# Deterministic Interleaver Design for Turbo Codes

Bohulu Kwame Ackah

平成 30 年 1 月 29 日

# Abstract

Turbo codes are known to be one of the forward-error correcting codes that closely approach the channel capacity for Additive White Gaussian Noise (AWGN) channels and this characteristic is attributed to the use of the interleaver in its design. Deterministic interleavers use algorithms for the interleaving process and when compared to random interleavers they require less memory, and due to their easier hardware implementation high-speed decoding methods such as parallel decoding is possible. Due to these reasons they are widely used in many practical applications. However, the deterministic interleavers that are currently known are outperformed by random interleavers for the case of long frame sizes.

In this research, we search amongst the deterministic interleavers for an easy to design linear interleaver which has an excellent performance for the case of long frame sizes when compared to the random interleaver. First, we analyze why for high Signal to Noise Ratio (SNR) and large frame sizes the Bit Error Rate performance of linear interleavers is dominated by a high error floor. We then propose a new interleaver, the multi-shift interleaver (MSI) which periodically changes the coefficient of the linear interleaver by a fixed constant and shifts all positions according to this coefficient to mitigate the occurrence of the error event of the weight 2 which is the cause of the error floor. In addition, the constant constraint search method is used to find the optimum parameters of the MSI in order to efficiently search a MSI with good performance. Via simulation we confirm that the proposed interleaver has a better performance than the linear interleaver for medium and long frame sizes.

# 目 次

# 第1章　Introduction

The construction of a turbo code is usually done by the parallel concatenation of two convolutional codes via an interleaver. The good BER performance of turbo codes at low SNR is attributed to the interleaver, which effectively thins the distance spectrum of the turbo code [3]. Due to its importance, extensive research has been conducted on interleavers for turbo codes. Interleavers for turbo codes are generally grouped into random and deterministic interleavers. The most common random interleaver can be achieved by rearranging elements in a set in pseudo-random fashion. This interleaver was used in [4] and was shown to achieve performance very close to the Shannon limit for long frame sizes.

The disadvantage associated with random interleavers arises from the necessity of storing interleaver tables in both the encoder and decoder. For applications where large interleaver sizes are required, the memory requirements to store the interleaver tables make the use of these interleavers undesirable. Deterministic interleavers are a solution to necessity of interleaver tables as the interleaving is done via algorithm. Deterministic interleavers are being used in many applications, most notably the Quadratic Permutation Polynomial (QPP) Interleaver [5] which is used in 4G LTE applications.

The most basic deterministic interleavers are the linear and block interleavers [2] For long frame sizes, linear interleavers perform worse than random interleavers due to prescence of a high error floor. Many other deterministic interleavers have been proposed, including the quadratic interleaver [2] which performs well but not as good as the random interleavers for long interleaver frame sizes. In this research, we attempt to design a deterministic interleaver which performs as well as the random interleaver especially for long interleaver frame sizes. Finding such a deterministic will greatly increase the attractiveness of turbo codes for use in future wireless networks.

# 第2章　Review of Turbo Codes

In this Chapter, we review the encoding and decoding process for turbo codes for BPSK modulation and transmission over the AWGN channel. We also introduce the concept of $a{-}\tau$ - seperated weight 2 errors and their importance in relation to Recursive Systematic (RSC) Encoders.

## 2.1 Encoding

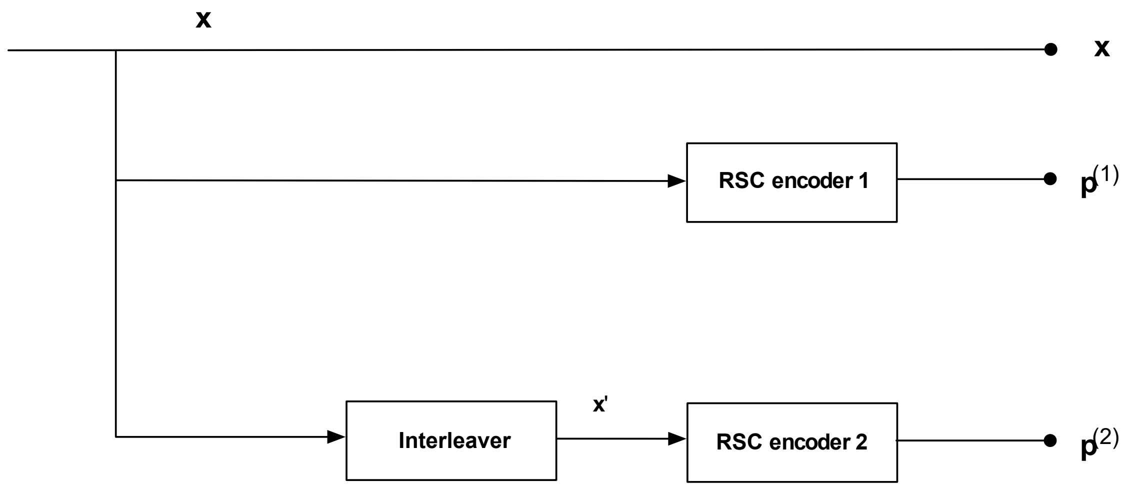

図 2.1: Turbo Encoder

The system diagram for the turbo encoder is shown in figure(2.1). It is made up of identical Recursive Systematic Convolutional (RSC) encoders which are connected in parallel via an interleaver. The RSC encoders have constraint length $K$ and output $n$ bits for every $k$ bits input at time $t$. From here onward we refer to the RSC encoders as component encoders (CE). The generator matrix of the component codes are written in the form $[1\ \frac{F(D)}{H(D)}]$ or simply as $[\frac{F(D)}{H(D)}]$ where the numerator and the denomenator represent the feedfoward and feedback connections of the component encoder. The $''1''$ represents the information (systematic) bits fed into the encoder.

The generator matrix for the one shown in Figure 2.2 is $[\frac{1+D^2}{1+D+D^2}]$ .The encoding process is as follows.

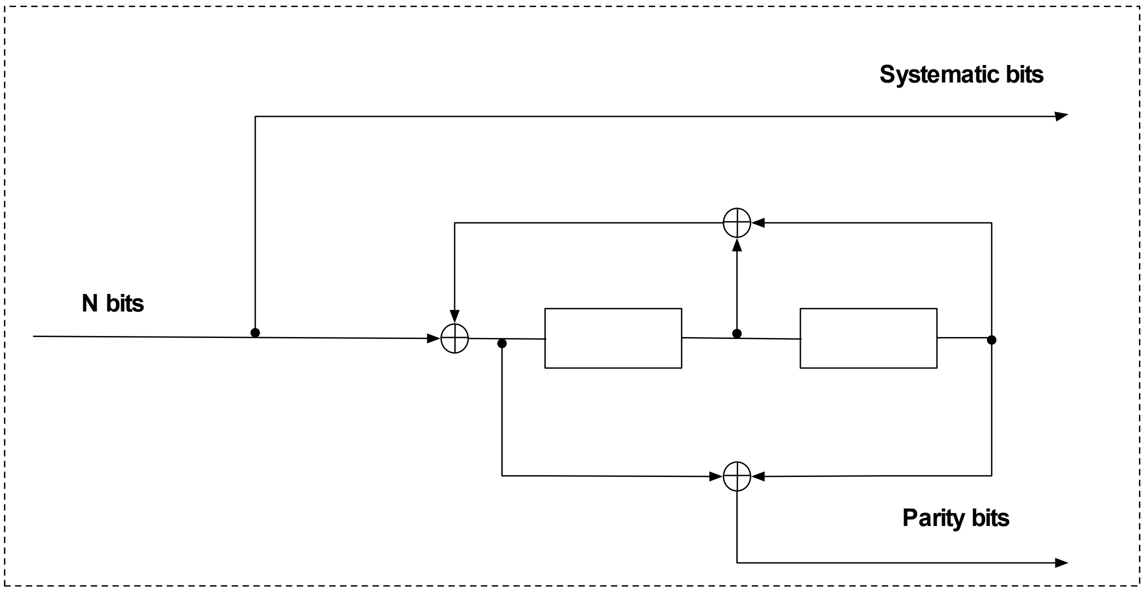

図 2.2: $[\frac{1+D^2}{1+D+D^2}]$ RSC Encoder

In our encoding scheme, we terminate the CE1 and leave CE2 open. To acheive this, an information sequence $\mathbf{x} = \{x_0, x_1, ..., x_{N-M-1}\}$ $M = K - 1$ of length $N - m$ is fed into the turbo encoder. This is fed directly into CE1 (assumed to begin in the all-zero state) and produces the upper parity sequence $\mathbf{p}^{(1)} = \{p_0^{(1)}, p_1^{(1)}, ..., p_{N-M-1}^{(1)}\}$ also of length $N - M$. Since it is desired to return CE1 to the all-zero state, $m$ extras tail bits are fed into CE1 which brings the total length of $p^{(1)}$ to $N$. The tail bits are also added to $\mathbf{x}$ transforming it into a vector of length $N$. The input to the CE2 (also assumed to begin in the all-zero state) is $\mathbf{x}' = \Pi(\mathbf{x}) = \{x'_0, x'_1, ..., x'_{N-1}\}$. This produce the lower parity sequence $\mathbf{p}^{(2)} = \{p_0^{(2)}, p_1^{(2)}, ..., p_{N-1}^{(2)}, ..., p_{N-1}^{(2)}\}$ which has total length $N$ since we wish to keep the trellis of CE2 open. $\mathbf{x}$ (along with the extra m tail-bits), $\mathbf{p}^{(1)}$ and $\mathbf{p}^{(2)}$ are multiplexed, BPSK modulated and transmitted over the AWGN channel. The turbo codeword generated

$$\mathbf{c} = \{x_0, p_0^{(1)}, p_0^{(2)}, ..., x_{N-1}, p_{N-1}^{(1)}, p_{N-1}^{(2)}\}$$

has length $3N$ and the turbo encoder has rate $R_c = \frac{N-M}{3N}$

## 2.2 Turbo Decoding

The turbo code transmitted over the AWGN channel is received by the turbo decoder as $\mathbf{y} = \{\mathbf{y}^x, \mathbf{y}^{p^{(1)}}, \mathbf{y}^{p^{(2)}}\}$ of length $3N$, where $\mathbf{y}^x, \mathbf{y}^{p^{(1)}}, \mathbf{y}^{p^{(2)}}$ correspond to the systematic, upper and lower parity sequence respectively.

$$\mathbf{y}^x = \{y_0^x, y_1^x, ..., y_{N-1}^x\}$$

$$\mathbf{y}^{p^{(1)}} = \{y_0^{p^{(1)}}, y_1^{p^{(1)}}, ..., y_{N-1}^{p^{(1)}}\}$$

$$\mathbf{y}^{p^{(2)}} = \{y_0^{p^{(2)}}, y_1^{p^{(2)}}, ..., y_{N-1}^{p^{(2)}}\}$$

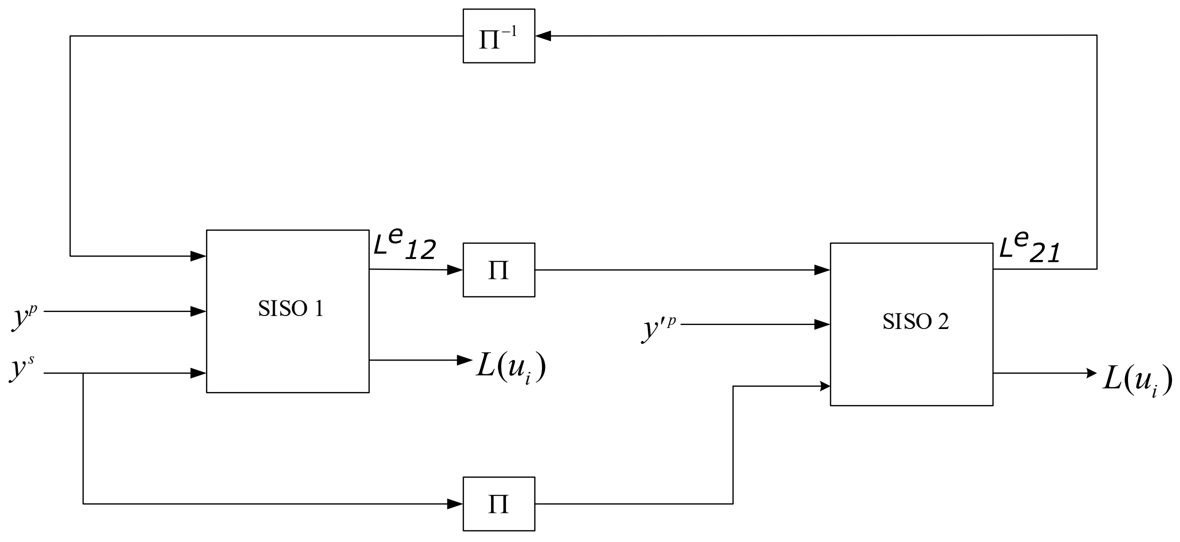

図 2.3: Turbo Decoder

The system diagram for the turbo decoder is shown in figure (2.3). It is made up of 2 Soft Input Soft Output (SISO) decoders (one for each encoder). The decoding process is carried out using turbo decoding algorithm which is based of the use of the BCJR algorithm or its variation. The Max-Log-MAP algorithm is used for its improved numerical stability and simplification of the calculations involved. The algorithm is used to calculate the a posteriori Log-Likelihood Ratio (LLR), $L(x_i|\mathbf{y}) = \ln \frac{P(x_i=1|\mathbf{y})}{P(x_i=0|\mathbf{y})}$ of the bits received. This also requires the calculation of state transition, feedfoward and feedback probabilities which we represent by the symbols $\gamma, \alpha, \beta$ respectively[from BCJR]. We represent the previous trellis state and current trellis state by $\sigma'$ and $\sigma$

respectively.

$$\begin{aligned} \gamma_i(\sigma',\sigma) &= \frac{x_i L_a(x_i)}{2} + \frac{L_c}{2}\sum_{l=1}^{n} c_{i,l}y_{i,l} \quad , \\ L_a(x_i) &= \frac{P(x_i=1)}{P(x_i=0)} \quad , Lc = 4R_c\frac{E_b}{N_0} \\ \alpha_i(\sigma) &= \max_{\sigma'}[\alpha_{i-1}(\sigma') + \gamma_i(\sigma',\sigma)] \\ \beta_i(\sigma') &= \max_{\sigma}[\beta_i(\sigma) + \gamma_i(\sigma',\sigma)] \end{aligned} \tag{2.1}$$

$c_i = \{x_i, p_i^v\}$ $, v = 1, 2$ and the initial values for $\alpha$ and $\beta$ are

$$\alpha_0(\sigma) = \begin{cases} 0, & \sigma = 0 \\ -\infty, & \sigma \neq 0 \end{cases}$$

$$\beta_N(\sigma) = \begin{cases} 0, & \sigma = 0 \\ -\infty, & \sigma \neq 0 \end{cases}$$

$L(x_i)$ is then calculated by using equation (2.2)

$$\begin{aligned} L(x_i) = &\max_{R_1}[\alpha_{i-1}(\sigma') + \gamma_i(\sigma',\sigma) + \beta_i(\sigma)] - \\ &\max_{R_0}[\alpha_{i-1}(\sigma') + \gamma_i(\sigma',\sigma) + \beta_i(\sigma)] \end{aligned} \tag{2.2}$$

where $R_0, R_1$ are the subset of transitions caused by "0" and "1" respectively.

The decoding process is as follows. The input to the SISO1 is $\mathbf{y}^x, \mathbf{y}^{p^{(1)}}$ and $\mathbf{L}_a = \{L_a(x_0), L_a(x_1), ..., L_a(x_L)\}$. For the first iteration, it is assumed that the input information bits have equal probability and $\mathbf{L_a}$ is an all-zero vector. These are used to calculate $\gamma, \alpha, \beta$ using (2.1) and finally $L(\mathbf{x})$ using (2.2) and $\mathbf{L_e^{(1)}}$ is obtained by subtracting $L_c\mathbf{y}^x$ from each element in $L(\mathbf{x})$ . $\mathbf{L_e^{(1)}}$ is then interleaved and fed into SISO2 as the value for $\mathbf{L_a}$ along with an interleaved version of $\mathbf{y}^x$ and $\mathbf{y}^{p^{(2)}}$ which correspond to the interleaved systematic bits and the lower parity bits. These are used to calculate $\gamma, \alpha, \beta, L(\mathbf{x})$ and finally the extrinsic LLR values of the second component decoder, $\mathbf{L_e^{(2)}}$. $\mathbf{L_e^{(2)}}$is deinterleaved、and fedback into the first component encoder as the new $\mathbf{L_a}$ value for SISO1.

The process is either repeated for a predetermined number of times, or untill a certain condition is met. At the final iteration $L(\mathbf{x})$ (from the second component decoder) is deinterleaved and used to estimate the values of $\mathbf{x}$.

It should be noted that, since CE2 was left open, the final state at time $N-1$ could end up being any of the $2^m$ states with probability $\frac{1}{2^m}$. Therefore in SISO2, the initial value of $\beta$, $\beta_L(\sigma) = -ln(2^m)$.

## 2.3 RSC encoders and $\tau$-seperated weight error events

RSC encoders are the component code of choice for turbo codes. They are characterized by their cycle length ($\tau$) which is defined as the length of the cycle of the parity output of the encoder when the input $\mathbf{x}$ is $[1, 0, 0, 0, ....]$[5]. For example, the RSC encoder in figure (2.2)has a parity output $\mathbf{y}$ of $[1, 1, 1, 0, 1, 1, 0, 1, 1, 0, ...]$ for the previously mentioned input. As can be observed, the cycle is $[1, 1, 0]$ and the cycle length $\tau = 3$. With the knowledge of the cycle and the cycle length $\tau$ of the RSC encoder we wish to explore the effect of weight-$2m$ inputs where the pair of "1" bits are seperated by $\tau - 1$ ' '0' ' bits. We shall refer to to these inputs as $\tau$-seperated weight-$2m$ input error events (or simply as $\tau$ weight-$2m$ errors for simplicity sake ), where $m = 1, 2$. In general, the minimum codeword weight associated with weight 2 inputs is known as the effective free distance ($d_{eff}$)[5] Figure(2.4) shows the effect of $\tau$ weight-2 errors on the codeword weight.

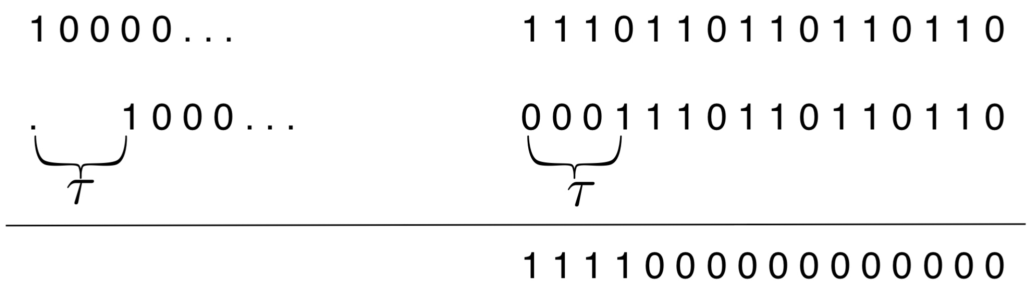

図 2.4: effect of $\tau$ weight-2 errors

The encoder used is the RSC encoder in figure (2.2) and the length $N$ of this vector is assumed to be 16. The error vector may then be written as $[1, \mathbf{0}_{\tau-1}1, \mathbf{0}_{N-\tau+1}]$ (where $\mathbf{0}_z$ is a zero vector of length $z$ ). This is equivalent to the modulo 2 addition of $\mathbf{x}$ and a version of $\mathbf{x}$ shifted by $\tau$ ,$\mathbf{x}_{\Delta\tau}$. As can be seen from figure(2.4) these inputs produce parity outputs $\mathbf{y}$ and $\mathbf{y}_{\Delta\tau}$ (version of $\mathbf{y}$ shifted by $\tau$). Modulo 2 addition of $\mathbf{y}$ and $\mathbf{y}_\tau$ result in a low-weight parity output, which inturn results in a low-weight parity codeword. It can easily be shown that a low-weight parity ouput will be produced by $\tau$ weight-2 errors irrespective of position of the "1" bits .

From the above example, we see that $\tau$ weight-$2m$ errors have the potential to produce low weight codeword with high multiplicity if they are present in both component encoders of the turbo encoder. A typical $a - \tau$ weight-$2m$ error event in relation to turbo codes is shown in Figure 2.5.

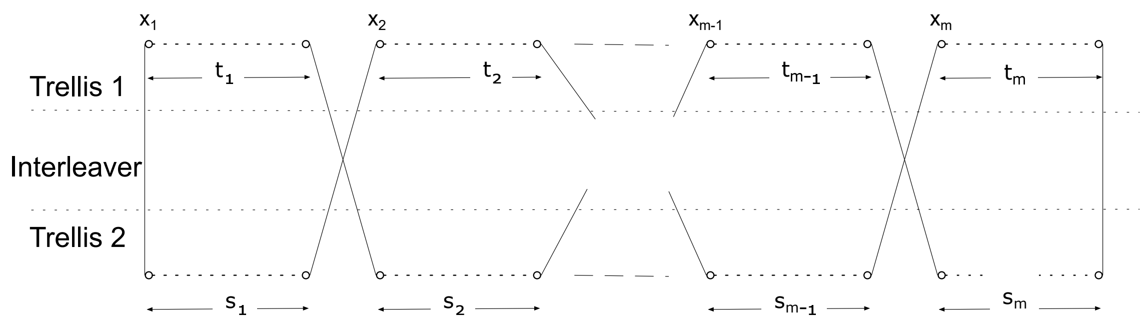

図 2.5: $a - \tau$ weight-$2m$ error event

$t_i$ and $s_i$ represent the distance between the $m$ “1” bit pairs before and after interleaving repectively. Also, $s$ and $t$ are multiples of $\tau$.

# 第3章 Interleaver Design for Turbo Codes

## 3.1 Turbo Codes with Linear Interleavers

We approach the design of our interleaver by breaking the $a-\tau$ weight-$2m$ error event into smaller subclasses and getting rid of as many errors as possible whiles increasing $d_{eff}$ before moving on to the next subclass. In our analysis we consider $a-\tau$ weight-$2m$ errors that originally have a length of $N-M$ and maintain weight $2m$ after $M$ tails bits are added to return CE1 to the all-zero state.

### 3.1.1 Linear Interleaver Design for $a-\tau$ weight-2 errors

The linear interleaver is a simple deterministic interleaver based on the circular shifting and its index mapping function is given by

$$\Pi_{\mathfrak{L}_N}(i) \equiv Di \mod N \quad 0 \le i < N,\ gcd(N,D)=1$$

$D$ is reffered to as the interleaver depth. The design of the linear interleaver is essentially picking a suitable value for $D$ for a given interleaver length $N$ and in [2] considerations for choosing a suitable value for $D$ is introduced.

We begin with the simplest subclass, which is the $a-\tau$ weight-2 error. The error is shown in Figure 3.1 . We attempt to design linear interleavers to get rid of such errors. Assuming the error event at the first encoder begins at $i$ and the “1” bit pairs are seperated by a distance of $t$,

$$\begin{aligned} s =& D(i+t) - D(i) \mod N \\ =& Dt \mod N \end{aligned} \tag{3.1}$$

in the case where $t = a\tau$ and $s = b\tau$, a low weight codeword may be produced. To prevent this error event from occuring we choose $D$ that produces the largest value of $\min t + s$. In the case where there are multiple values of $D$ that satisfy the above

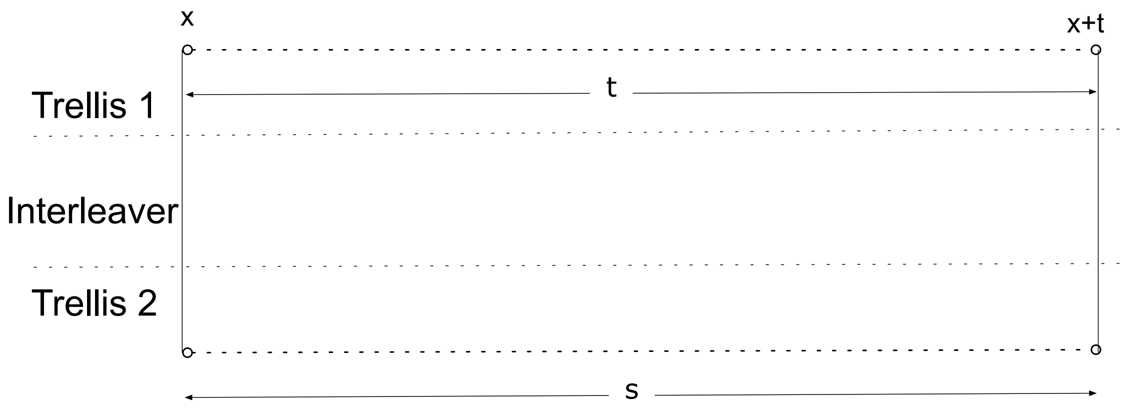

図 3.1: $a - \tau$ weight-2 error event

condition, we calculate the codeweight word associated with $a - \tau$ seperated weight 2 error events of length $N - M$ and choose the value of D that produces the largest value of effective free distance. The procedure for choosing the best linear interleaver is shown below.

[1]. For $D \in (3, N/2),\ \ gcd(D, N) = 1$, calculate $s$ and $-s$ using (3.1) for all values of $t = a\tau$ and $t = -a\tau$ respectively.

[2]. For all valid pairs, sum all $(t, s)$ pairs and record $\min(a + b)$

[3]. After this process is repeated for all values of $D$, choose the value of $D$ that produces the largest value of $\min(t + s)$

[4]. In the case where more than one value of $D$ satisfies the condition we calculate the $d_{eff}$ for each value of $D$ and select the value of $D$ with the largest value of $d_{eff}$

For an interleaver size of $N = 2^8$, and the 5/7 RSC compenent encoder we use the above procedure to find the best value of $D$ which are $D = 13$ and $D = 121$. The results are shown in Table 4.1. The corresponding values of $t$, $s$, $d_{eff}$ and the multiplicity $N_{free}$. In the same table we compare the best values of $D$ with other $D$ values close $\sqrt{N}$. We see that the weight produced by $D = 13$ and $D = 121$ have the largest value of $d_{eff}$ and therefore should bave a better decoding performance.

The Bit Error Rate (BER) upper bound for Turbo is derived in [6] and is given by (3.2).

$$P_e \approx \frac{1}{2} \sum_{w_c} D_{w_c} erfc\left(\sqrt{w_c \frac{R_c E_b}{N_o}}\right) \tag{3.2}$$

| $D$ | 13 | 121 | 17 | 23 | 21 |
|---|---|---|---|---|---|
| $t$ | 21 | 15 | 15 | 12 | 21 |
| $s$ | 17 | 23 | 1 | 20 | 4 |
| $d_{eff}$ | 30 | 30 | 15 | 26 | 15 |
| $N_{free}$ | 1 | 1 | 2 | 1 | 2 |

表 3.1: Best value of $D$ for $N = 256$ linear interleaver and 5/7 component encoder.

where

$$D_{w_c} \triangleq \sum_{w_x+w_p=w_c} \frac{w_x}{N} A_{w_x,w_p}$$

$w_x$ is the weight of the input sequence, $w_p$ is the weight of the parity sequence,$R_c$ is the rate of the turbo code , $w_c$ is the weight of the turbo codeword and $A_{w_x,w_p}$ is the multiplicity of $w_c$. Using (3.2), we estimate the BER performance of the turbo codes constructed using linear interleavers. From Table 4.1 we plot the BER curve for $D = 121$, $D = 23$ and $D = 21$. We also run simulations for $D = 121$ and compare it to its BER curve as shhown in figure 3.2.

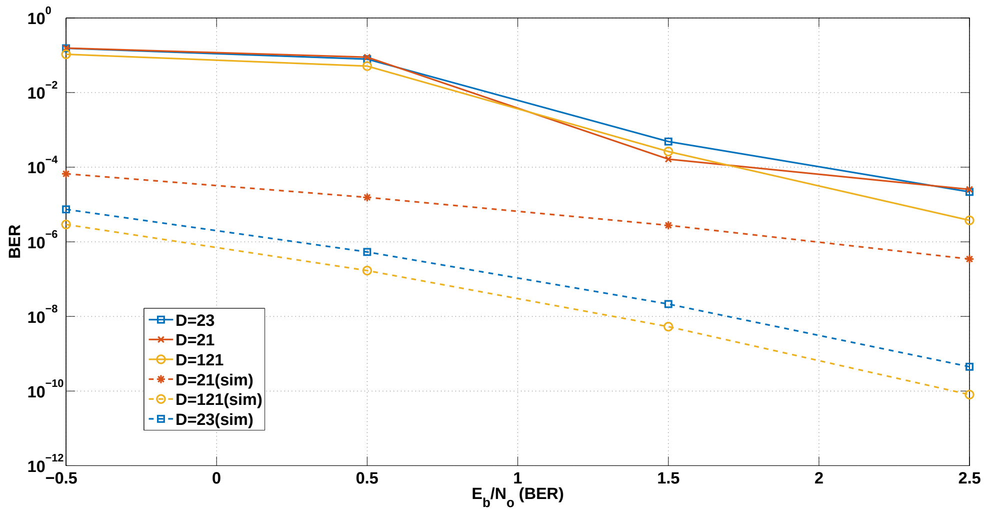

図 3.2: Theoretical and simulation results for $D = 21$, $D = 23$ and $D = 121$

We realize that the estimated BER curve fails to properly estimate the simulated BER performance. This suggests that there exists codeword with weight lower $d_{eff}$ caused by higher weight error events. We therefore proceed to the next error subclass which is the $a-\tau$ weight 4 error subclass.

### 3.1.2 Linear Interleavers and $a-\tau$ weight-4 errors

A typical $a-\tau$ weight-4 error event is shown in Figure 3.3. It is made up of 2 “1” bit pairs with the paired bits seperated by a distances of $t_1$ and $t_2$ respectively. After interleaving the distances are mapped to $s_1$ and $s_2$ respectively. We begin with the

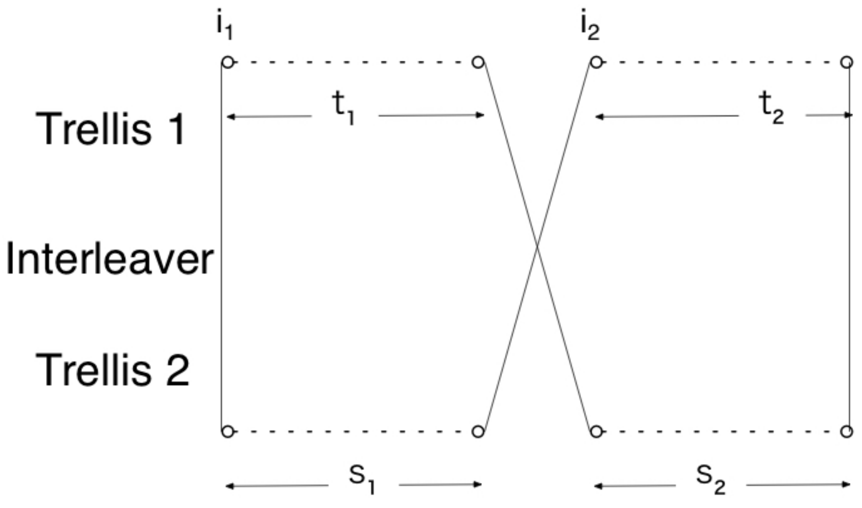

図 3.3: $a-\tau$ weight-4 error event

case where $t_1 = t_2 = \tau$. For linear interleavers, there exists an input distance $t = v$ or $t = w = -v \mod N$ such that the output distance $s = \tau$. The relationship between $D$ and $v$ is a result of the following linear congruence relationship

$$Dv \equiv \tau \mod N \tag{3.3}$$

Given the value of $D$ the value of $v$ can easily be solved for. If $t_1$ and $t_2$ are seperated by a distance $dp = v - \tau$ or $dp = w - \tau$, $s_1 = s_2 = \tau$ as shown in 3.4a and 3.4b. This output is valid for all $N-(v-\tau)$ times when the position of the “1” bits are shifted to the right causing the codeword weight produced to have a high multiplicity. However, there exists cases where $s_1 \neq s_2 \neq \tau$ when $dp = w - \tau$ as shown in Figure 3.4c where the value of $D = 121$ and $N = 256$.

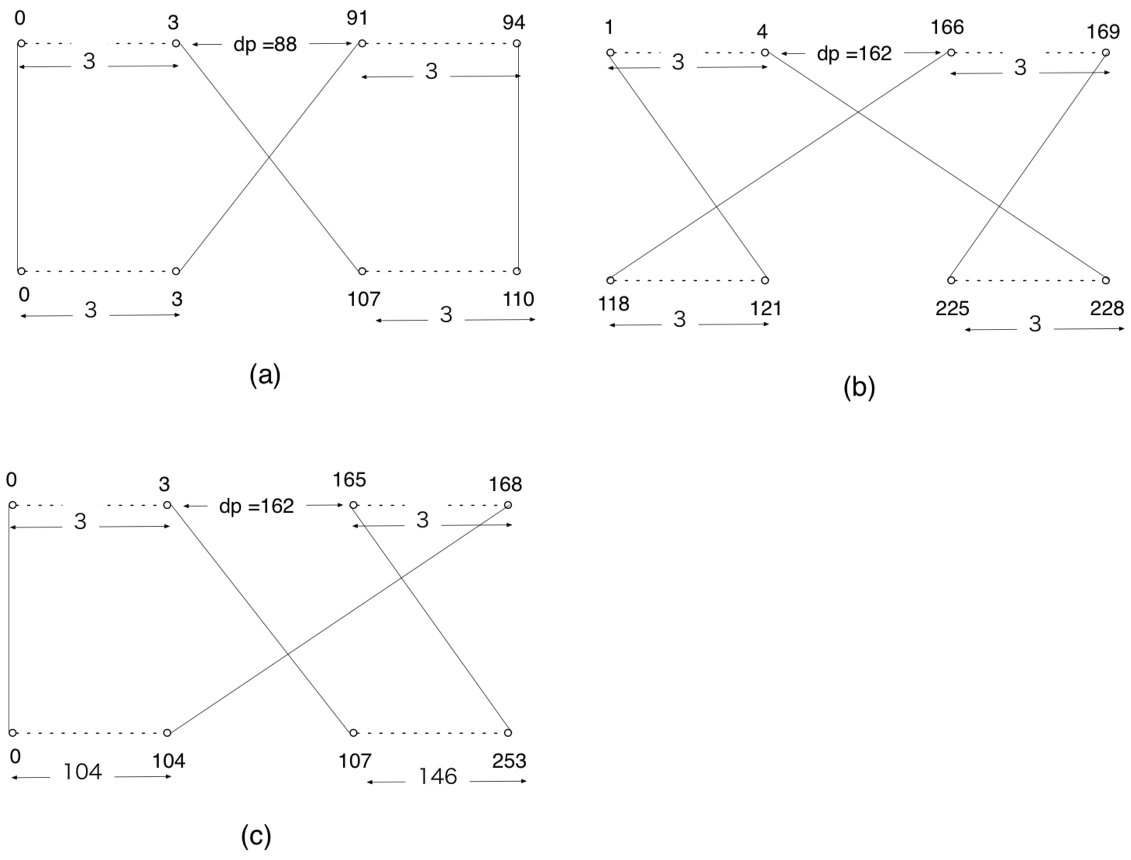

図 3.4: $a - \tau$ weight-4 error event mapping, $D = 121$, $N = 256$

Also the lowest codeword weight produced has a value of 20 and due to the large multiplicity, it dominated the BER performance of the turbo. The upper bound BER curve due to $\tau$ - seperated weight 4 errors is shown in figure **??** and compared to the simulation results obtained when $D = 121$ and $N = 256$. It is observed that this curve gives a tighter upper bound on the BER performance of the turbo code. These $\tau$ - seperated weight 4 errors present a big problem for linear interleaver design. This is because as the value of $N$ is increased it is possible to acheive large values for $d_{eff}$ via the method in Section 3.1.1. However the minimum codeword weight resulting from $\tau$ - seperated weight 4 errors remains constant for a given component code and increases in multiplicity as $N$ increases [2].

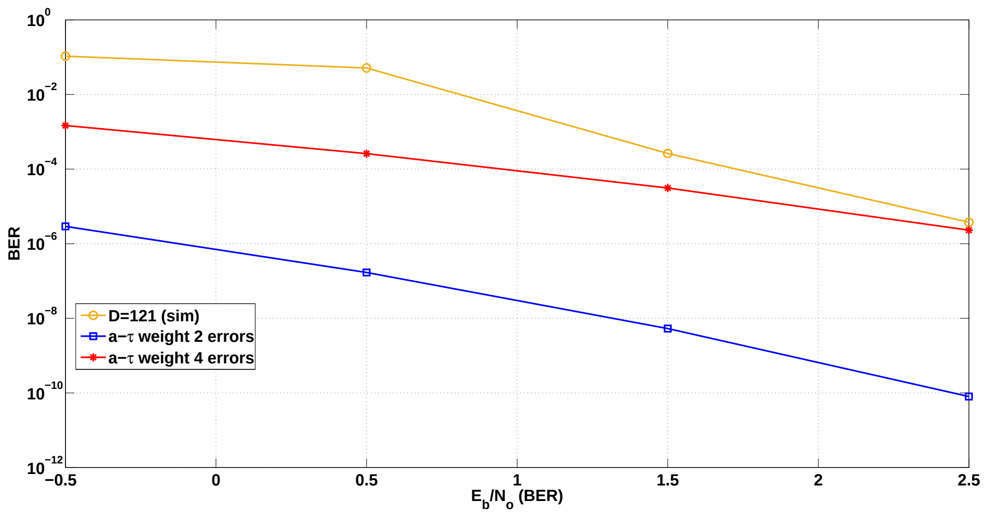

図 3.5: Simulation Results vs weight -2 and weight -4 Bounds, $D = 121$, $N = 256$

## 3.2 Turbo Codes with Multi-Shift Interleaver

We propose a method to solve the $\tau$ - seperated weight 4 errors dominating the BER performance of the linear which leads to a new interleaver design. We first present a sequential representation of the linear interleaver. We then present adjustments to the linear interleaver which get rids of the dominance of the $\tau$ - seperated weight 4 errors. A method for selecting good interleavers for a given interleaver length $N$ and component code.

### 3.2.1 Multi-Shift Interleaver Design via Sequential Representation of Linear Interleavers

The process of linear interleaving is basically shifting the positions of elements in a set by a certain factor $D \mod N$ and then use the following index mapping function.

$$\Pi_{L_N} : x \mapsto p_x \quad 0 \leq x < N$$

The algorithm below can be used to describe the process of linear interleaving

1. $p_0 = 0$
2. $p_i = (p_{i-1} + D) \mod N$,

$0 < i < N$, $D$ *is an odd integer*

Keeping $D$ constant along with the fact that $gcd(N, D) = 1$ causes a linear congruence constraint shown in 3.3 which in turn makes the linear interleaver succeptible to $\tau$ - seperated weight 4 errors as explained in Section 3.1.2.

One possible method for getting rid of the linear congruence constraint is to change the value of $D$ for every position shift. Based on this idea, the algorithm for a new interleaver is given by the algorithm below.

1. $p_0 = 0$
2. $p_i = p_{i-1} + d_{((i-1) \mod V)} \mod N, \quad 0 < i < N, \quad d_0 \textit{ is an odd integer}$

In this algorithm we introduce two new variable, the cycle set $\mathbb{D}$ and step size $\Delta s$. The cycle set $\mathbb{D} = \{d_0, d_1, ..., d_{V-1}\}, \ V = N/\Delta s$. The elements of $\mathbb{D}$ are calculated using (3.4)

$$d_i = d_{i-1} + \Delta s, \ \ d_0 = D \tag{3.4}$$

For $N = 2^r, \ r \in \{1, 2, ...\}$ we set $\Delta s = 2^q, \ q \in \{2, 3, ..., r-1\}$. As can be seen from the algorithm, the element positions as well as the values of the $\mathbb{D}$ are shifted. We therefore call this interleaver the multi-shift interleaver.It is denoted by the symbol $\Pi_{M|N:(d_0,\Delta s)}$ .

**Example 3.2.1** *: For $N = 32$, the original set is $\{0, 1, 2, ..., 31\}$. The range of values for $\Delta s = \{2, 4, 8, 16\}$. If we set the value of $\Delta s = 4$ and $d_0 = 5$, the cycle set* $\mathbf{d} = \{5, 9, 13, 17, 21, 25, 29, 1\}$,
$\mathbf{p} = \{0, 5, 14, 27, 12, 1, 26, 23, 24, 29, 6, 19, 4, 25, 18, 15, 16, 21$
$, 30, 11, 28, 17, 10, 7, 8, 13, 22, 3, 20, 9, 2, 31\}$ *and the interleaved set is* $\{0, 5, 30, 27, 12, 1, 10, 23, 24, 2$
$, 16, 21, 14, 11, 28, 17, 26, 7, 8, 13, 6, 3, 20, 9, 18, 31\}$

**Example 3.2.2** *: For $N = 32$, the original set is $[0, 1, 2, ..., 31]$. The range of values for $\Delta s = \{2, 4, 8, 16\}$. If we set the value of $\Delta s = 8$ and $d_0 = 5$, the cycle set* $\mathbf{d} = \{5, 13, 21, 29\}$,
$\mathbf{p} = \{0, 5, 18, 7, 4, 9, 22, 11, 8, 13, 26, 15, 12, 17, 30, 19, 16, 16, 21$
$, 2, 23, 20, 25, 6, 27, 24, 29, 10, 31, 28, 1, 14, 3\}$ *and the interleaved set is* $\{0, 29, 18, 31, 4, 1, 22, 3, 8, 5,$
$, 11, 16, 13, 2, 15, 20, 17, 6, 19, 24, 21, 10, 23, 28, 25, 14, 27\}$

### 3.2.2 Search for Good Interleavers

For the proposed interleaver the parameters of importance are $d_0$ and $\Delta s$ and by correctly choosing them we can design interleavers with performance better than the linear interleaver.

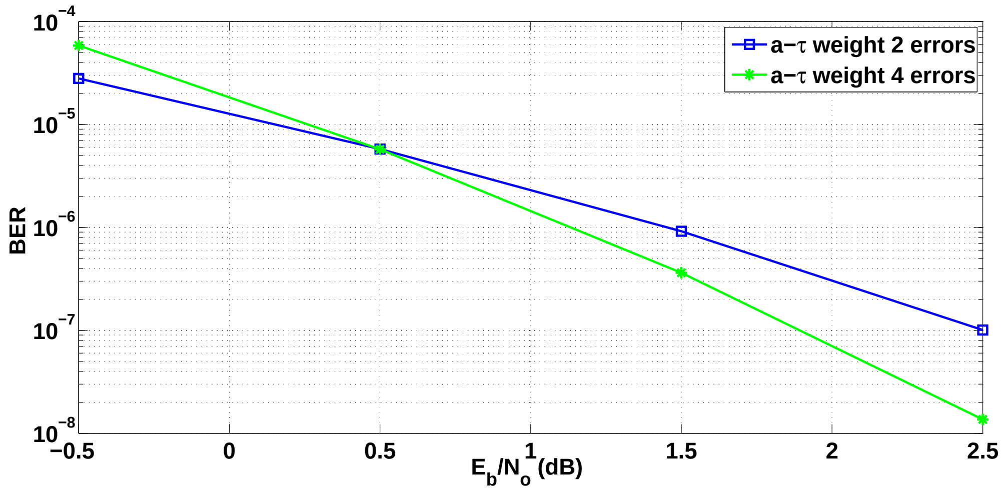

図 3.6: BER Upper Bound for $\Pi_{M|256:(17,64)}$.

The procedure we use for finding good interleavers is as follows. Assuming the interleaver length and the cycle length $\tau$ of the component encoder is known, we first fix the value d and determine the elements of the cycle set. For each element in the cycle set, we calculate the hamming weight of the turbo codewords due to $\tau$ weight-2 errors using the procedure in Figure 2.4 and record $d_{eff}$.

The value of $\Delta s$ that is chosen is the one that produces the largest value of $d_{eff}$ for a given value of $d_0$. This is repeated for all odd integer values of $d_0$ between $(\sqrt{N}, N/2)$ In the case where the codeword weight is the same, $d_0$ with the least value of $\Delta s$ and least multiplicity $N_{free,eff}$ is chosen.

### 3.2.3 Performance Against $a - \tau$ weight-4 errors

Using equation 3.2 we plot the upper bound BER curve for $a-\tau$ weight-2 and $a-\tau$ weight-4 errors for the $\Pi_{M|256:(17,64)}$ multi-shift Interleaver. This is shown in Figure 3.6. We see that at higher $E_b/N_o$ values the $a - \tau$ weight-2 curve dominates which suggest that the dominance of the $a - \tau$ weight-4 errors has been dealt with.

# 第4章　Results

## 4.1 Simulation Results for Interleaver Search

Table 4.1 shows the best values of $d_0$, corresponding value of$\Delta s$ as well as $d_{eff}$ and $N_{free,eff}$ for an interleaver of length, $N = 256$ and 5/7 RSC encoder .We observe that $d_{eff}$ is the same. Since $d_0 = 17$ has the least value of $\Delta s$ and $(N_{free,eff})$, it performs the best. This is followed by $d_0 = 47$ and $d_0 = 31$

| $d_0$ | 17 | 31 | 47 |
|---|---|---|---|
| $\Delta s$ | 64 | 128 | 64 |
| $d_{eff}$ | 38 | 38 | 38 |
| $N_{free,eff}$ | 207 | 208 | 209 |

表 4.1: Best value of $d_0$, corresponding value of $\Delta s$ and the value of the minimum weight codeword for turbo codes with 5/7 component encoder. $N = 256$

Figure 4.1 shows simulation results for Table 4.1 and the performance is as predicted with the best interleaver being $\Pi_{M|256:(17,64)}$.

## 4.2 Comparison with Linear Interleavers

Given an Interleaver of size $N = 2^r$ and component code, the search for good multi-shift interleavers is an evaluation of the best combination of $d_0$ and $\Delta s$. In this section, we present simulation results for turbo codes designed using the multi-shift interleaver using different component codes and compare its performance to the linear interleaver. For each simulation, we set the value of $d_0$ to an odd integer close to $\sqrt{N}$ and used the procedure outlined in Section 3.2.2 to find the best value of $\Delta s$ for the multi-shift interleaver(MSI)

Figure (4.2) and for Figure (4.3) are the simulation results for the 5/7 component code and the 7/5 component code with interleaver length $N = 1024$ and $D = d_0 = 31$.

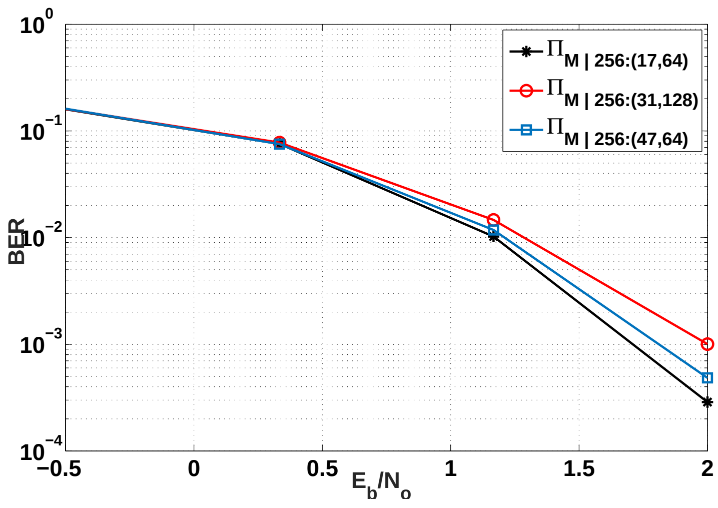

図 4.1: Simulation Results for Table 4.1

For the MSI, the best value of $\Delta s$ is found to be 128 and 32 for the 5/7 component encoder and the 7/5 component encoder respectively. We observe that in both cases the MSI outperforms the linear interleaver.

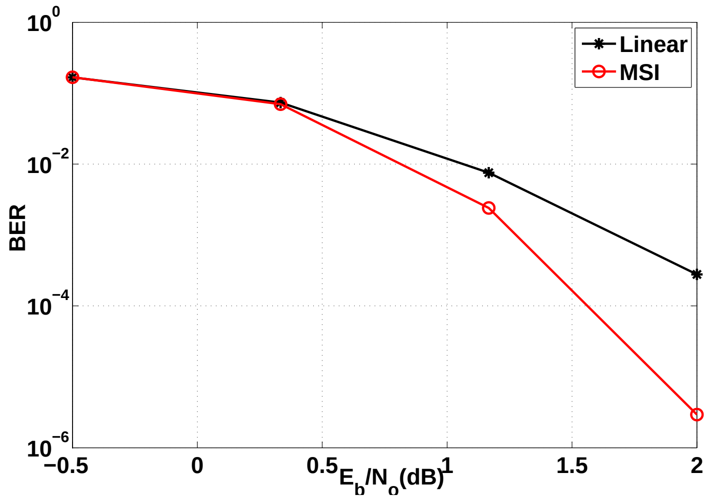

図 4.2: Turbo Code with 5/7 Component Code. $N = 1024$

In Figure (4.4), the performance of the multi-shift interleaver is compared with the Linear interleaver for the 5/7 component code with interleaver length $N = 16384$ and $D = d_0 = 127$. For the MSI the best value of $\Delta s$ is found to be 2048. Again, the MSI outperforms the Linear interleaver.

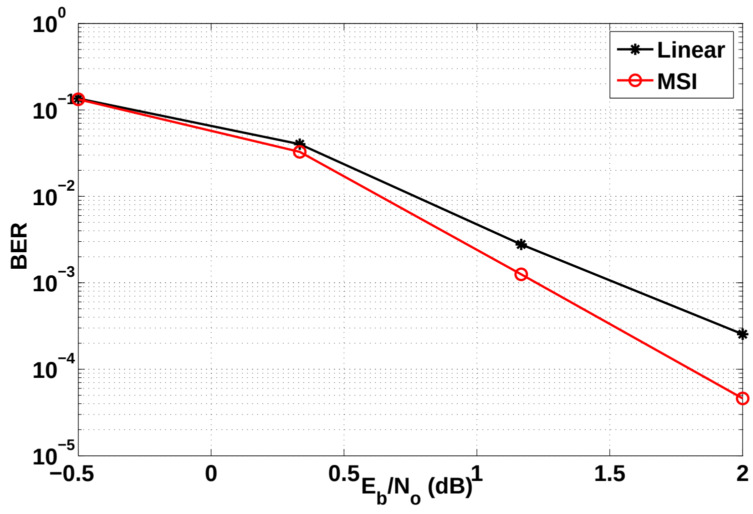

図 4.3: Turbo Code with 7/5 Component Code. $N = 1024$

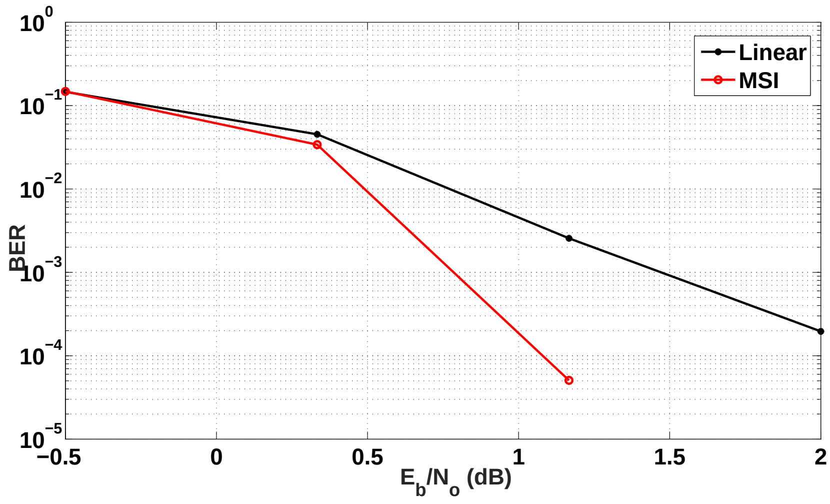

図 4.4: Turbo Code with 5/7 Component Code. $N = 16384$

# 第5章　Conclusion and Future Works

In this research, the multi-shift interleaver was introduced. It was designed with the aim to reduce the dominance of weight-4 errors in linear interleavers although it is non-linear in nature. For a turbo code with interleaver length $N$ and a given RSC encoder, a limited search is carried out to find the best value of $d_0$ and $\Delta s$ for the multi-shift interleaver. The new interleaver outperforms the linear interleaver for both medium and long frame sizes.

Future work in relation to this research include the comparison of the multi-shift interleaver to the PPI and the S-random interleaver. Also we plan to find theoretical BER upper bounds for the multishift interleaver.

# 謝辞

All glory to God who has brought me to the end of a challenging journey.I will like to thank my academic supervisor for all the assistance and direction he gave me during the research. Finally i will like to thank my labmates for their support as well.

# 参考文献

[1] John G. Proakis, Masoud Salehi. ”Digital Communications”, Fifth Edition,Chapter 8, McGraw-Hill.

[2] Oscar Y. Takeshita, Member, IEEE, and Daniel J. Costello , ”New Deterministic Interleaver Designs for Turbo Codes”,IEEE Trans. Inform. Theory, vol. 46,pp. 1988-2006,Nov. 2000.

[3] L. C. Perez, J. Seghers, D. J. Costello, Jr., ”A distance spectrum interpretation of turbo codes”, IEEE Trans. Inform. Theory, vol. 42, pp. 1698-1709, Nov. 1996.

[4] C. Berrou, A. Glavieux and P. Thitimajshima, ”Near Shannon limit error-correcting coding and decoding: Turbo codes”, Proc. Intern. Conf. Communications (ICC), Geneva, Switzerland, pp. 1064- 1070, May 1993.

[5] Jing Sun, Oscar Y. Takeshita ”Interleavers for Turbo Codes Using Permutation Polynomials over Integer Rings”, IEEE Trans. Inform. Theory, vol. 51, pp. 101 - 119 Jan. 2005.

[6] S. Benedetto and G. Montorsi, “ Unveiling turbo codes: Some results on parallel concatenated coding schemes, ” IEEE Trans. Inform. Theory, vol. 42, pp. 409 - 428, Mar. 1996.