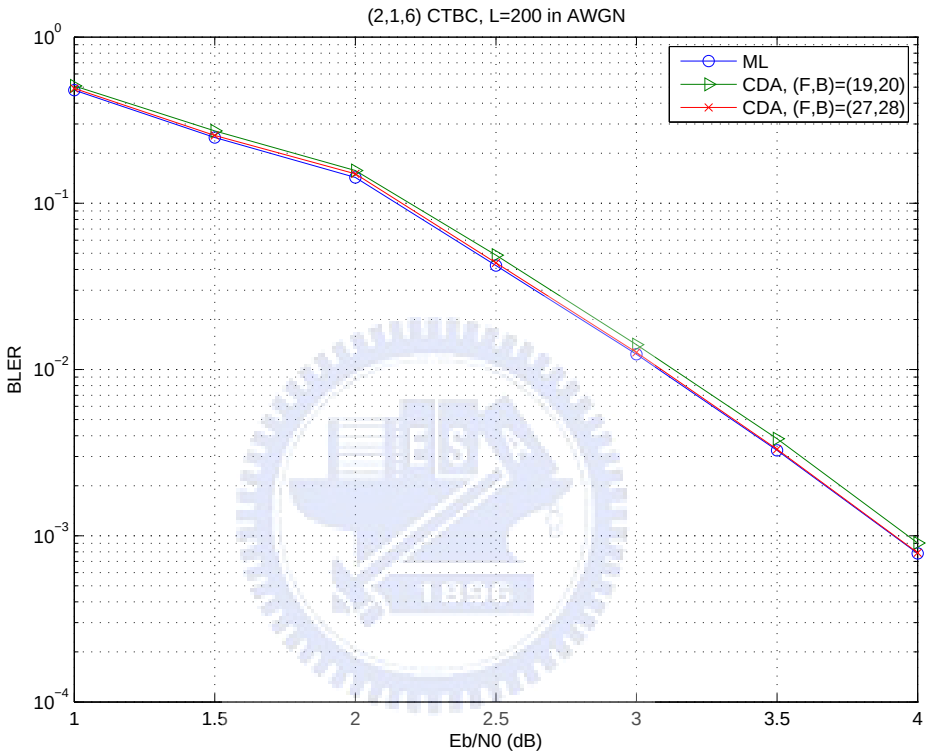


Figure 4.36: Block error rates (BLER) of the CDA for $(2, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$.

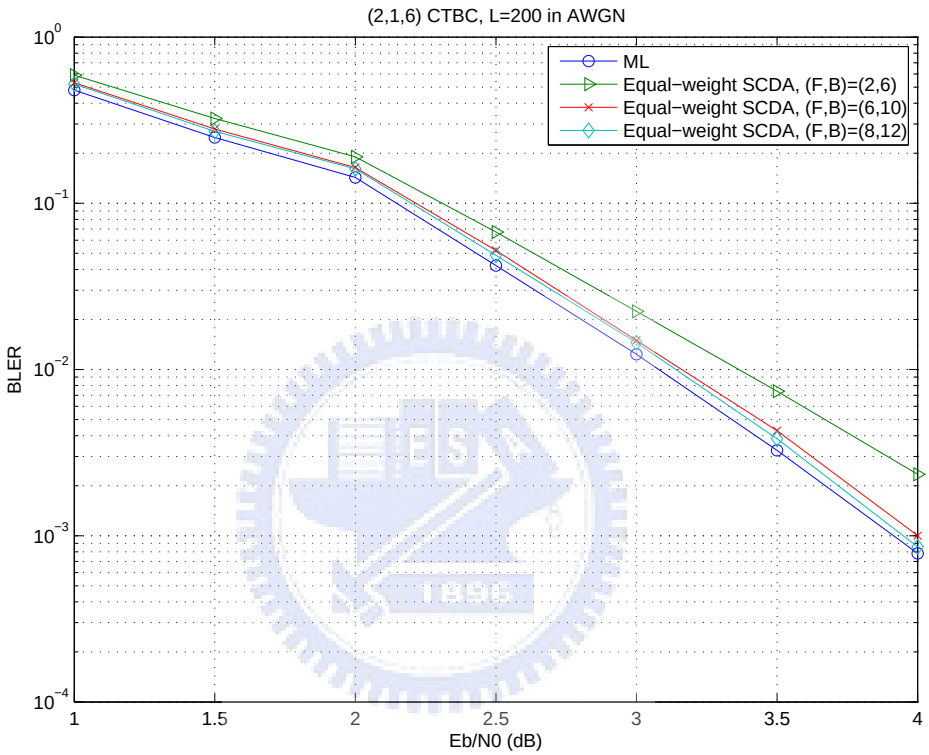


Figure 4.37: Block error rates (BLER) of the SCDA for $(2, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 200$ and the weighted average window is $W = 20$.

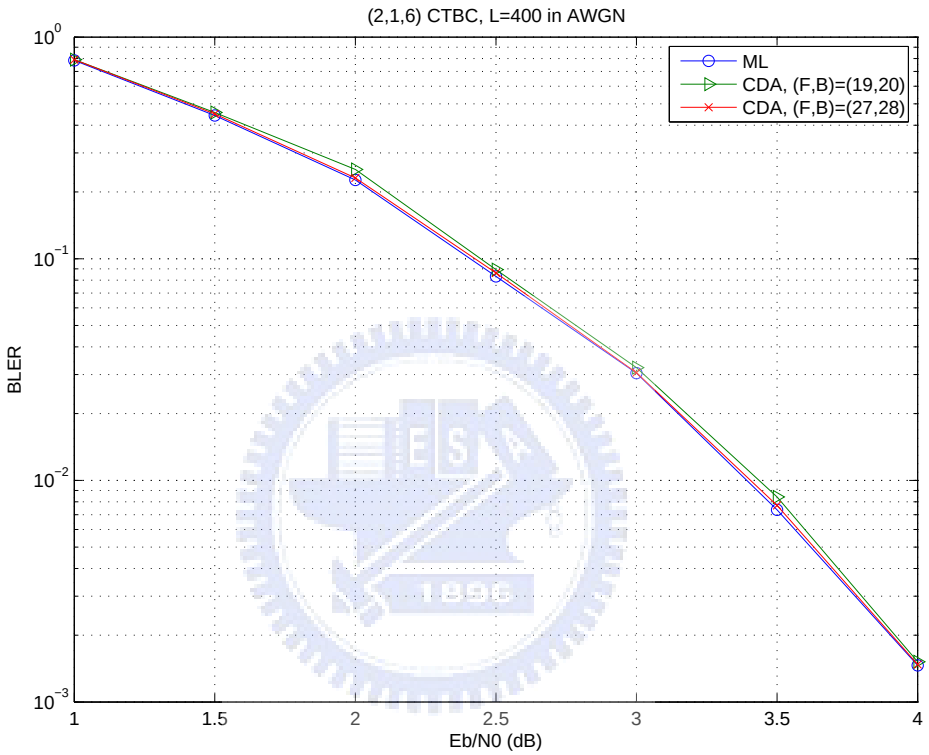


Figure 4.38: Block error rates (BLER) of the CDA for $(2, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$.

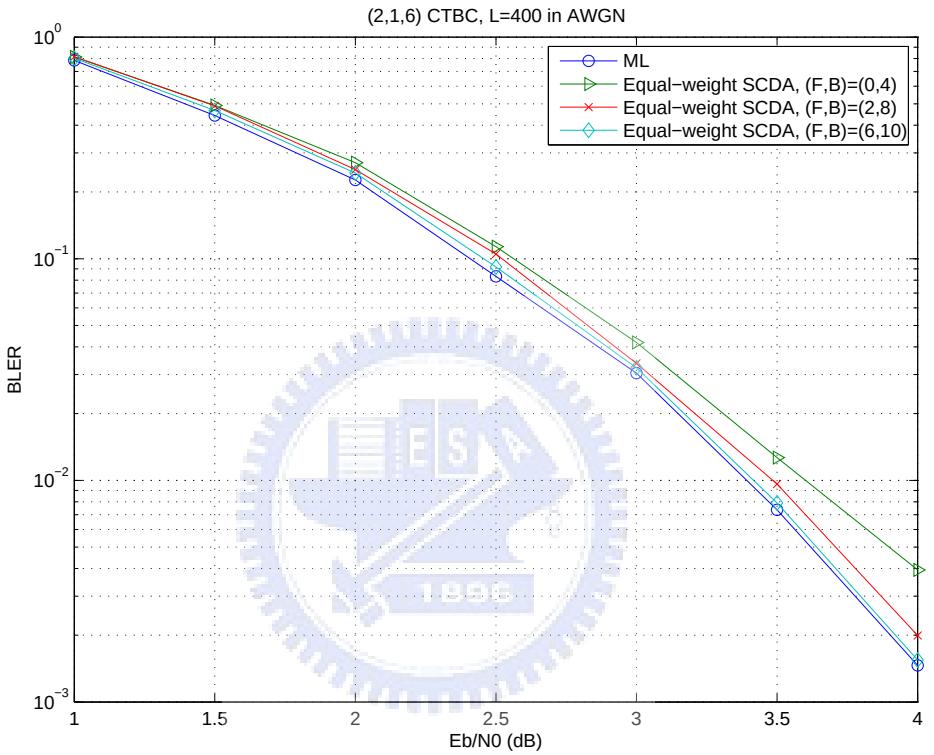


Figure 4.39: Block error rates (BLER) of the SCDA for $(2, 1, 6)$ CTBC in AWGN. F and B denote the forward and backward training window sizes, respectively. The length of information sequence is $L = 400$ and the weighted average window is $W = 20$.

Chapter 5

Conclusions

In this work, we proposed the shifting Viterbi algorithm and the shifting circular decoding algorithm with equal-weight and unequal-weight circular shifting methods. We found that the forward and backward training window sizes can be reduced by using the pre-decoding circular shifting. Furthermore, a sampled analysis of the required training window sizes required for near ML performances is also established for both BEC and AWGN based on a conjecture. A natural future work is thus to substantiate the conjecture that taking the new marginal probability of the shifted scalar in place of the original marginal probability can be an error bound.

Appendix A

Calculation of Unequal Weighting Coefficients

In the appendix, we list the equations for the calculation of unequal weighting coefficients according to code structure. These are provided simply for references.

A.1 $(3, 1, 6)$ CTBC

The noise power corresponding to $E_b/N_0 = 1$ dB is $\sigma^2 = 1.1915$. Assume the all-zero sequence is transmitted.

A.1.1 Coefficients corresponding to forward training window

1. *Probability for the all-zero path is eliminated at the 1st branch:*

   $\Pr(r_0 + r_1 + r_2 < 0) \approx 0.0563$

2. *Probability for the all-zero path is eliminated at the 2nd branch:*

   $\Pr(r_0 + r_3 + r_4 + r_5 < 0 \vee r_1 + r_2 + r_3 + r_4 + r_5 < 0) \approx 0.0335$

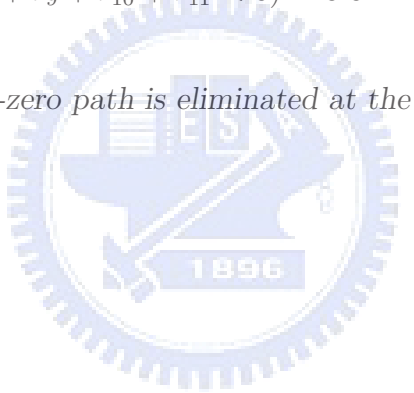
3. *Probability for the all-zero path is eliminated at the 3rd branch:*

$$\Pr(r_2 + r_3 + r_6 + r_7 + r_8 < 0 \vee r_0 + r_1 + r_3 + r_6 + r_7 + r_8 < 0$$
$$\vee r_0 + r_2 + r_4 + r_5 + r_6 + r_7 + r_8 < 0 \vee r_1 + r_4 + r_5 + r_6 + r_7 + r_8 < 0) \approx 0.0203$$

4. *Probability for the all-zero path is eliminated at the 4th branch:*

$$\Pr(r_0 + r_1 + r_5 + r_6 + r_9 + r_{10} + r_{11} < 0 \vee r_2 + r_5 + r_6 + r_9 + r_{10} + r_{11} < 0$$
$$\vee r_1 + r_3 + r_4 + r_6 + r_9 + r_{10} + r_{11} < 0 \vee r_0 + r_2 + r_3 + r_4 + r_6 + r_9 + r_{10} + r_{11} < 0$$
$$\vee r_0 + r_1 + r_2 + r_3 + r_5 + r_7 + r_8 + r_9 + r_{10} + r_{11} < 0$$
$$\vee r_3 + r_5 + r_7 + r_8 + r_9 + r_{10} + r_{11} < 0 \vee r_1 + r_2 + r_4 + r_7 + r_8 + r_9 + r_{10} + r_{11} < 0$$
$$\vee r_0 + r_4 + r_7 + r_8 + r_9 + r_{10} + r_{11} < 0) \approx 0.0124$$

5. *Probability for the all-zero path is eliminated at the 5th branch:*



$$\Pr(r_0 + r_1 + r_2 + r_3 + r_4 + r_8 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_3 + r_4 + r_8 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_1 + r_2 + r_5 + r_8 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_0 + r_5 + r_8 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_0 + r_1 + r_2 + r_4 + r_6 + r_7 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_2 + r_4 + r_6 + r_7 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_1 + r_3 + r_5 + r_6 + r_7 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_0 + r_2 + r_3 + r_5 + r_6 + r_7 + r_9 + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_2 + r_3 + r_4 + r_5 + r_6 + r_8 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_0 + r_1 + r_3 + r_4 + r_5 + r_6 + r_8 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_0 + r_2 + r_6 + r_8 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_1 + r_6 + r_8 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_4 + r_5 + r_7 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_0 + r_1 + r_2 + r_4 + r_5 + r_7 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_0 + r_3 + r_7 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0$$
$$\vee r_1 + r_2 + r_3 + r_7 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} < 0) \approx 0.0133$$

6. Probability for the all-zero path is eliminated at the 6th branch:

$$\Pr(r_6 + r_7 + r_{11} + r_{12} + r_{15} + r_{16} + r_{17} < 0) \approx 0.0077$$

A.1.2 Coefficients corresponding to backward training window

1. Probability for the all-zero path is eliminated at the 1st branch:

$$\Pr(r_{N-3} + r_{N-2} + r_{N-1} < 0) \approx 0.0563$$

2. *Probability for the all-zero path is eliminated at the 2nd branch:*

$$\Pr(r_{N-6}+r_{N-5}+r_{N-4}+r_{N-2}+r_{N-1}<0 \vee r_{N-6}+r_{N-5}+r_{N-4}+r_{N-3}<0) \approx 0.0335$$

3. *Probability for the all-zero path is eliminated at the 3rd branch:*

$$\begin{aligned}\Pr(&r_{N-9}+r_{N-8}+r_{N-7}+r_{N-5}+r_{N-4}+r_{N-3}+r_{N-2}+r_{N-1}<0\\ &\vee r_{N-9}+r_{N-8}+r_{N-7}+r_{N-5}+r_{N-4}<0 \vee r_{N-9}+r_{N-8}+r_{N-7}+r_{N-6}+r_{N-3}<0\\ &\vee r_{N-9}+r_{N-8}+r_{N-7}+r_{N-6}+r_{N-2}+r_{N-1}<0) \approx 0.0358\end{aligned}$$

4. *Probability for the all-zero path is eliminated at the 4th branch:*

$$\begin{aligned}\Pr(&r_{N-12}+r_{N-11}+r_{N-10}+r_{N-8}+r_{N-7}+r_{N-6}+r_{N-5}+r_{N-4}+r_{N-3}+r_{N-2}<0\\ &\vee r_{N-12}+r_{N-11}+r_{N-10}+r_{N-8}+r_{N-7}+r_{N-6}+r_{N-5}+r_{N-4}+r_{N-1}<0\\ &\vee r_{N-12}+r_{N-11}+r_{N-10}+r_{N-8}+r_{N-7}+r_{N-3}+r_{N-1}<0\\ &\vee r_{N-12}+r_{N-11}+r_{N-10}+r_{N-8}+r_{N-7}+r_{N-2}<0\\ &\vee r_{N-12}+r_{N-11}+r_{N-10}+r_{N-9}+r_{N-6}+r_{N-1}<0\\ &\vee r_{N-12}+r_{N-11}+r_{N-10}+r_{N-9}+r_{N-6}+r_{N-3}+r_{N-2}<0\\ &\vee r_{N-12}+r_{N-11}+r_{N-10}+r_{N-9}+r_{N-5}+r_{N-4}+r_{N-2}<0\\ &\vee r_{N-12}+r_{N-11}+r_{N-10}+r_{N-9}+r_{N-5}+r_{N-4}+r_{N-3}+r_{N-1}<0) \approx 0.0231\end{aligned}$$

5. *Probability for the all-zero path is eliminated at the 5th branch (Only dominant error conditions are listed here):*

$$\begin{aligned}\Pr(&r_{N-15}+r_{N-14}+r_{N-13}+r_{N-11}+r_{N-10}+r_{N-5}+r_{N-2}<0\\ &\vee r_{N-15}+r_{N-14}+r_{N-13}+r_{N-12}+r_{N-9}+r_{N-4}+r_{N-3}<0\\ &\vee r_{N-15}+r_{N-14}+r_{N-13}+r_{N-12}+r_{N-9}+r_{N-6}+r_{N-5}<0\\ &\vee r_{N-15}+r_{N-14}+r_{N-13}+r_{N-12}+r_{N-8}+r_{N-7}+r_{N-5}<0) \approx 0.0307\end{aligned}$$

6. *Probability for the all-zero path is eliminated at the 6th branch (Only dominant error conditions are listed here):*

$$\begin{aligned}\Pr(&r_{N-18}+r_{N-17}+r_{N-16}+r_{N-14}+r_{N-13}+r_{N-8}+r_{N-5}+r_{N-1}<0\\ &\vee r_{N-18}+r_{N-17}+r_{N-16}+r_{N-15}+r_{N-12}+r_{N-7}+r_{N-6}+r_{N-1}<0\\ &\vee r_{N-18}+r_{N-17}+r_{N-16}+r_{N-15}+r_{N-12}+r_{N-9}+r_{N-8}+r_{N-2}<0\\ &\vee r_{N-18}+r_{N-17}+r_{N-16}+r_{N-15}+r_{N-11}+r_{N-10}+r_{N-8}+r_{N-3}<0)\approx 0.0191\end{aligned}$$

In summary, for shifting circular decoding algorithm with $W = 12$, we let the normalized weighting coefficients be

$$\begin{aligned}&\left(\frac{0.0191}{0.0563},\frac{0.0307}{0.0563},\frac{0.0231}{0.0563},\frac{0.0358}{0.0563},\frac{0.0335}{0.0563},\frac{0.0563}{0.0563},\right.\\ &\quad\left.\frac{0.0563}{0.0563},\frac{0.0335}{0.0563},\frac{0.0203}{0.0563},\frac{0.0124}{0.0563},\frac{0.0133}{0.0563},\frac{0.0077}{0.0563}\right)\\ &\approx(\underbrace{0.34,0.55,0.41,0.64,0.60,1}_{\text{backward}},\underbrace{1,0.60,0.36,0.22,0.23,0.13}_{\text{forward}}).\end{aligned}$$

For shifting Viterbi algorithm with $W = 6$, we will use $(1, 0.60, 0.36, 0.22, 0.23, 0.13)$ as the normalized weighting coefficients.

A.2 $(2, 1, 6)$ CTBC

The noise power corresponding to $E_b/N_0 = 1$ dB is $\sigma^2 = 0.7943$. Assume the all-zero sequence is transmitted.

A.2.1 Coefficients corresponding to forward training window

1. *Probability for the all-zero path is eliminated at the 1st branch:*

$$\Pr(r_0 + r_1 < 0) \approx 0.0563$$

2. *Probability for the all-zero path is eliminated at the 2nd branch:*

$$\Pr(r_1 + r_2 + r_3 < 0 \vee r_0 + r_2 + r_3 < 0) \approx 0.0441$$

3. *Probability for the all-zero path is eliminated at the 3rd branch:*

$$\begin{aligned}\Pr(&r_3 + r_4 + r_5 < 0 \vee r_0 + r_1 + r_3 + r_4 + r_5 < 0 \\ &\vee r_1 + r_2 + r_4 + r_5 < 0 \vee r_0 + r_2 + r_4 + r_5 < 0) \approx 0.0260\end{aligned}$$

4. *Probability for the all-zero path is eliminated at the 4th branch:*

$$\begin{aligned}\Pr(&r_0 + r_1 + r_5 + r_6 + r_7 < 0 \vee r_5 + r_6 + r_7 < 0 \\ &\vee r_0 + r_2 + r_3 + r_5 + r_6 + r_7 < 0 \vee r_1 + r_2 + r_3 + r_5 + r_6 + r_7 < 0 \\ &\vee r_0 + r_1 + r_3 + r_4 + r_6 + r_7 < 0 \vee r_3 + r_4 + r_6 + r_7 < 0 \\ &\vee r_0 + r_2 + r_4 + r_6 + r_7 < 0 \vee r_1 + r_2 + r_4 + r_6 + r_7 < 0) \approx 0.0260\end{aligned}$$

5. *Probability for the all-zero path is eliminated at the 5th branch (Only dominant error conditions are listed here):*

$$\Pr(r_0 + r_7 + r_8 + r_9 < 0 \vee r_1 + r_7 + r_8 + r_9 < 0) \approx 0.0207$$

6. *Probability for the all-zero path is eliminated at the 5th branch (Only dominant error conditions are listed here):*

$$\Pr(r_3 + r_9 + r_{10} + r_{11} < 0) \approx 0.0124$$

A.2.2 Coefficients corresponding to backward training window

1. *Probability for the all-zero path is eliminated at the 1st branch:*

$$\Pr(r_{N-2} + r_{N-1} < 0) \approx 0.0563$$

2. *Probability for the all-zero path is eliminated at the 2nd branch:*

$$\Pr(r_{N-4} + r_{N-3} + r_{N-2} < 0 \vee r_{N-4} + r_{N-3} + r_{N-1} < 0) \approx 0.0441$$

3. *Probability for the all-zero path is eliminated at the 3rd branch:*

$$\begin{aligned}\Pr(&r_{N-6} + r_{N-5} + r_{N-4} + r_{N-2} + r_{N-1} < 0 \vee r_{N-6} + r_{N-5} + r_{N-4} < 0 \\ &\vee r_{N-6} + r_{N-5} + r_{N-3} + r_{N-1} < 0 \vee r_{N-6} + r_{N-5} + r_{N-3} + r_{N-2} < 0) \approx 0.0260\end{aligned}$$

4. *Probability for the all-zero path is eliminated at the 4th branch:*

$$\begin{aligned}\Pr(&r_{N-8} + r_{N-7} + r_{N-6} + r_{N-4} + r_{N-3} + r_{N-2} + r_{N-1} < 0 \\ &\vee r_{N-8} + r_{N-7} + r_{N-6} + r_{N-4} + r_{N-3} < 0 \vee r_{N-8} + r_{N-7} + r_{N-6} + r_{N-1} < 0 \\ &\vee r_{N-8} + r_{N-7} + r_{N-6} + r_{N-2} < 0 \vee r_{N-8} + r_{N-7} + r_{N-5} + r_{N-3} < 0 \\ &\vee r_{N-8} + r_{N-7} + r_{N-5} + r_{N-3} + r_{N-2} + r_{N-1} < 0 \\ &\vee r_{N-8} + r_{N-7} + r_{N-5} + r_{N-4} + r_{N-2} < 0 \\ &\vee r_{N-8} + r_{N-7} + r_{N-5} + r_{N-4} + r_{N-1} < 0) \approx 0.0372\end{aligned}$$

5. *Probability for the all-zero path is eliminated at the 5th branch (Only dominant error conditions are listed here):*

$$\Pr(r_{N-10} + r_{N-9} + r_{N-8} + r_{N-3} < 0 \vee r_{N-10} + r_{N-9} + r_{N-7} + r_{N-5} < 0) \approx 0.0231$$

6. *Probability for the all-zero path is eliminated at the 6th branch (Only dominant error conditions are listed here):*

$$\Pr(r_{N-12} + r_{N-11} + r_{N-10} + r_{N-5} < 0 \vee r_{N-12} + r_{N-11} + r_{N-9} + r_{N-7} < 0) \approx 0.0231$$

In summary, for shifting circular decoding algorithm with $W = 12$, we let the normalized weighting coefficients be

$$\begin{aligned}&\left(\frac{0.0231}{0.0563}, \frac{0.0231}{0.0563}, \frac{0.0372}{0.0563}, \frac{0.0260}{0.0563}, \frac{0.0441}{0.0563}, \frac{0.0563}{0.0563},\right.\\&\left.\quad\frac{0.0563}{0.0563}, \frac{0.0441}{0.0563}, \frac{0.0260}{0.0563}, \frac{0.0260}{0.0563}, \frac{0.0207}{0.0563}, \frac{0.0124}{0.0563}\right)\\&\approx (\underbrace{0.41, 0.41, 0.66, 0.46, 0.78, 1}_{\text{backward}}, \underbrace{1, 0.78, 0.46, 0.46, 0.36, 0.22}_{\text{forward}})\end{aligned}$$

For shifting Viterbi algorithm with $W = 6$, we will use $(1, 0.78, 0.46, 0.46, 0.36, 0.22)$ as the normalized weighting coefficients.

A.3 $(3, 1, 4)$ CTBC

The noise power corresponding to $E_b/N_0 = 1$ dB is $\sigma^2 = 1.1915$. Assume the all-zero sequence is transmitted.

A.3.1 Coefficients corresponding to forward training window

1. *Probability for the all-zero path is eliminated at the 1st branch:*

$$\Pr(r_0 + r_1 + r_2 < 0) \approx 0.0563$$

2. *Probability for the all-zero path is eliminated at the 2nd branch:*

$$\Pr(r_1 + r_2 + r_3 + r_4 + r_5 < 0 \vee r_0 + r_3 + r_4 + r_5 < 0) \approx 0.0335$$

3. *Probability for the all-zero path is eliminated at the 3rd branch:*

$$\begin{aligned}&\Pr(r_0 + r_2 + r_4 + r_5 + r_6 + r_7 + r_8 < 0 \vee r_1 + r_4 + r_5 + r_6 + r_7 + r_8 < 0\\&\quad\vee r_0 + r_1 + r_3 + r_6 + r_7 + r_8 < 0 \vee r_2 + r_3 + r_6 + r_7 + r_8 < 0) \approx 0.0203\end{aligned}$$

4. *Probability for the all-zero path is eliminated at the 4th branch (Only dominant error conditions are listed here):*

$$\Pr(r_4 + r_7 + r_8 + r_9 + r_{10} + r_{11} < 0 \vee r_1 + r_5 + r_6 + r_9 + r_{10} + r_{11} < 0) \approx 0.0231$$

A.3.2 Coefficients corresponding to backward training window

1. *Probability for the all-zero path is eliminated at the 1st branch:*

$$\Pr(r_{N-3} + r_{N-2} + r_{N-1} < 0) \approx 0.0563$$

2. *Probability for the all-zero path is eliminated at the 2nd branch:*

$$\begin{aligned}\Pr(&r_{N-6} + r_{N-5} + r_{N-4} + r_{N-2} + r_{N-1} < 0 \\ &\vee r_{N-6} + r_{N-5} + r_{N-4} + r_{N-3} < 0) \approx 0.0335\end{aligned}$$

3. *Probability for the all-zero path is eliminated at the 3rd branch:*

$$\begin{aligned}\Pr(&r_{N-9} + r_{N-8} + r_{N-7} + r_{N-5} + r_{N-4} + r_{N-3} + r_{N-1} < 0 \\ &\vee r_{N-9} + r_{N-8} + r_{N-7} + r_{N-5} + r_{N-4} + r_{N-2} < 0 \\ &\vee r_{N-9} + r_{N-8} + r_{N-7} + r_{N-6} + r_{N-3} + r_{N-2} < 0 \\ &\vee r_{N-9} + r_{N-8} + r_{N-7} + r_{N-6} + r_{N-1} < 0) \approx 0.0203\end{aligned}$$

4. *Probability for the all-zero path is eliminated at the 4th branch(Only dominant error conditions are listed here):*

$$\begin{aligned}\Pr(&r_{N-12} + r_{N-11} + r_{N-10} + r_{N-8} + r_{N-7} + r_{N-5} < 0 \\ &\vee r_{N-12} + r_{N-11} + r_{N-10} + r_{N-9} + r_{N-4} + r_{N-2} < 0) \approx 0.0231\end{aligned}$$

In summary, for shifting circular decoding algorithm with $W = 8$, we let the normalized weighting coefficients be

$$\left(\frac{0.0231}{0.0563}, \frac{0.0203}{0.0563}, \frac{0.0335}{0.0563}, \frac{0.0563}{0.0563}, \frac{0.0563}{0.0563}, \frac{0.0335}{0.0563}, \frac{0.0203}{0.0563}, \frac{0.0231}{0.0563}\right)$$
$$\approx (\underbrace{0.41, 0.36, 0.60, 1}_{\text{backward}}, \underbrace{1, 0.60, 0.36, 0.41}_{\text{forward}}).$$

For shifting Viterbi algorithm with $W = 4$, we will use $(1, 0.60, 0.36, 0.41)$ as the normalized weighting coefficients.

References

[1] R. V. Cox and C. E. W. Sundberg, "An efficient adaptive circular Viterbi algorithm for decoding generalized tailbiting convolutional codes," *IEEE Trans. Veh. Techol.*, vol.43, no. 11, pp. 57-68, February 1994.

[2] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, pp. 4:37-47, 1955.

[3] Y. S. Han, T.-Y. Wu, H.-T. Pai, P.-N. Chen and S. -L. Shieh, "Priority-first search decoding for convolutional tail-biting codes," *2008 International Symposium on Information Theory and its Applications*, Auckland, New Zealand, December 7-10 2008.

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

[5] M. Handlery, R. Johannesson and V. V. Zyablov, "Boosting the error performance of suboptimal tailbiting decoders," *IEEE Trans. Commun.*, vol.51, no. 9, pp. 1485-1491, September 2003.

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

[7] S. Lin and D. J. Costello, Jr., *Error Control Coding*, second edition, Prentice Hall, 1983.

[8] H. H. Ma and J. K. Wolf, "On tail biting convolutional codes," *IEEE Trans. Commun.*, vol.34, no. 2, pp. 104-111, February 1986.

[9] P. Shankar, P. N. A. Kumar, K. Sasidharan, B. S. Rajan and A. S. Madhu, "Efficient convergent maximum likelihood decoding on tail-biting," available at http://arxiv.org/abs/cs.IT/0601023.

[10] R. Y. Shao, S. Lin and M. P. C. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Trans. Commun.*, vol.51, no. 10, pp. 1658-1665, October 2003.

[11] W. Sung, "Minimum decoding trellis lengths for tail-biting convolutional codes," *Electronics Letters*, vol.36, no. 7, pp. 643-645, March 2000.

[12] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. 13, pp. 260-269, April 1967.

[13] Q. Wang and V. K. Bhargava, "An efficient maximum likelihood decoding algorithm for generalized tail biting convolutional codes including quasicyclic codes," *IEEE Trans. Commun.*, vol.37, no. 8, pp. 875-879, August 1989.

[14] Y. P. E. Wang and R. Ramesh, "To bite or not to bite - a study of tail bits versus tail-biting," *IEEE Personal, Indoor and Mobile Radio Communications*, vol. 2, pp. 317-321, October 1996.