Deterministic Interleaver Design for Turbo Codes

Kwame Ackah Bohulu, 1631133

22-06-2017

1 Abstract

2 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 [distance spec interp]. 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 an alphabet 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 were large interleaver sizes are required, the memory requirements to store the interleaver tables alone makes 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

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

The design of the linear interleaver is essentially picking a suitable value of 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 based on the weight-2 and weight-4 input analysis of turbo codes constructed using linear interleavers. Our interleaver breaks the dominance of weight-4 inputs in linear interleavers and effectively reduces the error floor present in linear interleavers.

The beakdown of the paper is as follows, in section 2, the encoding and decoding process of turbo codes is briefly reviewed. In section 3 the BER uppper-bound analysis based on weight-2 and weight-4 inputs is carried out. Next, the definition and design of the interleaver is given in section 4 based on the analysis carried out in the previous section. Comparisons are made with

the linear interleaver via simulation and the obtained results are are discussed in section 5. Finally conclusions are drawn in section 6.

3 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.

3.1 Turbo Encoding

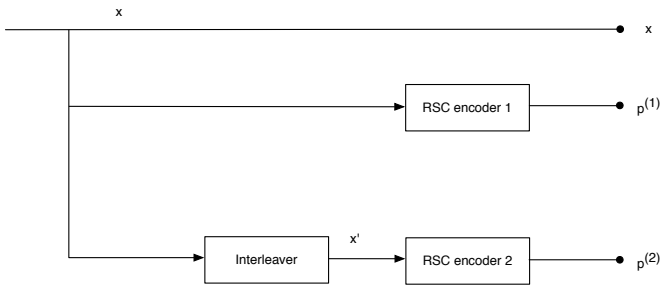

Figure 1: Turbo Encoder

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.

An information sequence of $\mathbf{x} = \{x_0, x_1, ..., x_{N-1}\}$ of length N 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 $\mathbf{p}^{(1)} = \{p_0^{(1)}, p_1^{(1)}, ..., p_{N-1}^{(1)}\}$ also of length N. In the case where it is desired to return CE1 to the all-zero state, $m = K - 1$ tail bits are fed into CE1 which brings the total length of $p^{(1)}$ to L=N+m. 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 $\mathbf{p}^{(2)} = \{p_0^{(2)}, p_1^{(2)}, ..., p_{N-1}^{(2)}, ..., p_{L-1}^{(2)}\}$ which has total length L due to the extra

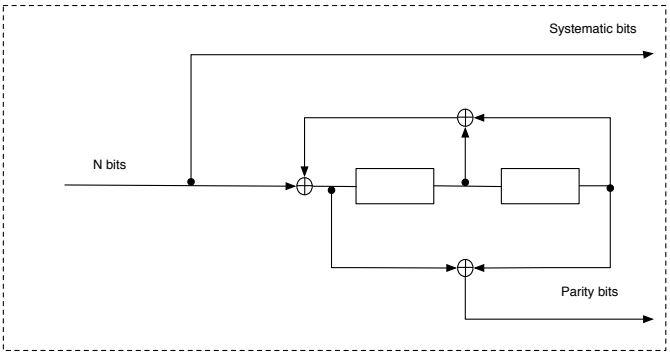

Figure 2: $[\frac{1+D^2}{1+D+D^2}]$ RSC Encoder

tail bits added to force CE2 back to the all-zero state. $\mathbf{x}$ (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

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

has length $3 \times (L)$ and the turbo encoder has rate $R_c = \frac{N}{3L} \approx \frac{1}{3}$

3.2 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 $3L$, where $\mathbf{y}^x, \mathbf{y}^{p^{(1)}}, \mathbf{y}^{p^{(2)}}$ correspond to the systematic, upper and lower parity sequence respectively.

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

$$\mathbf{y}^{p^{(1)}} = \{y_1^{p^{(1)}}, y_2^{p^{(1)}}, ..., y_{N-1}^{p^{(1)}}, y_N^{p^{(1)}}, ..., y_L^{p^{(1)}}\}$$

$$\mathbf{y}^{p^{(2)}} = \{y_1^{p^{(2)}}, y_2^{p^{(2)}}, ..., y_{N-1}^{p^{(2)}}, y_N^{p^{(2)}}, ..., y_L^{p^{(2)}}\}$$

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

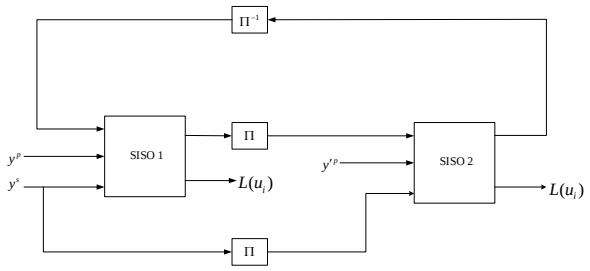

Figure 3: Turbo Decoder

represent by the symbols γ, α, β respectively[from BCJR].

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

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

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

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

$L(x_i)$ is then calculated by using equation (2)

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

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 deinterleavedand 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 $\mathbf{x}$.

4 BER Performance Bounds for Linear Interleaver-based Turbo Codes

In this section, we analyze τ-seperated weight-2m input error and their effect on RSC component encoders. We then analyze their effect on the BER performance of turbo codes.

4.1 τ-seperated Weight-2m Input Error Events

RSC encoders are mostly used as component encoders in turbo codes. This makes it possible to estimate their BER performance of turbo can be estimated using equation (3) [new det].

$$Pb(e) \approx \frac{1}{2}\sum_{w_c} D_{w_c} erfc\left(\sqrt{w_c \frac{R_c E_b}{N_o}}\right) \tag{3}$$

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. RSC encoders 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 $[1, 0, 0, 0, ...]$[PPI]. 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.

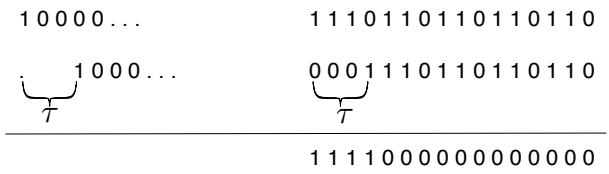

Figure 4: effect of τ weight-2 errors

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 $[1, 0_{\tau-1}1, 0_{N-\tau+1}]$ (where 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 ,$\mathbf{x}_\tau$. As can be seen from figure(4) these inputs produce parity outputs $\mathbf{y}$ and $\mathbf{y}_\tau$ (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 . The same observation can be made for τ weight-4 errors.

Irrespective of the kind of input fed into the Turbo encoder, we can use the procedure in figure(4) to calculate the weight of the codeword produced. The total codeword weight is given by

$$w_H(c) = w_H(x) + w_H(p^{(1)}) + w_H(p^{(2)}) \tag{4}$$

where $c, x, p^{(1)}, p^{(2)}$ are the codeword, input bits and upper and lower parity checkbits respectively.

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 is consistent with the analysis done in[PPI] and [new det]

4.2 Analysis of τ-seperated Weight-2 Input Error Events for Linear Interleaver-based Turbo Codes

As mentioned in previous sections, the index mapping function of the linear interleaver is given by

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

Due to the condition that $gcd(N, d) = 1$, for weight -2 inputs, there exists a value t(t is the distance between the "1" bits in the weight-2 input) that satisfies the linear congruence in equation (5)[new det].

$$dt \equiv \tau \mod N \tag{5}$$

This means that for a given d, a weight-2 input with the "1" bits seperated by t-1 "0" bits is transfromed into an τ weight-2 error by the interleaver which causes a low weight error event to be produced in the second component encoder.

By carefully choosing the value of d, we can make sure that t is large enough to produce a high weight parity bit in the first component encoder and therefore prevent the occurence of low-weight turbo codewords.

4.3 Analysis of τ-seperated Weight-4 Input Error Events for Linear Interleaver-based Turbo Codes

According to [new det] weight-4 inputs of the form $\mathbf{x} = [1, 0_{\tau-1}, 1, 0_{t-1}, 1, 0_{t+\tau-1}, 1, 0_{N-(t+\tau)}]$ produce low-weight parity bits in both component codes, which results in a low-weight turbo codewords. Since shifted versions of $\mathbf{x}$ also produce low-weight turbo codewords, the multiplicity of such low-weight codewords is high and is very likely to dominate BER performance of the turbo code at high E_b/N_o

The estimated BER union bound performance of the rate 1/3 turbo encoder with the linear interleaver depth, d=31 and $[\frac{1+D^2}{1+D+D^2}]$ RSC encoder as component code is shown in figure (5).

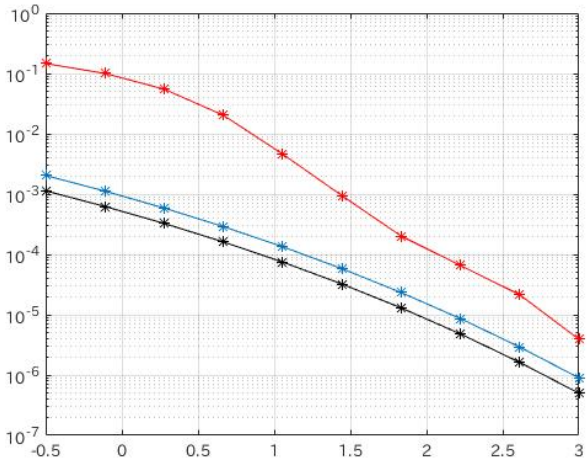

Figure 5: Comparison of Linear interleaver BER performance: Simulation and Theoretical

This graph is drawn using equation(3)

A simplified version of equation (3) is also ploted with the assumption that the multiplicity is approximately N. By running a computer search, we found that for frame length $N = 1024$, $w_c = 20$ and $A_{w_x,w_p} = 1006$. Therefore equation (3) becomes

$$Pb(e) \approx \frac{2}{N} 1006\, erfc\left(\sqrt{20\frac{1}{3}\frac{E_b}{N_o}}\right) \tag{6}$$

and the simplification with the assumption that $A_{w_x,w_p} = 1024$ is

$$Pb(e) \approx 2\, erfc\left(\sqrt{\frac{20}{3}\frac{E_b}{N_o}}\right)$$

The approximations are compared to simulation results for the linear interleaver-based turbo code is also plotted in figure (5). It can be seen that the performance of the turbo code is close to the bound calculated using equation (6). We therefore conclude that the performance of linear interleavers is upper-bound by τ weight-4 errors This is consistent with results obtained in [new det].

5 Interleaver Design

In this section we present the design for a new deterministic interleaver which effectively splits τ weight-2m (m=1,2) error events, that are problematic in the linear interleaver. We begin with an alternate view of how the linear interleaver works and build upon that idea to design our interleaver. Guidlines for selecting parameters to design good interleavers are presented.

5.1 Linear Interleaver from a Position-Shifting Point of View

The process of linear interleaving is basically the shifting the positions of elements in set by a certain factor d mod N. Keeping d constant along with the fact that gcd(d,N)=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 $\mathbf{x}_4 = [1, 0_{\tau-1}, 1, 0_{t-1}, 1, 0_{t+\tau-1}, 1, 0_{N-(t+\tau)}]$

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 5.1 and 5.2 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 which is of length of $N/\Delta s$. We shall refer to this set as the cycle set. To maintain the idea of position shifting, the index mapping function used is of the form

$$\Pi_{M_{N:(d,\Delta s)}} : x \mapsto M(p_x) \quad 0 \leq x < N$$

The algorithm for the proposed interleaver is shown below.
1. $p_0 = 0$
2a. $p_i = p_{i-1} + d \mod N, \quad 0 < i < N, \quad d$ *is an odd integer*
2b. $d = \Delta s + d \mod N, \quad \Delta s = 2^q, \quad q = 1 : r - 1$

Example 5.1 *: 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 the initial value of* $d = 5$*. The cycle set is* $\{5, 9, 13, 17, 21, 25, 29, 1\}$ *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 5.2 *: 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 the initial value of* $d = 5$*. The cycle set is* $\{5, 13, 21, 29\}$ *and the interleaved set is* $\{0, 29, 18, 31, 4, 1, 22, 3, 8, 5, 26, 7, 12, 9, 30, 11, 16, 13, 215, 20, 17, 6, 19, 24, 21, 10, 23, 28, 25, 14, 27\}$

It can be seen from the algorithm that the value of 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 $\Pi_{M_{N:(d,\Delta s)}}$. where d is the initial value in the cycle set.

5.2 Selecting Good Interleavers

For the proposed interleaver the parameters of importance are d 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 equation (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 a given value of d.

It is essential to limit the range of d to search from. The range of d to search from is odd integers between $\sqrt{N}$ and N

Table 1 shows the best values for chosen for d and Δs as well as the value of the minimum weight codeword for an interleaver of length, $N = 256$. The component encoder is $\frac{1+D^2}{1+D+D^2}$ (octal notation: 5/7).

d	17	21	47
$\Delta s(5/7)$	64	8	64
d_{eff}	15	14	15

Table 1: Initial value of d and corresponding value of Δs and the value of the minimum weight codeword for turbo codes with 5/7 component codes

figure 6 shows simulation results for Table 1.

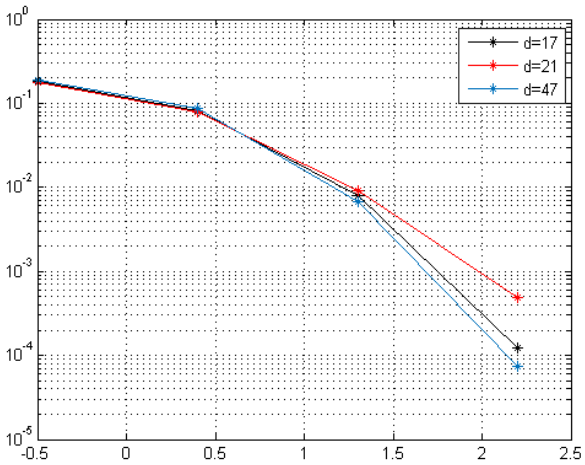

Figure 6: Simulation Results for Table 1

6 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 and Δs. Using the recommendation in [new Det] we set the initial value of d to an odd integer value close to $\sqrt{N}$. We then use the procedure outlined in Section 5 to find the best value of Δs. In this section, we present simulation results for turbo codes designed using the multi-shift interleaver using different component codes. Figure () and for Figure () are the simulation results for the 5/7 component code and the 7/5 component code. The performance is compared with the Linear interleaver. The interleaver length $N = 1024$. In Figure (), the performance of the multi-shift interleaver is compared with the Linear interleaver for the 5/7 component code with interleaver length 16384

7 Conclusion

8 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