

TURBO CODES: a tutorial on a new class of powerful error correcting coding schemes

Part I: Code Structures and Interleaver Design

S. Adrian Barbulescu (adrian@spri.levels.unisa.edu.au)

Institute for Telecommunications Research
University of South Australia

Steven S. Pietrobon (steven@sworld.com.au)

Small World Communications

Revised: 26 October 1998

Abstract

This is a tutorial paper meant to introduce the reader to the new concept of turbo codes. This is a new and very powerful error correction technique which outperforms all previous known coding schemes. It can be used in any communication system where a significant power saving is required or the operating signal-to-noise ratio is very low. Deep space communications, mobile satellite/cellular communications, microwave links, paging, etc., are some of the possible applications of this revolutionary coding technique.

Part I of the paper discusses the history of turbo codes, why they are different from traditional convolutional/block codes, turbo encoder structures and issues related to the interleaver design.

Part II will address the turbo decoder architecture, the achievable performance for turbo codes for a wide range of coding rates and modulation techniques, and will discuss delay and implementation issues.

1 Introduction

Increasing demand for information exchange is a characteristic of modern civilisation. The transfer of information from the source to its destination has to be done in such a way that the quality of the received information should be as close as possible to the quality of the transmitted

information. A typical communication system may be represented by the block diagram shown in Figure 1.1.

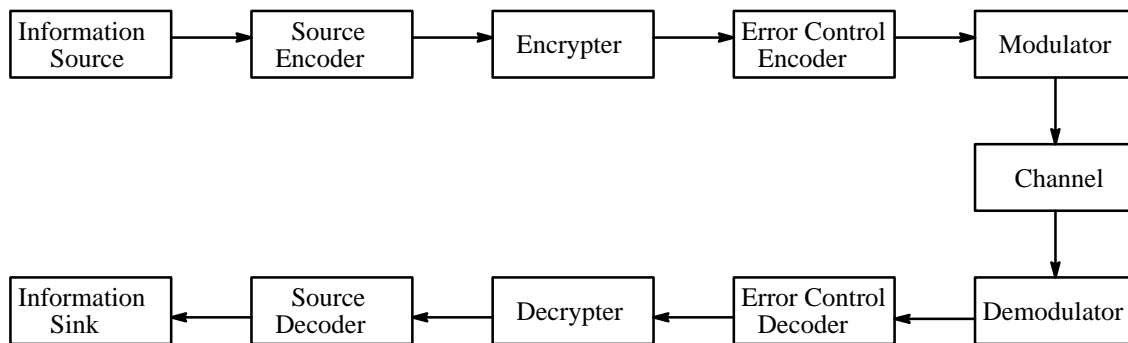


Figure 1.1: A block diagram of a communication system.

The information to be transmitted can be machine generated (*e.g.*, images, computer data) or human generated (*e.g.*, speech). Regardless of its source, the information must be translated into a set of signals optimised for the channel over which we want to send it. The first step is to eliminate the redundant part in order to maximise the information transmission rate. This is achieved by the source encoder block in Figure 1.1. In order to ensure the secrecy of the information we want to transmit, an encryption scheme must be used. The data must also be protected against perturbations introduced by the communication channel which could lead to misinterpretation of the transmitted message at the receiving end. This protection can be achieved through error control strategies: forward error correction (FEC), *i.e.*, using error correcting codes that are able to correct errors at the receiving end, or automatic repeat request (ARQ) systems. The modulator block generates a signal suitable for the transmission channel.

In this tutorial paper we discuss the error control aspects of a communication system. From coding theory [1], it is known that by increasing the codeword length or the encoder memory, greater protection, or coding gain, can be achieved. At the same time the complexity of typical decoding algorithm such as maximum likelihood decoding algorithms (MLDA) increases exponentially with the encoder memory and the algorithms become difficult to implement. The increased error correction capability of long codes requires a very high computational effort at the decoder. This has led to research for new coding schemes which could replace the MLDA with simpler decoding strategies.

These strategies combine simpler codes in such a way that allows each code to be decoded separately with less complex decoders. By using soft-in-soft-out (SISO) decoders, information can be passed from one decoder to the next in an iterative fashion. This is a “divide-and-conquer” strategy which in an iterative process can approach the performance of the MLDA.

Encoding techniques used in conjunction with iterative decoding combine different codes in such a way that each of them can be decoded independently. We include in this category product block codes [2], concatenated codes [3], multilevel codes [4, 5, 6], and, most importantly, turbo codes [7]. The common feature of all these techniques is that decoding can be done sequentially, using one decoder at a time (*i.e.*, after one code is decoded another decoder is used for the next code, and so on). This is a necessary condition for simpler decoding algorithms to replace the MLDA.

The structure of this paper is as follows. Section 2 presents the historical evolution of the concept on which turbo codes are built and summarises the differences between turbo codes and convolutional codes. Section 3 deals with the encoding process: different turbo encoder structures based on convolutional codes are given in Section 3.1, block turbo codes in Section 3.2 and turbo trellis coded modulation (TTCM) in Section 3.3. Section 4 addresses the interleaver design.

2 Why turbo codes?

Historically, product codes represented the first attempt at achieving a higher error correction capability without the decoding complexity required by long codes. Unfortunately, the alternative decoding of the component codes did not produce the improvements in performance as expected, even though the error correction power of the product code increased as a whole. This was mostly due to the hard-in-hard-out (HIHO) decoding algorithms that were traditionally used. There is one exception, the work on low-density parity-check codes by Gallager [8] which was rediscovered after thirty years with the advent of turbo codes. It should be mentioned that Gallager used single-error detecting parity-check codes. His work was extended recently [9] to single-error correcting Hamming codes with very promising results.

Better results than for block codes were obtained with concatenated codes. The component codes are called the inner and the outer codes. First, the decoding is done for each inner code vector. The decoded bits are then decoded again according to the outer code used [3].

Multilevel coding uses several error-correcting codes with different error correcting capabilities. The transmitted symbols are constructed by combining symbols of codewords of these codes. In [5], a staged decoder for a multilevel partition code passes reliability information only in one direction, from the first decoding stage through intermediate stages to the last one.

The new class of turbo codes encodes the same information twice, but in a different order. The more “scrambled” the information sequence is for the second encoder, the more “uncorrelated” the information exchange is between the decoders. This is one of the key ideas that allows a continuous improvement in correction capability when the decoding process is iterated.

In the traditional approach, the demodulator block from Figure 1.1 makes a “hard” decision of the received symbol and passes it to the error control decoder block. This is equivalent to deciding which of two logical values – say 0 and 1 – was transmitted. No information was passed about how reliable the hard decision was. Better results are obtained when the quantised analogue received signal was passed directly to the decoder.

This led to the development of “soft” input decoding algorithms. This was the best solution if only one code was used. For the same reason, in the case of combining more codes as explained above, new SISO algorithms were developed in order to pass more information from the output of one decoder to the input of the next decoder.

Soft output decision algorithms provide as an output a real number which is a measure of the probability of error in decoding a particular bit. This can also be interpreted as a measure of the reliability of the decoder’s hard decision. This extra information is very important for the next stage in an iterative decoding process, as will be shown later. There are two important categories of soft output decision algorithms. The first category includes the maximum likelihood decoding algorithms which minimise the probability of symbol error, such as the *maximum a posteriori* (MAP) algorithm [10, 11]. The second category includes the maximum likelihood decoding

algorithms which minimise the probability of word or sequence error, such as the Viterbi algorithm [12] or soft output Viterbi algorithm (SOVA) [13].

The inner code's primary function is to change the distribution of errors and to effectively increase the signal to noise ratio of the received signal. The inner decoder can be thought of as a noise filter. It can be shown that the channel capacity of a discrete-input real-output memoryless channel (C_{soft}) is greater than that for a discrete-input discrete-output (hard output) memoryless channel (C_{hard}) [14, 15] (see Figure 2.1).

The most important conclusion from Figure 2.1 is that C_{soft} is greater than C_{hard} by approximately 2 dB at low signal to noise ratios. This is the main reason for using soft output algorithms.

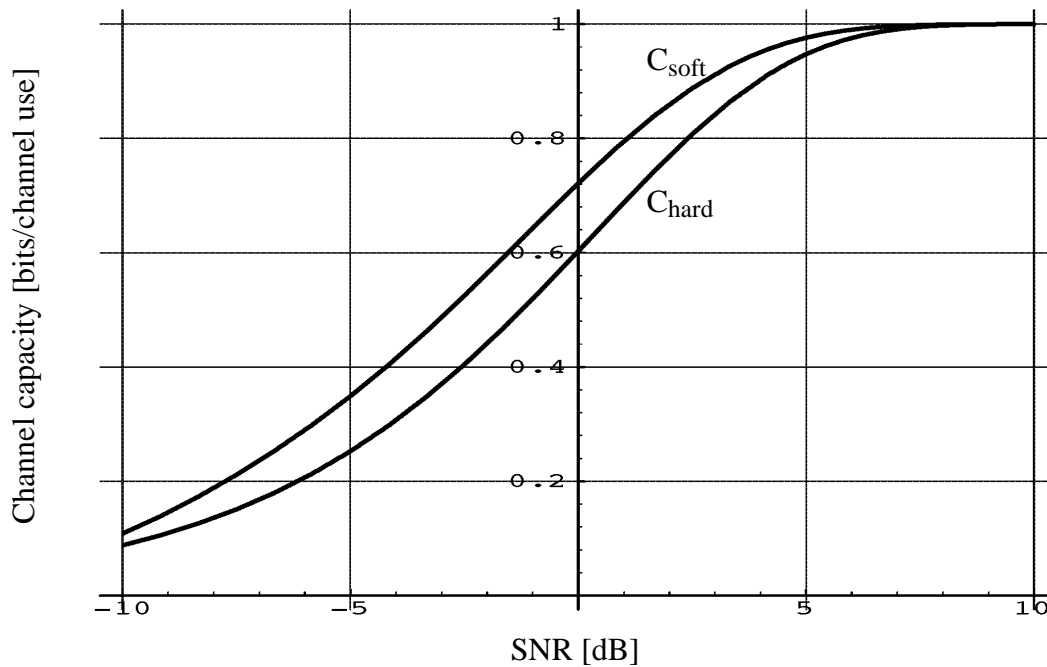


Figure 2.1: C_{soft} and C_{hard} as a function of signal-to-noise ratio (SNR).

Iterative decoding relies on a link between soft output decision algorithms, special encoding, and information transfer techniques. Since the early 1990's, these concepts have been combined to create more powerful decoders. The link between the three created a new powerful decoding technique and led to the appearance of turbo codes, which made possible communications very close to channel capacity.

The definition of turbo codes was given for the first time in [7]. They represent a particular class of parallel concatenation of two recursive systematic convolutional codes. Since then, the

term “turbo codes” has been extended to different concatenation schemes as explained in the following section.

There are a few differences between the behaviour of turbo codes and convolutional codes. It is well known that the performance of convolutional codes improves with increasing constraint length (a measure of code complexity) as shown in Figure 2.2. This is not the case for turbo codes: the best constituent codes of turbo codes have a very small constraint length as will be seen in Part II in the Section 3 on performance.

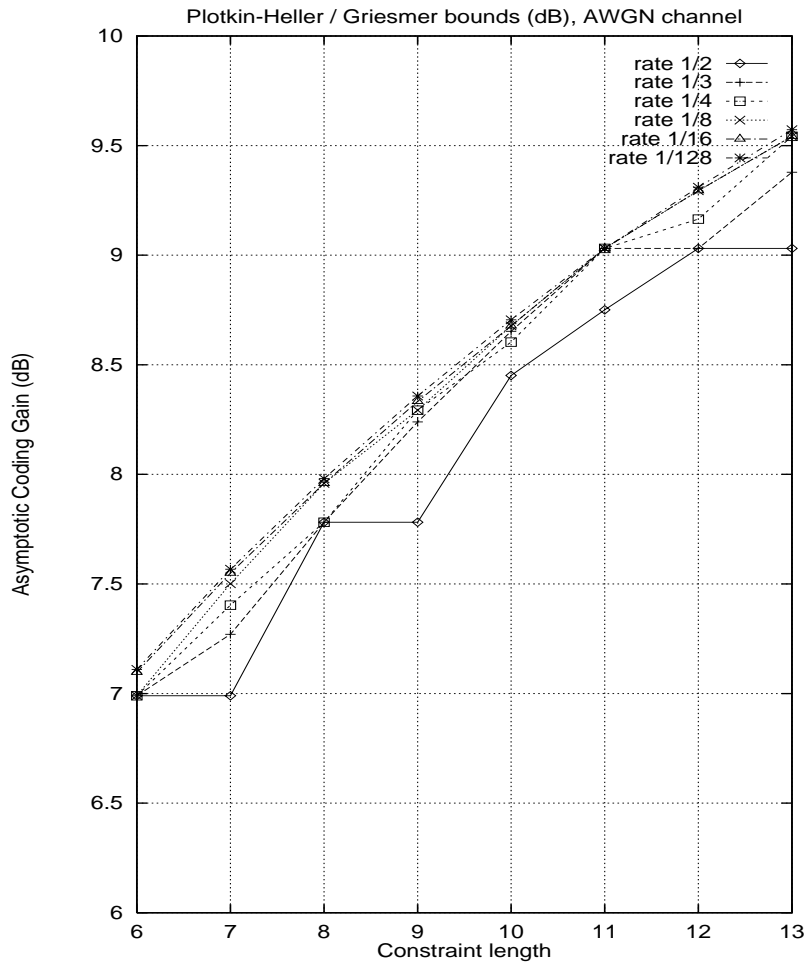


Figure 2.2: Convolutional code performance.

The performance of convolutional codes does not improve significantly with decreasing code rate as shown by the bounds in Figure 2.2.. The difference between rate 1/3 and rate 1/128 is of the order of a few tenths of one dB for convolutional codes. As we shall show later, turbo codes, achieve a very significant coding gain for lower coding rates.

Practical systems typically employ code rates between 1/2 and 1/6. For systems of this type Table 2.1 shows the E_b/N_0 (the SNR per information bit) required to achieve a bit error rate (BER)

of 10^{-6} as a function of coding rate for both turbo codes (10,000 bit interleaver size) and convolutional codes [16, 17].

Table 2.1: E_b/N_0 required to achieve a BER = 10^{-6} for convolutional codes (CC) and turbo codes (TC).

Coding rate	E_b/N_0 [dB]		E_b/N_0 difference [dB]	
	CC	TC	CC	TC
1/2	4.80	0.98	0.00	0.00
1/3	4.49	0.37	0.31	0.61
1/4	4.37	0.13	0.43	0.85
1/6	4.28	-0.12	0.52	1.10

As can be seen from Table 2.1, the turbo codes' E_b/N_0 difference is about twice the convolutional codes' E_b/N_0 difference. Thus it can be concluded that lower rate turbo codes provide significantly more coding gain than lower rate convolutional codes.

From the implementation point of view, recursive encoders and soft output decoders are essential in a turbo code scheme, whereas they do not matter for convolutional codes except in a concatenated configuration.

Therefore, the most important differences between convolutional codes and turbo codes may be summarised in Table 2.2.

Table 2.2: Comparison between convolutional codes (CC) and turbo codes (TC).

Criteria	CC	TC
Larger constraint length	Good	Bad
Larger free distance	Good	indifferent
Lower coding rate	indifferent	Good
Recursive encoders	indifferent	Good
Soft output decoders	indifferent	Good

3 Encoding

3.1 Turbo Code Structures

In Figure 3.1 we present a generic turbo encoder in two dimensions. The input sequence of the information bits is organised in blocks of length N. The first block of data will be encoded by

the ENC^- block which is a rate half recursive systematic encoder. The same block of information bits is interleaved by the interleaver INT, and encoded by $ENC^|$ which is also a rate half systematic recursive encoder. The coded bit produced by the encoder is the output of each encoder block.

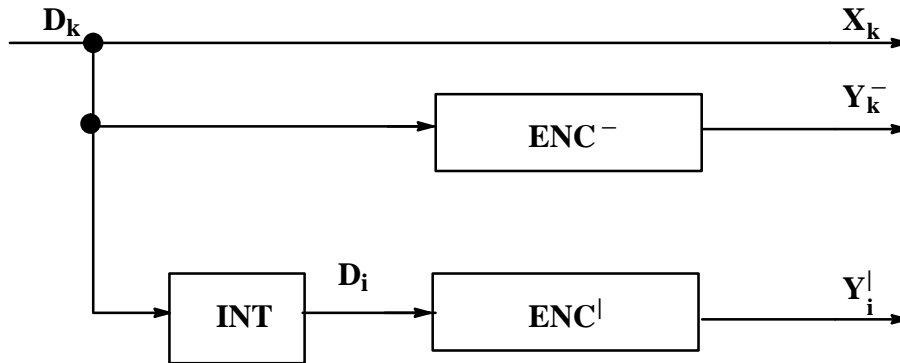


Figure 3.1: Generic rate 1/3 turbo encoder.

Due to similarities with product codes, we can call the ENC^- block the encoder in the “horizontal” dimension and the $ENC^|$ block the encoder in the “vertical” dimension. The interleaver block, INT, rearranges the order of the information bits for input to the second encoder. The main purpose of the interleaver is to increase the minimum distance of the turbo code such that after correction in one dimension the remaining errors should become correctable error patterns in the second dimension.

Ignoring for the moment the delay for each block, we assume both encoders output data simultaneously. This is a rate 1/3 turbo code, the output of the turbo encoder being the triplet $(\mathbf{X}_k, \mathbf{Y}_k^-, \mathbf{Y}_k^|)$. This triplet is then modulated for transmission across the communication channel, which for the purpose of this paper, is assumed to be additive white Gaussian noise (AWGN) channel. Since the code is systematic, $\mathbf{D}_k = \mathbf{X}_k$ is the input data at time \mathbf{k} . \mathbf{Y}_k^- and $\mathbf{Y}_k^|$ are the two parity bits at time \mathbf{k} .

The two encoders do not have to be identical. In Figure 3.1 the two encoders are rate 1/2 systematic encoders with only the parity bit shown. The parity bits can be “punctured” as in Figure 3.2 where puncturing is implemented by a multiplexing switch in order to obtain higher coding rates.

A rate 1/2 turbo code can be implemented by alternatively selecting the outputs of the two encoders in order to produce the following output sequence:

$$(X_1, Y_1^-, X_2, Y_2^+, X_3, Y_3^-, X_4, Y_4^+ \dots).$$

The symbol \boxed{D} represents a D flip-flop (the clock is not shown) and the symbol \oplus represents an exclusive-OR gate.

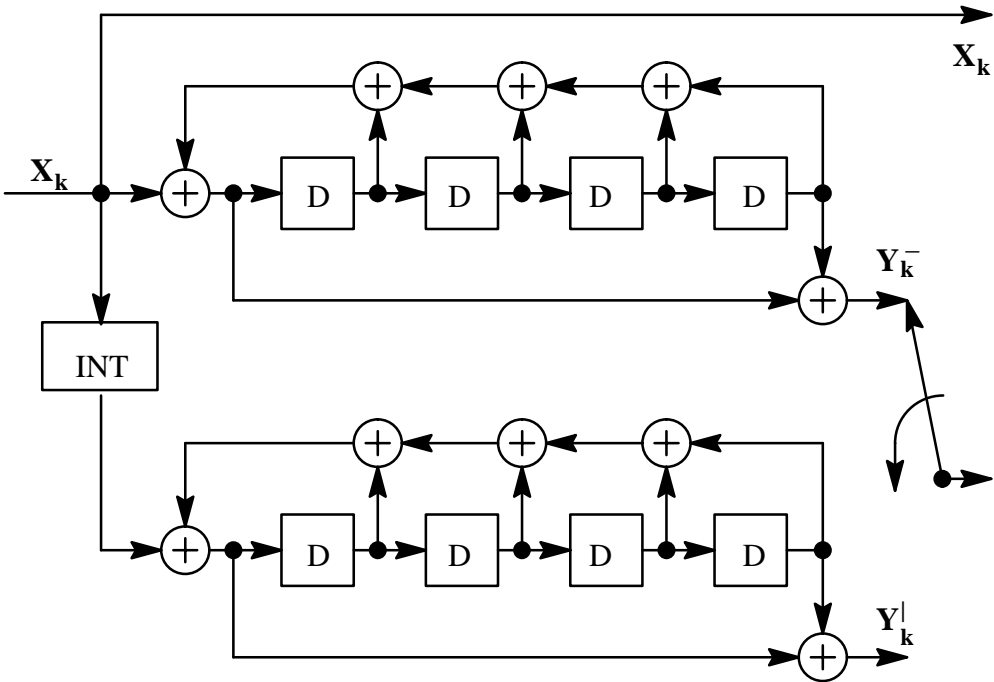


Figure 3.2: A rate half turbo code.

Figure 3.2 shows a particular implementation of a two dimensional turbo code using recursive systematic codes (RSC). Lower coding rates can be achieved using either less puncturing or more interleaver and encoder blocks as shown in Figure 3.3 for an n -dimensional turbo code. The advantage of using more interleavers will become clear in Part II which presents the performance of low rate turbo codes.

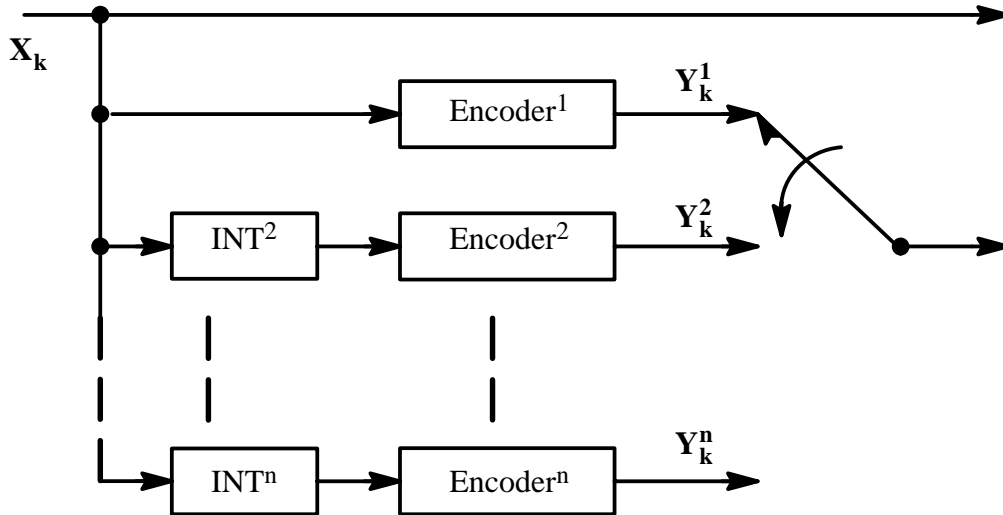


Figure 3.3: Low rate turbo encoder.

The turbo code presented in [7], achieved a BER $< 10^{-5}$ within 0.7 dB of Shannon capacity [18]. The basic turbo encoder has two constituent convolutional encoders separated by a random interleaver. This structure is called the parallel concatenated convolutional code (PCCC) structure since the same information stream is encoded twice, in parallel, using the straight and interleaved sequences of the information bits.

An obvious alternative is to interleave the output of one encoder and re-encode it again. This is called the serial concatenated convolutional code (SCCC) structure. This structure has been proposed in [19, 20, 21]. Any other possible combination of PCCC and SCCC can be used. One particular structure defined in [23, 24] was labelled as a hybrid concatenated convolutional code (HCCC) structure. A brief description of these code structures is given below.

3.1.1 PCCC structures

The first commercial communication system that used turbo codes [25, 26] was designed at the Institute for Telecommunications Research (ITR). It was based on a PCCC structure as shown in Figure 3.1.

The PCCC encoder structure was described above. The first turbo codec chip [27] used a rate half PCCC structure. This structure was at the time and still is the most power efficient code structure for a targeted BER greater than 10^{-6} under specific system constraints.

However, for much lower BER requirements, like 10^{-10} for data transmissions, the PCCC structure might not be the optimum solution. Based on analytical studies [28] and from our experience in implementing PCCC in hardware and software [26, 29], a change in the slope of the

BER curve appears for $\text{BER} < 10^{-7}$. The slope is a function of the interleaver size and interleaver design. This is discussed in Section 1 Part II.

3.1.2 SCCC structures

An alternative to PCCC is the SCCC structure. An example of a rate 1/3 SCCC is shown in Figure 3.4. Using the concept of a “uniform interleaver” (see Section 4.8) it was shown in [20, 21] that there is a great difference in the interleaver gain between PCCC and SCCC structures. For PCCC structures the interleaver gain is defined by a multiplication factor of N^{-1} in the BER bound. For SCCC structures, the interleaver gain is defined by $N^{-\frac{d_0+1}{2}}$ where d_0 is the free distance of the outer code (e.g., encoder 1 in Figure 3.4). For example, for an outer code with free distance $d_0 = 5$, the interleaver gain is N^{-3} .

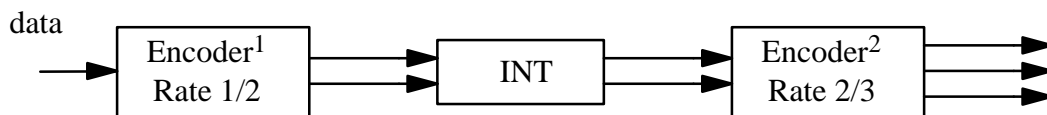


Figure 3.4: Serial concatenated convolutional encoder.

The results from [21] show a coding gain of the order of 2 dB for the serial scheme over the parallel scheme at a BER of 10^{-11} for the codes considered.

Performance of serial and parallel concatenated convolutional schemes with iterative decoding techniques for different interleaver designs were investigated in [21, 22]. Figure 3.5 shows initial results obtained for the parallel and serial concatenated schemes, respectively, based on inner and outer convolutional codes, and 8 iterations.

For the parallel concatenated scheme illustrated in Figure 3.5, both the inner and outer codes are identical rate 2/3 16 state RSC codes. For the serial concatenated scheme, the outer code is the same RSC code as used in the parallel scheme, while the inner code is a rate 3/4 16 state RSC code. Serial and parallel schemes having the same delay (60, 600, 6000) are compared. It is characteristic for PCCC schemes to perform better than SCCC schemes at low SNRs. However, increasing the SNR, SCCC schemes outperform PCCC schemes. The cross-over point depends on the interleaver size and interleaver design.

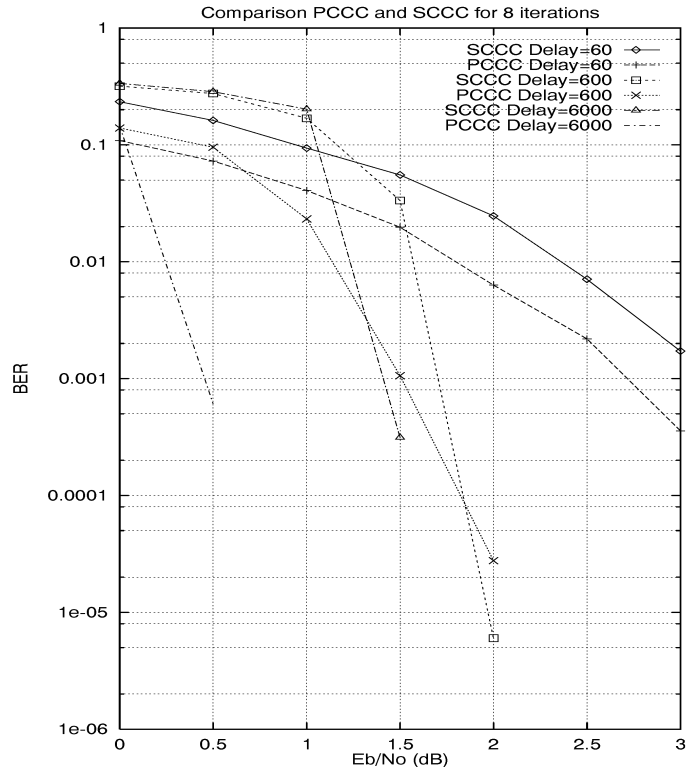


Figure 3.5: PCCC and SCCC comparison.

3.1.3 HCCC structures

A further alternative code structure is the recently reported hybrid scheme [23]. This scheme is a combination of serial and parallel concatenation and may offer performance advantages over either PCCC or SCCC structures. A HCCC structure is shown in Figure 3.6.

As for the SCCC structures, the interleaver gain for the HCCC depends on the free distance of the outer code in the serial concatenation part of the HCCC (i.e., Encoder¹ in Figure 3.6). The multiplication factor is N^{-d_0} for the BER bound, where d_0 is the free distance of the outer code. Therefore the HCCC structure is a further improvement on the SCCC structure.

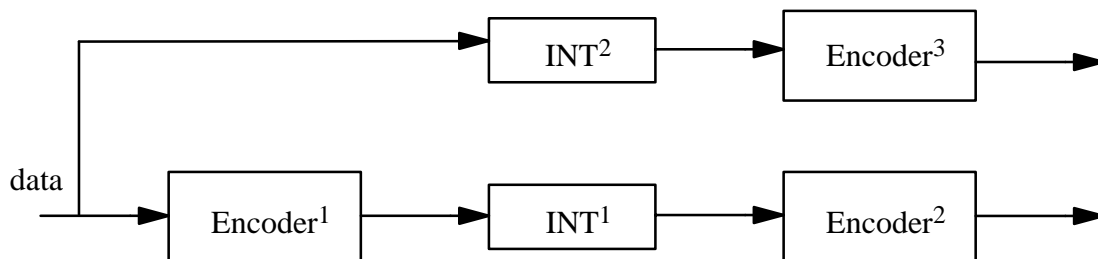


Figure 3.6: Hybrid concatenated convolutional encoder.

We are currently investigating the possibility of other hybrid schemes and optimising the constituent codes.

3.2 Block Turbo Codes

Iterative decoding of two concatenated block codes was introduced in [30 – 32]. The same structures as described in the previous section can be applied to block codes.

Block turbo codes are suitable for high code rates, typically with rates greater than 0.7, for systems that require high spectral efficiencies. Their performance does not depend on the interleaver design [33, 14] because the rows and columns of concatenated block codes are independent. Random interleaving of data bits does not produce any significant improvement. Serial concatenation is shown to be more efficient than parallel concatenation at low BER.

Results from [31] and [34] conclude that for spectral efficiencies greater than 4 bits/s/Hz, block turbo coded quadrature amplitude modulation (QAM) systems outperform convolutional coded turbo coded QAM. The performance improvements of block coded based turbo codes applied to quadrature phase shift keying (QPSK) is not as great as that obtained for 16QAM. Another interesting observation from [34] is that block turbo codes using 16QAM are not degraded in performance when the input data is quantised to only 4 bits, unlike convolutional based turbo codes which exhibit a minimum degradation of 0.5 dB. The product codes used in [31] and [34] were generated using identical Bose–Chaudhuri–Hocquenghem (BCH) codes ($C_1=C_2$).

Similarly, [35] reports the performance of a sub–optimum iterative decoding algorithm for two dimensional product codes using Reed–Solomon (RS) [9, 15] codes and a Hamming (7,4) code. The resultant minimum Hamming distance is 147 and the code rate is 0.2. Results indicate that a BER of 10^{-6} would be possible at an E_b/N_0 of 2.5 dB.

One advantage of using block codes is that error detection can be performed after iterative decoding without using additional check bits for error detection. This solution is attractive for small data block time division multiple access (TDMA) systems with high code rates and where ARQ systems are in place.

As an example of the performance that can be achieved with block turbo codes, a (512, 502, 4)² BCH code achieved a BER $< 10^{-5}$ at 0.8 dB and 0.45 dB from Shannon’s limit after four

iterations and forty iterations, respectively [33]. The iterative decoding process is similar to turbo codes.

A comparison is made in [21] between parallel concatenated block codes (PCBC) using systematic cyclic codes and serial concatenated block codes (SCBC) using Hamming and BCH codes. The SCBCs perform better than PCBCs, but significantly worse than the equivalent structures based on convolutional codes.

3.3 Turbo TCM

The body of knowledge of turbo code has been expanded to encompass trellis coded modulation concepts [36, 37]. Trellis coded modulation (TCM) can be applied to both PCCC [37] and SCCC [22]. Turbo trellis coded modulation (TTCM) only performs marginally better than a Gray coded QAM structure with rate 1/2 turbo coding in an AWGN channel. This is shown graphically in Figure 3.7 where a turbo TCM 16QAM structure [38] is compared to a rate 1/2 turbo code with 16QAM [39].

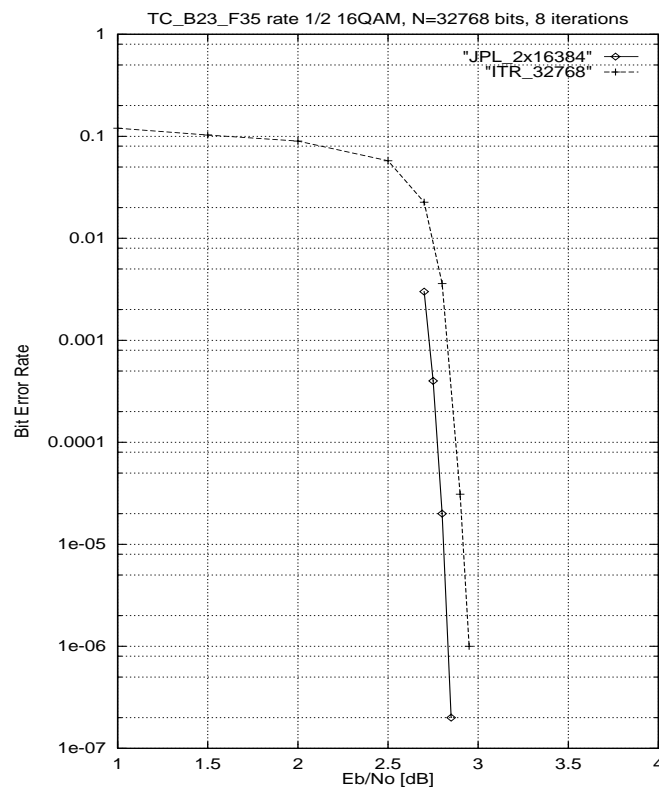


Figure 3.7: BER for 32,768 bits interleaver size, AWGN channel.

The turbo TCM transmitter and receiver are more complicated to implement than the much simpler 16QAM scheme with the rate 1/2 turbo decoder proposed in [39].

TTCM structures have also been proposed in [36]. They offer improvements over conventional TCM techniques. The same authors investigated in [40] TTCM schemes for 8PSK, 16QAM and 64QAM modulation schemes with varying overall bandwidth efficiencies. The simulation results are only for BER higher than 10^{-6} and show an “error floor” that can not be lowered below the 10^{-6} threshold.

4 Interleaving

The interleaver design is a key factor which determines the good performance of a turbo code. Some interleaver types used in turbo codes are presented in the following sections.

4.1 “Row–Column” interleaver

The simplest interleaver is a memory in which data is written row–wise and read column–wise. This is called a “row–column” interleaver and belongs to the class of “block” interleavers. For example, data is written as shown in Table 4.1.

Table 4.1: Writing data row–wise in memory.

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9
x_{10}	x_{11}	x_{12}
x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}
x_{19}	x_{20}	x_{21}

The interleaving process consists in reading data as shown in Table 4.2.

Table 4.2: Reading data column–wise from memory.

x_1	x_4	x_7	x_{10}	x_{13}	x_{16}	x_{19}	x_2	x_5	x_8	x_{11}	x_{14}
-------	-------	-------	----------	----------	----------	----------	-------	-------	-------	----------	----------	-----	-----	-----

A few interesting constructions for other block interleavers are given below.

4.2 “Helical” interleaver

A “helical” interleaver writes data row–wise as in Table 4.1 but reads data diagonal–wise as shown in Table 4.3.

Table 4.3: Reading data diagonal–wise from memory.

x_{19}	x_{17}	x_{15}	x_{10}	x_8	x_6	x_1	x_{20}	x_{18}	x_{13}	x_{11}	x_9
----------	----------	----------	----------	-------	-------	-------	----------	----------	----------	----------	-------	-----	-----	-----

4.3 “Odd–even” interleaver

We found that for a rate half encoder as shown in Figure 3.2, a particular type of interleaver called “odd–even”, gives significant improvements when used in a turbo encoder design [41]. Let us assume that we have a random sequence of binary data input to a rate one half systematic encoder and we store only the odd–positioned coded bits, as shown in Table 4.4.

Table 4.4: Odd–positioned coded bits.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
y_1	–	y_3	–	y_5	–	y_7	–	y_9	–	y_{11}	–	y_{13}	–	y_{15}

If we were now to interleave the same sequence of binary data in a pseudo–random order, encode it and store the even–positioned coded bits, the result would be as in Table 4.5.

Table 4.5: Even–positioned coded bits.

x_a	x_b	x_c	x_d	x_e	x_f	x_g	x_h	x_i	x_j	x_k	x_l	x_m	x_n	x_o
–	z_b	–	z_d	–	z_f	–	z_h	–	z_j	–	z_l	–	z_n	–

The data which is actually sent through the channel is shown in Table 4.6; the original sequence of information bits x_i , $i = 1, \dots, 15$ as in Table 4.4 and a multiplexed sequence of the odd– and even–positioned coded bits from Tables 4.4 and 4.5.

Table 4.6: The output for a pseudo–random interleaver.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
y_1	z_b	y_3	z_d	y_5	z_f	y_7	z_h	y_9	z_j	y_{11}	z_l	y_{13}	z_n	y_{15}

In Table 4.4 all the odd–positioned information bits have their own coded bit. Due to the pseudo–random way of interleaving, some of the coded bits stored in Table 4.5 can be for even–positioned information bits and some for odd–positioned information bits. This means that some of the information bits will have two coded bits associated with them and others will have no coded bit associated with them. Thus, the coding power is not uniformly distributed across all the bits. So for errors which affect information bits not associated with any coded bit the decoder will perform worse in both dimensions.

An example of an “odd–even” type of interleaver is a block interleaver with an odd number of rows and an odd number of columns as in Table 4.7, in which we store row–wise the sequence of random data.

Table 4.7: 3x5 block interleaver.

x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

We produce the coded bits and store only the odd–positioned coded bits as in Table 4.4. Now we read column–wise, encode and store the even–positioned coded bits as in Table 4.8.

Table 4.8: Even–positioned coded bits.

x_1	x_6	x_{11}	x_2	x_7	x_{12}	x_3	x_8	x_{13}	x_4	x_9	x_{14}	x_5	x_{10}	x_{15}
–	z_6	–	z_2	–	z_{12}	–	z_8	–	z_4	–	z_{14}	–	z_{10}	–

In Table 4.8 all the even–positioned information bits have their own coded bit present and in Table 4.4 all the odd information bits have their own coded bit present as well. When we multiplex the coded bits from both Table 4.4 and Table 4.8 we produce the coded sequence as in Table 4.9. This means that each of the information bits will have its own associate coded bit associated with it. Thus the coding power is now uniformly distributed.

Table 4.9: Information bits and multiplexed coded bits for an “odd–even” interleaver.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
y_1	z_6	y_3	z_2	y_5	z_{12}	y_7	z_8	y_9	z_4	y_{11}	z_{14}	y_{13}	z_{10}	y_{15}

The interleaver showed in Table 4.1 is also an “odd–even” interleaver.

4.4 “Simile” interleaver

In Section 4.3 we introduced an “odd–even” type of interleaver where each information bit is associated with one and only one coded bit. In this way the correction capability of the code is uniformly distributed over all information bits. We now impose another restriction on the interleaver design: after encoding both sequences of information bits, (the original and the interleaved one), the state of both encoders of the turbo code are to be the same. This allows only

one “tail” to be appended to the information bits, which drives both encoders to the same zero state. This is why we called it a “simile” type of interleaver [42].

The idea behind the simile interleaver is that the whole block of N information bits can be divided in $\nu + 1$ sequences, where ν is the memory length of the code. For $\nu = 2$, we get:

$$\text{Sequence 0} = \{d_k \mid k \bmod (\nu + 1) = 0\}$$

$$\text{Sequence 1} = \{d_k \mid k \bmod (\nu + 1) = 1\}$$

$$\text{Sequence 2} = \{d_k \mid k \bmod (\nu + 1) = 2\}$$

For example, consider the particular four state encoder shown in Figure 4.1.

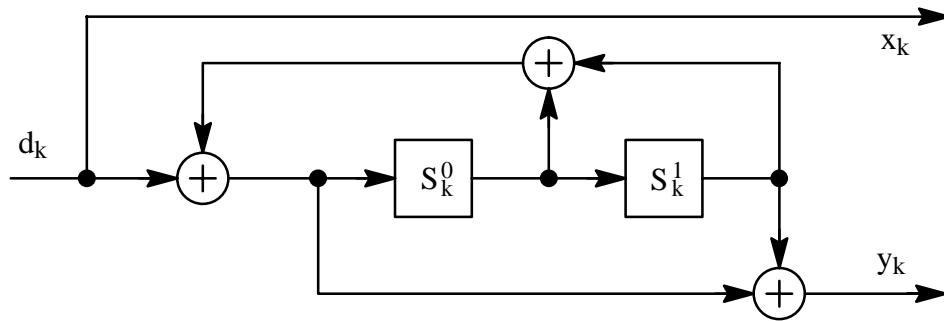


Figure 4.1: A four state systematic convolutional encoder.

For a given N , the final state of the encoder represented by the state of the two D flip-flops will be a combination of the above sequences as shown in Table 4.10.

Table 4.10: Final encoder state for $\nu = 2$ for N information bits.

$N \bmod (\nu + 1)$	S_N^0	S_N^1
0	Sequence1 + Sequence2	Sequence0 + Sequence1
1	Sequence2 + Sequence0	Sequence1 + Sequence2
2	Sequence0 + Sequence1	Sequence2 + Sequence0

The important conclusion is that from the point of view of the final encoder state, the order of the individual bits in each sequence does not matter, as long as they belong to the same sequence. The simile interleaver has to perform the interleaving of the bits within each particular sequence in order to drive the encoder to the same state as that which occurs without interleaving. Since both encoders end in the same state, we need only one tail to drive both encoders to state zero at the same time.

The above interleaver types can be combined in a single interleaver. An example of a “simile odd–even” block helical interleaver that can be used with the four state RSC encoder given in Figure 4.1 is shown in Table 4.11.

Table 4.11: Simile odd–even block helical interleaver.

x_1	x_2	x_3	x_4	x_5	x_6
x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}
x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}

Part of the interleaved sequence is shown in Table 4.12.

Table 4.12: The output of a simile odd–even block helical interleaver.

	x_{25}	x_{20}	x_{15}	x_{10}	x_5	x_{30}	x_{19}	x_{14}	x_9	x_4	x_{29}	x_{24}	x_{13}	x_8	x_3	x_{28}	x_{23}	x_{18}	x_7	...
$j \bmod 3$	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	...

4.5 “Frame” interleaver

If the simile interleaver uses only one tail to drive both encoders to the same state, the “frame” interleaver does not use any tail at all, but puts more constraints on the interleaver design [43]. Each RSC encoder can be characterised by a generator polynomial of period L [43]. In this case the N information bits to be interleaved are stored twice in a memory of size $2N$, at addresses such that their subsequent reading for encoding is time–separated by a number of periods which is a multiple of L . In this way, if the encoder started in state zero, it will end in state zero without the need of any tail bits.

4.6 Pseudo–random interleavers

These pseudo–random interleavers are defined by a pseudo–random number generator or a look–up table where all integers from 1 to N (the block size to be interleaved) can be generated. This approach can lead to good or bad interleavers, especially for small interleaver sizes. The only criterion for choosing between them is based on computer simulations; there seem to be no analytical criteria.

4.7 “S-type” interleavers

It was shown in [44] that weight-2 data sequences are an important factor in the design of the component codes. The weight of a data sequence, which is made of 0s and 1s, is the number of 1s in that sequence. If we randomly select an interleaver of size N , the probability that a particular weight-2 data sequence will be permuted by the interleaver into another sequence of the same form is roughly $2/N$ for large N . This probability is larger for smaller N .

Therefore, in order to avoid these identical permutations, an “S-random” permutation was defined as follows: each randomly selected integer is compared to S previously selected integers. The current integer is accepted if and only if it is at a distance larger than S from any of the S previous selections. The process is repeated for all N addresses. For a given N , the maximum S factor that should be used is less than $\sqrt{\frac{N}{2}}$. The interleaver gain is usually larger for larger S values.

4.8 “Uniform” interleavers

Performance bounds for turbo codes were given in [45] using a conceptual uniform interleaver which is the average of all possible interleavers. Let us consider a sequence made of w ones and $k-w$ zeros. A uniform interleaver of length k is a probabilistic device which maps this sequence to all distinct $\binom{k}{w}$ permutations with equal probability $\frac{1}{\binom{k}{w}}$. This technique allows

the analysis of a turbo code as if it were made of two independent elementary codes, due to the uniform distribution produced by the interleaver.

This method gives upper bounds on the error probability which are quite accurate at high values of SNRs. However, the bounds are significantly worse from the performance which can be achieved by turbo codes at very low SNR values. Despite this, the concept of uniform interleaving is a key to understanding turbo code performance.

4.9 The best interleaver

Interleavers are designed for specific system requirements. There is therefore no universal formula that can be used. For example, we have found that for short block sizes and low E_b/N_0 , an odd-even interleaver outperforms a pseudo-random interleaver [41] and vice-versa at higher

E_b/N_0 . For larger block sizes, an S-type interleaver outperforms a pseudo-random interleaver [37].

In Figure 4.2 the extrinsic information E_k (see Section 2, Part II) associated with the information bit d_k at time k is produced by the MAP algorithm using a forward and backward recursion applied to the whole received sequence.

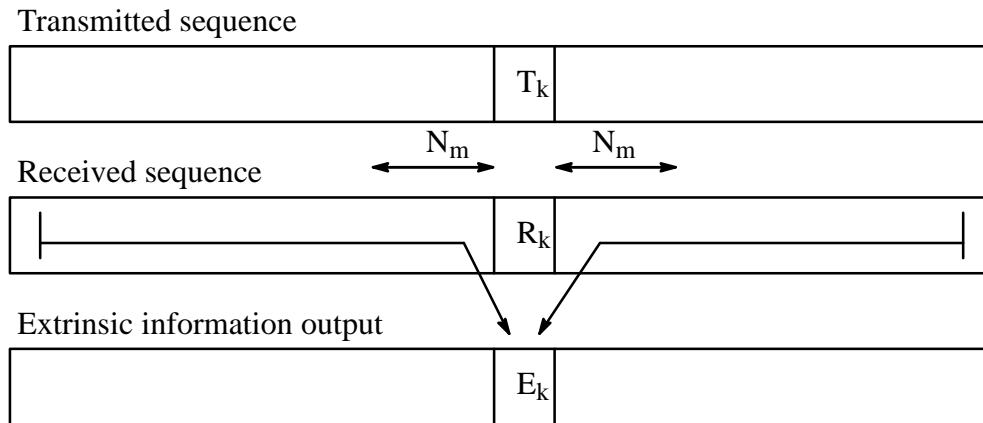


Figure 4.2: Extrinsic information for turbo codes.

Based on our computer simulations, 95% of the value of the extrinsic information E_k is determined by the previous N_m and the following N_m received symbols, where N_m is ten times the constraint length of the code. The larger the interleaver size, the more “uncorrelated” the $2N_m$ symbols of the straight sequence of data are, compared with the $2N_m$ symbols of the interleaved data. This process allows for two “independent” criteria to estimate the soft value of the same bit and then the estimates are passed as extra information from one decoder to another.

Once the above condition is fulfilled, to obtain a further improvement in the performance of turbo codes, a second limiting factor has to be considered: the degree of “randomness” with which the interleaver and the deinterleaver spread the bursts of errors from one decoder output to the next decoder input. From this point of view the ideal interleaver is a random interleaver.

In [47] we showed that performance improvements in the error floor problem could be made with interleaver design. This result is shown in Figure 4.3 for a block size of 8192 bits with rate 1/3 coding, and nine decoder iterations. This figure indicates the potential improvement of a better interleaver. Larger block sizes could possibly be used to provide even better performance results.

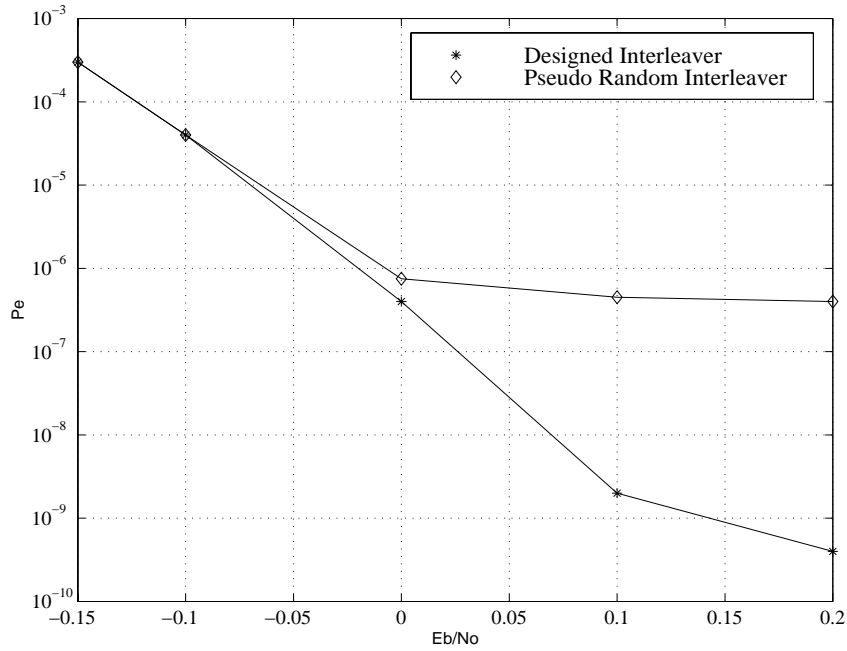


Figure 4.3: Performance of Designed and Random Interleaver for a Turbo Code.

In [48], a method to optimise the interleaver structure using the “Hungarian method” (linear sum assignment problem) [49] is presented. The goal is to break all the weight-2 sequences such that at least one of the outputs avoids a terminating zero-phase sequence.

In [50] a canonical form of the interleaver engine with minimal delay is defined as a finite state permuter (FSP). Two algorithms are developed for a systematic iterative construction of interleavers with a complexity that is polynomial with the interleaver size. Each transposition vector has associated with it a cost function, the algorithms aiming at the minimisation of this cost function. The complexity of the algorithm depends on the length of the error patterns which are taken into consideration.

Recent work has shown that the tails can be dispensed with, without any significant loss in performance for higher interleaver sizes.

From all the above examples of interleaver design it is clear that the role of the interleaver is to allow the decoders to make uncorrelated estimates of the soft values of the same information bit. The less “correlated” the two estimates are, the better the convergence of the iterative decoding algorithm. The role of the interleaver in the iterative process was identified by one of the authors in [51] as similar to the role of the random generator in the “chaos game” in ensuring the convergence to its attractor, the Sierpinski gasket [52].

5 References

1. F. MacWilliams and N. Sloane, *The theory of error-correcting codes*, North Holland Publishing Company, 1981.
2. P. Elias, "Error-free coding," *IRE Transactions on Information Theory*, PGIT-4, pp. 29–37, Sep. 1954.
3. G. D. Forney Jr., *Concatenated Codes*, MIT Press, Cambridge, Mass., 1966.
4. H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Transactions on Information Theory*, vol. IT-23, pp. 371–377, Sep. 1977.
5. G. J. Pottie and D. P. Taylor, "Multilevel codes based on partitioning," *IEEE Transactions on Information Theory*, vol. IT-35, pp. 87–98, Sep. 1977.
6. U. Wachsmann and J. Huber, "Power and bandwidth efficient digital communications using turbo codes in multilevel codes," *European Transactions on Telecommunications*, vol. 6, no. 5, pp. 513–526, Oct 1995.
7. C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," *ICC 1993*, Geneva, Switzerland, pp. 1064–1070, May 1993.
8. R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
9. M. Lentmaier, "Soft iterative decoding of generalised low-density parity-check codes based on MAP decoding of component Hamming codes," *Diploma Thesis*, University of Ulm, Sweden, Nov. 1997.
10. L. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimising symbol error rate," *IEEE Transactions on Information Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
11. S. S. Pietrobon and S. A. Barbulescu, "A simplification of the modified Bahl decoding algorithm for systematic convolutional codes," *ISITA 1994*, Sydney, NSW, pp. 1073–1077, Nov. 1994.

12. A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, Apr. 1967.
13. J. Hagenauer, "Source-controlled channel decoding," *IEEE Transactions on Communications*, vol. 43, pp. 2449–2457, Sep. 1991.
14. S. A. Barbulescu, "Iterative decoding of turbo codes and other concatenated codes," *Ph.D. dissertation*, Feb. 1996.
15. J. Hagenauer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Communications*, vol. 42, pp. 429–435, Mar. 1996.
16. P.J. Lee, "New short constraint length, rate 1/N convolutional codes which minimize the required SNR for given desired bit error rates," *IEEE Transactions on Communications*, vol COM-33, pp. 171–177, Feb. 1985.
17. S. Dolinar, D. Divsalar and F. Pollara, "Code performance as a function of block size," TDA Progress Report 42-133, May 1998.
18. C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, pp.379–423 (pt. I); 623–656 (pt. II), 1948.
19. J. Y. Couleaud, "High gain coding schemes for space communications," ENSICA Final Year Report, University of South Australia, Sep. 1995.
20. S. Benedetto, G. Montorsi, "Iterative decoding of serially concatenated convolutional codes," *IEE Electron. Lett.*, vol.32, pp. 1186–1188, June 1996.
21. S. Benedetto, G. Montorsi, D. Divsalar and F. Pollara, "Serial concatenation of interleaved codes: Performance Analysis, Design, and Iterative Decoding", *JPL TDA Progress Report 42-126*, August 15, 1996.
22. P. Gray "Serially concatenated trellis coded modulation", *PhD Dissertation*, Univ. of South Australia, submitted, Aug. 1998.
23. D. Divsalar and F. Pollara, "Hybrid concatenated codes and iterative decoding", *JPL TDA Progress Report 42-130*, Aug. 1997.
24. D. Divsalar and F. Pollara, "Serial and hybrid concatenated codes with applications", in *Proc. Int. Symp. on Turbo Codes and Related Topics*, Brest, France, pp. 80–87, Sep. 1997.

25. University of South Australia, "Reduced bandwidth study of the High Speed Data Service – Final Report", SCRC96–6532–D005, Sep. 1996.
26. University of South Australia, "Turbo–X proof–of–concept modem – Final Report" SCRC97/65306/INM/080, Feb 1998.
27. Comatlas, "CAS 5093 – 40 Mbit/s turbo code codec," rev 4.1, May 1995.
28. S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes", *IEEE Trans. on Inform. Theory*, vol. 42, No 2, pp. 409–428, Mar. 1996.
29. S. S. Pietrobon, "Implementation and performance of a turbo/MAP decoder," *Int. Journal Satellite Communications*, vol 16, pp.23–46, 1998.
30. J. Lodge, R. Young, P. Hoeher and J. Hagenauer, "Separable MAP 'Filters' for the decoding of product and concatenated codes", *IEEE ICC'93*, pp. 1740–1745, May 1993.
31. R. Pyndiah, A. Glavieux, A. Picart and S. Jacq, "Near optimum decoding of product codes", *IEEE GLOBECOM'94*, vol. 1/3, pp. 339–343, Nov. 1994.
32. J Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes", *IEEE Transactions on Information Theory*, vol. 42, pp.429–445, March 1996.
33. R. Pyndiah, "Iterative Decoding of Product Codes" in *Proc. Int Symp. on Turbo Codes and Related Topics*, (Brest, France), pp. 71–79, Sept. 1997.
34. R. Pyndiah, A. Picart, and A. Glavieux "Performance of block turbo coded 16–QAM and 64–QAM modulations," *Proceedings of GLOBECOM'95*, pp. 1039–1043.
35. M. Rice, "Soft–decision Reed–Solomon decoder," *Workshop on Coding for Radio Channels*, Adelaide, Australia, November 1993.
36. P. Robertson and T. Woerz, "A novel bandwidth efficient coding scheme employing turbo codes," in *Proc. ICC '96*, pp. 962–967, June 1996.
37. S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Parallel concatenated trellis coded modulation," in *Proc. ICC'96*, pp. 974–978, June 1996.
38. D. Divsalar and F. Pollara, "Multiple turbo codes for deep–space communications", *JPL TDA Progress Report 42–121*, JPL, Pasadena, Ca. USA, pp. 66–76, May 1995.

39. S. A. Barbulescu, W. Farrell, P. Gray, and M. Rice, "Bandwidth efficient turbo coding for high speed mobile satellite communications," in *Proc. Int. Symp. on Turbo Codes and Related Topics*, (Brest, France), pp 119–126, Sep. 1997.
40. P. Robertson and T. Woerz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes", *IEEE Journal on Selected Areas in Communications*, vol.16, no. 2, pp. 206–218, Feb. 1998.
41. S. A. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo codes," *Electron. Lett.*, vol. 30, No. 25, pp. 2107–2108, Dec. 1994.
42. S. A. Barbulescu and S. S. Pietrobon, "Terminating the trellis of turbo codes in the same state," *Electron. Lett.*, vol. 31, No. 1, pp. 22–23, Jan. 1995.
43. Berrou, C., Evano, S. and Battail, G., "Turbo block codes," *Proceedings of the Seminar on Turbo Coding*, Lund, Sweden, pp. 1–7, Aug. 1996.
44. D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications", *JPL TDA Progress Report 42-121*, JPL, Pasadena, Ca. USA, pp. 66–76, May 1995.
45. Benedetto, S. and Montorsi, G., "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol 42, no 2, pp. 409–428, Mar. 1996.
46. S. A. Barbulescu and S. Pietrobon, "Interleaver design for turbo codes", *IEE Electronic Letters*, vol. 30, pp. 2107–2108, 8 Dec. 1994.
47. J. Anderson and V. Zyablov, "Interleaver design for turbo coding," in *Proc. Int. Symp. on Turbo Codes and Related Topics*, (Brest, France), pp 227–230, Sep. 1997.
48. A. K. Khandani, "Design of the turbo-code interleaver using Hungarian method" submitted to *Electronics Letters*, 1997.
49. Bela Kreko, "Linear programming" translated by J. Ahrens and C. Safe, Sir Isaac Pitman & Sons Ltd., 1968.
50. F. Daneshgaran and M Mondin, "Iterative interleaver growth algorithms of polynomial complexity for turbo codes", submitted to *IEEE Int. Sym. on Information Theory*, US, Aug 1998.

51. S. A. Barbulescu, "Dynamical system perspective on turbo codes," *Electron. Lett.*, vol. 34, No. 8, pp. 754–755, Apr. 1998.
52. M. F. Barnsley and H. Rising, *Fractals everywhere*, 2nd Ed. Boston:Academic Press, 1993.