Design and Evaluation of Adaptive (Serial/Parallel) Concatenated Convolutional Codec

Asst. Prof. Dr. Khamis A. Zidan
Computer & Software Eng. Dept., College of Eng.
Al-Mustansiriya University, Baghdad, Iraq

Dr. Raghad Z.Yousif
Computer & Software Eng. Dept., College of Eng.
Al-Mustansiriya University, Baghdad, Iraq

Abstract

In this paper, parallel Concatenated Convolutional Codes (PCCC$_s$) is modeled as a special case of Serial Concatenated Convolutional Code (SCCC$_s$). Consequently, resulting in Adaptive (parallel/serial) concatenated convolutional code in which the same encoder and decoder can be used for both types of concatenated convolutional codes. To achieve this goal some interleaver restrictions are made to modify the classical structure of Semi-Random interleaver. The decoding stage is based on classical SCCC iterative decoder, with added interleaver restrictions. The core of decoder structure is a soft-input soft-output (SISO) a posteriori probability (APP) module. Log-Map is used as (APP) for its superior performance.

This work also presents some classical structured interleavers such as Block and Circular interleavers, adding to them the presented restrictions for sake of comparison. The resulted performance curves from computer simulation, shows that degradation signal per bit energy to noise ratio (E_b/N_o) for the proposed decoder as compared to the classical PCCC is no more than 0.45 dB at less than 10^{-5} bit error rates (BER),over Fading channel in worst case.

الخلاصـــــة

في هذا البحث تم تقديم نوع جديد من المرمزات المتوازية المتراصة الملتفة **PCCC** *والتي تم بنائها بالاعتماد على المرمزات المتسلسلة المتراصة الملتفة* **SCCCs** *وهكذا فان المرمز المتكيف المتراص الملتفِ* **Adaptive CCCs** *الناتج يستخدم نفس التقنية في عملية الترميز وعملية فتح الترميز للنوعين التقليديين انفي الذكر* **(PCCCs & SCCCs)**. *لتحقيق هذه الغاية تم أجراء تحوير للشكل التقليدي للمزحف من نوع شبة عشوائي* **(Semi-random)** *كما تم تقديم أنواع أخرى من المزحفات التقليدية مثل المزحف الدائري والمزحف* **Block interleaver** *مع تحوير عمليهما لكي يلائم المنظومة المقدمة لأجل المقارنة.*

تبين نتائج الأداء للمنظومة المقدمة مقارنة مع **PCCC** *التقليدي بان مقدار الاضمحلال في نسبة الإشارة إلى الضوضاء لا تتعدى* **0.45 dB** *عند معدل خطا اقل من* (10^{-5}) *في* **Fading channel** *في أسوا الحالات.*

1. Introduction

Channel Coding has become an indispensable tool in modern communication systems dominated by a transmitted power and bandwidth constraints [1,2,3]. Turbo-Codes and related. Concatenated coding schemes introduced in 1993 by Berrou [4], have near Shannon limit error correction capability and are among the most advanced channel coding schemes. The important innovation of Berrou was the introduction of iterative decoding schemes of convolutional codes by means of information exchange. The (PCCC) termed 'Turbo Code', have been shown to achieve remarkable power efficiencies for BER_s down to about 10^{-5} or 10^{-6} [4]. However for BER_s below 10^{-6} the performance is much less impressive due to a" flattening " of BER curves. The flattening problems can be improved by using $SCCC_s$, which offer performance superior to $PCCC_s$ for moderate and low BER regions. However, $PCCC_s$ still offered better performance than $SCCC_s$ for (E_b/N_o) very close to the capacity limits. Thus communication systems operating near the capacity limit should use $PCCC_s$ in favor of $SCCC_s$, while systems operating at high (E_b/N_o) should use $SCCC_s$ [2,3,5]. This paper gives an enhancement to the ideas presented in [3] by simulate $PCCC_s$ over additive white Gaussian noise (AWGN) and Flat Rayleigh fading channel using what is called structured Semi-random interleaver. Thus, the first part of this paper shows that a PCCC can be represented as a punctured SCCC with interleaver restriction. Interleaver structuring, next the design steps of Semi-random interleaver structuring is stated, then whole system model is proposed. Finally the simulation results for different frame sizes are presented.

2. Adaptive Encoder

In this section, adaptive encoder termed a parallel/serial concatenated convolutional code encoder is presented. This encoder consisting of two identical encoders concatenated in series. ***Appendix (A)*** shows the details of recursive systematic convolutional encoder (RSC) encoder used in this work. If the frame length input to the SCCC encoder is N-bits then the output encoded frame length should be 4N-bits. This action is abbreviated as (N/4N) SCCC, which can be punctured to form a (N/3N) PCCC. The block diagram of Adaptive encoder is presented in **Fig.(1)**, the codeword produced by RSC encoder is denoted by:

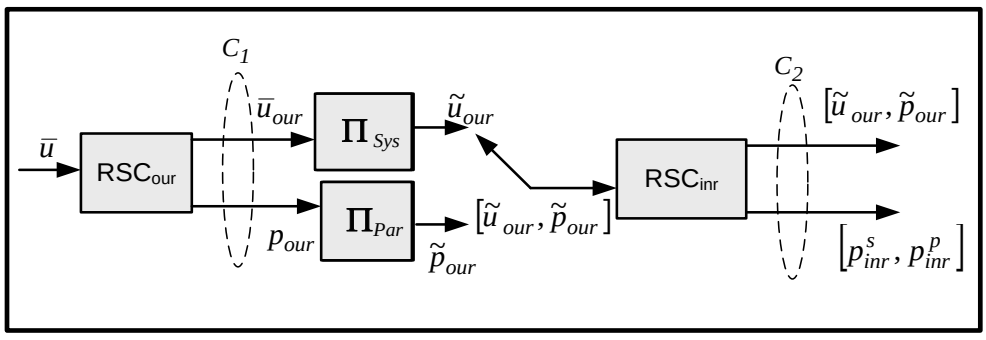


Figure (1) Proposed adaptive encoder

$$C_1 = [C_1^1, C_2^1, ..., C_N^1, ..., C_{2N}^1] \quad \text{..... (1)}$$

which is equal to:

$$\mathbf{C_1 = [\bar{u}^1_{our}, p^1_{our}, \bar{u}^2_{our}, p^2_{our}, ..., \bar{u}^N_{our}, p^N_{our}, ..., \bar{u}^{2N}_{our}, p^{2N}_{our}]} \quad \text{..... (2)}$$

where, $\bar{\mathbf{u}}_{\mathbf{our}}$ is the systematic output from outer encoder (RSC_{our}), while P_{our} is the parity output from the same encoder. The input vector to the RSC_{our} $\bar{\mathbf{u}}$ of N-information bits is denoted by:

$$\bar{\mathbf{u}} = [\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4, ..., \bar{u}_N] \quad \text{..... (3)}$$

As shown in the **Fig.(1)**, the data vector passes serially through a pair of RSC encoders, with interleaving between them. The interleaver is one-to-one mapping function that maps a sequence of τ -bits into another sequence of τ -bits. The interleaver performs this mapping by permuting the input bits:

$$\tilde{x}_k = \prod(x(k)), k \in [0, \tau - 1] \quad \text{..... (4)}$$

where, x is the input sequence, $\tilde{x}_k$ is the interleaved sequence, and Π is the permutation function.

Referring to the proposed concatenated encoder then:

$$\tilde{\mathbf{u}}_{\mathbf{our}} = \bar{\mathbf{u}}\ \Pi_{Sys} \quad \text{..... (5)}$$

and,

$$\tilde{\mathbf{p}}_{\mathbf{our}} = \mathbf{p}_{our}\ \Pi_{Par} \quad \text{..... (6)}$$

Are the interleaved versions of both systematic ($\bar{\mathbf{u}}_{\mathbf{our}}$) and parity ($P_{our}$) outputs from RSC_{our}. These sequences can be denoted by:

$$\tilde{\mathbf{u}}_{\mathbf{our}} = [\tilde{u}^1_{our}, \tilde{u}^2_{our}, ..., \tilde{u}^N_{our}] \quad \text{..... (7)}$$

while,

$$\tilde{\mathbf{p}}_{\mathbf{our}} = [\tilde{p}^1_{our}, \tilde{p}^2_{our}, \tilde{p}^3_{our}, ..., \tilde{p}^N_{our}] \quad \text{..... (8)}$$

The input to the RSC_{inr} (inner recursive systematic convolutional encoder) then is formed by the concatenation of these previously mentioned components. The output codeword produced by the RSC_{inr} encoder is denoted by:

$$\mathbf{C_2 = [C^2_1, C^2_2, ..., C^2_{2N}, ..., C^2_{4N}]} \quad \text{..... (9)}$$

The systematic output of RSC_{inr} is $[\tilde{\mathbf{u}}_{our}, \tilde{\mathbf{p}}_{our}]$, while the parity output $[\mathbf{p}^{s}_{inr}, \mathbf{p}^{p}_{inr}]$ can be partitioned into the parity bits due to the outer encoder's systematic bits $(\mathbf{p}^{s}_{inr})$ and the parity bits due to the outer encoder's parity bits $(\mathbf{p}^{p}_{inr})$. The resulting code can be viewed as a special type of product code, since the same information sequence is encoded twice in different orders. Thus recalling equation (4) the output codeword form RSC_{inr} can be further split in to four fields as illustrated in **Fig.(2)**, hence:

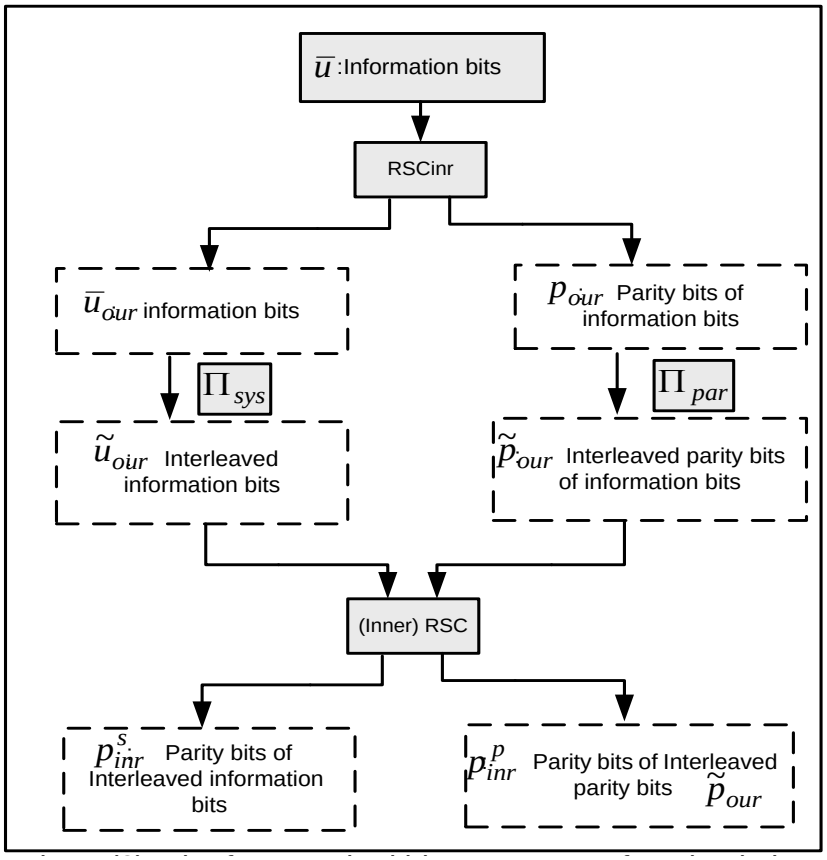


Figure (2) The four resulted bit sequences after simulating encoder circuit

$$\mathbf{C}_2 = [\tilde{\mathbf{u}}^{1}_{our}, \mathbf{p}^{s1}_{inr}, ..., \tilde{\mathbf{u}}^{N}_{our}, \mathbf{p}^{sN}_{inr}, \tilde{\mathbf{p}}^{1}_{our}, \mathbf{p}^{p1}_{inr}, ..., \tilde{\mathbf{p}}^{1N}_{our}, \mathbf{p}^{pN}_{inr}] \quad \text{..... (10)}$$

The proposed system encoder is slightly different than that of a conventional SCCC [1], since the inner encoder encodes all of the outer encoder's systematic bits before it encodes the outer encoder's parity bits. When only $[\tilde{\mathbf{u}}_{our}, \tilde{\mathbf{p}}_{our}, \mathbf{p}^{s}_{inr}]$, are transmitted the code is a rate 1/3 PCCC. Note that there is slightly difference between presented and conventional PCCC, because the output of (RSC_{our}) which play the role if RSC_1 (first recursive systematic convolutional encoder) in conventional PCCC is interleaved.

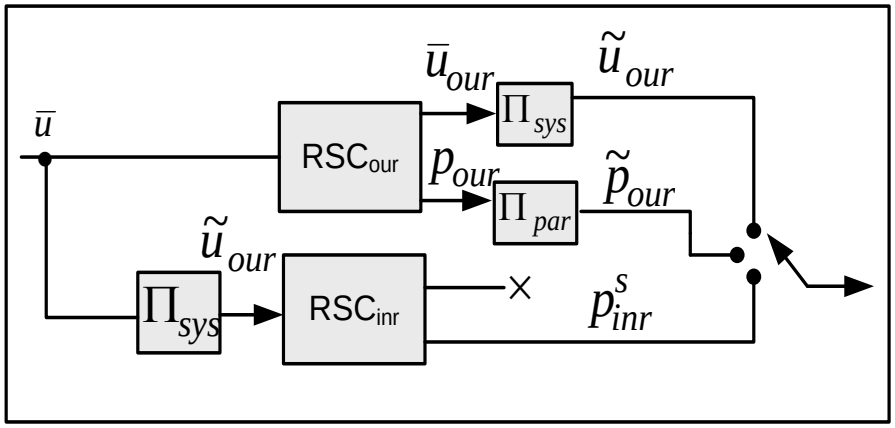


Figure (3) Equivalent rate (1/3) encoder

2-1 Puncturing

As deduced in the previous section, the proposed system transmits only [$\tilde{\mathbf{u}}_{\mathbf{our}}, \tilde{\mathbf{p}}_{\mathbf{our}}, \mathbf{p}^{\mathbf{s}}_{\mathbf{inr}}$]. This means we can completely suppressing $(\mathbf{p}^{\mathbf{p}}_{\mathbf{inr}})$ from taking part in the output code this is by definition puncturing (puncturing is the process of removing certain symbol/position from the codeword) [3]. If we consider a puncturing period of (9) then the puncturing matrix at the output of inner encoder will be

$$\mathbf{pun}_{\mathbf{M}} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} \quad \text{..... (11)}$$

2-2 Structured Interleaving

Interleaving is a key component of any "Turbo code" [7]. The role of interleaver is to break low weight input sequences and hence increase the code free Hamming distance or to reduce the number of codewords with small distance spectrum. Thus a structure of interleaver affects the performance at high signal to noise ratios (SNR_s), usually random or pseudorandom interleavers are recommended [7]. The structured interleaver designed such that it will maps the systematic bits, into the first half of interleaved frame and parity bits into second half. This is equivalent to interleaving the systematic and parity bits independently and then putting them in cascade so that all the interleaved systematic bits are introduced to the input of the inner encoder before any of the interleaved parity bits. Since the systematic and parity bits are interleaved separately there are actually two sub interleaving sequences: $(\Pi_{\mathbf{sys}}, \mathbf{and}, \Pi_{\mathbf{par}})$ with in this structured interleaver, each of which is implemented as a semi-random, block, or circular interleaver …etc.

2-2-1 Structured Semi-Random Interleaver

The semi-random interleaver in which permutations resulting in short cycles are avoided, a short cycle occurs when two bits that are close to each other remain close after interleaving. It is concluded that these short cycles degrade the performance of iterative

decoding. The reason of this degradation is that they cause nearby extrinsic inputs to be correlated to each others. Since extrinsic inputs that are close to each others influence the same part of the decoding trellis, it is natural that the performance of the decoder is degraded, if the inputs are correlated [2]. However, the semi-random interleaver is based on the random generation of N-integer numbers (0 to N-1) but with the following constraint. Each randomly selected integer is compared to the *S* most recently selected integers. If the current selection is within S-distance from the previous S-integers less than at least one of the previous S-integers, then it is rejected and a new integer is selected until the previous condition is satisfied. This process is repeated until all N-integers are extracted. The search time increases with S, and there is no guarantee that the process will finish successfully [7]. As a rule to produce a solution in a reasonable amount of time:

$$S \leq \sqrt{\frac{N}{2}} \quad \text{..... (12)}$$

Since the output of the SCCC encoder has multiplexed the systematic and parity information of the inner encoder. The spread interleaver algorithm presented previously now reduces to the following:

Step 1: Select the value of S according to equation (11)

Step 2: Generate a frame of random indexes (i) with i< N/2 (first half) and map each systematic bit to the appropriate position in the interleaved frame.

Step 3: Generate a frame of random indexes (i) with i< N/2 (second half) and map each parity bit to the appropriate position in the interleaved frame.

Step 4: After all bits are mapped, check only the last S number of bits of the first half of interleaved frame and the first S number of bits of the second half of the interleaved frame to see if adjacent bits in the original input frame are present and if they do then check to see if they satisfy the S criterion.

Step 5: If any two bit positions do not satisfy the condition, then rearrange any one of them by picking a random number i <(N/2) –S if a systematic bit and i<(N/2)+S□□, if a parity bit and exchange positions with the bit at this random index position.

Step 6: Repeat steps 4 and 5 (if required) for all 2*S number of bits.

This reduces the number of operations in the original spread interleaver per iteration and for large frame. There is a high probability for large frame sizes that the S positions on either side of the half way mark of the interleaved frame do not contain adjacent bits of the original frame.

Moreover, the quality of the interleaved frame (average of all the distances in the interleaved frame between adjacent bits of the original frame) generated with this interleaver restriction is exactly the same if not better than the conventional spread interleaver generated frame. This is revealed by the result in **Fig.(4)**, which is a comparison plot of the SCCC with and without incorporating the interleaver restriction. **Figure (5)** shows Input/output position

plot of 192 bit Semi-Random structured interleaver. **Figure (6)** shows the flow diagram of the proposed structured Semi-random interleaver.

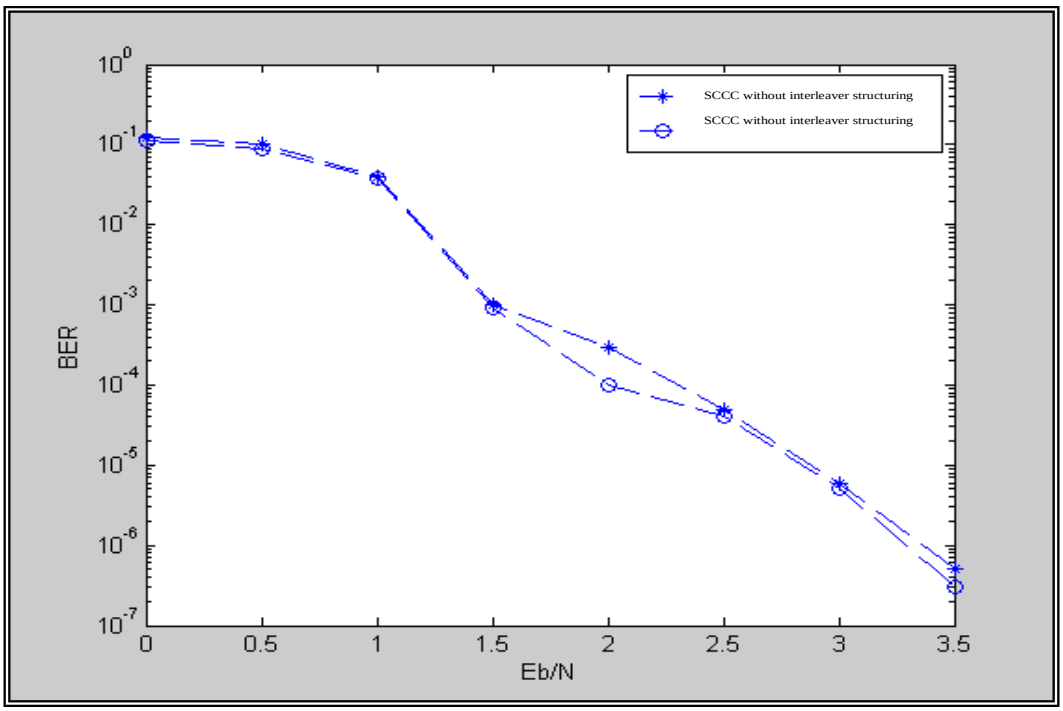


Figure (4) The performance of rate (1/4) SCCC with and without interleaver structuring

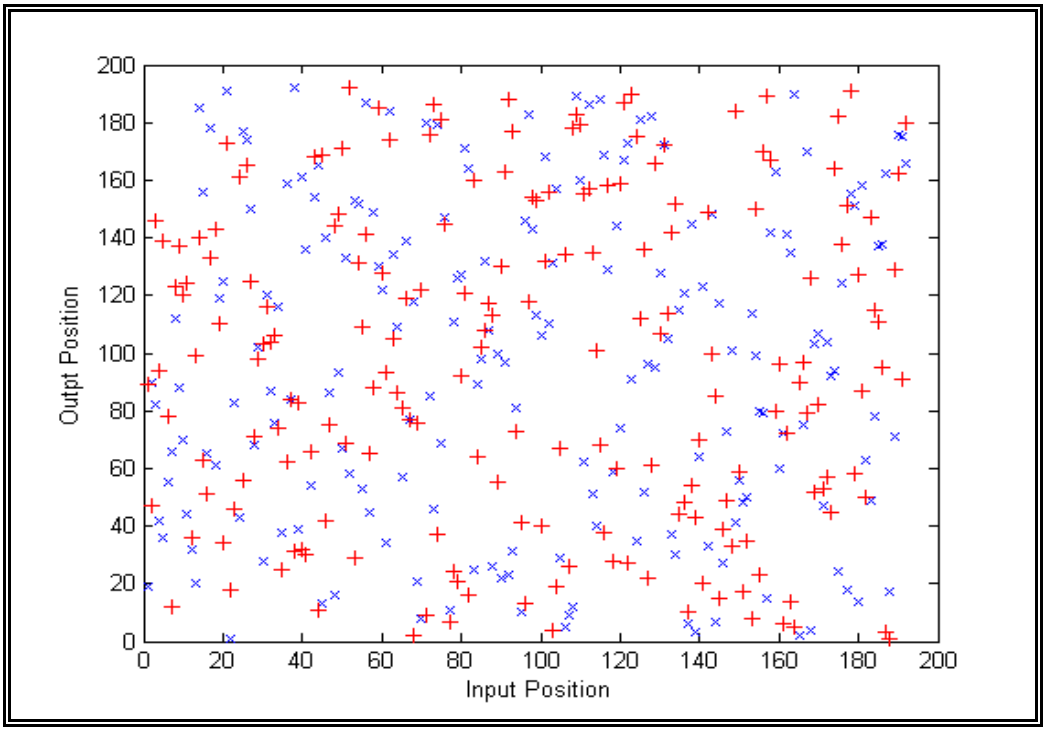


Figure (5) Input/output position plot of 192 bits structured semi-random interleaver

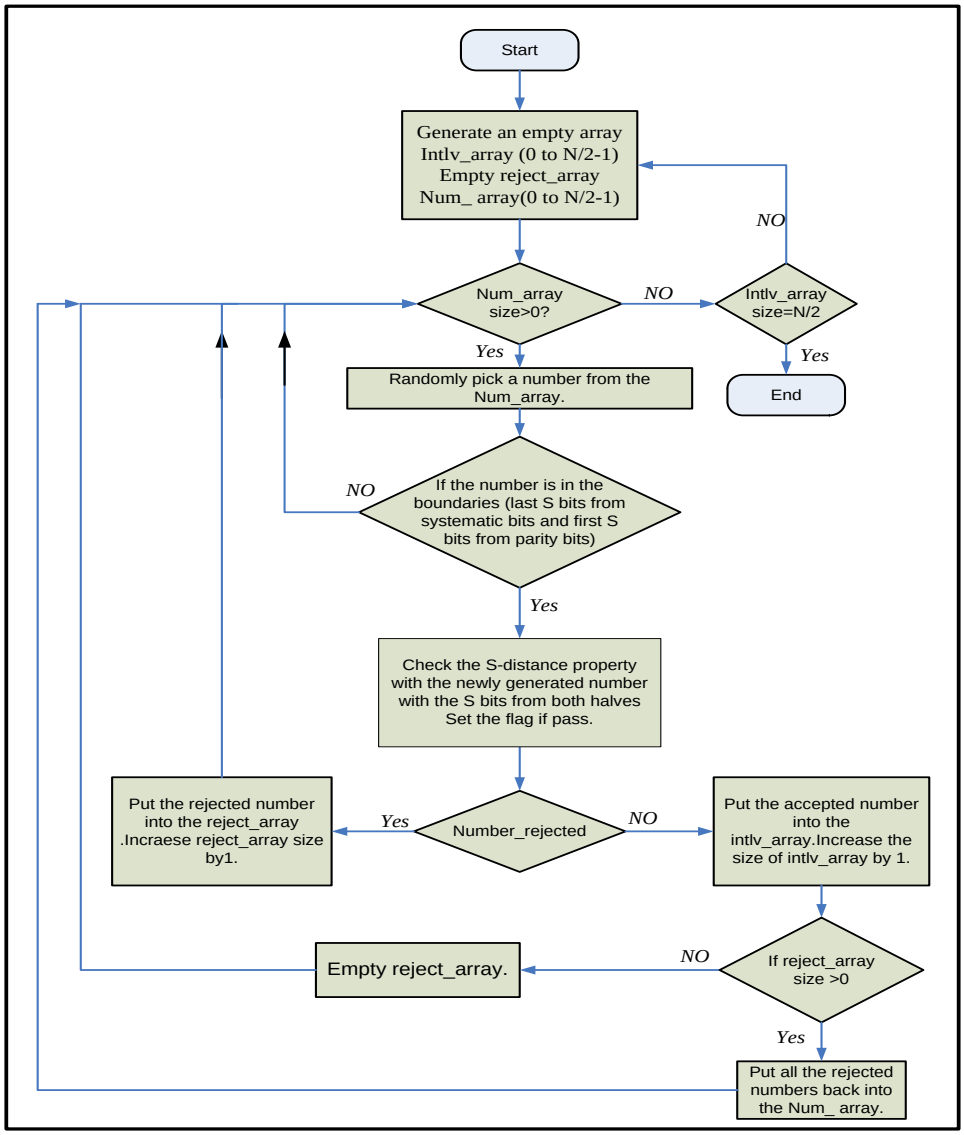


Figure (6) The flow diagram of the Semi-random structured interleaver design

2-2-2 Structured Block interleaver

The block interleaver is one of the frequently used interleavers' types [1,7] .It is implemented in wireless communication systems to break up burst errors that will be encountered by the decoder. Its involves writing information bits in to the rows of an (n*m) matrix bits and reading bits in the column direction of the same matrix that is:

$$\pi(i) = m(i \bmod n) + int[i/n] \quad \text{..... (13)}$$

Deinterleaving is performed simply by writing interleaved bits in column directions and reading them in rows direction that is:

$$\pi^{-1}(j) = n(i \bmod m) + int[j/n] \quad \text{..... (14)}$$

The structured block interleaver is performed as follows. If the input to the interleaver is N-bit including systematic and parity bits, then the design length of structured block interleaver is based on (N/2)-bits and the same resulted interleaving function is applied to both systematic part and parity part of the output of outer encoder. **Figure (7)** shows the input/output position plot for structured block interleaver of length (192) bits for input frame length of (384) bits with interleaving matrix of (16*12).

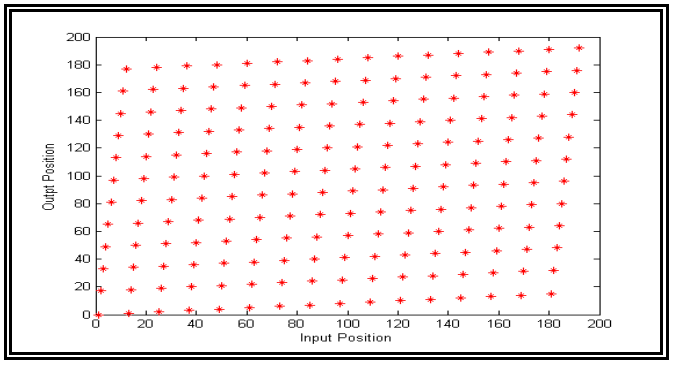


Figure (7) Input/output position plot of 192 bit structured Block interleaver

2-2-3 Structured Circular interleaver

The output of the circular interleaver is based on the following congruence modulo (N):

$$\pi(i) = (aj) \bmod N \quad \text{..... (15)}$$

where, $\pi(i)$ is the output position of an interleaved element, j is the input position of the elements, and $a \le [\sqrt{2N}]$ is the step size which is chosen to be relatively prime to the interleaver size (N). The step size determines the separation between two neighboring elements after interleaving. The structured circular interleaver is performed as follows. If the frame length input to the interleaver (includes systematic and parity bits) is N-bits then the design length of the interleaver will be (N/2) bits, this leads to calculating factor (a) as $a \le \lfloor \sqrt{N} \rfloor$. **Figure (8)** shows the input/output position for (192) bit structured circular interleaver with a=13.

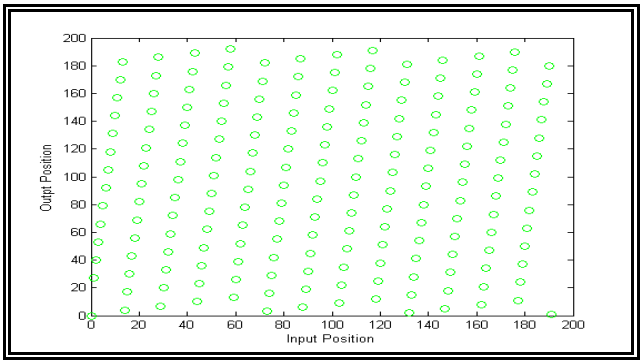


Figure (8) Input/output position plot of 192 bit structured Circular interleaver

3. Complete SISO Algorithm

The core of decoder is the Soft-Input Soft-Output (SISO) modules, The SISO module is a four-port device, with two inputs and two outputs. It accepts as inputs the probability distributions of the information and code symbols labeling the edges of the code trellis, and forms as outputs an update of these distributions based upon the code constraints [8]. The algorithm for the SISO module works on the trellis representation of the code (every code admits a trellis representation), with coding rate of ($R_c=k_o/n_o$) being, k_o and n_o the number of bits forming an input and output code symbols, respectively. In **Fig.(9)** a trellis encoder is presented, which is characterized by the following quantities. Capital letters U,C,S,E will denote random variables and lower case letters (u,c,s,e) their realizations. The subscript (k) will denote a discrete time, defined on the time index set (K). The letters (I, O) will refer to the input and output of the SISO module respectively.

1. $\mathbf{U=(U_k)_{k\in K}}$ is the sequences of input symbols, defined over a time index set (K). Each input symbol U_k consists of k_o bits $\mathbf{U_k^j}$, j=1,2,…, k_o with realization $\mathbf{u^j\in\{0,1\}}$. To the sequence of input symbols, the sequence of a priori probability distributions is associated:

$$\mathbf{P(u;I)=(p_k(u;I))_{k\in K}} \quad \text{.....(16)}$$

where:

$$\mathbf{p_k(u;I)=\prod_{j=1}^{k_o} p_k(u^j;I)} \quad \text{.....(17)}$$

2. $\mathbf{C=(C_k)_{k\in K}}$ is the sequences of output, or code symbols, defined over the same time index set (k). Each output symbol C_k consists of n_o bits $\mathbf{C_k^j, j=1,2,...n_o}$, with realization $\mathbf{c^j\in\{0,1\}}$. To the sequence of output symbols, the sequence of a priori probability distributions is associated:

$$\mathbf{P(c;I)=(p_k(c;I))_{k\in K}} \quad \text{.....(18)}$$

where:

$$\mathbf{p_k(c;I)=\prod_{j=1}^{n_o} p_k(c^j;I)} \quad \text{.....(19)}$$

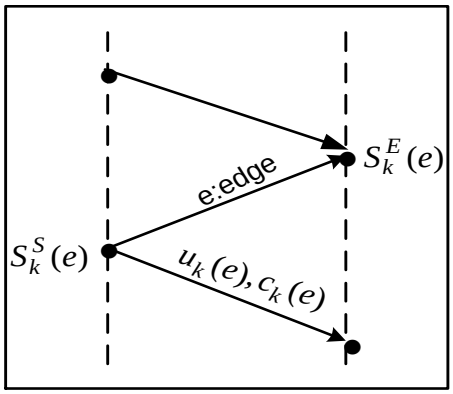


Figure (9) Trellis section in the SISO decoder

Refer to the trellis section notations shown in **Fig.(9)** (presented in Ref. [9]), where the trellis edges are distinguished, and denoted by "e". For each edge, its staring state **(S^S(e))**, its ending state **(S^E(e))**, and the input (uncoded **u(e)**) and output (encoded, C(e)) symbols that label it. The relationship between these functions depends on the particular encoder. As an example, in the case of systematic encoders **(S^E(e), C(e))** also identifies the edge since $u(e)$ is uniquely determined by C (e) [8, 9].

3-1 Generic Encoder/Decoder

A generic encoder and corresponding decoding stage are shown in **Fig.(10)**, where the encoder processes the input symbols (u) in the output ones (C) (as mentioned in section (2-2)); the decoding block receives the current estimation of the probability distributions of the encoder input and output symbols (**P(u;I)** and , **P(c;I)** respectively) and returns new refined values for these distributions [**P(u;O)** and **P(c;O)**]. As the decoding stage manages probability distributions and gives at each iteration reliability information instead of a hard decoding, it is usually called a soft-input soft-output (SISO) decoder.

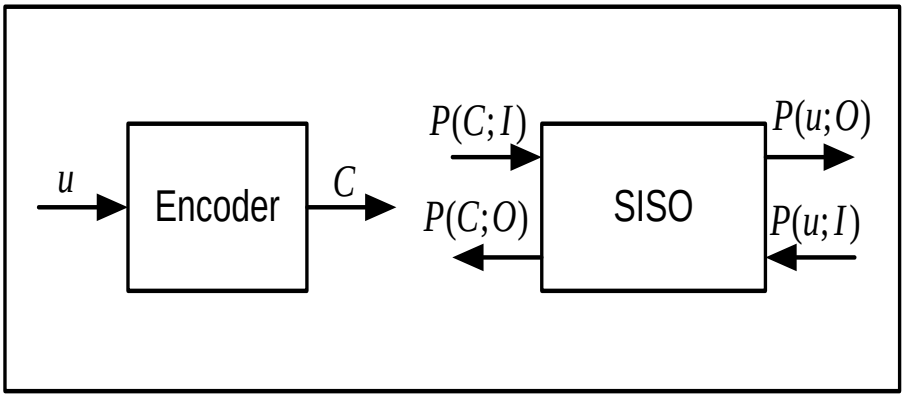


Figure (10) Generic encoder and decoder

According to the original maximum posteriori algorithm (MAP algorithm) presented in [8], the probability distributions obtained as SISO outputs is known as "extrinsic information" can be evaluated as:

$$P_k(c;O) = H_c \sum_{e:c(e)=c} A_{k-1}[S^S(e)]P_k[u(e);I]B_k[S^E(e)] \quad \text{..... (20)}$$

$$P_k(u;O) = H_u \sum_{e:u(e)=u} A_{k-1}[S^S(e)]P_k[u(e);I]B_k[S^E(e)] \quad \text{..... (21)}$$

where, H_C, and, H_u are normalization constants (these constants will be canceled when the values of forward and backward recursions and hence the resultant LLRs are calculated). $\mathbf{A_{k-1}, and, B_{k-1}}$ are equivalent to the path metrics in the Viterbi algorithm [8,9], and represent probability distributions accumulated in the forward and backward directions along the trellis according to the following updating relations:

$$A_k(s) = \sum_{e:S^E(e)=s} A_{k-1}[S^S(e)]P_k[u(e);I]P_k[c(e);I] \quad \text{..... (22)}$$

$$B_k(s) = \sum_{e:S^S(e)=s} B_{k+1}[S^E(e)]P_{k+1}[u(e);I]P_{k+1}[c(e);I] \quad \text{..... (23)}$$

The main modification to the algorithm is required for a practical implementation due to large number of required multiplications, which are eliminated in the additive version of the algorithm, introducing the following definitions:

$$\lambda^c_{k(SISO)}(I) = \log[P_k(c;I)] \quad \text{..... (24)}$$

$$\lambda^u_{k(SISO)}(I) = \log[P_k(u;I)] \quad \text{..... (25)}$$

$$\lambda^u_{k(SISO)}(O) = \log[P_k(u;O)] \quad \text{..... (26)}$$

$$\lambda^c_{k(SISO)}(O) = \log[P_k(c;O)] \quad \text{..... (27)}$$

$$\alpha_k(s) = \log[A_k(s)] \quad \text{..... (28)}$$

$$\beta_k(s) = \log[B_k(s)] \quad \text{..... (29)}$$

The updating equations given above for path and branch metrics take the form:

$$a = \log[\sum_{i}^{L} \exp\{a_i\}] \quad \text{..... (30)}$$

Which gives results very close to [8,9]:

$$a_M \underset{=}{\Delta} \max_i (a_i) \quad \text{...} (31)$$

where, $\max_i (a_i)$ is the maximum value of a_i. This approximation in log domain results in what is called Max-Log-Map algorithm. A recursive correction algorithm is used for improving the performance in the presence of low signal-to-noise ratios (SNR's).

$$a^l = \max(a^{l-1}, a_l) + \log[1 + \exp(-|a^{l-1} - a_l|)] \quad \text{.........................} (32)$$

For $l = 2,...,L$ with $a^1 = a_1$ and $a \equiv a^L$. The algorithm requires the execution of two types of operations: a comparison with maximum selection and the evaluation of the following logarithm:

$$\log[1 + \exp(-\Delta)] ....\Delta \geq 0 \quad \text{...} (33)$$

This is easily implemented as a look-up table. Introducing the operation max* for indicating algorithm (2), the basic APP relations are simplified as follows:

$$\alpha_k(s) = \max_{e:S^E(e)=s}{}^{*} \{\alpha_{k-1}(S^S(e)) + \lambda^u_{k(SIOS)}(I) + \lambda^c_{k(SIOS)}(I)\} \quad \text{.......................} (34)$$

$$\beta_k(s) = \max_{e:S^S(e)=s}{}^{*} \{\beta_{k+1}(S^E(e)) + \lambda^u_{k+1(SIOS)}(I) + \lambda^c_{k+1(SIOS)}(I)\} \quad \text{.......................} (35)$$

where, the initial values of (forward and backward recursions) are given below:

$$\alpha_o(s) = \beta_N(s) = \begin{cases} 0,.s = 0 \\ -\infty,.s \neq 0 \end{cases} \quad \text{...} (36)$$

$$\lambda^c_{k(SISO)}(O) = \max_{e:c(e)=c}{}^{*} \{\alpha_{k-1}(S^S(e)) + \lambda^u_{k+1(SIOS)}(I) + \beta_k(S^E(e))\} \quad \text{..................} (37)$$

$$\lambda^u_{k(SISO)}(O) = \max_{e:u(e)=u}{}^{*} \{\alpha_{k-1}(S^S(e)) + \lambda^c_{k+1(SIOS)}(I) + \beta_k(S^E(e))\} \text{..................} (38)$$

These are the basic relations to be implemented in the decoding stage of a turbo decoder (SISO stage).

4. Proposed Adaptive Iterative Decoder Design

Apart from interleaving and deinterleaving patterns, the soft input soft output (SISO) decoder used in this paper is identical to conventional SCCC decoder [1]. **Figure (11)** shows the channel values ($\lambda^{c}_{k(inr)}(I)$) (where I stands for Input and c, stands for log-likelihood ratio (LLR) for the codeword bits), which fed to the inner decoder from the communication channel .The a priori information of inner decoder is ($\lambda^{\bar{u}}_{k(inr)}(I)$) which is initially set to zero. Extrinsic information messages $\left(\lambda^{\bar{u}}_{k(inr)}(O),\lambda^{\bar{u}}_{k(our)}(O),and,\lambda^{c}_{k(our)}(O)\right)$ in terms of LLR_s are passed back and forth between constituent SISO decoders during iterations. The output from inner SISO is divided into two components: $\left(\lambda^{\bar{u}(Sys)}_{k(inr)}(O),and,\lambda^{\bar{u}(Par)}_{k(inr)}(O)\right)$, hence the LLR information of systematic part is deinterleaved before any of LLR information of parity part is deinterleaved. The resultant LLRs after deinterleaving process $\lambda^{c}_{k(our)}(I)$ consist of two concatenated components $[\lambda^{c(Sys)}_{k(our)}(I),\lambda^{c(Par)}_{k(our)}(I)]$ which can be represented as:

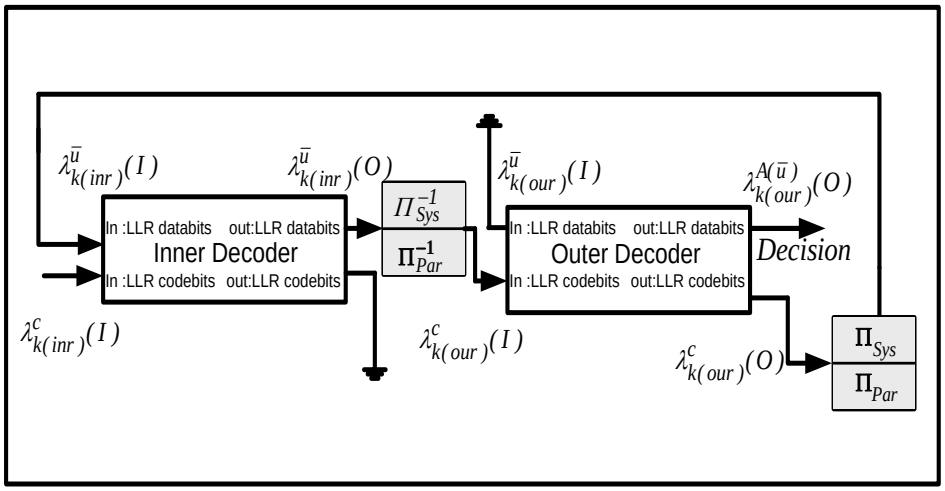


Figure (11) Decoder of adaptive encoder

$$\lambda^{c(Sys)}_{k(our)}(I)=\Pi^{-1}_{Sys}(\lambda^{\bar{u}(Sys)}_{k(inr)}(O)) \quad \text{..... (39)}$$

and

$$\lambda^{c(Par)}_{k(our)}(I)=\Pi^{-1}_{Par}(\lambda^{\bar{u}(Par)}_{k(inr)}(O)) \quad \text{..... (40)}$$

The same scenario is repeated to the output LLR information of codeword bits from outer SISO decoder $\lambda^{(c)}_{k(our)}(O)$ which is also split in to two components $[\lambda^{(c)(Sys)}_{k(our)}(O),\lambda^{(c)(Par)}_{k(our)}(O)]$, these components interleaved by the structured interleaver to produce the corresponding halves of LLR information that concatenated and enters the inner SISO as a priori information named $\lambda^{\bar{u}}_{k(inr)}(I)$ in the next iteration. Hence we can write:

$$\lambda^{\bar{u}(Sys)}_{k(inr)}(I) = \Pi_{Sys}(\lambda^{c(Sys)}_{k(inr)}(O)) \quad \text{..... (41)}$$

and

$$\lambda^{\bar{u}(Par)}_{k(inr)}(I) = \Pi_{Par}(\lambda^{c(Par)}_{k(inr)}(O)) \quad \text{..... (42)}$$

After the final iteration, the data bits are estimated based on the LLR_s of the information bits $\left(\lambda^{A(\bar{u})}_{k(our)}(O)\right)$ output of the outer decoder thus:

$$\bar{u}_k = \begin{cases} 1, if \lambda^{\bar{u}}_{k(our)}(O) \geq 0 \\ 0, if \lambda^{\bar{u}}_{k(our)}(O) < 0 \end{cases} \quad \text{..... (43)}$$

5. Simulation results

The performance of rate (1/3) Parallel concatenated convolutional code from rate 1/4 SCCC with BPSK (Binary Phase Shift Keying) modulator over AWGN channel is shown in **Fig.(12)**. The coding gain is obtained when increasing the number of iterations. However, after 7-iterations, performance improvement slows down and becomes saturated. **Figure (13)** shows the BER performance curves of rate 1/3 PCCC for different structured interleavers, and with different frame lengths (N=128, 512, 1024, and, 2048) it is clear that for N=128 and BER near 10^{-5} the degradation in SNR=0.29dB of the classical rate 1/3 PCCC with block interleaver in comparison to the proposed rate 1/3 PCCC with structured Semi-random interleaver.

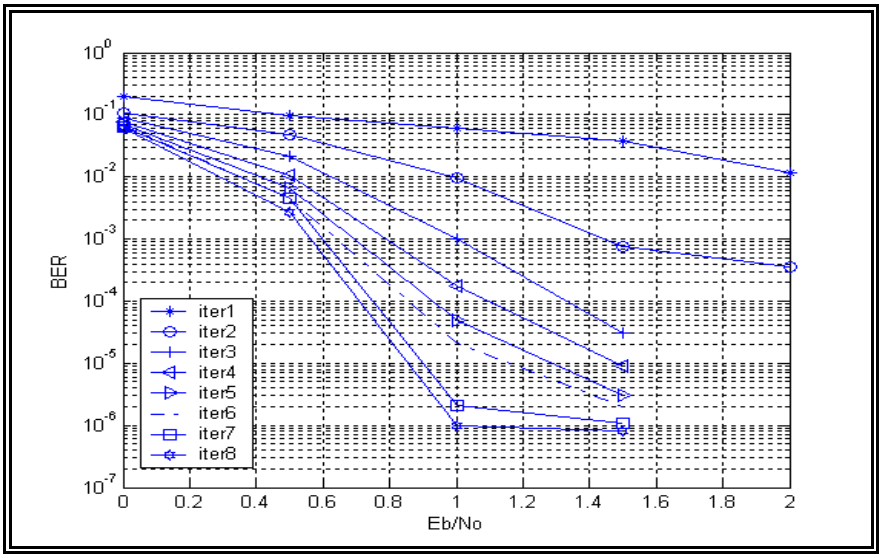


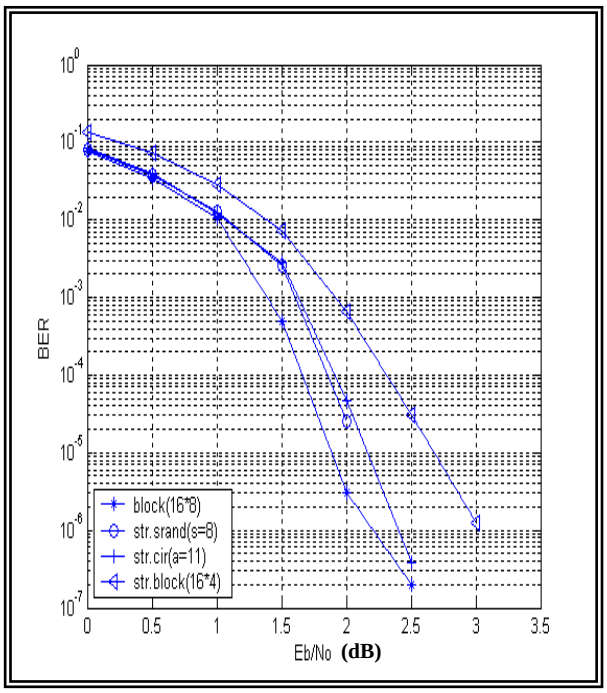
Figure (12) Performance of rate (1/3) PCCC over AWGN channel

Note that at N=512 and BER near 10^{-6} the degradation in SNR is about 0.08dB, while at N=2048 and BER near 10^{-6} there is again about 0.13dB.

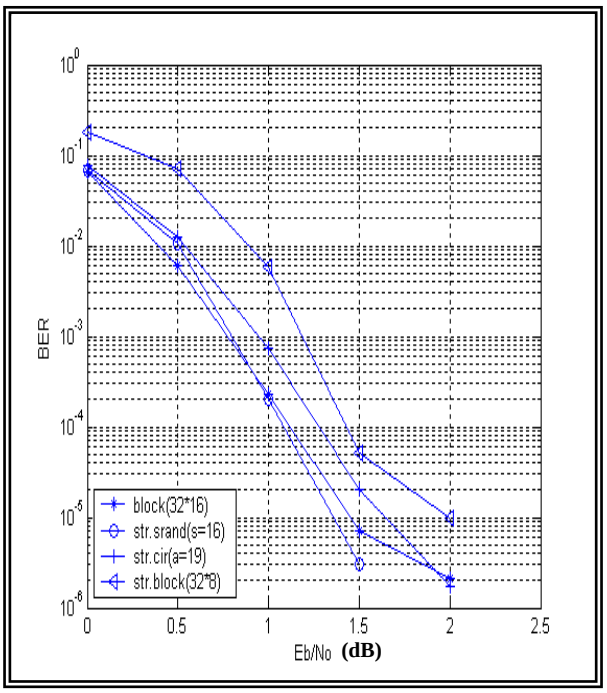
Figure (14) shows the BER performance curves of rate 1/3 PCCC for different structured interleavers, and with different frame lengths (N=128, 512, 1024, and, 2048) over Flat uncorrelated Fading channel. It is clear that for N=128 and BER near 10^{-3} the degradation in SNR=0.45 dB between the classical rate 1/3 PCCC with block interleaver and the proposed rate 1/3 PCCC with structured Semi-random interleaver.

Then at N=512 and BER near10^{-5} the degradation in SNR=0.38dB. While at N=2048 and BER near 10^{-6} there is again of about 0.09 dB. **Figures (15)** to **(17)** shows the original image with it is histogram (Lena), and the reconstructed images at the receiver side over fading channel (2dB) channel SNR for different iterations (3,7,11) with their histograms. ***Appendix B*** gives a source code for calculating the BER used in this research.

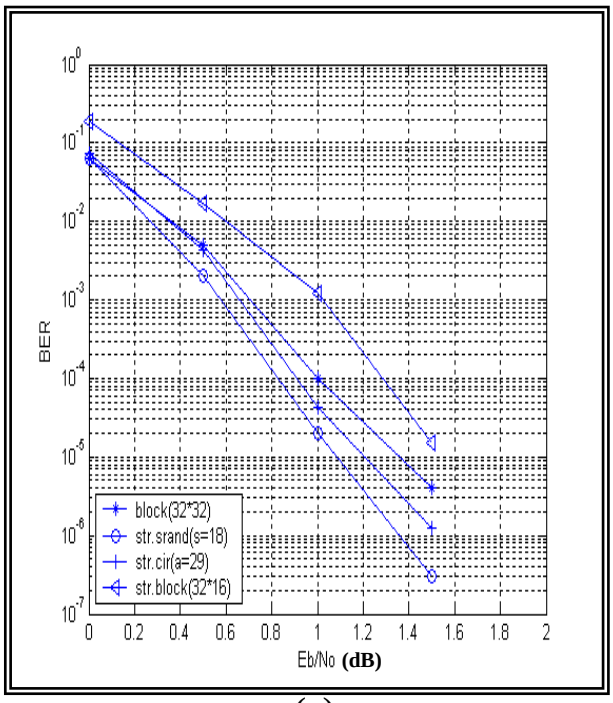
The results show that in the proposed system (PCCC from SCCC), the reconstructed images are with good quality and higher performance than the classical PCCC encoding system.



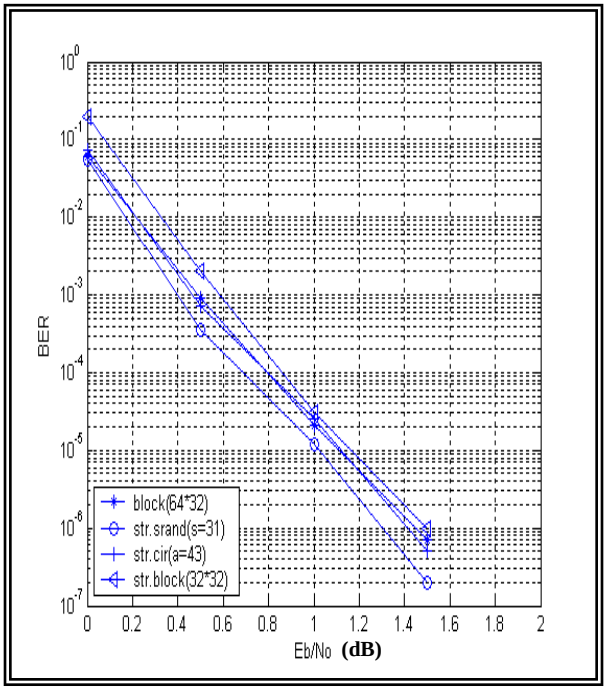
(a)



(d)



(c)



(d)

Figure (13) (a),(b),(c),and (d) Comparison of performance of rate(1/3) PCCC from SCCC for different structured interleavers with different lengths N=128,512,1024,and 2048 respectively over AWGN channel

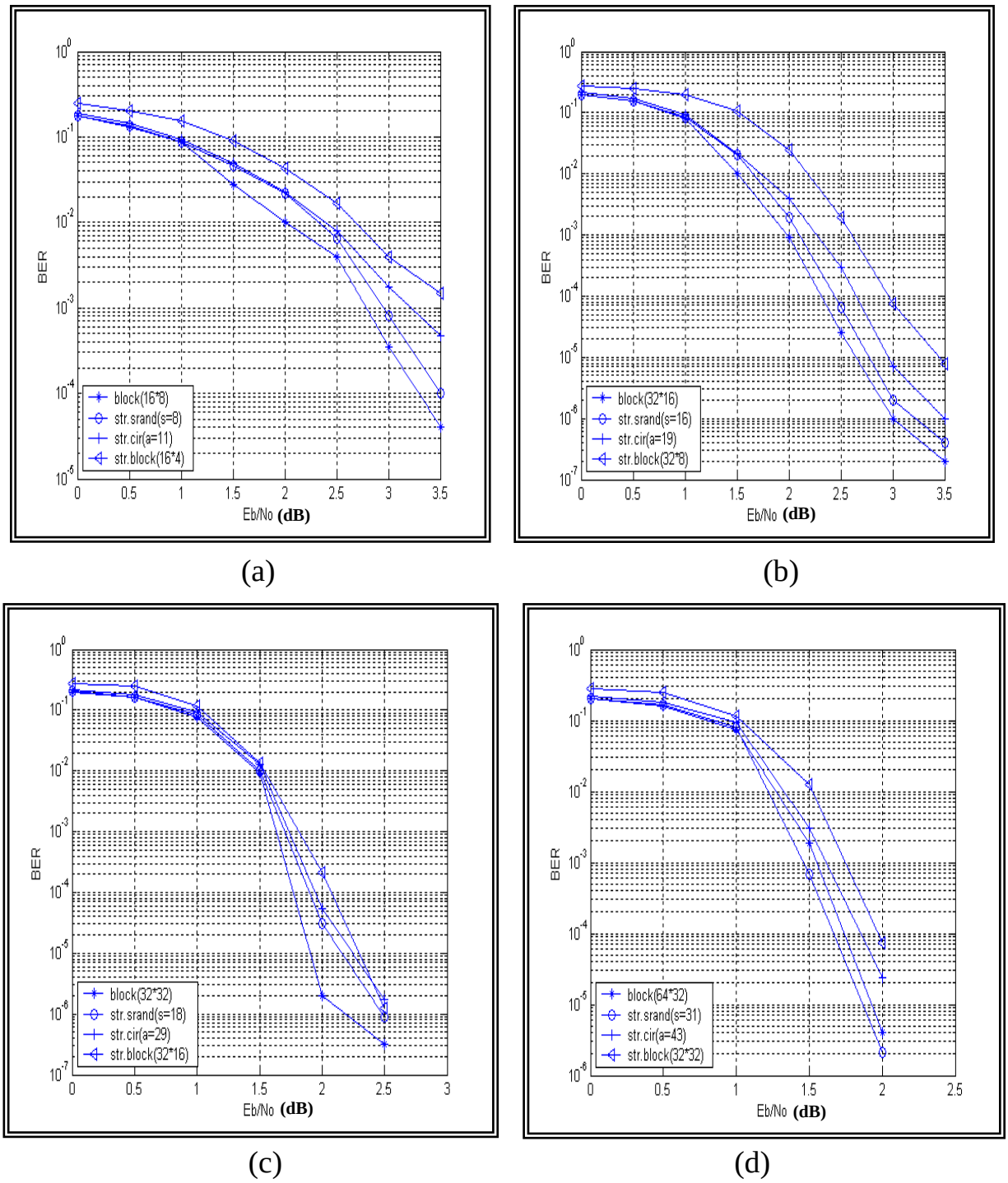
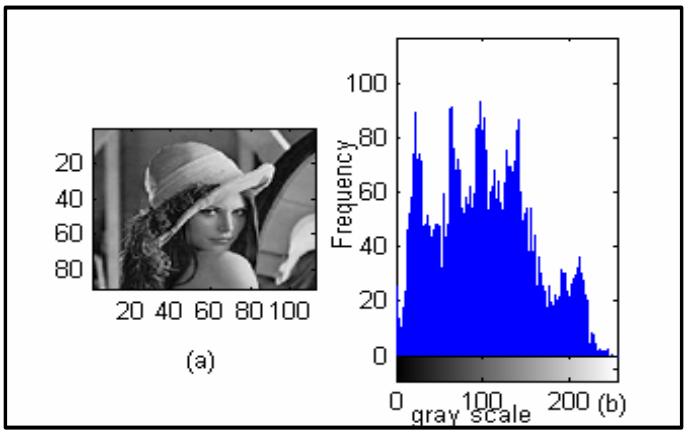


Figure (14) (a),(b),(c),and (d) Comparison of performance of rate(1/3) PCCC from SCCC for different structured interleavers with different lengths N=128,512,1024,and 2048 respectively over Rayleigh fading channel



Figure(15) : Original image with histogram

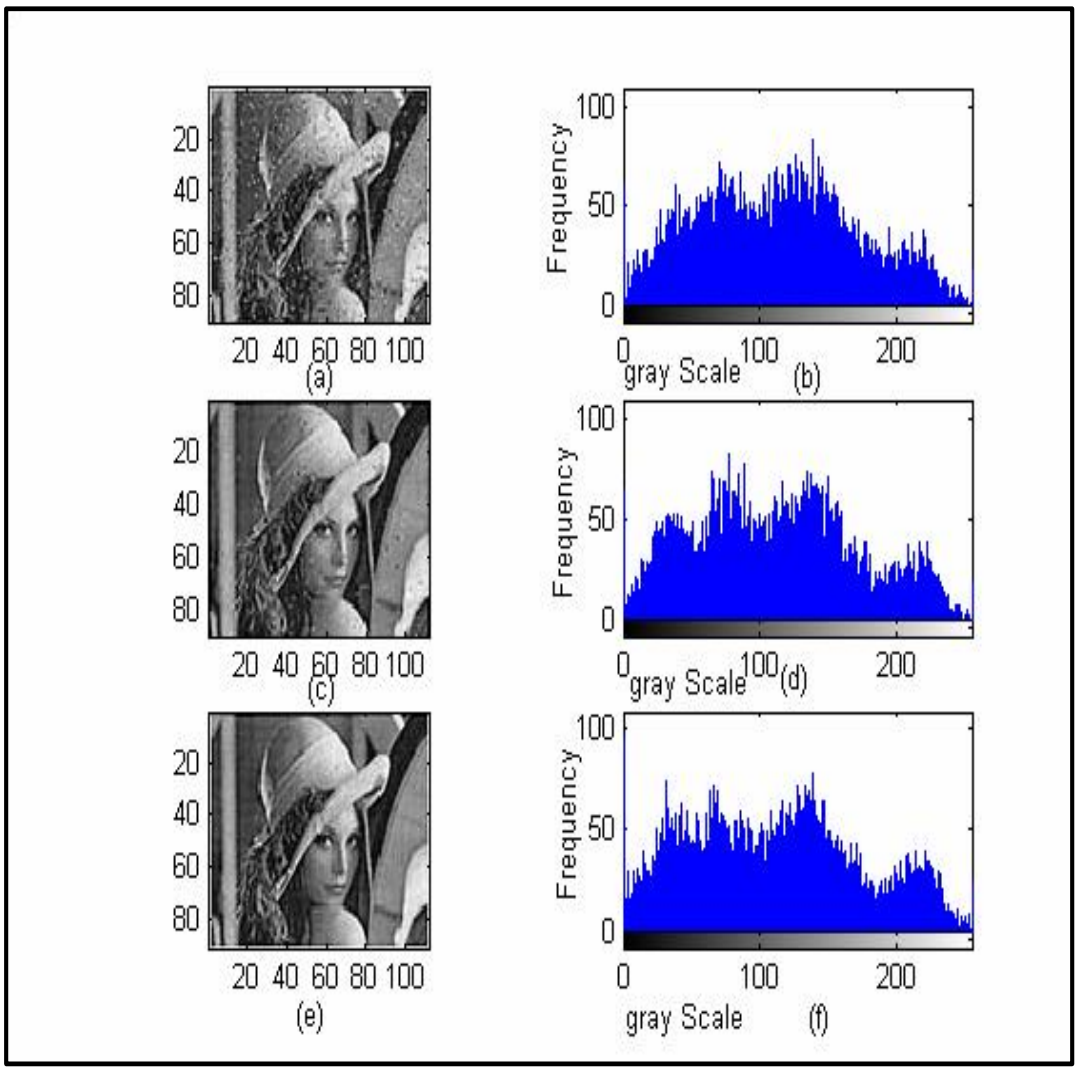


Figure (16) Reconstructed images with histograms after applying proposed adaptive system (PCCC from SCCC)

(a) After third iteration in fading channel SNR=2dB

(b) After seventh iteration in fading channel SNR=2dB

(c) After eleventh iteration in fading channel SNR=2dB

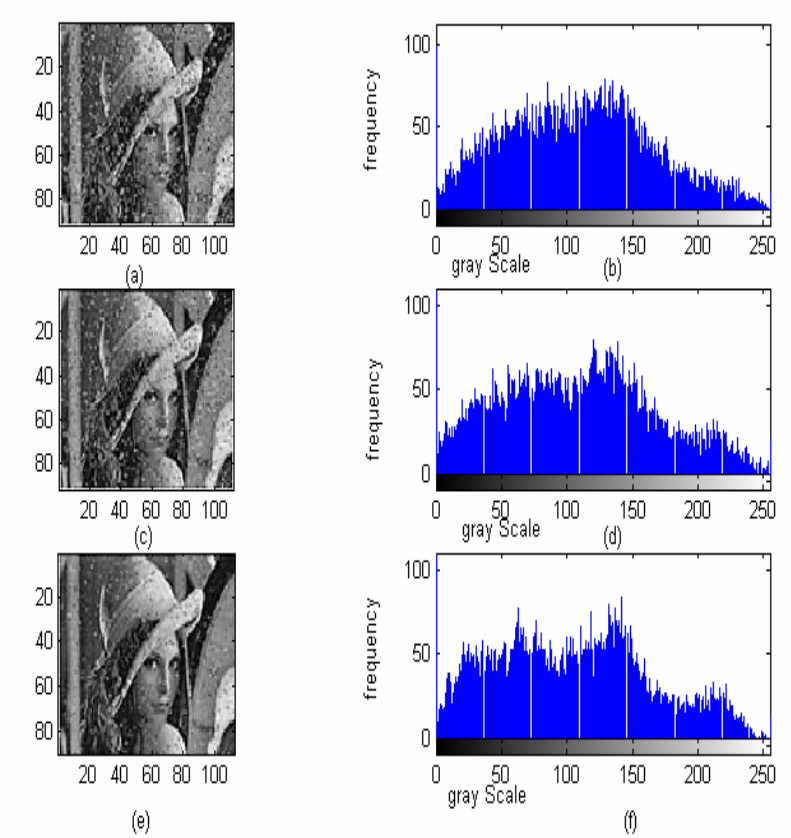


Figure (17) Reconstructed images with histograms after applying classical PCCC

(a) After third iteration in fading channel SNR=2dB
(b) After seventh iteration in fading channel SNR=2dB
(c) After eleventh iteration in fading channel SNR=2dB

6. Conclusions

PCCC is modeled as a special case from SCCC, then the proposed form of SCCC could be considered as adaptive encoder which could be used for both types of concatenated convolutional code. The interleaver restrictions by separating the interleaving/deinterleaving operations of systematic and parity output bits from each encoder with the puncturing mechanism employed in this work which completely suppress the double parity bit stream at the output of proposed encoder leads to generate concatenated convolutional code with a behavior and performance close to that of conventional PCCC with block interleaver. The proposed adaptive (serial/parallel) concatenated convolutional code with Semi-random structured interleaver is the main modification in our presented model shows a better performance over AWGN and Fading channel in comparison to other presented classical but structured interleavers like structured block interleaver, or structured circular interleaver, but its outperformed by the conventional PCCC with block interleaver. However form performance curves it is clear that increasing the frame length results in bridging the gap between conventional PCCC and the proposed PCCC derived from SCCC (Adaptive

Concatenated Convolutional Code). Image called Lena with dimensions of (128*128) and a depth of 8-bit/pixel is encoded and transmitted through the proposed system and the conventional PCCC over Flat Fading channel to test the ability of our system to transmit and receive real information not only random information .Subjectively the received image and decoded after 11th iteration via the proposed system is better than its peer image resulted from, classical (conventional) PCCCs iterative decoder

7. References

1. M. R., Soleymani, and Y., Gao, ***"Turbo Coding for Satellite and Wireless Communications"***, First Edition, Kluwer Academic Publishers, 2002.

2. S., Benedetto, and Divsalar, ***"A Soft-Input Soft Output APP Module for Iterative Decoding of Concatenated Codes"***, IEEE Communication Letters, Vol. 1, No. 1, January 1997.

3. M. S. C., Ho, and S. S., Petrson, ***"Interleavers for Punctured Turbo Coders"***, Institute for Telecommunications Research, University of South Australia,1999.

4. A., Glavieux, C., Berrou, and P., Thitimajshima, ***"Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes"***, In IEEE International Conference on Communications, Pages 1064-10701, May 1993.

5. Le, N. Thang, R. M. A. P., Rajatheva, ***"The Performance of Parallel Concatenated Convolutional Codes (Turbo Codes) And Serial Concatenated Convolutional Codes in AWGN Channel"***, Asian Institute of Technology (AIT), Bankok, Thiland 1999.

6. Y., Wu, and M. C., Valenti, ***"An ARQ Technique using Parallel and Serial Concatenated Convolutional Codes"***, In Proc. IEEE Int. Conference on Communication, Vol. 3, June 2000, pp. 1390-1394.

7. S., Benedetto, and G., Montors, ***"Concatenated Convolutional Codes with Interleavers"***, IEEE Communications Magazine, August 2003.

8. S., Benedetto, D., Divsalar, G., Montorsi, and F., Pollara, ***"Soft-Output Decoding Algorithms for Continuous Decoding of Parallel Concatenated Convolutional Codes"***, In Proc. ICC'96, Dallas, TX, June 1996.

9. Benedetto, S., Divsalar, D., Montorsi, G., and Pollara, F., ***"A Soft-Input Soft-Output Maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes"***, TDA Progress Report, Vol. 42-127, Jet Propulsion Laboratory, Nov. 15 1996.

Appendix A

In the encoder structure, the recursive systematic convolutional encoders are used as a core of encoding process. If we consider a half-rate RSC encoder with m-memory size. If $\bar{\mathbf{u}}_k$ is an input at time (k) the output $\mathbf{x}_k$ is equal to:

$$\mathbf{x}_k = \bar{\mathbf{u}}_k \quad \text{..... (A-1)}$$

Reminder r(D) can be found using feedback polynomial $\mathbf{g^{(0)}(D)}$ and feedback polynomial is $\mathbf{g^{(1)}(D)}$. The feedback variable is:

$$\mathbf{r}_i = \bar{\mathbf{u}}_k + \sum_{j=0}^{K} \mathbf{r}_{k-j} * \mathbf{g}_j^{(0)} \quad \text{..... (A-2)}$$

where, K is called constraint length and the RSC encoder output y_k which called parity data is:

$$\mathbf{y}_k = \sum_{j=0}^{K} \mathbf{r}_{k-j} * \mathbf{g}_j^{(1)} \quad \text{..... (A-3)}$$

RSC used in this paper is with memory (m=2) and rate =1/2 with feedback polynomial $\mathbf{g^{(0)} = 7}$ and feed forward polynomial $\mathbf{g^{(1)} = 5}$,thus g= [111;101] is illustrated in the **Figure (A-1)** and it has a generator matrix:

$$\mathbf{G(D)} = \left[\mathbf{1}, \frac{\mathbf{1+D+D^2}}{\mathbf{1+D^2}}\right] \quad \text{..... (A-4)}$$

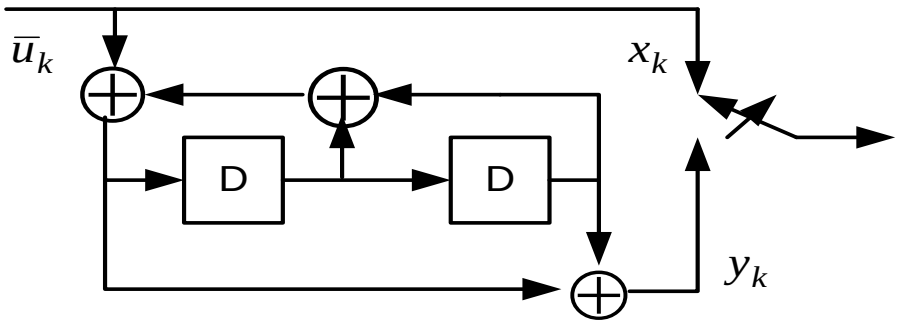


Figure (A-1) Recursive systematic convolutional (RSC) encoder, g=[111,101]

Appendix B

Mechanism of calculating BER

```
for nEN = 1:length(EbN0db)% loop for Eb/N0
%Convert Eb/N0 from unit dB to normal numbers
% reliability value of the channel
 sigma = 1/sqrt(2*rate*en);% standard deviation of AWGN noise
% Clear bit error counter and frame error counter
errs(nEN,1:niter) = zeros(1,niter);
nferr(nEN,1:niter) = zeros(1,niter);
nframe = 0;     % clear counter of transmitted frames
while nferr(nEN, niter)<ferrlim
nframe = nframe + 1;
x = round(rand(1, N-m)); % info. Bits generation
      %random interleaver mapping
      %encoder output (+1/-1)
y = en_output+sigma*randn; % received bits
           for iter = 1:niter
% simulating the functions of inner SISO
% simulating the functions of outer SISO
% Estimate the transmitted information bits
xhat(alpha) = (sign (λ_k(our)^A(ū)(O))+1)/2;
% Number of bit errors in current iteration
         err(iter) = length(find(xhat(1:N)~=u));
% Count frame errors for the current iteration
         if err(iter)>0
            nferr(nEN,iter) = nferr(nEN,iter)+1;
         end
      end %iter

% Total number of bit errors for all iterations
errs(nEN,1:niter)=errs(nEN,1:niter) + err(1:niter);
% Bit error rate
ber(nEN,1:niter)=errs(nEN,1:niter)/nframe/(N-m);
% Frame error rate
fer(nEN,1:niter)=nferr(nEN,1:niter)/nframe;
% Display intermediate results in process
   end%while
end%nEN
```