

Some Thoughts on the Equivalence of Systematic and Non-Systematic Convolutional Encoders*

Daniel J. Costello, Jr.
Hermano A. Cabral
Dept. of Elec. Engr.
Univ. of Notre Dame

Oscar Y. Takeshita
Dept. of Elec. Engr.
Ohio State Univ.

August 2, 2001

Abstract

In this paper, we re-examine the notion of equivalent systematic and non-systematic convolutional encoders within the context of turbo coding. In particular, we argue that using equivalent systematic and non-systematic constituent encoders in a parallel concatenated system does not yield the same performance. In fact, the performance can differ greatly, depending on the effective free distance and the encoder mapping. Since non-systematic encoders typically have larger effective free distances than their systematic counterparts, they are better in the “error floor”, or low BER, region of the performance curve. On the other hand, since systematic encoders typically have better encoder mappings than non-systematic encoders, they are better in the “waterfall”, or high BER, region of the performance curve.

1 Introduction

In Forney’s seminal 1970 paper, “Convolutional codes I: algebraic structure”[1], he introduced the notion of equivalent convolutional encoders, i.e., encoders whose set of output sequences is identical over the entire space of possible input sequences. Equivalent encoders are thus considered to generate the same code, i.e., the same set of codewords. Additional work in this area by Forney and others has focused on finding the encoder with the smallest state complexity, a so-called minimal encoder, that generates a particular code. (See the new book by Johannesson and Zigangirov[2] for a complete summary of this work.)

As an example of this notion of equivalence, it is well known that any non-systematic feedforward encoder is equivalent to a minimal systematic feedback encoder[1][3]. In other words, the weight spectrum of the code generated by the two encoders is identical. (The code generated by a feedback encoder has been referred to in the recent literature on turbo coding as a systematic recursive convolutional code (SRCC)[4].)

*This work was supported by NSF grant NCR95-22939 and NASA grant NAG5-8355.

Example 1 Consider the $(2, 1, 3)$ non-systematic feedforward encoder¹ given by

$$G_{\text{ns}}(D) = [1 + D + D^3 \quad 1 + D^2 + D^3]. \quad (1)$$

The equivalent systematic feedback encoder is given by

$$G_{\text{s}}(D) = [1 \quad (1 + D^2 + D^3)/(1 + D + D^3)]. \quad (2)$$

The input sequence $x_{\text{s}}(D) = 1 + D^7$ to the systematic encoder generates the codeword

$$y(D) = x_{\text{s}}(D)G_{\text{s}}(D) = [1 + D^7 \quad 1 + D + D^4 + D^5 + D^6 + D^7], \quad (3)$$

whereas for the non-systematic encoder the input sequence $x_{\text{ns}}(D) = 1 + D + D^2 + D^4$ generates the same codeword, i.e.,

$$y(D) = x_{\text{ns}}(D)G_{\text{ns}}(D) = [1 + D^7 \quad 1 + D + D^4 + D^5 + D^6 + D^7]. \quad (4)$$

Note that, although the codeword is the same in both cases, the input sequence has weight two in the systematic case and weight four in the non-systematic case. ■

This example illustrates an important property of equivalent encoders, viz., the input sequences that generate a given codeword are different and may have different weights. In other words, although the two codes are identical, the encoder mapping between input sequences and codewords is different.

Non-systematic feedback encoders, which can be used to generate non-systematic recursive convolutional codes (NSRCC's), have not received much attention in the literature, since it can be shown that they are equivalent to systematic feedback encoders. For example, consider a rate $R = k/n = 1/2$ NSRCC generated by

$$G_{\text{ns}}(D) = [n_1(D)/d_1(D) \quad n_2(D)/d_2(D)]. \quad (5)$$

Multiplying $G_{\text{ns}}(D)$ by $\text{LCM}[d_1(D), d_2(D)]$ results in an equivalent non-systematic feedforward encoder, which in turn has an equivalent systematic feedback encoder.

A general encoder for the class of rate $1/2$ NSRCC's described above, realized using two shift registers, is shown in Figure 1. If we let $m_i = \max\{\deg[n_i(D)], \deg[d_i(D)]\}$ and assume that $n_i(D)$ and $d_i(D)$ have no common factors, $i = 1, 2$, then the encoder of Figure 1 has $2^{m_1+m_2}$ states. An interesting subclass of rate $1/2$ NSRCC's results when we require that $d_1(D) = d_2(D) = d(D)$, i.e.,

$$G_{\text{ns}}(D) = [n_1(D)/d(D) \quad n_2(D)/d(D)]. \quad (6)$$

In this case the NSRCC can be realized with a single shift register, as shown in Figure 2, and the encoder has only 2^m states, where $m = \max\{\deg[n_1(D)], \deg[n_2(D)], \deg[d(D)]\}$ and $n_1(D)$, $n_2(D)$, and $d(D)$ have no common factors. Because of its simpler description, this is the form of NSRCC we will assume throughout the paper. (Note that the NSRCC becomes systematic if either $n_1(D) = d(D)$ or $n_2(D) = d(D)$.)

¹Throughout the paper, the notation (n, k, m) will be used to denote a convolutional encoder with k input sequences, n output sequences, and memory order m .

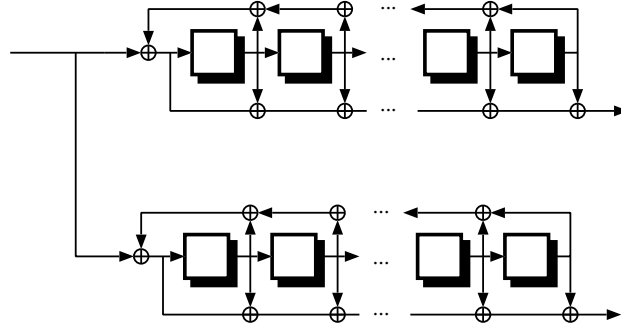


Figure 1: A general encoder for a rate 1/2 NSRCC.

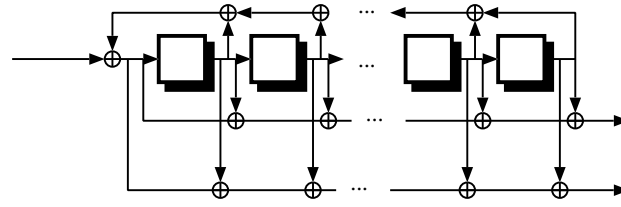


Figure 2: A rate 1/2 NSRCC realized with one shift register.

Example 2 Consider the (2,1,3) non-systematic feedback encoder given by

$$G_{\text{ns}}(D) = [(1 + D + D^3)/(1 + D^2 + D^3) \quad (1 + D + D^2 + D^3)/(1 + D^2 + D^3)]. \quad (7)$$

The equivalent systematic feedback encoder is given by

$$G_{\text{s}}(D) = [1 \quad (1 + D + D^2 + D^3)/(1 + D + D^3)]. \quad (8)$$

The input sequence $x_{\text{s}}(D) = 1 + D^7$ to the systematic encoder generates the codeword

$$y(D) = x_{\text{s}}(D)G_{\text{s}}(D) = [1 + D^7 \quad 1 + D^2 + D^3 + D^4 + D^6 + D^7], \quad (9)$$

whereas for the non-systematic encoder the input sequence $x_{\text{ns}}(D) = 1 + D + D^4 + D^5 + D^6 + D^7$ generates the same codeword, i.e.,

$$y(D) = x_{\text{ns}}(D)G_{\text{ns}}(D) = [1 + D^7 \quad 1 + D^2 + D^3 + D^4 + D^6 + D^7]. \quad (10)$$

Again note that the input sequences that generate the same codeword have different weights. ■

With standard decoding methods for convolutional codes such as the maximum likelihood (ML) or maximum a posteriori (MAP) algorithms, the performance of a code generated by equivalent encoders is expected to be essentially identical. In fact, if we plot the codeword bit error rate (CBER), the performance with ML decoding is exactly the same. However,

because of the different encoder mapping between information sequences and codewords, if we plot the information bit error rate (IBER)², the performance with either ML or MAP decoding will differ slightly. This is illustrated in Figure 3 for the two equivalent 8-state convolutional encoders of Example 2. However, the difference in performance is typically so small that it is commonly ignored.

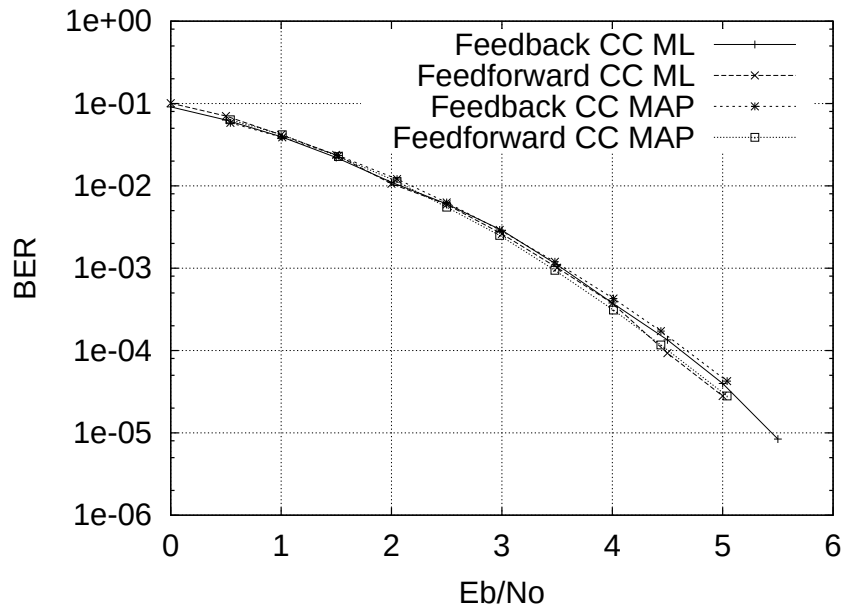


Figure 3: The performance of two equivalent convolutional encoders for ML and MAP decoding.

2 Turbo Codes With Equivalent Constituent Encoders

The recently discovered parallel concatenated convolutional codes, or turbo codes[4], have focused the attention of the research community on the properties of SRCC's, since feedback in the encoder is necessary to realize the astounding “interleaver gain” property of turbo codes. In other words, the performance of a turbo code degrades drastically when a non-systematic feedforward encoder is used as a constituent code instead of its equivalent systematic feedback encoder.

Example 3 Consider the two turbo codes illustrated in Figure 4, where a (2,1,2) non-systematic feedforward convolutional encoder with generator matrix

$$G_{\text{ns}}(D) = [1 + D + D^2 \quad 1 + D^2] \quad (11)$$

and its systematic feedback equivalent with generator matrix

$$G_{\text{s}}(D) = [1 \quad (1 + D^2)/(1 + D + D^2)] \quad (12)$$

²Throughout the remainder of the paper, we will use BER to refer to the information bit error rate.

are used as constituent codes in a rate 1/3 turbo code. Simulation results using iterative a

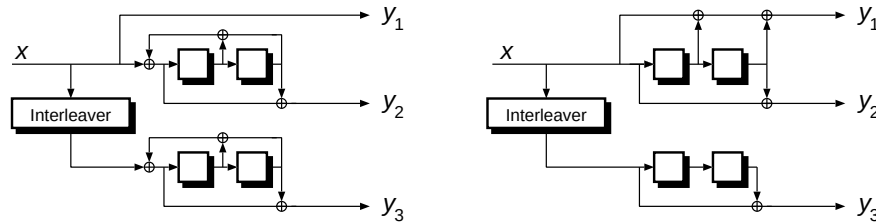


Figure 4: Two rate 1/3 turbo codes with equivalent (2, 1, 2) constituent encoders.

posteriori probability (APP) decoding show that, for an information block length of $N = 1024$ and a signal-to-noise ratio (SNR) of $E_b/N_0 = 1.76\text{dB}$, the BER of the systematic feedback encoder is 1.0×10^{-5} . On the other hand, the BER of the non-systematic feedforward encoder is 7.2×10^{-3} , almost three orders of magnitude worse. Consequently, non-systematic encoders have not previously been considered for use in turbo codes. ■

However, a recently proposed modified iterative decoding algorithm[5] has been shown to be able to effectively decode turbo codes whose constituent codes are NSRCC's rather than SRCC's. In other words, by replacing the non-systematic *feedforward* constituent encoders in Figure 4 with non-systematic *feedback* encoders, we remove the usual restriction that the constituent encoders must be systematic. This opens up a large class of additional constituent encoders that can be employed in turbo codes, viz., NSRCC's, thus creating the possibility of improved performance. An example of a rate 1/3 turbo code with the NSRCC constituent encoder given by (7) is shown in Figure 5. Note that it is only the top encoder that differs from the usual turbo code, since there is no systematic output sequence.

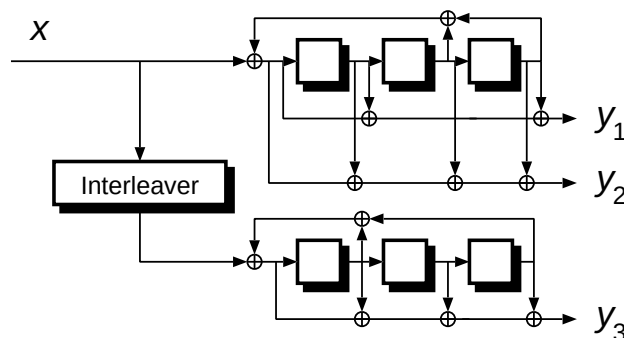


Figure 5: A rate 1/3 turbo code with an NSRCC constituent encoder.

As noted above, an NSRCC always has an equivalent SRCC, and thus it is reasonable to ask if there can be any advantage to using NSRCC's instead of SRCC's as constituent encoders in a turbo code. In other words, would we expect there to be any difference in the performance of the rate 1/3 turbo code shown in Figure 5 (with an NSRCC constituent

encoder) and the one shown in Figure 6, with the equivalent SRCC constituent encoder given by (8)?

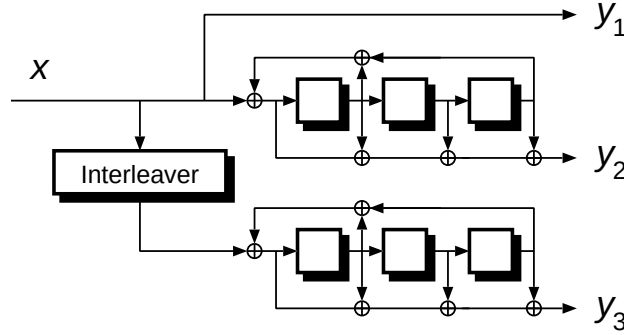


Figure 6: A rate 1/3 turbo code with an SRCC constituent encoder.

In Example 3, we saw that equivalent encoders, one feedback and one feedforward, resulted in drastically different BERs when employed as constituent encoders in a turbo code. Hence, even though two encoders may be equivalent, their performance when used as constituent encoders in a turbo code may be quite different. Thus it does make sense to consider the possible use of NSRCC's in turbo codes. In this paper we examine some of the properties of NSRCC's that may make them attractive alternatives to SRCC's for use in turbo codes.

3 Iterative Decoding of Turbo Codes with NSRCC Constituent Encoders

A block diagram of the recently proposed modified iterative algorithm[5] for decoding turbo codes with NSRCC constituent encoders is shown in Figure 7, along with the standard iterative decoder used with SRCC constituent encoders. Each APP decoder employs the usual BCJR algorithm[6].

As described in [7], the output probabilities of each APP decoder can be considered as the product of three terms: the channel values, the a priori probabilities, and the extrinsic probabilities, which are passed to the next decoder as a priori values. In Figure 7, we have adopted the following notation (the superscript $j = 1$ or 2 denotes values associated with APP decoder 1 or 2, respectively):

- x_t^j represents the total APP value (channel, a priori, and extrinsic),
- x_a^j represents only the a priori and extrinsic values,
- x_e^j represents only the extrinsic value,
- x_i^j represents the input values to each decoder,
- x_c^j represents the total APP value with the a priori value removed.

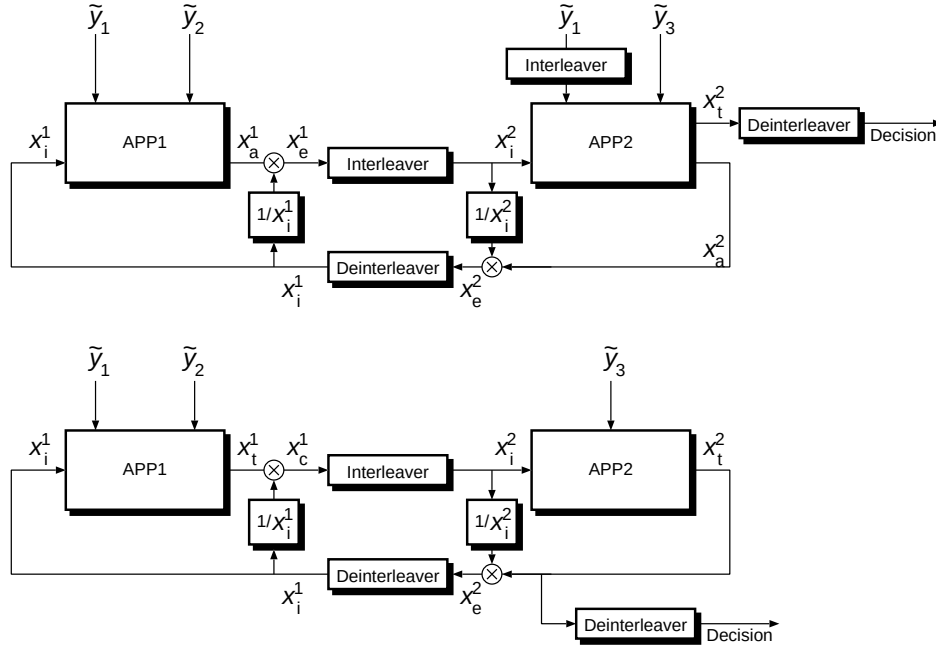


Figure 7: Decoder block diagrams for systematic (upper diagram) and non-systematic (lower diagram) turbo codes.

We see that decoder 1 is identical in the two cases. In the NSRCC case, however, decoder 1 passes the total APP value for each information bit, with only the a priori value removed, to decoder 2, whereas in the SRCC case, both the a priori and channel values are removed. This is because in the NSRCC case, there are no channel values for the information bits entering decoder 2. Thus soft values for the information bits must be passed to decoder 2 from decoder 1. These soft values are contained in the x_c^1 sequence. (Note that in the NSRCC case these soft values for the information bits are not received channel values.) The only difference in decoder 2 is that there is only one channel input in the NSRCC case. Also, the extrinsic values that are passed from the output of decoder 2 back to decoder 1 as a priori inputs are extracted from x_t^2 in the NSRCC case, whereas in the SRCC case they are extracted from x_a^2 .

Finally, it is worth noting that, when either y_1 or y_2 is a systematic bit, the modified iterative algorithm for decoding turbo codes based on NSRCC's shown in Figure 7 will also decode turbo codes based on SRCC's. The only difference from the standard decoder is that, instead of entering the soft received channel values for the systematic bits directly into decoder 2, they are passed through to the a priori input of decoder 2 along with the extrinsic values from decoder 1. Thus, in the case of SRCC's, the two decoders shown in Figure 7 yield identical results. However, the implementation differs in the two cases, because nothing in the modified decoder assumes the transmission of systematic bits, and hence it is a completely general and transparent decoder.

4 Effective Free Distance of NSRCCs

As noted in Example 2, every non-systematic feedback encoder has an equivalent systematic feedback encoder. So as normal convolutional codes with either ML or MAP decoding, their BER performance is essentially identical. However, equivalent encoders do not necessarily give equivalent performance in a turbo coding arrangement, as shown in Example 3. In particular, the encoder mapping between information sequences and code sequences, which has only a minor effect on performance for normal convolutional codes, plays a critical role in the performance of a turbo code.

This is due to the action of the interleaver which, after encoding by the first constituent code, scrambles the information sequence before encoding by the second constituent code. The weight of the information sequence is preserved by the interleaver, but the ordering of input bits to the two encoders is different. Since a weight 1 input sequence cannot terminate a feedback encoder, it must generate an infinite weight output sequence, and hence weight 1 inputs can be ignored in computing the low weight terms in the distance spectrum of turbo codes. Weight 2 inputs, however, can terminate a feedback encoder, and thus may lead to a low weight output sequence. If a weight 2 input sequence that terminates the first encoder, after interleaving, also terminates the second encoder, the overall codeword can have low weight. Thus codewords corresponding to weight 2 input sequences are important in determining the distance spectrum of turbo codes. For a typical random interleaver, relative to weight 2 input sequences, higher weight input sequences that terminate the first encoder are much less likely to also terminate the second encoder and generate a low weight codeword[8]. Thus, compared to weight 2 input sequences, higher weight input sequences contribute relatively few low weight codewords to the distance spectrum of turbo codes, and the codewords corresponding to weight 2 input sequences are said to determine the effective free distance d_{eff} of a turbo code. For this reason, d_{eff} is considered to be the single most important distance parameter determining turbo code performance[9].³

Definition 1 ([10]) The effective free distance d_{eff} of a turbo code is the minimum weight output sequence corresponding to an input sequence of weight 2. ■

The output of a systematic (n, k, m) convolutional encoder consists of k information sequences and $r = n - k$ parity sequences, where each of the last r columns, i.e., the non-systematic part, of the generator matrix generates a parity sequence. (In the case of a non-systematic encoder, every one of the n columns of the generator matrix generates a “parity” sequence.) Now let $g_i(D)$ denote the column of the generator matrix $G(D)$ that generates the i -th parity sequence ($1 \leq i \leq r$ for systematic encoders and $1 \leq i \leq n$ for non-systematic encoders) and call $g_i(D)$ the i -th parity generator.

Definition 2 ([11]) The weight-2 input minimum distance $d_2(i)$ of the i -th parity generator $g_i(D)$ of an (n, k, m) convolutional encoder is the minimum weight parity output sequence corresponding to an input sequence of weight 2. ■

³It is important to recognize that the relative importance of different weight input sequences on the distance spectrum of turbo codes depends critically on the interleaver design. Weight 2 input sequences have their greatest relative influence for random-like interleavers.

The following theorem follows directly from the above definition.

Theorem 1 *The weight-2 input minimum distance d_2 of an (n,k,m) convolutional encoder is given by*

$$d_2 = 2 + \sum_{i=1}^r d_2(i) \quad (\text{systematic case}) \quad (13)$$

$$d_2 = \sum_{i=1}^n d_2(i) \quad (\text{non-systematic case}) \quad (14)$$

Tables of d_2 -optimal code generators for systematic encoders with rates $R = k/n = 1/2, 2/3, 1/3, 3/4, 2/4$, and $1/4$ and memory orders as high as $m = 6$ are given in [11]. (We note from (13) and (14) that the tables in [11] for rate $1/3$ and $1/4$ systematic encoders also serve as d_2 -optimal non-systematic encoders for rates $1/2$ and $1/3$, respectively.) Since, except for trivial parity generators of the form $g_i(D) = D^j/(1 + D^m)$, $0 \leq j \leq m$, $d_2(i) \geq 2$, it is clear from (13) and (14) that non-systematic encoders can achieve larger values of d_2 than systematic encoders of the same complexity.

For the rate $1/3$ turbo codes considered in this paper, the first constituent encoder is either a systematic or non-systematic rate $1/2$ feedback convolutional encoder whose weight-2 input minimum distance is denoted either d_2^s or d_2^{ns} , respectively, and the second constituent encoder is a feedback parity generator whose weight-2 input minimum distance is denoted $d_2^{(2)}$. Then we can write the effective free distance of the corresponding rate $1/3$ turbo code as

$$d_{\text{eff}}^s = d_2^s + d_2^{(2)} \quad (\text{systematic case}) \quad (15)$$

$$d_{\text{eff}}^{ns} = d_2^{ns} + d_2^{(2)} \quad (\text{non-systematic case}) \quad (16)$$

Since larger values of d_2 can be achieved with non-systematic encoders than with systematic encoders of the same complexity, i.e., $d_2^{ns} \geq d_2^s$, (15) and (16) imply that turbo codes employing non-systematic rather than systematic constituent encoders can achieve larger values of d_{eff} . In Table 1, we list the largest possible effective free distances of rate $1/3$ turbo codes, for both systematic and non-systematic constituent encoders, constructed using d_2 -optimal code generators obtained from the tables in [11]. We see that, except for the trivial $m = 1$ case, turbo codes with non-systematic constituent encoders achieve the largest values of d_{eff} .

Although it is normal in practice to use a common parity generator for both constituent encoders in a turbo code (i.e., to use a symmetric design), it is also possible to use an asymmetric design in which the two constituent encoders do not have a parity generator in common[12]. (It should be noted that equations (15) and (16) for computing d_{eff} hold regardless of whether the turbo code is symmetric or asymmetric.) In fact, some of the non-systematic turbo codes⁴ listed in table 1 use asymmetric designs. For example, the 8-state

⁴Henceforth, we use the term “non-systematic turbo code” to refer to a turbo code constructed with non-systematic constituent encoders. Similarly, a “systematic turbo code” refers to a turbo code constructed with systematic constituent encoders.

m	d_{eff}^s	$d_{\text{eff}}^{\text{ns}}$
1	4	3
2	10	12
3	14	18
4	22	30
5	38	54
6	70	102

Table 1: Optimal effective free distances of rate 1/3 turbo codes.

($m = 3$) non-systematic rate 1/3 turbo code shown in Figure 5, which has optimal $d_{\text{eff}} = 18$, is asymmetric since a different feedback polynomial is used in the two constituent encoders.

The fact that non-systematic turbo codes have larger values of d_{eff} than systematic turbo codes of the same complexity makes it reasonable to expect that they will exhibit better performance. To test this hypothesis, we simulated the two “equivalent” 8-state turbo codes, one non-systematic and the other systematic, shown in Figures 5 and 6, respectively. (The first constituent encoder in the systematic code is equivalent to the first constituent encoder in the non-systematic code.) The non-systematic code has $d_{\text{eff}} = 18$, while the systematic code has $d_{\text{eff}} = 14$. The results of the simulation for an information block size of $N = 1024$ are shown in Figure 8.

We note that the non-systematic turbo code has a lower “error floor” than the systematic turbo code, i.e., it performs significantly better at high SNR’s. However, its performance is slightly worse than the systematic turbo code in the “waterfall” region of the BER curve, i.e., at low SNR’s. This example illustrates the well-known fact that minimum distance, in this case d_{eff} , is known to be the primary influence on performance at high SNR’s, i.e., in the “error floor” region of turbo codes, but at lower SNR’s other factors must also be considered. This is because at high SNR’s the iterative decoder is essentially maximum likelihood, but at low SNR’s the entire low weight portion of the distance spectrum must be considered, as well as the convergence properties of the iterative decoding algorithm.

Is it possible to eliminate the performance loss of non-systematic turbo codes in the “waterfall” region of the BER curve suggested by the above example while maintaining their superior performance performance in the “error floor” region? In the next section, we examine a possible explanation for the sub-par performance of non-systematic turbo codes in the “waterfall” region based on the distance spectrum of the code.

5 Non-Systematic Turbo Codes with Catastrophic Constituent Encoders

An anomaly noted in [11] occurs when convolutional encoders have repeated parity generators. For example, the systematic feedback (3, 1, 2) encoder with the best value of d_2 , viz., $d_2 = 10$, is given by the generator matrix

$$G_s(D) = \begin{bmatrix} 1 & (1 + D^2)/(1 + D + D^2) & (1 + D^2)/(1 + D + D^2) \end{bmatrix}. \quad (17)$$

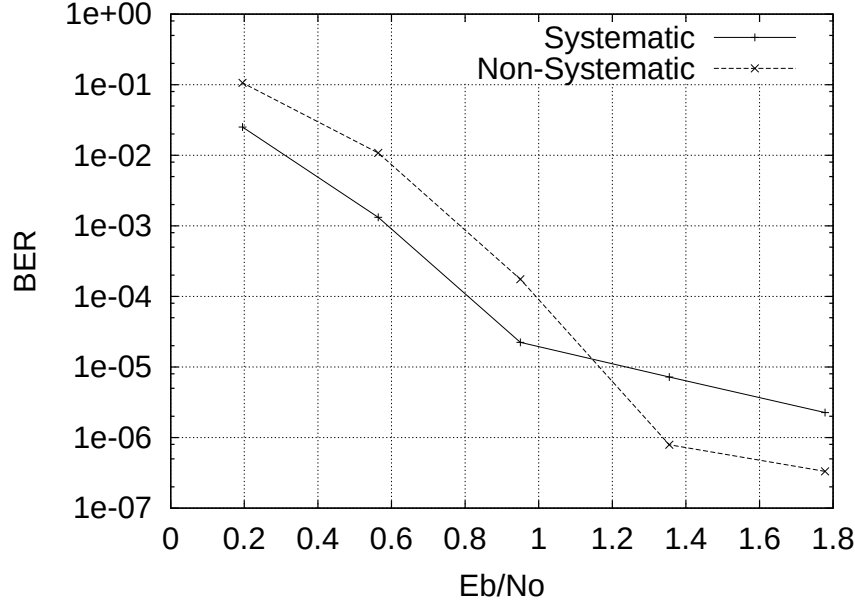


Figure 8: The BER performance of two “equivalent” 8-state non-systematic and systematic turbo codes.

However, it is stated in [11] that using such encoders as constituent encoders in a turbo code “yields poor results”, and whenever the encoder with the best value of d_2 contains repeated generators, as above, the tables in [11] also list the encoder without repeated generators having the best value of d_2 . A natural question arises. Why should the encoder of (17), with the best value of d_2 of any systematic feedback (3, 1, 2) encoder, perform poorly when used as a constituent encoder in a turbo code? (In this case, since the constituent encoder has rate 1/3, the associated turbo code would have rate 1/5.)

Insight into the answer to this question can be obtained by viewing the non-systematic portion of the above encoder as a (2, 1, 2) non-systematic encoder with generator matrix

$$G_{\text{ns}}(D) = \begin{bmatrix} (1 + D^2)/(1 + D + D^2) & (1 + D^2)/(1 + D + D^2) \end{bmatrix}. \quad (18)$$

We note that this encoder is catastrophic. However, in a non-systematic turbo code, it is not immediately obvious that the constituent encoder should not be catastrophic, since the randomization introduced by the interleaver implies that the overall encoder would still be non-catastrophic. In order to see if catastrophic encoders with large values of d_2 can be effectively used as constituent encoders in a turbo code, we simulated a rate 1/3, 4-state turbo code under two conditions:

1. with the non-systematic catastrophic encoder $G_{\text{ns}}(D) = \begin{bmatrix} g(D) & g(D) \end{bmatrix}$ as the first constituent encoder and the parity generator $g(D)$ as the second constituent encoder, and
2. with the systematic (and hence non-catastrophic) encoder $G_{\text{s}}(D) = \begin{bmatrix} 1 & g(D) \end{bmatrix}$ as the first constituent encoder and the parity generator $g(D)$ as the second constituent encoder,

where $g(D) = (1 + D^2)/(1 + D + D^2)$. Since $G_{\text{ns}}(D)$ has $d_2 = 8$, the effective free distance of the non-systematic turbo code is $d_{\text{eff}} = 12$, whereas $G_s(D)$ has $d_2 = 6$ and the effective free distance of the systematic turbo code is only $d_{\text{eff}} = 10$. The results are shown in Figure 9 for an information block size of $N = 1024$. We see clearly that the non-systematic turbo code performs much worse than the systematic turbo code in both the “waterfall” and “error floor” regions, even though it has a larger d_{eff} . Thus, in the design of non-systematic turbo codes, the use of catastrophic constituent encoders should be avoided. Also, since systematic encoders with repeated generators become catastrophic when the systematic part is removed, this may explain why such encoders should be avoided in the construction of systematic turbo codes, as noted in [11].

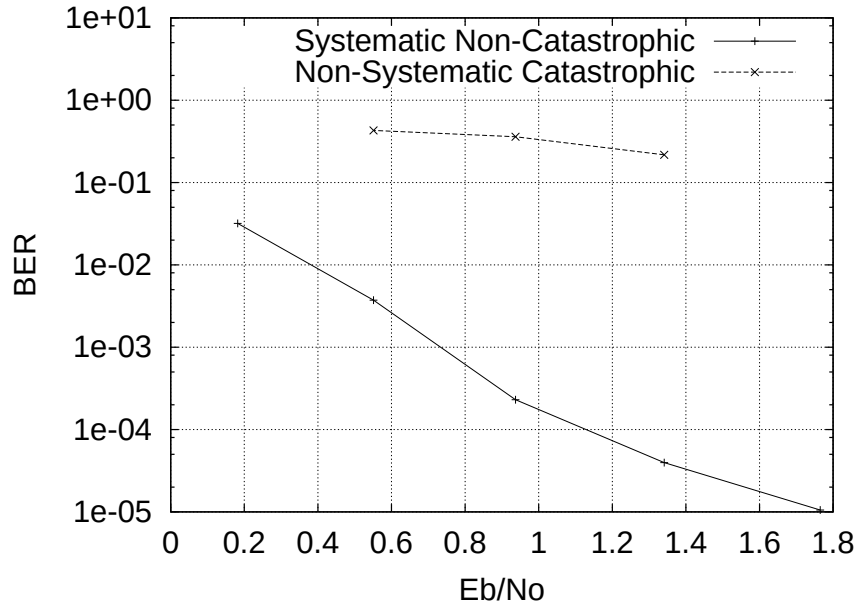


Figure 9: The BER performance of two 4-state turbo codes with non-catastrophic and catastrophic constituent encoders.

The probable explanation for the poor performance of non-systematic turbo codes with catastrophic constituent encoders is that the iterative decoding algorithm is ineffective in these cases. In other words, even though the turbo code is non-catastrophic and may have excellent distance properties, which would allow it to be effectively decoded with a maximum likelihood decoder, the constituent decoders cannot generate reliable extrinsic information to allow iterative decoding to converge. In the case of a rate 1/3 non-systematic turbo code whose first constituent encoder is catastrophic, the first constituent decoder is trying to decode the output of a rate 1/2 catastrophic encoder. For the low SNR's at which turbo codes operate, this leads to unreliable extrinsic values being passed to the second constituent decoder and thus the algorithm fails to converge.

For example, consider the above 4-state non-systematic turbo code. The catastrophic constituent encoder has a weight two codeword corresponding to the infinite weight input sequence $x(D) = (1 + D + D^2)/(1 + D^2)$. Since this codeword and the all-zero codeword differ

in only two positions, even moderate noise conditions can cause these two codewords to have almost the same APP value at the output of the corresponding decoder. Thus the associated log likelihood ratios, which are passed to the other decoder as extrinsic information, will be close to 0 for all the information bits in which the two codewords differ, i.e., all the extrinsic values will be close to 0. In other words, the extrinsic values passed to the other decoder contain no useful information, and the iterative decoding process will not converge.

This explanation suggests that, in order for a turbo decoder to properly converge, it is important that the low weight codewords in a constituent code be generated by even lower weight information sequences. This is always true for systematic codes, but it is not always true for non-systematic codes. In particular, catastrophic encoders have low weight codewords generated by infinite weight information sequences. Thus it is tempting to conjecture that the encoder mapping between information sequences and low weight codewords for the constituent encoders in a turbo code determines the convergence characteristics of the decoder⁵ and that this may explain why systematic turbo codes typically outperform non-systematic turbo codes in the “waterfall” region of the BER curve. The following example supports this conjecture.

Example 4 In this example, we examine the encoder mapping between information sequences and low weight codewords for the equivalent rate 1/2 encoders, one non-systematic and one systematic, whose generator matrices are given in (7) and (8), respectively. When used as constituent encoders in a turbo code, their relative performance was shown in Figure 8. From (7) and (8), we see that the information sequence

$$x_{\text{ns}}(D) = (1 + D^2 + D^3)u(D) \quad (19)$$

to the non-systematic encoder produces the codeword

$$y(D) = [(1 + D + D^3)u(D) \quad (1 + D + D^2 + D^3)u(D)], \quad (20)$$

and that the same codeword is produced by the information sequence

$$x_{\text{s}}(D) = (1 + D + D^3)u(D) \quad (21)$$

to the equivalent systematic encoder. Thus we can examine the encoder mapping between information sequences and low weight codewords for these two encoders by considering polynomials $u(D)$ of small degree. (Clearly, if the degree of $u(D)$ is large, the codeword $y(D)$ will have high weight.) Table 2 shows the mapping between information weight and codeword weight for these two encoders for all codewords $y(D)$ of weight 9 or less.

In this case we note that, for all reversible⁶ polynomials $u(D)$, the mapping between information weight and codeword weight is the same for the two encoders. Thus, only polynomials $u(D)$ that are not reversible can result in different mappings. For example, the polynomial $u(D) = 1 + D + D^3$, which is not reversible, produces a codeword of weight 7. In this case, the non-systematic encoder has information weight 7, whereas the systematic

⁵In a private communication, Urbanke indicated that this has been shown to be true in the case of a binary erasure channel.

⁶A polynomial $u(D)$ of degree m is said to be reversible if $u(D) = D^m u(D^{-1})$.

$w_H[y(D)]$	$w_H[x_s(D)]$	$w_H[x_{ns}(D)]$
6	4	4
7	3	3
7	3	3
7	3	7
8	2	6
8	4	4
8	4	4
8	4	8
8	6	6
9	3	7
9	3	7
9	5	5
9	5	5
9	5	5
9	5	5
9	5	5
9	5	5
9	5	9
9	5	9
9	5	9

Table 2: Mapping between information weight and codeword weight for two equivalent 8-state encoders.

encoder has information weight only 3. Similarly, all table entries with different mappings (viz., the italicized entries) have a lower information weight associated with the systematic encoder than with the non-systematic encoder. Hence we would expect the extrinsic values on the information bits passed to the other decoder to have fewer unreliable bits in the systematic case, thus leading to better decoder convergence for the systematic turbo code. Our conjecture is that this is at least part of the reason for the better “waterfall” region performance of the systematic turbo code shown in Figure 8. ■

Clearly the above example does not provide “proof” of our conjecture, and further investigation is needed before any conclusions can be drawn. In particular, it would be interesting to examine whether non-systematic encoders that satisfy the expansion property of systematic encoders, i.e., every information sequence produces a codeword of higher weight, result in good “waterfall” region performance when used as constituent encoders in a turbo code.

6 Conclusions

In this paper, we have re-examined the original work of Forney[1] on the equivalence of systematic and non-systematic convolutional encoders from the perspective of turbo coding and have reached the following conclusions:

- Equivalent convolutional encoders, when used as constituent encoders in a turbo code, do not produce similar results. Rather, the different encoder mapping between input and output sequences of equivalent encoders can have a substantial impact on the performance of the associated turbo codes.
- Turbo codes with non-systematic constituent encoders can be effectively decoded using a modification of the usual iterative APP decoding algorithm.
- Non-systematic turbo codes have larger effective free distances than systematic turbo codes of the same complexity. Thus they will have better performance in the “error floor” region of the BER curve.
- Catastrophic constituent encoders can produce non-catastrophic turbo codes with maximum effective free distance. However, iterative decoding cannot effectively decode such codes because of the decoder’s inability to converge.
- Systematic turbo codes generally have better “waterfall” region performance than non-systematic turbo codes. This may be explained by the fact that systematic constituent encoders, unlike most of their non-systematic counterparts, always map low weight information sequences into low weight codewords.

References

- [1] G. D. Forney, Jr., “Convolutional codes I: algebraic structure”, *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720-738, November 1970.

- [2] R. Johannesson and K. Sh. Zigangirov, *Fundamentals of Convolutional Coding*, New York: IEEE Press, 1999.
- [3] D. J. Costello, Jr., *Construction of Convolutional Codes for Sequential Decoding*, Ph.D. Dissertation, Dept. of Elec. Engr., Univ. of Notre Dame, Notre Dame, IN, August 1969.
- [4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes", in Proc. IEEE Int. Conf. on Commun., Geneva, Switzerland, pp. 1064-1070, May 1993.
- [5] O. M. Collins, O.Y. Takeshita, and D.J. Costello, Jr., "Iterative decoding of non-systematic turbo-codes", submitted to ISIT 2000, September 1999.
- [6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284-287, March 1974.
- [7] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol.42, pp. 429-445, March 1996.
- [8] S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol.42, pp. 409-428, March 1996.
- [9] 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, November 1996.
- [10] C. Berrou and A. Glavieux, "Turbo codes: general principles and applications," in Proc. 6th Tirrenia Int. Workshop on Digital Commun., Tirrenia, Italy, pp. 215-226, September 1993.
- [11] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *Electronics Letters*, vol. 32, pp. 445-446, February 1996.
- [12] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, Jr, "A note on asymmetric turbo-codes," *IEEE Communications Letters*, vol.3, pp. 69-71, March 1999.