社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

ターボ符号器における決定論的インタリーバの設計

KwameBOHULU† 　韓承鎬†,††

† 電気通信大学　大学院情報理工学研究科〒 182-8585　東京都調布市調布ヶ丘 1-5-1
E-mail: †b16631133@edu.cc.uec.ac.jp, ††han.ic@uec.ac.jp

あらまし　決定論的インタリーバは適切なアルゴリズムを用いることでインタリーブを行うため，ターボ符号で広く用いられる．決定論的インタリーバの中で最も設計が容易なのは線形インタリーバである．しかし線形インタリーバは高 SNR (Signal to Noise Ratio) の場合，ビット誤り率 (BER: Bit Error Rate) 特性が劣化する問題点がある．この欠点に対し本稿では，線形インタリーバの係数を決められた定数内で周期的に変化させ，この係数に従ってすべての位置をシフトさせる，新しいインタリーバの設計方法を提案する．そして，提案手法で性能の良いインタリーバを効率よく探索するために，定数制約探索法を用いる．中程度と長いフレームサイズにおいて，提案するインタリーバが線形インタリーバーより優れた BER を達成できることをシミュレーションにより確認した．

キーワード　ターボ符号，決定的なインタリーバー

Deterministic Interleaver Design for Turbo Codes

Kwame BOHULU† and Han CHENGGAO†,††

† Graduate School of Informatics and Engineering, The University of Electro-Communications
Choufugaoka 1-5-1, Chofu-shi, Tokyo, 182-8585 Japan
E-mail: †b16631133@edu.cc.uec.ac.jp, ††han.ic@uec.ac.jp

Abstract　Deterministic interleavers are preferred in the constructing of turbo codes due to the ability to peform interleaving and de-interleaving operations via algorithm. Amongst these, the linear interleaver is the simplest to design but it performs poorly at high SNR values. In this paper, we introduce a new interleaver design which is based of changing the angular coefficient of the linear interleaver by a constant for every position shift. A limited search is performed to find good interleavers. The decoding performance of the new interleaver is better than the linear interleaver for medium and long frame sizes.

Key words　turbo code, deterministic interleaver

1. 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] The index mapping function of the linear interleaver is

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

d is reffered to as the angular coefficient[2]. 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. 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 paper, we design our interleaver by reviewing the linear interleaver and making a slight modification to it to improve its performance.

2. Turbo Codes:Review

In this section, we review the encoding and decoding process for turbo codes for BPSK modulation and transmission over the AWGN channel.

2.1 Turbo Encoding

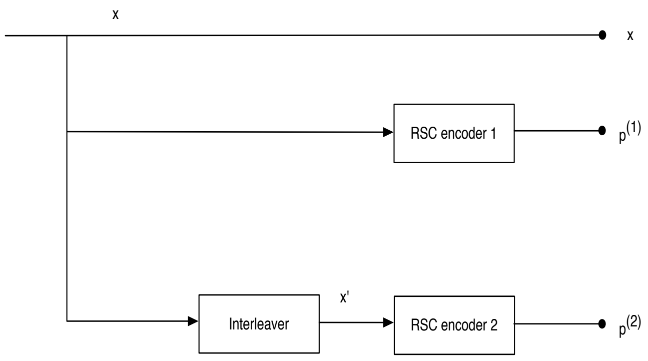

図 1: Turbo Encoder

The system diagram for the turbo encoder is shown in figure(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) is $[\frac{1+D^2}{1+D+D^2}]$ .The encoding process is as follows.

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

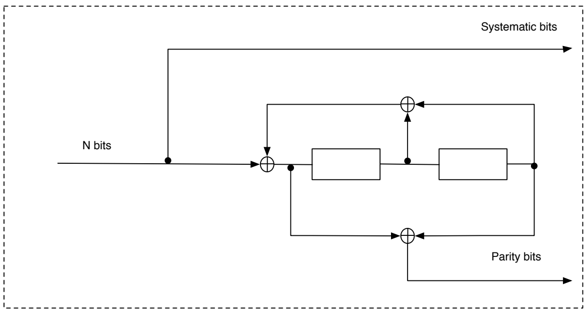

図 2: $[\frac{1+D^2}{1+D+D^2}]$ RSC Encoder

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 $R_c = \frac{N}{3N} = \frac{1}{3}$

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.

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

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

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

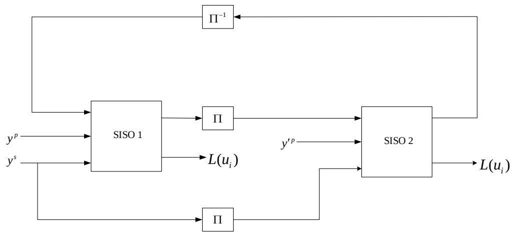

図 3: Turbo Decoder

The system diagram for the turbo decoder is shown in figure (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 γ, α, β respectively[from BCJR]. We represent the previous trellis state and current trellis state by σ' and σ respectively.

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

$c_i = \{x_i, p_i^v\}$, $v = 1, 2$ and the initial values for α and β 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)

$$\begin{aligned}
L(x_i) = &\max_{R_1}[\alpha_{i-1}(\sigma') + \gamma_i(\sigma', \sigma) + \beta_i(\sigma)] - \\
&\max_{R_0}[\alpha_{i-1}(\sigma') + \gamma_i(\sigma', \sigma) + \beta_i(\sigma)]
\end{aligned} \qquad (2)$$

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 γ, α, β using (1) and finally $L(\mathbf{x})$ using (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 β, $\beta_L(\sigma) = -ln(2^m)$.

3. Review of Linear Interleavers for Turbo Codes

RSC encoders are the component code of choice for turbo codes. They are characterized by their cycle length (τ) 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)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 τ 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 τ-seperated weight-$2m$ input error events (or simply as τ weight-$2m$ errors for simplicity sake), where $m = 1, 2$.

Figure(4) shows the effect of τ weight-2 errors on the codeword weight.

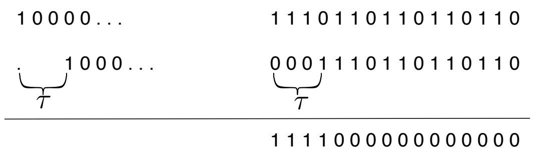

図 4: effect of τ weight-2 errors

The encoder used is the RSC encoder in figure (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 τ ,$\mathbf{x}_{\Delta\tau}$. As can be seen from figure(4) these inputs produce parity outputs $\mathbf{y}$ and $\mathbf{y}_{\Delta\tau}$ (version of $\mathbf{y}$ shifted by τ). 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 τ weight-2 errors irrespective of position of the "1" bits .

From the above example, we see that τ 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. This can be prevented by proper interleaver design.

The linear interleaver is a simple deterministic interleaver based on the circular shifting[2]. Its index mapping function is given by

$$\Pi_{\mathfrak{L}_N}(i) \equiv di \mod N \quad 0 \leqq i < N,\, gcd(N,d) = 1 \tag{3}$$

From the mapping function, we can deduce that in an input sequence, if there are 2 elements seperated by a constant $t \mod N$ they will be mapped to an output sequence with the corresponding elements seperated by a fixed constant $dt \mod N$. When used in turbo codes, there exists a linear congruence relationship (4) which is solvable and has a unique solution.

$$dt \equiv \tau \mod N \tag{4}$$

where t is the distance between the "1" bits present in the weight-2 input. For example for $N = 32$, $d = 5$ and component encoder in figure 2, $t = 7$ satisfies the (4) . This means that if the weight-2 input to the turbo code is of the form $(1 + D^t)D^q$, $q = 0 : N - t$ it is transformed into a τ weight-2 error. As long as the upper parity sequence has a large weight, the codeword produced will most likely have a large weight. For weight-2 inputs, This is acheivable by carefully selecting the value of d such that $t \not\equiv a\tau$, $a = \{1, 2, ..\}$

The input weight-2 analysis done above is a good method for ruling out bad interleavers. However it is not sufficient for analyzing the performance of turbo codes at higher E_b/N_o. Doing analysis for higher input weights may offer deeper insights into the performance of the turbo code, specifically weight 4 inputs.

According to [2], weight 4 inputs of the form $D^q(1+D^t)(1+D^\tau)$ have the potential to produce low weight parity bits in both component codes. Using the same parameters as the previous example and assuming q=0, weight 4 inputs of the form $1+D^3+D^7+D^{10}$. This is essentially 2 τ weight 2 errors seperated by $t-\tau-1$ zeros. This is transformed into an input of the form $1 + D^3 + D^{15} + D^{18}$. The parity bits for this input both have weight 8 and thus the total codeword weight is 20. Though this value is large, the multiplicity of such codeword weights is approximately N and therefore dominates the BER performance of the turbo code. Also changing the size of the interleaver does not improve the performance.

In [6], the approximate expression for BER P_e is derived and is given by (5)

$$P_e \approx \frac{1}{2} \sum_{w_c} D_{w_c} erfc\left(\sqrt{w_c \frac{R_c E_b}{N_o}}\right) \tag{5}$$

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 (5), The BER curves are obtained for τ weight-2 and τ weight-4 errors. This is shown in Figure (5). $N = 1024$, $D = 31$ and the component code used is the same as the one in Figure (2) The simulation results are also shown in the same graph. It can be observed that the approximation provided by the τ weight-2 error alone is largely over estimated, as the performance of the turbo code is clearly bound by τ weight-4 error's curve.

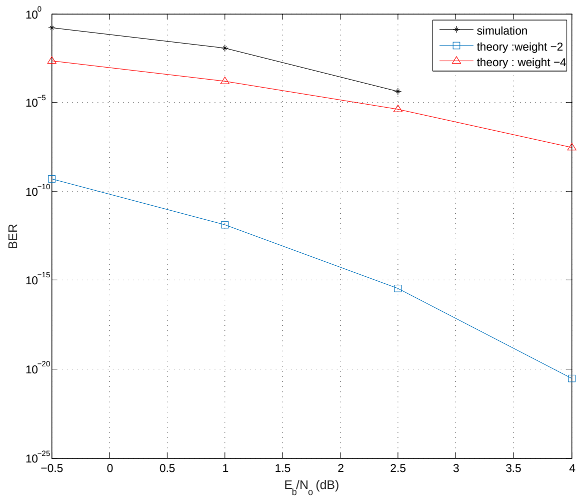

図 5: Comparison of Linear interleaver BER performance: Simulation and Theoretical Results

4. Interleaver Design

In this section we present the design for a new deterministic interleaver. Our design is based on the idea of shifting the value of D by a factor for every shift in position of the original set . Guidlines for selecting parameters to design good interleavers are presented.

4.1 Multi-Shift Interleaver

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.

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

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

$$Dt \equiv \tau \mod N,$$

. which in turn makes the linear interleaver succeptible to weight 4 inputs of the form $D^q(1+D^t)(1+D^\tau)$[2]

One possible method for getting rid of the linear congruence constraint is to increase the previous value of D by a value $\Delta s \mod N$ for every position shift . It is shown in Example 4.1 and 4.2 that for $N = 2^r,\ r = \{1, 2, ...\}$, changing the value of D for every shift still causes the original sequence to be interleaved once D remains an odd number and Δs is an even number. By limiting the value of Δs to $2^q,\ q = 1 : r-1$, we have a set of D values of length $V = N/\Delta s$. We shall refer to this set as the cycle set $\mathbf{d}$. The index mapping function used is similar to that of the linear interleaver.

$$\Pi_{M|N:(d_0,\Delta s)} : x \mapsto p_x \quad 0 \leqq x < N$$

The algorithm for the proposed interleaver is shown below.

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

[Example 4.1] : 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 $\{0, 5, 30, 27, 12, 1, 10, 23, 24, 29, 22, 19, 4, 25, 2, 15$
$, 16, 21, 14, 11, 28, 17, 26, 7, 8, 13, 6, 3, 20, 9, 18, 31\}$
[Example 4.2] : 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 $\{0, 29, 18, 31, 4, 1, 22, 3, 8, 5, 26, 7, 12, 9, 30$
$, 11, 16, 13, 2, 15, 20, 17, 6, 19, 24, 21, 10, 23, 28, 25, 14, 27\}$

It can be seen from the algorithm that the elements of $\mathbf{d}$ as well as the position of the elements of the set x are "shifted". We thus refer to this interleaver as the multi-shift interleaver, which we shall denote by the symbol $\Pi_{M|N:(d_0,\Delta s)}$. where d_0 is the initial value in the cycle set.

4. 2 Selecting Good Interleavers

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

The procedure we use for finding good interleavers is as follows. Assuming the interleaver length and the cycle length τ 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 produced by τ weight-2 errors using the procedure in Figure 4 and record the minimum codeword weight.

The value of Δs that is chosen is the one that produces the largest minimum codeword weight for τ weight-2 errors 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 Δs is chosen. In the case where Δs is also the same, d_0 with the least multiplicity $N_{free,eff}$ is chosen.

Table 1 shows the best values of d_0, corresponding value ofΔs as well as d_{eff} and $N_{free,eff}$ for an interleaver of length, $N = 256$. The component encoder used is $\frac{1+D^2}{1+D+D^2}$ (octal notation: 5/7) .We observe that d_{eff} is the same. Since $d_0 = 17$ has the least value of Δs and $(N_{free,eff})$, it performs the best. This is followed by $d_0 = 47$ and $d_0 = 31$

d_0	17	31	47
Δs	64	128	64
d_{eff}	38	38	38
$N_{free,eff}$	207	208	209

表 1: Best value of d_0, corresponding value of Δs and the value of the minimum weight codeword for turbo codes with 5/7 component encoder. $N = 256$

Figure 6 shows simulation results for Table 1 and the performance is as predicted.

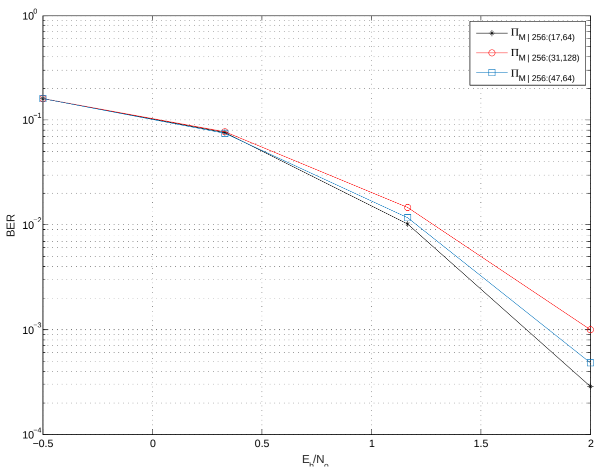

図 6: Simulation Results for Table 1

5. Results

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 Δs. In this section, we present simulation results for turbo codes designed using the multi-shift interleaver using different component codes and

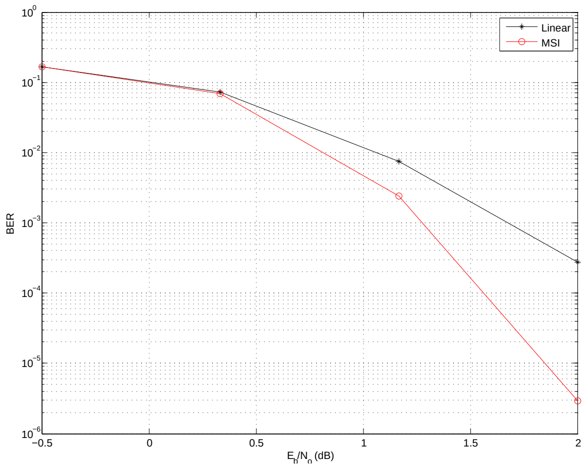

図 7: Turbo Code with 5/7 Component Code. $N = 1024$

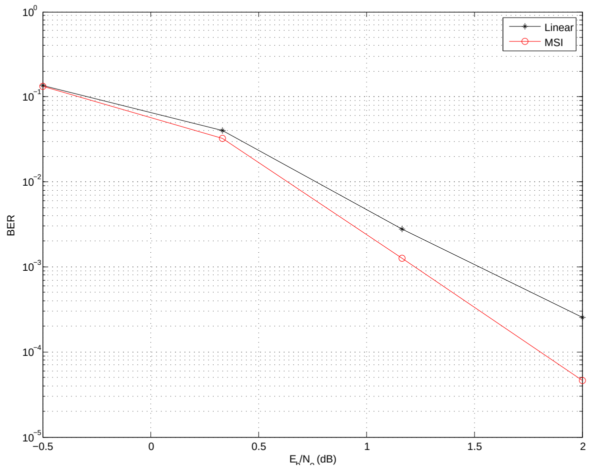

図 8: Turbo Code with 7/5 Component Code. $N = 1024$

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 4. to find the best value of Δs for the multi-shift interleaver(MSI)

Figure (7) and for Figure (8) are the simulation results for the 5/7 component code and the 7/5 component code with interleaver length $N = 1024$ and $D = d_0 = 31$. For the MSI, the best value of Δs 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.

In Figure (9), 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 Δs is found to be 2048. Again, the MSI outperforms the Linear interleaver.

6. Conclusion

In this paper, the multi-shift interleaver was introduced.

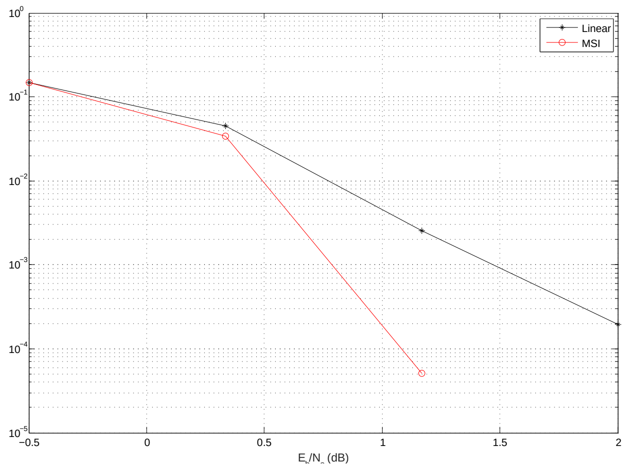

図 9: Turbo Code with 5/7 Component Code. $N = 16384$

It was designed with the aim to reduce the dominance of weight-4 errors in linear interleavers. 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 Δs for the multi-shift interleaver. The new interleaver outperforms the linear interleaver for both medium and long frame sizes.

7. References

[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.