New Deterministic Interleaver Designs for Turbo Codes

Oscar Y. Takeshita, *Member, IEEE,* and Daniel J. Costello , Jr., *Fellow, IEEE*

***Abstract*—It is well known that an interleaver with random properties, quite often generated by pseudo-random algorithms, is one of the essential building blocks of turbo codes. However, randomly generated interleavers have two major drawbacks: lack of an adequate analysis that guarantees their performance and lack of a compact representation that leads to a simple implementation. In this paper we present several new classes of deterministic interleavers of length N, with construction complexity $O(N)$, that permute a sequence of bits with nearly the same statistical distribution as a random interleaver and perform as well as or better than the average of a set of random interleavers. The new classes of deterministic interleavers have a very simple representation based on quadratic congruences and hence have a structure that allows the possibility of analysis as well as a straightforward implementation. Using the new interleavers, a turbo code of length 16384 that is only 0.7 dB away from capacy at a bit-error rate (BER) of 10^{-5} is constructed. We also generalize the theory of previously known deterministic interleavers that are based on block interleavers, and we apply this theory to the construction of a nonrandom turbo code of length 16384 with a very regular structure whose performance is only 1.1 dB away from capacity at a BER of 10^{-5}.**

***Index Terms*—Capacity, interleavers, permutations, pseudorandom sequences, quadratic congruences, turbo codes.**

I. Introduction

TURBO codes, a class of near channel capacity achieving codes, which were discovered in 1993 by Berrou *et al.* [1], are currently the subject of much research activity. This is not only because their encoding and decoding schemes provide such excellent performance, but also because they use some new methods, distinct from the usual coding theory techniques, for their analysis [2]. In spite of the many variations of turbo codes, all of them contain at least two constituent encoders that are linked (concatenated) by a crucial component: a so-called (random) interleaver. The original scheme presented in [1], for example, uses a pair of rate $1/2$, 16-state, systematic recursive convolutional codes (SRCCs) linked by a pseudorandom interleaver $\mathcal{I}_{65536}$ of length 65536, as shown in the slightly modified description of Fig. 1.

The top SRCC is terminated, i.e., driven to state 0, by a precoder that adds ν tail bits, where ν is the memory of the SRCC, and the other SRCC is simply truncated.[1] Thus turbo codes can be viewed as block codes that are generated by a convolutional encoder. Alternate parity-check bits of each component code are punctured and then multiplexed so that the overall code rate is $1/2$. In the absense of puncturing, the code rate becomes $1/3$. In this paper we will denote turbo codes with the structure shown in Fig. 1 as $C_T(\mathcal{I}_N, (37, 21), 1/2)$, where $\mathcal{I}_N$ is an interleaver of length N (the block length), $(37, 21)$ is the octal notation for the feedback and feedforward polynomials of the SRCCs, respectively, and $1/2$ is the puncturing rate of the parity sequence.

A component code of a turbo code has a given distance spectrum and a given average weight over all its codewords. If a component code contains the all 1 codeword, then the average weight of the code is exactly half the length of the code.[2] Terminated SRCCs cannot have the all 1 codeword. However, if any linear code has a codeword of maximum weight $w_{\max}$, then the average weight of the code is lower-bounded by $w_{\max}/2$, and $w_{\max}$ is lower-bounded by the maximum input weight. The central idea of turbo codes is to statistically manipulate the encoders such that most of the codewords have a weight that is close to the average weight of their component codes by somehow combining the low-weight codewords of one component code with the high-weight codewords of the other component code. The result of this process has been called spectral thinning [4].

In [2] it is shown empirically that a memory 1 terminated SRCC has the property that, for a small even input Hamming weight, the weights of the codewords vary almost uniformly from low weights to high weights, depending on the spacing of the input ones. This wide range of the distance spectrum is a characteristic of terminated SRCCs for all small input weights, unlike nonrecursive convolutional codes that always have small codeword weights for small input weights [5]. Based on this property, it is then easy to understand that a turbo code works by linking the component codes using a random interleaver that changes the relative positions of the input ones to each component code, thus matching low-weight codewords from one component code with high-weight codewords from the other component code with high probability.

An input sequence that generates a low-weight codeword[3] in a component code of a turbo code will be called a "bad" input sequence. Otherwise, an input sequence will be referred to as a "good" sequence. The interleaver has as its objective to transform "bad" input sequences to "good" input sequences,

Manuscript received November 17, 1997; revised June 8, 1999. This work was supported by the National Science Foundation under Grant NCR95-22939 and by the National Aeronautics and Space Administration under Grants NAG5-557 and NAG5-8355.

O. Y. Takeshita is with the Department of Electrical Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: Takeshita.3@osu.edu).

D. J. Costello, Jr. is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: Costello.2@nd.edu).

Communicated by E. Soljanin, Associate Editor for Coding Techniques.

Publisher Item Identifier S 0018-9448(00)06999-6.

[1]This technique is described in [3]. The termination process involves a negligible rate loss for large block sizes.

[2]We assume binary linear codes throughout the paper.

[3]Since the code is systematic, this also implies a low-weight input sequence, but a low-weight input sequence may generate a high-weight codeword because the code is recursive.

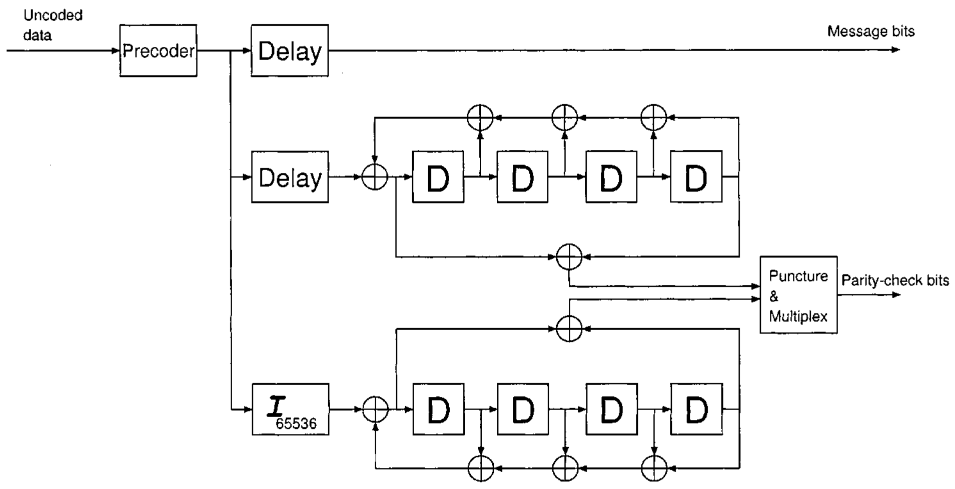

Fig. 1. Block diagram of a turbo encoder.

and it is known that choosing a random interleaver, as currently practiced (at least for long interleavers), yields excellent average performance with high probability; however, some of the "bad" input sequences escape transformation, so there remain a few low-weight codewords that cause what has been called the "error floor" [6], [7]. The "error floor" usually appears in the region of the bit-error rate (BER) curve of turbo codes below 10^{-5}, such that in order to further improve the BER by an order of magnitude, it is necessary to provide about a 1-dB enhancement in the signal-to-noise ratio (SNR), i.e., the "error-floor" region of the BER curve has a much shallower slope than the "waterfall" region.

Summarizing, the "turbo effect," i.e., the high slope of the BER curve in the "waterfall" region and the "error-floor" region at low BERs, is mainly a consequence of two factors [2]: using 1) an SRCC as a component code; and 2) a long random interleaver.

The problem with pseudorandom interleavers, especially for long block lengths, is that we must store the interleaving patterm in tables. It would be preferable to have simple algorithms that generate good interleavers, not only for implementation purposes but also for establishing a common language: practically all papers referring to long pseudorandom interleavers do not describe their generation. One area of current research in turbo codes is the development of methods to lower the "error-floor" region. It is apparent that the "error floor" depends on the interleaver structure [31], [8], and thus it is desirable to know its exact generation. Another problem is that for randomly generated interleavers, simulations indicate a range of performance, where the discrepancy manifests itself especially in the region of the "error floor." This implies that before a random interleaver is used, its performance must be checked by simulation. In this paper, we construct good interleavers that are generated by simple algorithms and with an algebraic structure such that some performance analysis is possible.

The original idea of a turbo code shown in Fig. 1 uses a block length that is too large for some applications. Using shorter interleavers in the same scheme degrades performance but still achieves good coding gains [9]. For short interleavers, say of length less than 1000, the results in [10], [11] indicate that there are some block, and thus deterministic, interleavers of specific sizes and interleaver depths that produce more favorable results than random interleavers. In [12], a class of nonrandom permutations is presented but found to give poor performance; we show that they are in a sense equivalent to block interleavers and thus perform poorly if long block lengths are used. We also show that for short block lengths, they are as good as block interleavers if a suitable interleaver depth is chosen. Further, we give a single description of both interleavers in [10], [12] and determine their important design parameters. Finally, we show that the cycle-length of the feedback polynomial of the component code has a great impact on the BER performance of a turbo code using block interleavers by constructing a surprisingly good nonrandom turbo code of length 16384 based on a block interleaver that is only 1.1 dB away from capacity at a BER of 10^{-5}.

To design an interleaver with a simple construction and with good performance has been considered a difficult task, since most of the currently known work on designing interleavers for turbo codes involves either modifications of random interleavers [3] or heuristic searches based on cost functions [9], [13]–[16]. We believe that part of the difficulty arises from the use of turbo codes with nonoptimum parameter values and, as we will see, that the optimum values depend on the block length of the code.

Although good turbo codes using block interleavers are constructed in this paper, for long block lengths random interleavers are still superior. Our main result is the construction of several new classes of deterministic interleavers with random-like properties that are defined by very simple rules and yet preserve the excellent performance of turbo codes that use long pseudorandom interleavers.[4] Moreover, it is shown by simulation that the new classes of deterministic interleavers perform at least as well as the average performance of a randomly chosen set of interleavers over the entire range of the BER curve. A turbo code of length 16384 based on the new interleavers that is only 0.7 dB away from capacity at a BER of 10^{-5} is constructed. The new classes of deterministic interleavers have an algebraic structure that is based on quadratic congruences. This leads to an anal-

[4]For short block lengths, and among interleavers with a simple construction, block interleavers are usually the best choice.

ysis, for a given interleaver, that guarantees its performance. The construction complexity of the new classes of deterministic interleavers of length N is the smallest possible: $O(N)$.

Randomly generated interleavers are also required in other applications such as spread-spectrum systems [17]. An algebraic approach to constructing pseudorandom interleavers based on a Costas array for spread-spectrum systems is given in [18]. However, our algebraic approach is simpler, since a Costas array would require finding a primitive element of a Galois field, while our method only needs to evaluate a quadratic congruence.

Long random codes, and random-like codes such as turbo codes, are known to have performance close to channel capacity [5]. The results presented in this paper can be viewed as the initial steps in the construction of a class of codes with the performance of random-like codes, but with well-defined and relatively simple building blocks.

The paper is organized as follows. Section II introduces the terminology used throughout the paper. In Section III, we present a unified analysis of known deterministic interleavers for turbo codes based on block interleavers. We also establish new results such as the asymptotic behavior of the BER curves, optimization using a new simplified distance spectrum analysis, and parameter selection for the best interleavers over a wide range of interleaver sizes. In Section IV, we revisit the definition of uniform interleavers given by Benedetto and Montorsi [2] and show that there are several properties that can be deduced from this definition; we then introduce several new classes of deterministic interleavers that have many of the properties of uniform interleavers. In Section V, we present an analysis of the new classes of interleavers. In Section VI we show simulation results of the BER performance of turbo codes using the new interleavers compared with random and block interleavers. Finally, in Section VII, we give some concluding remarks.

II. Terminology

Let $\overline{x} = (x_0, x_1, \ldots, x_{N-1})$ be a sequence in $\{0,1\}^N$. An interleaver $\mathcal{I}_N$ maps $\overline{x}$ to a sequence $\overline{y} = (y_0, y_1, \ldots, y_{N-1})$ such that $\overline{y}$ is a permutation of the elements of $\overline{x}$, i.e., if we consider $\overline{x}$ and $\overline{y}$ as a pair of sets with N elements, there is a one-to-one and onto correspondence $x_i \leftrightarrow y_j$ between every element of $\overline{x}$ and every element of $\overline{y}$. Let $A = \{0, \ldots, N-1\}$; $\mathcal{I}_N$ can then be defined by the one-to-one and onto *index mapping function* $d_{\mathcal{I}_N}\colon A \to A$, $d_{\mathcal{I}_N}\colon i \mapsto j$, $i, j \in A$, and can be expressed as an ordered set called the *permutation vector*

$$\overline{\mathcal{I}}_N = [d_{\mathcal{I}_N}(0), d_{\mathcal{I}_N}(1), \ldots, d_{\mathcal{I}_N}(N-1)].$$

An interleaver is nothing more than an element of the permutation group of N letters, where the group operation is defined by compositions of mappings, i.e., if $\mathcal{F}_N$ and $\mathcal{G}_N$ are interleavers, then a new interleaver $\mathcal{FG}_N(\overline{x}) = \overline{y}$ can be defined by $\mathcal{F}_N(\mathcal{G}_N(\overline{x})) = \overline{y}$. A special kind of permutation $\mathcal{E}_N$ is said to be an L-cycle if and only if it can be defined by an ordered set of distinct elements

$$\mathcal{E}_N \overset{\text{def}}{=} (e_0, e_1, \ldots, e_{L-1})$$

with length $L \le N$, such that its *index-mapping function* $d_{\mathcal{E}_N}$ forms a *chain* of mappings

$$\begin{cases} d_{\mathcal{E}_N}\colon e_m \mapsto e_{m+1 \bmod N}, & 0 \le m < L,\ e_m \in \mathcal{E}_N \\ d_{\mathcal{E}_N}\colon e_i \mapsto e_i, & e_i \notin \mathcal{E}_N \end{cases}$$

where $\operatorname{mod} N$ expresses the usual modulo arithmetic. Note that any cyclic shift of $(e_0, e_1, \ldots, e_{L-1})$ will not change an L-cycle permutation. One may thus choose the canonical format as the one in which the leftmost element is the smallest integer in the chain.

It is known that an element of a permutation group, and hence an interleaver, can be decomposed in a unique way, up to the order in which its disjoint cycles are written [19]. A similar view of interleavers as elements of a permutation group was also presented in [20]. Some interleaver properties have been studied in [21] and additional properties can be found in [22].

Example 2.1: The interleaver $\overline{\mathcal{I}}_9 = [0,2,1,5,3,4,7,8,6]$ is composed of exactly four disjoint cycles (permutations)

$$\mathcal{I}_{9_1} = (0) \Leftrightarrow \overline{\mathcal{I}}_{9_1} = [0,1,2,3,4,5,6,7,8]$$
$$\mathcal{I}_{9_2} = (1,2) \Leftrightarrow \overline{\mathcal{I}}_{9_2} = [0,2,1,3,4,5,6,7,8]$$
$$\mathcal{I}_{9_3} = (3,4,5) \Leftrightarrow \overline{\mathcal{I}}_{9_3} = [0,1,2,5,3,4,6,7,8]$$

and

$$\mathcal{I}_{9_4} = (6,8,7) \Leftrightarrow \overline{\mathcal{I}}_{9_4} = [0,1,2,3,4,5,7,8,6].$$

Note that there are no common elements among the different cycles, i.e, they are disjoint.

It is common in group theory to represent a permutation of N letters by its decomposition into disjoint cycles, which is a very compact notation, e.g., $\mathcal{I}_9$ in Example 2.1 can be represented by

$$\mathcal{I}_9 = \mathcal{I}_{9_2}\mathcal{I}_{9_3}\mathcal{I}_{9_4} = (1,2)(3,4,5)(6,8,7).$$

(1-cycles are usually not written, since it is the identity permutation, and the length of the permutation should be clear from the context.) We then have several equivalent representations for a permutation (interleaver) $\mathcal{I}_N$, namely,

1) index mapping function $d_{\mathcal{I}_N}\colon A \to A$;
2) permutation vector
$$\overline{\mathcal{I}}_N = [d_{\mathcal{I}_N}(0), d_{\mathcal{I}_N}(1), \ldots, d_{\mathcal{I}_N}(N-1)];$$
3) disjoint cycle decomposition $\mathcal{I}_N = (\,)(\,)\ldots(\,)$;

and we shall make use of each of them, depending on the context: 1) The *index-mapping function* representation will be used to study the algebraic structure of interleavers; 2) the *permutation vector* gives the final interleaver format used in turbo codes; and 3) the *disjoint cycle decomposition* is an intermediate representation used in the new construction algorithm.

Throughout the paper, we will denote an interleaver (or a class of interleavers) with the notation

$$\mathcal{I}_{N\colon \pi|\sigma}$$

where the calligraphic symbol may indicate a particular class or type of interleavers, e.g., block interleavers are denoted by the symbol $\mathcal{B}$ rather than $\mathcal{I}$, N is the length of the interleaver, π is a primary parameter that indicates a particular subclass of interleaver, and σ is a secondary parameter that completely defines the interleaver. If notation such as $\mathcal{I}_N$ does not carry the primary and secondary parameters, it should be clear from the context if we are referring to either a class of interleavers or a particular interleaver.

Definition 2.1: An interleaver $\mathcal{I}_N$ has cycle-structure $C\{1_{a_1}, 2_{a_2}, \ldots, N_{a_N}\}$ if it is composed of a_i i-cycles.

Obviously $N = \sum_i ia_i$.

Example 2.2: The interleaver $\overline{\mathcal{I}}_9 = [0,2,1,5,3,4,7,8,6]$ has cycle-structure $C\{1_1, 2_1, 3_2\}$ and $N = 9 = 1 + 2 + 6$.

Definition 2.2: An interleaver is said to "fix" a sequence if its action over a sequence $\overline{x}$ results in the same sequence, i.e., $\mathcal{I}_N(\overline{x}) = \overline{x}$.

Example 2.3: Let us take the interleaver $\mathcal{I}_9$ of Example 2.1. $\mathcal{I}_9$ fixes the sequence $(0,1,1,0,0,0,0,0,0)$, which maps to the same sequence, but does not fix the sequence $(1,1,0,0,0,0,0,0,0)$, which maps to $(1,0,1,0,0,0,0,0,0)$.

Definition 2.3: The order $\mathrm{ord}\,(\mathcal{I}_N)$ of an interleaver $\mathcal{I}_N$ is the smallest positive integer such that for any sequence $\overline{x}$, $\mathcal{I}_N^{\mathrm{ord}\,(\mathcal{I}_N)}(\overline{x}) = \overline{x}$, where the exponent of $\mathcal{I}_N$ is the number of applications of the mapping.

Proposition 2.1: The order of an interleaver $\mathcal{I}_N$ is the least common multiple of the lengths in its cycle structure.

Proof: This follows directly from Definitions 2.1 and 2.3. □

Example 2.4: $\mathcal{I}_9$ in Example 2.1 has order $\mathrm{ord}\,(\mathcal{I}_9) = LCM(1,2,3) = 6$.

Definition 2.4: An interleaver $\mathcal{I}_N$ is said to be CM if $\mathrm{ord}\,(\mathcal{I}_N) = M$. A CM interleaver $\mathcal{I}_N$ is said to be strongly CM if it is composed only of M cycles. Otherwise, it is called weakly CM.

Example 2.5: An interleaver $\mathcal{I}_{2F}$ of size $2F$ is strongly $C2$ if it maps $\overline{x}$ to a sequence $\overline{y}$ such that if $y_j = x_i$, then $x_j = y_i$, $i \neq j$, $i = 1, \ldots, 2F$. That is, strongly $C2$ interleavers swap pairs of elements in the sequence. The disjoint cycle decomposition representation of $\mathcal{I}_{2F}$ has F 2-cycles and

$$\mathcal{I}_{2F} = (e_0^0, e_1^0)\,(e_0^1, e_1^1) \ldots (e_0^{F-1}, e_1^{F-1}).$$

Proposition 2.2: A strongly CN interleaver $\mathcal{I}_{N:CN}$ is composed of a single N-cycle.

Proof: This follows immediately from Definition 2.4. □

Let us now define the polynomial representation of a sequence $\overline{x} = (x_0, x_1, \ldots, x_{N-1})$ of length N by $\overline{x}(D) = \sum_i x_i D^i$. (From now on we shall use these two equivalent notations interchangeably.)

Definition 2.5: For a sequence $\overline{x}$, its signature $\mathrm{sig}\,(\overline{x})$ is the result of a right cyclic shift of $\overline{x}$ so that the degree of its polynomial representation is minimized. Moreover, the smallest integer necessary to minimize the degree is called the shift $\mathrm{sh}\,(\overline{x})$ of $\overline{x}$.

A sequence $\overline{x}$ is thus completely and uniquely characterized by its signature $\mathrm{sig}\,(\overline{x})$ and its shift $\mathrm{sh}\,(\overline{x})$.

Example 2.6: For the sequence $\overline{x} = (1,0,0,1,0,0,0)$, $\mathrm{sig}\,(\overline{x}) = (1,0,0,1,0,0,0)$ and $sh(\overline{x}) = 0$.

Example 2.7: For the sequence $\overline{x} = (1,0,0,0,0,0,1)$, $\mathrm{sig}\,(\overline{x}) = (1,1,0,0,0,0,0)$ and $\mathrm{sh}\,(\overline{x}) = 1$.

Proposition 2.3: The set of sequences of a given length N and a given signature $\mathrm{sig}\,(\overline{x})$ form an equivalence class, where $\mathrm{sig}\,(\overline{x})$ can be taken as the representative of smallest degree.

Proof: This follows directly from the definitions of signature and shift, where a given equivalence class is formed by all shifts of a given signature, and it is immediate that sequences with different signatures cannot belong to the same equivalence class. □

Definition 2.6: The size of an equivalence class $|\mathrm{sig}\,(\overline{x})|$ of a given sequence $\overline{x}$ of length N is given by the smallest positive integer i such that $\overline{x}(D)D^i \equiv \overline{x}(D)(\mathrm{mod}\, D^N)$.

Proposition 2.4: A sequence $\overline{x}$ of length N and weight relatively prime to N has the size of its equivalence class $|\mathrm{sig}\,(\overline{x})| = N$.

Proof: The proof follows directly from Proposition 2.3 and Definition 2.6. □

Definition 2.7: The delay $\mathrm{del}\,(\overline{x})$ of a sequence $\overline{x}$ is the degree of the term of smallest degree in $\overline{x}(D)$.

Definition 2.8: The span $\mathrm{sp}\,(\overline{x})$ of a sequence $\overline{x}$ is defined as $\mathrm{sp}\,(\overline{x}) = \deg\,(\overline{x}) - \mathrm{del}\,(\overline{x}) + 1$, where $\deg$ is the degree of its argument.

Definition 2.9: Let τ be the cycle-length of the feedback polynomial of an SRCC. The parity density P_d of an SRCC is defined as the ratio of the number of ones in a single period of the impulse response of the SRCC divided by τ.

Example 2.8: The impulse response of the SRCC with generators $(37, 21)$ in Fig. 1 is

$$(1,1,0,0,1,0,1,0,0,1,0,1,0,0,1,0,1,\ldots)$$

the feedback polynomial has cycle-length $\tau = 5$, and therefore the parity density is $P_d = 2/5$.

Example 2.9: The impulse response of an SRCC with generators $(23, 35)$ is

$$(1,1,1,1,1,0,0,0,1,0,0,1,1,0,1,0,1,1,\\ 1,1,0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,\ldots)$$

and its parity density is $P_d = 8/15$, since the feedback polynomial $1 + D^3 + D^4$ has cycle-length $\tau = 15$. This SRCC has a primitive feedback polynomial. It is well known that the impulse response of an SRCC with a primitive feedback polynomial has period $2^\nu - 1$, where ν is the constraint length of the SRCC, and that the number of ones in a period is $2^{\nu-1}$ [23].

It has been shown that a weight-2 input sequence of the form

$$\overline{x}_\tau(D)D^s \triangleq (1 + D^\tau)D^s$$

applied to an SRCC with cycle-length τ, produces a low-weight parity sequence [2] [4].

Proposition 2.5: [5] If the feedback polynomial of an SRCC is primitive, then for weight-2 input sequences of the form $\overline{x}(D) = (1 + D^{\tau u})D^s$, $s \geq 0$, we have a number of nonzero parity bits w_p that satisfies $w_p = P_d\tau = u$ or $w_p = P_d\tau u + 2$, for any positive integer u. (This is also valid if the feedback polynomial

[5]See also [24] and [23].

is not primitive and has the property that the periodic part of the impulse response starts no later than the second parity bit.)

Proof: This follows from the fact that the periodic part of the impulse response of an SRCC with a primitive feedback polynomial starts no later than the second parity bit. □

Example 2.10: A weight-2 input sequence of the form $\overline{x}(D) = (1 + D^5)D^s$ applied to the SRCC $(37, 21)$ produces the following parity sequence:

$$(\ldots, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, \ldots)$$

i.e., the weight of the parity sequence is $4 = 0.4 \times 5 + 2$. A weight-2 input sequence of the form $\overline{x}(D) = (1 + D^{10})D^s$ generates the parity sequence

$$(\ldots, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0 \ldots)$$

with weight $6 = 0.4 \times 10 + 2$.

Turbo codes are often called parallel concatenated codes [2] because of the positioning of the component codes. Recalling Fig. 1, however, we can also interpret the precoder as a very high rate outer block code similar to the usual notion of a concatenated code [25]. We may thus consider a turbo code as a hybrid concatenation scheme.

The task of the precoder is to drive the top component code to the 0 state at the end of the data block. This is equivalent to the following proposition.

Proposition 2.6: Let $\overline{d}$ be a data block of length $N - \nu$. The output $\overline{x}$ of the precoder must be divisible by the feedback polynomial of the top component code.

Proof: This follows from the definition of the precoder. □

Definition 2.10: The estimated codeword weight $\tilde{w}(\overline{x})$ for an input sequence $\overline{x}$ to an SRCC is defined as

$$\tilde{w}(\overline{x}) = \begin{cases} (N - \mathrm{del}\,(\overline{x}))P_pP_d + w_H(\overline{x}), & \text{if } P(D) \nmid \overline{x}(D) \\ \mathrm{sp}\,(\overline{x})P_pP_d + w_H(\overline{x}), & \text{if } P(D) = |\overline{x}(D) \end{cases}$$

where $w_H(\overline{x})$ is the Hamming weight of $\overline{x}$, $P(D)$ is the feedback polynomial of the SRCC, P_p is the puncturing rate of the parity sequence, and P_d is the parity density of the SRCC.

Now we claim without formal proof that a sensible definition of a "bad" input sequence for a component code is an input sequence with low estimated codeword weight. Using Definition 2.10, a "bad" input sequence $\overline{x}_{\mathrm{bad}}$ for a component code can be classified into two types:

Type E: Input sequences with a very high delay that are not divisible by $P(D)$.

Type I: Input sequences with a small span that are divisible by $P(D)$.

However, the following example shows that some combinations of "bad" input sequences lead to input sequences that do not have low estimated codeword weight, and thus estimated codeword weight alone cannot be used to classify "bad" sequences.

Example 2.11: Let $\overline{x}_I(D)$ be a Type I input sequence. An input sequence of low weight, such as $\overline{x}_I(D) + \overline{x}_I(D)D^s$, with a sufficiently large s, does not have low estimated codeword weight because of its large span $\mathrm{sp}\,(\overline{x}_I(D) + \overline{x}_I(D)D^s) \geq s$.

This leads to the following claim.

Claim 2.7: All "bad" input sequences are of Type E or Type I, or a linear combination of Type E or Type I sequences. A linear combination of two Type I sequences cannot be of Type E and is not necessarily of Type I.

Argument. The claim follows from the definition of estimated codeword weight and from Example 2.11. □

Type E and Type I input sequences can thus be viewed as elementary types of "bad" input sequences. However, some "bad" sequences, such as the one in Example 2.11, that result from a linear combination of elementary "bad" sequences, do not necessarily have low estimated codeword weight $\tilde{w}(\overline{x})$.

The estimated codeword weight $\tilde{w}_c(\overline{x})$ of a code sequence generated by an input sequence $\overline{x}$ in a turbo code with an interleaver $\mathcal{I}_N$ then becomes

$$\tilde{w}_c(\overline{x}) = \tilde{w}(\overline{x}) + \tilde{w}(\mathcal{I}_N(\overline{x})) - w_H(\overline{x}). \qquad (1)$$

Note that the input sequences to the top component code always are divisible by $P(D)$ because they are outputs of the precoder. We, therefore, have a low-weight codeword in a turbo code when the top component code has an input sequence of Type I and the lower component code has an input sequence either of Type E or of Type I.

Because of its simplicity, (1) is a very convenient way to evaluate the expected distance spectrum of a turbo code, and for weight-2 input sequences it provides a very good approximation to the actual distance spectrum. The emphasis on weight-2 input sequences is consistent with the turbo code literature, where it has been noted that most of the low-weight codewords are caused by "bad" weight-2 input sequences [2], [4].

III. Block and Linear Interleavers

Before introducing our major result, we present in this section a generalization of previously known results on deterministic interleavers with simple algebraic descriptions. This unified treatment lays the groundwork for the introduction of serveral new classes of deterministic interleavers in the next section. We also present a few new results on block interleavers. Finally, we note that for short block lengths, heuristic constructions without a particular algebraic structure may result in very good performance. Some of these heuristic methods are described in [13], [14].

A. The Angular Coefficient of Linear Interleavers

It is known that block interleavers can effectively break up bad weight-2 sequences, but not bad weight-4 sequences. A pictorial explanation of why this happens was shown in [4], [12]. In this section, we give a new interpretation of why a block interleaver does not work on the basis of what we call its angular coefficient.

Block interleavers $\mathcal{B}_N$ are usually defined by rectangular matrices of size $N = m \times n$ that are written row-wise and read column-wise. The *index mapping function* $d_{\mathcal{B}_N}$ of a block interleaver $\mathcal{B}_N$ is of the form

$$d_{\mathcal{B}_N}(i) \equiv ni + \lfloor i/m \rfloor \pmod{N}, \qquad 0 \leq i < N. \qquad (2)$$

To analyze a block interleaver, we shall use a slightly different interleaver $\mathcal{L}_N$ that "linearizes" the "floor" function $\lfloor\cdot\rfloor$ in (2).

In [12], the interleaver $\mathcal{L}_N$ is called a *nonrandom permutation based on circular shifting*. Its *index mapping function* is

$$d_{\mathcal{L}_N}(i) \equiv ki + v \pmod N, \qquad 0 \le i < N \tag{3}$$

where k is a fixed integer relatively prime to N and v is a fixed integer.

The fact that $GCD(N, k) = 1$ in (3) guarantees that the linear congruence in i has exactly one solution [26, p. 92] (the function is then injective), a role played by the "floor" function in the *index mapping function* $d_{\mathcal{B}_N}$ of a block interleaver. The inverse *index mapping function* $d_{\mathcal{L}_N}^{-1}$ is given by

$$d_{\mathcal{L}_N}^{-1}(j) \equiv \frac{1}{k}\,(j - v) \pmod N, \qquad 0 \le j < N$$

which is a well-defined function because $\frac{1}{k} \equiv l \pmod N$ for some unique integer $l \pmod N$. We can see that $GCD(N, l)$ must be 1, and thus the inverse mapping function $d_{\mathcal{L}_N}^{-1}$ is also injective and $\mathcal{L}_N$ is indeed a permutation of N letters.

Interleavers $\mathcal{B}_N$ and $\mathcal{L}_N$ cannot be exactly the same by definition, but they have much in common: they are both based on linear congruences, and hence they have essentially the same properties, e.g., two elements separated by a constant $t \pmod N$ in an input sequence almost always map to output sequences with the corresponding elements separated by a fixed constant $tn \pmod N$ for $\mathcal{B}_N$ interleavers, and they always map to output sequences with the corresponding elements separated by a fixed constant $tk \pmod N$ for $\mathcal{L}_N$ interleavers. The constant k is called the *angular coefficient* of an $\mathcal{L}_N$ interleaver, an analogy to the angular coefficient of a linear curve. For a block interleaver, the angular coefficient is the interleaver depth. Henceforth, we will refer to both $\mathcal{B}_N$ and $\mathcal{L}_N$ as linear interleavers. Note that many of the interleavers with simple constructions that have been proposed in the literature fall in this category [27], [28]. The interleaver in [28] is a variation of the helical interleaver in [29] and has the interesting property that it terminates the trellis of the lower component code to the same state as the top component code.

For an interleaver $\mathcal{L}_N$, since by definition $GCD(N, k) = 1$, the following congruence is always solvable in t with a unique solution

$$tk \equiv \tau \pmod N$$

where τ is the cycle-length of the SRCC. This means that if we have an input sequence of the form $\overline{x}_i(D) = (1 + D^t)D^q$, then the interleaver $\mathcal{L}_N$ transforms it to a "bad" sequence of the form $(1 + D^\tau)D^s = \overline{x}_\tau(D)D^s$, which produces a low-weight parity sequence for the second component code. For example, we can conclude from Example 2.10 that the sequence obtained by puncturing every other bit of the parity sequence generated by an input sequence of the form $\overline{x}(D) = (1 + D^5)D^s$ applied to the SRCC $(37, 21)$ code has weight 2. Now if we have a weight-4 input sequence of the form

$$\overline{x}_{\text{bad4}} = \overline{x}_i(D) + \overline{x}_i(D)D^\tau = (1 + D^t)D^q + (1 + D^t)D^{q+\tau}$$

we have low-weight parity sequences in both component codes. Moreover, each of the N possible cyclic shifts of $\overline{x}_{\text{bad4}}$ produces the same result.[6] Hence a weight-4 input sequence of the form

$$\overline{x}_{\text{bad4}} = (1 + D^t)D^q + (1 + D^t)D^{q+\tau}$$

in the turbo code $C_T(\mathcal{L}_N, (37, 21), 1/2)$, produces a parity sequence with weight 8 and thus a codeword with weight 12. In [2], the following approximate expression for the BER $P_b(e)$ is derived:

$$P_b(e) \approx \frac{1}{2}\sum_{w_c} D_{w_c}\,\text{erfc}\left(\sqrt{w_c \frac{R_c E_b}{N_o}}\right) \tag{4}$$

where

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

w_i is the weight of the input sequence, w_p is the weight of the parity sequence, and A_{w_i, w_p} is the number of codewords with input and parity weights w_i and w_p, respectively. If we take only the terms in (4) corresponding to weight-4 input sequences of the form $\overline{x}_{\text{bad4}} = (1 + D^t)D^q + (1 + D^t)D^{q+\tau}$, we have

$$P_b(e|\overline{x}_{\text{bad4}}) \approx \frac{1}{2}\,\frac{4}{N} N\text{erfc}\left(\sqrt{12 \frac{0.5E_b}{No}}\right)$$

which simplifies to

$$P_b(e|\overline{x}_{\text{bad4}}) \approx 2\text{erfc}\left(\sqrt{6 \frac{E_b}{No}}\right) \tag{5}$$

which is independent of the interleaver length N, i.e., there is no interleaver gain associated with this term in the approximation of the BER curve given by (4).

We can now classify the following types of "bad" input sequences for a turbo code using a linear interleaver $\mathcal{L}_N$:

Globally-bad: If the input sequences are of Type I (or a combination of Type I sequences) for both component codes. This implies that the multiplicity of such sequences is roughly on the order of N, thus causing a "spike" in the distance spectrum, since every shift of the input sequence produces the same low-weight parity sequence (disregarding edge effects).

Locally-bad: If the input sequence for the top component code is of Type I and for the lower component code is of Type E. In this case, we can obtain other "bad" input sequences by shifting the lower Type E input sequence to the left a number of times without the top Type I input sequence changing its type. This process results in a sequence of codewords with a small increasing weight. This can be identified in the distance spectrum by a "door" function starting at a low weight.

The input sequence $\overline{x}_{\text{bad4}}$ considered in this section is thus *globally-bad*, and it cannot be avoided by simply changing the angular coefficient. Linear interleavers are prone to other *globally-bad* input sequences, particularly of weight 2, but these can be avoided by using a technique explained in the next section.

B. Optimization of the Angular Coefficient

Since (5) is valid for the turbo code $C_T(\mathcal{L}_N, (37, 21), 1/2)$, using any linear interleaver $\mathcal{L}_N$, any such code will have BER performance no better than the curve defined by (5). We now confirm by simulations previously known results [10], [11] that

[6]We suppose that the size of the equivalence class of $\overline{x}_{\text{bad4}}$ is N, and we disregard edge effects.

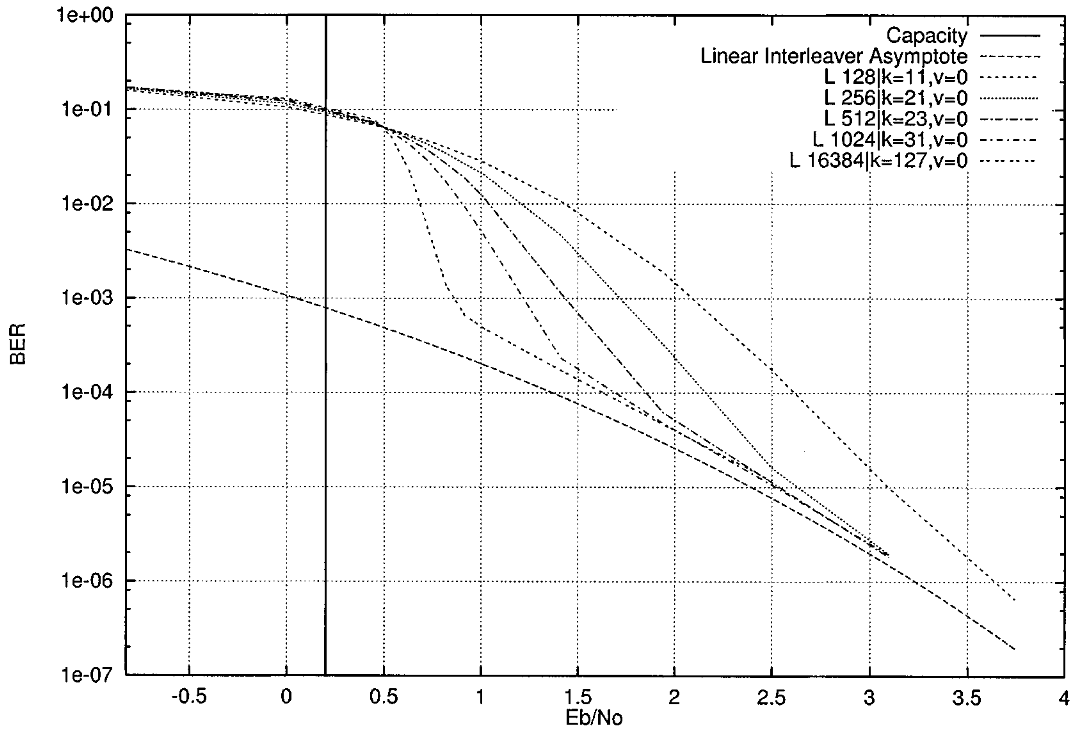

Fig. 2. The BER performance of the best linear interleavers found for the turbo code $C_T(\mathcal{L}_N, (37, 21), 1/2)$.

the performance of linear interleavers is dependent on their angular coefficient (interleaver depth); moreover, we establish a new result that for a wide range of interleaver lengths and for a particular choice of the angular coefficient, we obtain BER curves that have the same asymptote as (5). In Fig. 2, we present simulations showing the best linear interleavers that we found for a wide range of interleaver lengths along with the linear interleaver asymptote of (5).

A linear interleaver $\mathcal{I}_N$ with *index mapping function* $d_{\mathcal{I}_N}: i \mapsto j$ can be represented by N points in the (i, j)-plane. We show in Figs. 3 and 4 the interleavers $\mathcal{L}_{128|k=11,v=0}$ and $\mathcal{L}_{128|k=63,v=0}$. Note that the points in both Figs. 3 and 4 form a lattice. However, the configuration of points for the interleaver $\mathcal{L}_{128|k=11,v=0}$ represented in Fig. 3 is closer to that of a random interleaver, whose points are typically spread randomly throughout the (i, j)-plane.

In [12] it is stated that an angular coefficient of $\sqrt{2N}$ (or an integer value close to it) guarantees that the minimum value of the sum of the distance between the projection of two lattice points on the i–axis and the distance between the projection of the same two points on the j–axis is approximately $\sqrt{2N}$. However, it can be deduced from the interleaver parameters of Fig. 2 that a good choice for the angular coefficient for several interleaver lengths is approximately $\sqrt{N}$; this produces a good scattering of the points in the graphical representation of the interleaver, i.e., there is a large minimum distance between the points in the lattice, thus ensuring a large minimum distance for the weight-2 input sequences of turbo codes using this interleaver.[7] The size of the 21×19 interleaver in [10] is consistent with this design parameter. However, since an interleaver size of 20×20 is known to be noticeably worse [10] than a size of 21×19, other factors must be considered.

An additional factor to consider is the cycle-length τ of the component code. Thus one should not only maximize the sum of the distances in the previous paragraph for weight-2 input sequences, but should also avoid the situation that the distances between the projections of two lattice points on the i– and j–axes are both small multiples of τ, i.e., avoid *globally-bad* input sequences. In the optimization process, we will also want to avoid *locally-bad* input sequences. The optimization is done by examining the approximate distance spectrum of the code due to Type I weight-2 input sequences by using (1) and then evaluating (4) for angular coefficients on the order of $\sqrt{N}$. This can be carried out by a computer program that examines Type I weight-2 input sequences by computing (1) for all weight-2 sequences separated by small multiples of the cycle-length τ of the component SRCC.

In Figs. 5 and 6, we show the estimated distance spectrum for a set of linear interleavers of block length 128 and for a random interleaver of the same size. Note that the existence of *locally-bad* and *globally-bad* input sequences is manifested by the appearance of "doors" and "spikes," respectively, in the distance spectrum curves. For example, in Fig. 5, a "spike" occurs at weight 10 for the $\mathcal{L}_{128|k=7,v=0}$ interleaver and a "door" starts at weight 8 for the $\mathcal{L}_{128|k=11,v=0}$ interleaver. (Fractional weights appear in the estimated distance spectrum because of the puncturing rate described in Definition 2.10.) Although we do not show this in the figures, the amplitudes of the spikes are on the order of N.

[7]An approximate hexagonal lattice is obtained if $k \approx \sqrt{2\sqrt{3}N/3}$; however, this may not be the best value for the angular coefficient.

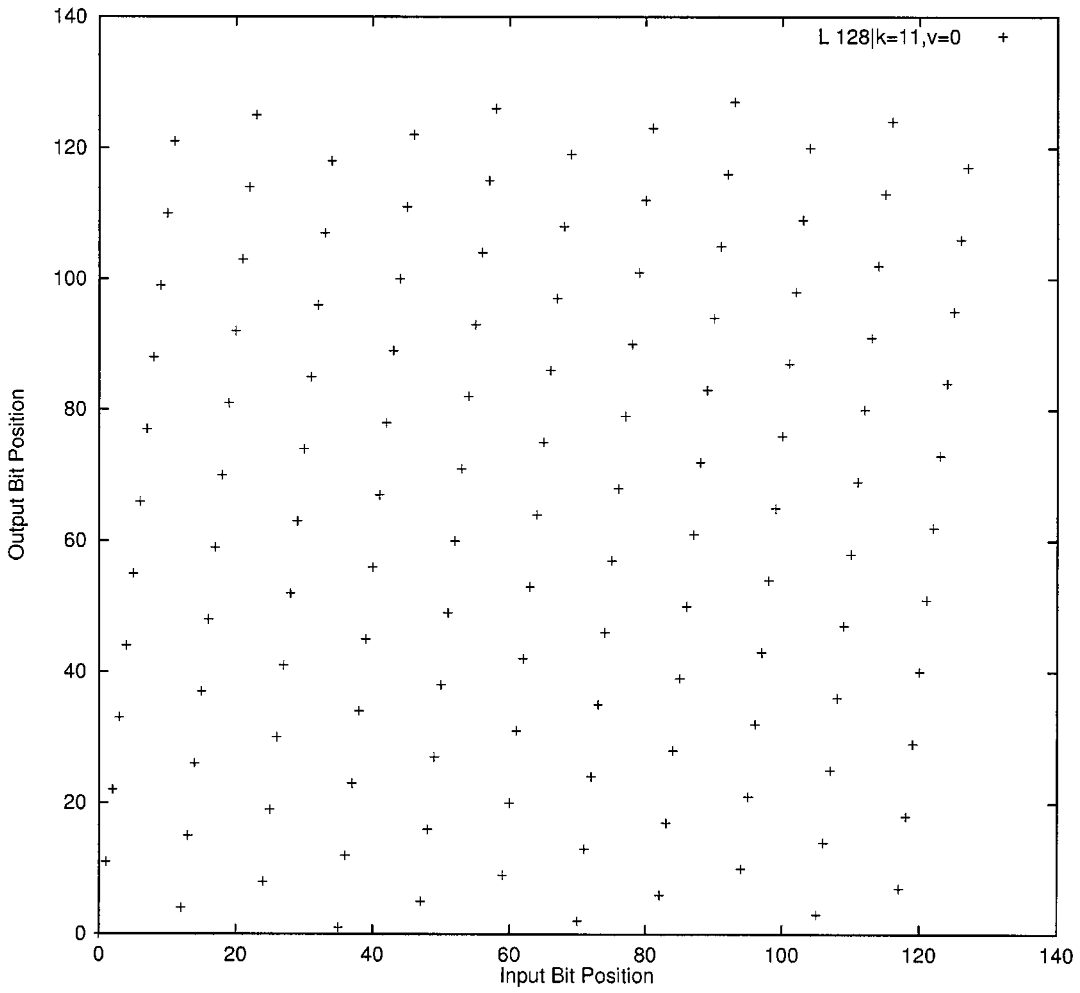

Fig. 3. Graphical representation of the $\mathcal{L}_{128|k=11,v=0}$ interleaver.

In Fig. 7, we show the BER curves, obtained by simulations, for each of the interleavers in Figs. 5 and 6.[8] From Fig. 6 we see that the estimated minimum distance of interleaver $\mathcal{L}_{128|k=63,v=0}$ is only 6, due to a *globally-bad* input sequence: this explains its poor performance in Fig. 7. The estimated minimum distance of interleaver $\mathcal{L}_{128:k=17,v=0}$ is actually smaller, but it has no low-weight codeword due to a *globally-bad* input sequence, so its performance is better. From Fig. 5 we see that the estimated minimum distances of interleavers $\mathcal{L}_{128|k=11,v=0}$ and $\mathcal{L}_{128|k=7,v=0}$ are larger, giving them very good performance, even though they contain some moderate-weight codewords due to *globally-bad* input sequences. To compare these linear interleavers with random interleavers, we also show the average performance of seven random interleavers in Fig. 7.

Now we digress to introduce some concepts of random interleavers that are related to the new classes of deterministic interleavers to be introduced in Section IV. One can observe that the distance spectrum of a random interleaver is erratic, but it typically does not contain low weights with multiplicities that are on the order of the block length. This explains the interleaver gain [2], where the effects on the BER curve of such low weights and low multiplicities, represented by (4), are attenuated by a factor of N. For the same reason, the effect on the BER curve of *locally-bad* input sequences for linear interleavers are also attenuated by a factor of N. This means that the advantage of random interleavers over linear interleavers is that random interleavers do not have *globally-bad*, but only *locally-bad*, input sequences. On the other hand, *locally-bad* input sequences for linear interleavers are much easier to eliminate by just changing a simple parameter such as the angular coefficient. Compared to linear interleavers, random interleavers can be viewed as a "nonlinear" device, and hence a Type I input sequence in the top component code combined with a Type I input sequence in the lower component code does not have multiplicity on the order of N and thus is not classified as *globally-bad*. In Section IV, we introduce a new class of interleavers that are based on quadratic (and hence "nonlinear") congruences.

In optimizing the angular coefficient for $\mathcal{L}_{128}$ interleavers, it was not possible to make the BER curve approach the linear asymptote given in (5). This is due to the fact that the block length of the interleaver is too short (i.e., the interleaver gain is too small). The estimated minimum distances for $\mathcal{L}_{256}$ interleavers do not differ much compared with $\mathcal{L}_{128}$ interleavers; however, there are interleavers of this block length that achieve the linear asymptote, as shown in Fig. 8. (We also show the average performance of seven random interleavers in Fig. 8.) Note

[8]We use iterative maximum *a posteriori* (MAP) decoding [3] with nine complete iterations in all simulations throughout this paper.

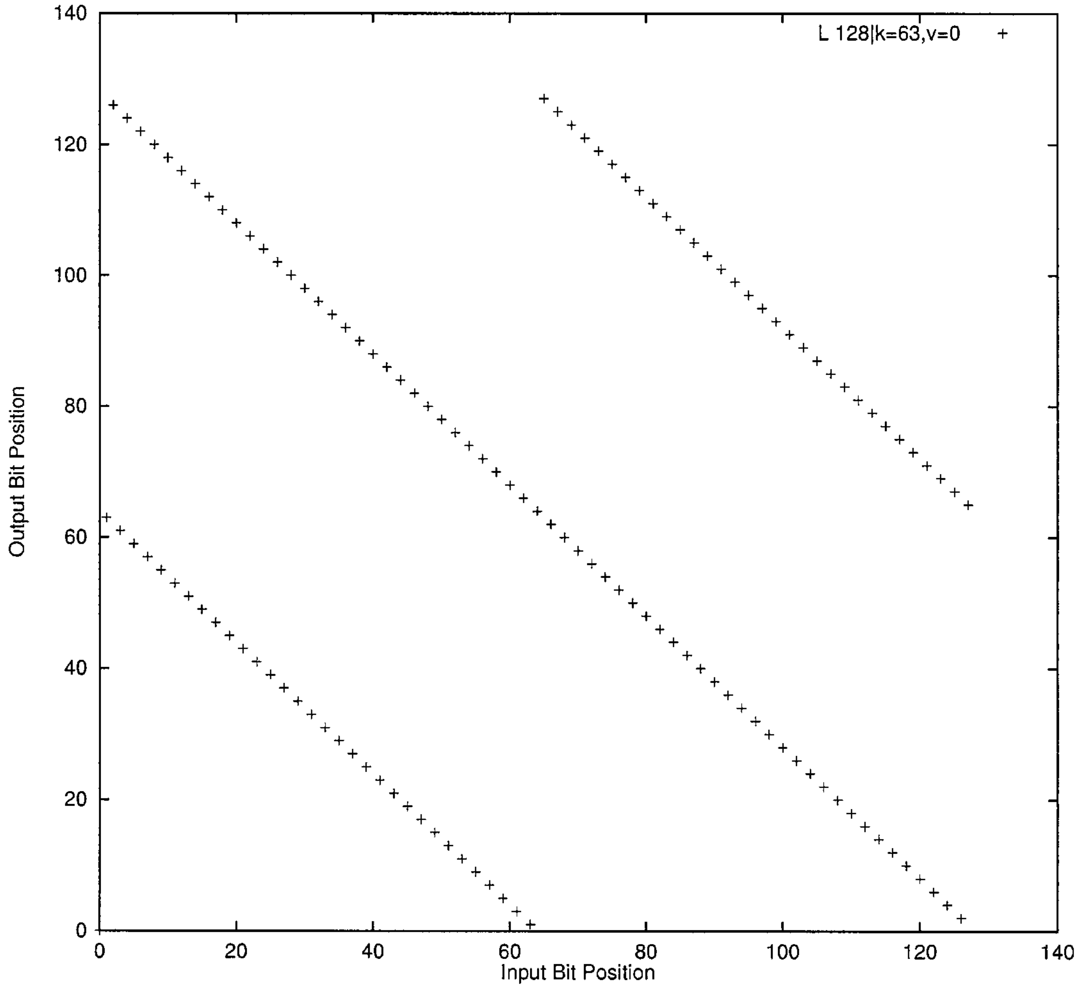

Fig. 4. Graphical representation of the $\mathcal{L}_{128|k=63,v=0}$ interleaver.

from Figs. 7 and 8 that the performance of linear interleavers can be dramatically changed by making a small change in the angular coefficient.

C. Using SRCCs with Primitive Feedback Polynomials

It has been observed in [3], [4] that the use of SRCCs with a primitive feedback polynomial in the component codes improves the BER performance of turbo codes using random interleavers, especially in the "error-floor" region. This is attributed to the fact that the minimum distance of the code increases. We now show that similar beneficial effects are also obtained for turbo codes using linear interleavers.

If we calculate the asymptote due to "bad" weight-4 input sequences (this essentially depends only on the cycle-length τ of the SRCC), as explained in Section II, for the "primitive" turbo code $C_T(\mathcal{L}_N, (23, 35), 1/2)$, we achieve a much lower linear interleaver asymptote than for the "nonprimitive" $C_T(\mathcal{L}_N, (37, 21), 1/2)$ turbo code, since the minimum distance is on the order of 20 rather than 12. We can see that by using a larger cycle-length τ, we are able to avoid (or diminish) the effect of a *globally-bad* input sequence. There is, however, a problem in using a "nonprimitive" turbo code. One can verify that no weight-3 Type I precoded input sequences exist for the "nonprimitive" turbo code, but there are several different types (up to cyclic shifts) of such "bad" precoded input sequences for the "primitive" turbo code. In particular, a weight-3 input sequence of the form $(\ldots, 0, 1, 0, 0, 1, 1, 0, \ldots)$ produces the parity sequence (before puncturing) $(\ldots, 0, 1, 1, 1, 0, 1, \ldots)$. Reasoning similar to that used for $\overline{x}_{\text{bad4}}$ sequences, which were a combination of two weight-2 input sequences, can be used to select a combination of three "bad" weight-3 input sequences. We then obtain an $\overline{x}_{\text{bad9}}$ input sequence with an estimated codeword weight[9] of 19 that is *globally-bad* and is again invariant for all block lengths and angular coefficients. The linear interleaver asymptote of the "primitive" $C_T(\mathcal{L}_N, (23, 35), 1/2)$ turbo code can thus be estimated as

$$P_b(e|\overline{x}_{\text{bad4}} \text{ or } \overline{x}_{\text{bad9}}) \approx 2 \text{ erfc}\left(\sqrt{10\frac{E_b}{No}}\right) + 9/2 \text{ erfc}\left(\sqrt{9.5\frac{E_b}{No}}\right).$$

In Fig. 9, we show simulation results of the "nonprimitive" $C_T(\mathcal{L}_N, (37, 21), 1/2)$ and "primitive" $C_T(\mathcal{L}_N, (23, 35), 1/2)$ turbo codes. Note that when a component code with a primitive feedback polynomial is used, the performance at low BERs is

[9]We omit the details; however, in addition to the contribution of nine input bits, each component code contributes at least five parity bits to the codeword weight.

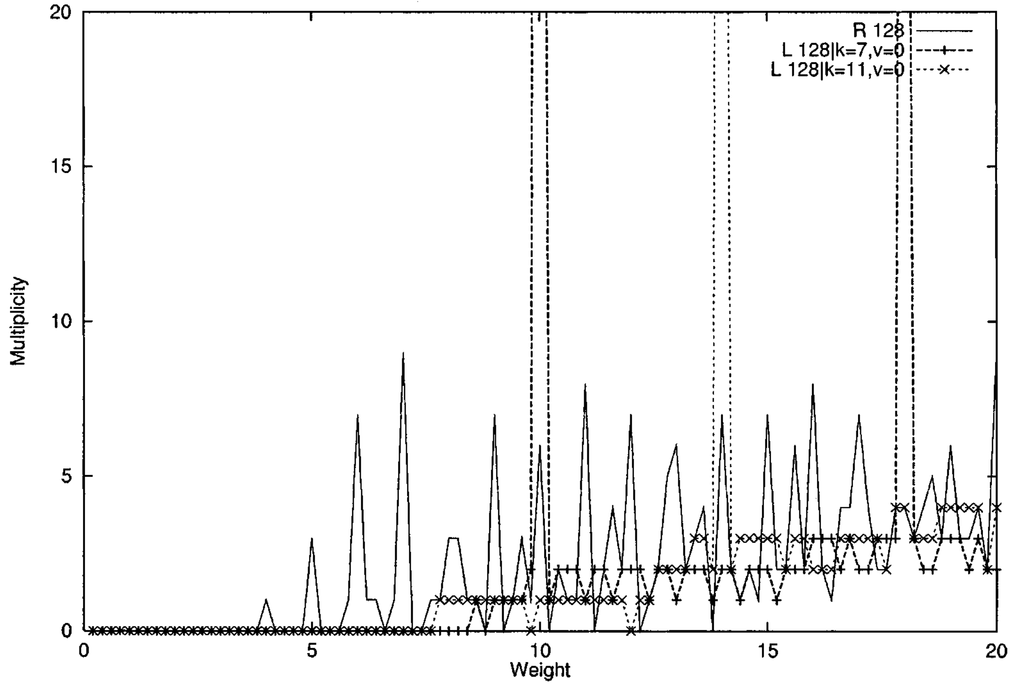

Fig. 5. Estimated distance spectrum for a random intearleaver and for two linear interleavers with large estimated minimum distance.

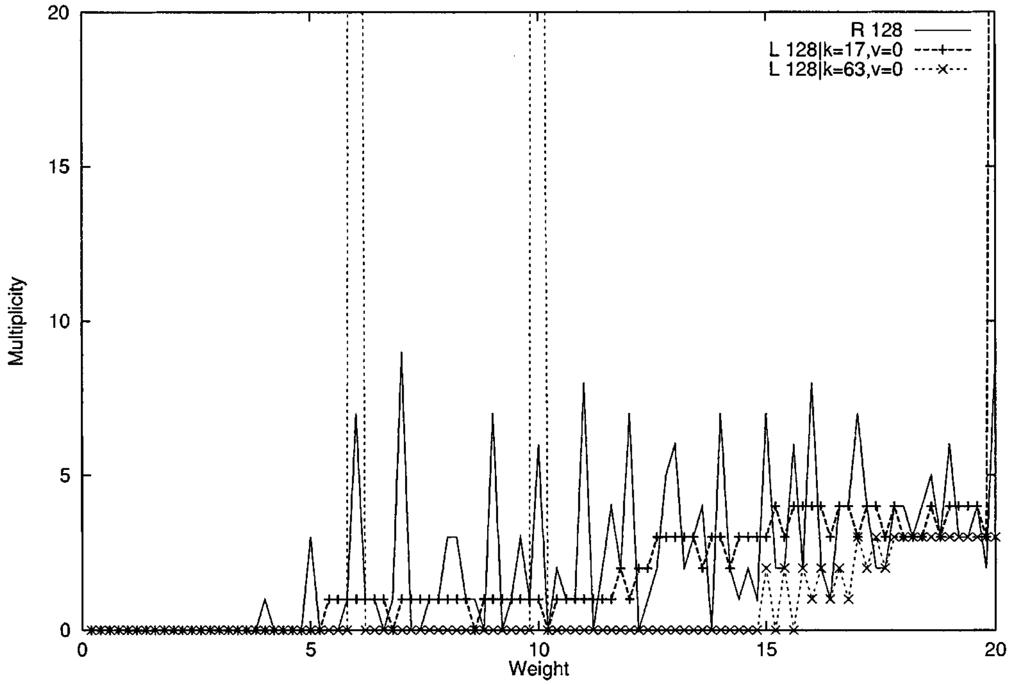

Fig. 6. Estimated distance spectrum for a random interleaver and for two linear interleavers with small estimated minimum distance.

slightly worse. However, the "error floor" is dramatically lowered for sufficiently long block lengths. Of particular interest ia a block length of 16384, for which we obtain codes only 1.1 dB away from capacity at a BER of 10^{-5} using a highly structured linear interleaver.

Interleavers that are essentially composed of several small sections of linear interleavers with different angular coefficients have also been proposed [30]. This construction makes the interleaver less regular and thus less susceptible to globally bad input sequences and achieves better performance for larger block lengths than pure linear interleavers.

IV. A New Design for Deterministic Interleavers

The subject of the previous section was linear interleavers, which basically means that cyclic shifts of input sequences

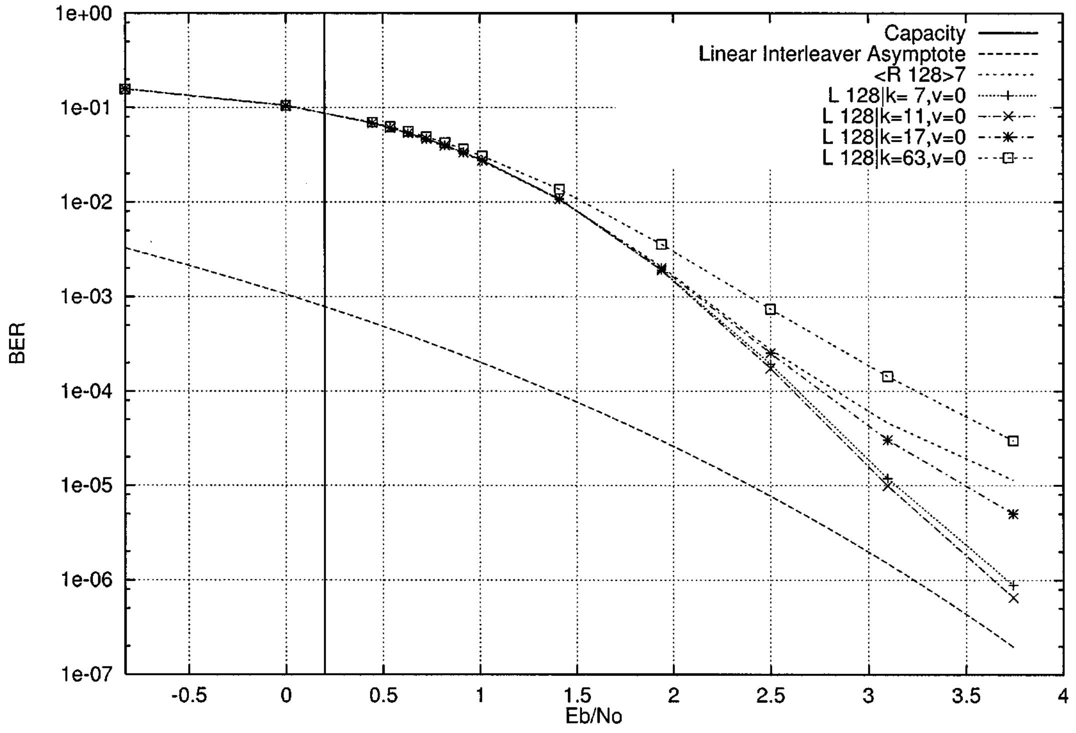

Fig. 7. The performance of linear interleavers of length 128 with several angular coefficients and the average performance of seven random interleavers.

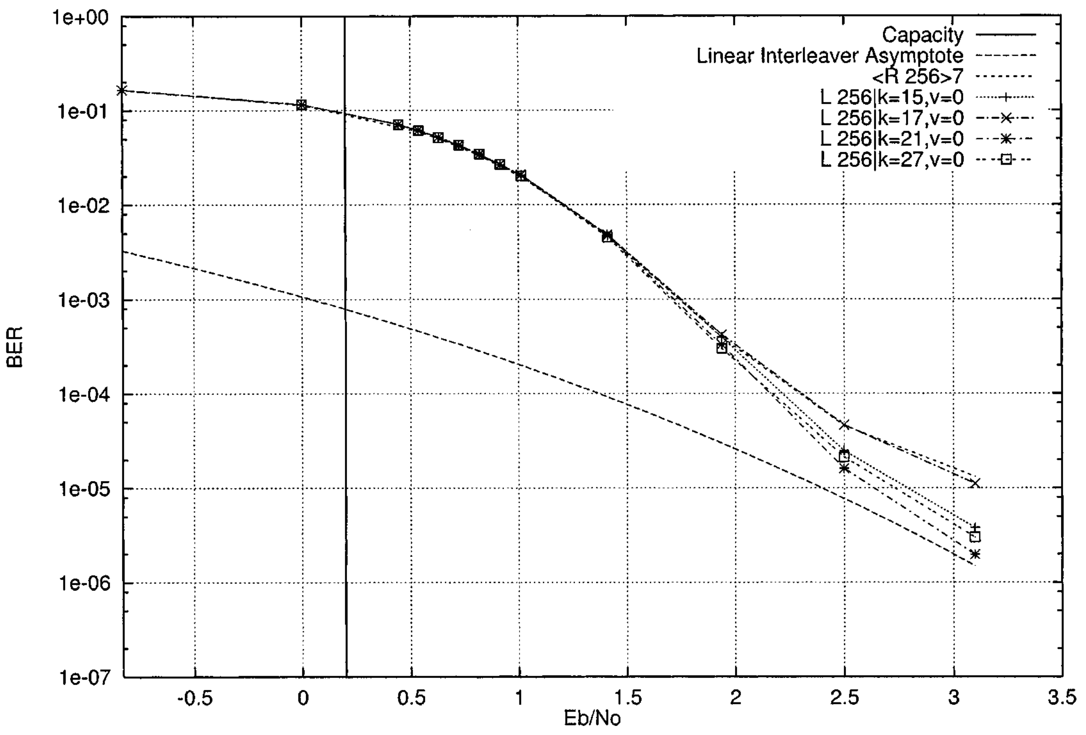

Fig. 8. The performance of linear interleavers of length 256 with several angular coefficients and the average performance of seven random interleavers.

to the interleaver produce cyclic shifts of the corresponding output sequences. Random interleavers and the new classes of interleavers introduced in this section do not have the same cyclic shift invariance as linear interleavers, and hence they

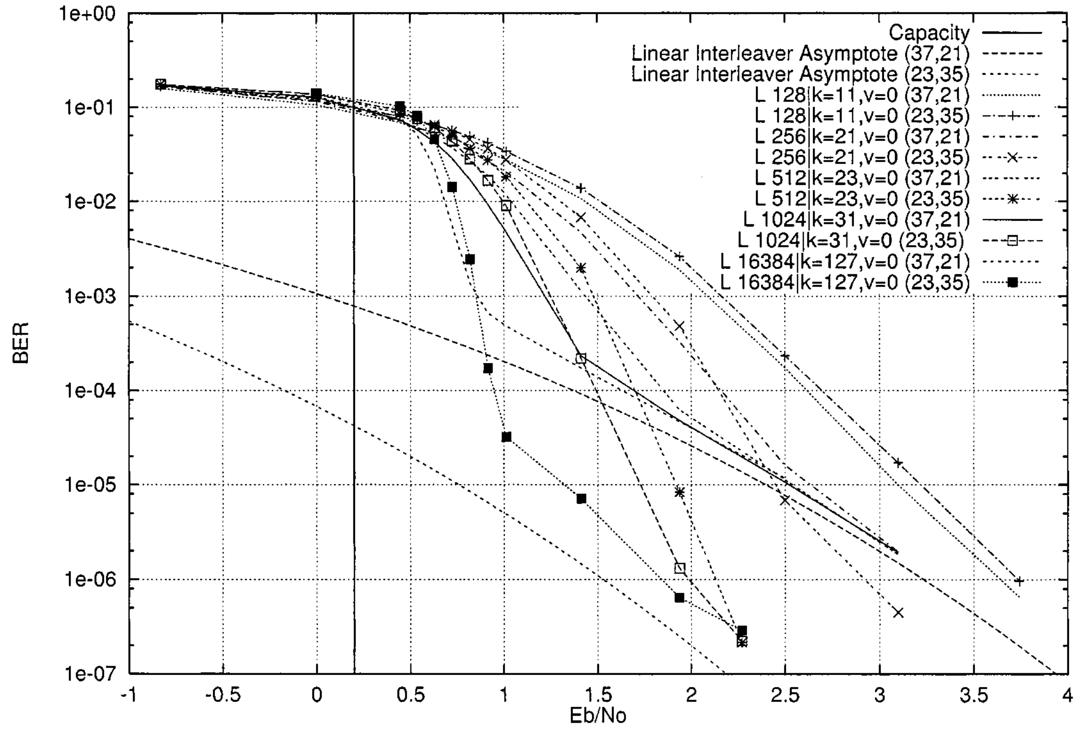

Fig. 9. The BER performance of the best linear interleavers found for the turbo codes $C_T(\mathcal{L}_N, (37, 21), 1/2)$ and $C_T(\mathcal{L}_N, (23, 35), 1/2)$.

are "nonlinear" in that sense. Before attempting to design "nonlinear" interleavers, however, we must understand the statistics of good random interleavers.

A. Random Interleavers Modeled as Uniform Interleavers

In [2] Benedetto and Montorsi derive an upper bound on the BER of a turbo code by introducing the concept of a uniform interleaver.

Definition 4.1 [2]: A *uniform interleaver* of length N is a probabilistic device that maps an input sequence of length N and weight w to all its distinct $\binom{N}{w}$ permutations with equal probability $1/\binom{N}{w}$.

It is also shown in [2] that the BER performance calculated using a uniform interleaver is indeed an average over all possible interleavers. This proves the existence, for every BER, of at least one interleaver that achieves at least the same performance as a hypothetical turbo code using a uniform interleaver.

Actual interleavers used in a turbo code are in a sense structured, since they are generated by a deterministic process, typically a pseudo-random sequence generator. If we can learn what kinds of structure affect the performance of a turbo code, we can constrain interleavers without sacrificing performance in order to reduce implementation complexity.

The definition of a uniform interleaver is formulated in terms of the input and output weights of sequences. In spite of this simple definition, a uniform interleaver is a very good statistical model of a random interleaver, and, as we shall see, the model contains more useful information than just the statistics of the weight distribution of the input and output sequences. If we intend to design an interleaver with performance close to that of a random interleaver, it should be sufficient to make its statistics as close as possible to a uniform interleaver.

Most statistical properties of a uniform interleaver are desirable for a turbo code. However, not all of them are necessary, and some of them may actually have a negative effect on performance.

It is not difficult to see, for example, that to exactly meet Definition 4.1,[10] a necessary condition is to have interleavers of size $2^r - 1$, r an integer, such that their cycle-structure is exactly of the form $C\{1_1, 2_1, 4_1, 8_1, \ldots, 2_1^{r-2}\}$.[11] We shall see, however, that this is not a restriction that must be followed in the construction of good interleavers.

B. New Classes of Deterministic Interleavers

Our design is based on the concept of trying to construct a deterministic interleaver that has statistical properties close to those of a uniform interleaver for weight-1 and weight-2 input sequences. For a weight-1 input sequence, an interpretation of the definition of a uniform interleaver is that the 1 in the input sequence should be displaced relative to the 1 in the output sequence by all possible distances modulo N, where N is the length of the interleaver. A very simple construction that closely approximates a uniform interleaver is to map the first element in the sequence to itself $+1$ (modulo N), then to map this element to itself $+2$ (modulo N), and so on. An amazing fact of this algorithm, proved in Theorem 4.1, is that for $N = 2^r$, r an integer, the elements in this sequence define an N-cycle of a

[10]Note that uniform interleavers fix a sequence of weight w with probability $1/\binom{N}{w}$.

[11]R. McEliece has kindly pointed out that the Russian mathematician Goncharov has studied the cycle structure of random permutations.

permutation, i.e., the elements neither repeat nor map to 0 (the first element), except for the last element in the sequence, which must be mapped back to the first element in order to close the cycle. Moreover, the increasing steps can be any odd integer k instead of 1. We thus have a strongly CN deterministic interleaver that we denote by $\mathcal{D}_{N:CN}$. The *index mapping function* of $\mathcal{D}_{N:CN}$

$$d_{\mathcal{D}_{N:CN}}\colon c_m \mapsto c_{m+1(\bmod N)}, \qquad 0 \le m < N \tag{6}$$

is defined by the following algorithm.

Algorithm 4.1

```
Step 1: $c_0 = 0$
Step 2: $c_m \equiv c_{m-1} + km \,(\bmod N), 0 < m < N,$
        $k$ an odd constant.  □
```

This can be expressed as the quadratic congruence

$$c_m \equiv \frac{km(m+1)}{2} \;(\bmod N), \qquad 0 \le m < N,\; k \text{ an odd constant.} \tag{7}$$

Example 4.1: For a the $\mathcal{D}_{N:CN}|_{k=1}$ interleaver with $N = 8$, we first compute its unique 8-cycle

$$\mathcal{D}_{8:C8}|_{k=1} = (c_0, c_1, \ldots, c_7) = (0, 1, 3, 6, 2, 7, 5, 4)$$

using either Algorithm 4.1 or (7). Then we have the *index mapping function* $d_{\mathcal{D}_{8:C8}|_{k=1}}\colon 0 \mapsto 1, 1 \mapsto 3, \ldots$ defined by (6), and the *permutation vector* representation becomes

$$\overline{\mathcal{D}}_{8:C8}|_{k=1} = [1, 3, 7, 6, 0, 4, 2, 5].$$

Example 4.2: Following the same procedure as in Example 4.1 for the $\mathcal{D}_{N:CN}|_{k=1}$ interleaver with $N = 16$, we obtain

$$\mathcal{D}_{16:C16}|_{k=1} = (0, 1, 3, 6, 10, 15, 5, 12, 4, 13, 7, 2, 14, 11, 9, 8)$$

and the *permutation vector*

$$\overline{\mathcal{D}}_{16:C16}|_{k=1} = [1, 3, 14, 6, 13, 12, 10, 2, 0, 8, 15, 9, 4, 7, 11, 5].$$

Example 4.3: Now consider the same example for $k = 3$.

$$\mathcal{D}_{16:C16}|_{k=3} = (0, 3, 9, 2, 14, 13, 15, 4, 12, 7, 5, 6, 10, 1, 11, 8)$$

and

$$\overline{\mathcal{D}}_{16:C16}|_{k=3} = [3, 11, 14, 9, 12, 6, 10, 5, 0, 2, 1, 8, 7, 15, 13, 4].$$

Theorem 4.1: For $N = 2^r$, r an integer, Algorithm 4.1 defines an interleaver $\mathcal{D}_{N:CN}$ of length N that is strongly CN.

Proof: The proof has two parts: to show 1) that $c_m \not\equiv c_n$, $m \not\equiv n$, $0 \le m, n < N$, and 2) that 1) holds only when k is odd.

For part 1), we must show that no solutions exist for

$$c_{m+\alpha} - c_m \equiv k\alpha(\alpha + 2m + 1)/2 \equiv 0(\bmod N)$$

for $0 \le m < N - 1$, $0 < \alpha < N - m$. First, suppose that α is even. Then $c_{m+\alpha} - c_m \equiv k(\alpha/2)\omega \not\equiv 0(\bmod N)$ where ω is odd, which must be true because one can find solutions for $\alpha/2 \equiv 0(\bmod N)$ only for $\alpha \ge 2N$. Now suppose that α is odd. In this case, we must show that no solutions exit for $(\alpha + 2m + 1)/2 \equiv 0(\bmod N)$. But

$$0 < (\alpha + 2m + 1)/2 \le (N + m - 1)/2 < N$$

and hence the first part follows.

For part 2), it is not difficult to see that any odd k, but no even k, will produce the desired result. □

Equation (7) can be transformed in order to obtain the *index mapping function* $d_{\mathcal{D}_{N:CN}}$ of $\mathcal{D}_{N:CN}$ as follows:

$$j = d_{\mathcal{D}_{N:CN}}(i) \equiv \frac{k}{2} + i + \frac{\sqrt{k^2 + 8ki}}{2} \;(\bmod N) \tag{8}$$

with the exception that $d_{\mathcal{D}_{N:CN}}(2^{r-1} = N/2) = 0$. The reader should note, however, that (8) presents the difficulty that solving square roots over a modulo arithmetic is not the same as over the integers. In order to generate the interleaver, it is easier to use Algorithm 4.1.

Equation (8) has some interesting properties. By construction, it must have an inverse, which is indeed

$$i = d^{-1}_{\mathcal{D}_{N:CN}}(j) \equiv \frac{k}{2} + j - \frac{\sqrt{k^2 + 8kj}}{2} \;(\bmod N). \tag{9}$$

Further, we can cyclically shift the *permutation vector* representation of

$$\overline{\mathcal{D}}_{N:CN} = [d_{\mathcal{D}_{N:CN}}(0), \ldots, d_{\mathcal{D}_{N:CN}}(N-1)]$$

to the right by a constant h, $0 \le h \le N - 1$, without changing the essential properties of the interleaver, since only the relative positions are affected. A similar operation is to add a constant v modulo N to all the elements of

$$\overline{\mathcal{D}}_{N:CN} = [d_{\mathcal{D}_{N:CN}}(0), \ldots, d_{\mathcal{D}_{N:CN}}(N-1)].$$

These operations may, however, change the cycle-structure and in general the interleaver is no longer CN. Hence we denote this broader class of interleavers simply as $\mathcal{D}_N$, for which the *index mapping function* is

$$j = d_{\mathcal{D}_N}(i) \equiv \frac{k}{2} + i + \frac{\sqrt{k^2 + 8k(i-h)}}{2} - (h - v) \;(\bmod N) \tag{10}$$

with the exception that $d_{\mathcal{D}_N}(2^{r-1} + h(\bmod N)) = v$.

The inverse function of (10) is

$$i = d^{-1}_{\mathcal{D}_N}(j) \equiv \frac{k}{2} + j - \frac{\sqrt{k^2 + 8k(j-v)}}{2} + (h - v) \;(\bmod N). \tag{11}$$

We refer to the class $\mathcal{D}_N$ of interleavers as quadratic interleavers because they are based on quadratic congruences.

Example 4.4: By cyclically shifting one unit to the right the interleaver of Example 4.2 we obtain

$$\overline{\mathcal{D}}_{16|k=1,h=1,v=0} = [5, 1, 3, 14, 6, 13, 12, 10, 2, 0, 8, 15, 9, 4, 7, 11]$$

with cycle-structure $C\{1_1, 1_2, 1_6, 1_7\}$, and hence the interleaver is weakly $C42$.

However, for $h - v = 2^{r-1} = N/2$, it follows that $h - v \equiv -(h - v)(\bmod N)$ and for this special case we obtain

$$d^{-1}_{\mathcal{D}_N|_{h-v=N/2}}(i) = d_{\mathcal{D}_N|_{h-v=N/2}}(i). \tag{12}$$

We thus have another class of interleavers that we call $\mathcal{D}_{N:C2}$ (a subclass of $\mathcal{D}_N$), because it is $C2$ by the symmetry of its *index mapping functions*. $C2$ interleavers provide an interesting implementation advantage, since decoders for turbo codes use interleaver–deinterleaver pairs and deinterleaving is identical to interleaving.

Example 4.5:

$$\overline{\mathcal{D}}_{16:C2}|_{k=1,h=8} = [0, 8, 15, 9, 4, 7, 11, 5, 1, 3, 14, 6, 13, 12, 10, 2]$$

has cycle-structure $C\{2_1, 7_2\}$, i.e., it is weakly $C2$.

If we square both sides of (10), the equation takes the following form:

$$i^2 - 2ij + j^2 + (-k - 2(h-v))i + (-k + 2(h-v))j + (h-v)^2 + k(h+v) \equiv 0 \pmod{N}$$

which turns out to be a conic.[12] $\mathcal{D}_N$ permuters can thus be viewed as a mapping defined by the integer points of a conic $(\mathrm{mod}\, N)$.

As mentioned earlier, an algorithmic approach is preferred to generate the interleavers. The algorithm for generating $\mathcal{D}_N$ interleavers of length $N = 2^r$, r an integer is given by

Algorithm 4.2

```
Step 1: compute the permutation vector $\overline{\mathcal{D}}_{N:CN}$ using
        Algorithm 4.1
Step 2: cyclically shift by $h$ units the result of Step 1
Step 3: add a constant $v (\mathrm{mod}\, N)$ to each element of the result
        of Step 2
```

where two special cases have been identified

$$\mathcal{D}_{N:CN} = \mathcal{D}_N|_{h-v=0} \quad \text{and} \quad \mathcal{D}_{N:C2} = \mathcal{D}_N|_{h-v=N/2}. \quad \square$$

We now present a brief description of some straightforward extensions of the deterministic interleavers introduced above.

- An Alternate Quadratic Interleaver:
  Instead of taking c_m as an intermediate step in Algorithm 4.1, we can define it as a new *index mapping function*

  $$d_{\mathcal{Q}_N}(i) = \frac{ki(i+1)}{2} + v \pmod{N}$$

  for a $\mathcal{Q}_N$ interleaver, which can also be called a quadratic interleaver. Simulation results show that $\mathcal{Q}_N$ interleavers have similar performance to $\mathcal{D}_N$ interleavers.

- Interleavers of Higher Degree:
  Because of the property that the composition of interleaver mappings forms another interleaver, we can form $d_{\mathcal{Q}_{N|k=k_2,v=v_2}}(d_{\mathcal{Q}_{N|k=k_1,v=v_1}}(i))$, which results in an *index mapping function* of a new interleaver defined by a polynomial of fourth degree.

- Shortened Interleavers:
  Linear interleavers do not have constraints on the length of the interleaver. However, the construction of $\mathcal{D}_N$ interleavers has a restriction on the interleaver length: it must be a power of 2. Based on the properties of $\mathcal{D}_{N:C2}$ interleavers, we can use the following algorithm to produce a shortened $C2$ inteleaver of length S, $S \le N$, $v = 0$, and $h = N/2$ that can be generated *on the fly* and with no storage tables. The algorithm generates pairs of numbers that represent positions that are swapped in the original sequence to generate the interleaved sequence.

[12]It is more specifically a parabola rotated clockwise by $\pi/4$ degrees.

Algorithm 4.3

```
Step 1: stored ← false
Step 2: $i \leftarrow 0$
Step 3: while($i \le (N-2)/2$)
  $x \leftarrow c_{i+1}$
  $y \leftarrow c_i + N/2 \pmod{N}$
  if ($x < S$ and $y < S$)
    swap positions $x$ and $y$
  if($x < S$ and $y \ge S$)
    if(stored = true)
      swap positions $x$ and $t$
      stored ← false
    else
      $t \leftarrow x$
      stored ← true
  if($x \ge S$ and $y < S$)
    if (stored = true)
      swap positions $y$ and $t$
      stored ← false
    else
      $t \leftarrow y$
      stored ← true
  increment $i$ by 1.                                   □
```

V. Analysis of the New Classes of Interleavers

In this section we analyze some of the properties of turbo codes driven by weight-1 and weight-2 input sequences using the new classes of interleavers.

A. Analysis for Weight-1 Input Sequences

As mentioned earlier, it follows from the definition of a uniform interleaver that a weight-1 input sequence is cyclically shifted by each possible distance i, $0 \le i < N$, with equal probability. Since there are exactly N sequences of weight-1, and exactly N possible shifts, there is exactly one occurrence of each shift.

Theorem 5.1: For the N distinct weight-1 sequences $\overline{x}_i$, $0 \le i < N$, the $\mathcal{D}_{N:CN}$ interleaver cyclically shifts them such that there is exactly one occurrence of each shift, except that there are no shifts of length 0 and the shift of length $N/2$ occurs twice.

Proof: This follows directly from the definition of $\mathcal{D}_{N:CN}$ interleavers. $\square$

Example 5.1: The ith element in a sequence of length 16 is shifted by the interleaver $\mathcal{D}_{16:C16}|_{k=1}$ in Example 4.2 by a distance corresponding to the ith element in the sequence $(1, 2, 12, 3, 9, 7, 4, 11, 8, 15, 5, 14, 8, 10, 13, 6)$. Note that there are no shifts of length zero and two shifts of length $16/2 = 8$.

Theorem 5.1 says that if we take all possible weight-1 sequences, then we will observe all possible shifts with equal frequency at the output of a $\mathcal{D}_{N:CN}$ interleaver, just as for a uniform interleaver, except for the shifts of length 0 and $N/2$.

Corollary 5.2: For the N distinct weight-1 sequences $\overline{x}_i$, $0 \le i < N$, the $\mathcal{D}_N$ interleaver cyclically shifts them such that there is exactly one occurrence of each shift, except that there

TABLE I
PROBABILISTIC DISTRIBUTION OF SIGNATURES OF WEIGHT-2 SEQUENCES FOR A UNIFORM INTERLEAVER

From Signature	To Signature	Probability	Expected Number
$sig(1+D^{N/2})$	not $sig(1+D^{N/2})$	$\frac{2}{N-1}$	$\frac{N}{N-1}$
$sig(1+D^{N/2})$	$sig(1+D^{N/2})$	$\frac{1}{N-1}$	$\frac{N}{2(N-1)}$
not $sig(1+D^{N/2})$	not $sig(1+D^{N/2})$	$\frac{2}{N-1}$	$\frac{2N}{N-1}$
not $sig(1+D^{N/2})$	$sig(1+D^{N/2})$	$\frac{1}{N-1}$	$\frac{N}{N-1}$

TABLE II
DISTRIBUTION OF THE INPUT–OUTPUT SIGNATURES OF WEIGHT-2 SEQUENCES APPLIED TO THE $\mathcal{D}_{32:C32}$ INTERLEAVER

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	1	2	2	2	3	2	3	1	4	0	2	2	2	3
2	2	3	2	0	4	3	4	0	2	3	3	0	1	3	2	0
3	1	2	1	2	2	1	1	2	2	3	2	3	2	3	3	2
4	2	0	2	7	2	0	3	0	3	0	0	9	0	0	4	0
5	2	4	2	2	1	3	2	2	2	2	2	3	2	0	1	2
6	2	3	1	0	3	5	1	0	1	3	2	0	2	5	4	0
7	3	4	1	3	2	1	1	3	2	3	1	1	3	1	1	2
8	2	0	2	0	2	0	3	16	1	0	2	0	2	0	2	0
9	3	2	2	3	2	1	2	1	3	3	3	1	2	1	1	2
10	1	3	3	0	2	3	3	0	3	5	1	0	2	5	1	0
11	4	3	2	0	2	2	1	2	3	1	3	3	2	1	1	2
12	0	0	3	9	3	0	1	0	1	0	3	7	3	0	2	0
13	2	1	2	0	2	2	3	2	2	2	2	3	3	2	2	2
14	2	3	3	0	0	5	1	0	1	5	1	0	2	7	2	0
15	2	2	3	4	1	4	1	2	1	1	1	2	2	2	3	1
16	3	0	2	0	2	0	2	0	2	0	2	0	2	0	1	0

are no shifts of length $0-(h-v)(\operatorname{mod} N)$ and the shift of length $N/2-(n-v)(\operatorname{mod} N)$ occurs twice.

Proof: It is easy to see that the only shifts that do not occur exactly once in the $\mathcal{D}_N$ interleaver are the corresponding shifts in the $\mathcal{D}_{N:CN}$ interleaver minus $h-v(\operatorname{mod} N)$, i.e., the shifts of lengths $0-(h-v)(\operatorname{mod} N)$ and $N/2-(h-v)(\operatorname{mod} N)$. □

Example 5.2: The ith element in a sequence of length 16 is shifted by the interleaver $\mathcal{D}_{16|k=1,h=1,v=0}$ in Example 4.4 by a distance corresponding to the ith element in the sequence $(5,0,1,11,2,8,6,3,10,7,14,4,13,7,9,12)$. The only shifts that do not appear exactly once and $0-(1-0)\equiv 15(\text{mod}16)$ and $8-1\equiv 7(\operatorname{mod} 16)$.

B. Analysis for Weight-2 Input Sequences

It is easy to check that there are $N/2$ distinct signatures of weight-2 sequences of length N. Moreover, the size of each equivalence class is N, except for the equivalence class $\operatorname{sig}(\overline{x}(D)) = 1 + D^{N/2}$, which has size $|\operatorname{sig}(\overline{x}(D))| = N/2$. The sum of the sizes of each equivalence classes must be $(N/2-1)N + N/2 = \binom{N}{2}$. In order to obtain a mapping similar to a uniform interleaver, each signature must map to the other signatures with the probabilities listed in Table I. For example, the first line of Table I is explained as follows: given that the original signature is $1+D^{N/2}$ and that the signature of the mapped sequence is $1+D^q$, $q \neq N/2$, then the probability of such a mapping is $2/(N-1)$. The expected number of these mappings for a given $q \neq N/2$ is then given by this probability multiplied by the size of the equivalence class of the original signature, i.e.,

$$(2/(N-1))|\operatorname{sig}(1+D^{N/2})| = N/(N-1).$$

To simplify the analysis, let us assume that there are $N/2$ signatures of weight-2, each of size N. With this simplification, a uniform interleaver maps each weight-2 input sequence in a given equivalence class to two different shifts of every possible signature.

Table II shows the input–output signature distribution for the $\mathcal{D}_{32:C32}$ interleaver; an entry u in row r and column c means that there are u weight-2 input sequences with signature $1+D^r$ that are mapped to sequences with signature $1+D^c$.

In spite of the simple algebraic structure of the new interleavers, the distribution of the input and output signatures of weight-2 input sequences applied to $\mathcal{D}_{N:CN}$ interleavers is too irregular for a precise characterization. Claim 5.3 will argue why it is approximately uniform and explain some of the entries with a 0 value.

We can, however, compare the statistical distribution of the multiplicities in Table II (e.g., we have 86 entries equal to the expected value of 2 for a uniform interleaver) with an average of the same table for random interleavers. In Table III we show such a comparison for an average of 500 random interleavers $\langle\mathcal{R}_{32}\rangle_{500}$. Note that the $\mathcal{D}_{32:C32}$ interleaver has a larger number of mappings with the value 2, i.e., the expected value for a uniform interleaver, than the random interleaver average. Also, we see that the average of random interleavers does not have a maximum value of 2. Thus it is fair to say that the statistics of the

TABLE III
DISTRIBUTION OF SIGNATURE MAPPINGS

entry	0	1	2	3	4	5	6	7	8	9	16
$\mathcal{D}_{32:C32}$	51	46	86	51	10	6	0	3	0	2	1
$<\mathcal{R}_{32}>_{500}$	38.0	69.9	68.5	45.5	22.0	8.6	2.6	0.7	0.2	0.0	0.0

$\mathcal{D}_{32:C32}$ interleaver are closer to those of a uniform interleaver than a typical random interleaver.

Claim 5.3: The set of all N elements formed by input sequences $\overline{x}$ of weight-2 and fixed signature is mapped by the $\mathcal{D}_{N:CN}$ interleaver to approximately two elements of each signature of weight-2 sequences. Moreover, there are at least $N^2/32$ zero entries in the signature distribution matrix.

Argument: It is clear that to examine all possible weight-2 input sequences once, we only need to examine the input sequences $\overline{x}_i = D^{c_m} + D^{c_{m+\alpha}}$, for all $1 \leq \alpha \leq N-1$, $0 \leq m < N - \alpha$. Since, by the definition of $\mathcal{D}_{N:CN}$ interleavers, a 1 at position c_m in the input sequence maps to position c_{m+1} in the output sequence, the problem then reduces to examining the relation between the signatures of the input sequences $\overline{x}_i = D^{c_m} + D^{c_{m+\alpha}}$ and the signatures of the corresponding output sequences $\overline{x}_o = D^{c_{m+1}} + D^{c_{m+\alpha+1}}$.

Let us examine the congruence

$$c_{m+\alpha} - c_m \equiv k\alpha m + k(\alpha^2+\alpha)/2 \pmod N. \quad (13)$$

We can conclude the following.

i) Recalling that k is odd, if α is also odd, and if we allow $0 \leq m < N$, then all possible types of input signatures are generated by changing m, and all input signatures have the same cardinality for a fixed α. Since we have a restriction on the range of m, however, this statement not quite exact.

ii) If $\alpha \equiv 0 \pmod 4$, the difference $c_{m+\alpha} - c_m$ must be even. If $\alpha \equiv 2 \pmod 4$, the difference $c_{m+\alpha} - c_m$ must be odd.

iii) The relations

$$(c_{m+\alpha+1} - c_{m+1}) - (c_{m+\alpha} - c_m) \equiv k\alpha \pmod N$$
$$(c_{m+\alpha} - c_m) - (c_{m+\alpha-1} - c_{m-1}) \equiv k\alpha \pmod N$$

hold for almost all allowable values of m and α, except in the $2N-3$ situations when $m=0$ or $m+\alpha+1=N$. This means that the difference in the signatures of the input and output sequences are effectively independent of m for a fixed α.

i) and iii) imply that we must have an approximately uniform distribution in the input–output signatures for odd α's. We can also conclude from ii) and iii) that if the difference $(c_{m+\alpha+1} - c_{m+1}) - (c_{m+\alpha} - c_m) \equiv 2 \pmod 4$, then $c_{m+\alpha} - c_m$ cannot be even, i.e., there must be at least $N^2/32$ zero entries. (For example, Table II contains $(N/2)^2 = 256$ entries, and the $N/4 = 8$ rows with even input signatures have zero entries corresponding to every fourth output signature, thus guaranteeing a total of at least $N^2/32 = 32$ zero entries.) This completes the argument. □

Lemma 5.4: If the *permutation vector* $\overline{\mathcal{I}}_N$ of any interleaver is modified by a right circular shift of h units or the addition of a constant $v \pmod N$, the distribution of the input and output signatures of the input sequences of a given weight does not change for the resulting interleaver.

Proof: The relative positions $(\bmod N)$ of the ones in the output sequences are not changed by these operations for any fixed input sequence. □

Corollary 5.5: The distribution of input and output signatures of weight-2 input sequences is invariant over the interleavers $\mathcal{D}_N$ with fixed lengths and k, independent of h and v.

Let us assume that if an interleaver behaves statistically like a uniform interleaver for input sequences of weight 1 and weight n, then it will perform statistically like a uniform interleaver for input sequences of weight $(n+1)$. We then have the following conjecture.

Conjecture 5.1: The set of all elements formed by input sequences $\overline{x}$ of weight w and fixed signature is almost uniformly mapped by the $\mathcal{D}_N$ interleaver to each sequence with a signature of weight w.

Finally, we have computed the dispersion [22] of the $\mathcal{D}_N$ interleavers. Dispersion is a randomness measure for an interleaver. Since random-like interleavers generally give very good performance for long turbo codes, and interleavers with a large dispersion generally do not have *globally-bad* input sequences, good interleavers are expected to have a large dispersion. Dispersion is defined as

$$\gamma = |\{(\Delta_x, \Delta_y) | \Delta_x = b - a, \Delta_y = d(b) - d(a), \\ 0 \leq a < b < N\}| / (N(N-1)/2) \quad (14)$$

where $1/(N-1) \leq \gamma \leq 1$. The smallest value of γ is achieved by a trivial identity permutation. A typical block (linear) interleaver achieves a small dispersion on the order of $1/N^2$, an interleaver based on a Costas array achieves the maximum dispersion $\gamma = 1$, and a typical randomly chosen interleaver achieves a large dispersion of $\gamma \approx 0.81$ [22]. The $\mathcal{D}_N$ interleavers based on quadratic congruences achieve a dispersion of $\gamma \approx 0.74$, which is close to that of randomly chosen interleavers.

VI. PERFORMANCE OF THE NEW INTERLEAVERS

Based on Conjecture 5.1, we can say that $\mathcal{D}_N$ interleavers very closely approximate the definition of a uniform interleaver, and hence the performance can be analyzed as in [2]. We obtain, however, not an average upper bound over the ensemble of all interleavers, but an approximate upper bound for any particular interleaver in the $\mathcal{D}_N$ class.

A. Numerical Results

In Fig. 10 we show the simulated BER performance of turbo codes using the new deterministic $\mathcal{D}_{N:CN}$ and $\mathcal{D}_{N:C2}$ interleavers compared with the average performance of seven randomly chosen interleavers $\langle \mathcal{R}_N \rangle_7$ of the same size. The simulations used rate $1/2$, 16-state, SRCC component codes (the

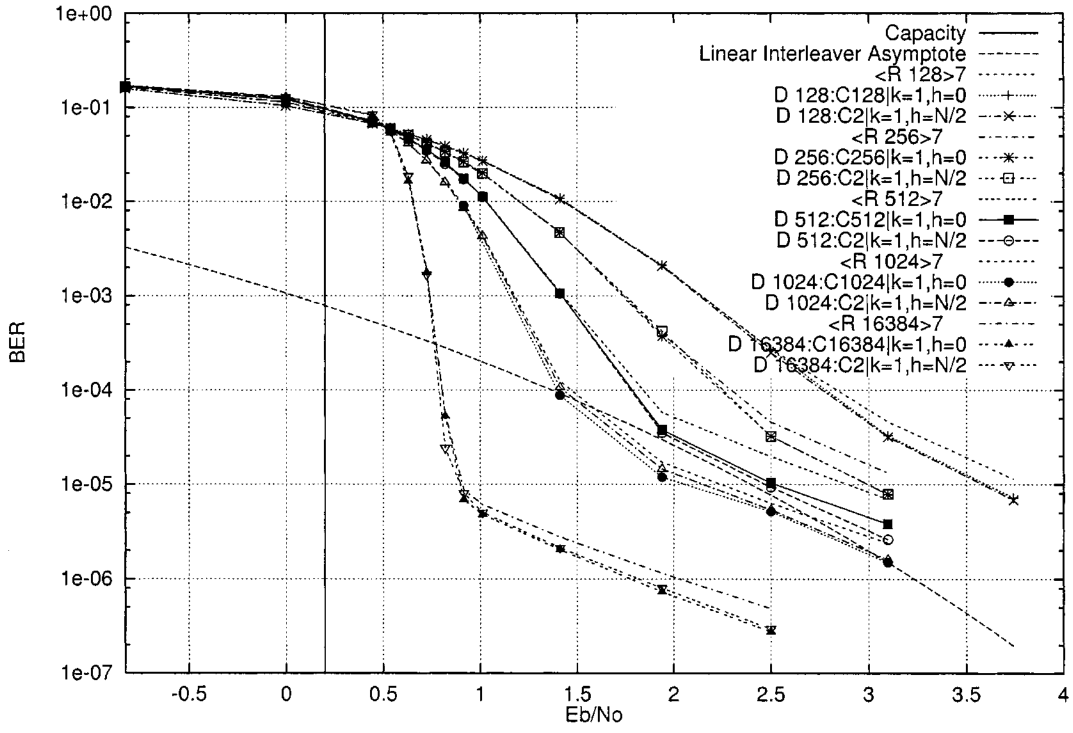

Fig. 10. Simulation results for the new $\mathcal{D}_{N:CN}$ and $\mathcal{D}_{N:C2}$ interleavers for lengths $128, 256, 512, 1024$, and 16384 compared with the average performance of seven random $\langle\mathcal{R}_N\rangle_7$ interleavers with the same lengths.

same SRCC used in Fig. 1). Every other parity bit of each convolutional code was punctured so that the overall code rate was $1/2$. All the curves look essentially the same in the "waterfall" region; however, the deterministic interleavers typically have better performance than the random interleavers in the "error-floor" region. Based on Corollary 5.5, the performance of a turbo code should not change for all $\mathcal{D}_N$ interleavers, at least with respect to weight-2 input sequences. Fig. 10 confirms this result by showing that the $\mathcal{D}_{N:CN}$ and $\mathcal{D}_{N:C2}$ interleavers have nearly the same performance. We can thus exploit the implementation advantage of $\mathcal{D}_{N:C2}$ interleavers with no sacrifice in BER performance.

We show in Fig. 11 the BER performance of turbo codes using the quadratic $\mathcal{D}_{N:C2}$ interleavers and the best linear $\mathcal{L}_N$ interleavers that we found. An interesting observation is that the "waterfall" region is nearly the same for all good interleavers of a fixed length before they start to diverge to their "error floor" performance. We may say that a turbo code is characterized by the sum of two asymptotes: one for the "waterfall" region, which is invariant with respect to the type of interleaver (as long as the interleaver is good), and one for the "error-floor" region. We see that good linear interleavers have an "error-floor" asymptote that is invariant to the length of the interleaver. Quadratic interleavers, on the other hand, have an "error-floor" asymptote that is a function of the length of the interleaver, so that for short block lengths, a linear interleaver is preferable, but for long block lengths, quadratic interleavers give superior performance.

Another interesting observation drawn from Figs. 10 and 11 is that we have a realization of simple deterministic interleavers that performs strictly better than an average of randomly chosen interleavers for a wide range of interleaver lengths and signal-to-noise ratios (SNRs).

B. Turbo Codes Using an SRCC with a Primitive Feedback Polynomial

The BER performance of turbo codes using the new quadratic $\mathcal{D}_{N:C2}$ interleavers and the best linear $\mathcal{L}_N$ interleavers, along with component codes that have primitive feedback polynomials, is shown in Fig. 12. If we compare with the results shown in Fig. 11 for "nonprimitive" codes, the performance of the "primitive" codes is better for longer block lengths and low BERs. For quadratic interleavers, the performance for block length 16384 is only 0.7 dB away from capacity at a BER of 10^{-5} and, at least down to a BER of 10^{-6}, no change in the slope of the curves is apparent for block lengths larger than 256.

We also observe that $C_T(\mathcal{L}_{16384|k=127,v=0}, (23, 35), 1/2)$ is a deterministically constructed code only 1.1 dB away from capacity at BER of 10^{-5} that is formed from a very simple linear interleaver.

VII. Conclusions

We have presented a new view of block interleavers and other similar interleavers within the unified concept of linear interleavers. For turbo codes, this view provides an algebraic tool that not only gives new insight into the fact that weight-4 input sequences have the dominant influence on the BER performance of linear interleavers, but it also leads to a derivation of the asymptotic behavior of optimized linear interleavers. We have shown that the asymptotic behavior due to the "bad" weight-4 input sequences of optimized linear interleavers is a function of

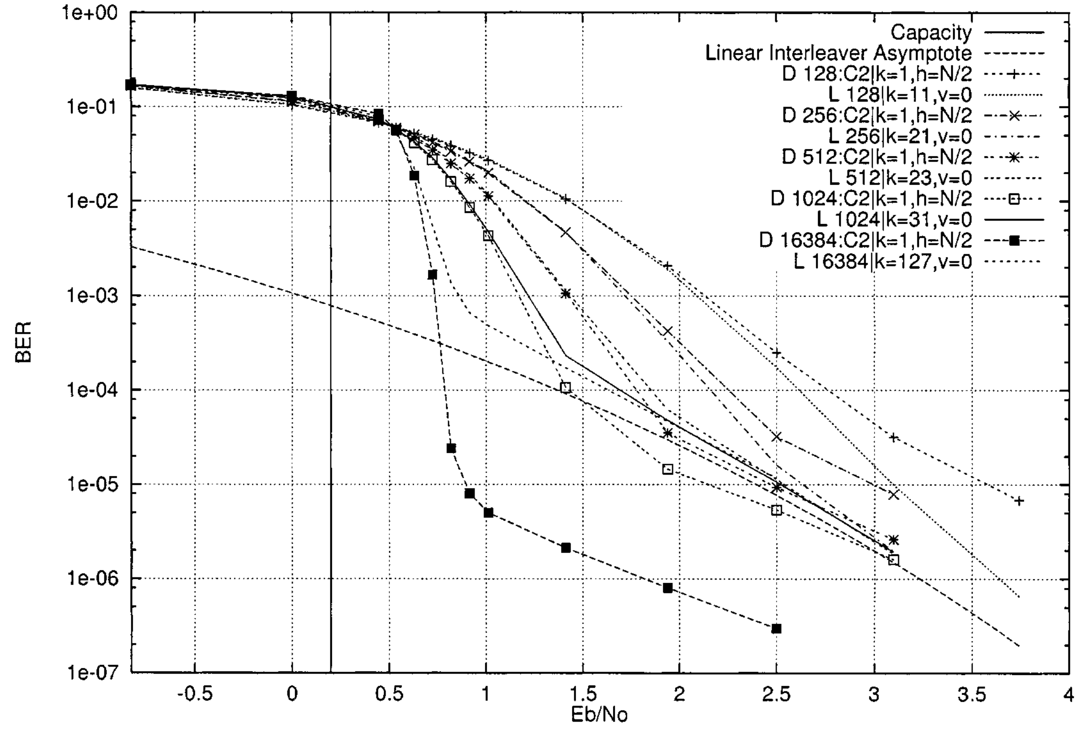

Fig. 11. A comparison of the BER curves of turbo codes using the quadratic $\mathcal{D}_{N:C2}$ interleavers and the best linear $\mathcal{L}_N$ interleavers.

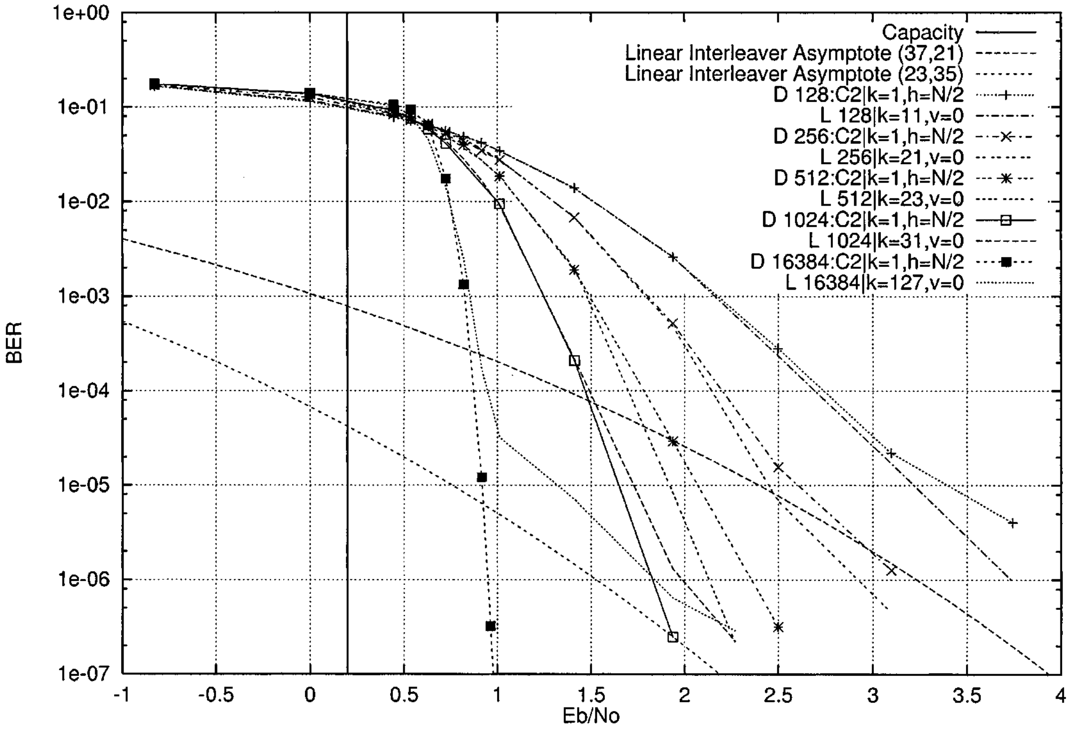

Fig. 12. A comparison of the BER curves of turbo codes using the quadratic $\mathcal{D}_{N:C2}$ interleavers and the best linear $\mathcal{L}_N$ interleavers along with a 16-state primitive feedback polynomial for the component code.

the cycle-length of the feedback polynomial of the SRCC component code, so that if a 16-state SRCC with a primitive feedback polynomial is used to maximize the cycle-length, weight-4 input sequences are no longer the constraining factor in the performance of turbo codes using optimized linear interleavers. With this observation, we have constructed a new turbo code with block length 16384 using a linear interleaver, i.e., a structured code whose performance is only 1.1 dB away from capacity at a BER of 10^{-5}. We have also confirmed previously known results that carefully chosen linear interleavers are preferred to random interleavers for short block lengths. We note, however, that for short block lengths more involved constructions have the potential for larger coding gains [31].

In addition, several new classes of deterministic interleavers $\mathcal{D}_N$ have been proposed. When these interleavers are applied to turbo codes, we always obtain performance equal to or better than the average performance of turbo codes using randomly chosen interleavers, with the better performance typically manifesting itself in the "error-floor" region. Their construction is very simple and can be viewed as a mapping defined by the integer points of quadratic curves, as opposed to the integer points of linear curves that define block interleavers, under modular arithmetic. Since decoders for turbo codes need interleaver–deinterleaver pairs, the particular subclass $\mathcal{D}_{N:C2}$ of the new interleaver class $\mathcal{D}_N$ has an implementation advantage, since interleaving and deinterleaving are identical. When used with SRCCs with a primitive feedback polynomial, these new interleavers form a deterministically constructed class of $C_T(\mathcal{D}_{N:C2}, (23, 35), 1/2)$ turbo codes that have excellent performance over a wide range of block lengths. In particular, the $C_T(\mathcal{D}_{16384:C2}, (23, 35), 1/2)$turbo code achieves a performance that is only 0.7dB away from capacity at a BER of 10^{-5}. Finally, the new $C_T(\mathcal{D}_{N:C2}, (23, 35), 1/2)$ turbo codes do not suffer a noticeable change of slope (i.e., an "error-floor") in their BER curves down to at least a BER of 10^{-6} for block lengths larger than 256.

Acknowledgment

The authors wish to thank the referees and the associate editor for their very helpful comments and suggestions.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshaima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, May 1993, pp. 1064–1070.

[2] 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.

[3] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. IEEE Globecom Conf.*, Dec. 1994, pp. 1298–1303.

[4] L. C. Perez, J. Seghers, and D. J. Costello, Jr., "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1698–1709, Nov. 1996.

[5] G. Battail, "On random-like codes," in *Proc. 4th Canadian Workshop on Information Theory*, Lac Delage, Que., Canada, May 1995, pp. 76–94.

[6] J. D. Andersen, "The TURBO coding scheme," Tech. Univ. Denmark, Lyngby, Denmark, Tech. Rep. IT-146, June 1994.

[7] J. Hagenauer and L. Papke, "Decoding 'Turbo'-codes with the soft output Viterbi algorithm," in *Proc. 1994 IEEE Int. Symp. Information Theory*, June 1994, p. 164.

[8] D. Arnold and G. Meyerhans, "The realization of the turbo-coding system," Swiss Federal Inst. Technol., Zurich, Switzerland, Semester project, July 1995.

[9] P. Jung and M. Naßhan, "Performance evaluation of turbo codes for short frame transmission systems," *Electron. Lett.*, vol. 30, no. 2, pp. 111–113, Jan. 1994.

[10] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo codes," *Electron. Lett.*, vol. 30, no. 25, pp. 2107–2108, Dec. 1994.

[11] A. S. Barbulescu, "Iterative Decoding of Turbo Codes and Other Concatenated Codes," Ph.D. dissertation, Univ. South Australia, Feb. 1996.

[12] S. Dolinar and D. Divsalar, "Weight distribution of turbo codes using random and nonrandom permutations," JPL, TDA Progr. Rep. 42-122, Aug. 1995.

[13] F. Daneshgaran and M. Mondin, "On design of interleavers of turbo codes," in *Proc. 31st Annu. Conf. Information Sciences and Systems*, Mar. 1997, pp. 509–514.

[14] A. K. Khandani, "Design of the turbo-code interleaver using Hungarian method," *Electron. Lett.*, vol. 34, no. 1, pp. 63–65, Jan. 1998.

[15] J. Hokfelt and T. Maseng, "Methodical interleaver design for turbo codes," in *Proc. Int. Symp. Turbo Codes*, Brest, France, Sept. 1997, pp. 212–215.

[16] K. S. Andrews, C. Heegard, and D. Kozen, "Interleaver design methods for turbo codes," in *Proc. 1988 IEEE Int. Symp. Information Theory*. Cambridge, MA, Aug. 1998, p. 420.

[17] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications*. New York: Plenum, 1991.

[18] S. V. Maric, "Class of algebraically constructed permutations for use in pseudorandom interleavers," *Electron. Lett.*, vol. 30, no. 17, pp. 1378–1379, Aug. 1994.

[19] P. M. Neumann, G. A. Stoy, and E. C. Thompson, *Groups and Geometry*. Oxford, U.K.: Oxford Univ. Press, 1994.

[20] F. Daneshgaran and M. Mondin, "Realization of permutations with minimal delay with applications to turbo coding," in *Proc. 1996 IEEE Int. Symp. Information Theory and Its Applications (ISITA '96)*, Victoria, B.C., Canada, Sept. 1996, pp. 590–593.

[21] J. L. Ramsey, "Realization of optimum interleavers," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 338–345, May 1970.

[22] C. Heegard and S. B. Wicker, *Turbo Coding*. Boston, MA: Kluwer Academic, Nov. 1998, Kluwer International Series in Engineering and Computer Science, vol. 476.

[23] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *Electron. Lett.*, vol. 32, no. 5, pp. 445–446, Feb. 1996.

[24] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, pp. 591–600, May 1996.

[25] G. D. Forney, Jr., *Concatenated Codes*. Cambridge, MA: MIT Press, 1966.

[26] B. M. Stewart, *Theory of Numbers*, 2nd ed. New York: Macmillan, 1964.

[27] H. Herzberg, "Multilevel turbo coding with a short latency," in *Proc. 1997 IEEE Int. Symp. Information Theory*, Ulm, Germany, June 1997, p. 112.

[28] A. Barbulescu and S. Pietrobon, "Terminating the trellis of turbo codes in the same state," *Electron. Lett.*, vol. 31, no. 1, pp. 22–23, Jan. 1995.

[29] D. Ci, "A new block helical interleaver," in *Proc. MILCOM'92*, San Diego, CA, Nov. 1992, pp. 799–804.

[30] J. D. Andersen and V. V. Zyablov, "Interleaver design for turbo-coding," in *Proc. Int. Symp. Turbo Codes*, Brest, France, Sept. 1997, pp. 154–156.

[31] F. Daneshgaran and M. Mondin, "Iterative interleaver growth algorithms of polynomial complexity for turbo codes," in *Proc. 1998 IEEE Int. Symp. Information Theory*. Cambridge, MA, Aug. 1998, p. 418.