2006

CONVERGENCE IMPROVEMENT OF ITERATIVE DECODERS

PAPAGIANNIS, EVANGELOS

CONVERGENCE IMPROVEMENT OF ITERATIVE DECODERS

E. Papagiannis

Ph.D. 19th of April, 2006

CONVERGENCE IMPROVEMENT OF ITERATIVE DECODERS

by

EVANGELOS PAPAGIANNIS

A thesis submitted to the University of Plymouth
in partial fulfillment for the degree of

DOCTOR OF PHILOSOPHY

Department of Computing, Communications and Electronics
Faculty of Technology
19th of April, 2006

Convergence Improvement of Iterative Decoders
by Evangelos Papagiannis

Abstract

Iterative decoding techniques shaked the waters of the error correction and communications field in general. Their amazing compromise between complexity and performance offered much more freedom in code design and made highly complex codes, that were being considered undecodable until recently, part of almost any communication system. Nevertheless, iterative decoding is a sub-optimum decoding method and as such, it has attracted huge research interest. But the iterative decoder still hides many of its secrets, as it has not been possible yet to fully describe its behaviour and its cost function.

This work presents the convergence problem of iterative decoding from various angles and explores methods for reducing any sub-optimalities on its operation. The decoding algorithms for both LDPC and turbo codes were investigated and aspects that contribute to convergence problems were identified. A new algorithm was proposed, capable of providing considerable coding gain in any iterative scheme. Moreover, it was shown that for some codes the proposed algorithm is sufficient to eliminate any sub-optimality and perform maximum likelihood decoding. Its performance and efficiency was compared to that of other convergence improvement schemes.

Various conditions that can be considered critical to the outcome of the iterative decoder were also investigated and the decoding algorithm of LDPC codes was followed analytically to verify the experimental results.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my sincere gratitude to everyone who has helped and supported me in my research and funding. In particular;
My supervisors Professor Martin Tomlinson, Dr. M.Z. Ahmed and Dr. A. Ambroze for their constant support and guidance at all stages of my research. Also for their trust and concern to help me enjoy my research.
Zaki and Adrian for the long Friday night movie sessions (Lord of the Rings is excluded), dinners and beautiful tours we had together. But most of all for their huge support at any sort of difficulties that I had, and their friendship.
My colleagues (Andrew, Will, CJ, Jing, Purav, Keiko and Andrea) for their pleasant company during the long hours spent in the office and for the nice parties of course.
CJ for his immediate support and help at the worse night of my life.
All the close friends I've made during my stay in UK.
My close friends back in Greece for their understanding and support to my effort.
Finally, and most importantly, my beloved family for all the sacrifices they made all these years for me. I don't think that it is possible express my deep and sincere gratitude for them with words.

Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.
The fees for this study were financed partly by EPSRC and by the school of Computing, Communications and Electronics.
A programme of advanced study was undertaken, which included the extensive reading of literature relevant to the research project and attendance of international conferences on coding and communications.
The author has presented papers at:

- The 2003 ESA International workshop on Signal Processing for Space Communications, Catania, Italy
- The 2004 International Zurich Seminar on Communications,Zurich, Switzerland
- The 2005 4th IASTED International Conference on Communication Systems, Benidorm, Spain

A paper has also been published in the 8th International Symposium on Communication Theory Applications (ISCTA '05), Ambleside, Lake District, UK.
In addition the author has participated in two patent applications: 0320126.6, July 2003 and 0409306.8, April 2004

Evangelos Papagiannis : E Papagiannis

Date : 19/04/06

word count : 45,100

In memory of my father, Ioannis.

1 Introduction

1.1 The problem of reliable communication, the role of redundancy and the idea of channel capacity

The essence of communication theory deals, among others, with the reliable transmission of data from one point to another. In between the sending point (transmitter) and the receiving end (receiver) exists the channel, through which the data propagates. The communication problem is a result of the random disturbance that is introduced to the propagating data by the channel. The noise, as this disturbance is known, affects the ability of the receiver to identify reliably the sent data and target of the communication theory is to offer the techniques that solve the communication problem in the optimal way.

Until the work of Shannon in 1948 [59], it was widely accepted that reliable communication at low error rates with the presence of noise, was unattainable. Earlier, in 1924 Nyquist [40] had concluded that the maximum number of pulses that can be transmitted over a channel of bandwidth B and successfully be resolved at the receiving end is $2B$ pulses per second. Hartley, in 1928 [32], combined Nyquist's result with the concept of accuracy in signal reception to obtain the maximum amount of data that can be reliably communicated over a channel. Due to Hartley M messages can be reliably transmitted by a pulse with amplitude confined in the voltage range $[-A, +A]$ and a receiver that can estimate the amplitude with accuracy $\pm\Delta$, i.e. with separation between different messages of 2Δ volts. Hence, the maximum amount of data that could be transmitted reliably over a channel would be equal to

$$M = \left(1 + \frac{A}{\Delta}\right)^{2B} \tag{1.1}$$

The presence of noise would cause fluctuations to the amplitude of the received pulse that would lead the receiver to error decisions. Thus, the maximum amount of data that could be reliably transmitted by the channel for certain amount of noise would be dictated by the minimum accuracy Δ that is necessary in order to keep the different transmitted

messages distinguishable by the receiver. In other words, this model suggested that the only way to transmit the M messages and still guarantee with high probability that they will all be distinguishable at the receiving end (therefore ensuring reliable communication) would be either to increase the amplitude of the transmitted pulse (i.e. the transmitted power) or the bandwidth. However, Hartley's model is very simplistic and does not describe properly the interrelation between the various parameters.

1.1.1 The effect of redundancy

In 1948, Shannon [59] proved that reliable communication over a noisy channel of restricted bandwidth is feasible, and incorporated the concepts of noise, bandwidth and signal magnitude into a new parameter called *channel capacity* (C). Shannon showed that as long as the information rate R is kept lower than or equal to the capacity C, the RT information bits can be delivered reliably to the receiving end with probability of error that reduces as the transmission period T increases. The addition of redundant bits that makes the information rate R less than unity is the main idea behind error control. Consider the case of an uncoded system i.e. a system where no redundancy is added. Assume that $R = 1$ and $k = RT$ information bits are transmitted, each represented by a wave-shaped pulse $x(t)$ of duration $\tau = 1/R$ and energy $E_b = P_b\tau$ where P_b is the power contained on each pulse. The transmitted signal $s(t)$ will then be

$$s(t) = \sum_{i=1}^{k} s_i x(t - i\tau) \tag{1.2}$$

where $s_i = +1$ when the i^{th} bit is 1 and $s_i = -1$ when the i^{th} bit is -1. For an AWGN channel, the probability p that any of the received bits is in error is

$$p = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) = Q\left(\sqrt{\frac{2P_b}{RN_o}}\right) \tag{1.3}$$

So the probability P_e that at least one error occurs among the k bits is

$$P_e = 1 - (1-p)^k = 1 - (1-p)^{RT} \tag{1.4}$$

From the above it can be deduced that by increasing T, thus sending more information bits k, the probability of error increases too. Hence, the only two choices for reducing P_e would be either to increase the transmitted power or to decrease the tranmission period

T. Figure 1.1 offers a geometric interpretation of this result. As k increases the number of neighbours grows with the same rate and consequently the probability of error gets higher as more opponents appear in the same distance from the transmitted signal.

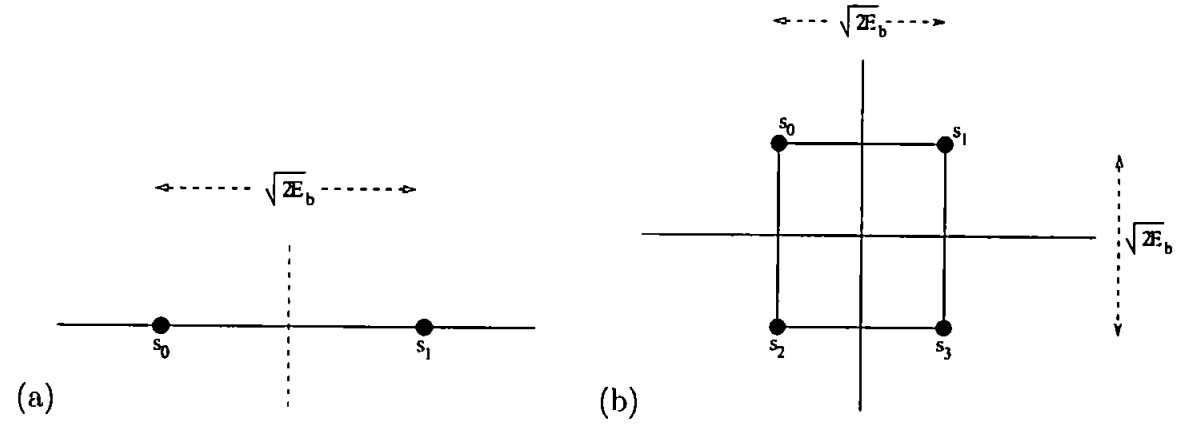


Figure 1.1: Signal constellations for (a) k=1 and (b) k=2

In conjunction to the previous example let us assume that the code rate $r = \frac{R}{n}$ is reduced ($r < 1$), so that $n > R$ bits are transmitted to convey each information bit. Therefore, $T(n - R) = nT(1 - r)$ extra bits are sent over the channel. From the available signal combinations at the receiving end, only 2^{RT} of them represent information signals; the rest are redundant. The effects of redundancy are two. First, the energy per information bit E_b is reduced by a fraction r since more bits are transmitted with the same amount of power. Secondly, since the dimension $nT = N$ of the transmitted signal is larger than the dimension of the information sequence $RT = k$, the null extra dimensional space can be used to increase the separation (distance) among the information signals and therefore reduce the probability of error. This is the main idea behind error control. A *code* sets the way that the additional dimensional space is managed so that any pair of valid signals (*codewords*) are separated by a distance of at least d_{min} (*minimum distance*). Note that in contrast to the uncoded case, incrementing T offers the ground for better code design with higher distances among the codewords, and therefore lower error probabilities.
A vector space of N dimensions can be visualised as a N-dimensional hypercube on the vertices of which the 2^k codewords are arranged. As figure 1.2 depicts, the potential advantage that is offered by the redundant bits disappears if the codewords are not mapped properly on the 2^N available vertices to ensure high minimum distance separation. Shannon calculated the bounds of error probability based on the average error probability over all 2^{Nk} distinct ways of assigning the 2^k codewords on the 2^N available vertices (i.e. over all codes). The fact that the average error probability was bounded over very low

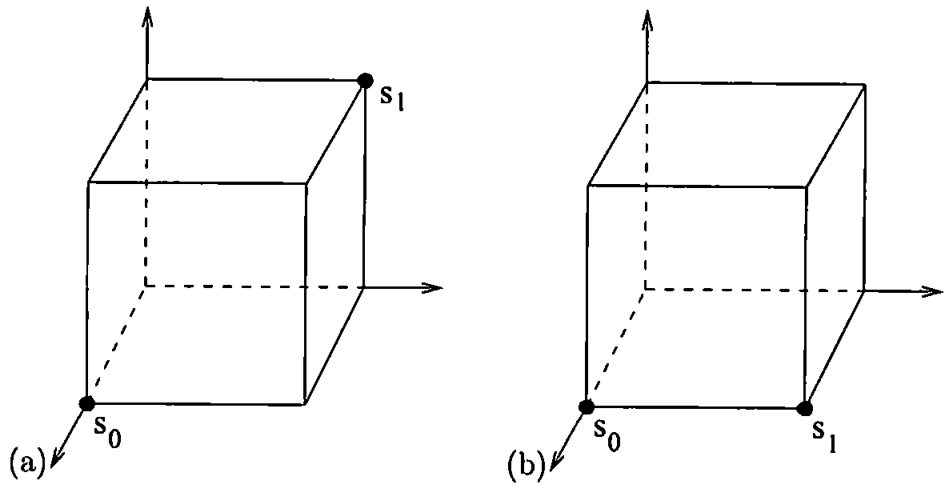


Figure 1.2: Signal constellations with k=1 and N=3 for (a) a good code (b) a bad code

values proved the existence of codes that can offer such performance.

1.1.2 Shannon's capacity theorem

A necessary condition for transmission with low probability of error is that the information rate is kept lower or equal to the channel capacity $R \leq C$. This section offers a brief derivation of channel capacity and a proof for the validity of the previous statement, based on the *sphere packing bound* [66] concept.

Assuming an *Added White Gaussian Noise* (AWGN) channel, the received noisy replicas of the transmitted signals $\mathbf{s_i}$ of energy $|\mathbf{s_i}^2| = NE_s$ is simply the sum of $\mathbf{s_i}$ with the noise random variable $\mathbf{n}$.

$$\mathbf{r_i} = \mathbf{s_i} + \mathbf{n}$$

The receiver's task is to associate each received vector $\mathbf{r_i}$ with the correct (the transmitted) signal $\mathbf{s_i}$. The receiver will decide $\mathbf{r_i}$ as $\mathbf{s_i}$ only if $\mathbf{r_i}$ falls within the decision bound of $\mathbf{s_i}$. The decision bounds can be visualised as N-dimensional spheres of radius $\rho = \mathbf{s_i} = \sqrt{NE_s}$ centred at $\mathbf{s_i}$. An error decision will occur if $\mathbf{r_i}$ lies within the volume of the sphere of $\mathbf{s_j}$ (meaning that the Euclidean distance $Euc(\mathbf{r_i}, \mathbf{s_j}) < Euc(\mathbf{r_i}, \mathbf{s_i})$). Since the size of the sphere is dependant on the code-length N, it is convenient to consider normalised vectors.

$$\hat{\mathbf{s}}_\mathbf{i} = \mathbf{s_i}/\sqrt{N}$$
$$\hat{\mathbf{n}} = \mathbf{n}/\sqrt{N}$$
$$\hat{\mathbf{r}}_\mathbf{i} = \mathbf{r_i}/\sqrt{N}$$

By using the *sphere hardening* theorem it is shown in [60, 76] that although the mean squared value of $\hat{\mathbf{n}}$ remains constant and equal to the single sided noise density $\frac{N_o}{2}$ for any value of N, its variance $\sigma^2(|\hat{\mathbf{n}}|^2)$ is inversely proportional to N.

$$\sigma^2(|\hat{\mathbf{n}}|^2) = \frac{2}{N}\left(\frac{N_o}{2}\right)^2 \tag{1.5}$$

For large enough N, the variance of $\hat{\mathbf{n}}$ gets almost 0 and therefore, it can be assumed with probability that approaches 1 that the magnitude of the noise vector $\hat{\mathbf{n}}$ will always be

$$|\hat{\mathbf{n}}| = \sqrt{N_o/2} \pm \delta \tag{1.6}$$

where δ is a number arbitrarily close to zero. Equation 1.6 implies that the N-dimensional spheres around signals $\mathbf{s_i}$ that define the decision bounds, should have a radius ρ of at least $\sqrt{N_o/2} \pm \delta$ to guarantee with high probability that the receiver will associate the received vector with the transmitted signal. In any other case, reliable decoding is impossible and the probability for erroneous decisions is very close to 1. Figure 1.3 offers a *2-D* visualisation of the predescribed situation.

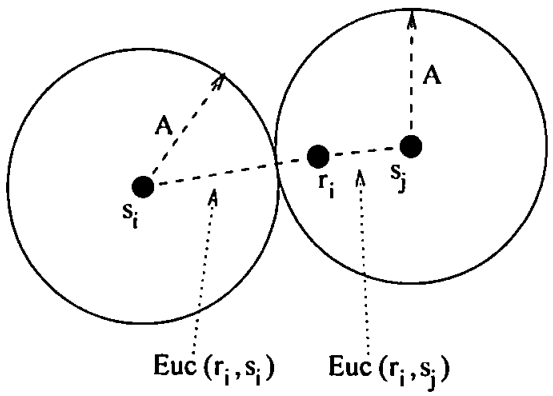


Figure 1.3: The received vector $\mathbf{r_i}$ is decoded in error since the radius A of the spheres is less than $\sqrt{N_o/2} \pm \delta$, and therefore $Euc(\mathbf{r_i}, \mathbf{s_j}) < Euc(\mathbf{r_i}, \mathbf{s_i})$

It is essential then that the radius of the spheres around each of the k signals $\mathbf{s_i}$ is comparable or larger than $\sqrt{N_o/2} \pm \delta$. The question is if the available power and bandwidth are large enough to accommodate all those k spheres. In other words if the channel has the capacity to contain all these k spheres. This is the argument that is used for obtaining the capacity bounds of a channel for a predefined information rate, block-length, power and bandwidth.

In [76] it is proved that the volume V and radius ρ of an N-dimensional sphere are related

by

$$V = B_N \rho^N \tag{1.7}$$

where B_N is a positive constant that depends only on the block-length N. In our case, each of the k spheres around signals $\mathbf{s_i}$ should have a minimum volume of

$$V_i = B_N \left(\frac{N_o}{2}\right)^{\frac{N}{2}} \tag{1.8}$$

where δ which is arbitrarily close to zero has been omitted. Considering the energy for each bit of $\mathbf{s_i}$ as E_b, the volume of the overall N-dimensional sphere with radius $E_b + \frac{N_o}{2}$ that will contain the k smaller spheres will be

$$V = B_N \left(E_b + \frac{N_o}{2}\right)^{\frac{N}{2}} \tag{1.9}$$

From 1.8 and 1.9 we obtain the inequality that if solved for the information rate R, will give us the channel capacity C.

$$\frac{V}{k} > V_i \tag{1.10}$$

which becomes

$$\frac{B_N \left(E_b + \frac{N_o}{2}\right)^{\frac{N}{2}}}{k} > B_N \left(\frac{N_o}{2}\right)^{\frac{N}{2}} \tag{1.11}$$

and solving for k

$$k < \left(\frac{E_b + \frac{N_o}{2}}{\frac{N_o}{2}}\right)^{\frac{N}{2}} \tag{1.12}$$

Since $k = 2^{RN}$

$$R < \frac{1}{2} \log_2 \left(\frac{E_b + \frac{N_o}{2}}{\frac{N_o}{2}}\right) \tag{1.13}$$

and equivalently

$$R < \frac{1}{2} \log_2 \left(1 + 2\frac{E_b}{N_o}\right) \tag{1.14}$$

Equation 1.14 sets the maximum value of the information rate R for reliable transmission, which is known as the capacity C. Due to Nyquist [40], for a channel of bandwidth B the maximum number of bits per second that can be transmitted and be resolved by the receiver is $2B$. Hence, the information rate is upper bounded by the bandwidth B.

$$R \text{ (bits per second)} \leq 2B \tag{1.15}$$

As a consequence, the duration per information bit is $\tau = \frac{1}{R} = \frac{1}{2B}$, the energy per bit can be expresed as $E_b = P\tau = \frac{P}{2B}$. For reliable communication, the information rate should be lower than the maximum amount of information that the channel can accomodate i.e. $R < 2B \cdot C$. Equation 1.14 comes then to the well known form

$$C = B\log_2\left(1 + \frac{P}{BN_o}\right) = B\log_2\left(1 + SNR\right) \tag{1.16}$$

The theoretical limit in terms of E_b/N_o when transmitting at the maximum rate (capacity rate) through a channel of infinite bandwidth is obtained below. For R_I representing the information rate in bits per second we have

$$C = B\log_2\left(1 + 2\frac{R_I E_b}{BN_o}\right) \tag{1.17}$$

Using the fact that $\log_2(x) = \frac{\ln(x)}{\ln(2)}$, the channel capacity equation becomes

$$C = 1.44B\ln\left(1 + \frac{R_I E_b}{BN_o}\right) \tag{1.18}$$

and as bandwidth approaches to infinity

$$C \approx 1.44\frac{R_I E_b}{N_o} \tag{1.19}$$

For transmission at the channel capacity $R_I = C$

$$\frac{E_b}{N_o} = \frac{1}{1.44} = -1.6\ dB \tag{1.20}$$

Of course in real life infinite bandwidth channels do not exist so the cut-off limit of -1.6dB is not useful for practical applications. For a given block-length and code rate it is possible though to obtain the sphere packing bound [66] that defines how many disjoint N dimensional spheres can be located at the available N-Euclidean space so that the fraction of the total space that is covered by the spheres is maximised. The sphere packing bound offers a very useful tool that can be used as a figure of merit for the performance of any designed code.

1.2 Error correction coding over the years

The work of Shannon in 1948 changed the widely accepted view at the time, that reliable communication can only be achieved by increasing either the signal power or the bandwidth. Shannon showed that with the addition of redundancy the probability of error can be made arbitrarily low for large lengths. Since the way that the redundancy should be incorporated within the information message was not specified, the coding communities were challenged to find the codes and decoding techniques that would show Shannon's promised results in practise. It was apparent that any proposed codes should have a structured form so that the complexity of the encoder and decoder is affordable.

The first efficient schemes were devised by Hamming [31] (1950) and Golay [30] (1949). Golay's (23,12) code ($k = 12$,$N = 23$) was the first example of a *perfect code*, i.e. a code where all error patterns up to its error correcting capabilities can be exactly represented by the $(N - k)$ parity bits. Although far from Shannon's promised results, these schemes found many applications and are still used even today.

The need for better codes had driven many researchers to approach code design from different angles. A new class of codes called *cyclic* were first introduced in 1957 and 1958 by Prange [49]. Shortly after, in 1959, we have the well-known code with *algebraic* structure, the Bose-Chaudhuri-Hocquenghem (BCH) [10]. The first error correction procedure for binary BCH codes was formulated by Peterson in 1960. The non binary error correction procedures are attributed to D.Gorenstein and N.Zierler in 1961 in their paper "A class of Error-Correcting Codes in p^m symbols". More efficient decoding methods were developed later by E.Berlekamp in 1966 [8] and J.Massey in 1969 [37]. BCH codes are used in the Global System for Mobile communications (GSM) and in the Hybrid Automatic-Repeat-Request (ARQ) systems (Costello et al. 1998 [13]). In 1990 the BCH(511,439) was used for protecting digital video transmissions under the ITU-T H.261 standard (Costello et al. 1998).

The Reed-Solomon [51] codes which are named,after their inventors, were discovered in 1960 and they are another example of algebraic codes. The RS codes are the subclass of BCH codes and they are maximum-distance-separable (MDS) codes in the sense that, for a given length and field, they have the optimum error correction capability. The powerful burst error correcting capabilities of RS codes have monopolised the error correction in the fields of magnetic/optical data storage. The Cross-Interleaved RS codes (CIRS), which are a combination of two shortened 2-error correcting RS codes, are used in the Compact Disc (CD) (Costello et al. 1998).

Another class of cyclic codes with simple decoding algorithm is the *majority-logic de-*

codable codes. Some examples of majority-logic decodable codes are the difference-set [Weldon] and the finite and projective geometry codes. The precursor of the majority-logic decodable codes is the algorithm given by Reed [50] to decode the codes of Muller [39]. These are the Reed-Muller (RM) codes, that were used in the 1969 and 1971 Mariner space missions, and in the 1976 Viking missions to Mars (Costello et al. 1998).

All the codes referred up to this point, belong to the family of *block codes.* Another important type of codes was invented by Elias in 1954 and are called *convolutional codes.* Unlike the block codes, the structure of convolutional codes allows efficient implementation of *soft decision* algorithms which can provide coding gain of 2-3dB over *hard decision* decoders. Soft decision decoders for convolutional codes were first introduced by Wozencraft in 1957, which were then improved by Fano in 1963 [18]. The inefficiency of these techniques for long constraint length convolutional codes was solved by Viterbi in 1967 with the introduction of the Maximum-Likelihood (ML) decoder (known as Viterbi decoder) [71]. The Viterbi decoder was further developed by Forney in 1973 with the introduction of the *trellis diagrams* [23]. Shortly after, in 1974, Bahl, Cocke, Jelinek and Raviv invented the BCJR decoder [5] (acronym from the authors' names) which is an optimal bit-by-bit *Maximum a Posteriori* decoder (that is why it is also widely known as MAP decoder).

The idea of *concatenation*, i.e. the use of codes in cascade, was first introduced by Forney in 1966 [22] as a way o constructing long codes from shorter ones. Concatenated codes consist of two codes: the *inner* and the *outer* code. Inner codes are usually simpler and with lower bit error capability than the outer code, which is responsible for correcting all residual errors of the inner code. The most common arrangement is to use short constraint length convolutional codes with soft decision Viterbi decoding as the inner codes and non binary RS codes with algebraic hard decision decoder as the outer codes. This arrangement was accepted by the Consultative Committee on Space Data Systems (CCSDS) as the Telemetry Standard in 1987 (Costello et al. 1998). Concatenation schemes in general have been used in many applications such as the space missions of Mariner in 1971 and Voyager in 1977 (Costello et al. 1998).

A significant milestone in the field of coding theory, and communications theory in general, was the idea of combined coding and multilevel modulation (known as *trellis-coded modulation* TCM) by Ungerboeck in 1982 [69]. Before TCM, it was believed that error correction coding is only useful for *power-limited channels* which have plenty of bandwidth available to accommodate the introduced redundancy. For *bandwidth-limited channels*, as are the telephone lines, the bandwidth efficiency is crucial and the "power for bandwidth" property of coding did not seem to be the best policy. The introduction of TCM

improved the modem standards and allowed (because of the power reduction) full-duplex operation. The modem technology owes much of its evolution on TCM.

1.2.1 The idea of iterative decoding: turbo codes and LDPC's

For many years the Shannon's challenge of approaching the capacity limits still seemed far away. Optimum decoding of long powerful codes was computationally prohibited, and the best code scheme known so far had been the concatenation of RS and covolutional codes.

In 1993, Berrou et al. [9] achieved a major breakthrough in the field of error correction coding with the introduction of turbo codes. Turbo codes were shown to achieve very low error rates at SNR a fraction of decibel above the Shannon capacity limit, in a period that the best coding scheme could not perform closer than 3-4dB away from that. The first presented turbo coding scheme had achieved BER of 10^{-5} at $E_b/N_o = 0.7$dB by parallel concatenation of two convolutional codes of constraint length 5, and block length *N=64,000*. In his proposed scheme, the information stream is interleaved prior to entering to the second encoder. The interleaving operation implies that the trellis diagrams of the constituent codes are not sufficient to define the structure of the code. To perform ML decoding we would need a trellis diagram of approximately $2^{s_1+s_2+N_{syst}}$ states, where s_1 and s_2 denote the number of states of the trellis of each constituent code. The turbo decoder however uses the trellis structure of the constituent convolutional codes but still, with performance very close to the optimum. The new concept that had managed to combine relatively low complexity with near to optimum performance, was the decoding of the received messages iteratively. The *iterative decoder* consists of two MAP decoders, one for each constituent code, that exchange information in an iterative form. That is, each decoder takes advantage of the progress made by the other by circulating their probability estimations for the systematic bits. Positive feedback of information is avoided by the use of only the *extrinsic* part of the decoders' posterior decisions at the end of each iteration. The sub-optimality of turbo (and generally iterative) decoding lies on the fact that the algorithm can not guarantee convergence to the ML solution, and in some cases it cannot even guarantee convergence to any solution. In [7] the turbo concept was used in serial concatenated schemes with similar results.

Benedetto et al. in 1995 [6], unveiled the secret behind the excellent performance of turbo codes by investigating their weight spectrum and denoting the very low multiplicity of the low weight codewords. The problem of convergence to the ML solution that accounts for the sub-optimality of turbo decoders has been investigated by many researchers of the

field. In [14] the convergence properties of turbo schemes are assessed by investigating the evolution of the densities of the extrinsic information. The authors use the close resemblance of the extrinsic distribution to a Gaussian, firstly denoted by Wiberg in [74]. In [65] the convergence properties of turbo schemes are visualised into *extrinsic information transfer* (EXIT) charts that are based on mutual information measurements. The modes of convergence are observed in [52] and categorised. The relation between turbo and ML decoding is described by Richardson in [56] where the turbo decoder is formalised as a discrete time non linear dynamic system. In the same paper, the existence of fixed points of the decoding algorithm for a class of turbo codes has been shown, together with a set of sufficient conditions for their uniqueness and stability. Following up Richardson's work, the authors of [2] describe a bifurcation analysis approach of the iterative decoding process as a dynamic system parameterised by SNR. The paper shows that at different SNR, fixed points with different characteristics exist.

Shortly after the appearance of turbo codes and the idea of iterative decoding, MacKay and Neal in 1996 [38] rediscovered the low-density parity-check (LDPC) codes. LDPC codes had been originally invented by Gallager back in 1962 [28] but they were abandoned for many years due to their high computational complexities at that time. LDPC codes are linear codes with sparse parity check matrix. The decoding can be performed by the *sum-product* (SP) algorithm which is an iterative message passing type of algorithm. Each parity check equation is processed individually and produces extrinsic information which is passed on to the next iteration. Like turbo codes, LDPC's operate very close to the channel capacity limit and it has been shown that for huge block-lengths it is possible to approach within 0.0045dB [12] of the Shannon limit.

Belief propagation (and Sum-Product algorithm as a form of belief propagation) was developed to operate on single connected graphs, i.e. graphs with no cycles. The presence of cycles on LDPC codes implies that belief propagation works sub-optimally. However, if cycles are kept sufficiently long, belief propagation performs exceptionally well. Short cycles should in general be avoided when constructing LDPC codes. Due to [53] the presence of short cycles within the structure of the code and their contribution to small stopping sets are the major sources for the convergence problems of belief propagation on LDPC's. However, as it has been shown in [67], not all short cycles are equally harmful. In [55] the authors have linked the evolution of the extrinsic densities to the convergence capabilities of the code (density evolution for turbo decoders was inspired by this paper). Wiberg in his Ph.D. thesis proves that the SP algorithm works optimally on graphs that do not contain any loops (*tree graphs*) and that SP decoding in a graph with loops is equivalent to optimal decoding on the associated *computation graph*. Soljanin and Offer

in [61, 41] denote that the SP algorithm works optimally for tree codes and nearly optimally for very sparse codes with loops when the group algebra product is applied. In [27] the authors present a signal space characterisation of iterative decoders and link the presence of loops with the emergence of *pseudo-codewords*.

Various schemes have been presented in the literature, aiming to improve the performance of iterative decoding. Among these, the *ordered statistics decoder* (OSD) [25, 26] (which has been applied on turbo codes and LDPC's), the information correction method of [70, 48] and the Chase algorithm [11, 24]. Various schemes have been proposed that investigate the concatenation of turbo codes with BCH codes (in an attempt to correct residual errors that escape from the turbo decoder) [63] and CRC codes [57]. Additionaly, alternative approaches to the iterative decoding problem originating from the computing science such as the *Linear Programming* (LP) methods have been also presented [19, 20]. Despite of the fact that nowadays iteratively decoding schemes are widely used in many applications, their exact behaviour and cost function is not really known. Many features of iterative decoding still remain unexplained and suffer from a lack of theoretical foundation. Questions concerning the conditions under which the iterative decoding algorithm accepts a single fixed point solution, and the relation of fixed points with the ML codeword still wait for an answer.

1.3 Thesis Outline

Chapter 2 serves as an introduction into turbo codes and LDPC's. The chapter discusses aspects such as the structure of these codes, the encoding and the decoding mechanisms. Particular interest is given to the iterative decoding algorithms.

Chapter 3 is dealing with the convergence problem of iterative decoders. Based on a turbo scheme and by using the correlation coefficient as a measure, it is shown with practical examples how the randomness assumption gradually fades with increasing iterations.

Chapter 4 presents the two basic tools for the evaluation of the convergence properties of iterative decoders. Both methods exploit the simplicity in the representation of the extrinsic information, that originates from the Gaussian-like distribution of the latter.

In chapter 5 three methods are proposed for the convergence improvement of iterative decoders. In the first method convergence improvement is attempted by variation of the variance, as a means of affecting the sensitivity of the iterative decoder to certain coordinates of the channel output. In the second method a bounding function is used to limit the maximum growth rate (per iteration) of the extrinsic probabilities. Finally, the

last method attempts to improve the convergence performance of iterative decoders by re-processing all non-convergent blocks, after first applying a structured permutation to their received vector.

In chapter 6 the SP decoding process is followed analytically and it is demonstrated how and why the iterative decoder diverges from the ML decision. In the same chapter it is also attempted to relate the iterative decoder failures to the correlation of pseudo-codewords with the received vector.

Inspired by the findings of chapter 6, chapter 7 presents a novel algorithm (*Received-Vector-Coordinate-Modification* RVCM) which aims to the reduction of the effect of pseudo-codewords, and generally the reduction of any distortion effects that are introduced by the SP algorithm when applied on codes with loops.

Chapter 8, presents the performance improvement imposed by RVCM algorithm on LDPC codes. The chapter also discusses the different methods that can be used for improving the efficiency and throughput of RVCM. In accordance to chapter 8, chapter 9 discusses application of RVCM on turbo schemes.

In chapter 10 RVCM is compared to alternative convergence-improvement schemes for LDPC codes, in terms of performance and added computational load.

Finally, the author's conclusions and future research directions are presented in the last chapter.

2 Iterative Decoding Algorithms

Iterative decoding techniques bridged the gap between code design and decoding complexity, with the minimum cost in performance. This chapter focuses on the two major iteratively decoded schemes, namely turbo and low-density parity-check (LDPC) codes. In the following pages the reader will be introduced into the mechanism of turbo and Sum Product (SP) iterative decoding algorithms. The reasons for the exceptional performance of turbo codes will be attributed to the way that the interleaving action shapes the weight spectrum. Basic ideas of message passing algorithms, such as the concept of extrinsic information, will be also explained.

2.1 Turbo codes

Turbo codes where first presented in 1993 by *Berrou et al* [9] and are considered as a milestone in the field of error correction coding. The authors presented results just 0.5dB away from the Shannon sphere packing capacity limit with a scheme of affordable complexity. The classical turbo scheme was based on parallel concatenation (PCCC) of two recursive systematic convolutional (RSC) encoders but the turbo concept can be extended to serial concatenation (SCCC) [7], concatenation of more than two encoding/decoding stages [29], or hybrid [16, 15] concatenation. Schematic graphs of the encoding and decoding stages of PCCC and SCCC systems are shown in figure 2.1. In the description of the algorithms we assume BPSK modulation with antipodal signals $c_i \in \{-1, 1\}$, i=0,1,...,N, and AWGN channel.

2.1.1 Encoding of PCCC and SCCC turbo schemes

Figures 2.1(a) and 2.1(b) depict the block diagrams of the encoding stage of PCCC and SCCC schemes. Starting with the case of the PCCC encoder two different versions of the information sequence $\mathbf{x_1} = x_{10}, x_{11}, ..., x_{1(k-1)}$ are streamed into the RSC encoders. RSC1 accepts $\mathbf{x_1}$ in the normal order while the same sequence is interleaved prior entering to RSC2. The interleaving action is crucial for the performance of turbo codes [6, 47]. Its beneficial effect in the outcome of the decoding can be explained and justified in various

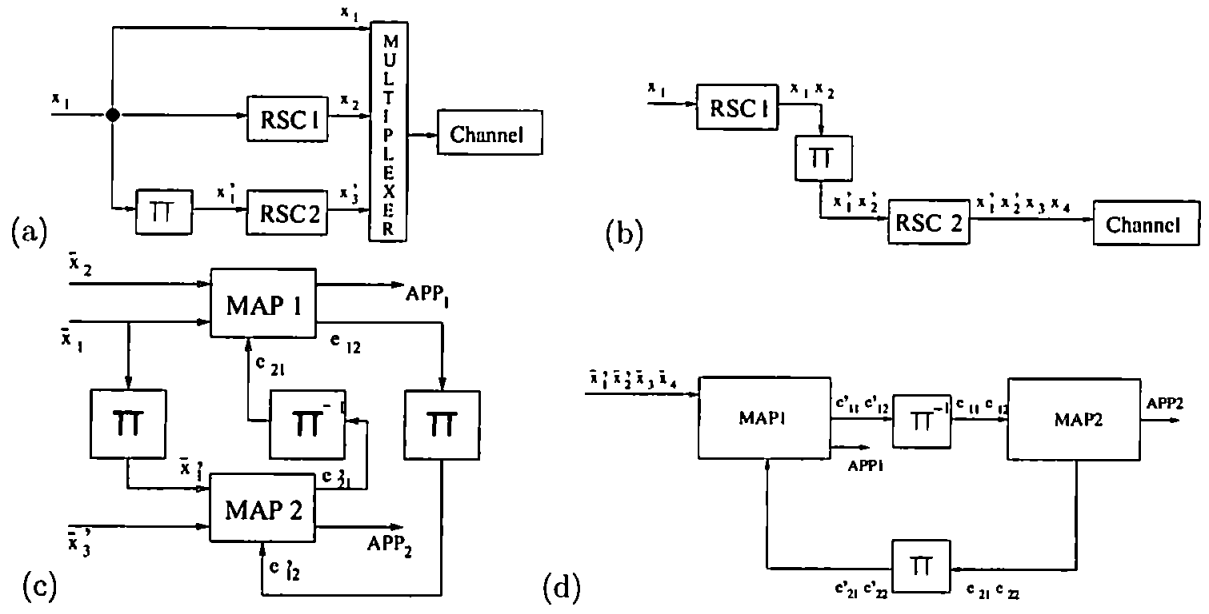


Figure 2.1: Schematic flow graphs of (a) PCCC encoder (b) SCCC encoder (c) PCCC decoder and (d) SCCC decoder

ways:

- *Randomisation of burst errors:* Depending on the type of the channel, the appropriate choice between codes with burst or random error correcting capability (like turbo codes) can be made. However, even in random error channels small bursts of errors are frequent and can be the source of erroneous decisions at the output of the decoder, especially when short memory constituent codes are used. The use of an interleaver guarantees with high probability (at least for long block-lengths) that any burst errors produced by the channel will be randomised.

- *Pseudo-randomness:* Although random codes are considered to perform better, structured codes are preferred mainly because of the much lower complexity that is required at the decoder side. The presence of the interleaver gives a sort of randomness, in the sense that bits from widely separated positions of the block (hence uncorrelated) are interleaved in neighbouring positions at the input of the second MAP decoder. However, as it will be seen later, after the first few iterations correlations start developing among these bits and the sense of randomness gradually fades.

- *Reduction of low-weight sequences:* The effect of the interleaving process is crucial on the weight spectrum of the code. It is very unlike (especially for longer block-

lengths) that an information sequence that produces a low-weight parity sequence at the output of the first constituent encoder, will also produce a low-weight parity sequence after the second encoding stage when an interleaver is intervened between the two. That property is responsible for the so called *spectral-thinning* and is the major reason for the exceptional performance of turbo codes. More about this property will be discussed in later section.

Assuming half rate constituent RSC codes, the systematic information sequence $\mathbf{x}_1$ and the non-systematic parity sequences $\mathbf{x}_2$ and $\mathbf{x}_3$ from RSC1 and RSC2 are transmitted over the AWGN channel, and are received as $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$ and $\hat{\mathbf{x}}_3$.

$$\begin{aligned}
\hat{\mathbf{x}}_1 &= \{x_{10} + n_{10}, x_{11} + n_{11}, ..., x_{1(i+k-1)} + n_{1(i+k-1)}\} \\
\hat{\mathbf{x}}_2 &= \{x_{20} + n_{20}, x_{21} + n_{21}, ..., x_{2(i+N-k-1)} + n_{2(i+N-k-1)}\} \\
\hat{\mathbf{x}}_1 &= \{x_{30} + n_{30}, x_{31} + n_{31}, ..., x_{3(i+N-k-1)} + n_{3(i+N-k-1)}\}
\end{aligned}$$

Where n_{1i}, n_{2i} and n_{3i} are the Gaussian, mutually independent noise samples.
In the serial configuration the information bits and the parity sequence of RSC1 are all streamed into RSC2. In that case the parity sequence of RSC1 is also systematic and the overall rate of an SCCC scheme is lower than that of a PCCC scheme that uses the same constituent RSC codes.

2.1.2 Turbo decoding

Before describing the turbo decoding procedure it is essential to establish the optimal maximum a posteriori (MAP) rule [5], so that the compromises made by the iterative turbo decoding algorithm can be easily demonstrated later on. For a PCCC encoder, consider the code $\mathcal{C}$ consisting of 2^k codewords $\mathbf{c} \in \mathcal{C}$.

$$\begin{aligned}
\mathbf{c} =& c_0, c_1, ..., c_N = (2 \cdot x_{10} - 1), (2 \cdot x_{11} - 1), ..., (2 \cdot x_{1,(k-1)} - 1), (2 \cdot x_{20} - 1), ..., \\
& (2 \cdot x_{2,k(\frac{1-r_1}{r_1})-1} - 1), (2 \cdot x_{30} - 1), ..., (2 \cdot x_{3,k(\frac{1-r_2}{r_2})-1} - 1)
\end{aligned}$$

Where N is the block-length, r is the overall code rate, while r_1 and r_2 are code rates of the associated constituent codes.
The maximum a posteriori probability for the information bit x_{1i} in terms of likelihood

ratios is expressed as

$$\frac{Pr(x_{1i}=1|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3)}{Pr(x_{1i}=0|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3)},\ i=0,1,2,\ldots,k \tag{2.1}$$

By using Bayes's rule the previous equation becomes

$$\frac{Pr(x_{1i}=1|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3)}{Pr(x_{1i}=0|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3)}=\frac{\sum_{\mathbf{x}_1:x_{1i}=1}p(\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)}{\sum_{\mathbf{x}_1:x_{1i}=0}p(\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)} \tag{2.2}$$

Since we assume a AWGN channel, the noise samples are independent and the naive Bayes's rule can be applied so that

$$p(\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3|\mathbf{x}_1)=p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1) \tag{2.3}$$

Thus the optimal MAP equation of 2.2 can be written as

$$\frac{\sum_{\mathbf{x}_1:x_{1i}=1}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)}{\sum_{\mathbf{x}_1:x_{1i}=0}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)} \tag{2.4}$$

Under the assumption of uniform a priori probability function for the information stream $Pr(\mathbf{x}_1)$, the corresponding term can be omitted and the optimal MAP decision can be finally expressed as

$$\frac{\sum_{\mathbf{x}_1:x_{1i}=1}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)}{\sum_{\mathbf{x}_1:x_{1i}=0}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)} \tag{2.5}$$

Optimal decoding of the interleaved code based on the above rule would require the use of a $2^{v_1+v_2+N}$ state trellis. Hence, for practical block-lengths, application of the BCJR [5] algorithm would be a prohibitively complex task. Here comes our previous statement about the difficulty of decoding a random code, and the need for structured schemes.
Turbo decoders offer an alternative and feasible, in terms of complexity, approach. The joint trellis of the overall code $\mathcal{C}$ is substituted by the much simpler trellis diagrams of RSC1 and RSC2. So the overall decoding ($2^{v_1+v_2+N}$ trellis states, where v_1 and v_2 the number of states of the first and second RSC trellis diagram) is broken up into two individual MAP decoding operations which are performed at the constituent trellises ($2^{v_1}+2^{v_2}$ states).
For the case of a PCCC scheme with half-rate RSC codes, the first constituent decoder applies maximum a posteriori decoding (on the trellis of RSC1) on the sequences $\mathbf{x}_1$ and $\mathbf{x}_2$, while the second decoder operates optimally (on the trellis of RSC2) on the sequences $\mathbf{x}_1$ and $\mathbf{x}_3$. By that way the structure is retained and decoding becomes significantly easier. The optimum rule for each of the two constituent decoders would then be:

For the 1^{st} MAP decoder

$$\frac{Pr(x_{1i}=1|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2)}{Pr(x_{1i}=0|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2)}=\frac{\sum_{\mathbf{x}_1:x_{1i}=1}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)}{\sum_{\mathbf{x}_1:x_{1i}=0}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)} \tag{2.6}$$

For the 2^{nd} MAP decoder

$$\frac{Pr(x_{1i}=1|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_3)}{Pr(x_{1i}=0|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_3)}=\frac{\sum_{\mathbf{x}_1:x_{1i}=1}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)}{\sum_{\mathbf{x}_1:x_{1i}=0}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)} \tag{2.7}$$

Both equations are just approximations of the optimum decoding rule of 2.5. In both equations the a priori probability of the systematic bit $Pr(x_1)$ has substituted the variable that is missing from the constituent trellises. The question that rises is how is it possible to take the full advantage of the code's capabilities and perform optimal decoding based on the restrictions of the joint trellis.

At this point comes the iterative concept. To account for the missing variable on the constituent decoders' expressions, the two decoders exchange information regarding their own estimation for that variable. That estimation can be used to substitute the a priori probability $Pr(\mathbf{x}_1)$ which, under the uniform probability assumption for $\mathbf{x}_1$, weights all terms equally and does not contribute to the final decisions of the constituent decoders (equations 2.6 and 2.7). Direct substitution of $p(\mathbf{x}_3|\mathbf{x}_1)$ and $p(\mathbf{x}_2|\mathbf{x}_1)$ in equations 2.6 and 2.7 respectively, would make both identical to the optimal expression of 2.5 but would not make much sense since none of the substituting variables imposes the constraints of the constituent trellises. On the other hand, direct feeding of each decoder stage output to the next, would create a kind of positive feedback. Information that has been produced by the constituent decoder 1 would be fed back to itself. That would bias the final decision of the decoder, eliminate any independency assumption between the two decisions and would cause instability to the turbo decoder dynamic non-linear system. Let us examine that case by defining the set of APP decisions of each decoding stage for bit i at iteration t as $\mathcal{D}_{1i}^{(t)}(x_1)$ and $\mathcal{D}_{2i}^{(t)}(x_1)$ $\forall i{:}0\leq i\leq k$. Then

$$\frac{\mathcal{D}_{1i}^{(0)}(1)}{\mathcal{D}_{1i}^{(0)}(0)}=\frac{\sum_{\mathbf{x}_1:x_{1i}=1}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)}{\sum_{\mathbf{x}_1:x_{1i}=0}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1)\cdot Pr(\mathbf{x}_1)} \tag{2.8}$$

After decoder-1 has decided for bit i, $\mathcal{D}_{1i}^{(0)}(1)$ and $\mathcal{D}_{1i}^{(0)}(0)$ are fed to decoder-2, so that

$$\frac{\mathcal{D}_{2i}^{(0)}(1)}{\mathcal{D}_{2i}^{(0)}(0)}=\frac{\sum_{\mathbf{x}_1:x_{1i}=1}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot \mathcal{D}_{1i}^{(0)}(1)}{\sum_{\mathbf{x}_1:x_{1i}=0}p(\hat{\mathbf{x}}_1|\mathbf{x}_1)\cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1)\cdot \mathcal{D}_{1i}^{(0)}(0)} \tag{2.9}$$

At that point is the beginning of the first iteration and the decisions of decoder-2 are fed back to decoder-1 for the new estimations.

$$\frac{\mathcal{D}_{1i}^{(1)}(1)}{\mathcal{D}_{1i}^{(1)}(0)} = \frac{\sum_{\mathbf{x}_1:x_{1i}=1} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1) \cdot \mathcal{D}_{2i}^{(0)}(1)}{\sum_{\mathbf{x}_1:x_{1i}=0} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1) \cdot \mathcal{D}_{2i}^{(0)}(0)} \tag{2.10}$$

If we now expand $\mathcal{D}_{2i}^{(0)}(x_1)$ in the above equation, we have

$$\frac{\mathcal{D}_{1i}^{(1)}(1)}{\mathcal{D}_{1i}^{(1)}(0)} = \frac{\sum_{\mathbf{x}_1:x_{1i}=1} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1) \cdot \mathcal{D}_{1i}^{(0)}(1)}{\sum_{\mathbf{x}_1:x_{1i}=0} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1) \cdot \mathcal{D}_{1i}^{(0)}(0)} \tag{2.11}$$

which shows that the output of decoder-1 for bit i at iteration 0, is fed back to decoder-1 as a priori input at iteration 1. By continuing in the same manner we would notice that the same happens with decoder-2 and that this positive feedback effect is carried on at all iterations. The practical consequence of the positive feedback effect is that the turbo decoder resembles an oscillatory behaviour throughout the iterative process, with no signs of convergence to a solution.

To avoid that problem it is necessary that the concatenated decoders do not exchange raw APP estimations but only certain information extracted from this. Consider equation 2.7 for the decision of information bit i from decoder-2. This can be rewritten as

$$\frac{Pr(x_{1i}=1|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_3)}{Pr(x_{1i}=0|\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_3)} = \frac{p(\hat{x}_{1i}|x_{1i}=1)}{p(\hat{x}_{1i}|x_{1i}=0)} \cdot \frac{Pr(x_{1i}=1)}{Pr(x_{1i}=0)} \cdot \underbrace{\frac{\sum_{\mathbf{x}_1:x_{1i}=1} p(\hat{\mathbf{x}}_3|\mathbf{x}_1) \prod_{j\neq i} p(\hat{x}_{1i}|x_{1j}) p(\hat{x}_{1j}|x_{1j})}{\sum_{\mathbf{x}_1:x_{1i}=0} p(\hat{\mathbf{x}}_3|\mathbf{x}_1) \prod_{j\neq i} p(\hat{x}_{1i}|x_{1j}) p(\hat{x}_{1j}|x_{1j})}}_{extrinsic} \tag{2.12}$$

The denoted part on the above equation is the "extrinsic information" that is exchanged between the two decoders and it involves the contribution of all other bits except bit i to the estimation of the probability for bit i. By feeding back only the extrinsic part, the positive feedback problem that was denoted in equation 2.11 is eliminated. Decoder-1 does not accept as new a priori information the decision that itself produced in the previous iteration. It accepts only the independent decisions of decoder-2 which have been based on the previous iteration decisions of decoder-1 and vice versa.

Retaining the expressions for the APP decisions of the two decoders at iteration 0 ($\mathcal{D}_{1i}^{(t)}(x_1)$ and $\mathcal{D}_{2i}^{(t)}(x_1)$) from equations 2.8 and 2.9 respectively (note that at iteration 0 decoder-2 has not yet produced any decisions and the raw APP estimations from decoder-

1 can be fed directly), the new APP decisions at iteration 1 can be expressed as

$$\frac{\mathcal{D}_{1i}^{(1)}(1)}{\mathcal{D}_{1i}^{(1)}(0)} = \frac{\sum_{\mathbf{x}_1:x_{1i}=1} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1) \cdot \mathcal{E}_{2i}^{(0)}(1)}{\sum_{\mathbf{x}_1:x_{1i}=0} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_2|\mathbf{x}_1) \cdot \mathcal{E}_{2i}^{(0)}(0)} \tag{2.13}$$

$$\frac{\mathcal{D}_{2i}^{(1)}(1)}{\mathcal{D}_{2i}^{(1)}(0)} = \frac{\sum_{\mathbf{x}_1:x_{1i}=1} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1) \cdot \mathcal{E}_{1i}^{(1)}(1)}{\sum_{\mathbf{x}_1:x_{1i}=0} p(\hat{\mathbf{x}}_1|\mathbf{x}_1) \cdot p(\hat{\mathbf{x}}_3|\mathbf{x}_1) \cdot \mathcal{E}_{1i}^{(1)}(0)} \tag{2.14}$$

Where $\mathcal{E}_{vi}^{(t)}(x_{1i})$ denotes the extrinsic information produced by decoder-v at iteration t for the systematic bit i, and can be generally expressed as

$$\mathcal{E}_{1i}^{(t)}(x_{1i}) = \frac{\mathcal{D}_{1i}^{(t)}(x_{1i})}{\mathcal{E}_{2i}^{(t-1)}(x_{1i})} \tag{2.15}$$

$$\mathcal{E}_{2i}^{(t)}(x_{1i}) = \frac{\mathcal{D}_{2i}^{(t)}(x_{1i})}{\mathcal{E}_{1i}^{(t)}(x_{1i})} \tag{2.16}$$

The same turbo concept is retained in the case of SCCC schemes. The only difference is that in addition to the information bits, the parity bits of the first encoder are also systematic so extrinsic information is exchanged between the constituent decoders for those bits too.

2.1.3 Weight spectrum of turbo codes

The results from [6] and [46] came to justify the exceptional performance of turbo codes from a weight spectrum perspective. The effect of the interleaver on the weight spectrum of the code, and the impact on the performance of the code will be examined here.
Let us assume an RSC convolutional encoder. The input information sequence $\mathbf{s_i}$ ($i = 0, 1, ..., 2^k - 1$) will produce at the output of the encoder a parity sequence $\mathbf{p_i}$. The hamming weight of the resulted codeword $\mathbf{c_i}$ will be the sum of the hamming weights of the two sequences.

$$w(\mathbf{c_i}) = w(\mathbf{s_i}) + w(\mathbf{p_i}) \tag{2.17}$$

The codeword with the lowest hamming weight defines the minimum distance of the code d_{min}.

$$d_{min} = min(w(\mathbf{c_i})) \tag{2.18}$$

The number of codewords with hamming weight equal to j is expressed by the multiplicity m_j. If $\mathbf{s_i}$ is one of these information sequences that produce a d_{min} codeword at the output

of the encoder, then any shifted version $\dot{\mathbf{s}}_i$ of $\mathbf{s_i}$ will produce just a shifted version $\dot{\mathbf{p}}_i$ of $\mathbf{p_i}$ (assuming that the code is properly terminated and no ones are lost during the shifting operation). Thus, shifted versions of low weight generating input sequences will produce low weight parity sequences and use of longer block lengths N would proportionally increase the multiplicity of d_{min} codewords $m_{d_{min}}$. The ratio of multiplicity over block-length N would be

$$\lim_{N\to\infty} \frac{m_{d_{min}}}{N} \approx \kappa \tag{2.19}$$

where κ is a constant.

Consider now a turbo encoding scheme like the one in figure 2.1. Assume the parity sequences of the two constituent encoders as $\mathbf{p_{1i}}$ and $\mathbf{p_{2i}}$. Let $\mathbf{s_i}$ be an information input sequence that produces a low weight parity sequence $\mathbf{p_{1i}}$ at the output of the first constituent encoder. The interleaved version $\mathbf{s'_i} = \Pi(\mathbf{s_i})$ is encoded by the second constituent encoder. It is wise at this point to examine some properties of the interleaving function.

$$\Pi(E(\mathbf{s})) \neq E(\Pi(\mathbf{s})) \tag{2.20}$$

E denotes the convolutional encoding operation and Π represents the interleaving function. Direct consequence of the above is that

$$E(\mathbf{s}) \neq \Pi^{-1}(E(\Pi(\mathbf{s}))) \tag{2.21}$$

where Π^{-1} refers to the de-interleaving function. The above properties denote that the weight $w(\mathbf{s_i})$ of the sequence $\mathbf{s_i}$ at the output of the first encoding stage is not guaranteed to be equal to the weight of the sequence $\mathbf{s'_i} = E(\Pi(\mathbf{s}))$ at the output of the second encoder stage. Therefore, it is quite probable that $w(\mathbf{p_{1i}}) \neq w(\mathbf{p_{2i}})$. The crucial question is what are the chances that a low weight input sequence (and particularly a minimum weight input sequence) will be interleaved in such a way so that a sequence of higher weight is produced at the output of the second encoder, i.e. the turbo encoder output.

Consider a minimum weight information sequence $\mathbf{s_i}$ (for terminated RSC codes the minimum weight cannot be less than 2) and assume that after encoding it produces a minimum weight parity sequence $w(\mathbf{p_{1i}})$. All shifted versions of the information sequence $\mathbf{s_i}$ will cause the same minimum weight error event. Given that the interleaver length is N_Π, there will be N_Π such permutations. So, the ratio of bad permutations (that just shift the low weight event) over all possible permutations will be

$$\textit{Ratio of bad } (w_{min}) \textit{ permutations} = \frac{N_\Pi}{\binom{N_\Pi}{w_{min}}} \tag{2.22}$$

Consequently, given that $w(\mathbf{p}_{1l}) = w_{min}$, the probability that the input sequence will be permuted in such a way that $w(\mathbf{p}_{2l}) = w(\mathbf{p}_{1l}) = w_{min}$ can be approximated by the following expression.

$$P_{higher\ weight} = 1 - \frac{N_\Pi}{\binom{N_\Pi}{w_{min}}} = 1 - \frac{N_\Pi}{N_\Pi^{w_{min}}} = 1 - N_\Pi^{1-w_{min}} \tag{2.23}$$

Equation 2.23 shows that the probability that RSC2 will also generate a low weight codeword reduces as N_Π grows longer and for higher information sequence weight w. This phenomenon is called *interleaver gain* and as a result of that, the multiplicity of low weight codewords reduces, i.e. the spectrum of turbo codes becomes *thinner* at the lower terms' side (*spectral thinning*).

The impact of the spectral thinning on the performance is clearly verified by the *union bound* equations [46]. Considering just a pair of codewords, the probability that an optimum decoder will erroneously choose codeword $\mathbf{c_j}$ instead of the transmitted codeword $\mathbf{c_i}$ is a function of the rate r, the E_b/N_o and the hamming distance between the two codewords.

$$P_j = Q\left(\sqrt{d \cdot \frac{2rE_b}{N_o}}\right) \tag{2.24}$$

To draw an upper bound for the overall code the above calculation is repeated for all codeword pairs.

$$P \leq \sum_{d=d_{min}}^{d_{max}} m_d Q\left(\sqrt{d \cdot \frac{2rE_b}{N_o}}\right) \tag{2.25}$$

where, as before, m_d is the multiplicity of distance d. The above equations are expressed in terms of frame error rate (FER). To express the union bound in terms of the bit error rate (BER) it is a necessity to include the average information weight (w_d) of all paths with Hamming distance d, multiplied by the multiplicity of distance d and normalised by N.

$$P \leq \sum_{d=d_{min}}^{d_{max}} \frac{m_d w_d}{N} Q\left(\sqrt{d \cdot \frac{2rE_b}{N_o}}\right) \tag{2.26}$$

At higher E_b/N_o the lower distance term d_{min} dominates and the above equations can be accurately approximated by

$$P \leq \frac{m_{d_{min}} w_{d_{min}}}{N} Q\left(\sqrt{d_{min} \cdot \frac{2rE_b}{N_o}}\right) \tag{2.27}$$

which is called the *minimum distance asymptote.*

The ratio $\frac{m_d w_d}{N}$ is called the *effective multiplicity* of distance d (d_{min} in this case). For convolutional codes this will be well above 1 because all shifted versions of a d_{min} generating information sequence produce d_{min} events. So, in order to improve the asymptote performance of the convolutional code, the only option is to increase the minimum distance of the code with the associated rise in the complexity of the decoder of course.

In turbo codes though, the previously described time-varying property of the interleaver ensures with high probability, which increases for longer block lengths, that the second encoder parity sequence will be of higher weight. As a consequence of that, the low weight terms are reduced and the effective multiplicity ratio becomes

$$\frac{m_{d_{min}} w_{d_{min}}}{N} \ll 1 \tag{2.28}$$

If we consider a high d_{min} convolutional code and a low d_{min} turbo code, the asymptotic performance of the convolutional code due to equation 2.27 will be much steeper. However, the lower effective multiplicity coefficient exhibited by turbo codes results in lower asymptotic performance for sufficiently small E_b/N_o. As E_b/N_o becomes larger, the rapidly decreasing convolutional code and the slowly decreasing turbo code performances come closer and eventually they cross. After this point the role of d_{min} becomes dominant and the error performance of the higher d_{min} convolutional code keeps reducing while the low d_{min} turbo code performance flattens out.

The above reveal that minimisation of the multiplicity of low distance terms is equally important to maximisation of d_{min} and that turbo codes owe their exceptional performance to their thin weight spectrum structure. At higher SNR where the minimum distance becomes dominant, their performance flattens out and that SNR region is known as the *error floor.* We could say that in contrast to the traditional code design philosophy where d_{min} had been considered as the one and only figure of merit, turbo codes offered a new perspective and redefined the standard idea of what a good code is. In a simplified 2-D visualisation, the practical effect of the reduced number of low weight terms is demonstrated in figure 2.2.

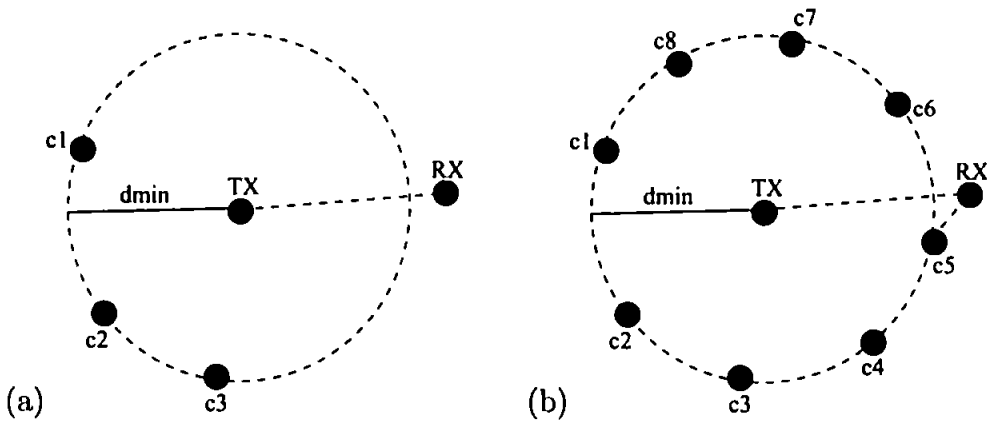


Figure 2.2: (a) Low multiplicity of d_{min} terms (b) high multiplicity of d_{min} terms

2.2 Low-Density Parity-Check (LDPC) codes

LDPC codes were first discovered by Gallager [28] in the early 1960s. The limiting computing power of those years prohibited their practical implementation and the existence of LDPC codes was ignored for several years. The discovery of turbo codes in the early 1990s turned the attention of researchers to iterative decoding and led to the rediscovery of LDPCs. Since then, LDPCs are becoming more and more popular due to their simplicity, structured form and performance. With proper design and use of long-block lengths, LDPCs can offer performance very close to the channel capacity. Characteristically, simulations have shown performance within 0.0045dB from the Shannon limit at a bit error rate of 10^{-6} for $N = 10^7$ LDPC codes [12].

As their name implies, LDPC codes are based on low density (sparse) $\mathbf{H}$ matrix structures. The sparse structure has a double advantageous effect when considering iterative decoding; It reduces complexity and at the same time it boosts the performance of the iterative decoder.

The encoding operation in LDPC codes is relatively simple and efficient. Simple manipulation of the $\mathbf{H}$ matrix (by the Gaussian elimination method) can easily transform it into encodable form.

In his original work, Gallager did not provide a specific method for constructing good LDPC codes, but he proposed a general method for constructing a class of pseudo-random codes. Later works introduced various construction methods based on finite-geometries [36] that will not be dealt in this thesis. The following sections will offer a description of the structure and philosophy behind LDPC codes with emphasis on the decoding side.

2.2.1 Description of LDPC codes

LDPC codes are specified by their $J \times N$ parity check matrix $\mathbf{H}$, where J is equal to the number of parity bits $J = N - k$ of the code. An N-tuple binary sequence $\mathbf{c} = (c_0, c_1, ...c_{N-1})$ is considered as a valid codeword only if $\mathbf{cH^T} = \mathbf{0}$ over the binary field $GF(2)$. The code C is defined as the null space of $\mathbf{H}$. Each row $\mathbf{h}_j = (h_{j,0}, h_{j,1}, ..., h_{j,(N-1)})$ of $\mathbf{H}$ defines a parity check so that $\mathbf{ch_j^T} = \mathbf{0}$.

The parameters w_r and w_c represent the row and column weight (the number of ones on each row and column) of $\mathbf{H}$, respectively. If w_r and w_c are constant for all rows and columns of the parity check matrix, then the code is called *regular*. In the opposite case the code is called *irregular*.

In this section's introduction we referred to the need of keeping $\mathbf{H}$ sparse, so that the complexity is kept low and the iterative decoder performs satisfactorily (by minimising the probability that short cycles exist within the code structure). The density ρ of the parity check matrix is defined as the ratio of the total number of 1's in $\mathbf{H}$ to the total number of entries $N \times J$. Thus, for a regular code

$$\rho = \frac{w_r}{N} \cdot \frac{w_c}{J}$$

The minimum distance of an LDPC code can be determined from its $\mathbf{H}$ matrix as normal, by examining the minimum number of columns d, the J-tuple sum (over $GF(2)$) of which becomes $\mathbf{0}$.

2.2.2 Graphical representation of LDPC codes

Tanner's work [64] in 1981 provided a new interpretation of LDPC codes from a graphical point of view. Tanner showed how LDPC codes can be represented by *bipartite* graphs that closely mirror the parity-check matrix representation of the code. Bipartite graphs that represent a code structure are generally known as Tanner graphs.

A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists of a set of vertices $\mathcal{V} = \{v_0, v_1, ...\}$ and edges $\mathcal{E} = \{\varepsilon_0, \varepsilon_1, ...\}$, and can be represented as a diagram in which the vertices are represented as points and the edges as lines that connect the vertices. A *path* within a graph $\mathcal{G}$ is defined as an alternating sequence of vertices and edges, that starts and ends to a vertex. The length of the path is equal to the number of edges. A path that starts and ends in the same vertex (closed path) is called a *cycle* of length t. The effect of the cycles, particularly of short (small length) cycles in the outcome of iterative decoding will be examined further on. The length of the shortest cycle in a certain graph is called the *girth* of the graph. If

a graph exhibits no cycles, it is called a *tree graph.*

As bipartite are considered those graphs whose vertices are partitioned into two disjoint sets, and no two vertices of the same set can be connected by an edge. In the representation of LDPC codes the vertices are partitioned into two disjoint sets. The set $\mathcal{U} = u_0, u_1, ..., u_{N-1}$ represents the N code bits and $\mathcal{S} = s_0, s_1, ..., s_{J-1}$ the J parity check equations. An edge ε connects the code bit node u_i with the parity check node s_j only if the entry of the **H** matrix $h_{j,i}$ is a 1. As an example consider the following parity check matrix.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
s_0	0	0	1	0	1	1	1
s_1	0	1	0	1	0	1	1
s_2	1	0	0	1	1	0	1

The Tanner graph of the matrix is shown in the following figure.

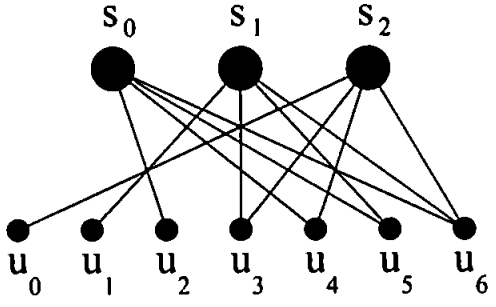


Figure 2.3: Tanner graph of the (7,4) Hamming code

2.2.3 Decoding of LDPC codes

There are several decoding methods for LDPC codes, most of them differing in decoding complexity and performance. *Majority decision* [4] and *Bit Flipping* (BF) [28] represent low complexity hard decision decoding methods. The *Sum Product* (SP) algorithm (equivalent to *Belief Propagation*) is an iterative SISO (Soft In Soft Out) symbol-by-symbol algorithm that involves higher complexity but offers superior performance. Its excellent trade-off between complexity and performance makes it the most popular decoding method for LDPC codes. The rest of the section will provide an overview of the SP algorithm.

The SP algorithm is included to the general family of *message passing* algorithms. The channel estimations for the probabilities of the transmitted bits propagate through the

code graph in an iterative manner until a valid solution that satisfies the stopping criteria has been reached. Considering the Tanner graph of a code C, messages (estimated a posteriori probabilities) are exchanged between the code bit nodes $u_i \in \mathcal{U}$ and the parity check nodes $s_i \in \mathcal{S}$.

The transmitted sequence $\mathbf{x}$ is disturbed by AWGN so that the received replica $\hat{\mathbf{x}}$ of $\mathbf{x}$ is

$$\hat{\mathbf{x}} = \{x_0 + n_0, x_1 + n_1, ..., x_{N-1} + n_{N-1}\}$$

Initially, the received vector $\hat{\mathbf{x}}$ is hard decided ($\hat{\mathbf{x}}'$) to check if it represents a valid codeword ($\mathbf{c} \in C$), i.e. if all J individual parity checks $\mathbf{h}$ are satisfied so that $\hat{\mathbf{x}}'\mathbf{H} = \mathbf{0}$ over $GF(2)$. If not, the decoding procedure commences. We define the messages from the code bit nodes $u_i \equiv x_i$ towards the check nodes s_j at iteration t as $m(b)^{(t)}_{\uparrow i,j}$ and the messages with direction from s_j towards u_i as $m(b)^{(t)}_{\downarrow j,i}$ (where $b \in [0,1]$ for BPSK). At the beginning of the decoding algorithm ($t = 0$) we initialise $m(b)^{(0)}_{\uparrow i,j} = Pr(u_i = b|\hat{x}_i)$ and $m(b)^{(0)}_{\downarrow j,i} = 1$ for all j and i. We want to find the probability that the transmitted bit at position i is $b \in [0,1]$, conditional on the set of received replicas $\hat{\mathbf{x}}$ and on the event F that all J parity check equations are satisfied.

$$Pr(u_i = b|\hat{\mathbf{x}}, F) \tag{2.29}$$

By the definition of conditional probabilities, and considering as F_j the event that parity check s_j is satisfied, we can write

$$Pr(u_i = b|\hat{\mathbf{x}}, F_j) = Pr(u_i = b|\hat{x}_i) \cdot Pr(F_j|u_i = b, \hat{\mathbf{x}}) \tag{2.30}$$

The straight forward way for calculating the probability $Pr(F_j|u_i = b, \hat{\mathbf{x}})$ that parity check s_j is satisfied conditional on the bit u_i, would be to normalise the sum of the probabilities of all sequences with even number of 1's that can occur in s_j given that $u_i = b$. Gallager came up with a much more efficient method; we refer to the Lemma 4.1 from [28]: If we consider a sequence of m independent bits in which the l^{th} bit is a 1 with probability P_l, then the probability that an even number of digits are 1 is

$$\frac{1 + \prod_{l=1}^{m}(1 - 2P_l)}{2} \tag{2.31}$$

and the probability that an odd number of bits are 1 is

$$\frac{1 - \prod_{l=1}^{m}(1 - 2P_l)}{2} \tag{2.32}$$

If $u_i = 1$ then the parity check s_j will be satisfied if the remaining $w_r - 1$ positions in the parity check set contain an odd number of ones.

$$Pr(u_i = 1|\hat{\mathbf{x}}, F_j) = K_1 \cdot Pr(u_i = 1|\hat{x}_i) \cdot \frac{1 - \prod_{l=0, l\neq i}^{(w_r-1)}(1 - 2m(b)_{\uparrow l,j}^{(0)})}{2} \tag{2.33}$$

Similarly

$$Pr(u_i = 0|\hat{\mathbf{x}}, F_j) = K_2 \cdot Pr(u_i = 0|\hat{x}_i) \cdot \frac{1 + \prod_{l=0, l\neq i}^{(w_r-1)}(1 - 2m(b)_{\uparrow l,j}^{(0)})}{2} \tag{2.34}$$

where K_1 and K_2 are normalization factors, and $Pr(u_i = 0|\hat{x}_i) = 1 - Pr(u_i = 1|\hat{x}_i)$. Since all bits are statistically independent, the probability that all J parity checks are satisfied (event F) is just the product of the probabilities of the individual parity checks being satisfied, that is

$$Pr(u_i = 1|\hat{\mathbf{x}}, F) = Pr(u_i = 1|\hat{x}_i) \prod_{j=0}^{J-1} \frac{1 - \prod_{l=0, l\neq i}^{(w_r-1)}(1 - 2m(b)_{\uparrow l,j}^{(0)})}{2} \tag{2.35}$$

which is the *a posteriori* probability for bit u_i being 1. Defining the messages $m(b)_{\downarrow j,i}^{(t)}$ exchanged between parity check equation s_j and bit node u_i at iteration t as

$$m(b)_{\downarrow j,i}^{(t)} = \frac{1 - \prod_{l=0, l\neq i}^{(w_r-1)}(1 - 2m(b)_{\uparrow l,j}^{(0)})}{2} \tag{2.36}$$

the up-going messages (from bit node u_i to parity check equation s_j) are then updated for the next iteration

$$m(1)_{\uparrow i,j}^{(1)} = Pr(u_i = 1|\hat{x}_i) \underbrace{\frac{\prod_{l=0}^{J} m(1)_{\downarrow l,i}^{(0)}}{m(1)_{\downarrow j,i}^{(0)}}}_{extrinsic} \tag{2.37}$$

The denoted ratio represents the extrinsic information, which means that the extrinsic concept had been identified by Gallagher long before it reappears with the invention of turbo codes in 1993. The SP algorithm continues in the same manner until all parity check equations have been satisfied or the maximum number of iterations has been reached. Like the turbo decoding algorithm, the SP algorithm is sub-optimal when loops exist in the $\mathbf{H}$ matrix of the code [74, 53]. If so, the vector space and consequently the decoder decisions, are distorted in a way that chapter 6 will attempt to clarify.

3 Convergence

Iterative decoders are considered suboptimal in the sense that they do not guarantee convergence to the ML solution. This chapter attempts to introduce the reader to the convergence problem, initially by describing the concept of "fixed point solutions" and the differences in the decision bounds between iterative and ML decoders. It then deals with the stability problem of fixed point solutions and the noncompliance between minimisation of the Euclidean distance and of the number of information bits in error at the same time. The last two sections deal with the role of correlations and short cycles to the convergence properties of an iteratively decoded code.

3.1 Convergence to a fixed point solution

Taking as a reference a PCCC scheme, the MAP decoding operations of the two decoders on the trellises of the constituent codes will be denoted by the functions g_1 and g_2. The arguments and outputs of the MAP functions are the sets of extrinsic log-likelihood ratios $\mathbf{L}_u^{(t)}$ at iteration t. Since the extrinsic information output of any decoder becomes the a priori input to the next, at any iteration t

$$\mathbf{L}_2^{(t)} = g_2(\Pi(\mathbf{L}_1^{(t)}))$$
$$\mathbf{L}_1^{(t+1)} = g_1(\Pi^{-1}(\mathbf{L}_2^{(t)}))$$

where Π and Π^{-1} represent the interleaving and de-interleaving operations. The combined turbo decoding function f can be expressed as

$$f(\mathbf{L}_1^{(t)}) = g_1(\Pi^{-1}(g_2(\Pi(\mathbf{L}_1^{(t)})))) \tag{3.1}$$

The decoder is considered to have converged to a solution when f becomes a contractor [58] of $\mathbf{L}_1$, i.e. when f returns its own argument.

$$\mathbf{L}_1^{(t+1)} = f(\mathbf{L}_1^{(t)}) \tag{3.2}$$

Generally, an iterative decoder function f has converged to a solution when the Caunchy criterion [58] is satisfied for a constant $a \ll 1$.

$$||f(\mathbf{L}^{(t+1)})|| \leq a||\mathbf{L}^{(t)}|| \tag{3.3}$$

After convergence has been achieved the decoder is said to have settled to a *fixed point solution* i.e. a single point in the *N dimension Euclidean space.* Considering an extrinsic LLR vector $\mathbf{L}^{(t)}$ wandering in the N dimension Euclidean space through out the iterative process, it will be attracted by a fixed point solution $\mathbf{X}$ and fulfil the Caunchy criterion of convergence only if it exists within a certain N dimension area around the fixed point solution $\mathbf{X}$, that is called *contraction region* [58]. Then, the iterative decoding function f is a *contractor* to the vector $\mathbf{L}^{(t)}$.
Figure 3.1 visualises this condition in a simplified 2-D sketch, comparing it to the case of optimal ML decoding. In contrast to ML decoding where the decision boundaries are

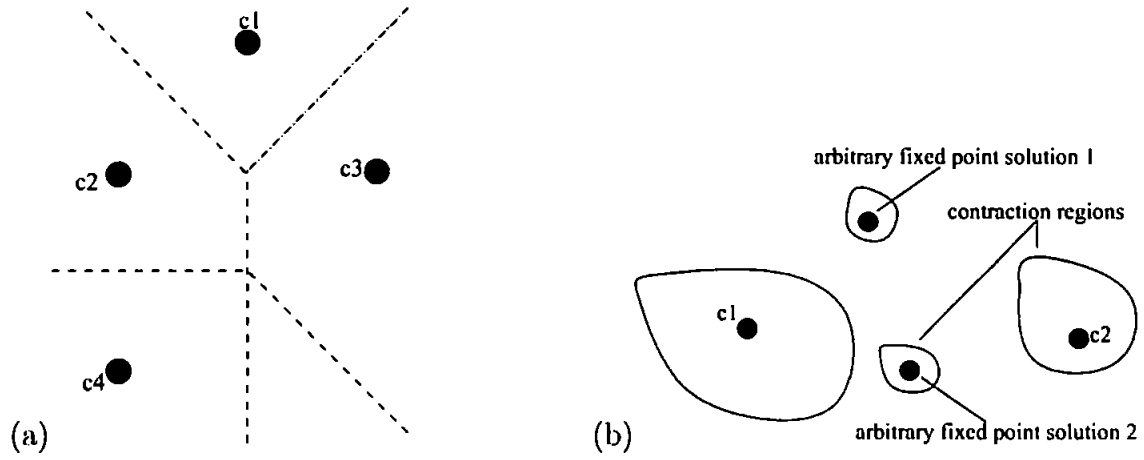


Figure 3.1: 2-D visualisation of Euclidean space for (a) ML and (b) iterative decoding

clearly set and the received vector is always associated with the closest codeword, the iterative decoder cannot guarantee optimal decoding. In fact, it is not even guaranteed that the decoder will settle to a solution after a certain limited number of iterations. Assuming that a solution has been achieved though, any of the three following outcomes is possible:

- The fixed point solution corresponds to the ML codeword, in which case the iterative decoder has operated optimally

- The fixed point solution corresponds to a valid codeword $\mathbf{c} \in C$ which is not the ML
- The fixed point solution does not correspond to a valid codeword

In any of the last two cases the iterative decoder has operated sub-optimaly.
The fixed points can be categorised into *unequivocal* and *indecisive* [1, 56]. For unequivocal fixed point solutions the extrinsic LLR vector L is extreme (clearly decided) and most often it corresponds to mostly correct decisions. In the indecisive case the LLR values are not clearly decided (low reliability) and invoke several errors. Evaluating the fixed points as a function of the SNR [2], their characteristics vary at different SNR regions. At very low SNR the great majority of fixed point solutions are indecisive and the BER high. As the SNR is increased, unequivocal solutions become more frequent and eventually at moderate to high SNR indecisive fixed point solutions disappear [1]. The gradual reduction, and eventually disappearance , of the "error-containing" indecisive fixed point solutions is the reason for the dramatical reduction of BER/FER at the *waterfall region*, the steepest part of the code's error probability curve. It will be shown in later section how the *density evolution* theory [14] uses that concept together with some additional assumptions to determine the *threshold* of a particular code, i.e. the E_b/N_o at which the waterfall region starts.

Eb/No [dB]	Average Block Entropy	BER
0.0	0.5998	0.1352
0.4	0.2895	0.0482
0.8	0.137	6.02×10^{-3}

Table 3.1: Average entropy of fixed point solution at various SNR in a SCCC, RSC(1,5/7) N=2000 code

3.2 Stability of the ML fixed point solution and relation between Euclidean distance and information errors

For powerful codes of high d_{min} the most frequent convergence problem at low SNR is the incapability of the algorithm to converge to a valid solution (codeword). In contrast to that, when the iterative algorithm operates on weaker codes it rarely fails to settle to a valid solution but this is not always the closest to the received vector.

The author has tried to use the reliability of the a posteriori decisions at the output of the iterative decoder as a criterion for deciding whether the offered solution is the ML or not. It was proved in practise that the reliability of the decoder's decisions is not an adequate condition for that. In a similar manner, identifying the error positions from the behaviour of their extrinsic values throughout the iterative process was proved to be impossible.

Another important aspect concerning the convergence properties of iterative algorithms is the stability of the fixed point solutions (a theoretical approach is given in [56]), and particularly the stability of the fixed point that represents the ML solution. It has been observed that in some cases, during the iterations the decoder comes up with the correct solution but does not stabilise on it. Figure 3.2 shows a typical case where the turbo decoder reaches the zero errors only at iteration 25.

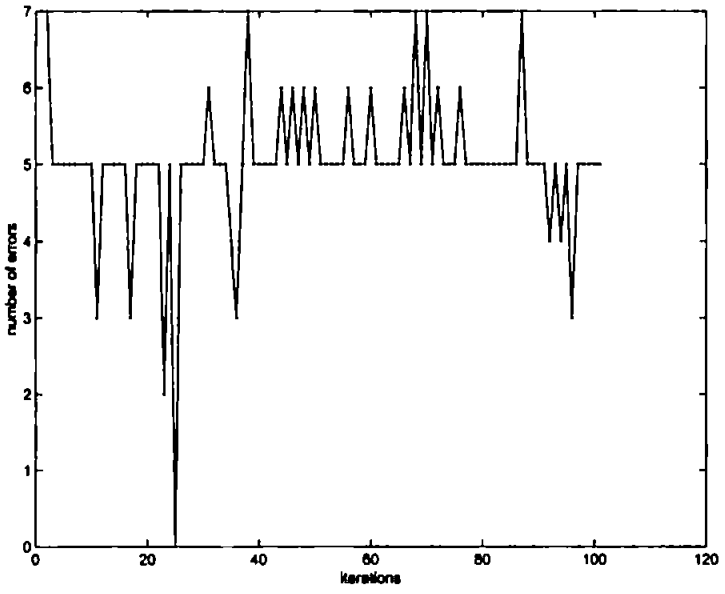


Figure 3.2: The graph shows the number of information errors with iterations

A good argument to overcome this problem would be to use the Euclidean distance of the decoded vector (at the end of each iteration) from the received vector as a measure for the most likely solution. The plots of figure 3.3 show an occasion where this method works effectively. The iterative decoder converges to the ML solution at iteration 1 but it fails to stabilise on that and converges to a sub-optimal solution of higher Euclidean distance that involves 2 information bits in error. Selecting the decoder output with the minimum Euclidean distance automatically eliminates any sub-optimality by picking up the most likely codeword.

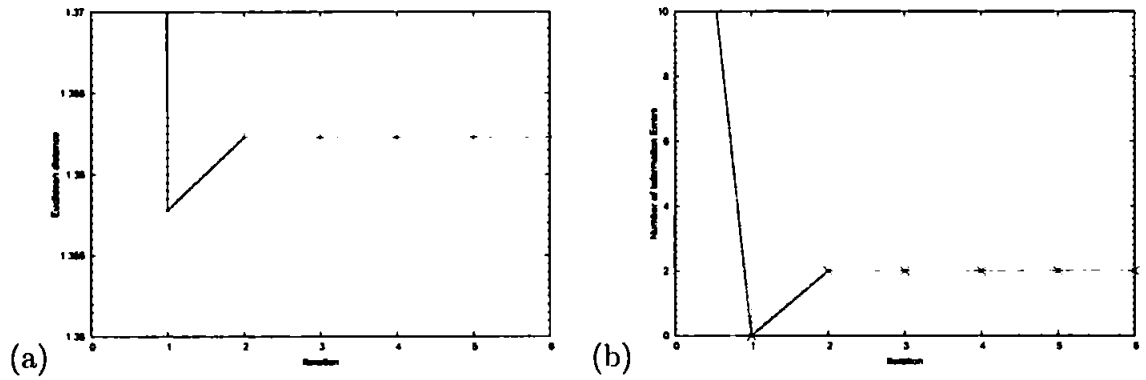


Figure 3.3: Plots of (a) Euclidean distance *vs* Iterations and (b) Number of information errors *vs* Iterations for the same block

Both the turbo and the sum-product algorithms operate on a bit-by-bit basis hence, selecting the minimum Euclidean distance decision (among the decisions at the end of each iteration) does not necessarily mean that the number of errors is minimised too. Figure 3.4 shows a typical case, based on a PCCC scheme, in which the lowest Euclidean distance decision (at iterations 6-8) does not represent the solution with the minimum number of information bits in error.

3.3 The effect of correlation in convergence

In this section we are examining the effect of correlation to the convergence properties of iterative decoders. It will be shown through experiments that the independency assumption among the variables of the iterative decoder is not valid (at least for the whole iterative process) and the emerging correlations among the variables affect the final decision [33].

In contrast to ML (MAP) decoding where the complete structure of the code is taken into account simultaneously, iterative decoders work on sub-codes with fewer variables

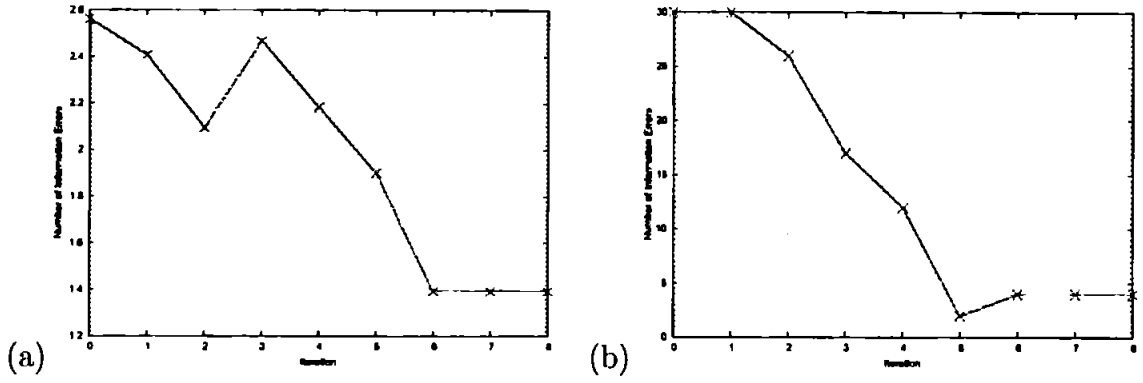


Figure 3.4: Plots of (a) Euclidean distance *vs* Iterations and (b) Number of information errors *vs* Iterations for the same block

and simpler code restrictions to reduce the complexity. To compensate for the missing information, sub-codes exchange information regarding their common bits. Therefore, the restrictions of the overall code are passed indirectly to each of the sub-codes through their common bits that have the duty to convey the new information from other parts of the code. The whole procedure is iterative thus the decoder will need a certain number of iterations until all probabilities have been spread and the sub-codes have enough information to decide optimally.

Correlation interrupts this process by restricting the propagation of probabilities. In other words, some of the decisions are mainly based on parts of the code that exhibit high correlation and the effect of the rest of the code restrictions for these decisions is minimised or ignored. After some point the two decoders do not exchange new information and the independency assumption cannot be considered valid any more.

Table 3.2 shows the error positions for a number of non-convergent blocks of a PCCC scheme with constituent codes RSC(1,5/7), block-length N=1500 and random interleaving. It can be seen that certain error combinations dominate as they occur most of the time.

The structure of the interleaver can, up to an extend, explain why some of the positions are prone to errors. Consider the pair of bits 495 and 1 that is one of the dominating error events on the previous table. The use of tail-bite termination means that those bits are only 6 positions away during the first decoding operation at MAP1 (before interleaving), therefore some considerable correlation exists between the two. The interleaving function is supposed to break this correlation by spreading the two bits far from each other within the block. The random interleaver used in this experiment though, maps the two bits next to each other ($\Pi(495) = 130$ and $\Pi(1) = 129$). The information that

Error Block No.	Error Positions	Error Block No.	Error Positions
1	163, 166	16	269, 272
3	163, 166	18	163, 166
4	117, 120	20	269, 272
5	163, 166	21	1, 495
6	269, 272	24	480, 486
7	163, 166	25	294, 297
8	480, 486	27	269, 272
9	271, 272, 277	28	1, 495
10	117, 120	29	103, 106
14	1, 495	31	471, 477
15	103, 106	34	163, 166

Table 3.2: Error positions for various error blocks

is sent back to MAP1 for the next iteration does not contribute much to its knowledge for the two bits i.e. the information is barely new. It is then highly probable that if the transmitted symbols 1 and 495 have been received with high distortion from the channel, the decoder will not be able to correct them. Two neighbouring bits that still remain in close proximity after interleaving, create a *short cycle*. Short cycles are one of the main reasons for the convergence problems of iterative decoders.

3.3.1 Correlation Coefficient

A graphical and more in depth analysis of the correlations after each decoding operation can be obtained by calculation of the *correlation coefficient* [33] among the systematic bits. Through out the iterative process the output of MAP1 becomes the input to MAP2 and vice versa. The correlation coefficient ρ can be used here to examine the validity of the independency assumption of the exchanged information. If the correlation between the extrinsic output of decoder v and its extrinsic input from decoder u is high, the information fed back to decoder u has minimum contribution to its knowledge and the whole process might get stuck. We are interested in the correlation between the output extrinsic probability e_j and the input extrinsic probability e_i. The correlation coefficient ρ_{e_i,e_j} is defined as

$$\rho_{e_i,e_j} = \frac{Cov[e_i, e_j]}{\sigma_{e_i}\sigma_{e_j}} \tag{3.4}$$

where σ is the standard deviation for e_i and e_j, and Cov is the *covariance* expressed as

$$Cov[e_i, e_j] = E((e_i - \mu_{e_i})(e_j - \mu_{e_j})) \tag{3.5}$$

Note that at the initial decoding operation (MAP1 iteration 0) where there is no available extrinsic information yet from MAP2, the correlation coefficient is calculated with respect to the channel observations of the systematic bits.

Bit 272 forms a cycle of length 4 with bit 273 ($\Pi(272) = 261$ and $\Pi(273) = 262$) and a dominating error event with bit 269. We will examine the correlations induced between bit 272 and the rest of the bits at different stages of the decoding procedure. Initially, at iteration 0 correlation peaks exist at positions adjacent to 272 as expected (figure 3.5(a)). Without the use of interleaver the correlations at the output of the second decoder would be similar. The interleaving action spreads the bits at different positions within the block and new correlation peaks should exist at different positions. In the case of bit 272 though the major correlation peaks remain around position 272 as a result of the bad interleaving at that position (figure 3.5(b)).

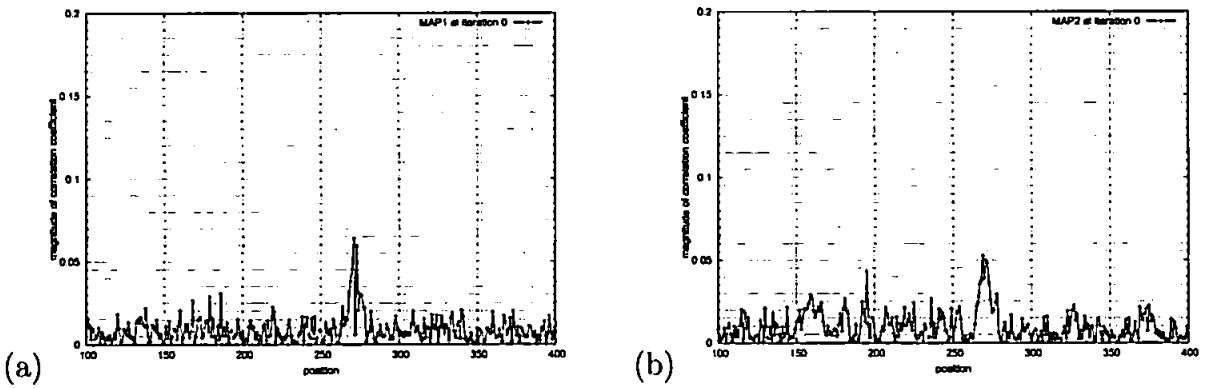


Figure 3.5: Extrinsic Output/Extrinsic Input correlation coefficient graphs for bit 272 for iteration 0 at the output of (a) MAP1 and (b) MAP2

As the decoding proceeds further the correlations increase (look at figure 3.6 and compare with figure 3.5) and the independency assumption cannot be considered valid, i.e. the extrinsic information fails to decorrelate the two decoders' decisions. This increasing dependence among the exchanged extrinsic information is the major reason for the gradual reduction of the improvement achieved by later iteration increments.

The effect of correlation to the convergence properties of turbo codes is enforced by the graphs of figure 3.7 where the peaks of the correlation coefficient ρ match to the error events of table 3.2. For example, plotting the correlation coefficients between bit 166 and the rest of the block at the output of MAP2 after the end of the third iteration, it is clear that the highest correlation peak occurs at the position 163. From table 3.2 it is seen that the pair 163,166 is one of the most frequent error events. Similar conclusions we get from the graphs for the bit positions 272, 120 and 486 where positions 269, 117 and 480

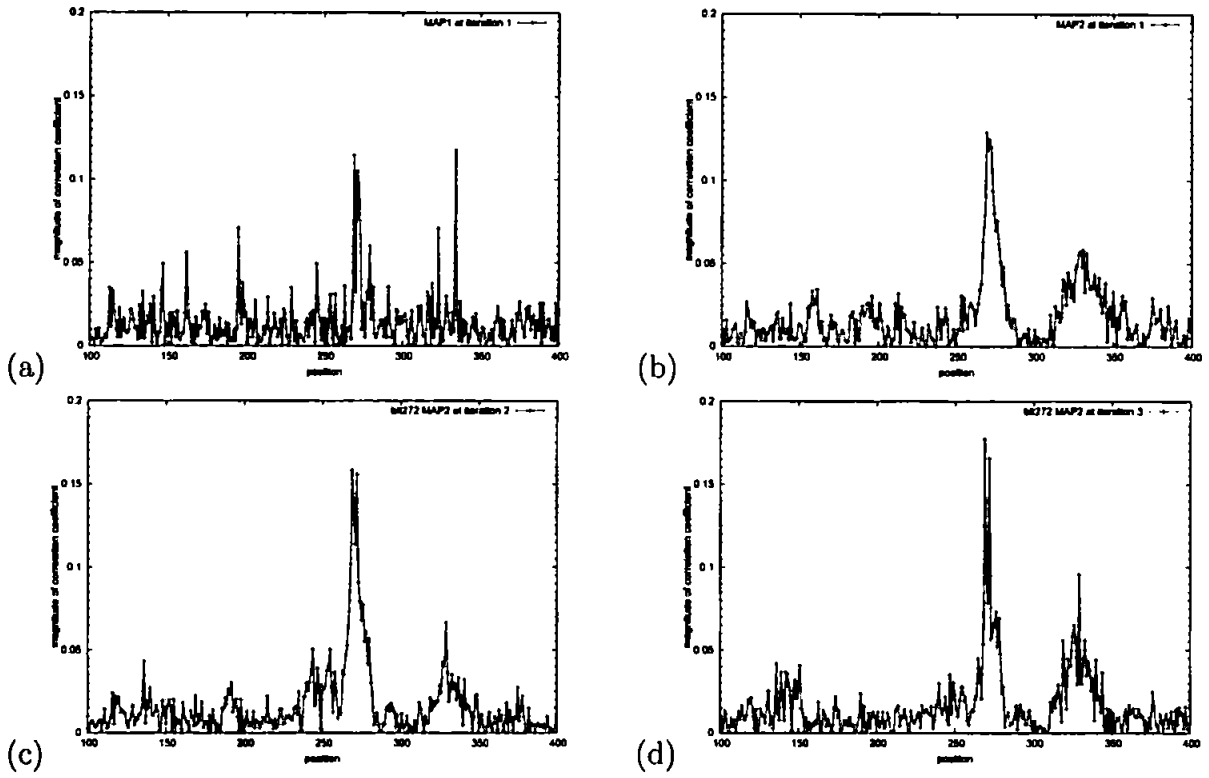


Figure 3.6: Extrinsic Output/Extrinsic Input correlation coefficient graphs for bit 272 at (a)iteration 1 MAP1 (b)iteration 1 MAP2 (c)iteration 2 MAP2 and (d)iteration 3 MAP2

respectively, exhibit the highest dependence and coincide with the error events listed on table 3.2.

3.4 The effect of cycles in the performance of iterative decoders

By using the correlation coefficient graphs for turbo decoders, it was shown how the increasing correlations between the exchanged extrinsic information are related to the most frequent error events and to the ability of the iterative decoder to converge to the maximum likelihood solution. It was also explained that the existence of short cycles (also called *short loops*) contributes to the correlation of the decisions of the two decoders. In the case of turbo codes short cycles are the result of bad interleaving. When two bits are located in close proximity before and after interleaving, a short cycle is formed.
Short cycles have the same unwanted effect on LDPC codes so it is essential that the parity check matrix **H** is free of short cycles, particularly of length 4. In general, the presence

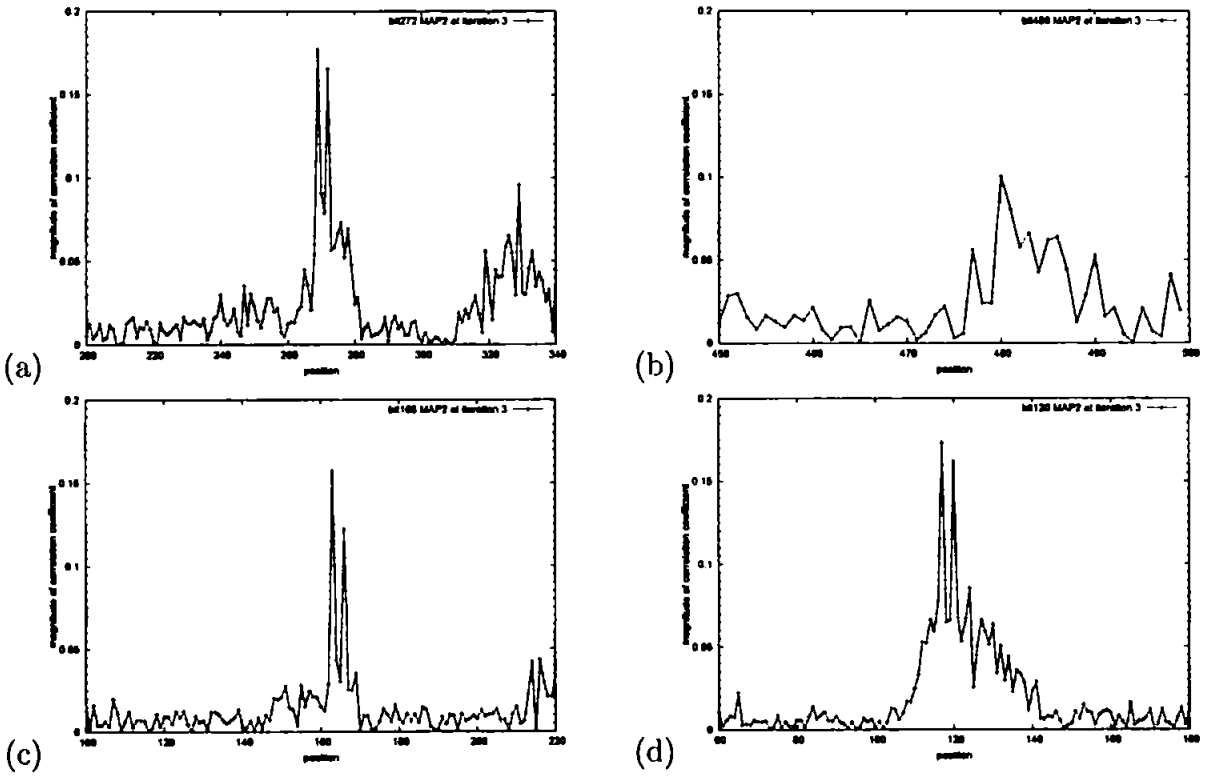


Figure 3.7: Extrinsic Output/Extrinsic Input correlation coefficient graphs of MAP2 at iteration 3 for (a)bit 272 (b)bit 486 (c)bit 166 and (d)bit 120

of short cycles results in information fed back very quickly without being influenced by many parity checks. In this chapter it is attempted to relate the existence and length of cycles with the degradation of the code's performance, by the use of *computation graphs*. A computation graph is a *tree graph* (free of loops) that traces recursively back in time all the interconnections of a certain bit (the root bit) with check and code bit nodes. It provides the same information with the original "loopy" Tanner graph but in a tree graph, thus making the analysis simpler.

Wiberg in his thesis [74] proved that the SP algorithm decides optimally when applied on a tree graph. Hence, if (U, S, Q) defines a computation graph where U represents the set of code bit nodes u that participate to the set of check nodes $S = \{s_0, s_1 ....s_j\}$ and Q defines the check sets (the code bit nodes that are connected with edges to each of the check nodes $s \in S$), application of the SP algorithm will give the optimal maximum a posteriori decisions

$$APP_i(x) = \sum_{\mathbf{f} \in F: f_{u_i} = x} Pr(\mathbf{f}|\mathbf{r}) \tag{3.6}$$

Where F includes all the binary sequences $\mathbf{f} = \{f_{u_0}, f_{u_1}, ...\}$ that satisfy the check sets Q, and $\mathbf{r}$ is the received vector.

Optimal decoding in the computation graph is not equivalent to optimal decoding in the Tanner graph when loops exist in the latter. As it will be shown in later chapter iterative decoders perform optimal decoding on the computation graph. Hence, explanation of the effect of the cycles on the decoder's behaviour is literally an answer to the reasons for the sub-optimality of iterative decoders. Chapter 6 investigates among others, the direct consequences from the presence of cycles on iteratively decoded codes. This section approaches the same problem from a different point of view and investigates how the existence and the length of cycles (we alternate the use of words cycle and loop but they both have the same meaning) alter the weight distribution of the sequences $\mathbf{f} \in F$ that contribute to the error decision of a particular bit.

Starting with a tree (loop-free) code, its parity check $\mathbf{H}$ matrix is shown in figure 3.8 together with its associated computation graph when bit position 0 (u_0) is used as root bit. Assume transmission of the all-zeros codeword $\mathbf{0}$ over a AWGN channel. The decision

(a)

	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7
s_0	1	1	0	0	1	0	0	0
s_1	0	1	1	0	0	1	0	0
s_2	0	0	1	1	0	0	1	0
s_3	1	0	0	0	0	0	0	1

(b)

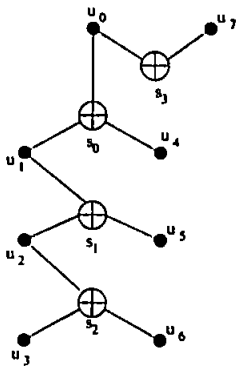


Figure 3.8: (a)tree code $\mathbf{H}$ matrix (b) computation graph

for the root bit u_0 will be in error only if

$$\sum_{\mathbf{f}\in F: f_{u_0}=0} Pr(\mathbf{f}|\mathbf{r}) < \sum_{\mathbf{f}\in F: f_{u_0}=1} Pr(\mathbf{f}|\mathbf{r}) \tag{3.7}$$

In other words an error will occur if the sum of the probabilities of those sequences that satisfy all parity checks of the computation graph and contain a 1 at u_0 is higher than those that contain a 0. The weight distribution of the error contributing sequences is given in table 3.3. Next, a cycle of length 8 is introduced at the loop-free parity check matrix of 3.8, as shown in figure 3.9. The associated computation graph is also depicted in the same figure. The weight distribution of the sequences that contribute to an error

Weight Distribution	Sequences (Positions of 1's)
$1 \cdot W^3$	u_0, u_7, u_4
$1 \cdot W^4$	u_0, u_7, u_1, u_5
$3 \cdot W^5$	u_0, u_7, u_1, u_2, u_6
	u_0, u_7, u_1, u_2, u_3
	u_0, u_7, u_4, u_6, u_3
$3 \cdot W^6$	$u_0, u_7, u_4, u_2, u_5, u_6$
	$u_0, u_7, u_4, u_2, u_5, u_3$
	$u_0, u_7, u_1, u_5, u_6, u_3$

Table 3.3: Weight distribution of tree code

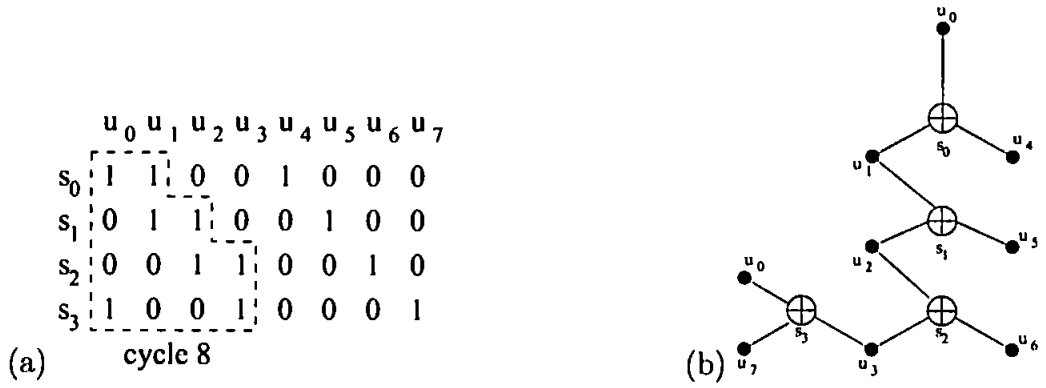


Figure 3.9: (a)**H** matrix with a cycle of length 8 (b) computation graph

decision for u_0 has now changed (table 3.4). The important point is that the number of lower weight sequences (weight 4) has now increased. The addition of a cycle in the parity check matrix **H** has slightly weakened the immunity of the code to errors.

Weight Distribution	Sequences (Positions of 1's)
$1 \cdot W^3$	u_0, u_7, u_4
$3 \cdot W^4$	u_0, u_4, u_3, u_6
	u_0, u_1, u_2, u_3
	u_0, u_7, u_5, u_1
$3 \cdot W^5$	u_0, u_4, u_2, u_5, u_3
	u_0, u_7, u_1, u_2, u_6
	u_0, u_1, u_5, u_6, u_3
$1 \cdot W^6$	$u_0, u_2, u_4, u_5, u_6, u_7$

Table 3.4: Weight distribution when a cycle of length 8 is introduced

The addition of an even shorter cycle of length 4 this time (figure 3.10), weakens the code even more by reducing the weight to the minimum value of 2.

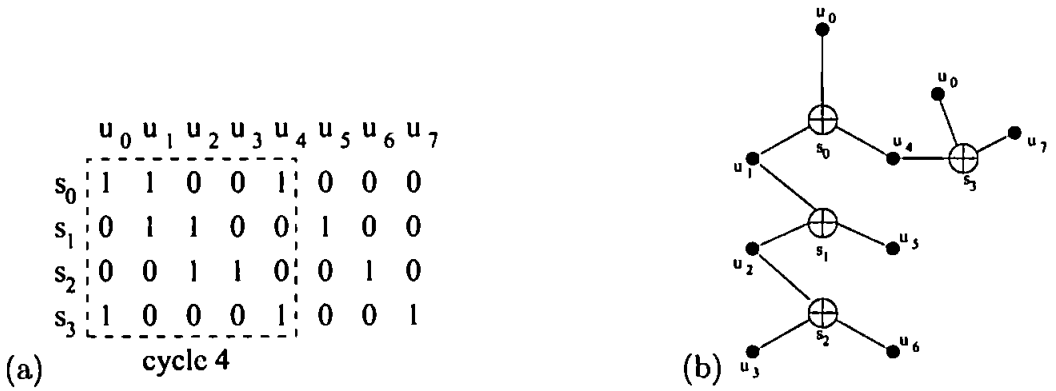


Figure 3.10: (a)**H** matrix with a cycle of length 4 (b) computation graph

Weight Distribution	Sequences (Positions of 1's)
$1 \cdot W^2$	u_0, u_4
$2 \cdot W^4$	u_0, u_1, u_7, u_5
	u_0, u_4, u_3, u_6
$4 \cdot W^5$	u_0, u_1, u_2, u_7, u_6
	u_0, u_7, u_1, u_2, u_3
	u_0, u_4, u_5, u_2, u_6
	u_0, u_4, u_5, u_2, u_3
$1 \cdot W^6$	$u_0, u_1, u_3, u_5, u_6, u_7$

Table 3.5: Weight distribution when a cycle of length 4 is introduced

4 Methods for evaluating the convergence properties of iterative decoders

This chapter describes and implements the two major methods for analysing the convergent properties of iterative decoders, namely the *Density Evolution*[14] and the *Mutual Information Evolution* [65].

4.1 Density Evolution

The *density evolution* is a useful tool for the analysis of the convergence properties of iterative decoders. It can be used to determine accurately the E_b/N_o *thresholds*, i.e. the minimum E_b/N_o required by the decoder to operate in the *waterfall region*. By this method many mysteries of iterative decoding such as the reasons why certain codes converge better than powerful codes of higher minimum distance and the role of systematic bits and recursive convolutional constituent codes to the iterative process were explained. What density evolution does is to track the probability density function of the extrinsic information messages as this density evolves iteration after iteration. In section 3.1 the fixed point solutions were evaluated as a function of the SNR and it was stated that indecisive solutions are mostly associated with the low SNR region where the BER is relatively high. At the waterfall region indecisive solutions gradually disappear and the convergence rate improves dramatically. Hence, by observing the evolution of the densities of the extrinsic extrinsic probabilities for a certain SNR it can be decided whether the code operates on the waterfall region or alternatively, what is the minimum SNR required for the code to operate on the waterfall region.
Tracking the evolving extrinsic probability densities is simplified by the fact that, as it has been investigated in [74], these can be closely approximated as Gaussian and *consistent* [54]. Density evolution was initially proposed by Richardson and Urbanke [55] for LDPC codes on AWGN channels. Divsalar et.al applied the same method on turbo-like decoders. In the rest of this subsection, the density evolution method will be described and applied on PCCC turbo schemes.

The log-likelihood ratio Z of an AWGN channel output for BPSK antipodal transmission can be written as

$$Z = \ln \frac{p(z \mid x = 1)}{p(z \mid x = -1)} \tag{4.1}$$

Where z is the noisy received signal $x + n$. If we develop the conditional probability functions, 4.1 is simplified to

$$Z = \frac{2}{\sigma_n^2} z \tag{4.2}$$

Since z is the transmitted symbol plus noise

$$Z = \frac{2}{\sigma_n^2}(x + n) \tag{4.3}$$

Where x is the systematic symbol ($x = \pm 1$) and n is a Gaussian distributed parameter with zero mean and variance $\sigma^2 = \frac{N_0}{2}$. What we have in 4.3 is multiplication of the Gaussian distributed z by a constant, and Z can be formulated as a Gaussian function.

$$Z = \mu_z x + n_z \tag{4.4}$$

Assuming transmission of the all-zeros codeword for simplicity, the mean would be

$$\mu_z = \frac{2}{\sigma_n^2} \tag{4.5}$$

For the Gaussian parameter n_z, its standard deviation σ_n is multiplied with the constant $\frac{2}{\sigma_n^2}$ and squared to give us the variance of Z

$$\sigma_z^2 = \frac{4\sigma_n^2}{\sigma_n^4} = \frac{4}{\sigma_n^2} \tag{4.6}$$

It turns out that mean and variance of the LLR distribution are related by

$$\mu_z = \frac{\sigma_z}{2} \tag{4.7}$$

which is the consistency (symmetrical) condition. For large interleavers the extrinsic LLRs λ can be considered uncorrelated from the channel LLRs Z. Additionally, it has been already observed in [74] the Gaussian like distribution of λ. Hence, λ can be modelled by applying an independent Gaussian random variable n_λ with variance σ_λ^2 and mean zero

in conjunction with the transmitted systematic bits x.

$$\lambda = \mu_\lambda x + n_\lambda \tag{4.8}$$

Since λ is (like Z) a log-likelihood ratio value based on Gaussian distribution, the consistency condition can be considered valid for λ too. The constituent decoders' decisions can be evaluated iteration after iteration based on the distribution of the extrinsic LLRs. As the decoders' outputs become more and more decisive the pdf curve moves away from the zero axis, as a clear indication that the decoder has converged and that the distance between $pdf(\lambda)$ and $pdf(-\lambda)$ has been maximised. In figure 4.1 is shown the evolution of the extrinsic LLRs pdf for a convergent block after 3 iterations have been performed. Note the Gaussian shape of the distribution and the validity of the consistency approx-

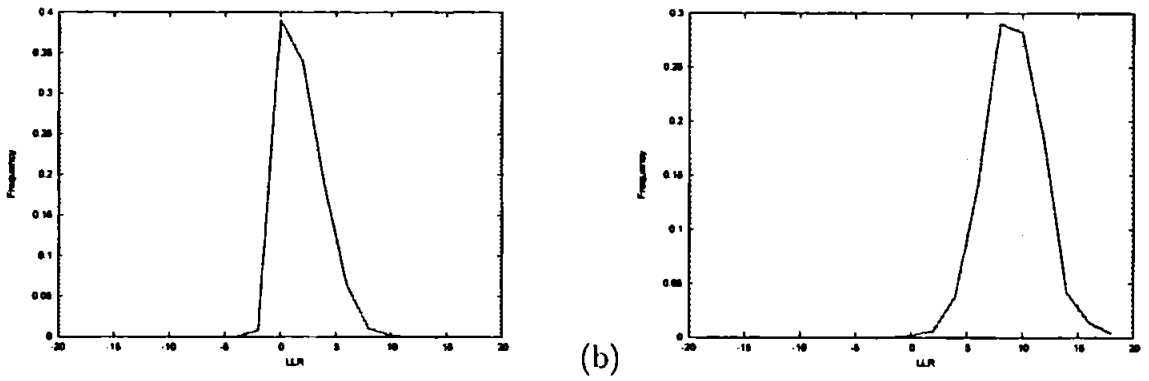


Figure 4.1: PCCC pdf plots of λ at the output of MAP2 for a convergent block at (a) iteration 0 (b) iteration 3

imation. For a non-convergent block though where there is high uncertainty involved in the decisions of the two decoders even after 10 iterations, the evolution is limited and the pdf curve remains at small distance from the zero axis as shown in figure 4.2.
The plots of figures 4.1 and 4.2 have been based on single blocks, just for showing in practise the difference in the evolution of the extrinsic λ densities for blocks that fail to converge. To evaluate the convergence properties of a code it is essential to average over a large number of blocks at each E_b/N_o. To do that, a quantity is needed to be used as a qualitative measure. Clearly, successful evolution of the pdf curves is expressed by a gradual increase of the mean λ value and a subsequent reduction of the pdf spread as the number of performed iterations increments. The qualitative quantity that has been used to evaluate the evolution of the extrinsic λ's is the signal-to-noise ratio SNR_λ which can be thought as the discrimination between the consistent density $pdf(\lambda)$ from $pdf(-\lambda)$. It

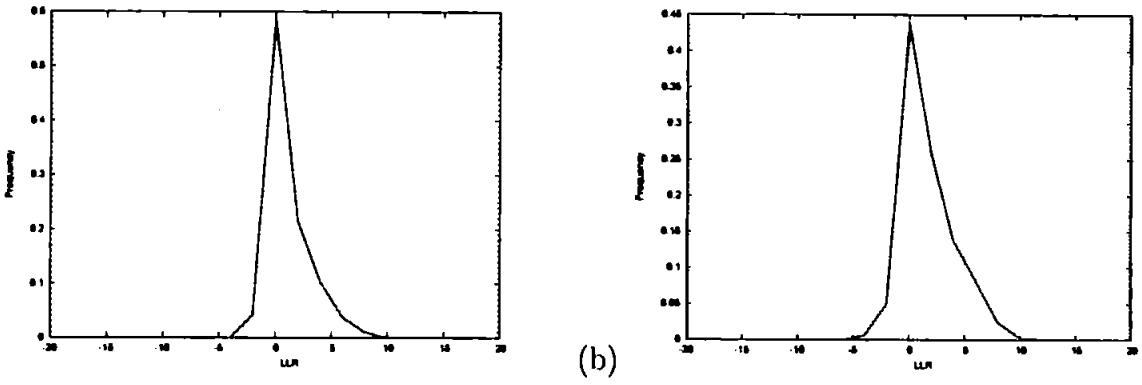


Figure 4.2: PCCC pdf plots of λ at the output of MAP2 for a non-convergent block at (a) iteration 0 (b) iteration 10

is defined as the ratio between the squared value of the mean μ_λ and its variance σ_λ^2.

$$SNR_\lambda = \frac{\mu_\lambda^2}{\sigma_\lambda^2} \tag{4.9}$$

From the consistency assumption of 4.7 it follows that

$$SNR_\lambda = \frac{\mu_\lambda}{2} \tag{4.10}$$

Thus, knowledge of the mean μ_λ is adequate for estimating the quality of the λ values and by tracking the magnitude of SNR_λ throughout the iterative process, it is possible to assess the convergence properties of a code and determine the E_b/N_o threshold point.

4.1.1 Dynamic models of density evolution

There are two ways for modelling the evolution of the λ pdf. The *full-iteration model* treats the iterative decoder as a black box with one input (input to MAP1) and one output (output of MAP2). The ratio of input and output SNR_λ is defined as the noise figure F.

$$F = \frac{SNR1_{\lambda in}}{SNR2_{\lambda out}} \tag{4.11}$$

Clearly, when F is below 1 the turbo decoder improves its estimations at that iteration. Generally, this model concentrates on the SNR_λ change of the overall decoder to assess convergence to any solution without giving any further clues about the quality of this solution. Because of the early correlations created between components of the extrinsic information the decoder might stop producing any more gain and converge before it

reaches the correct solution. As we'll see later, investigation of the evolution of the mutual information of the extrinsic λ values, offers direct feedback about the conditional entropy at each iteration. The same can be achieved by using the *half-iteration* model that keeps track of the SNR_λ at the output of both constituent decoders at the end of an iteration. The turbo concept is based on the fact that each decoder takes advantage of the progress that has been made by the previous stage. This progress can be expressed in terms of the SNR_λ. As it was explained in the previous chapter that dealt with the convergence problem of iterative decoders, the growing correlation between the decisions of the constituent decoders is regarded as the main reason for the lack of progress after a certain instant of the iterative procedure. The use of the output of decoder stageA as input to the decoder stageB and vice versa, allows us to plot the SNR_λ quantities of the two decoders on the same graph.

Figure 4.3 presents the half-iteration model plots of density evolution for two PCCC schemes with different constituent RSC codes, at higher and lower SNR. The distance between the two representative curves for $SNR1_\lambda$ and $SNR2_\lambda$ denotes the SNR_λ *gain* and can be used as a measure for the convergence capability of the code at a certain E_b/N_o. The narrow pass (known as *tunnel*) at the first few iterations is critical for the convergence properties of the decoder. Once the decoder has managed to evolve through the tunnel, the SNR gain increases and convergence can be achieved withing a few iterations. The SNR decreases again as the decoder approaches convergence due to the saturation of the extrinsic probabilities. As expected, at lower SNR the tunnel is narrower (keep in mind the different scaling among the graphs) and the convergence rate lower. Note also from figure 4.3 the difference in the width of the tunnel path when constituent codes of longer constraint length are used. This verifies the better converging properties of weaker codes at low SNR.

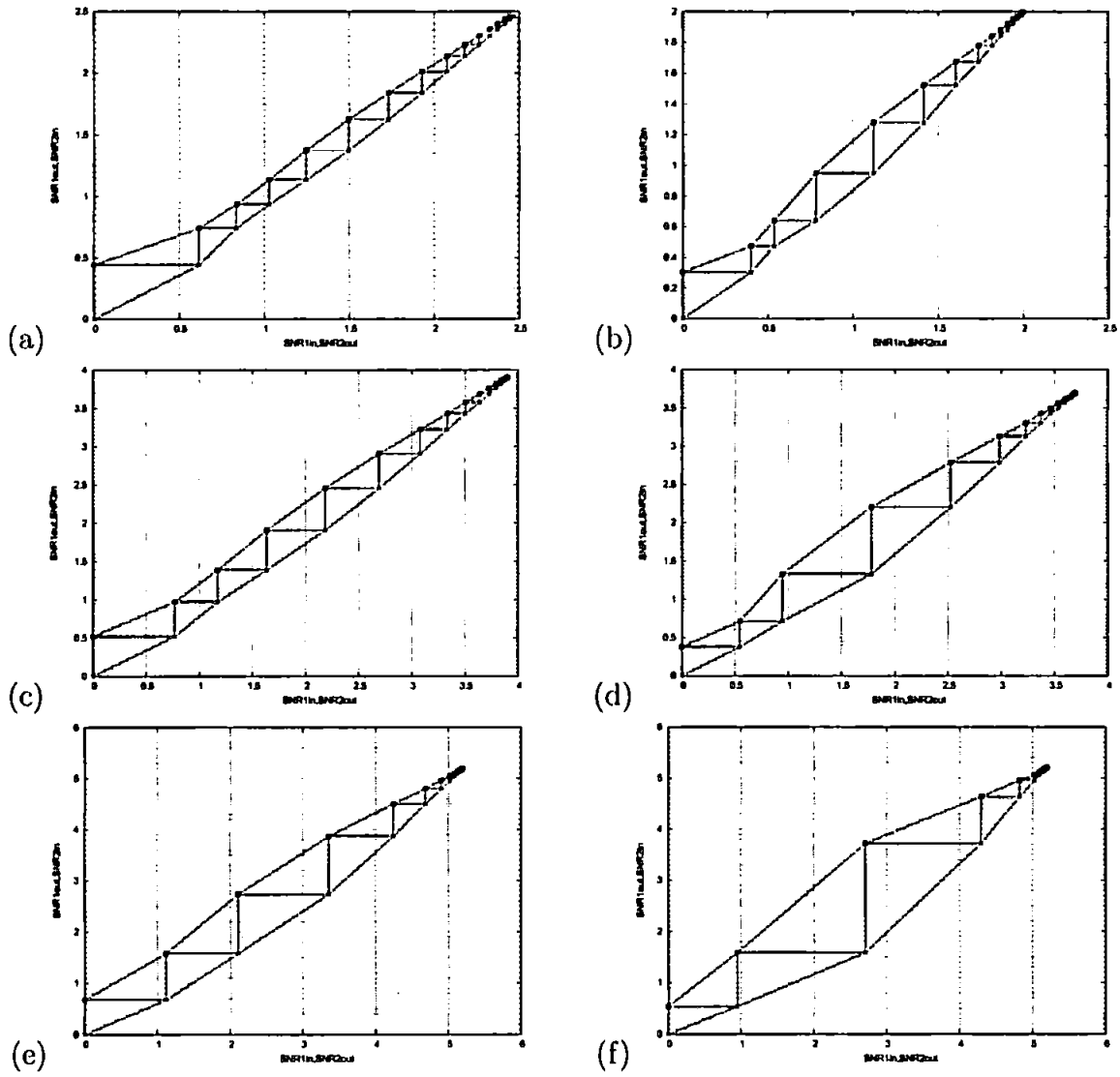


Figure 4.3: Half-iteration density evolution model: (a) PCCC (1,5/7) at 0dB (b)PCCC (1,25/37) at 0dB (c) PCCC (1,5/7) at 0.3dB (d)PCCC (1,25/37) at 0.3dB (e) PCCC (1,5/7) at 0.8dB (f)PCCC (1,25/37) at 0.8dB. For all schemes PCCC (1,5/7) N=1500 and for PCCC (1,25/37) N=1503. The consistency approximation has been used in all cases

4.2 Mutual information evolution: EXIT charts

Similar analysis and conclusions about decoder convergence can be conducted using alternative averages over the λ statistics, such as the *mutual information* between the systematic bits and the extrinsic λ values at the input and output of the two decoding stages.

Assume transmission of symbol a with a priori probability $p(a)$. At the output of the noisy channel the received value b can be used to define the a posteriori probability of a, that is the conditional probability $P(a|b)$. The change between the information uncertainty before (a priori probability) and after reception of b (a posteriori probability) is the gain in information and is called mutual information $I(a;b)$:

$$I(a;b) = \log_2\left(\frac{1}{p(a)}\right) - \log_2\left(\frac{1}{P(a|b)}\right) = \log_2\left(\frac{P(a|b)}{p(a)}\right) \tag{4.12}$$

The mutual information $I(a;b)$ expresses the amount of knowledge learnt about a by reception of b. Multiplying numerator and denominator of the above equation with the a priori probability $p(b)$, we get

$$I(a;b) = \log_2\left(\frac{P(a,b)}{p(a)p(b)}\right) \tag{4.13}$$

From equation 4.13 we can directly point out the essence of mutual information, that is if a and b are independent then

$$P(a|b) = p(a) \tag{4.14}$$

and

$$P(a,b) = p(a)p(b) \tag{4.15}$$

and as a consequence of that

$$I(a;b) = \log_2\left(\frac{p(a)p(b)}{p(a)p(b)}\right) = 0 \tag{4.16}$$

Hence, if a and b are independent, which means that reception of b does not provide any information at all about a, then the mutual information becomes zero.

To obtain the mutual information over an alphabet of more than one symbols, it is essential to average using the appropriate probabilities of the symbols occurring:

$$I(A;b) = \sum_i P(a_i|b)I(a_i;b) = \sum_i P(a_i|b)\log_2\left(\frac{P(a_i|b)}{p(a_i)}\right) \tag{4.17}$$

Similarly

$$I(a;B) = \sum_j P(a|b_j) I(a;b_j) = \sum_j P(a|b_j) \log_2 \left(\frac{P(a|b_j)}{p(a)} \right) \tag{4.18}$$

Finally, the mutual information between two symbol alphabets A and B can be obtained as:

$$I(A;B) = \sum_i \sum_j P(a_i, b_j) I(a_i;b_j) = \sum_i \sum_j P(a_i, b_j) \log_2 \left(\frac{P(a_i, b_j)}{p(a_i)p(b_j)} \right) \tag{4.19}$$

which is equivalent to the uncertainty (entropy) reduction

$$I(A;B) = H(A) - H(A \mid B) \tag{4.20}$$

and for equiprobable transmitted symbols a:

$$I(A;B) = 1 - H(A \mid B) \tag{4.21}$$

The conditional entropy $H(A \mid B)$ represents the uncertainty introduced by the channel which is also known as *equivocation*. The decoder's task is to eliminate any uncertainty from the received vector, i.e. to make the conditional entropy equal to zero (and therefore the mutual information equal to 1). We can track the evolution of the decoder as it attempts to reduce iteration after iteration the conditional entropy and increase the mutual information to the maximum value of 1. The main task of the mutual information evolution theory is to monitor the mutual information between the extrinsic λ values and the systematic bits x at the input and output of both constituent decoders. The Gaussian and consistency assumption of the λ pdf can also be used here for finding the conditional probabilities of the λ values, and from these the $I(B;A)$ and $I(B;E)$ (considering A and E as the set of λ values at the input and the output of the constituent decoders, respectively). Hence, treating λ before and after have been processed by the MAP decoders as being the outputs of a fictitious LLR channel, we can determine the capacity at each iteration. Averaging over many blocks provides the average achievable capacity of the code at a fixed E_b/N_o level. Since I_E is directly affected by I_A (constituent decoder A accepts from decoder B a set of LLRs A with mutual information I_A as a priori information and produces a set of λ probabilities E with I_E which is then used as a priori information from the next decoding stage) it can be expressed as a function of I_A and E_b/N_o.

$$I_E = f(I_A, E_b/N_o) \tag{4.22}$$

and for fixed E_b/N_o

$$I_E = f(I_A) \tag{4.23}$$

I_E is a monotonically increasing function so we can express I_A as

$$I_A = f^{-1}(I_E) \tag{4.24}$$

Figure 4.4 shows the *Extrinsic Information Transfer* (EXIT) [65] charts of two PCCC

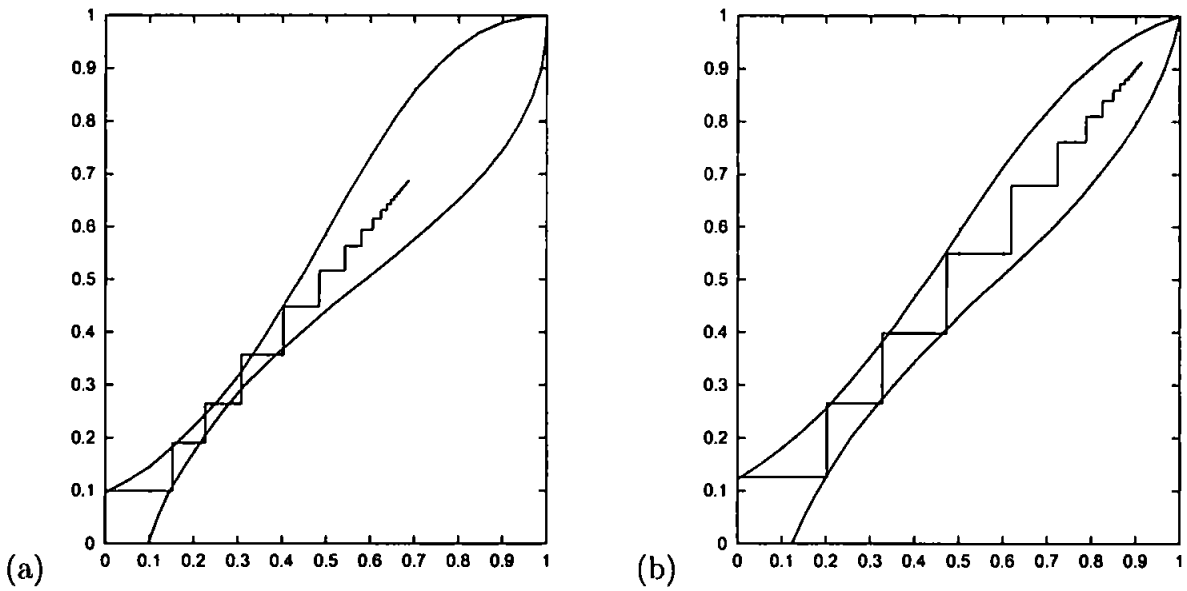


Figure 4.4: EXIT charts for two r=4/5 codes of information length k=4000 operating at 1.6dB and with feed-forward memory of (a) 4 and (b) 2

schemes of rate 4/5 with consistent RSC codes that differ only at the feedback polynomial. The trajectory within the curves represents the gain of the two decoders at each iteration which is analogous to the step widths. As in the density evolution case, the bottleneck in the diagrams is the tunnel region of the code. The opening of the tunnel is analogous to the convergence rate of the code at a given E_b/N_o. The narrower opening of the tunnel at 4.4(a) denotes worse convergence and as a result of that the trajectory stops evolving quite early, before reaching 1. At the point where the two curves are close enough to touch each other the code operates well into the low convergence region. The EXIT charts indicate the better converging properties of the PCCC scheme that uses the RSC codes with the shorter memory feedback polynomials.

5 Attempts to improve the convergence properties of iterative decoders

This chapter discusses three basic attempts for improving the performance of iterative decoders. The first deals with the sensitivity of the iterative decoder performance with respect to modification of the variance. In [62] and [75] the authors have investigated the problem of SNR estimation for turbo decoders and they have concluded that turbo decoders are much more tolerant to over-estimation than under-estimation of SNR. These investigations were mainly concerned with the problem of SNR estimation and the impact of estimation errors to the decoding performance. From the results though it can be observed that tiny reductions in terms of BER and FER are attainable when a small offset is added to the real SNR value. In this chapter the investigation is approached from a different angle, as the variance is modified only for specific bit positions where the quality of the associated channel output is below a certain pre-defined level.

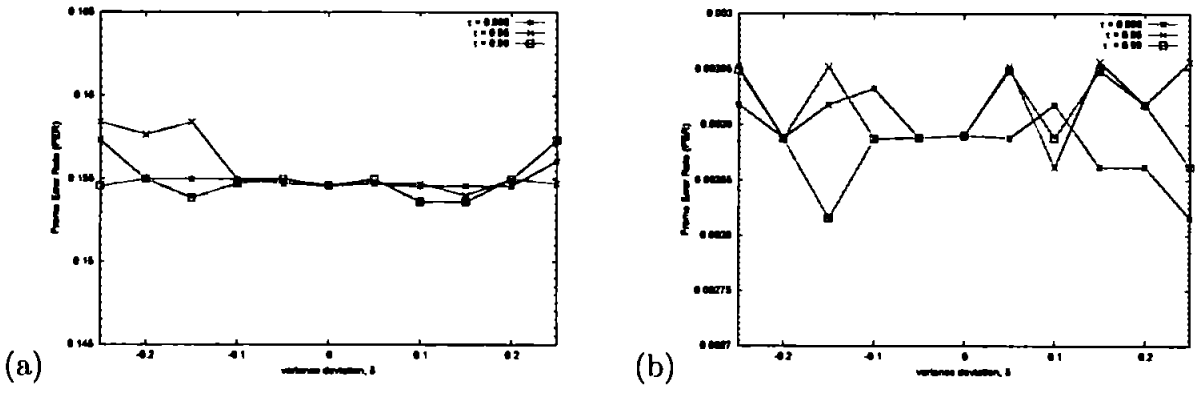
The second attempt has been inspired by the observations of [14] due to which a side effect of the correlations that are created during the iterative process (and up to a high extend can be blamed for the convergence problems) is the fast saturation of the decoders' decisions. In the experiments that will be presented in this chapter the rate of growth of the extrinsic probabilities is limited so that convergence is delayed and, hopefully, the effect of correlation is minimised. A similar scaling operation is briefly referred to the original turbo code paper of Berrou et.al.[9].

The third scheme under consideration deals with the impact of burst errors to the ability of the iterative decoder to converge to the optimum solution. Given that the iterative decoder has failed to converge optimally, the input probabilities are permuted so that either burst error events that persist after interleaving or formed randomly after the interleaving operation, are randomised.

5.1 Modification of the variance

The variance, as this follows from the value of the SNR, can be alternatively thought as a measure of the sensitivity of the MAP decoder to the channel observations. For a certain channel output its associated conditional probability will tend to be more decisive when the variance is low. On the other hand the sensitivity of the conditional probability towards the same channel output reduces when higher variance is used. This property can be used as a way to affect the conditional probabilities of those bits that exhibit high uncertainty. For that reason an *uncertainty threshold* τ is used so that whenever the entropy of the conditional input probability of bit i is higher than the uncertainty threshold τ, the variance used for the calculation of its conditional probability is varied by an amount $\pm\delta\%$ from the original channel variance.

Figures 5.1 and 5.2 display the results obtained from application of this method to PCCC and LDPC schemes at various E_b/N_o, and for several choices of the parameters τ and δ. It has not been possible in neither of the two schemes to achieve some significant reduction on the FER. Any observed improvements are small, sporadic and not consistent with the parameters τ and δ.



(a) (b)

Figure 5.1: Change of FER with respect to τ and δ for a PCCC scheme of block-length N=1500 at (a) 0.5dB and (b) 1.25dB

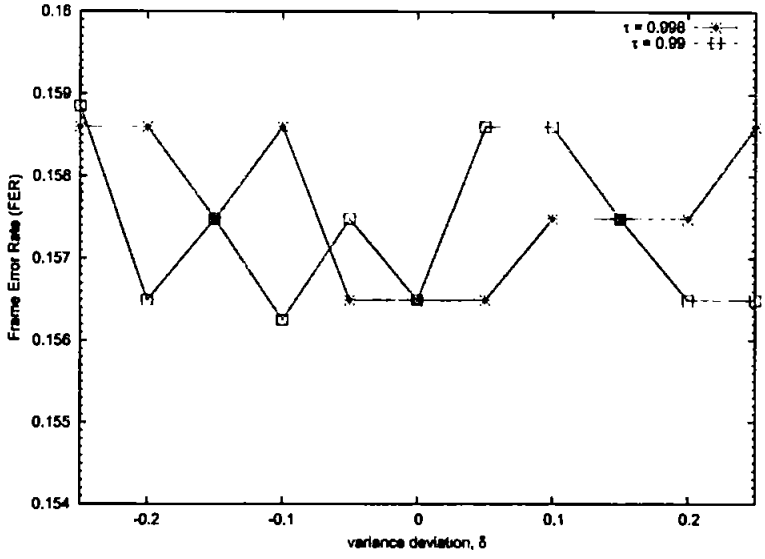


Figure 5.2: Change of FER with respect to τ and δ for a Tanner(155,64) LDPC code at 2.0dB

5.2 Bounding the growth rate of extrinsic probabilities

In this experiment we attempt to minimise the side effects of the induced correlations among the systematic bits by bounding the extrinsic probabilities' maximum rate of growth per iteration. The rate of growth must not exceed the exponential function $exp(G)$, that is

$$\frac{e_i^{t+1}(x)}{e_i^t(x)} < exp(G) \tag{5.1}$$

where $x \in [0,1]$ and $e_i^t(x)$ is the extrinsic probability of x for bit i at the end of the t^{th} iteration. For all investigated cases on PCCC schemes, there has been observed at least one value of the parameter G for which considerable reduction of the iterative decoder error rate has been achieved. Figures 5.3 and 5.4 summarise these results by showing the variation of the FER performance versus G when the bounding method is applied on PCCC schemes. For comparison reasons, the standard turbo performance is indicated by a straight line on the same graphs.

The same method did not deliver similarly good results on LDPC codes. In all tested cases, only tiny improvements were managed, and these only when the bounding method had been applied to the individual (local) extrinsic probabilities. All attempts to apply bounding on the overall extrinsic probabilities (the product of the local extrinsics) at

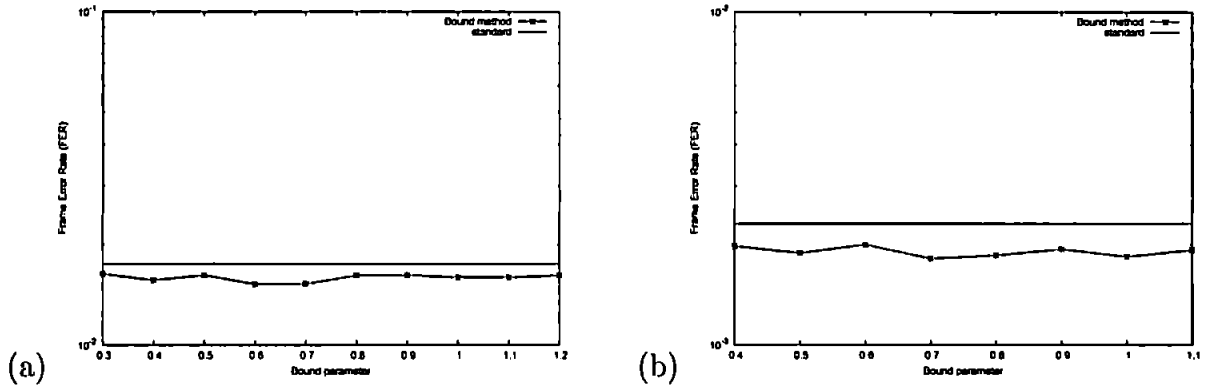


Figure 5.3: FER performance of the bounding method for various values of G for a PCCC(1,13/15) scheme of N=1500 at (a) 0.75 and (b) 1.0dB

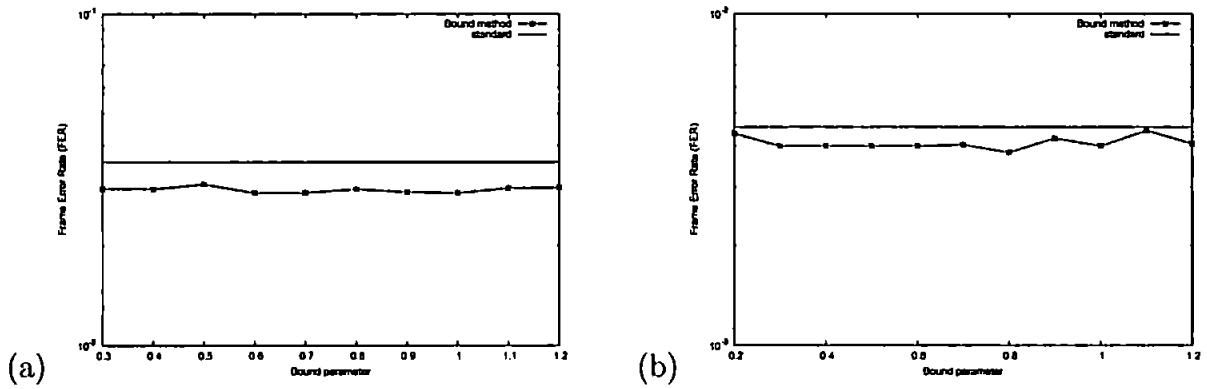


Figure 5.4: FER performance of the bounding method for various values of G for a PCCC(1,25/37) scheme of N=1503 at (a) 0.75 and (b) 1.0dB

the end of each iteration, resulted on error rates higher than these of the standard SP algorithm.

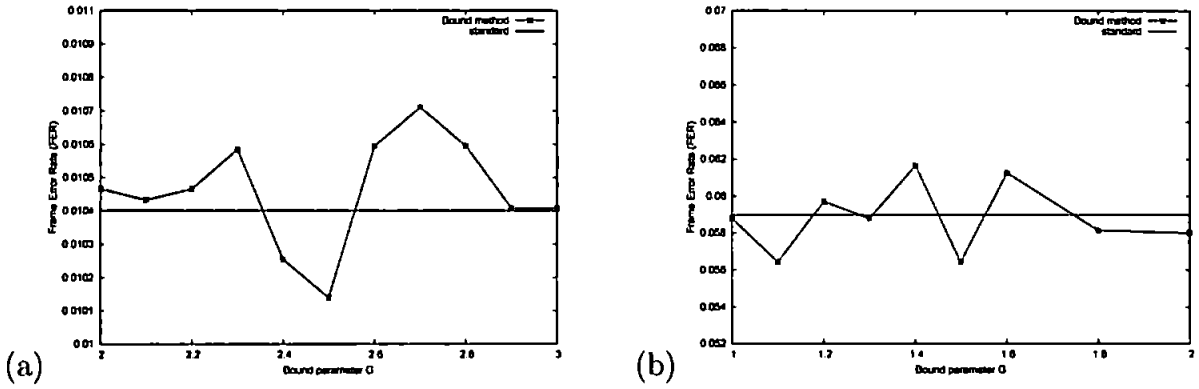


Figure 5.5: FER performance of the bounding method for various values of G when applied on (a) LDPC (105,53) at 3.0dB and (b) LDPC (341,205) at 2.5dB

5.3 Structured permutation of the received vector

The presence of long bursts of errors can significantly influence the error capability of a convolutional code. Its immunity on long bursts is dependant on the memory of the convolutional code, i.e. higher memory codes exhibit higher resistance on burst errors. One of the advantageous properties of the use of interleavers in turbo codes is the randomisation of any bursts of errors that exist at the input of the first decoding stage. However, even after interleaving the possibility that certain bursts persist or that the interleaving action happens to create new error bursts, still exists.

An observation that is implied by the error bursts concept and inspired this experiment, was that the actual Euclidean distance of the received vector from the transmitted can not be considered as a definite parameter to the decoding outcome. In other words, the way that the errors (and consequently the conditional probabilities of the input vector) are mapped within the block is equally important to whether the iterative decoder will successfully converge to the ML solution for a certain block. The question is if, and by how much, the performance of the iterative decoder would be enhanced from a different mapping of the input probabilities. To show the potential of this method, the following experiment has been made.

Experiment description

Consider a non-convergent block. The received vector (the input probabilities) are permuted but in a way that the structure of the code is not affected. In other words, permutations are allowed only among those systematic bits of which the associated transmitted symbol is the same (look at figure 5.6). By this way, the code structure is maintained.

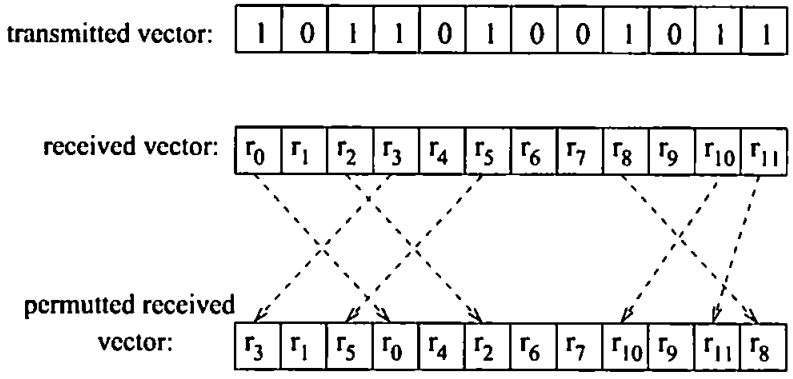


Figure 5.6: Permutations are allowed only among the received values of those positions that are associated with the same transmitted symbol value (1 in this case)

Obviously, in practise the transmitted symbols are unknown. However, in this experiment it is attempted to display the potential of such a method and to prove our previous argument concerning the significance of implementing different mapping at the input vector. Assuming equiprobable transmission and BPSK modulation, more or less half of the systematic bits will be 1's or 0's. For practical block-lengths, trying all possible permutations would be a highly complex task. To account for that, a parameter that sets the maximum number of performed permutations is defined. We call this parameter χ. Figure 5.7 displays the gain that is achieved over standard turbo decoding by applying the structured permutation method for various values of χ. The coding gain is significant even for $\chi = 2$ and increases steadily with higher χ. It should be expected that for higher values of χ the performance will get arbitrarily close to the mrl bound. The obtained results reveal the great potential that is offered by the structured permutation method for the convergence improvement of turbo codes. But how much of this potential gain can be achieved in practise, where the actual transmitted symbols are not known in advance? The proposed algorithm is summarised below:

- *Step 1:* Given a non-convergent block, store the received vector $\mathbf{r}$ and the hard decisions of the standard decoder after last iteration, for all systematic bits as $\mathbf{D} = d_0, d_1, ..., d_{N_{syst}}$
- *Step 2:* Set the parameter χ to the number of desired permutations, j equal to zero and specify a new parameter called *SPAN*
- *Step 3:* Choose the *SPAN* most unreliable among those positions i where $d_i = x$ (where $x \in [0, 1]$ is the transmitted symbol)

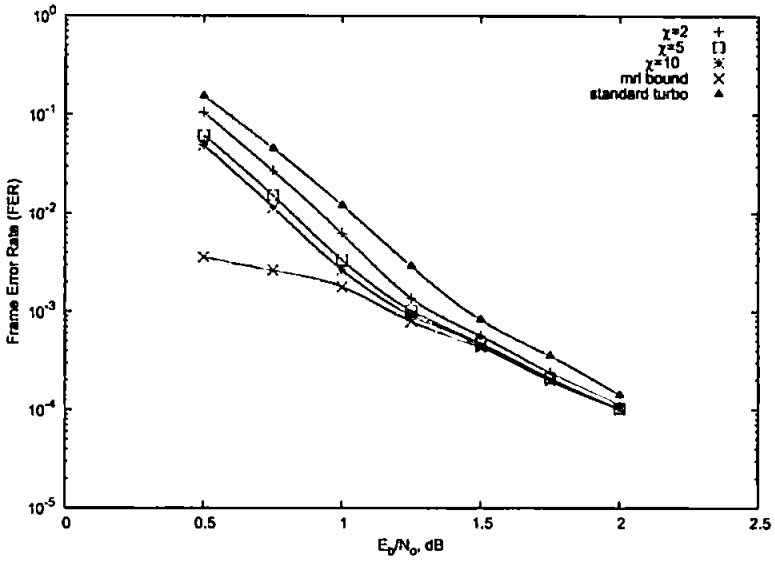


Figure 5.7: Optimum implementation of the structured permutation method on a PCCC (1,5/7) scheme

- *Step 4:* Restart decoding based on the permuted received vector $\mathbf{r}'$
- *Step 5:* Substitute $\mathbf{D}$ with the new decisions and store the new vector. Calculate the Euclidean distance of $\mathbf{D}$ from the original received vector $\mathbf{r}$.
- *Step 6:* Increment j and if $j < \chi$ return to *Step 3.* If $j = \chi$ proceed to the next step
- *Step 7:* Choose the sequence $\mathbf{D}$ that is associated with lower Euclidean distance from $\mathbf{r}$

The size of the permutation has been constrained to $SPAN$ in order to reduce the number of erroneous components d_i that are involved in the permutation process. For the tested PCCC scheme of block-length $N = 1500$ and rate 1/3, the best results were obtain for $SPAN$=100. The comparison to the standard turbo decoding is shown in figure 5.8. The coding gain increases steadily with χ and at moderate to high SNR the structured permutation method offers significant gain.

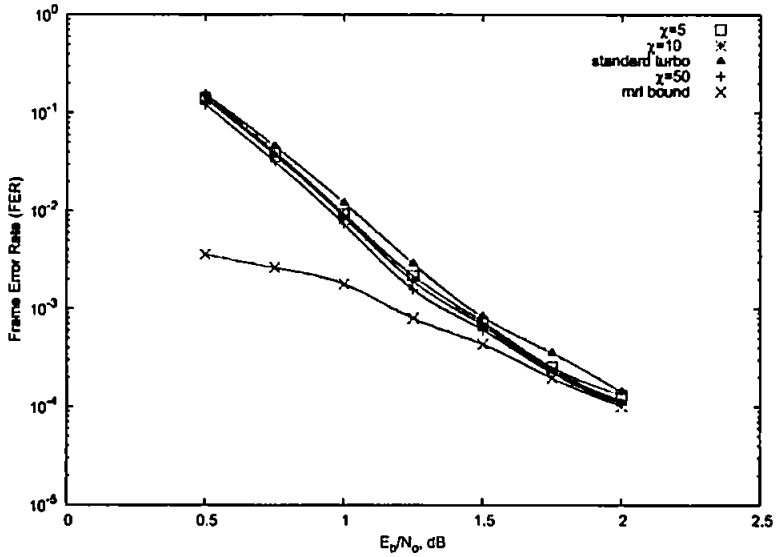


Figure 5.8: Implementation of the structured permutation algorithm on a PCCC (1,5/7) scheme

6 Generation of pseudo-codewords and their role in the outcome of the Sum-Product algorithm

As it is already known, belief propagation is a sub-optimum decoding method when operating on graphs with loops. Optimum maximum likelihood decoding involves complexity that increases exponentially with block-length N, hence for practical lengths optimum decoding is unrealistically complex. In this chapter it is attempted to explain the reasons why the iterative decoder does not always converge to the maximum likelihood solution. The mechanism of SP algorithm is followed analytically through examples on short codes. It is shown that the iterative decoder performs optimum decoding based on a distorted vector space. In literature the distorted vectors are called pseudo-codewords [34, 35, 72] and the same name will be used here. It will be shown further on that the number of pseudo-codewords can be computed by the column and row weight of the parity check matrix.

The influence of pseudo-codewords to the outcome of iterative decoding will be also investigated. The failure of of the SP decoding algorithm to converge to the ML optimum solution will be linked with the correlation among the received vector and the pseudo-codewords.

6.1 Background

An algebraic analysis of iterative decoding has been attempted by Offer et. al. in [41] and by Soljanin et. al. in [61]. The authors show that by the use of the *group algebra product*, SP decoding of very sparse codes and tree codes becomes equivalent to optimum decoding. However, the analysis in any of these approaches does not give an insight into the way that pseudo-codewords are generated during the iterative decoding procedure, and the way that they affect the final decision. Moreover, application of the group algebra product is limited only to a theoretical basis.

The effect of pseudo-codewords is discussed in [27]. The authors use the factor graph to describe the generation of pseudo-codewords and to define the number of these. The

iterative decoder is modelled by a loop-free graph (computation graph) that expands with increasing iterations. For each of the parity check nodes of the graph, as pseudo-codewords are defined all the associated even error sequences. Their number is shown to increase exponentially with iterations.
The examples presented in this chapter will provide a detailed description of pseudo-codewords. It will be also shown that the number of pseudo-codewords for a given code can be obtained by the structure of its parity check matrix and that pseudo-codewords are equal contributors to the decisions of the iterative decoder.

6.2 Generation of pseudo-codewords during iterative decoding

6.2.1 A quick reminder of the Sum-Product (SP) decoding algorithm operations

The SP algorithm was described analytically in chapter 2. This section acts as a quick reminder of the key points so that there is a smooth transition to the ideas of this chapter.
The SP algorithm acts locally in an equation-by-equation basis. Each of the $N-k$ parity check equations forms a single parity check code (we will call them sub-codes). Messages are exchanged between the sub-codes about their independent estimations of their common participants probabilities. The procedure is repeated in an iterative manner until convergence to a predefined stopping criterion has been reached (or until the maximum number of iterations has been performed). Given a parity check matrix $\mathbf{H} = \{h_{ji}\}_{(N-k)\times N}$, Gallagher's "product of differences" [28] operation offers the basic expression for the probability of an equation being satisfied.

$$\prod_{\forall i: i \in S_j} (p_i(0|r_i) - p_i(1|r_i)) \tag{6.1}$$

Where S_j, the set of variable nodes connected to check node (parity check equation) j (i.e. the set of positions with 1 in the particular row), and r_i the i^{th} coordinate of the received vector $\mathbf{r}$. By expanding equation 6.1 it can be shown that it expresses the difference of the joint probability of all the odd sequences from the joint probability of all the even sequences (i.e. the joint probability difference between these sequences that satisfy the parity check and those that do not) within the local group S_j.

The extrinsic information of bit i at equation j can be calculated as a sum of products that includes all even or odd combinations (depending on the wanted extrinsic probability for 0 or 1 respectively) that exist within the subgroup S_j, excluding though the particular bit the extrinsic probability of which is to be found. Since this is evaluated locally (only for equation j) it will be called local extrinsic $e_{l_{ji}}$.

$$e_{l_{ji}}(1) = \mathcal{P}_{odd_{S_j/i}} \tag{6.2}$$

$$e_{l_{ji}}(0) = \mathcal{P}_{even_{S_j/i}} \tag{6.3}$$

The overall extrinsic information $e_i(x)$ where $x \in [0, 1]$ will be the normalised product of all local extrinsic probabilities for bit i.

$$e_i(x) = \prod_{\forall i: h_{ji}=1} e_{l_{ji}}(x) \tag{6.4}$$

At the end of an iteration, the associated APP decisions for any bit i can be obtained by the product

$$APP_i(x) = p(x|r_i) \cdot e_i(x) \tag{6.5}$$

The original channel probabilities are then updated to be used as a priori for the next iteration.

$$\forall i : h_{ji} = 1,\ upd_{ji}(x) = \frac{APP_i(x)}{e_{l_{ji}}(x)} \tag{6.6}$$

The following sections investigate the reasons for the sub-optimality of the SP algorithm with respect to optimum MAP decoding.

6.2.2 Vector space of iterative decoder

Consider a parity check matrix $\mathbf{H} = \{h_{ji}\}_{(N-k)\times N}$. We denote as w_{r_j} the row weight of parity check j and as w_{c_i} the column weight of position i. The 2^k information sequences are mapped into the 2^k codewords $\mathbf{c}$ that form the code $\mathcal{C}$. Assuming MAP decoding the decision for the transmitted symbol x at position i will be based on the vector set $\mathcal{C}$, by summing up the probabilities of those codewords where $c_i = x$.

$$APP_i(x) = \sum_{\mathbf{c}\in\mathcal{C}, c_i=x} P(\mathbf{c}|\mathbf{r}) \tag{6.7}$$

Equivalently, the extrinsic information for bit i is expressed by the same equation but with the effect of bit i excluded.

$$e_i(x) = \sum_{\mathbf{c}\in\mathcal{C}, c_i=x} \frac{P(\mathbf{c}|\mathbf{r})}{p(x|r_i)} \tag{6.8}$$

Considering now an iterative decoder, for each parity check equation j the extrinsic probabilities for the participating bits are calculated locally by equations 6.2 and 6.3. The $\mathcal{P}_{odd}$ and $\mathcal{P}_{even}$ normalised probabilities are the sum of the total $2^{w_{r_j}}$ odd and even sequence probabilities that exist at parity check equation j. For calculation of the overall extrinsic information (equation 6.8) the number of terms increases exponentially due to w_{c_i}. Hence:

$$e_i(1) = \prod_{\forall i:h_{ji}=1} e_{l_{ji}}(1) = \prod_{\forall i:h_{ji}=1} \mathcal{P}_{odd_{S_{j/i}}} = \prod_{\forall i:h_{ji}=1} \sum_{v=0}^{v=2^{w_{r_j}-2}} p_{o_{j,v}} \tag{6.9}$$

and

$$e_i(0) = \prod_{\forall i:h_{ji}=1} e_{l_{ji}}(0) = \prod_{\forall i:h_{ji}=1} \mathcal{P}_{even_{S_{j/i}}} = \prod_{\forall i:h_{ji}=1} \sum_{v=0}^{v=2^{w_{r_j}-2}} p_{e_{j,v}} \tag{6.10}$$

where $p_{o_{j,v}}$ and $p_{e_{j,v}}$ represent the probability of the v^{th} even and odd respectively, sequence of parity check equation j. So, after processing all the $(N-k)$ parity check equations, the extrinsic probability of bit i will be based on $\Psi(i)$ terms. Where $\Psi(i)$ is a function of the $\mathbf{H}$ matrix column weight at position i and the row weight at those parity checks where bit i participates.

$$\Psi(i) = 2 \cdot \left(\prod_{\forall i:h_{ji}=1} 2^{(w_{r_j}-2)} \right) \tag{6.11}$$

The $\Psi(i)$ vectors $\mathbf{u}$ form the vector space $\mathcal{U}_i \subset \mathcal{U}$ based on which the decoder will decide for bit i.

Two distortion effects appear and can be summarised as:

- If $\Psi(i) > 2^k$, vectors which are not part of the codebook $\mathcal{C}$ will be involved to the decision in the form of an *offset* [3].
- The presence of cycles in the parity check matrix $\mathbf{H}$ will cause multiplication of the involved probability terms by themselves. In literature this is referred as unequal scaling distortion [27] or double counting [77] problem.

The following examples will demonstrate the distortion that is introduced by SP algorithm on various short codes.

The effect of short loops: Example on a Hamming (7,4) code

$$\begin{array}{c} 0\ 0\ 1\ 0\ 1\ 1\ 1 \\ 0\ 1\ 0\ 1\ 0\ 1\ 1 \\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \end{array}$$

Consider the first column as position 0. Bits 3 and 6 form a cycle of length 4. The analytical expressions for the local extrinsic information for bit 3 from parity checks 1 and 2 are

$$\begin{aligned} \textit{parity check 1: } e_{l_{13}}(1) = &[p_1 \cdot q_5 \cdot q_6] + [q_1 \cdot p_5 \cdot q_6] + \\ &[q_1 \cdot q_5 \cdot p_6] + [p_1 \cdot p_5 \cdot p_6] \end{aligned} \tag{6.12}$$

$$\begin{aligned} \textit{parity check 2: } e_{l_{23}}(1) = &[p_0 \cdot q_4 \cdot q_6] + [q_0 \cdot p_4 \cdot q_6] + \\ &[q_0 \cdot q_4 \cdot p_6] + [p_0 \cdot p_4 \cdot p_6] \end{aligned} \tag{6.13}$$

Where $p_j = p(1|r_j)$ and $q_j = 1 - p_j = p(0|r_j)$ for $j = 0, 1, ..., 6$. The overall extrinsic for bit 3 would be the product of the two local extrinsics of equations 6.12 and 6.13.

$$\begin{aligned} e_3(1) = &[p_0 \cdot p_1 \cdot q_4 \cdot q_5 \cdot q_6^2] + [q_0 \cdot p_1 \cdot p_4 \cdot q_5 \cdot q_6^2] + [q_0 \cdot p_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + \\ &[p_0 \cdot p_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + [p_0 \cdot q_1 \cdot q_4 \cdot p_5 \cdot q_6^2] + [q_0 \cdot q_1 \cdot p_4 \cdot p_5 \cdot q_6^2] + \\ &[q_0 \cdot q_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [p_0 \cdot q_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] + [p_0 \cdot q_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + \\ &[q_0 \cdot q_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + [q_0 \cdot q_1 \cdot q_4 \cdot q_5 \cdot p_6^2] + [p_0 \cdot q_1 \cdot p_4 \cdot q_5 \cdot p_6^2] + \\ &[p_0 \cdot p_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [q_0 \cdot p_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] + [q_0 \cdot p_1 \cdot q_4 \cdot p_5 \cdot p_6^2] + \\ &[p_0 \cdot p_1 \cdot p_4 \cdot p_5 \cdot p_6^2] \end{aligned} \tag{6.14}$$

From equation 6.14 the following types of distortion are observed:

- The vector space $\mathcal{U}_3$ is formed by 16 terms, 8 more than the codewords that should contribute to the decision for $e_3(1)$.
- None of the terms of equation 6.14 represents a valid codeword $\mathbf{c} \in \mathcal{C}$. Bit 6 which participates with bit 3 in a cycle of length 4, appears scaled (it is double counted).

The decision for the extrinsic probability of bit 3 at iteration 0 will be based on the vector set $\mathcal{U}_3 \neq \mathcal{C}$. In contrast to that, optimum decoding would only involve the 2^{k-1}

codewords for which $c_{j3} = 1$.

$$\begin{aligned} e_{3,ml}(1) = &\underbrace{[p_0 \cdot p_1 \cdot q_2 \cdot q_4 \cdot q_5 \cdot q_6]}_{ML[1]} + \underbrace{[q_0 \cdot p_1 \cdot p_2 \cdot p_4 \cdot q_5 \cdot q_6]}_{ML[2]} + \underbrace{[p_0 \cdot q_1 \cdot p_2 \cdot q_4 \cdot p_5 \cdot q_6]}_{ML3]} + \\ &\underbrace{[q_0 \cdot q_1 \cdot q_2 \cdot p_4 \cdot p_5 \cdot q_6]}_{ML[4]} + \underbrace{[q_0 \cdot q_1 \cdot p_2 \cdot q_4 \cdot q_5 \cdot p_6]}_{ML[5]} + \underbrace{[p_0 \cdot q_1 \cdot q_2 \cdot p_4 \cdot q_5 \cdot p_6]}_{ML[6]} + \\ &\underbrace{[q_0 \cdot p_1 \cdot q_2 \cdot q_4 \cdot p_5 \cdot p_6]}_{ML[7]} + \underbrace{[p_0 \cdot p_1 \cdot p_2 \cdot p_4 \cdot p_5 \cdot p_6]}_{ML[8]} \end{aligned} \tag{6.15}$$

Having the solution for e_3 obtained by both the optimum brute force decoder and the sub-optimum iterative belief propagation decoder, we can compare the two and identify the way that they differ. A careful look at the two equations shows that the iterative algorithm induces a scaling part and an offset to the actual ML solution for bit 3.

$$\begin{aligned} e_{it_3}(1) = &\frac{q_6}{q_2} \cdot (ML[1] + ML[4]) + \frac{p_6}{q_2} \cdot (ML[6] + ML[7]) + \\ &\frac{q_6}{p_2} \cdot (ML[2] + ML[3]) + \frac{p_6}{p_2} \cdot (ML[5] + ML[8]) + \ offset \end{aligned} \tag{6.16}$$

The offset includes all the remaining terms, of which none is a codeword.

$$\begin{aligned} offset = &[q_0 \cdot p_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + [p_0 \cdot p_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + \\ &[q_0 \cdot q_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [p_0 \cdot q_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] + \\ &[p_0 \cdot q_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + [q_0 \cdot q_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + \\ &[p_0 \cdot p_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [q_0 \cdot p_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] \end{aligned} \tag{6.17}$$

In [27] the non uniform scaling has been represented in the 3-D space as a skewness in the decision spheres that distort the optimal MAP decision bounds and lead the iterative decoder to errors. In practise, the distortion of the valid codewords and the addition of non-codeword sequences means that terms which are outside the code restrictions contribute to the decision of the iterative decoder. Their contribution is analogous to their correlation with the received vector as it will be discussed later on.

While the effect of the shortest loops (length 4) is becoming apparent even from the beginning of iteration 0, the symptoms from the existence of longer loops within the code structure become noticeable only after the beginning of iteration 1. As an example, consider the case of a (7,3) code derived from cyclotomic cosets [68]. The **H** matrix of

the code is shown below.

$$
\begin{array}{ccccccc}
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}
$$

From its structure we can deduce the following.

- The H matrix is regular, so $w_{r_j} = w_{r_m} \forall m, j$. Moreover, $w_{c_m} = w_{c_i} \forall i, m$ and $w_{r_j} = w_{c_j} \forall j$. That effectively means that for the decision of any bit i the vector space $\mathcal{U}_i$ involves $\Psi(i) = 2 \cdot \left(\prod_{i:\forall h_{ji}=1} 2^{(w_{r_j}-2)} \right)$ terms.
- $\Psi(i) = 16 > 2^k$, $\forall i$. Thus, there will be $16 - 2^3 = 8$ additional terms involved in the decision of any bit i.
- Every bit i neighbours (exists in the same parity check equation) with every other bit exactly once. This is a perfect difference set cyclic code [73].
  - Absence of any cycles of length 4.

The analysis will be based on the extrinsic information of bit 0. Its local extrinsic probabilities $e_{l_{j0}}$ at parity check equations 0, 4 and 6 where bit 0 participates, are calculated as

Equation 0:

$$e_{l_{00}}(1) = [p_1 \cdot q_3] + [q_1 \cdot p_3] \tag{6.18}$$

$$e_{l_{00}}(0) = [q_1 \cdot q_3] + [p_1 \cdot p_3] \tag{6.19}$$

Equation 4:

$$e_{l_{40}}(1) = [p_4 \cdot q_5] + [q_4 \cdot p_5] \tag{6.20}$$

$$e_{l_{40}}(0) = [q_4 \cdot q_5] + [p_4 \cdot p_5] \tag{6.21}$$

Equation 6:

$$e_{l_{60}}(1) = [q_2 \cdot p_6] + [p_2 \cdot q_6] \tag{6.22}$$

$$e_{l_{60}}(0) = [q_2 \cdot q_6] + [p_2 \cdot p_6] \tag{6.23}$$

Where $p_j = p(1|r_j)$ and $q_j = 1 - p_j = p(0|r_j)$. The overall extrinsic probability of bit 0 at iteration 0 is given by the product of the local extrinsic expressions. For convenience,

only $e_0(1)$ is shown below.

$$\begin{aligned} e_0(1) = & \underbrace{[p_1 \cdot q_2 \cdot q_3 \cdot p_4 \cdot q_5 \cdot p_6]}_{ML[1]} + [p_1 \cdot p_2 \cdot q_3 \cdot p_4 \cdot q_5 \cdot q_6] + \\ & [p_1 \cdot q_2 \cdot q_3 \cdot q_4 \cdot p_5 \cdot p_6] + \underbrace{[p_1 \cdot p_2 \cdot q_3 \cdot q_4 \cdot p_5 \cdot q_6]}_{ML[2]} + \\ & [q_1 \cdot q_2 \cdot p_3 \cdot p_4 \cdot q_5 \cdot p_6] + \underbrace{[q_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot q_5 \cdot q_6]}_{ML[3]} + \\ & \underbrace{[q_1 \cdot q_2 \cdot p_3 \cdot q_4 \cdot p_5 \cdot p_6]}_{ML[4]} + [q_1 \cdot p_2 \cdot p_3 \cdot q_4 \cdot p_5 \cdot q_6] \end{aligned} \tag{6.24}$$

The output of the iterative decoder at iteration 0 is the optimum MAP expression plus an offset which includes the remaining $\Psi(0) - 2^{k-1}$ products that have emerged from the multiplication of local extrinsics. Therefore, due to the absence of cycles of length 4 in the structure of the code, no scaling problems exist at iteration 0. From the uniform structure of the parity check matrix it follows that analogous results can be derived for $e_0(0)$ and for every $e_i(x)$. Thus, for the (7,3) code, the vector space $\mathcal{U}_i$ for every bit i at the end of iteration 0 is

$$\mathcal{U}_i = \mathcal{C} \bigcup \mathcal{O}_i \tag{6.25}$$

where $\mathcal{O}_i \subset \mathcal{U}_i$ is the offset vector set, and in this case

$$\mathcal{O}_0 = \left\{ \underbrace{1110100, 1100011, 1001101, 1011010}_{\textit{associated with bit 0 decision equal to 1}}, \underbrace{0010001, 0000110, 0101000, 0111111}_{\textit{associated with bit 0 decision equal to 0}} \right\} \tag{6.26}$$

The set of vectors $\mathbf{o} \in \mathcal{O}$ do not represent valid codewords but pseudo-codewords which are equal contributors (subject to their correlation with the received vector) to the a posteriori decisions of the iterative decoder. The APP decision of the iterative decoder for any bit i at iteration 0 is in this case expressed by

$$APP_i(x) = \sum_{\mathbf{u} \in \mathcal{U}_i, u_i = x} p(\mathbf{u}|\mathbf{r}) = \sum_{\mathbf{c} \in \mathcal{C}, c_i = x} p(\mathbf{c}|\mathbf{r}) + \sum_{\mathbf{o} \in \mathcal{O}_i, o_i = x} p(\mathbf{o}|\mathbf{r}) \tag{6.27}$$

Equation 6.27 is the MAP equation over the vector space $\mathcal{U}_i$ and demonstrates that the iterative decoder operates optimally but on a vector space which is not exclusively formed by codewords $\mathbf{c} \in \mathcal{C}$.

At the end of iteration 0 the input probabilities $p(x|r)$ are updated according to equa-

tion 6.6. At iteration 1 it is the updated channel probabilities that will be used for the computation of the new extrinsic probabilities. As a representative example, the local extrinsic equations of 6.18 and 6.19 would look as

Equation 0:

$$e'_{l_{00}}(1) = \left[\frac{e_0(1)\cdot p_1}{e_{l_{01}}(1)} \cdot \frac{e_3(0)\cdot q_3}{e_{l_{03}}(0)}\right] + \left[\frac{e_1(0)\cdot q_1}{e_{l_{01}}(0)} \cdot \frac{e_3(1)\cdot p_3}{e_{l_{03}}(1)}\right] \tag{6.28}$$

$$e'_{l_{00}}(0) = \left[\frac{e_0(0)\cdot q_1}{e_{l_{01}}(0)} \cdot \frac{e_3(0)\cdot q_3}{e_{l_{03}}(0)}\right] + \left[\frac{e_1(1)\cdot p_1}{e_{l_{01}}(1)} \cdot \frac{e_3(1)\cdot p_3}{e_{l_{03}}(1)}\right] \tag{6.29}$$

Where $e'_{l_{00}}(x)$ are the new local extrinsics at iteration 1. By substituting $e_0(x)$ with the analytical expression of the overall extrinsic probability for bit 0 (equation 6.24) it is easy to observe the existence of duplicated terms and therefore, the presence of scaling problems. It can be concluded that scaling problems due to duplicated terms appear even on these codes that do not contain any cycles of length 4 in their structure. When short cycles exist though, the scaling effect is apparent at the very beginning of the iterative algorithm (iteration 0), that is before the first feedback of information.

Codes with no loops (tree codes)

It was shown how the loops in the structure of a code contribute to the distortion of the iterative decoder vector space $\mathcal{U}$. But does this mean that the iterative decoder will operate optimally when applied on a code with no loops? Table 6.1 summarises the FER performance obtained by application of the SP algorithm and optimal MAP decoding for the loop-free (8,4) code (its **H** matrix is shown below).

$$\begin{matrix} 1\,1\,0\,0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,0\,1\,0\,0 \\ 0\,0\,1\,1\,0\,0\,1\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1 \end{matrix}$$

Despite the absence of loops, there is still a slight deviation (very small though) from the optimal results. Using the same analysis as in the previous examples, the small deviation is performance can be explained. Considering bit 0, the local extrinsic information expressions from parity check equations 0 and 3 will be

Eb/No[dB]	SP FER	optimum MAP FER
2.0	0.090161	0.08965
2.5	0.070358	0.07
3.0	0.052056	0.05192
3.5	0.035682	0.035601
4.0	0.023855	0.023774
4.5	0.015573	0.015558
5.0	0.009829	0.0098079

Table 6.1: FER performance of the (8,4) tree code when decoded with the SP algorithm and optimally

Equation 0:

$$e_{l_{00}}(0) = [q_1 \cdot q_4] + [p_1 \cdot p_4] \tag{6.30}$$

$$e_{l_{00}}(1) = [p_1 \cdot q_4] + [q_1 \cdot p_4] \tag{6.31}$$

Equation 3:

$$e_{l_{30}}(0) = q_7 \tag{6.32}$$

$$e_{l_{30}}(1) = p_7 \tag{6.33}$$

So the overall extrinsics for bit 0 is given by

$$e_0(0) = [q_1 \cdot q_4 \cdot q_7] + [q_0 \cdot p_1 \cdot p_4 \cdot q_7] \tag{6.34}$$

$$e_0(1) = [p_1 \cdot q_4 \cdot p_7] + [p_0 \cdot q_1 \cdot p_4 \cdot p_7] \tag{6.35}$$

The overall extrinsics for bits 1 and 4 at the end of iteration 0 can be obtained in the same manner

$$\begin{aligned} e_1(0) = &[q_0 \cdot q_2 \cdot q_4 \cdot q_5] + [q_0 \cdot p_2 \cdot q_4 \cdot p_5] + \\ &[p_0 \cdot q_2 \cdot p_4 \cdot q_5] + [p_0 \cdot p_2 \cdot p_4 \cdot p_5] \end{aligned} \tag{6.36}$$

$$\begin{aligned} e_1(1) = &[p_0 \cdot p_2 \cdot q_4 \cdot q_5] + [p_0 \cdot q_2 \cdot q_4 \cdot p_5] + \\ &[q_0 \cdot p_2 \cdot p_4 \cdot q_5] + [q_0 \cdot q_2 \cdot p_4 \cdot p_5] \end{aligned} \tag{6.37}$$

$$e_4(0) = [q_0 \cdot q_1] + [p_0 \cdot p_1] \tag{6.38}$$

$$e_4(1) = [p_0 \cdot q_1] + [q_0 \cdot p_1] \tag{6.39}$$

At the next iteration all channel probabilities are updated and the new local extrinsics will be computed by the updated set of probabilities. From the previous examples on codes with loops, it was noticed that the problems of longer loops (longer than 4) become apparent only after iteration 0, in the form of unequal (non-uniform) scalings. In this case where no loops are applicable, the expressions for the new local extrinsics $e'_{l_{ji}}$ of bit 0 at iteration 1 would be

Equation 0:

$$e'_{l_{00}}(0) = \left[\frac{e_1(0) \cdot q_1}{e_{l_{01}}(0)} \cdot \frac{e_4(0) \cdot q_4}{e_{l_{04}}(0)}\right] + \left[\frac{e_1(1) \cdot p_1}{e_{l_{01}}(1)} \cdot \frac{e_4(1) \cdot p_4}{e_{l_{04}}(1)}\right] \tag{6.40}$$

$$e'_{l_{00}}(0) = \left[\frac{e_1(1) \cdot p_1}{e_{l_{01}}(1)} \cdot \frac{e_4(0) \cdot q_4}{e_{l_{04}}(0)}\right] + \left[\frac{e_1(0) \cdot q_1}{e_{l_{01}}(0)} \cdot \frac{e_4(1) \cdot p_4}{e_{l_{04}}(1)}\right] \tag{6.41}$$

The local expression for parity check equation 3 will remain the same as in equation 6.33 since bit 7 participates in only one equation and it is not updated. From equation 6.40 we get

$$e'_{l_{00}}(0) = \left[\frac{e_1(0) \cdot q_1}{e_{l_{01}}(0)} \cdot q_4\right] + \left[\frac{e_1(1) \cdot p_1}{e_{l_{01}}(1)} \cdot p_4\right] \tag{6.42}$$

Since bit 4 participates only in parity check equation 0, $e_{l_{04}}(0) = e'_{l_{04}}(0)$ and $e_{l_{04}}(1) = e'_{l_{04}}(1)$. By substituting equations 6.34 and 6.35 into 6.42 we obtain the analytical expression for the local extrinsic information of bit 0 at iteration 1. Obviously, unequal scaling will occur since the updating equations of 6.34 and 6.35 both include q_4 or p_4. Thus, even for codes without loops there will be some minimum distortion in the vector space of the iterative decoder. That distortion accounts for the slight difference between the iterative and the optimum FER performance.

In [61, 41] the authors claim optimal iterative performance for very sparse and loop-free codes when the group algebra product $\otimes$ rule is applied:

$$l_i \otimes l_j = \begin{cases} l_i \cdot l_j & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{6.43}$$

Where $l_i = \frac{q_i}{p_i}$ the log-likelihood ratio for bit i. By using the $\otimes$ rule on equation 6.42, any double counting is eliminated ($p_i \cdot p_i = q_i \cdot q_i = 1$) and the iterative decoder becomes identical to the the optimal MAP. Practical use of the group algebra product $\otimes$ is unrealisable though, because it implies knowledge of the exact analytical expressions for the extrinsic information of each bit. Such an approach would involve complexity equivalent to straight forward implementation of optimal MAP decoding.

Modelling the iterative decoder at later iterations

At iteration 0 it is relatively easy to obtain the analytical probabilities expressions when the block-length is short. After iteration 0 though we have an explosion in the number of terms, which makes such an analysis very difficult even for very short codes. As an example consider the local extrinsic expressions of 6.28 and 6.29 for parity check 0. Each of the overall extrinsics that update the channel probabilities involve 16 terms (the 8 terms that are associated with $e_0(1)$ are listed in equation 6.24) which effectively means that each of the local extrinsics at iteration 1 will contain $2 \cdot 16^2 = 512$ terms. Considering all 3 parity checks where bit 0 participates, the number of terms explodes to much higher values. Modelling the iterative decoder purely based on the conditional channel probabilities $p_j(x|r_j)$, becomes extremely complex after iteration 0. Simulations can easily verify that the output of the iterative decoder at iteration t is equivalent to MAP decoding (with respect to the received vector $\mathbf{r}$) on the complete set of vectors $\mathbf{u}$ that have emerged at that iteration. Alternatively, the same outcome can be obtained by applying MAP decoding to the exact set of Ψ vectors that have emerged during iteration 0, but with respect to the updated set of probabilities. The advantage of this approach is that the size of the vector space $\mathcal{U}$ remains constant at all iterations. So at iteration t the APP for bit i is calculated as

$$APP_i(x) = \sum_{\mathbf{u} \in \mathcal{U}_i, u_i = x} p(\mathbf{u}|\mathbf{A}^{(t)}) \tag{6.44}$$

Where $\mathbf{A}^{(t)}$ is the updated input vector at iteration t. When $t = 0$, $\mathbf{A}^{(t)} = \mathbf{r}$ and for $t > 0$ $\alpha_i^{(t)} = \frac{\ln(upd_{ji}(x)) \cdot \sigma}{\sqrt{2}} + x$, $\forall i : h_{ji} = 1$.

6.2.3 Comparison with the computation graph based pseudo-space analysis

In [27] the authors end up with a similar observation, that the iterative decoder performs optimum decoding in the pseudo-code defined by the computational tree of the code's factor graph. However, defining the pseudo-code based on the computational tree turns out to be an inefficient approach since a large amount of the defined pseudo-codewords can be considered redundant. Figure 6.1 shows the factor graph and computation tree after three iterations for a (3,2) code with parity check matrix H=111;111. The pseudo-

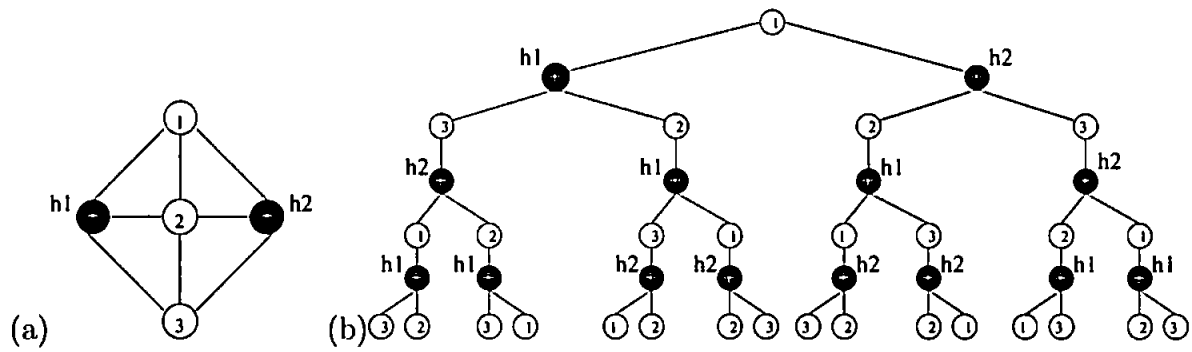


Figure 6.1: a) Factor graph and b) computation tree after three iterations for the (3,2) code

codewords are obtained by flipping the values of any pair of leaf bits that are connected to the same check in the computation tree. Following this approach the number of spurious pseudo-codewords (by spurious the authors define these pseudo-codewords that are not associated with codewords) turns out to be 32,764 after just three iterations. From these, the authors claim that only 480 are considered to occupy unique positions in signal space and play a decisive role in the decision of the iterative decoder for bit 1 (the root bit in the computation tree).

Applying the previously presented analysis on the (3,2) code, it can be shown that the iterative decoder's decision for any bit i can be determined by optimum decoding on a much smaller set of pseudo-codewords. At any iteration t there will be the same $2 \cdot \left(\prod_{\forall i:h_{ji}=1} 2^{(w_{r_i}-2)}\right) = 8$ pseudo-codewords that, given the input probabilities of the iterative decoder at iteration t, can accurately determine the APP decisions. Analytically, the local expressions from the two parity check equations for the extrinsic probability of

0 for bit 1 would be:

$$\textit{Equation1: } e_{l_{11}}(0) = [q_2 \cdot q_3] + [p_2 \cdot p_3] \tag{6.45}$$

$$\textit{Equation2: } e_{l_{21}}(0) = [q_2 \cdot q_3] + [p_2 \cdot p_3] \tag{6.46}$$

Where $p_j = p(1|r_j)$ and $q_j = 1 - p_j = p(0|r_j)$. The overall extrinsic of 0 for bit 0 would be

$$e_1(0) = q_2^2 \cdot q_3^2 + q_2 \cdot p_2 \cdot q_3 \cdot p_3 + p_2 \cdot p_3 \cdot q_2 \cdot q_3 + p_2^2 \cdot p_3^2 \tag{6.47}$$

and the APP output of the iterative decoder is just

$$APP_1(0) = q_1 \cdot q_2^2 \cdot q_3^2 + q_1 \cdot q_2 \cdot p_2 \cdot q_3 \cdot p_3 + q_1 \cdot p_2 \cdot p_3 \cdot q_2 \cdot q_3 + q_1 \cdot p_2^2 \cdot p_3^2 \tag{6.48}$$

Comparing this with the decision of the optimal MAP decoder which is

$$APP_{1,ML}(0) = \underbrace{q_1 \cdot q_2 \cdot q_3}_{ML[1]} + \underbrace{q_1 \cdot p_2 \cdot p_3}_{ML[2]} \tag{6.49}$$

it can be seen that the iterative decoder's decision at iteration 0 is a scaled version of the optimal ML, plus an offset.

$$APP_1(0) = \underbrace{(q_2 \cdot q_3 \cdot ML[1]) + (p_2 \cdot p_3 \cdot ML[2])}_{\textit{scaled ML solution}} + \textit{offset} \tag{6.50}$$

In a similar way the analytical decision of the iterative decoder for $APP_1(1)$ can be easily obtained. Given the updated input probabilities at each iteration t, the iterative decoder will perform optimal MAP decoding for the decision $APP_1(x)$ based on the set of pseudo-codewords $\mathbf{u} \in \mathcal{U}_1$, where in this case

$$\mathcal{U}_1 = \left\{ \underbrace{00000, 00101, 01100, 01111}_{x=0}, \underbrace{11100, 11001, 10110, 10011}_{x=1} \right\} \tag{6.51}$$

6.3 Correlation of pseudo-codewords with the received vector

The previous examples have demonstrated some simple cases in which the iterative decoder performs optimal MAP decoding in the pseudo space $\mathcal{U}$ instead of the code space $\mathcal{C}$. As a consequence the iterative decoder occasionaly fails to converge to the optimal ML solution. In [27] the authors have presented similar conclusions but through the code's computation graph (figure 6.1) that is used to specify the pseudo-code. The authors claim that iterative decoding is optimal but on the pseudo-code $\mathcal{C}'$ and not the actual code $\mathcal{C}$. In the same paper the pseudo-codewords are categorised as "good" and "bad" based on whether their component u_i is in favour or not of the transmitted symbol.
It can be deduced from all these that during the iterative procedure, codewords are competing with "good" and "bad" pseudo-codewords. Those that exhibit higher correlation with the received vector will dominate. As long as the codewords are strongly correlated to the received vector, the iterative decoder should be able to converge to the ML solution. In the opposite case where none of the codewords is strongly correlated to the received vector, the chances that pseudo-codewords dominate increase.
The correlation of the valid codewords $\mathbf{c} \in \mathcal{C}$ with the received vector $\mathbf{r}$ can be evaluated by the final decisions of the MAP decoder. When all decisions are saturated to a posteriori probabilities very close to 1 or 0 (extreme decisions), it is clear that one of the valid codewords dominates. The case where the a posteriori probabilities of the MAP decoder are not strongly decided, suggests that there is no clear domination of a codeword and that one or more opponents exist in close Euclidean distance to the received vector. In that latter case it is very probable that if iterative decoding is applied, one or more pseudo-codewords might exhibit sufficient correlation with $\mathbf{r}$ to challenge and compete the valid codewords. In that case they will be capable of providing significant contribution to the final decision. The representative histogram of figure 6.2 proves that. Based on a sample of 500 blocks that the optimal MAP decoder has successfully decoded, the SP algorithm fails to converge to the ML solution for those blocks that the optimal MAP has decoded with low reliability.

6.3.1 Modifying the weakest contributor

The position (coordinate) with the least reliable APP decision at the optimal MAP decoder output can be considered as the weakest contributor to the correlation of the codewords with the received vector. Similarly, it can be considered as the strongest con-

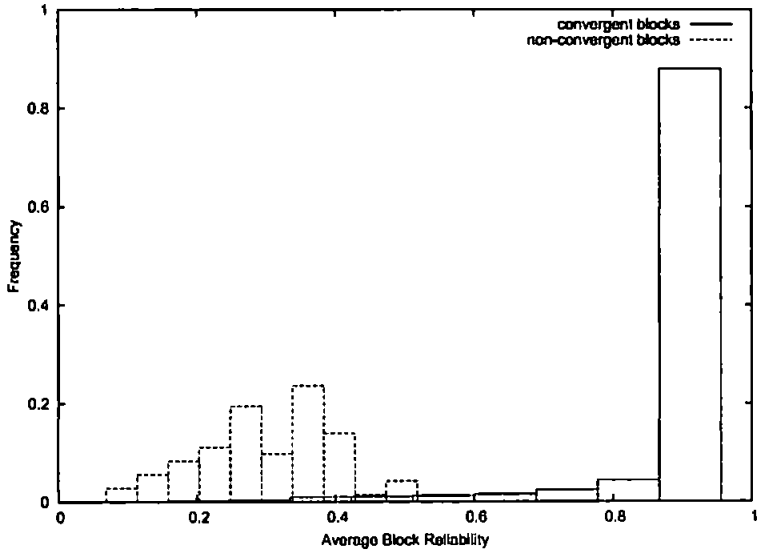


Figure 6.2: Histogram of the average reliabilities of the optimal MAP decoder APP decisions for blocks that converge and not converge to the ML solution when iteratively decoded by the Sum-Product algorithm. Results based on the perfect difference set (7,3) cyclic code

tributor to the correlation of pseudo-codewords with the received vector. The immediate question that arises is if and by how much the performance of the iterative decoder would improve if the correlation of this specific bit with "good" pseudo-codewords and codewords that are associated with the transmitted symbol at that position, was maximised. That could be achieved by modifying the received vector at that position, to one of the available transmitted values (± 1 for BPSK modulation).

The results of figure 6.3 show that the improvement achieved by modification of just one of the received vector coordinates is remarkable. For the (7,3) code the iterative decoder performance becomes identical to the ML. The same method applied to the (63,16) code improves the iterative performance by almost an order of magnitude to just fractions of dB away from the optimum. For codes with low information length k as the ones in figure 6.3, implementation of the optimal MAP decoder is feasible and straight forward. For codes of higher k straight forward MAP decoding is computationally prohibited. In order to prove the significance of the modification of the least reliable position of the MAP output on codes with higher k, it is crucial to find an indirect way of approximating as accurately as possible the exact APP decisions of the optimal MAP decoder. This can be achieved by the means of a list decoder.

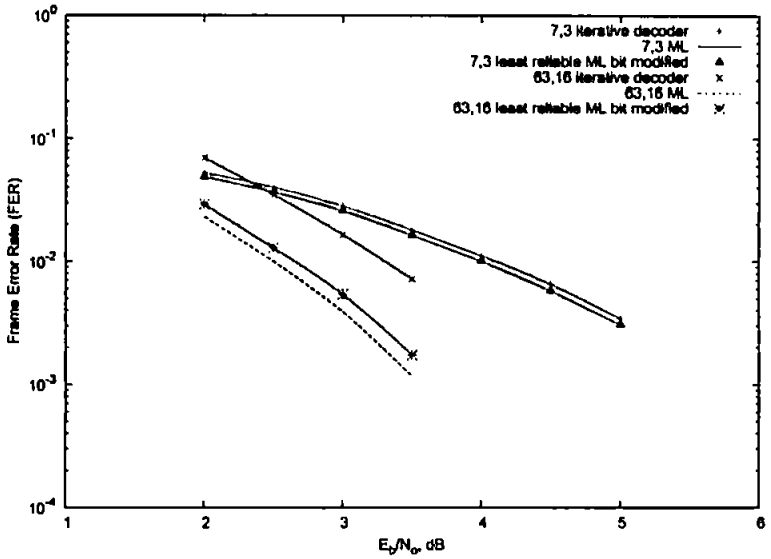


Figure 6.3: Improvement of iterative decoding performance by modifying the position with the least reliable MAP decision

6.3.2 Approximating the optimal MAP a posteriori probabilities by the use of an Ordered Statistics list decoder

The list of codewords used for approximating the optimal MAP a posteriori probabilities can be created by the use of an *ordered statistics decoder* (OSD) [25, 26]. The main idea of this decoding method was invented originally by Dorsch [17] back in 1974 (Dorch algorithm). However, nowadays it is widely known as OSD.

Since the created list will have only a fraction of the size of the full codeword list, it is expected that the computed APPs will only approximate the real ones. However, as the size of the list increases the approximation should be improving and the least reliable position of the optimal MAP should be identified more accurately. As a consequence the obtained improvement by a single bit modification should be analogous to the list size. That would prove the major role of the least reliable bit of the optimal MAP to the correlation between pseudo-codewords and the received vector, even for longer codes where the phenomenon cannot be investigated directly, via implementation of the optimal MAP decoder.

Description of the OSD

The OSD is based on the idea that the least reliable channel outputs attract higher probability of being decoded in error, and that those positions can be recovered by the more reliable ones if the latter are error free. The received vector is permuted in such a way that its components are in descending order in terms of entropy, from the most unreliable to the most reliable. If the original received vector is $\mathbf{r}$ and its corresponding entropy vector is $\mathcal{H}$, $\mathbf{r}$ is permuted due to $\mathcal{H}$ into a new vector $\mathbf{r}'$. If we define the permutation function as $\lambda_{\mathcal{H}}$ then the $\mathbf{r}'$ can be expressed as

$$\mathbf{r}' = \lambda_{\mathcal{H}}(\mathbf{r}) \tag{6.52}$$

The H matrix columns are reordered accordingly so that

$$(\forall i : 0 \leq i < N), H'_i = \lambda_{\mathcal{H}}(H_i) \tag{6.53}$$

where $\mathbf{H}'$ is equivalent to the original $\mathbf{H}$ matrix but with its columns reordered. The next step is to bring $\mathbf{H}'$ into the echelon form so that the $N-k$ least reliable positions can be derived from the k most reliable ones by simple encoding operation. The resulted codeword is then a candidate ML solution. The above procedure summarises the order-0 OSD. It is easy to realise that an order-0 OSD will only provide the correct solution if transmission errors occur only within the $N-k$ least reliable positions.
The OSD idea can be exploited further to higher order implementations that improves the performance but also increases the complexity of the scheme. An $order-\kappa$ OSD assumes κ errors among the k most reliable bits and will trial all $\binom{k}{\kappa}$ combinations. From the list of the candidate codewords, the one that exhibits the minimum Euclidean distance from the received vector will be chosen as the decoded output. The OSD is condemned to fail to converge to the ML solution whenever

$$\epsilon > \kappa \tag{6.54}$$

Where ϵ is the number of errors among the k most reliable bits, and κ is the order of the OSD. The binomial rise of the complexity with higher order implementations makes the use of the decoder impractical for orders higher than 3 and practical block-length. At the limit (order-k) the OSD becomes equivalent to the maximum likelihood decoder.

Creation of the list and computation of the approximated APP bit decisions

The ordered statistics decoder can be used as a way of listing a number of candidate codewords, hopefully the most likely ones, through which the a posteriori bit decisions of the optimal MAP decoder can be approximated. Assuming an *order* $-\kappa$ OSD the list size $\mathcal{L}$ will be equal to the sum of the binomial coefficients.

$$\mathcal{L} = \sum_{i=0}^{i=\kappa} \binom{k}{i} \tag{6.55}$$

Based on the obtained codewords list we can compute the associated APP bit decisions d_i using the MAP equation below.

$$d_i(x) = \sum_{j=0, c_{ji}=x}^{\mathcal{L}} p(\mathbf{c}_j|\mathbf{r}) \tag{6.56}$$

where $x \in [0,1]$, $\mathbf{c}_j \in \mathbf{C}$ and $\mathbf{c}_j = [c_{j,0}, c_{j,1}, ..., c_{j,N-1}]$.

Approximation error with increasing order

It is reasonable to assume that the bit APP values obtained by the list decoder should approach the real APPs of the optimal MAP decoder as the order κ increases. Eventually, when $\kappa = k$ the two decoders should be identical. For the case of the (63,16) code, implementation of the optimal MAP decoding is realisable due to the low value of k and direct comparison between the two methods can be made. Figure 6.7 shows that implementation of an *order* $-$ 3 list decoder is enough to determine accurately the APP decisions of the optimal MAP decoder. So, by using only 697 out of the 65,536 available codewords we can still get the exact optimal MAP decisions.

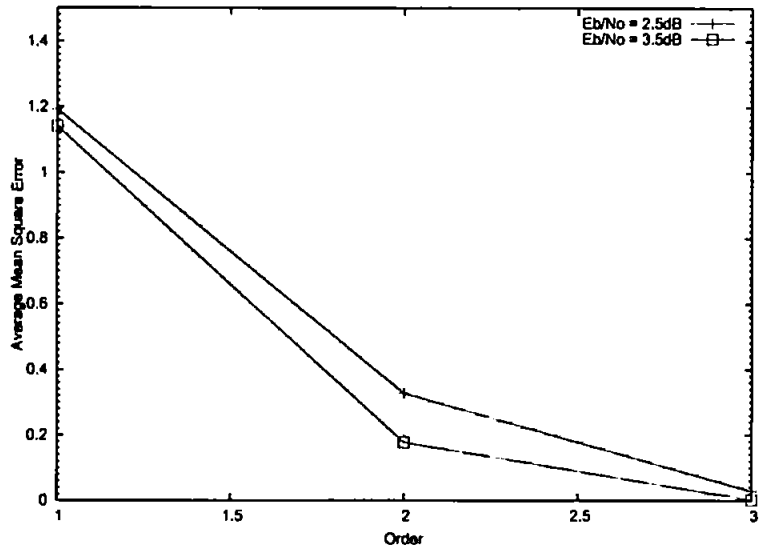


Figure 6.4: The average mean square error per processed block between the APP values obtained by optimal and list decoding of order 1, 2 and 3 (based on the 63,16 code)

Performance improvement by modifying the position that corresponds to the least reliable APP decision

The following figures depict the FER performance of various LDPC codes when the least reliable bit, as this is estimated by a list decoder of *order* $-\ \kappa$, is modified. For all codes, better estimation of the least reliable bit position by higher order list decoding results in better performance. Notice that for short codes, such as the (63,37) and (105,53), the FER curves are arbitrarily close (almost identical) to the ML performance of the codes with $\kappa = 2$ or $\kappa = 3$. For the longer (255,64) code the improvement steps are smaller but steady and the order should be increased further to approach closer to the ML curve.

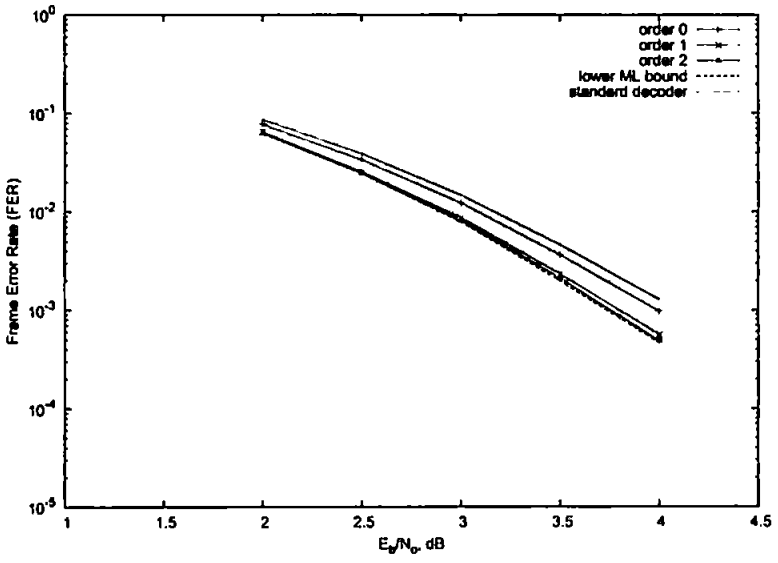


Figure 6.5: FER performance of the (63,37) code when the least reliable APP position, as this is estimated by a list decoder of $order - \kappa$, is modified.

6.4 Conclusive Remarks

Concluding, some of the major points of this chapter.

- By following analytically the operation of the SP algorithm, it was shown that the lack of convergence to ML solution can be attributed to the presence of pseudo-codewords.
  - Iterative decoding performs optimum decoding on a vector space which is not identical to the codebook $\mathcal{C}$.
  - The number of pseudo-codewords can be computed by using the column and row weight of the parity check matrix.
  - Feedback of information creates unequal scaling problems (the probability of some of the terms is double counted).
  - For codes with minimum length loops, the scaling effects are apparent even before the beginning of iteration 1.
  - Distortion of the vector space is also noticeable on loop-free codes, with minimum impact to the iterative decoder performance though.
- When the correlation of one or more pseudo-codewords with the received vector is high, the chances that the iterative decoder will fail to converge to the ML solution increase.

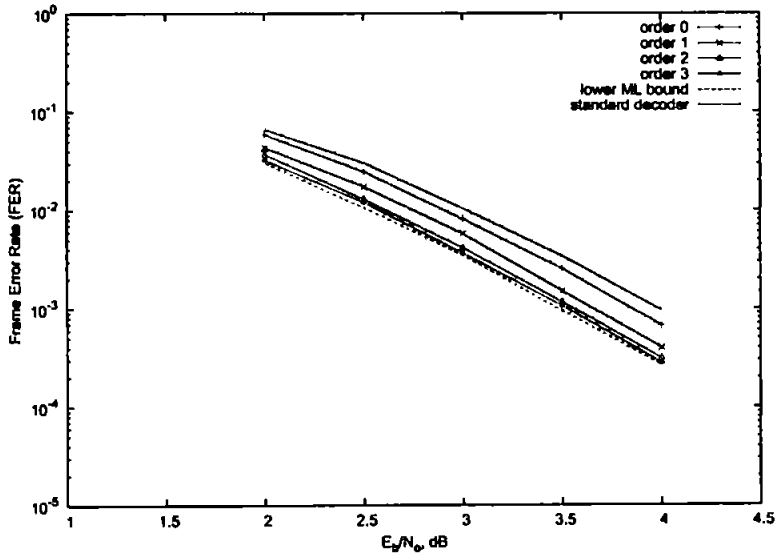


Figure 6.6: FER performance of the (105,53) code when the least reliable APP position, as this is estimated by a list decoder of $order - \kappa$, is modified.

- Loose correlation of the actual valid codewords with the received vector is associated with considerably higher probability that pseudo-codewords will dominate and affect critically the final decisions.
- The least decisive positions at the output of the optimal MAP can be considered as the strongest contributors to the correlation of pseudo-codewords with the received vector.
  - Modification of the received value of the single least reliable (decisive) bit position, offers significant coding gain.

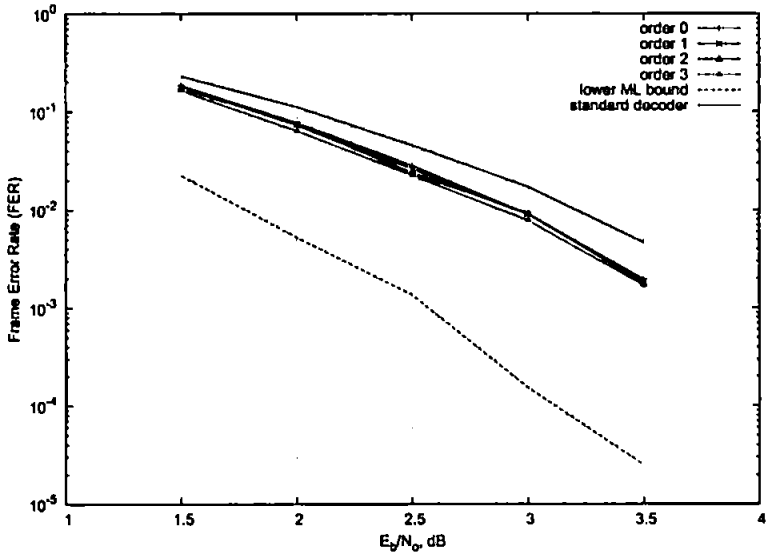


Figure 6.7: FER performance of the (255,64) code when the least reliable APP position, as this is estimated by a list decoder of $order - \kappa$, is modified.

7 The Received Vector Coordinate Modification (RVCM) algorithm

The previous chapter revealed the significant impact that the modification of a single coordinate of the received vector has to the final decision of iterative decoders. This impact is maximised when the modified coordinate coincides with a strong contributor to the correlation of the received vector with one or more pseudo-codewords.
The potential gain from the modification of a single bit, even for long codes, has inspired the invention of the RVCM algorithm [45]. Its description and some of its general characteristics are presented in the rest of this chapter.

7.1 Description of the algorithm

Assume an AWGN channel with BPSK modulation and output $\mathbf{r} = \{r_0, r_1, .., r_N\}$. Consider $\mathbf{L} = \{l_0, l_1, .., l_N\}$ as the log-likelihood-ratios vector. Given that the decoder fails to satisfy the set stopping criteria, the RVCM algorithm commences. The latter can be summarised as follows

- Step 1: Vector $\mathbf{L}$ is stored
- Step 2: Define parameter β_{max}, $0 < \beta_{max} \leq N$
- Step 3: Create a size-β_{max} list of candidate positions $\mathbf{I} = I_0, I_1, ..., I_{\beta_{max}}$ to be modified
- Step 4: Initialise $\beta = 0$
- Step 5: l_{I_β} is substituted by $l'_{I_\beta} = +\infty$ and decoding restarts
- Step 6: We store the Euclidean distance $Euc_1(\beta)$ of the decoder output $d_1(\beta)$ from the received vector $\mathbf{r}$ when bit I_β has been modified and the associated output of the decoder as a candidate solution.
- Step 7: l_{I_β} is substituted by $l'_{I_\beta} = -\infty$ and decoding restarts

- Step 8: We store the Euclidean distance $Euc_0(\beta)$ of the decoder output $d_0(\beta)$ from the received vector $\mathbf{r}$ when bit I_β has been modified and the associated output of the decoder as a candidate solution.

- Step 9: If $\beta < \beta_{max}$ we restore the original l_{I_β} value, increment β and return to step 5. If $\beta >= \beta_{max}$ we proceed to the next step

- Step 10: Choose the decoder output $d_x(\beta)$ which is associated with the minimum $Euc_x(\beta)$ ($x \in [0, 1]$) as the most likely solution

The above algorithm can in many cases offer improvement of an order of magnitude or more in terms of FER and BER on both LDPC and turbo code schemes [44, 43]. For many short codes, application of RVCM delivers maximum likelihood performance as it will be shown in later chapters. Note that the complexity added in the case of LDPC codes is at most $2N$ additional decoding operations when $\beta = \beta_{max} = N$, and it increases linearly with the block-length N since only one coordinate is modified at any time. However, this is just the worse case scenario since with just a fraction of modifications ($\beta_{max} \ll N$) we can get performance arbitrarily close to the optimum especially at the high SNR region. For turbo codes only the systematic bits are used in the algorithm, thus the highest value for β can only be equal to the number of the systematic bits involved. For convenience, the bit positions that are capable of correcting a prior non-convergent block if modified, will be called *critical bits* throughout the rest of the thesis.

7.2 The mrl criterion

Target of any algorithm that attempts to improve the convergence of iteratively decoding schemes is the optimal ML performance. Since ML decoding is prohibitively complex for practical codes, the optimum performance is approximated by a bound. The *mrl* criterion [21] serves as a practical lower ML bound. Let $\mathbf{R}$ be the received, $\mathbf{T}$ be the transmitted and $\mathbf{D}$ the error decoded vectors in the euclidean space. Then a block is considered as *mrl* if

$$Euc(\mathbf{D}, \mathbf{R}) < Euc(\mathbf{T}, \mathbf{R}) \tag{7.1}$$

In this case we know for sure that the correct codeword does not coincide with the closest to the received vector since at least one error codeword is closer (is more likely) than that. Thus, the maximum likelihood decoder would fail. After application of the RVCM algorithm the same check can be applied to these blocks that have failed to converge. RVCM algorithm has achieved the optimal performance when all the standard decoder's

non-convergent blocks that it has failed to correct satisfy equation 7.1. In other words, optimum performance has been achieved when all non-mrl blocks have been corrected by RVCM.

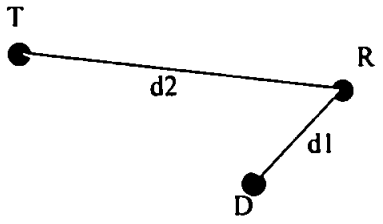


Figure 7.1: Representation of an mrl decoded block in terms of Euclidean distances

7.3 The effect of RVCM in the behaviour of iterative decoders

The previous chapter presented a detailed analysis of the SP iterative algorithm with examples based on various LDPC codes, and highlighted the main problems that lead the iterative decoder to suboptimal solutions. The emergence of non-valid sequences (named as pseudo-codewords) that compete with the valid codewords was considered as a major reason for any convergence problems. It was also shown that the presence of loops in the code's parity check matrix is the source of unequal scaling problems which in [27] have been visualised in the Euclidean space as skewness effects in the decision boundaries of the iterative decoder.
Considering the dummy (3,2) code used in [27] ($\mathbf{H} = [111;111]$), we recall from the analysis of the previous chapter that the output of the iterative decoder for bit 1 at the end of iteration 0 is

$$APP_1(0) = q_1 \cdot q_2^2 \cdot q_3^2 + q_1 \cdot q_2 \cdot p_2 \cdot q_3 \cdot p_3 + q_1 \cdot p_2 \cdot p_3 \cdot q_2 \cdot q_3 + q_1 \cdot p_2^2 \cdot p_3^2 \qquad (7.2)$$

The decision of the optimal MAP decoder is just the sum of the two valid codewords that involve q_1.

$$APP_{1,ML}(0) = \underbrace{q_1 \cdot q_2 \cdot q_3}_{ML[1]} + \underbrace{q_1 \cdot p_2 \cdot p_3}_{ML[2]} \qquad (7.3)$$

The two codewords, based on which the optimal decision should be made, have been unequally scaled at the output of the iterative decoder. Moreover, more terms have

emerged that contribute to the decision. Keeping track of the distortion so that unequal scaling and emergence of pseudo-codewords is prevented, would require exact analysis of the decoding procedure for all bits. Such a procedure is equivalently complex to ML decoding, thus unrealisable for practical codes.

Let us assume that RVCM is applied on bit 2, so that $q_2 = 1$ and $p_2 = 0$. Then, equation 7.2 becomes

$$APP_1(0) = q_1 \cdot q_2 \cdot q_3^2 \tag{7.4}$$

It can be seen that the additional terms of 7.2 have been eliminated and the scaling problems of bit 2 have been also eliminated since $q_2^2 = q_2$ and $p_2^2 = p_2$. Given that the transmitted value for bit 2 is $x_2 = 0$, so that the modification is made in favour of the correct decision, the second term (ML[2]) of the MAP decoder has no effect and the only distortion present at the output of the iterative decoder is the scaling of bit 3. Hence, with the minimum effort, RVCM is providing reduction of any distortion effects induced by iterative decoding. The interaction between all bits and the increasing correlation among them with iterations, spreads the effect of a single modification and improves the convergence even on long codes as it will be seen on later chapters. Literally, the decoder is self-corrected during the iterative procedure. Similar conclusions can be drawn by investigation of the iterative output after RVCM has been applied on the Hamming (7,4) code.

0 0 1 0 1 1 1
0 1 0 1 0 1 1
1 0 0 1 1 0 1

The analytical output of the SP algorithm for bit 3 at iteration 0 is given by

$$\begin{aligned} e_3(1) = & \frac{q_6}{q_2} \cdot (ML[1] + ML[4]) + \frac{p_6}{q_2} \cdot (ML[6] + ML[7]) + \\ & \frac{q_6}{p_2} \cdot (ML[2] + ML[3]) + \frac{p_6}{p_2} \cdot (ML[5] + ML[8]) + \ \mathit{offset} \end{aligned} \tag{7.5}$$

Where ML denotes the valid codewords on which, ideally, the decision should be based on. The offset includes 8 non-valid sequences (pseudo-codewords), readily:

$$\begin{aligned} \mathit{offset} = & [q_0 \cdot p_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + [p_0 \cdot p_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + \\ & [q_0 \cdot q_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [p_0 \cdot q_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] + \\ & [p_0 \cdot q_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + [q_0 \cdot q_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + \\ & [p_0 \cdot p_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [q_0 \cdot p_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] \end{aligned} \tag{7.6}$$

It is easy to see that applying RVCM on bit 6, i.e. setting either q_6 or p_6 to 0, eliminates all terms of the offset and the scaling problem of bit 6. The decision of the iterative decoder for bit 3 at iteration 0 resembles much closer the output of the optimal decoder.

$$e_3(1) = \frac{ML[1] + ML[4]}{q_2} + \frac{ML[6] + ML[7]}{q_2} + \frac{ML[2] + ML[3]}{p_2} + \frac{ML[5] + ML[8]}{p_2} \quad (7.7)$$

For SCCC and PCCC turbo schemes, an analysis similar to that of the SP algorithm is difficult. The sub-optimality of turbo decoders is due to the fact that MAP decoding is performed on the trellises of the component codes and not on the complex joint trellis of the overall code. In contrast to the SP algorithm where the analysis can be performed locally on each parity check equation, MAP decoding would involve the whole set of bits that participate on the trellis. Consider the joint transition probability $\sigma_t(m, m')$ from state m' to state m at position t, as this is defined in the description of the BCJR algorithm [5].

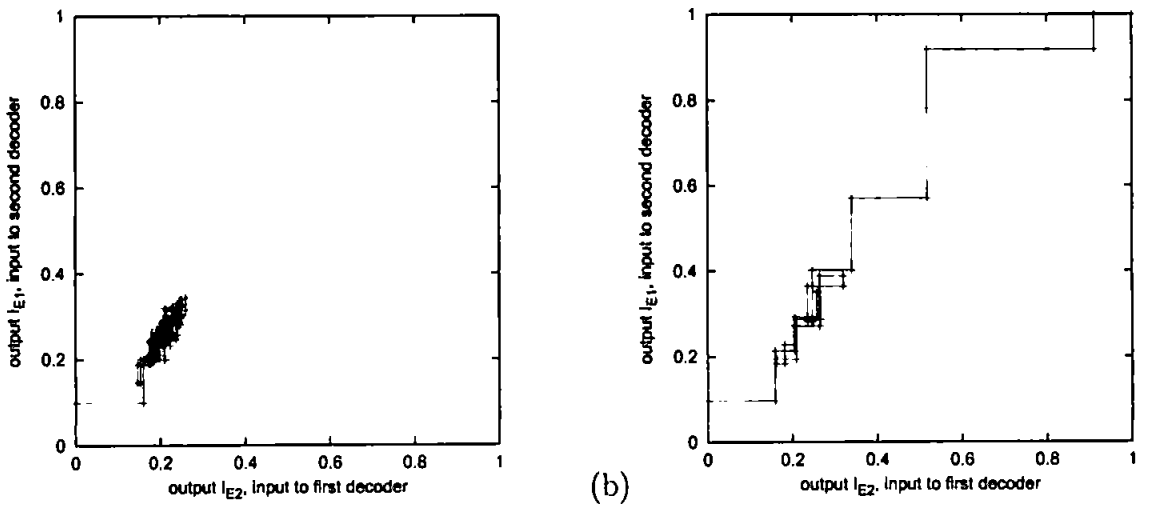
$$\sigma_t(m', m) = \alpha_{t-1}(m') \cdot \gamma_t(m', m) \cdot \beta_t(m) \quad (7.8)$$

It seems from the above equation that the decision at the decoding instant t can be expressed based on the available information at $t-1$ and t. From the definitions of the forward and backwards state probabilities though (denoted as α and β) it can be seen how the total trellis participates on that decision.

$$\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m') \cdot \gamma_t(m', m) \quad (7.9)$$

$$\beta_t(m) = \sum_{m'} \beta_{t+1}(m') \cdot \gamma_{t+1}(m, m') \quad (7.10)$$

Thus, in order to produce analytically the MAP decision for bit i at instant t it is essential to use the total information from the whole trellis. Even for short trellises this is a complex procedure. Instead, the effect of RVCM on turbo decoding can be demonstrated by the use of the existing convergence theory. Figure 7.2 shows the EXIT charts [65] of a non-convergent block before and after applying the RVCM algorithm. The EXIT graphs are based on a SCCC scheme with block-length N equal to 2000. Although the modification of just one bit among a block-length of 2000 seems to be insignificant, its effect proves to be quite important with later iterations. Note that in the first few iterations the mutual information gain before and after application of RVCM is almost the same. The real improvement becomes apparent only after the decoder has processed the small modification, escapes the random walk and heads towards a solution which is

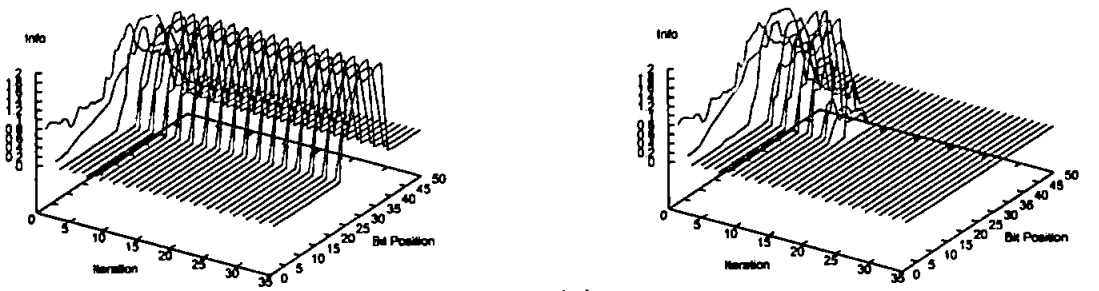


(a) (b)

Figure 7.2: (a)EXIT chart of non convergent block (b)EXIT chart for the same block after successful application of RVCM

the maximum likelihood in this case.

A representative case of the effect of RVCM algorithm on the parameters α and β of each MAP decoder can be seen from graph 7.3 where the information content of the state probabilities $(\alpha_i \cdot \beta_i)$ at position i is plotted for a number of iterations.



(a) (b)

Figure 7.3: Uncertainty of state probabilities: (a)non-convergent block (b)application of RVCM on the same block

7.4 General characteristics of RVCM

This section will deal with some general characteristics and statistics of RVCM, obtained by application of the algorithm to LDPC and turbo codes. The performance of the algorithm and the methods for identifying critical bits will be dealt separately for each scheme in later chapters.

7.4.1 Dependence of critical bits on the structure of the code

Figure 7.4 depicts the distribution of the critical bit positions for an SCCC scheme. The two graphs have been obtained for the same code, the same transmitted data, for a channel with the same variance but different noise seed. The distribution is random and

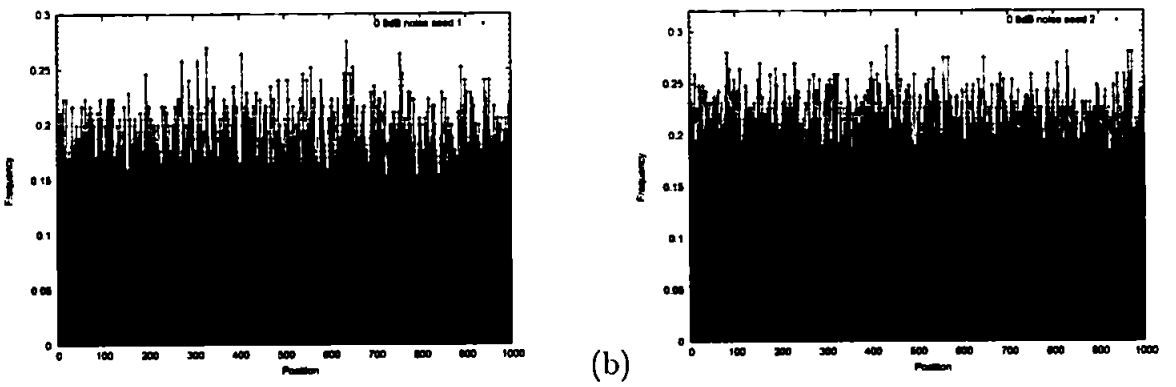


Figure 7.4: Distribution of critical positions for a rate 1/4 SCCC scheme of block-length N=2000 and random interleaving. The two graphs have been both obtained at Eb/No of 0.8dB but with different noise seed

many bits are assigned more or less the same probability of being critical. Additionally, the actual frequency of occurrence of the most frequent bit does not exceed the value of 0.3, thus the minimum effort solution (i.e. predefining a unique position on which the RVCM would be applied in case of non-convergence) would correct slightly less than a third of the error blocks in the best case.

The fact that changing only the noise seed changes the distribution of the critical positions proves that the structure of the interleaver by itself is not adequate for determining these critical positions. Similar results have been obtained for PCCC schemes (figure 7.5) and LDPC codes with regular structure.

For non-regular LDPC codes there has not been observed any strong dependency between the column weight (number of participations in parity check equations) of any bit i and the corresponding frequency of occurrence of the same bit as critical. Figure 7.6 displays

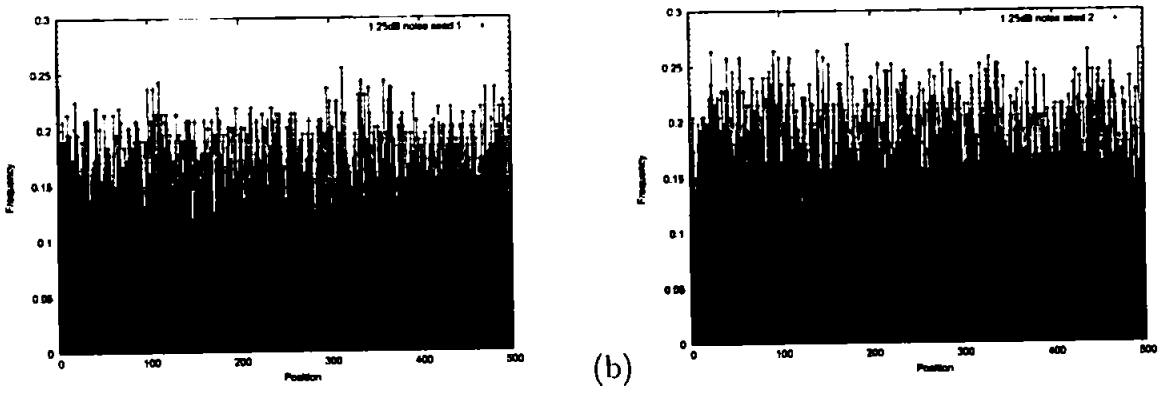


Figure 7.5: Distribution of critical positions for a rate 1/3 PCCC scheme of block-length N=1500 and S-interleaving. The two graphs have been both obtained at Eb/No of 1.25dB but with different noise seed

the distribution of critical bits for the (200,100) irregular LDPC code at 2.0dB. The same figure depicts a plot of the column weight of each of the 200 participating bits. It is easy to observe that the distribution is not analogous to the column weight of the code bits.

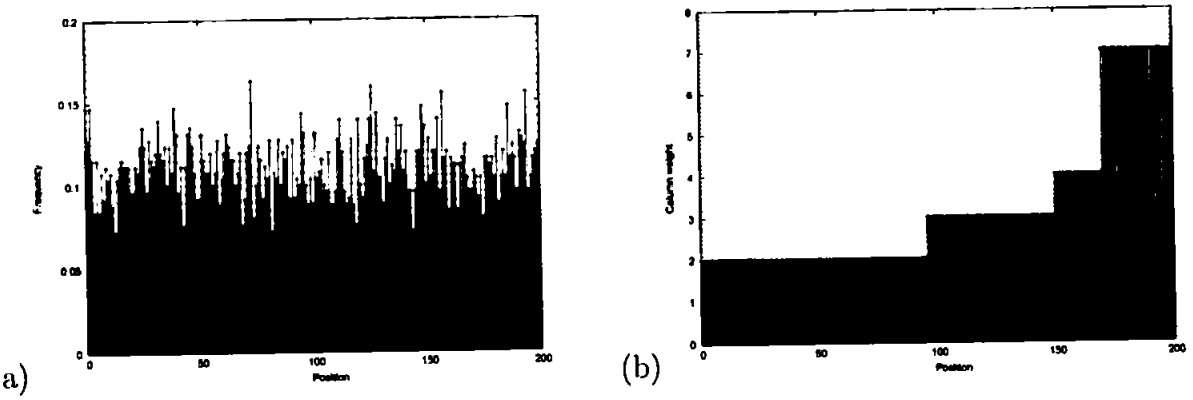


Figure 7.6: (a) Distribution of critical positions for a (200,100) irregular LDPC code at E_b/N_o of 2.0dB; (b) Column weight distribution for the same code

On the other hand, critical bits associated with high column weight lead the decoder to convergence in less iterations as figure 7.7 reveals.

7.4.2 Number of existing critical bits per block

The number of existing critical bits per block is a significant parameter when considering the practical application of RVCM. Since the bit(s) with the least reliable MAP decision(s) is(are) not known, various selection methods have been created so that the critical bits

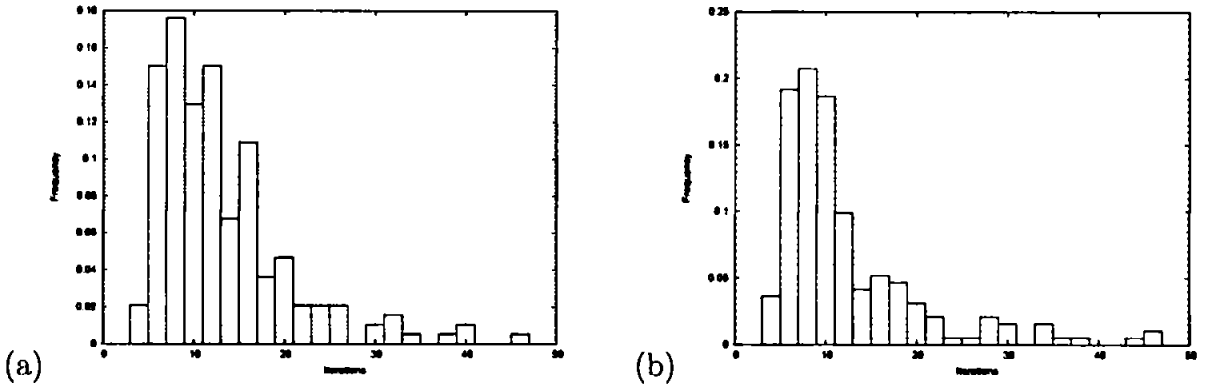


Figure 7.7: Histograms of the number of iterations required for the RVCM algorithm to converge to the ML solution when RVCM (a) 30 candidate critical bits with column weight $w_c = 2$ and (b) 30 candidate critical bits with $w_c = 7$ are applied to the irregular (200,100) LDPC code at E_b/N_o of 2.0dB (note the different scaling of the two histograms)

are identified with the highest accuracy. An error is always involved in these selection methods so the larger the number of critical bits in a block the higher it is the chance of successfully identifying at least one of them.

The number of critical bits is dependent on the SNR as, on average, more critical bits exist per block as the SNR increases. Figures 7.8 and 7.9 display the associated histograms for both LDPC and turbo schemes when operating at lower and higher SNR values.

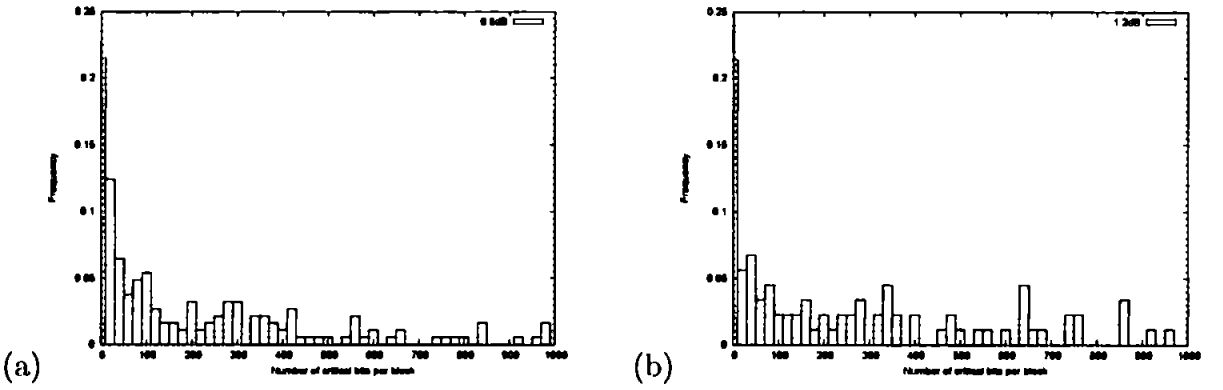


Figure 7.8: Histograms of the number of critical bits per block for a rate 1/4 SCCC scheme of block-length N=2000 and random interleaving at (a)Eb/No of 0.8dB and (b)Eb/No of 1.2dB

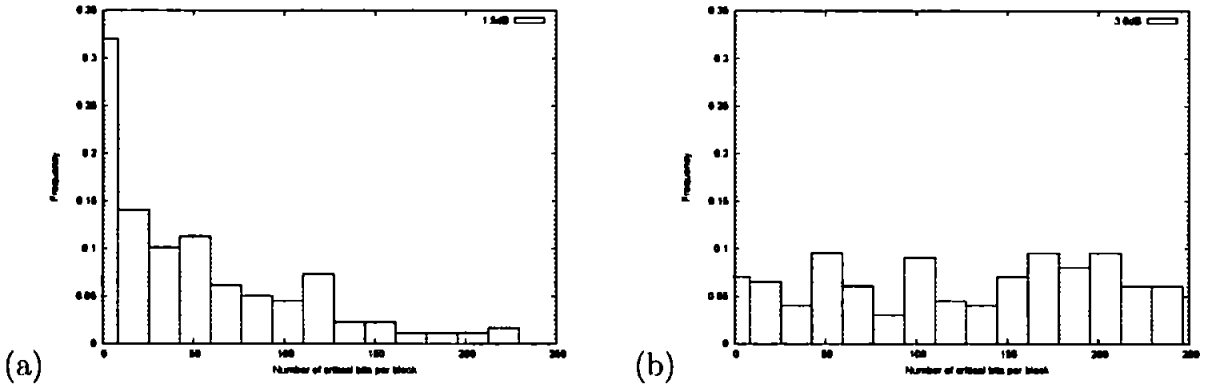


Figure 7.9: Histograms of the number of critical bits per block for the (255,64) cyclic LDPC code at (a)Eb/No of 1.5dB and (b)Eb/No of 3.0dB

7.4.3 Number of iterations required for the RVCM to converge

The number of iterations required by the iterative decoder to converge when modifying a critical position, is of great importance as it affects the overall complexity and the latency imposed by RVCM to the system. The depicted histograms are based on the critical position that provides the fastest convergence (convergence at the minimum number of iterations among all critical bits in a block).

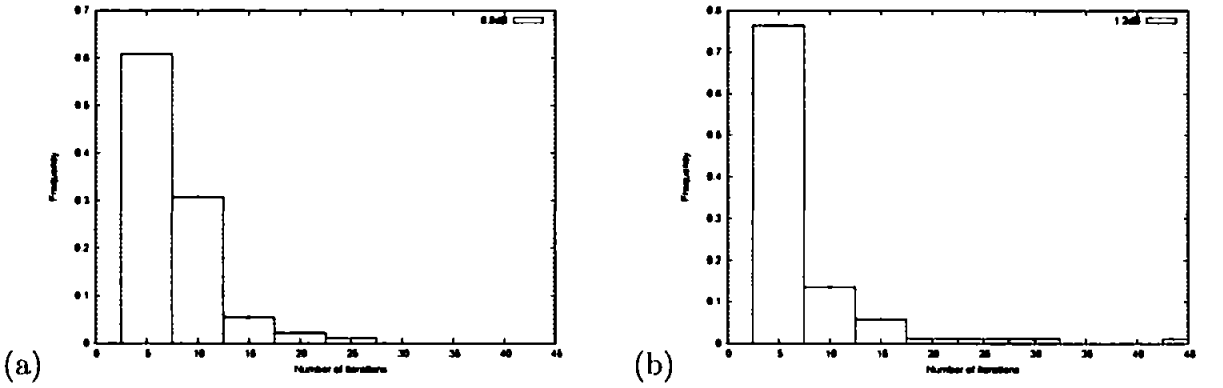


Figure 7.10: Histograms of the minimum number of iterations required when a critical bit is modified for a rate 1/4 SCCC scheme of block-length N=2000 and random interleaving at (a)Eb/No of 0.8dB and (b)Eb/No of 1.2dB

As expected, at higher Eb/No convergence is achieved faster for all iterative schemes. The information obtained by these graphs can be used to increase the efficiency of practical applications of RVCM. As it will be discussed in the following chapters, various methods have been developed for determining the critical positions and reducing the

number of trials ($\beta_{max} < N$ in the description of the RVCM algorithm at the beginning of the chapter) with minimum loss in performance. An alternative strategy for schemes where modification of a critical bit is guaranteed to provide convergence within very few iterations (as in the case of graph 7.11) would be to keep the trials at the maximum value $\beta_{max} = N$ but at the same time to limit the maximum iterations performed per trial.

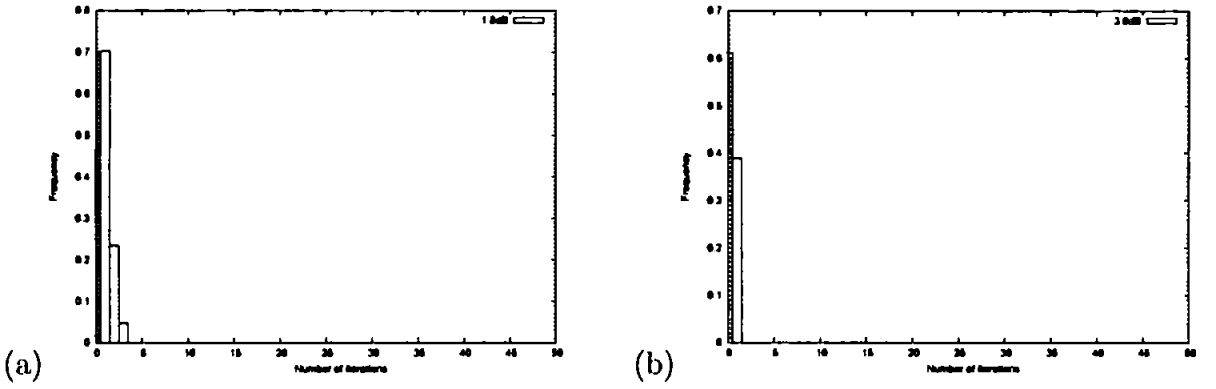


Figure 7.11: Histograms of the minimum number of iterations required when a critical bit is modified for a rate 1/3 PCCC scheme of block-length N=1500, RSC(1,5/7) and random interleaving at (a)Eb/No of 1.5dB and (b)Eb/No of 3.0dB

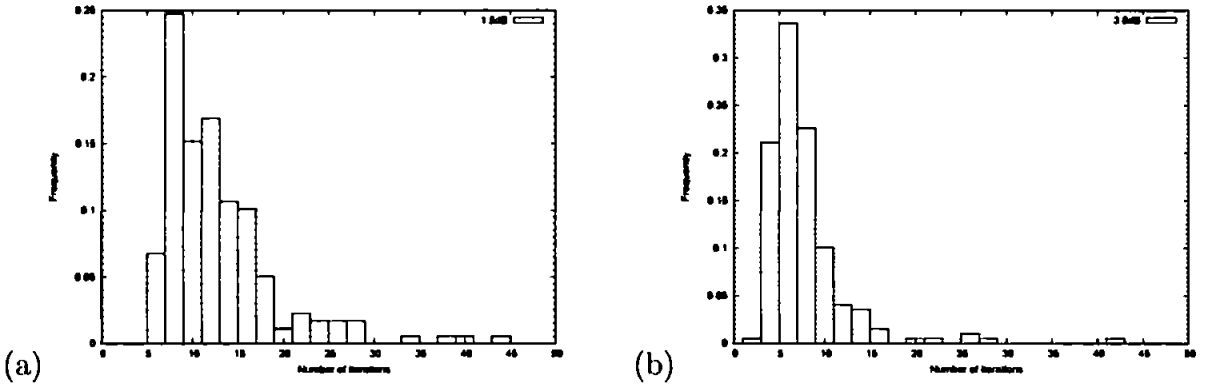


Figure 7.12: Histograms of the minimum number of iterations required when a critical bit is modified for a (255,64) cyclic LDPC code at (a)Eb/No of 1.5dB and (b)Eb/No of 3.0dB

8 RVCM on LDPC codes

This chapter investigates the performance improvement delivered by RVCM when applied on LDPC codes [42]. Some of the characteristics of critical bits (like the entropy of their input probability and their participation in unsatisfied equations) are demonstrated and are then used as measures for candidate critical bits selection criteria. It is also attempted to determine the code characteristics that should be taken into account for the choice of the most appropriate selection criterion, as in many cases the deviation in the results obtained by the two is significant.

8.1 Performance of RVCM on LDPC codes

Figures 8.1 and 8.2 display typical performances of RVCM algorithm when applied to LDPC codes. All graphs have been obtained with $\beta_{max} = N$, hence they represent the maximum performance that can be achieved by RVCM on these codes. All codes used are regular, derived from cyclotomic cosets [68] and guarantee the absence of short cycles within the H matrix structure (Tanner 155,64 is not derived from cyclotomic cosets but it also guarantees absence of short cycles). Notice that for the two short codes (105,53 and 63,37) the optimum RVCM performance is identical to the mrl bound. At longer codes where the gap between iterative and ML decoding is larger, the RVCM achieves improvement of more than one order of magnitude in terms of FER (the BER improvement is similar).

8.2 Distribution and frequency of critical bits

RVCM algorithm was shown to work with high success on LDPC codes. The next challenge is to find a way of identifying any critical bits as accurately as possible. For that, it is necessary to extract information about distinctive characteristics shared among critical bits that make them distinguishable from the rest.

It is straightforward to assume that critical bits belong to that group of bits that either during the iterative decoding process are estimated erroneously or their channel output value is closer to the wrong symbol. Table 8.1 reveals though that this is not entirely

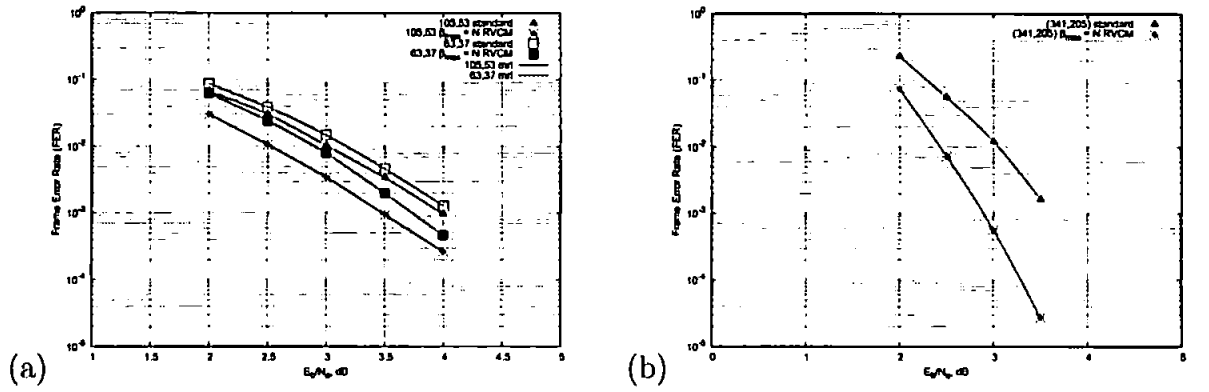


Figure 8.1: Comparison between standard iterative SP decoding and maximum complexity implementation of RVCM $\beta_{max} = N$

the case. Depending on the code, a considerable percentage of the critical bits exhibit correct APP values at all iterations and their corresponding channel outputs are closer to the transmitted ones.

Eb/No[dB]	%Error		
	(63,37)	(105,53)	(85,37)
2.0	75.6%	68.6%	85.4%
3.5	80.6%	65.7%	83%

Table 8.1: Percentage of critical bits which either exhibit at least one erroneous APP estimation during decoding or their corresponding channel output is closer to the wrong symbol

The results of table 8.1, although interesting from a theoretical point of view, can be of little use in practise since of course the error bits are not known and additionally there will be many bits during the standard decoding process that exhibit similar behaviour. Narrowing down the choice for candidate critical bits requires the use of measures that are directly related to the evolution of iterative decoding. Such measures could be based on the information provided by the channel observations and the bits' participation in unsatisfied equations during the standard iterative decoding process.
Figure 8.3 shows histograms based on the ranking of all critical bits in terms of input probability uncertainty and participation in unsatisfied equations through out the iterations. The ranking method (e.g. rank of 1 means that a critical bit exhibited the most unreliable input probability or the highest rate of participation in unsatisfied equations) provides a better visualisation since different non-convergent blocks behave differently and plots of their raw measured values would encourage misleading conclusions. Acknowledging

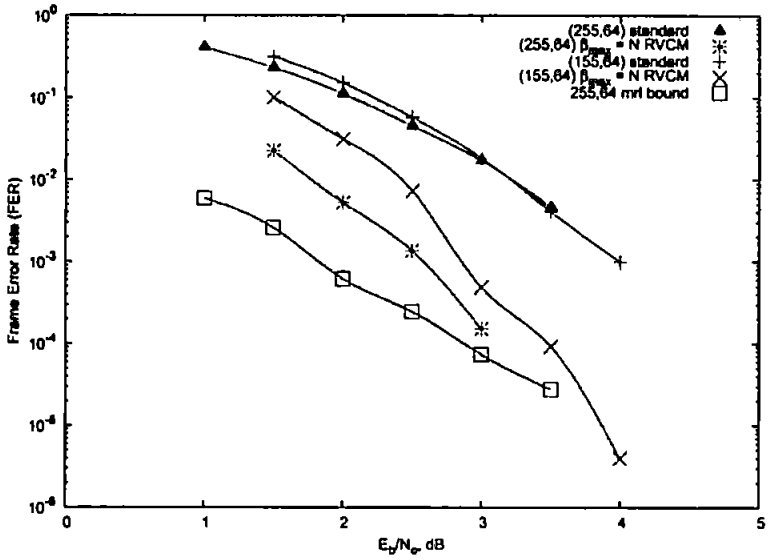


Figure 8.2: Comparison between standard iterative SP decoding and maximum complexity implementation of RVCM $\beta_{max} = N$

the fact that identification of only one of the possible available critical bits in a block is sufficient, the previous graphs can be replotted so that only the highest ranked bit per block is taken into consideration. Graph 8.4 indeed uses this argument and reveals that there exists a clear correlation between the critical property of a bit and the two major measures considered, namely the unreliability of the individual input probabilities and the participation in unsatisfied equations. For some codes this correlation is in favour of one of the two measures.

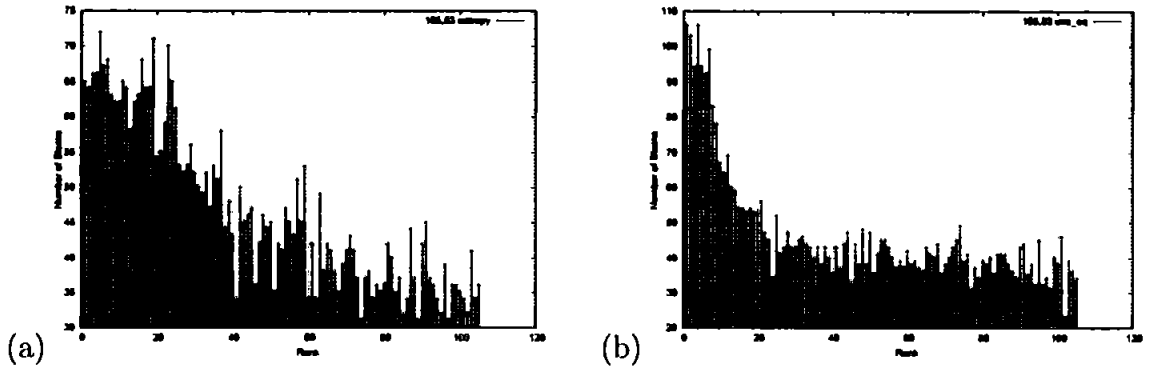


Figure 8.3: Histograms of the ranking of critical bits in terms of (a) input entropy and (b) participation in unsatisfied equations for the 105,53 code at 3.5dB

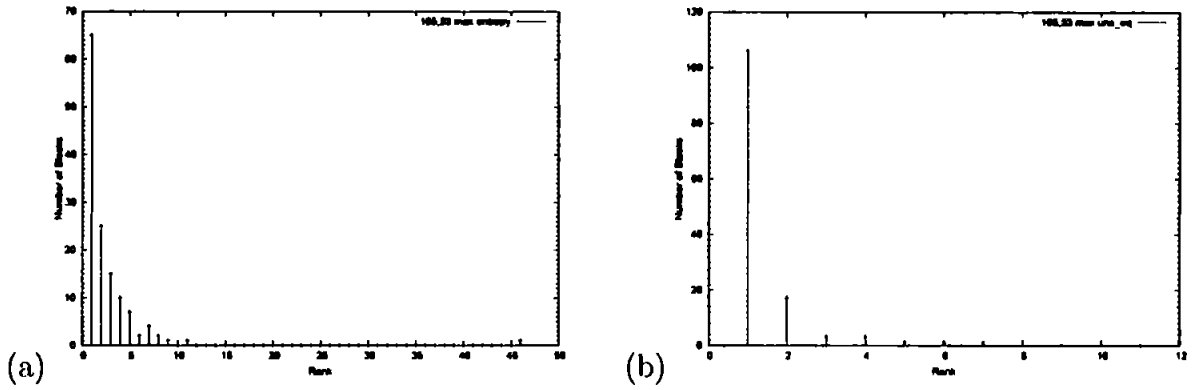


Figure 8.4: Histograms of the maximum ranking of critical bits per block in terms of (a) input entropy and (b) participation in unsatisfied equations for the 105,53 code at 3.5dB

8.3 Candidate critical bits selection criteria

In figure 8.4 it was demonstrated the correlation of the critical bits with two major measures, namely the unreliability (uncertainty) of the input probability and the participation of bits to unsatisfied equations during standard iterative decoding. So, it is possible to improve the throughput of RVCM by reducing the number of selected bits while keeping the performance very close to the optimum. Information about the selection of bits can be obtained either by the received vector or by the behaviour of the processed bits during standard decoding. Two selection criteria are proposed [42], namely

- *Input unreliability criterion (UNR)*
- *Participation in unsatisfied equations criterion (UNSEQ)*

Their description, performance and suitability on different codes are discussed in the following sections.

8.3.1 The UNR selection criterion

The UNR criterion provides the simplest and most straight forward method for selecting candidate critical bits. The selection is based on the information content of the received vector coordinates. For an AWGN channel the conditional probabilities $p(\mathbf{x}|\mathbf{y})$ are given by

$$p(x_i|y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot exp\left[\frac{-(x-y_i)^2}{2\sigma^2}\right] \tag{8.1}$$

The components with the highest entropy (or equivalently with the lowest information content) are given higher priority for the selection of candidate critical bits. The entropy vector $\mathbf{E}$ is defined for each of the coordinates i as

$$E_i = p(0|y_i) \cdot \log_2\left(\frac{1}{p(0|y_i)}\right) + p(1|y_i) \cdot \log_2\left(\frac{1}{p(1|y_i)}\right) \tag{8.2}$$

The UNR criterion offers an efficient way of selecting candidate critical bits and can improve significantly the throughput of RVCM by narrowing down the number of potential critical bits. Figure 8.5 shows that case on the 105,53 cyclic code where selection of only 10 bits based on the UNR criterion is enough to achieve performance almost identical to the optimum. Some more characteristics about the performance of the UNR criterion on

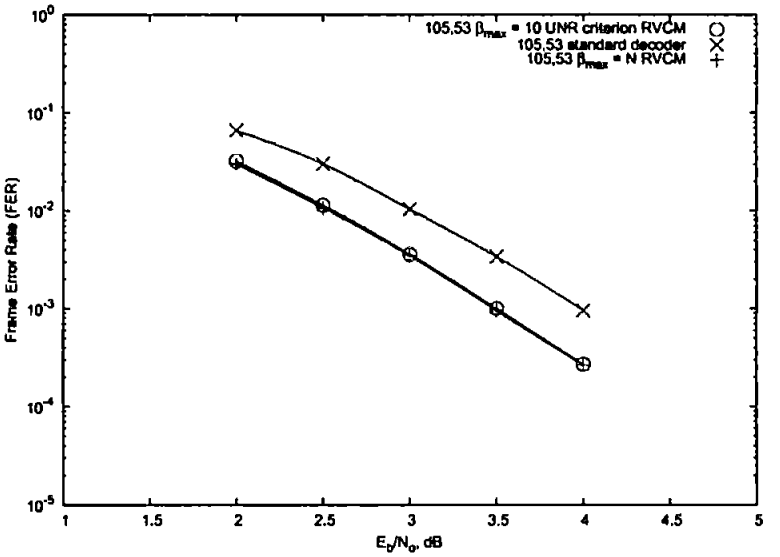


Figure 8.5: Sub-optimum implementation of RVCM based on the UNR selection criterion for a 105,53 code

different codes will be discussed after presenting the second proposed selection criterion.

8.3.2 The UNSEQ selection criterion

The UNSEQ criterion is basically a combination of two measures which gather information in parallel at all iterations of standard decoding.

part 1: The first measure deals with the participation of the bits in unsatisfied equations and bases its results on the standard decoder's output at the end of each iteration. Let $\mathbf{D}^t$

be the set of thresholded APP decisions of the standard decoder at iteration t. Assuming non convergence there will be a set of equations $\mathbf{U}^t$ that remain unsatisfied. The number of occurrences of each bit i at the set of unsatisfied equations $\mathbf{U}^t$ at iteration t can be easily computed as

$$O_t(i) = \sum_{j \in U^t} h_{j,i} \tag{8.3}$$

At the end of the standard decoding procedure the records of the participation of each bit i in unsatisfied equations per iteration $F_1(i)$ can be obtained by

$$F_1(i) = \frac{\sum_{t=0}^{T} O_t(i)}{T} \tag{8.4}$$

where T the number of iterations performed by the standard decoder.
The information from $F_1(j)$ can be combined with a second measure and considerably enhance the performance of the UNSEQ selection criterion.

part 2: This second measure deals with the set of the updated probabilities obtained at the end of each iteration of the SP algorithm. Initially the extrinsic probabilities are computed locally for each one of the parity check equations. These extrinsic probabilities will be called local extrinsics $e_{l_{ji}}$ (the name that was used in the previous chapters) and are associated with all coordinates of the $\mathbf{H}$ matrix where $h_{ji} = 1$. When all parity checks have been processed, the overall extrinsic information e_i for each bit i is calculated as the product of all local extrinsics for this specific bit.

$$e_i(x) = \prod_{\forall j : h_{ji}=1} e_{l_{ji}}(x) \tag{8.5}$$

The updated probabilities upd_{ji} are then calculated for all individual participants by multiplication of the channel probabilities $\mathbf{P}$ with the overall extrinsic information vector $\mathbf{e_i}$, omitting though the corresponding local extrinsic value to prevent positive feedback.

$$upd_{ji}(x) = \frac{P_i(x) \cdot e_i(x)}{e_{l_{ji}}(x)} \tag{8.6}$$

Consider now a thresholded version $\mathbf{upd'} \in [0, 1]$ of the set of updated probabilities. Since the updated values are locally optimised it is guaranteed that they satisfy all parity check equations ($\mathbf{U} = \emptyset$), but on the other hand it is not guaranteed that

$$upd'_{ji} = upd'_{ki} \tag{8.7}$$

In other words we have the case where both equations j and k, that bit i participates, are satisfied but $upd'_{ji} \neq upd'_{ki}$. This implies that one of the two equations involves even number of errors. To decide which of the two thresholded versions of the updated values for i is the erroneous one, a majority decision is applied. If among all w_{c_i} participations of i to parity check equations the associated upd' appears as $x \in [0, 1]$ in less than $w_{c_i}/2$ of these, that particular value x is considered to be in error and all parity check equations j where $upd'_{ji} = x$ are considered to involve an even number of errors. The measure $F_2(i)$ uses the above method to evaluate the participation of bit i in even error equations throughout the iterative process.

Thus, the overall $F(i)$ parameter of the UNSEQ criterion which forms the basis for the selection of the candidate critical bits, is given by the sum of $F_1(i)$ and $F_2(i)$.

$$F(i) = F_1(i) + F_2(i) \tag{8.8}$$

Similarly to the UNR criterion, selecting the bits with the highest $F(i)$ values narrows down the choice for the potential critical bits with minimum losses from the optimum RVCM performance. Figure 8.6 shows the performance of UNSEQ criterion based on the same code that was used in the UNR criterion case. Again, setting the β_{max} parameter as low as 10 it is sufficient to deliver performance arbitrarily close to that of an optimum RVCM scheme that exploits all bits ($\beta_{max} = N$).

The two criteria seem to deliver very similar performance. However, there are major differences between the two and the choice of the right selection criterion depends on the code and the number of candidate bits selected (the parameter β_{max}) as it will be seen in the next subsection.

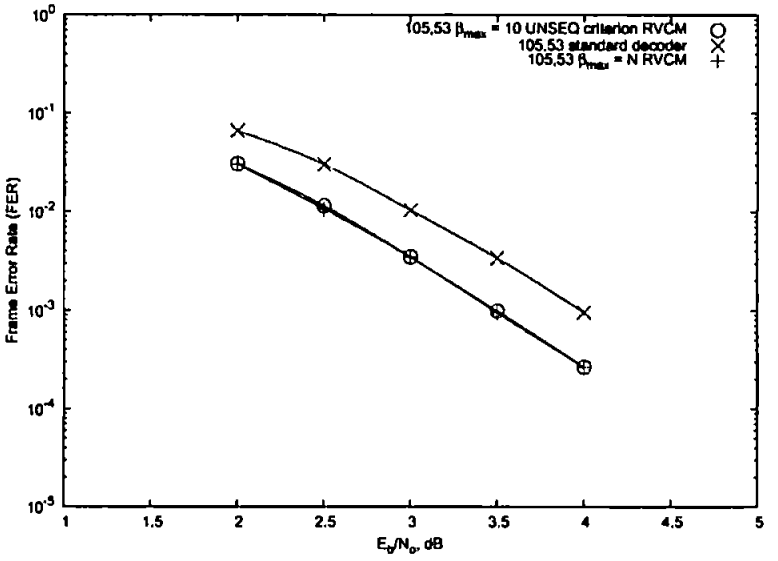


Figure 8.6: Sub-optimum implementation of RVCM based on the UNSEQ selection criterion for a 105,53 code

8.3.3 Suitability of selection criteria

Under certain circumstances, the choice of the appropriate selection criterion might have a large impact on the obtained performance. It has been observed that the main parameters that dictate the use of the UNR or the UNSEQ criterion on regular codes are two:

- The size of the selection (β_{max})
- The code used

For all regular codes tested, the UNSEQ criterion performs consistently better than the UNR criterion when the selection size is kept low. The difference between the two can be quite significant in some cases. In figures 8.5 and 8.6 it can be seen that both criteria perform identically when applied to a 105,53 code for a selection size of 10 bits. However, figure 8.7(a) shows that when the selection size reduces to the minimum ($\beta_{max} = 1$ bit) the difference in the performance offered by each of the two criteria is considerable. In terms of the cumulative distribution function (cdf) the UNSEQ criterion initially exhibits a very steep rise which gradually flattens out as the selection size increases and eventually becomes almost identical to that of the UNR criterion. Similar behaviour has been observed for various other codes typical cases of which are shown in the plots of figures 8.8 and 8.9. Further simulations have also shown that the performance of the two criteria is related to the structure of the code's **H** matrix and the rate. For codes of

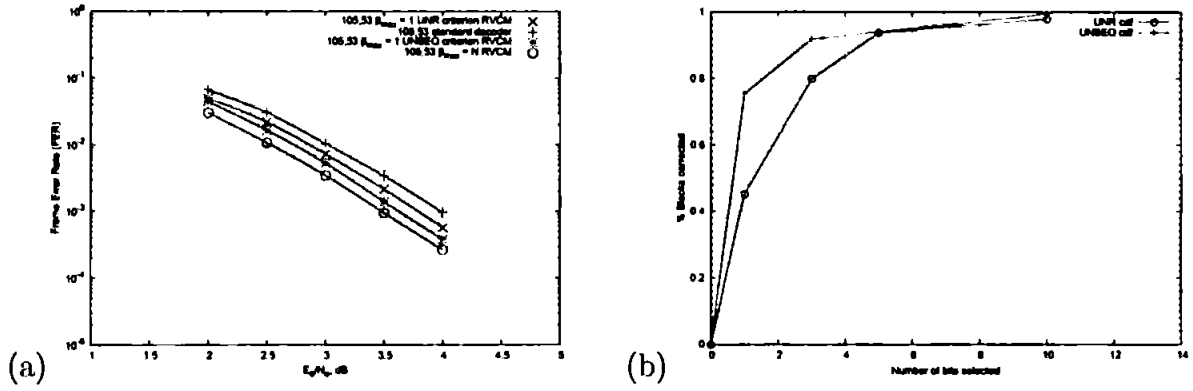


Figure 8.7: a)Comparison of selection criteria and b)cdf curves of selection criteria for the 105,53 code

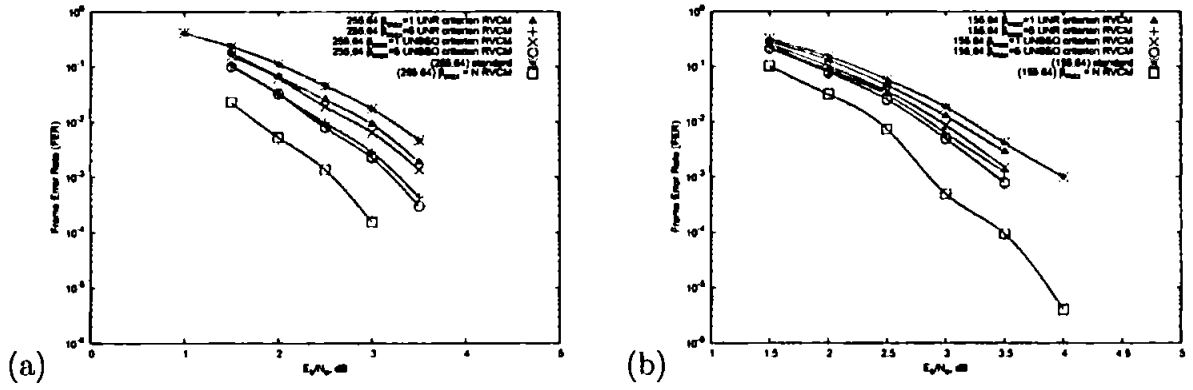


Figure 8.8: Comparison of the UNR and UNSEQ selection criteria for the low w_r a) 255,64 (w_r=5) and b) 155,64 (w_r=5) codes

high row weight (w_r) the UNSEQ criterion performs better than the UNR only when the selection size β_{max} is limited to very small numbers. For codes of low w_r it is possible for the UNSEQ to offer improvement even for slightly larger selection sizes, especially when the rate of the code is not too low. Tables 8.2 and 8.3 summarise the two cases for codes of low and high w_r.

Since in a practical case it would be pretty uncommon to use a selection size of only 1 bit, the UNR method can be definitely considered as the preferable selection criterion for codes with high w_r. On the other hand, the superiority of the UNSEQ method for codes with low w_r can in many cases be maintained even for selection sizes of 5 to 10 bits and can be proved to be useful for systems which can only afford to apply a certain limited selection size.

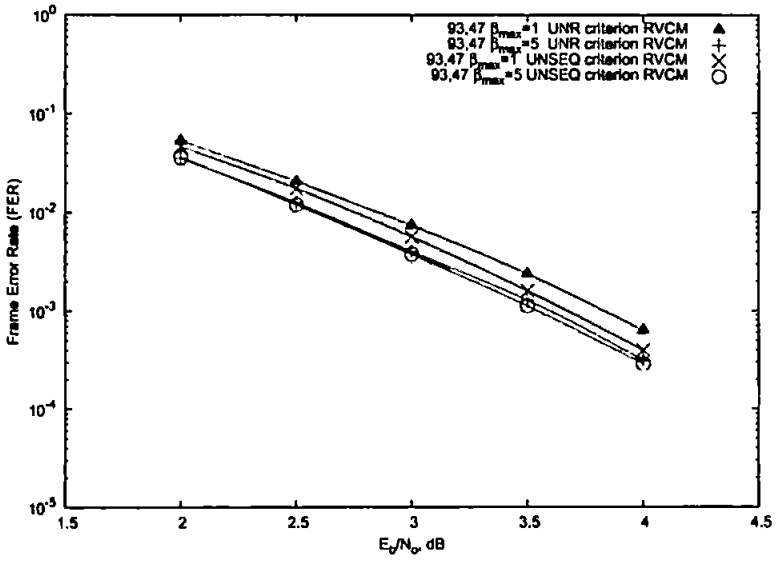


Figure 8.9: Comparison of the UNR and UNSEQ selection criteria for the 93,47 code (w_r=7)

	NUMBER OF BLOCKS CORRECTED											
	105,53				63,16				255,64			
	w_r=7,r=0.505				w_r=4,r=0.254				w_r=5,r=0.251			
Eb/No	1UNR	1UNSEQ	10UNR	10UNSEQ	1UNR	1UNSEQ	3UNR	3UNSEQ	1UNR	1UNSEQ	5UNR	5UNSEQ
1.5	–	–	–	–	–	–	–	–	80	111	187	196
2.0	127	176	257	269	86	126	144	161	134	148	245	246
2.5	138	228	312	310	114	144	171	186	146	197	273	286
3.0	151	252	327	332	115	149	177	192	171	223	295	306
3.5	184	292	352	354	142	169	204	215	201	240	315	326
4.0	201	302	258	260	146	182	230	239	–	–	–	–

Table 8.2: Summary of the performance of the two criteria for low w_r codes.

8.3.4 The effect of correlation to the performance of the UNSEQ selection criterion

In the previous subsection it was shown that the UNSEQ criterion has the potential of offering superior results to these of the UNR criterion, but only when the selection size β_{max} is limited to small numbers. Moreover, the UNSEQ criterion exhibits higher tolerance to larger selection sizes for codes with low w_r. These two observations point out the effect of correlation to the validity of the information provided by the participation of bits to unsatisfied equations during the standard decoding procedure. For codes with high w_r more bits are directly related with each other (by participating in the same parity check equations), thus the correlation among these is stronger. So, many bits' high participation records mirror the participation records of their correlated bits and not their own. Therefore, by increasing the selection size we end up including many correlated

	NUMBER OF BLOCKS CORRECTED							
	255,175				127,84			
	w_r=16,r=0.686				w_r=15,r=0.661			
Eb/No	1UNR	1UNSEQ	5UNR	5UNSEQ	1UNR	1UNSEQ	5UNR	5UNSEQ
1.5	40	46	90	92	45	57	100	97
2.0	67	85	139	139	75	90	138	132
2.5	114	117	202	198	100	123	182	182
3.0	156	176	266	257	148	158	250	232
3.5	208	225	311	304	203	206	296	297

Table 8.3: Summary of the performance of the two criteria for high w_r codes.

selections that do not contribute and waste computational effort. Hence, selection of bits that exhibit strong correlation links with each other, reduces the effectiveness of the UNSEQ criterion as the selection size increases (look at he cdf curve of figure 8.7(b)).
In an attempt to identify and remove correlated selections, two characteristics are taken into account. If two or more bits have the same participation in unsatisfied equations and if in addition to that they coexist at a parity check equation, then only the bit with the highest input entropy is selected while the rest are rejected and not included in the selection list.
The above selection method works for the majority of the cases. Although the improvements whenever they exist are small, they provide an indication that the correlation among the bits is one of the reasons for the reduction of the effectiveness of the UNSEQ criterion for large selection sizes and especially for codes of high w_r. Table 8.4 compares the results of this technique with the straight forward UNSEQ criterion for various codes.

	NUMBER OF BLOCKS CORRECTED					
	85,37		93,47		105,32	
Eb/No	5UNSEQ	5UNSEQuncor	5UNSEQ	5UNSEQuncor	5UNSEQ	5UNSEQuncor
1.5	130	134	–	–	–	–
2.0	202	210	248	254	247	247
2.5	226	238	302	300	298	303
3.0	304	320	337	344	350	352
3.5	355	357	355	361	380	373
3.5	–	–	364	364	414	414

Table 8.4: Comparison of the UNSEQ criterion with the same criterion when correlation constraints are applied

8.3.5 Convergence properties and selection criteria

Powerful codes of high d_{min} generally seem to converge worse at low SNR. Their convergence capability improves though at moderate and high SNR values where they exhibit very steep error rate reduction and low error floors. However, at all SNR regions it is a

common characteristic that whenever convergence is achieved it is highly probable that the converged solution is the maximum likelihood. This is equivalent to saying that high d_{min} codes exhibit a very low number of mrl blocks for a wide range of SNR. All these indicate that when such a code fails, the chances that the decoded output represents a valid solution are very low. That is the case for the high d_{min} 155,64 Tanner code. Returning to the problem of selecting the candidate critical bits, the selection can be based on the non valid decoded vector of the decoder by picking up those bits with the highest participation in unsatisfied equations. Figure 8.10 compares this method with the existing pre-described RVCM selection criteria and shows that selecting bits based on the non-valid decoded output of the decoder does not provide any gain, but on the contrary performs slightly worse.

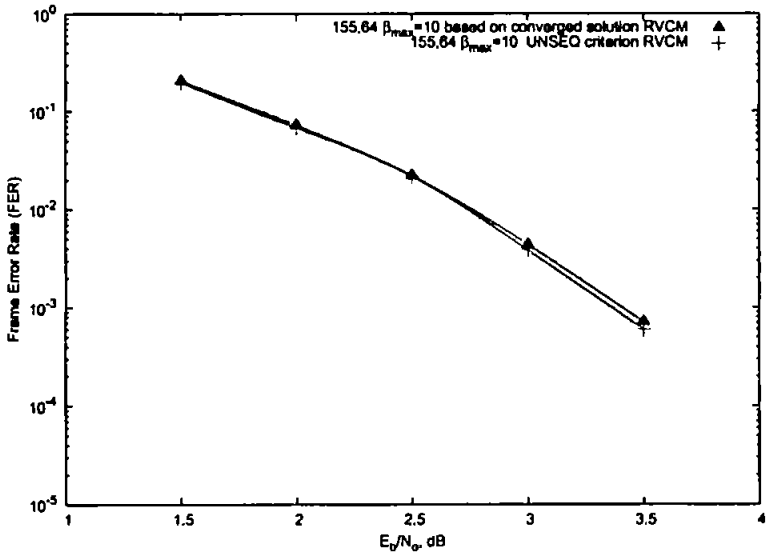


Figure 8.10: Comparison of the selection method based on the decoded output for the Tanner 155,64 code

8.4 An equation-wise approach for approximating the positions of critical bits

Up to this point the investigation for the search of critical bits has focused on characteristics and measures of the bits themselves. In this section the same problem is approached from a different angle, i.e. the search focuses on parity check equations that gather high chances of containing at least one of the critical bits.

8.4.1 High Failure (HF) parity check equations

We define as HF those equations that during the iterative procedure remain unsatisfied for more than $\phi \cdot 100\%$ of the performed iterations. It has been observed that at all times there is participation of at least one critical bit with $\phi \geq 0.5$. Figure 8.11 shows histograms of the maximum value of the parameter ϕ among those equations in which a critical bit participates. This result can be used as a way to narrow down the number of candidate bits by excluding any bit that does not participate in at least one equation of $\phi \geq 0.5$.

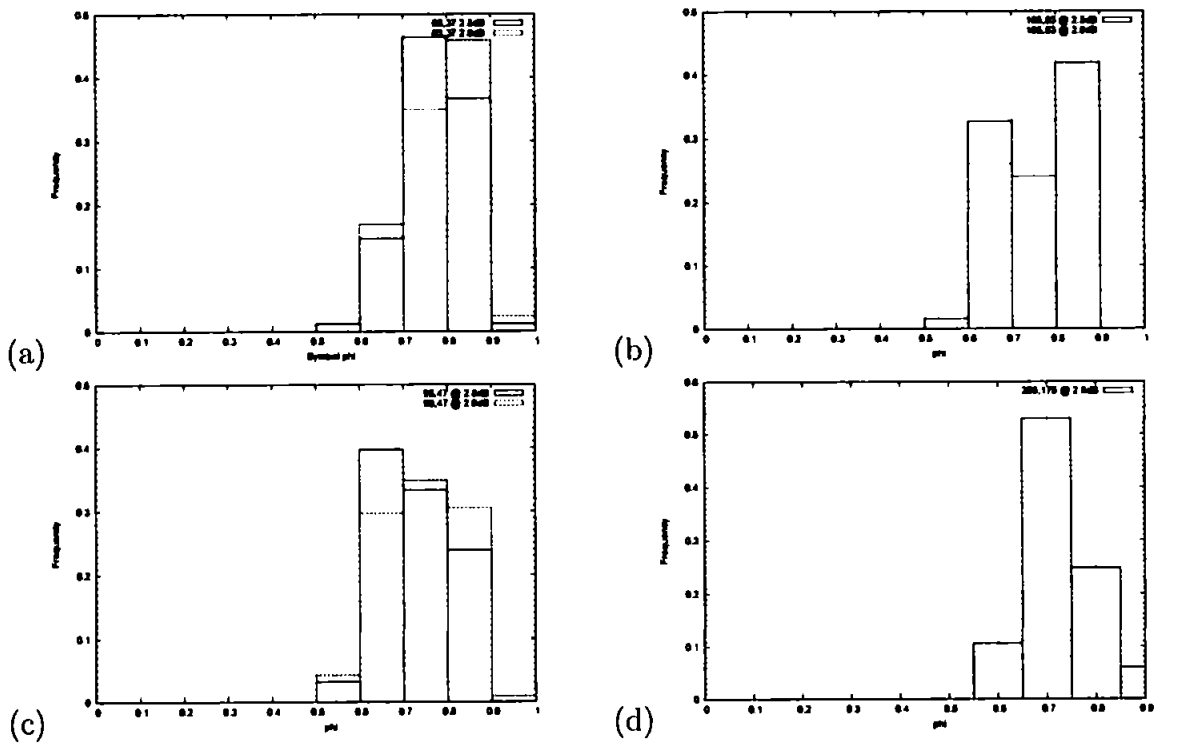


Figure 8.11: Histograms of the highest per block value ϕ among parity check equations that contain at least 1 critical bit.

To assess any gain that can be achieved by this reduction on the number of candidate critical bits, we need to know how many equations are associated with high and low ϕ values. As a typical case, figure 8.12 shows the distribution of the number of HF equations ($\phi \geq 0.5$) for the 105,53 LDPC code when operating at 2.5dB. The large number of equations that fulfil this criterion makes this method inefficient.

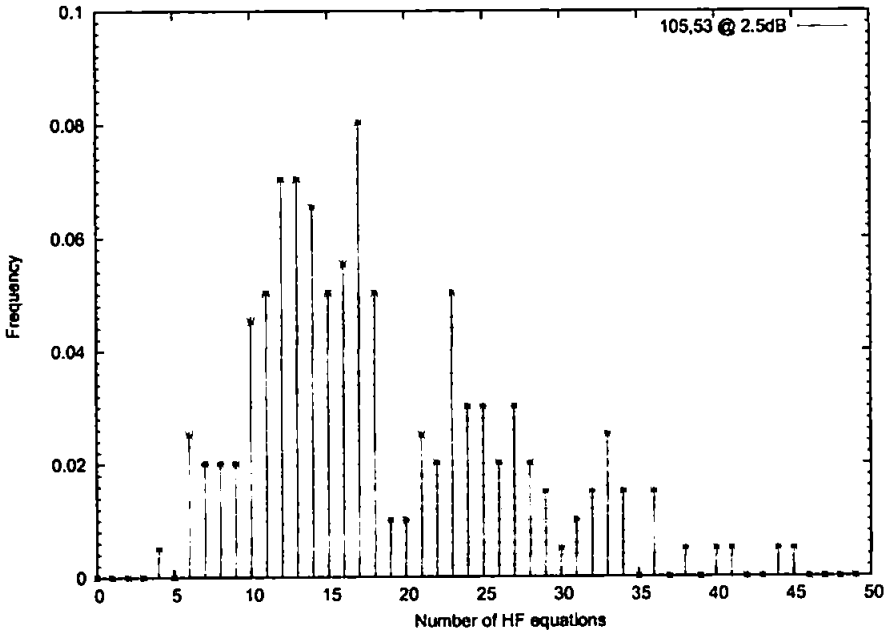


Figure 8.12: Distribution of the number of HF equations among 200 prior non-convergent blocks that have been successfully decoded with RVCM

9 RVCM on Turbo Codes

This chapter investigates application of RVCM algorithm on turbo codes [45, 44]. It is shown that significant gains are achieved for all tested codes on PCCC and SCCC [43] schemes. Furthermore, application of RVCM at high SNR minimises the losses of turbo decoding with respect to the ML. It is also demonstrated that the optimum RVCM performance can be closely approximated by keeping the parameter β_{max} much lower than the number of systematic bits when certain properties of the critical bits, such as the entropy of the input probabilities, the participation of the systematic bits in unsatisfied equations or the induced correlation between the bits, are taken into account.
Finally, as in the case of LDPC codes, it is shown that the improvement achieved by RVCM is closely related to the reduction of pseudo-codewords. This idea is enhanced by the fact that modification of the strongest contributor to the correlation of the received vector with pseudo-codewords accounts for most of the improvement achieved by RVCM.

9.1 Performance of RVCM on turbo codes

Figure 9.1 demonstrates the improvement imposed by RVCM on various turbo schemes in terms FER and BER performance. In all cases the coding gain varies from 0.15 to 0.25dB. For moderate to high SNR the RVCM performance gets much closer and even achieves the lower mrl bound.
Investigation of the characteristics of the critical bits in terms of their input probability entropy and their participation in unsatisfied equations, reveals that for the great majority of the blocks there exists at least one critical bit that resembles high values for one of the two measures. If all systematic bits are ordered in terms of any of the two measures, then as figure 9.2 shows the highest ranked bit is critical in more than 95% of the blocks that are correctable by RVCM. This precentage reduces at lower SNR but still selecting the candidate critical bits with respect to any of the two measures proves to be much more efficient than random selection. The two measures can be used in favour of the selection for candidate critical bits in the form of the UNR and UNSEQ criteria that were described in the previous chapter for LDPC codes.

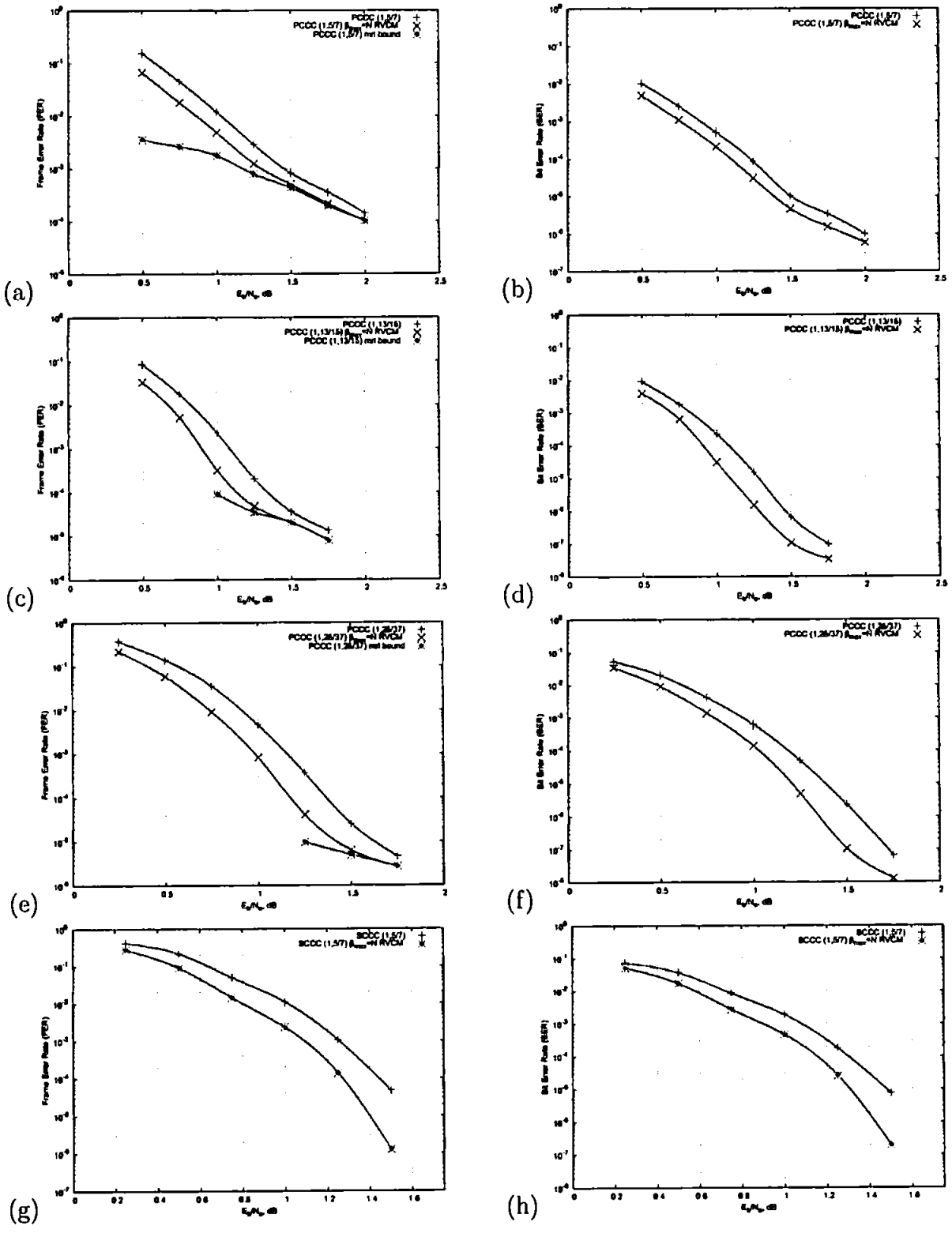


Figure 9.1: RVCM performance on turbo codes. PCCC(1,5/7) (figures (a) and (b)), PCCC(1,13/15) (figures (c) and (d)), PCCC(1,25/37) (figures (e) and (f)) and SCCC(1,5/7) (figures (g) and (h)). For (a)-(d) N=1500, (e) & (f) N=1503, (g) & (h) N=2000. S-interleaving and tail-bite termination has been used in all cases

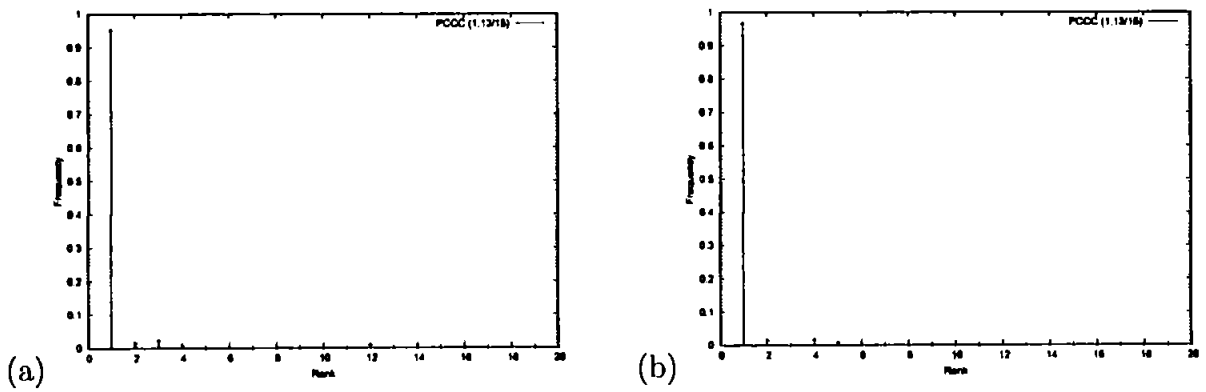


Figure 9.2: Frequency plots of the percentage of blocks that are corrected by RVCM versus the highest rank among critical bits. Bits are ordered in terms of a) Input entropy and b) participation in unsatisfied equations. Results are based on a PCCC, RSC(1,13/15) with S-interleaver at 0.8dB

9.1.1 The UNR and UNSEQ selection criteria as an approximation to the optimal RVCM performance

Figures 9.3 and 9.4 compare the results obtained by the two selection criteria versus the RVCM maximum performance ($\beta_{max} = N_{syst}$) when applied on turbo schemes. It can be seen that with $\beta_{max} \leq 0.1 N_{syst}$ it is possible to achieve performance in close resemblance to the maximum.

Comparison between the two selection methods (UNR and UNSEQ) reveals that the difference in performance is tiny. Moreover, it has not been investigated any relation between the delivered performance of the two criteria and the selection size (β_{max}) as in the case of LDPC codes.

9.1.2 Correlation coefficient measure and critical bits

By tracking the values of the extrinsic information for all iterations of the standard turbo decoder it is possible to obtain the correlation coefficients between any pair of bits in a code of block-length N. If the extrinsic probability of bit i for a decision of b at iteration t is $e_i^{(t)}(b)$ then

$$\Delta_{it} = \begin{cases} 1 & \text{if } e_i^{(t)}(b) \geq e_i^{(t-1)}(b) \\ 0 & \text{if } e_i^{(t)}(b) < e_i^{(t-1)}(b) \end{cases} \tag{9.1}$$

Keeping a record of the fluctuations of the extrinsic probabilities throughout the iterations incorporated into the matrix Δ, it is easy to measure the similarities between any two bits.

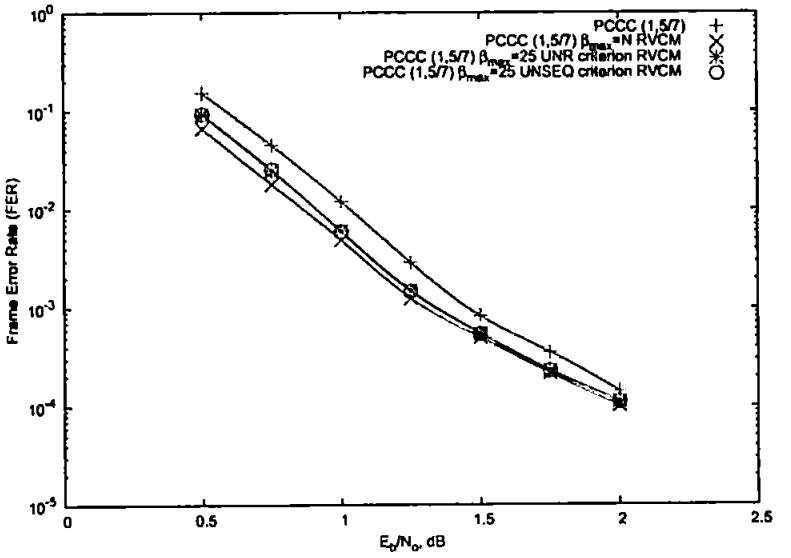


Figure 9.3: Performance comparison of the UNR and UNSEQ selection criteria for a PCCC (1,5/7) scheme, when $\beta_{max} = 25$

The similarities information is kept into the matrix $\mathcal{S}$ and can be defined for $0 \leq t \leq T$ (where T is the maximum number of iterations) as

$$\mathcal{S}_{ij} = \begin{cases} \mathcal{S}_{ij} + 1 & \text{if } \Delta_{it} = \Delta_{jt} \\ \mathcal{S}_{ij} + 0 & \text{if } \Delta_{it} \neq \Delta_{jt} \end{cases} \tag{9.2}$$

The correlation coefficient ϱ_{ij} is then

$$\varrho_{ij} = \left| \mathcal{S}_{ij} - \frac{T}{2} \right| \tag{9.3}$$

which means that pairs of bits that exhibit identical or completely opposite behaviour are considered as maximally correlated. On the other hand, bits that exhibit identical behaviour just for half of the total iterations T can be considered totally uncorrelated. Let's see now the relation between the correlation coefficient and the critical ability of a bit. Normally it should be expected that bits which exhibit on average the highest correlations among the systematic bits would gather more chances for being critical. However, as figure 9.5 shows, exactly the opposite happens i.e. critical bits are associated with systematic bits of low average correlation coefficient.

Using this measure as a selection criterion for candidate critical bits can in many cases offer slightly superior results with respect to the UNR and the UNSEQ criteria. It can

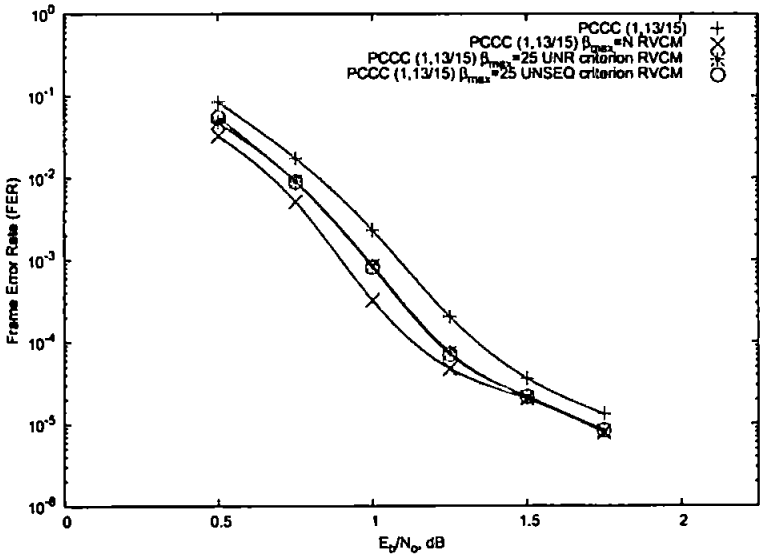


Figure 9.4: Performance comparison of the UNR and UNSEQ selection criteria for a PCCC (1,13/15) scheme, when $\beta_{max} = 25$

not be determined though when selection based on the correlation coefficients should be preferred. Even in the cases where this selection method works with better results, the undergone improvement is not sufficient to justify the significant computational load over the other two selection criteria.

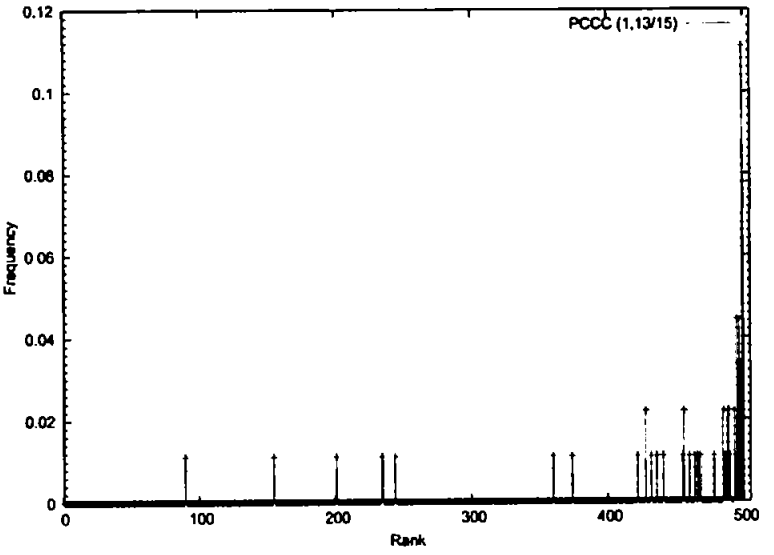


Figure 9.5: The systematic bits are ranked in terms of their correlation coefficient value ϱ. Lower ranked bits are most often among the critical bits

9.1.3 Achieving efficient implementation of RVCM by reduction of the performed iterations

In chapter 7, in the discussion about the RVCM algorithm, it was shown that the RVCM is capable of converging to the ML solution in only a few iterations. The number of iterations to be performed so that all or most critical bits can be identified depends on the SNR and the constituent codes used. The results of chapter 7 can be used to further reduce the computational complexity of the algorithm (by reducing both β_{max} and the maximum number of iterations performed per trial, denoted as θ), or as an alternative method to the selection criteria UNR and UNSEQ (by keeping β_{max} high and reducing θ).

A good way to compare the different approaches is to use as a measure of embedded complexity the number of additional iterations for a given β_{max} and θ, and as a reference the number of iterations T performed by the standard decoder. For example, a system with $T = 50$, $\beta_{max} = 10$ and $\theta = 5$ would have an additional complexity factor $\eta = \frac{2\beta_{max}\theta}{T} = 2$.

As a comparison figures 9.6 and 9.7 display the performance of schemes with the same complexity factor η, achieved either by reducing the selection size (β_{max}) or by reducing the number of performed iterations by RVCM (θ). The results reveal that by increasing β_{max} and at the same time reducing θ in such a way that the complexity factor η remains

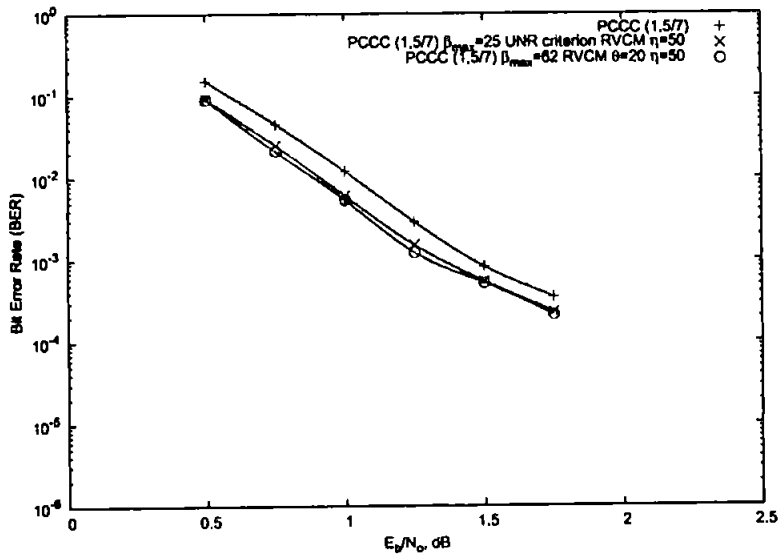


Figure 9.6: Comparison of different but of equal complexity ($\eta = 50$) implementations of RVCM on a PCCC (1,5/7) scheme

constant, the performance is slightly better. However, the tricky task in this approach is the optimum tuning of the two parameters β_{max} and θ for a given code and SNR, so that maximum performance is delivered.

9.2 The relation between pseudo-codewords and ML decoding for turbo codes

Following the ideas of chapter 6, in this section it is attempted to demonstrate that (similarly to LDPC's) the high correlation between pseudo-codewords and the received vector is one of the main reasons for the convergence problems of the iterative decoding algorithm. An analysis similar to that of the SP algorithm in chapter 6 seems to be impractical for the turbo algorithm, basically because of the difficulty of braking up the complex turbo decoding function into small localised equations of only a few variants each. Alternatively, we will investigate the validity of the observations that were made in chapter 6; these are:

- With high probability, the iterative decoder will fail to correct those blocks for which the received vector is not correlated strongly enough with a valid codeword.
  - For these blocks optimum MAP decoding would offer decisions of relatively low reliability

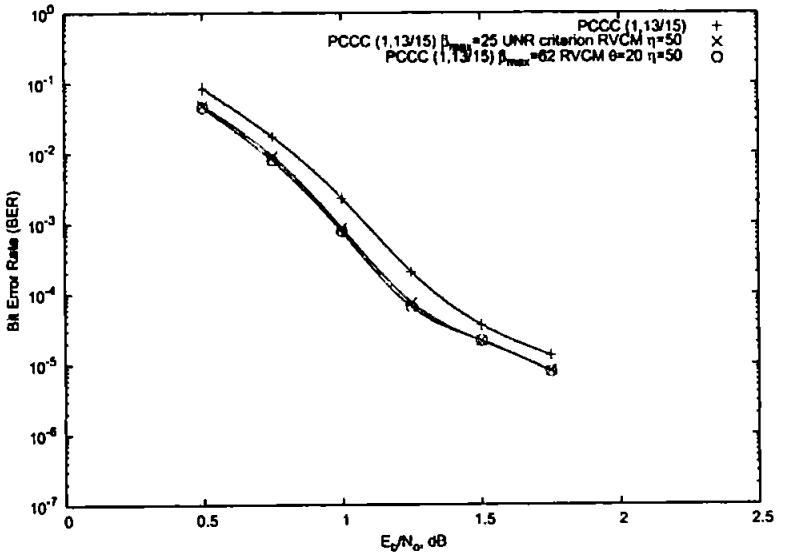


Figure 9.7: Comparison of different but of equal complexity ($\eta = 50$) implementations of RVCM on a PCCC (1,13/15) scheme

- The bit with the least reliable decision of the MAP decoder is with high probability a critical bit

Figure 9.8 depicts the histograms of the average per block reliability of the optimal MAP decoder decisions. Histogram (a) displays only those blocks for which the iterative decoder has successfully converged to the ML solution, while (b) includes only blocks for which the iterative decoder has converged to a solution other than the ML. From these it can be deduced that the iterative decoder will mostly fail to decode optimally those blocks that the optimal MAP decoder would decode with low reliability (although correctly). The lower the average reliability of the optimal MAP decisions, the higher the chances that the turbo algorithm will fail to converge to the optimal solution and vice versa. When the average reliability of the MAP decoder decisions is low, it can be considered as a sign that none of the codewords exhibits high correlation with the received vector and therefore, there is no codeword that clearly dominates. The similarity with the histograms obtained from the same experiment on LDPC codes encourages us to follow the same sequence of thoughts and assumptions. We can therefore deduce the existence of pseudo-codewords in turbo codes and link the incapability of the turbo algorithm to converge to the ML solution, with the high correlations existing between the pseudo-codewords and the received vector. When the correlation between codewords and the received vector is not strong enough (low reliability of the optimum MAP decoder decisions) the chances that one or more pseudo-codewords will compete the codewords in the final decision of

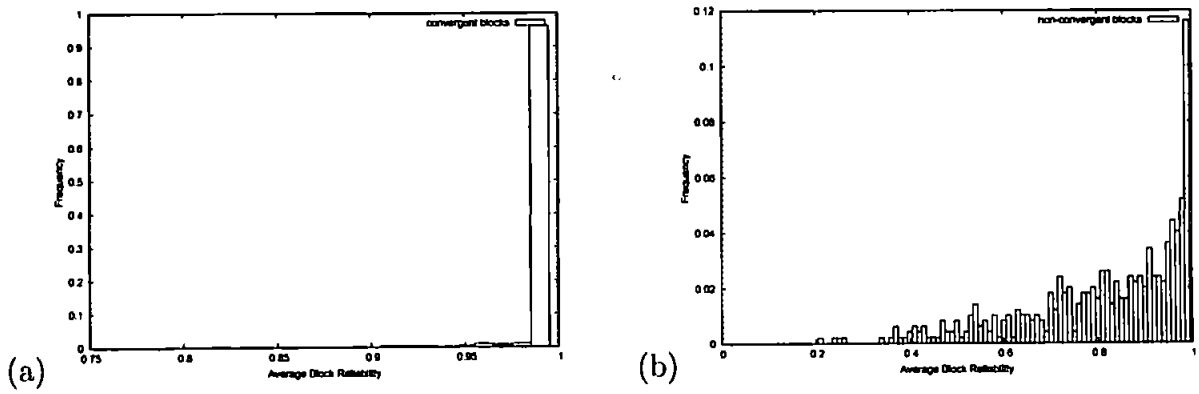


Figure 9.8: Histograms of the average reliability of the decisions of the optimum MAP decoder for blocks that (a) have been optimally decoded by iterative decoding and (b) have been decoded sub-optimally by the iterative decoder. Results based on a PCCC (1,5/7), N=48 scheme

the turbo decoder increases and accordingly, the probability that the algorithm will not converge to the ML solution increases too.

Similarly, the least reliable bit decision of the MAP decoder can be considered as the weakest contributor to the correlation of the codewords with the received vector. In figure 9.9 it is shown that for a very high percentage of the processed blocks this bit is among the critical bits. Hence, the optimum RVCM performance ($\beta_{max} = N_{syst}$) would be very closely approximated with minimum effort if there could be an efficient way to determine the bit position that would exhibit the lowest reliability when optimally decoded.

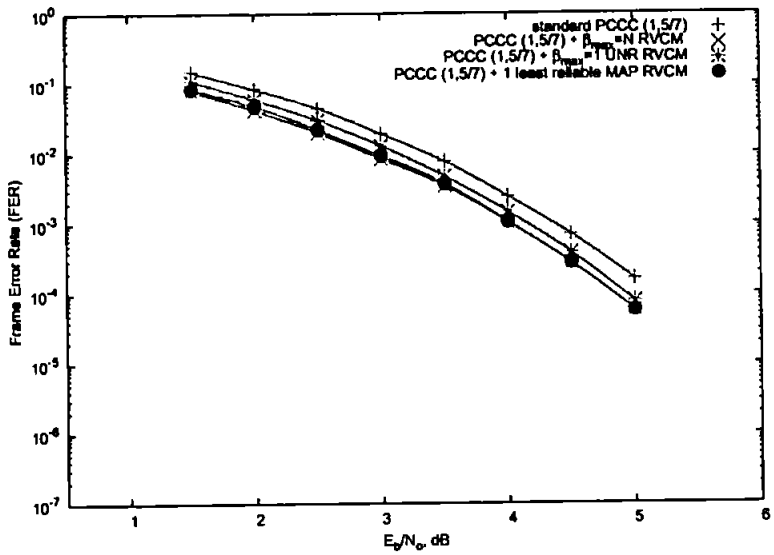


Figure 9.9: Comparison between standard turbo decoding, RVCM $\beta_{max} = 1$ with UNR criterion, optimum ($\beta_{max} = N_{syst}$) and RVCM $\beta_{max} = 1$ using the least reliable bit of the MAP decoder PCCC (1,5/7), N=48 scheme

10 Comparison of RVCM with other convergence improvement methods

In this chapter the RVCM algorithm is compared with alternative methods that have been developed by various researchers for improving the convergence properties of LDPC codes. The comparison takes into consideration the offered performance gain, but also the complexity involved.
The first scheme is the Ordered Statistics Decoder (OSD) which is based on the Dorsch algorithm [17] and was re-discovered by Fossorier [25, 26].
The second scheme is the *information correction* method presented by Varnica et. al. in [70] and independently by Pishro-Nik et. al. in [48]. Both methods use a similar to RVCM approach, as they are based on modification of the received vector.

10.1 RVCM compared to the ordered statistics decoder (OSD)

Detailed description of the OSD can be found in chapter 6, where the basic idea of the particular scheme has been used for the implementation of a list decoder. As a brief description, the **H** matrix is reordered in such a way that the $N-k$ bits that have been received with the least reliable values from the channel (hence gather higher chance of being in error) can be obtained from the k most reliable bits by simply encoding them. As long as the k encoded bits contain no errors, the decoder will decide the correct codeword. If one or more errors exist within the k reliably received bits, it is necessary to flip one or more of those so that the correct codeword is obtained. An *order-*κ OSD assumes κ errors among the k most reliable bits, and since the error positions are unknown, all $\binom{k}{\kappa}$ possibilities should be tried. The Euclidean distance of the decoded vector (from the received vector) after each trial is stored and at the end of the procedure the decoder output that minimises the Euclidean distance is chosen as the most probable decision.
The performance and complexity of the OSD method is determined by its order κ. For low orders there is certainly an advantage over the classical SP decoding in terms of complexity, but with the exchange of higher error rates. In order to overcome the performance

of standard SP decoding it is necessary to increase the OSD order with the associated price in complexity of course. So, an order-2 OSD would require $1 + \binom{k}{1} + \binom{k}{2}$ encoding operations (trials). Note that for each trial there is an extra computational load dealing with the manipulation of the reordered $\mathbf{H}$ matrix to the echelon form so that encoding is feasible.

As figure 10.1 shows for a particular code, the OSD of order-2 improves the standard performance of the SP algorithm. To achieve higher improvement over SP decoding it is necessary to increment the order of OSD. Each increment of the OSD order by 1 is followed by an exponential rise in the complexity of the process. The RVCM algorithm can achieve higher gain much more efficiently with just a fraction of the complexity of the OSD. Application of RVCM with $\beta_{max} = 1$ (only 2 additional operations) overcomes the performance of an order-2 OSD and at the same time operates 170 times faster in terms of CPU time. With $\beta_{max} = 10$ RVCM achieves the ML performance of the 105,53 cyclic code with less than 1% of the CPU time required by the order-2 OSD.

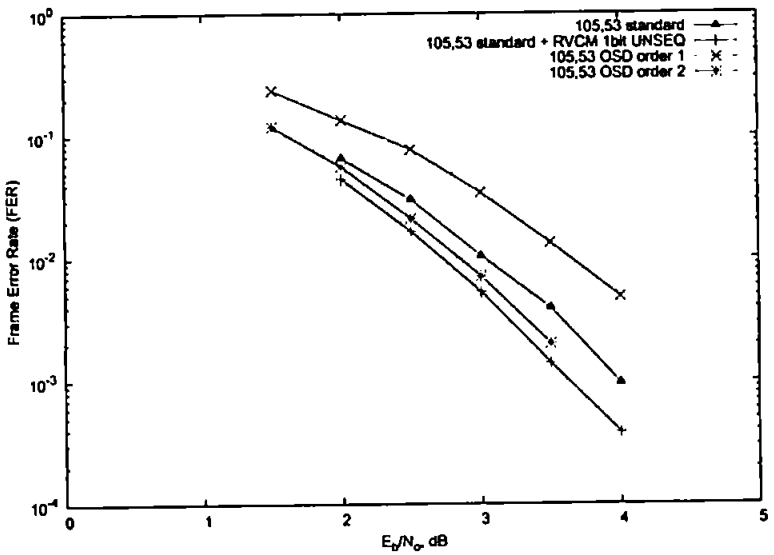


Figure 10.1: Comparison of RVCM versus OSD

Alternatively, the OSD method can be used together with SP decoding like RVCM does, i.e. only for blocks that the SP algorithm has failed to converge. The performance of the two schemes then varies from code to code. In the case of the (255,64) code, RVCM can offer lower error rates and faster implementation. For the (105,53) the performance and implementation time is slightly in favour of RVCM while for the (127,84) code an OSD of order 3 offers better performance than an optimum implementation of RVCM ($\beta_{max} = N$) but with the cost of significantly higher complexity.

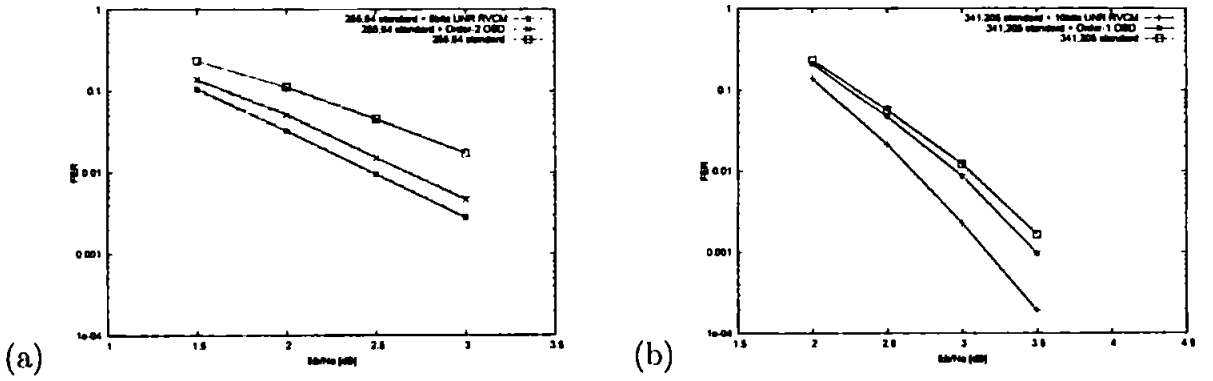


Figure 10.2: Comparison of RVCM versus OSD for (a) LDPC (255,64) and (b) LDPC (341,205)

Generally, a complexity/performance comparison of the two schemes is difficult. However, RVCM can be considered as a more flexible scheme that can adapt much easier to the demands of a system since the additional complexity increases linearly with the β_{max} parameter. On the other hand, incrementing the order of OSD for achieving a certain target performance adds up considerably more computational load that might not worth the actual improvement in performance.

10.2 RVCM compared to information correction decoding method

The information correction method is based on the fact that bit positions with unreliable input probabilities and participation in many unsatisfied equations, gather more chances of being decoded in error [70]. After standard SP decoding and if the SP algorithm has failed to converge, bits that fulfil the above two conditions (unreliable input probability and high participation in unsatisfied equations) are considered suspicious of being in error and are given high priority for information correction.

Considering a non-convergent block, at the end of the final iteration of the standard decoder there will be a set of unsatisfied equations $\mathbf{U} = \{u_j : \mathbf{h_j} \cdot \mathbf{x^T} \neq \mathbf{0}\}$, where $\mathbf{x^T}$ is the transposed output vector of the standard decoder and $\mathbf{h_j}$ is the j^{th} row of the parity check matrix $\mathbf{H}$. The degree $d(v)$ of variable nodes (bits) $v \in V$ is defined as

$$d(v) = \sum_{u_j \in \mathbf{U}} h_{u_j,v} \tag{10.1}$$

Hence, $d(v)$ is simply the number of participations of bit v to the set of unsatisfied equations. The set of suspicious bits $\mathbf{S}$ that are candidates for information correction is then

$$\mathbf{S} = \{v : d(v) = d^{max}\} \tag{10.2}$$

where $d^{max} = max_{v \in V} d(v)$. For two bits v and i with the same degree, higher priority is given to the bit with the higher uncertainty on its input probability.

The next step is to apply the information correction at the input probabilities of the suspicious bits $s \in \mathbf{S}$ by setting them to ± 1 and restarting the iterative decoding procedure. For each trial the Euclidean distance of the decoder's decision is stored, and at the end the decision vector that is associated with the minimum Euclidean distance is chosen as the most probable solution.

The difference of this method to the RVCM algorithm lies on the way and the order that the changes are applied to the received vector. Figures 10.3 and 10.4 show flow graphs of the modification mechanisms for the two algorithms. In RVCM only a single bit is modified at any time and the maximum number of decoding operations (trials) for a given β_{max} is calculated as $2 \cdot \beta_{max}$. Therefore, the relation between complexity and β_{max} is linear. On the other hand, in the case of the information correction scheme every increment of the selection parameter j causes significant rise in the maximum number of decoding operations to be performed, and the relation between the two resembles a geometric pro-

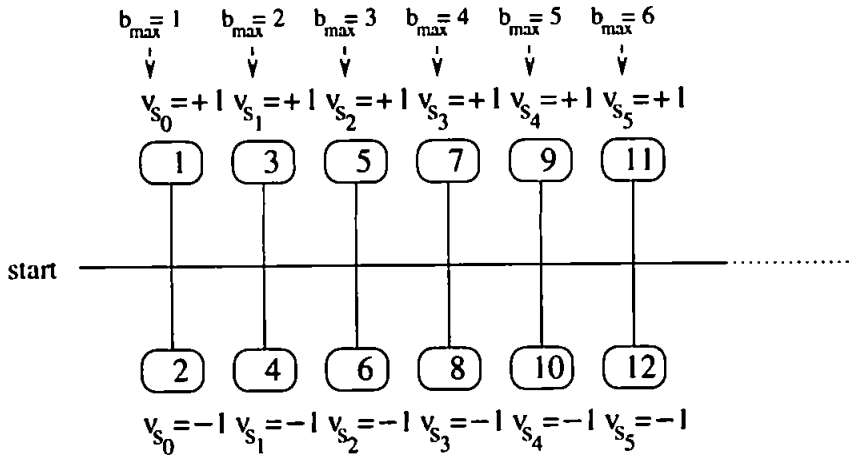


Figure 10.3: Flow graph of performed trials for the RVCM algorithm

gression (*max number of operations* $=2^{j+1}$). The information correction method assumes that if the bit with the highest priority in the **S** list (v_{s_0} with $d(v_{s_0}) = d^{max}$ and the least reliable input probability among any other bits v with $d(v) = d^{max}$) is not capable to converge the decoder to the ML decision when modified, then no other bit is capable of achieving that by itself. Consequently, combinations of more than one bits from the **S** list are tried. The performance of RVCM proves that most of the times, it is worth continuing the search for a single bit instead of unnecessarily moving on to trials of long combinations.

Figures 10.5 and 10.6, compare the performance of the two schemes when applied on the cyclic (255,64) and the Tanner(155,64) LDPC codes. For a fair comparison, the selection parameters β_{max} and j for the two schemes have been chosen in such a way that the complexity involved, in terms of additional decoding operations, is exactly the same. In both graphs RVCM algorithm performs significantly better as it is capable of delivering superior performance with the same additional complexity. The advantage of RVCM is mainly on the moderate to higher SNR region where the gain over its opposed algorithm reaches 1dB for the 255,64 code.

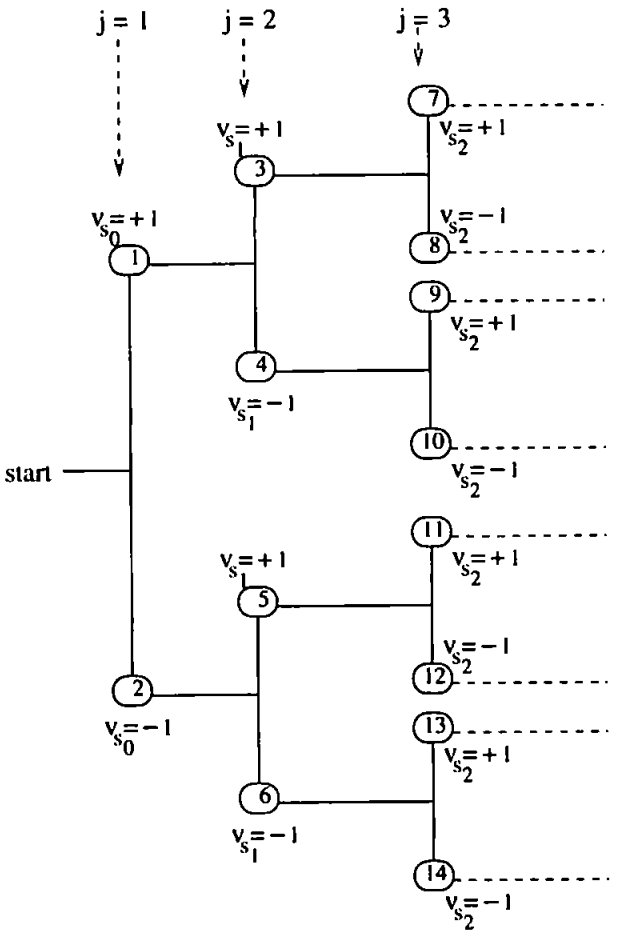


Figure 10.4: Flow graph of performed trials for the information correction method

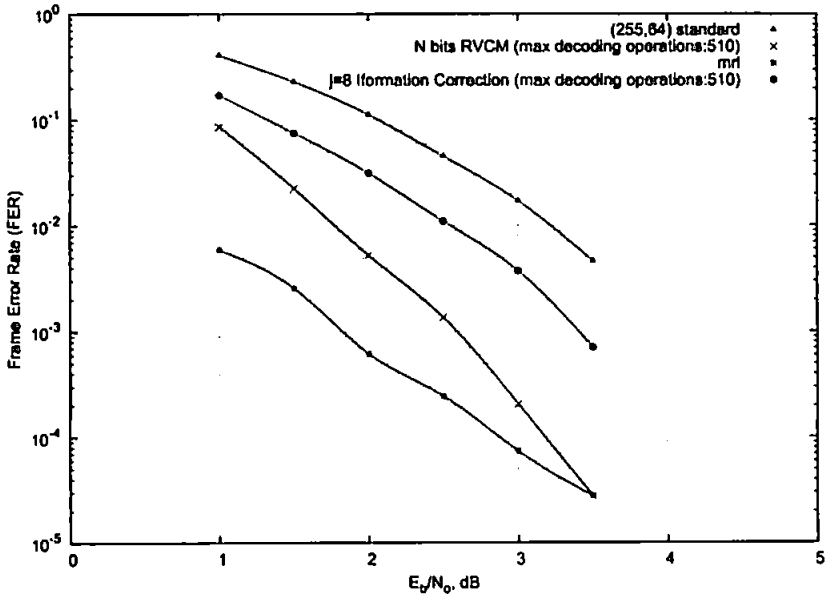


Figure 10.5: Comparison of RVCM with the Information Correction algorithm on a cyclic (255,64) LDPC code

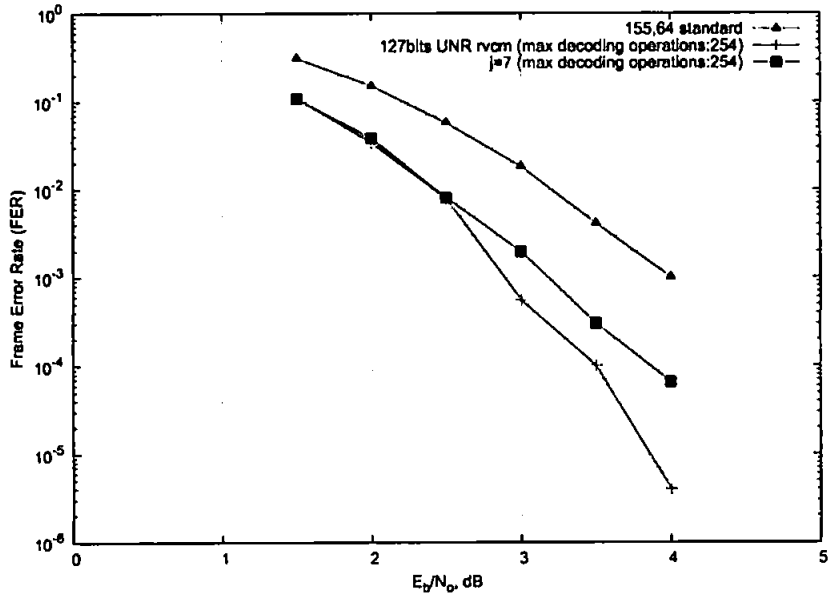


Figure 10.6: Comparison of RVCM with the Information Correction algorithm on a Tanner(155,64) LDPC code

11 Conclusions and directions for future research

In this thesis, the convergence problem of iterative decoders was investigated. The operation of the iterative decoding algorithms for turbo codes and LDPCs was explained. The convergence problem of iterative decoders was initially introduced and it was linked with the concept of correlation and short cycles. Following analytically the operation of the SP algorithm for LDPC codes it was shown the mechanism for the generation of pseudo-codewords and their direct effect on the sub-optimality of iterative decoders. Furthermore, the ability of the iterative algorithm to decode optimally was related to the correlation of pseudo-codewords with the received vector. A new algorithm was presented, capable of providing significant coding gain with affordable added complexity. Its exceptional performance was attributed to its ability to reduce the number of pseudo-codewords and to also minimise the effect of unequal scaling among the probabilities of the bit nodes. The algorithm was applied with remarkable success on turbo and LDPC codes, and it was also compared to alternative approaches. The following is a summary of the main results and contributions.

Analytical description of the mechanism responsible for the creation of pseudo-codewords during SP decoding of LDPC codes

- The iterative decoder was treated as an optimum algorithm that operates on a vector space different than the set of valid codewords $c \in \mathcal{C}$. That is equivalent to the conclusions of [27] due to which the iterative decoder performs optimally on the code's cycle-free computation graph.
- Following the ideas of [3, 77] where the two main types of distortion imposed by SP algorithm are defined, the relation that determines the number of pseudo-codewords as a function of the parity check matrix characteristics (column and row weight) was specified. From the same relation it was deduced that the existence of pseudo-codewords is an inherent characteristic of any iteratively decoded code.

Correlation of pseudo-codewords with the received vector

- The level of correlation between pseudo-codewords and received vector was identified as the main cause for the convergence problems of iterative decoders.
- The level of correlation exhibited between the two was evaluated indirectly, by investigation of the received vector's correlation with the valid codewords. The latter can be estimated by the reliability of the optimum MAP decoder decisions.
- Low average reliability at the final decisions of the optimum MAP decoder suggests with high probability that the correlation of the received vector with one or more pseudo-codewords is comparable to the correlation of the former with the actual codewords. Consequently, the contribution of pseudo-codewords to the final decisions of the iterative decoder is significant.
- The above assumption was verified by experiments carried out upon turbo and LDPC schemes. It was shown that the great majority of the iterative decoder's non-convergent blocks, are blocks that would be decoded with low average reliability by the optimum MAP.
- The positions that exhibit the lowest reliability among the optimum MAP decisions can be considered as the weakest contributors to the correlation of the codewords with the received vector. The same positions involve the highest potential of being the strongest contributors to the correlation of the received vector with pseudo-codewords, if iterative decoding is applied.
- A simple way to eliminate the unwanted contribution coming from those bit positions is to modify the associated coordinates of the received vector by setting them equal to the corresponding transmitted symbols. Simulation results showed that modification of a single coordinate is sufficient to improve dramatically the iterative decoder performance and bring it very close to the optimum.
- To investigate the same phenomenon on longer codes and generalise the previous findings, it was necessary to substitute the direct implementation of optimum MAP decoding with an approximate method of manageable complexity. The OSD decoder was utilised to create a list of $\mathcal{L}$ codewords, significantly shorter than the 2^k size of the complete codeword list. The optimum MAP decisions of the full code were approximated by MAP decoding on the smaller vector space of size $\mathcal{L}$.

For higher list size $\mathcal{L}$, the approximation error reduces and the actual least reliable position of the optimum MAP can be identified with higher accuracy. As the approximation error reduces, modification of the estimated least reliable position offers gradual improvement in terms of BER and FER. The loss with respect to optimum decoding is minimised or even eliminated in many cases. Therefore, with the aid of a list decoder it was shown that:

- The least reliable bit decision of the optimum MAP decoder is indeed the strongest contributor to the correlation of pseudo-codewords with the received vector.
- A very large percentage of the prior non-convergent blocks of an iterative decoder can be decoded optimally if the received value of the single strongest contributor is optimised (modified to the transmitted value).
- The above method is equally effective on long codes too.

The Received Vector Coordinate Modification (RVCM) algorithm

- Takes full advantage of the potential gain that is offered when a single position of the received vector is modified. Applicable on turbo codes and LDPCs. Improvements that exceed one order of magnitude (and in some cases even more than two orders of magnitude) have been observed in terms of BER and FER.
- In contrast to other schemes, the added computational load increases linearly with block-length as only a single bit is modified at any time.
- Performance identical to ML has been observed for certain short LDPC codes.
- At higher SNR it is capable of delivering optimum performance for both LDPC and turbo codes.
- Selection criteria have been developed so that a critical coordinate is identified with high accuracy.
- Comparison of RVCM with alternative convergence improving schemes for LDPC codes has revealed that RVCM is generally more efficient and more flexible.

The structured permutation algorithm

- It was shown that by permuting certain positions of the received vector of a non-convergent block and by then restarting the decoding algorithm it is possible to

correct many of the prior non-convergent blocks. The improvement gets much closer to the ML performance at high SNR. At moderate and higher SNR its performance becomes comparable to that of RVCM.

Some ideas and questions for future research directions that may shed more light to the behaviour of iterative decoding and the convergence problem are:

- Further investigation for a selection method that will identify critical bits with 100% accuracy.
- What is the reason that a makes a non-convergent block not correctable by RVCM? Is it possible to identify those blocks?
- The performance of some LDPC codes (such as the 105,53, the 63,37 and the 93,47) becomes identical to the ML when RVCM is applied (at any E_b/N_o). For other codes of similar length (e.g. the 85,37) the ML performance is only reached at very high SNR. Investigation of the weight spectrum of these codes might give the answers why this happens. The outcomes of this investigation might reveal some interesting conclusions in code design and pseudo-codewords generation.
- Performance of RVCM on irregular LDPC codes.
- Performance of RVCM on high rate LDPC and turbo codes
- Extension of the convergence research and the performance of RVCM on different channels (e.g. magnetic)
- Investigation of the concept of pseudo-distance and how can this be used for further understanding of the non-convergence issue of iterative decoders.
- Application of RVCM to watermarking
- It has been investigated that for some LDPC codes the method that is used by the SP algorithm for updating the extrinsic probabilities (row or column updating method) affects significantly the performance. The variation in the performance of the two updating methods might be explained by analysing the distortion of the vector space (as in chapter 6) and the generation of pseudo-codewords. These results might be useful in identifying those code characteristics that affect most the performance of the two methods, so that the best choice is made for any given code.

- Further investigation of the extrinsic probabilities bounding method. How can we optimise the choice of the bounding parameter G? Why it does not work for LDPC codes?

- Further investigation of the structure permutation algorithm. Various questions are still open, such as:

  - How can we improve or even optimise the criterion of selection for the bits that participate in the permutation?
  - Is the optimum $SPAN$ constant for any code of the same block-length?
  - Can the same method be applied on LDPC codes?

- Investigation of the Linear Programming decoding methods. Analysis of pseudo-codewords based on Linear Programming.

- How the concepts of minimal pseudo-codewords and pseudo-distance can be incorporated into the findings of this thesis?

  - New code-design apporaches?
  - Understanding of the cost function of iterative decoding?
  - Identifying the conditions for minimum participation of pseudo-codewords in the decisions of iterative decoder?
  - Development of optimum RVCM selection criterion?

Papers Published

European Space Agency (ESA) 8th International workshop on Signal Processing for Space Communications (SPSC 2003) Conference Paper

Analysis of non convergent blocks at low and moderate SNR in SCCC turbo schemes

ANALYSIS OF NON CONVERGENT BLOCKS AT LOW AND MODERATE SNR IN SCCC TURBO SCHEMES

Evangelos Papagiannis, Marcel Ambroze, Martin Tomlinson

University of Plymouth
Drake Circus, Plymouth, PL4 8AA (UK)
E-mail: (epapagiannis,mambroze,mtomlinson)@plymouth.ac.uk

1 ABSTRACT

The paper presents a method to improve significantly the convergence of iteratively decoded concatenated schemes, such as serial concatenated convolutional codes, turbo codes and product codes. It is shown that many of the error blocks produced by the iterative decoder can be corrected by modifying a single, critical coordinate (channel value) of the received vector and repeating the decoding. Various statistics are presented about the critical coordinates, the evolution of the extrinsic vector and the effect of the memory of the component codes. The results are explained form an N-dimensional space point of view as a fixed-point problem, by analysing the contraction regions of the decoder and the evolution of the extrinsic information vector (EXIT charts). Results are shown mainly for serial concatenated codes, althought similar results have been obtained for parallel concatenated turbo codes and product codes.

2 INTRODUCTION

Iteratively decoded concatenated schemes, such as turbo codes [4], serial concatenated convolutional codes (SCCC) [3], have been shown to approach Shannon limit. They have low decoding complexity due to the suboptimal iterative decoder. Their performance curve has two regions: the waterfall region and the error floor region. The waterfall region is attributted to the lack of convergence of the iterative decoder, and its starting point can be determined by using density evolution [5]. In this paper, we show that the waterfall region can be improved significantly by using a simple algorithm which modifies one of the coordinates (channel value) of the received vector. We detect error blocks produced by the iterative decoder, modify one of the channel values at the input of the decoder (we call this value a critical coordinate of the received vector) and then repeat the decoding. We show that in this way an improvement of about 0.2dB can be achieved in the waterfall region of the iterative decoder. This algorithm is exemplified by using serial concatenation of convolutional codes of low memory, but it is also applicable to other concatenated schemes.
The iterative decoding procedure is the actual practical method of iteratively solving a system of N simultaneous equations with N unknowns. Analytical solution of such a system of a practical size would be extremely complex. A more efficient way of representing the turbo equations is in terms of fixed points. Mathematically the iterative decoder can be described as a problem of iteratively solving the equations [1]:

$$\begin{aligned} P_E^1 &= f(P_E^2) \\ P_E^2 &= g(P_E^1) \end{aligned} \tag{1}$$

Where f is the first MAP decoder's function, g represents the interleaving/de-interleaving process and the second MAP function while P_E are the corresponding extrinsic information vectors. When the iterative decoder converges to a fixed point solution the extrinsic information produced by the two decoders is identical and stable.

$$P_E^1 = f(g(P_E^1)) = h(P_E^1) \tag{2}$$

Here h is the turbo decoding function. Convergence to a fixed point doesn't imply that the decoder has necessarily converged to the zero error solution. It only means that the decoder has decided on a single solution and as a consequence of that no further changes and no decoding gain can be made with increasing iterations.

2.1 Convergence and Contraction Regions

Considering the N-dimensional space, fixed points are represented as single points in it. The extrinsic vector is initially undecided (all of its N components are equal to 0.5). After each iteration the extrinsic vector evolves within the space, with a certain magnitude and direction. If we denote with h the turbo decoding function then the system of N simultaneous equations can be expressed as:

$$\begin{aligned} e_{n+1}^{(0)} &= h_{n+1}^{(0)}(e_n^{(0)}) \\ e_{n+1}^{(1)} &= h_{n+1}^{(1)}(e_n^{(1)}) \\ &\vdots \\ e_{n+1}^{(N)} &= h_{n+1}^{(N)}(e_n^{(N)}) \end{aligned} \tag{3}$$

When all the components of the vector stabilize with iterations, h has converged to a fixed point solution. For that to happen h should be a contractor [7, 6], so that the change in the extrinsic vector with respect to that of the previous iteration is getting smaller and smaller.

$$||h(e_{(n+1)})|| \leq k||e_n||\,,\ 0 \leq k < 1 \tag{4}$$

The magnitude of the differential extrinsic vector will keep reducing and will finally end up to the fixed point if h is a contractor for that vector. Function h will be a contractor for the vector if the latter lies within one of the contraction regions of the N-dimensional space. So based on this geometrical interpretation we can categorize the behaviour of the extrinsic vectors in the N-dimensional space into three types:

- Non converging vectors that even after a large number of iterations are still wandering around without being trapped by any of the contraction regions.
- Converging vectors the initial point of which is into a contraction region, so they converge to the corresponding fixed point solution.
- Converging vectors which initially wander around until a certain iteration where they enter a contraction region and converge.

As it was previously stated even if the extrinsic vector converges to a stable solution this doesn't mean that it is the correct. So the problem of improving the performance of the iterative decoder can be defined as a problem of guiding the extrinsic vector into the zero error contraction region. Alternatively, it is the problem of reshaping the contraction region in such a way that the extrinsic vector starts, or at a certain stage ends up, within the region. These two approaches represent the main ideas of this paper.

3 RECEIVED VECTOR COORDINATE MODIFICATION (RVCM) ALGORITHM FOR CONCATENATED CODES

3.1 Description and Performance

Given a non convergent block (at least 100 iterations ran without convergence) the decoding algorithm is restarted but with one of its systematic bits' channel probabilities modified in such a way that the residual error of the specific component becomes zero. For that to happen, the bit should take either an initial probability of 1 or 0. Both of these should be tested starting from the one that is further from the observed channel probability. If no convergence to a suitable stopping criterion is achieved, the modified bit returns to its real channel value and the next bit follows the same procedure. As a result of this algorithm a remarkable improvement is achieved. Depending on the Eb/No level and the actual constituent codes, more than half of the prior non convergent blocks are successfully decoded even at as low Eb/No as the threshold value right at the beginning of the waterfall region of the error performance. Some typical success rates for the tail-biting terminated SCCC with Recursive Systematic Convolutional constituent codes of forward/feedback polynomials 5/7 (RSC(1, 5/7)) and random interleaving are presented in table 1. Figure 1 shows the performance improvement achieved by the RVCM algorithm applied to two SCCC schemes. One of them consists of two RSC(1, 5/7) codes and the other one of two RSC(1, 7/5) codes.

Note the considerable amount of gain that the very simple RVCM algorithm produces at such low SNR. Both of the tested codes are of memory-2 which means that their convergence properties at the low Eb/No region are very good and even a tiny improvement of fractions of dB at that steep region is difficult.
At the same figure we plot the Shannon's sphere packing bound [8] as this is drawn for quarter rate codes of block length N=2000. Shannon's sphere packing curve sets the FER performance upper bound for finite block length and code rate codes. Both of the tested schemes are less than 1dB away from that after applying the RVCM algorithm. The better converging RSC(1, 7/5) code is just 0.75dB away of it at Eb/No of 0.4dB and 0.8dB away at Eb/No of 1dB. At the lower to moderate Eb/No region the RSC(1, 5/7) SCCC FER improves by 0.16-0.175dB and reduces its distance from the bound to 0.9dB.

Eb/No dB	% success
0.4	51%
0.5	57%
0.6	63%
0.7	65%
0.8	73%
0.9	77%
1.0	81%

Table 1: Percentage of prior non convergent blocks that are successfully decoded by optimizing the initial value of a single bit (tail-biting RSC(1, 5/7) SCCC, block length N=2000, random interleaving).

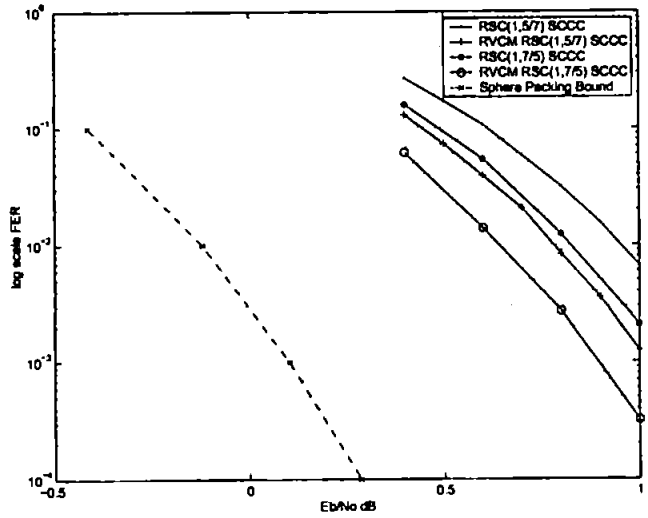


Fig. 1: FER improvement of RSC(1,5/7) and RSC(1,7/5) SCCCs by applying the RVCM algorithm, and comparison with sphere packing bound.

3.2 Discussion of the Algorithm and Results

It is rather surprising the effect that such a small change of only one bit has in the decoding performance. Recalling the introduction's discussion we can form an idea about the reasons that cause this performance improvement. We can think of the shape of the contraction regions, particularly of the optimum solution contraction region, as being a function of the code structure and the Eb/No. The reliability of the received channel observations is strongly dependant upon the Eb/No level. So the shape of the contraction region of the zero-error fixed point solution can be thought as a direct function of the code structure and the received channel observations, the initial values of the algorithm. By optimising (minimising the residual error) one of the components of the initial values we reshape all contraction regions in the N-dimensional space but more

importantly, we reshape the zero-error contraction region in favour of a successful decoding result. But how can such a small change in the initial values of the algorithm cause such a big improvement? Unlike an ordinary system of simultaneous equations, in turbo decoding a small change in the initial values produces a chain of reactions due to the interrelation of each bit with the whole sequence. Figure 2 shows the variation of the normalised norm of the differential extrinsic vectors for a non convergent block and for the same block after optimising the initial values of two individual bits (only one bit optimised at each time). Depending on the actual bit choice, the differential extrinsic vector becomes zero (the extrinsic vector stabilises) after a certain number of iterations and the block converges. The fact that the differential extrinsic vector is zero does not guarantee a zero-error solution but it denotes that the algorithm has provided a set of solutions for the equations' system of (3). In this particular example there are more than 100 bits that correctly converge the block to zero errors, among which bits No.39 and No.939. It is difficult to say whether the dominant effect that forces the block to converge is the reshaping of the contraction region or the change of the extrinsic vector's path in the N-dimensional space. Looking at the effect of bit 939 in the pattern of the differential extrinsic vector it seems that the contraction region of the optimal solution was reshaped in such a way that it included the initial extrinsic vector ([0.5,0.5,...,0.5]) at the beginning of the algorithm. As a result of that, the block converged within a few iterations. However, the large peak at the 10th iteration anticipates the previous scenario and can be interpreted as a critical change in the path of the vector that directs it into the zero-error reshaped contraction region and to convergence. The pattern caused by the optimisation of the initial value of bit 39 can be explained as an initial random walk of the extrinsic vector that is similar to that of the original non convergent block up to around iteration 44. At that point we have a critical peak change in the extrinsic vector that directs it to a solution for the system of (3). Summarising we can point out a few observations regarding the RVCM algorithm's behaviour:

- The RVCM algorithm alters the evolution of the extrinsic vector through out the number of iterations within the N-dimensional space
- In most of the cases the algorithm provides solutions for the set of equations (3) of the iterative decoder. Given that the initial change is towards the right direction (the correct value of the individual component) it is likely that some of the changes will force the block to the optimum solution
- In terms of the differential extrinsic vectors' evolution plots we can notice a common peak change in the extrinsic vector right before the vector enters into the contraction region and converges
- For the blocks that finally converge, it is not yet clear if the oscillatory parts of their differential extrinsic vectors patterns are associated with random walks of the extrinsic vector within or outside the contraction region

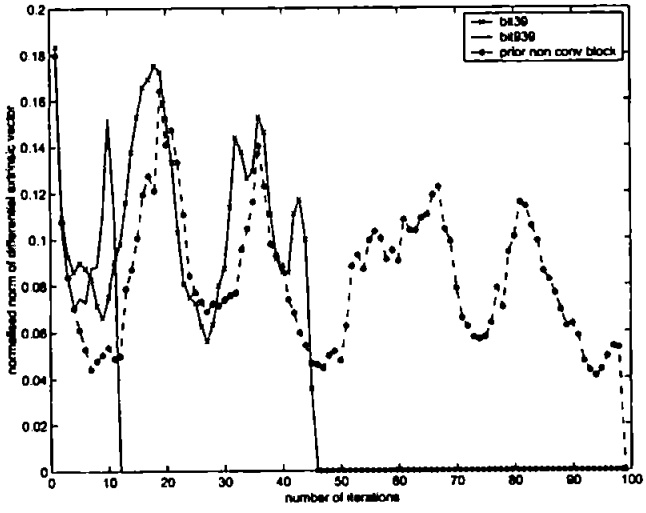


Fig. 2: Evolution of differential vectors

3.3 Extrinsic Information Transfer (EXIT) charts

A way of visualising the effect that a successful single bit optimisation has in terms of the mutual information exchanged between the decoders, is the EXIT chart [9]. The fact that the extrinsic output of each decoder becomes the *a priori* input for the next, allows us to plot the trajectory of the exchanged information in a single plot for successive iterations. The main point of such a chart is to iteratively assess the information gain achieved by each of the constituent decoders. In other words, the contribution of each decoder to the whole decoding procedure. The proper means for this is the mutual information at the output of the decoders, averaged over the whole sequence. Mutual information can be thought as a measure of reliability or information gain, in this case the gain between the input and the output extrinsic sequence of each MAP decoder. If at iteration n the averaged reliability of the input extrinsic vector is I_{E2} then, at iteration $n+1$, I_{E2} should be greater for the decoding procedure to improve towards the correct solution. Since I_{E2} is the input to constituent decoder-1, the previous condition will only apply if decoder-1 has increased its mutual extrinsic information I_{E1} at iteration $n+1$. In EXIT charts, the gain is proportional to the step width towards higher mutual information values. Ideally, after a certain number of iterations the mutual information should reach 1 with both of the decoders saturated and stable. Then, the block has converged.
Figures 3a and 3b show the EXIT charts of a particular block before and after optimising a single coordinate at the starting values of the iterative algorithm. In figure 3a, the iterative improvement stops at the third iteration. The extrinsic vector wanders around in the N-dimensional space and the mutual information cannot move higher than around 0.35. Small improvements and degradations occur through out the procedure and the block does not manage to escape from this loop. In figure 3b, we have the graph for the same block after optimising the initial value of one of the bits that force the decoder to convergence (critical bits) and restarting the iterative decoding procedure. Clearly the decoder finds the right path, escapes the loop after a short wandering of about 10 iterations and converges to the correct solution.

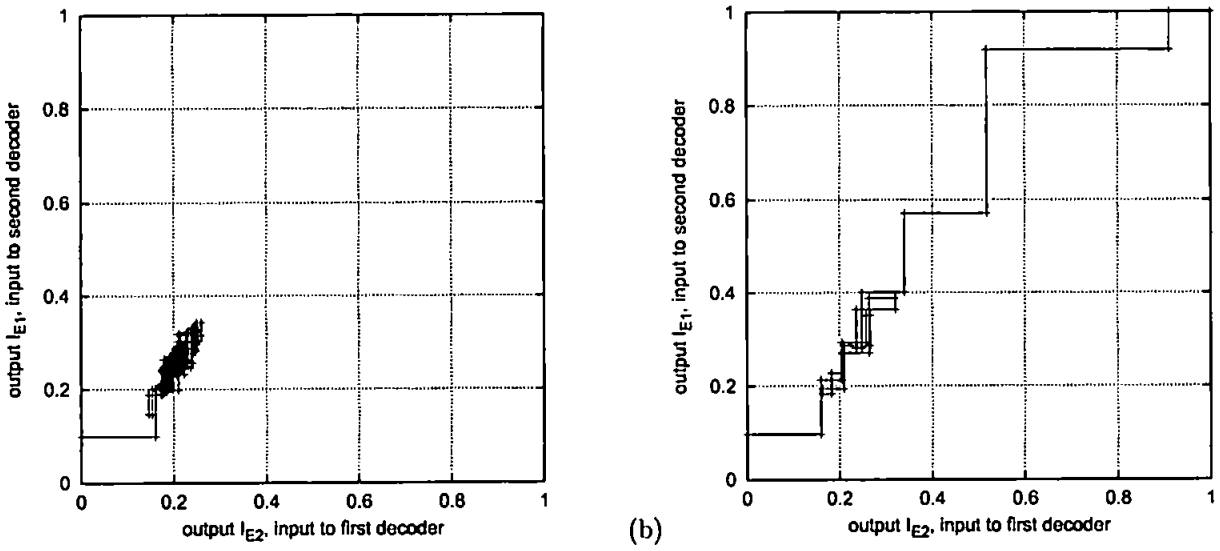


Fig. 3: (a)EXIT chart of non convergent block (b)EXIT chart for the same block after optimising the starting point of a single bit

3.4 Critical Bits

An immediate question that arises from all previously discussed findings is about the criteria that should be used for selecting the individual bits that force the block to convergence (critical bits). From the current simulations it seems that there is a variety of individual bits that can correct a given block if the RVCM algorithm is applied. Blocks with as many as 400 critical bits have been found, each of them determining the convergence of the block. On the other extreme, blocks with only 1 critical bit have been also observed for the same Eb/No level. Graphs of the successful bits distribution for a given interleaver and Eb/No reveal no strong indication

of bit positions within the sequence that are more like to converge a block if their initial value is optimised. Figures 4a and 4b, show the plotted distribution for the same random interleaver at 0.6 and 0.8dB. In addition to the fact that the plots seem to reveal a noise-like random distribution, there is no correlation between them. Figures 5a and 5b reveal a characteristic of the RVCM algorithm that makes its practical implementation easier even without knowledge of the critical bit positions. It would be desirable if one of the bits that successfully converge the decoder is towards the beginning of the block so that the search time is minimised. As shown in figures 5a and 5b, the first critical bit is usually detected quite early in the block. That is not surprising since the distribution of critical bits within the block is uniform if averaged over a large number of trials. From the cdf curve of figure 6 we can extract some useful results. Considering the 1dB curve, almost 86% of the total critical positions are within the first 100 bits of the block and 76% within the first 50 bits. Even better, half of the total critical positions are within the first 10 bits. For the 0.7dB case there is a slight degradation of no more than 7%. So the span of the search for a critical bit that might correct a non convergent block can be significantly reduced depending on the performance loss that we are willing to tolerate. Even for very tight specifications, the search domain can be reduced by 75% with only 5% loss from the maximum achievable performance. The situation improves at higher Eb/No levels.

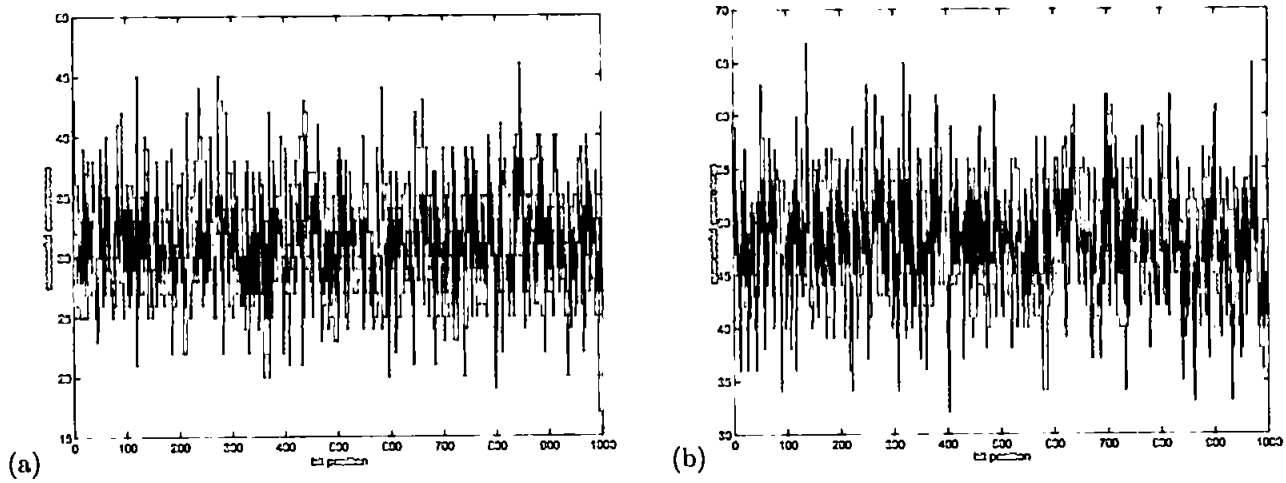


Fig. 4: Distribution of successfull bit positions for the same random interleaver at (a)0.6dB and (b)0.8dB

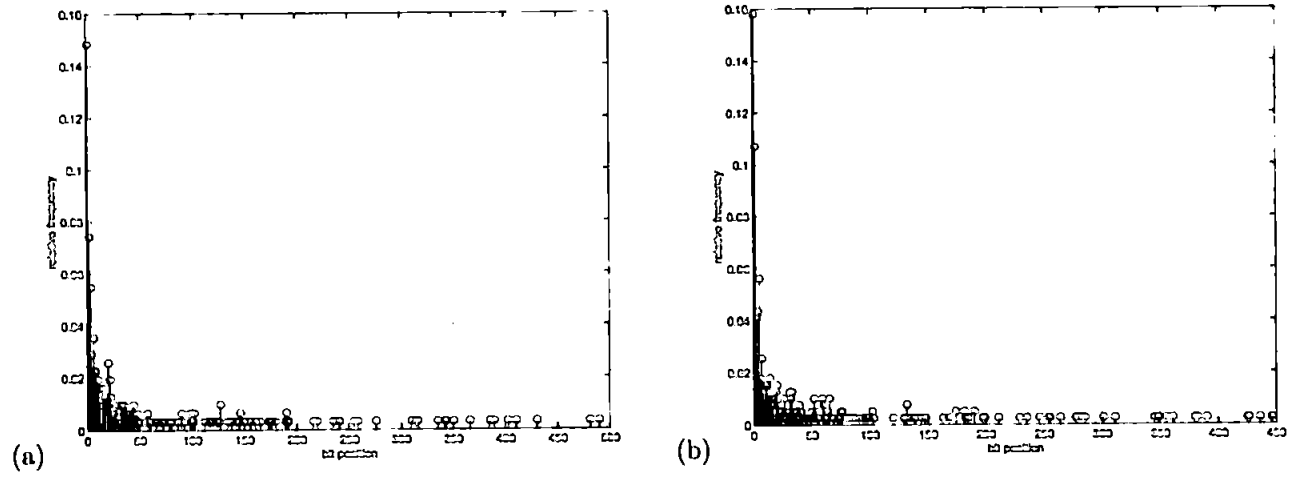


Fig. 5: Distribution of the bits that first converge a block at (a)0.7dB and (b)1dB for the same code and interleaver

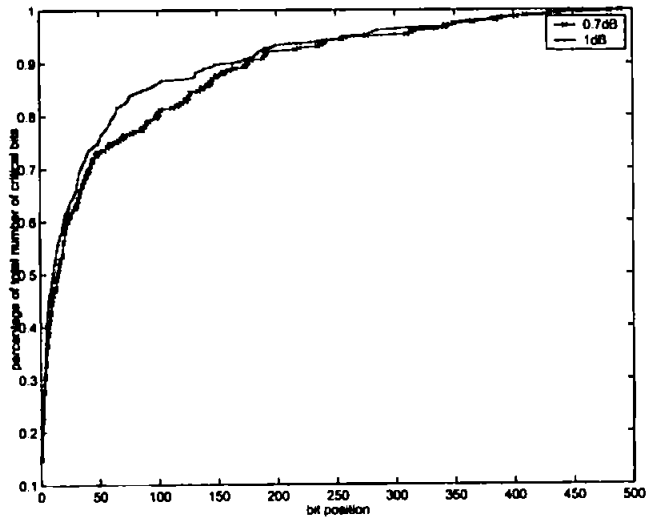


Fig. 6: cdf curves of histograms 5a and 5b

3.5 The RVCM Algorithm for Higher Memory Codes

Compared with low-memory, SCCC schemes with constituent codes of higher memory perform worse at the lower Eb/No region but exhibit a lower error floor [2]. Figure 7 shows plots of the FER performance curves of an SCCC scheme with the RSC(1,17/13) memory-3 constituent codes. The coding gain achieved by using the RVCM algorithm is very similar to that of the previously discussed memory-2 codes, about 0.17 to 0.18dB. However, the number of the non convergent blocks that can be successfully corrected by using the RVCM is lower than that of the memory-2 codes' case. At 0.8dB for an SCCC with constituent codes RSC(1, 5/7), 73% of the prior non convergent blocks will be corrected. If we change the constituent codes with the RSC(1, 7/5) the success rate increases to 78% while for the memory-3 RSC(1, 17/13) codes it is just 61%. The convergence of the memory-3 code is slower and the error performance curve less steep, so even though the success rate has decreased the coding gain of RVCM remains very similar with that of the memory-2 codes.

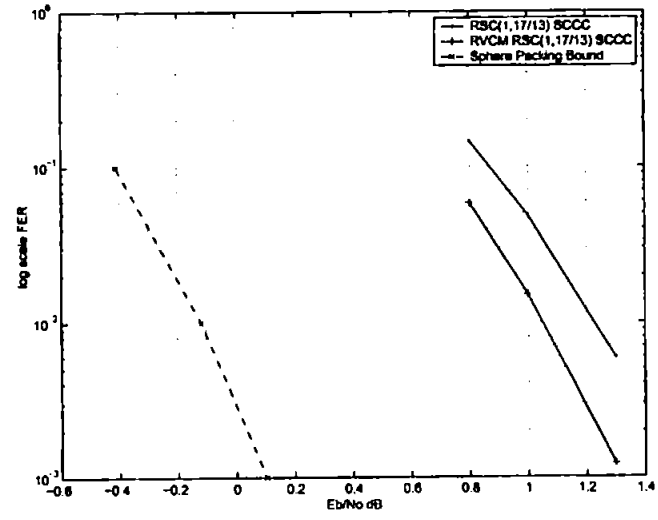


Fig. 7: FER improvement of RSC(1,17/13) SCCC by applying the RVCM algorithm, and comparison with sphere packing bound

4 CONCLUSIONS

A large number of non-convergent error blocks can be corrected by modifying a single, critical coordinate of the received vector at the input of the iterative decoder. For each error block, there are generally more than one critical coordinates. Their number and position in the received vector is different for each error block. Investigation shows that there are no preferred critical values that work for all blocks. However, most of the blocks have at least one critical value in the first few coordinates of the received vector. Based on this observation, we have proposed an algorithm which improves the performance of the iterative decoder by about 0.2dB. Different component codes show a similar improvement. The improvement due to the algorithm has also been verified for parallel concatenated turbo codes and product codes. Future work will be directed towards identifying the critical coordinates by examining the received vector and the evolution of the iterative decoder.

References

[1] A. Ambroze, G. Wade, and M. Tomlinson. Practical aspects of iterative decoding. *IEE Proceedings Communications*, 147(2):69–74, April 2000.

[2] M.A. Ambroze. *On turbo codes and other concatenated schemes in communication systems.* PhD thesis, University of Plymouth, Plymouth, UK, November 2000.

[3] S. Benedetto, M. Montorsi, D. Divsalar, and F. Pollara. Serial concatenation of interleaved codes: Performance analysis, design and iterative decoding. *JPL TDA Progress Report*, pages 42–126, August 1996.

[4] C. Berrou, P. Thitimajshima, and A. Glavieux. Near Shannon limit error correcting coding and decoding: turbo codes. In *Proc. IEEE International Conference on Communications*, pages 1064–1070, Geneva, Switzerland, May 1993.

[5] D. Divsalar, S. Dolinar, and F. Pollara. Iterative turbo decoder analysis based on density evolution. *tdapr*, (42-124):1–33, feb 2001.

[6] W.Lee Johnson and R.Dean Riess. *Numerical Analysis.* ADDISION-WESLEY PUBLISHING COMPANY, 1977.

[7] W. Sawyer. *Numerical functional analysis.* Oxford University Press, 1978.

[8] C.E. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois Press, 1949.

[9] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

IEEE International Zurich Seminar 2004 Conference Paper

Approaching the ML performance with iterative decoding

Approaching the ML performance with iterative decoding

Evangelos Papagiannis, Marcel Adrian Ambroze and Martin Tomlinson
University of Plymouth
Drake Circus
PL4 8AA Plymouth, UK
Email: (epapagiannis,mambroze,mtomlinson)@plymouth.ac.uk

Abstract—**The paper presents a method to significantly improve the convergence of iteratively decoded concatenated schemes and reduce the gap between iterative and maximum likelihood (ML) decoding. It is shown that many of the error blocks produced by the iterative decoder can be corrected by modifying a single critical coordinate (channel value) of the received vector and repeating the decoding. This is the basis of the RVCM (Recieved Vector Coordinate Modification) algorithm. Its description, performance and drawbacks are discussed later on. The paper also presents a practically obtained lower bound on ML performance based on the Euclidean distances of the transmitted and the iteratively decoded codewords from the received vector. At low SNR this bound is assuming an unrealistic perfect code, while at high SNR the aproximations are getting closer to the real characteristics of the code and the RVCM iterative decoder is shown to achieve the ultimate ML performance.**

I. Introduction

For many years the main difficulty in using powerful codes that approach the capacity limits was the complexity involved in the decoding side. Turbo decoders [2] approximate the code performance with only a fraction of the effort required by an ML decoder. The worse performance of the turbo towards ML decoding, is attributed to the lack of convergence of the iterative algorithm to the most likely codeword decision.

We can establish a lower bound for the ML performance and compare the turbo decoder with that, by using the MRL (more likely) codeword criterion [4]. Although we cannot define how many times the ML decoder will choose the right codeword (the ML codeword is not always correct), we can actually identify all those cases that the ML decoder would definitely make an error.

Over the years many methods have been proposed for improving the performance of iterative decoding. In the context of LDPCs the bit-flipping algorithm [6] targets the most unreliable bit of each partial parity equation. Chase [3] type decoding methods have also been proposed for algebraic decoding of block codes [5]. This paper attempts to improve the convergence performance of iterative turbo decoders by applying the RVCM (Received Vector Coordinate Modication) algorithm [8] [7]. If RVCM is capable of correcting all the non-converged (at least 100 iterations ran without convergence) non-MRL blocks, the iterative decoder should be identical to the code's optimum maximum likelihood decoding scheme.

The difficulty associated with the RVCM algorithm is the increased computational effort required. For maximum gain the complexity increases significantly. For delay critical applications a compromise can be made by choosing a small group of coordinates that meet a certain set of criteria like the reliability of their received channel values.

II. RVCM Algorithm for Concatenated Codes

A. Description and Performance

In order to apply the RVCM algorithm to a particular block we need to know the corresponding received vector. So each received vector is buffered for as long as the block is processed by the decoder. Given a non-convergent block the decoding algorithm is restarted, but with one of its systematic bits' received values modified to one of the possible transmitted values. In this paper we assume BPSK modulation, so the chosen coordinate (systematic bit's position) should be made either +1 or -1. Obviously, one of the two choices will make the residual error equal to zero and the other will maximise it. Since we can't know which is the appropriate value we have to try both. If no convergence to a suitable stopping criterion is achieved with any of the two, the modified bit is returned to its real channel value and the next systematic bit follows the same procedure. The algorithm offers gains of 0.15 to 0.2dB (the application is restricted to just non-MRL blocks for a fair comparison). The percentage of the prior non-convergent blocks that are corrected depends on the Eb/No level and the constituent codes themselves. The memory of a code is a major factor in the success rate of the RVCM algorithm. Nevertheless, the coding gain remains at the same levels eventhough the effectiveness of the RVCM algorithm reduces, because of the inherent inferior convergent characteristic of the higher memory codes [1] (on a less steep FER curve the same success rate of RVCM would be translated into higher coding gain). Table I shows the algorithm's success rates (percentage of prior non-convergent blocks corrected) based on a serial concatenated convolutional code (SCCC) scheme of memory-2 and memory-3 recursive systematic convolutional (RSC) constituent codes.

B. Discussion and Results

The modification of the received vector reshapes the contraction region (the N-dimensional volume into which the extrinsic vector's change over iterations shrinks and stabilises

Eb/No dB	RSC(1,5/7)	RSC(1,7/5)	RSC(1,17/13)
0.6	63%	74.4%	–
0.8	73%	78%	61%
1.0	81%	84.6%	68.8%
1.3	–	–	79.4%

TABLE I

PERCENTAGE OF PRIOR NON CONVERGENT BLOCKS THAT ARE SUCCESSFULLY DECODED BY THE RVCM ALGORITHM (TAIL-BITE, R=1/4, BLOCK LENGTH N=2000, RANDOM INTERLEAVING).

to the corresponding fixed point solution) and alters the path followed by the extrinsic vector throughout the decoding process. As long as the modification is towards the correct solution, all changes act in favour of successful decoding and most of the times the extrinsic vector stabilises to the zero-error solution.

The EXIT charts [10] in figure 1 give a very good picture of the effect that the modification of a single coordinate produces. The fact that the extrinsic output of each decoder becomes the *a priori* for the next, allows us to plot the trajectory of the exchanged information in a single plot for successive iterations. The target is to assess the information gain achieved by the individual decoders at each iteration.

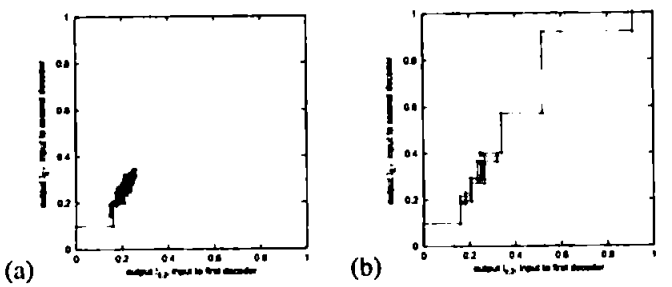


Fig. 1. (a)EXIT chart of non convergent block (b)EXIT chart for the same block after optimising the starting point of a single bit

The information gain (step-width in the graph) is expressed by the mutual information, in other words the difference between the *a posteriori* (decoder's output) and the *a priori* (decoder's input) probabilities. As we can point out from figure 1(a), initially the two constituent MAP decoders produce gain and the reliability increases continuously. But after the first 2 iterations the turbo decoder enters in a loop and the average mutual information never exceeds 0.35. This is a non-convergent block. In 1(b) we have the chart of the same block after applying the RVCM algorithm. The decoder wanders for around 10 iterations but it eventually manages to escape the loop and converge. Obviously, the small initial modification has a huge impact in the decoding outcome although any gain increase is hardly noticed in the first few iterations. The turbo decoder builds iteration by iteration on the tiny extra information provided by the RVCM algorithm, and most of the times it corrects the block. The interrelation among the bits (all bits take part more or less to each decision of the MAP decoder) and mainly the continous exchange of information between the decoders are the main reasons for the algorithm's behaviour.

Figure 2 shows the FER performance improvement achieved by the RVCM algorithm against the standard turbo code when applied in a SCCC scheme with the RSC(1,5/7) memory-2 constituent codes. The Shannon's sphere packing bound [9] (drawn for N=2000, r=1/4) is shown in the same graph for direct comparison.

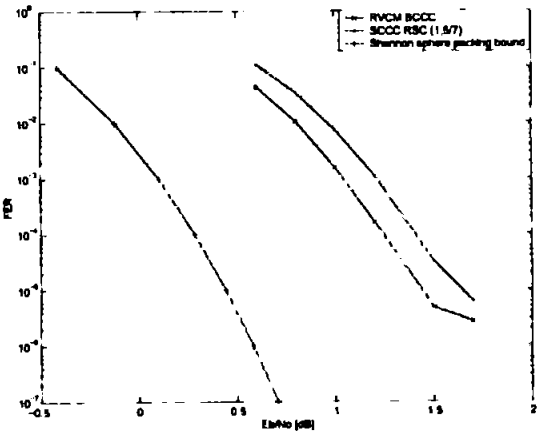


Fig. 2. RVCM improvement towards standard SCCC (block-length=2000, tail-bite, r=1/4, RSC(1,5/7), random interleaving)

Figure 3 presents the results of a parallel concatenated turbo code scheme with m-binary [4] component codes. It can be observed that the improvement provided by the RVCM algorithm is similar to that of the serial concatenated scheme. The figure also shows the sphere packing bound and the bound corresponding to binary modulation for rate equal to 2/3.

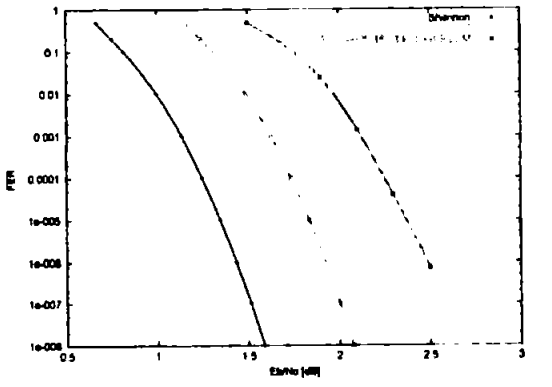


Fig. 3. RVCM algorithm for parallel concatenated turbo codes of code rate 2/3 and information block-length 1054 using m-binary component codes

C. Critical Bits

One of the major problems concerning the RVCM algorithm is the difficulty involved in finding the bit(s) that if modified can converge the decoder to the zero-error solution. We name these as critical bits. The number of critical bits

in a block varies from 400 to 1 out of 1000 systematic bits. The distribution of critical bits is uniform with no sign of vulnerable or favourite bit positions [8]. So for full exploitation of the capabilities of RVCM, all of the block's systematic bits should be tried.

Because of the uniform distribution of the critical bits along the sequence it is very likely that if we limit our search to a smaller group, at least one of them will be found (given of course that a critical bit exists in the block). By using this method we can significantly reduce the computational effort but for the price of suboptimum performance. Furthermore, by choosing the candidate critical bits based on the reliability of their channel value instead of randomly, the algorithm can achieve up to 10% better success rates for small groups of 10 or 20 bits size. However, as it can be seen from figure 4, the gain in dB is negligible. Table II summarises figure 4 and supplies some additional information.

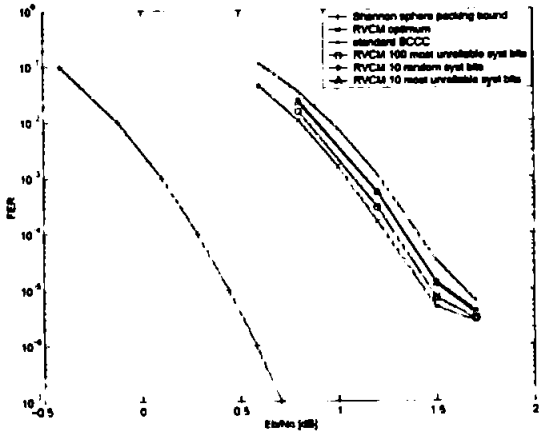


Fig. 4. Suboptimum RVCM schemes

Eb/No	10u	10r	Avg(u/r)
0.8	$292/798 = 36.6\%$	$241/798 = 30.2\%$	7/8
1.2	$212/394 = 53.8\%$	$203/394 = 51.5\%$	6/6
1.5	$185/266 = 69.5\%$	$175/266 = 65.8\%$	4/5
1.7	$71/101 = 70.3\%$	$60/101 = 59.4\%$	5/6

TABLE II

SUCCESS RATES AND AVERAGE NUMBER OF TRIALS FOR GROUPS OF 10 RANDOM (10R) AND OF THE 10 MOST UNRELIABLE (10U) BITS.

III. Aproaching the ML Performance of the Code

A. Optimum and Iterative decoding

The capabilities of a concatenated encoding scheme can be fully exploited if the sent data is recovered by an optimum combined ML decoder. In terms of FER the ML decoder gives the maximum performance we can get. The price paid for the optimality offered is a significant increase in the complexity since the number of states of the combined trellis will be equal to the product of the number of states of each constituent code. If an interleaver is placed between the two codes then complexity increases enormously. Even for short lengths it is almost impossible to implement an ML decoder because of the random-like interrelations, among the sequence bits, introduced by the interleaver.

Iterative decoding offers a compromise between performance and complexity. It approximates ML decoding by exchanging information throughout a series of iterations. Turbo codes in particular, exchange extrinsic information which represent the independent decisions of each individual decoder. The iterative decoder algorithm though, does not always converge to the ML solution. Sometimes it doesn't converge to any solution. So, a straight forward question is about the scale of the loss of the iterative approximation and possible methods of reducing this.

B. Explaination of the MRL Notion

The MRL notion will serve as an approximation to the ML criterion. Since we perform suboptimum decoding (turbo decoding) the decoder's outcome is not always the most likely choice (the most likely codeword). In fact, most of the times the output of the turbo decoder is not even a codeword (at low to moderate SNR). The reader should be aware that the information output is re-encoded to give a valid codeword, and that is considered as the output of the turbo decoder. It wouldn't make sense to use the raw output of the turbo decoder in the effort of approaching the ML performance.

By checking the Euclidean distances of the decoded and the transmitted codewords from the received vector, we can decide whether or not the iterative decoder's output is more likely than the transmitted (correct) codeword. If it is, we can be sure that the hypothetical exact ML decoder would definetely not pick the correct codeword since there is at least one erroneous codeword closer to the received vector. Recall that the best that any optimum probabilistic decoder can do is to find the valid codeword that is closer to the received vector (the actual ML criterion).

Figure 5 visualises the above case in the Euclidean space. Where T is the transmitted vector, D is the decoded and R the received vector. If vector D is closer than T to the received vector R in terms of Euclidean distance ($d_1 < d_2$), then D is considered an MRL decision and due to the previous paragraph's discussion we can be sure that the optimum ML decoder would be in error.

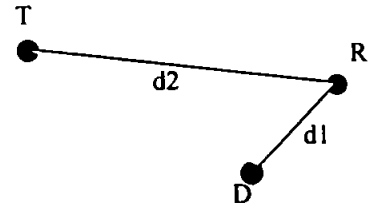


Fig. 5. Represantation of an MRL decoded block in terms of Euclidean distances

The results obtained by this method set the lower bound on the ML performance since only the minimum of the possible error events are taken into account. All the non-MRL blocks are assumed to be corrected by an ML decoder.

C. The RVCM-Turbo Decoder closer to the ML Performance

The RVCM algorithm offers significant improvement to the turbo decoder's performance. As table I has already revealed, a large number of non convergent blocks can be successfully corrected as a result of this.

If RVCM is capable of correcting all the non-MRL blocks that have not initially converged, then the iterative decoder performance will be identical to the optimum code performance (lower bound). The obtained bound is an optimistic estimation of the code performance, especially at lower SNR. But as SNR increases, the RVCM-turbo decoder should get very close and even be identical to the ML bound. Figure 6 shows the simulated results. Because of the weak, low memory consistuent codes (RSC(1,5/7)) the overall code has a high error floor. This is due to the increasing number of MRL blocks at high SNR. Eventhough at this Eb/No range the RVCM algorithm is capable of success rates of 100%, the increasing number of MRL blocks, as a result of the low d_{free} of the code, limits the performance.

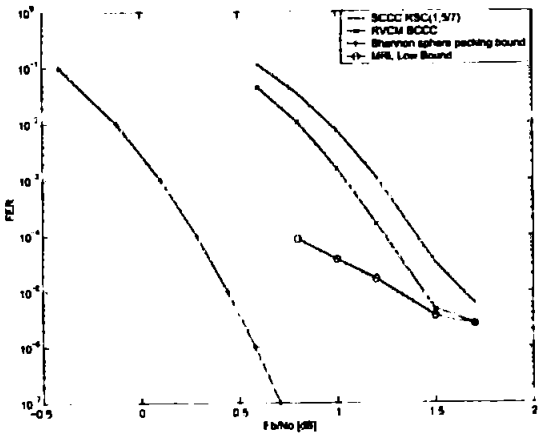


Fig. 6. Comparison with the MRL bound

IV. Conclusions

A large number of non-convergent error blocks can be corrected by modifying a single critical coordinate of the received vector at the input of the iterative decoder. Their number and positions vary among the error blocks. Generally the most unreliable coordinates have a higher probability of converging an error block, if modified.

The RVCM algorithm exploits the above characteristics and attains about 0.15 to 0.2dB of additional coding gain. The iterative decoder approaches the optimum ML performance of the code. The latter can be lower bounded with the aid of the MRL criterion. At high SNR the RVCM turbo decoder manages to correct all but the MRL blocks and, consequently, its performance becomes identical to the optimum ML decoder.

The capabilities of the RVCM algorithm are actually limited by the code's d_{free} which is responsible for the increasing number of MRL blocks at higher SNR. However, the d_{free} influences the convergence behaviour of the iterative decoder.

References

[1] M.A. Ambroze. *On turbo codes and other concatenated schemes in communication systems*. PhD thesis, University of Plymouth, Plymouth, UK, November 2000.

[2] C. Berrou, P. Thitimajshima, and A. Glavieux. Near Shannon limit error correcting coding and decoding: turbo codes. In *Proc. IEEE International Conference on Communications*, pages 1064–1070, Geneva, Switzerland, May 1993.

[3] D. Chase. A class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory*, pages 170–182, 1972.

[4] M. Ferrari, S. Bellini, M. Tomlinson, and M. Ambroze. Backbone interleaver design for multi-binary turbo codes. Brest, France, September 2003.

[5] M. Fossorier and S. Lin. Chase-type and gmd coset decodings. *IEEE Transactions on Communications*, 48(3):345–350, 2000.

[6] R.G. Gallager. *Information theory and reliable communication*. John Wiley and Sons, Inc., New York, 1968.

[7] E. Papagiannis, M.A. Ambroze, and M. Tomlinson. Uk patent 0320126.6.

[8] E. Papagiannis, M.A. Ambroze, and M. Tomlinson. Analysis of non convergent blocks at low and moderate snr in sccc turbo schemes. Catania, Italy, sep 2003. ESA.

[9] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, 1949.

[10] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

International Symposium on Communication Theory Applications (ISCTA 2005) Conference Paper

Improved iterative decoding for perpendicular magnetic recording

Mr C.Tjhai was the main contributor and presenter of this paper

Improved Iterative Decoding for Perpendicular Magnetic Recording

E. Papagiannis†, C. Tjhai†, M. Ahmed*, M. Ambroze†, M. Tomlinson†
† Fixed and Mobile Communications Research, University of Plymouth, Plymouth, PL4 8AA, UK
* Centre for Research in Information Storage Technology, University of Plymouth, PL4 8AA, UK
email: {epapagiannis,ctjhai,mahmed,mambroze,mtomlinson}@plymouth.ac.uk

Abstract

An algorithm of improving the performance of iterative decoding on perpendicular magnetic recording is presented. This algorithm follows on the authors' previous works on the parallel and serial concatenated turbo codes and low-density parity-check codes. The application of this algorithm with signal-to-noise ratio mismatch technique shows promising results in the presence of media noise. We also show that, compare to the standard iterative decoding algorithm, an improvement of within one order of magnitude can be achieved.

1. Introduction

Longitudinal recording has been the standard method in magnetic recording for decades. Recent research has shown that this method seems to reach its physical limits in the near future due to the superparamagnetic effects. On the other hand, the technique which has been known prior to the longitudinal recording–perpendicular recording, has recently been the centre of research attention. Perpendicular magnetic recording offers promising increased in recording densities, up to 1 Terabit per square inch seems feasible [1]. As the areal density is increased, however, the signal processing aspects of magnetic recording becomes more difficult. Sources of distortion including media noise, electronics and head noise, jitter noise, inter-track interference, thermal asperity, partial erasure and dropouts become more apparent and unless appropriate mitigation techniques are present, signals cannot be retrieved reliably from the recording media.

Since the discovery of turbo codes, soft-decision iterative decoding has been shown to be able to provide significant coding gain over the conventional detection method on magnetic recording. The utilisation of iterative decoding on the concatenation of partial-response (PR) channel and powerful error-correcting codes such as low-density parity-check (LDPC) and turbo codes has been proposed in many literatures. Iterative decoding is a reduced-complexity method to achieve the optimum solution–the maximum-likelihood solution and as such, iterative decoding is sub-optimal.

In this paper, we present a method to improve the sub-optimality of the iterative decoding and demonstrate its applications to perpendicular magnetic recording in the presence of media noise. The improved method, which is known as the Received-Vector-Coordinate-Modification (RVCM) algorithm, follows on the previous works of the authors [2], [3], [4], [5], [6]. This method is similar to the works of [7] and [8]. This paper also investigates the use of signal-to-noise ratio (SNR) mismatch [9] to mitigate the effect of media noise.

The rest of the paper is organised as follows. Section 2 describes the perpendicular recording channel used. The description of the RVCM algorithm is outlined in Section 3 and the performance of this algorithm is demonstrated in Section 4. Section 5 concludes this paper.

2. Channel Model

Figure 1 shows the block diagram of the perpendicular recording system model used in this paper. The user data, denoted as a_k, is a sequence of of input symbols taking values of $\{0, 1\}$. Some error-protection redundancy is added to the sequence a_k by the error-correcting-codes (ECC) encoder forming codeword sequence c_k. To simulate the write current, the sequence c_k is mapped to $\{-1, +1\}$ according to $2c_k - 1$ operation. The scaling factor of 0.5 is to ensure the transition takes values of $\{-1, 0, +1\}$.

We assume that the read head produces zero voltage in the region of magnetic transitions and some voltage in the region of constant magnetic polarity. We approximate the single-transition step response, denoted as $s(t)$, using the hyperbolic-

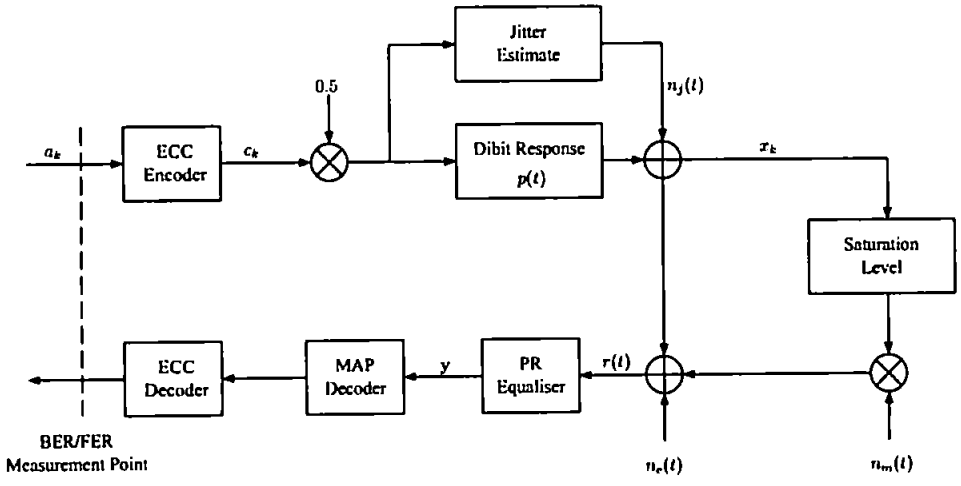


Figure 1. Perpendicular recording system model

tangent function [10]:

$$s(t) = A \cdot \tanh\left(\ln(3)\frac{t}{\mathrm{PW}_{50}}\right) \qquad (1)$$

where A is the saturation level or the amplitude from zero to peak (normalised to unity) and PW_{50} is the time taken for $s(t)$ to go from $-A/2$ to $+A/2$. It is assumed that t and PW_{50} are normalised to the symbol period, T. Throughout the paper, it is assumed that $\mathrm{PW}_{50} = 1.4$. We define the response of two adjacent transitions (dibit-response) $p(t)$ as:

$$p(t) = s(t) - s(t-1) \qquad (2)$$

and the readback signal $r(t)$ is simply the convolution of c_k and $p(t)$ plus some noise:

$$r(t) = \sum_k c_k p(t - kT) + n(t) \qquad (3)$$

where $n(t)$ is the overall noise in the recording system which consists of media, jitter and electronic noise, i.e. $n(t) = n_m(t) + n_j(t) + n_e(t)$.

The media noise, $n_m(t)$ originates from the imperfections of the media and its effect is significant in the magnetic transition regions. Typically, media noise is approximately four times the electronic noise at transition regions. In our system model, we consider the media noise as Additive-White-Gaussian-Noise (AWGN) with mean of 0 and variance of σ_m^2, which exists in the transition region only. As shown in Figure 1, the media saturation noise depends on the saturation level and it is evaluated as $1 - (x_k/A)^2$. Unlike media noise which is media dependent, the jitter noise $n_j(t)$ is due to timing imperfection only. To model the sampling jitter noise, $n_j(t)$, the nth order Taylor approximation is used. The jitter estimation block shown in Figure 1 is done with 6th order Taylor series expansion of $s(t)$. The jitter probability density function is assumed to be uniform, limited by a maximum value. The electronic noise, $n_e(t)$ is AWGN with mean of 0 and variance of σ_e. The recording system in Figure 1 caters for many different simulation cases with varying degree of electronics, media and sampling jitter noise. We define the channel SNR as:

$$\mathrm{SNR} = 10\log_{10}\left(\frac{1}{2\left(\sigma_e^2 + \sigma_m^2\right)}\right) \qquad (4)$$

The noisy readback signal is equalised to $(4, 6, 4, 2)$ PR target which is only optimal for electronic noise at the considered PW_{50} [11]. It serves for comparison purposes only. The Maximum-a-Posteriori (MAP) decoder of the PR channel exchanges extrinsic with the ECC decoder to deliver solution which is used for performance evaluation.

3. Received-Vector-Coordinate-Modification (RVCM) Algorithm

It has been shown that the RVCM algorithm provides considerable coding gain for parallel and serial concatenated turbo codes [3], [4] and LDPC codes [6]. The algorithm can be applied directly to perpendicular recording and is described briefly below.

3.1. Description of the Algorithm

Let $\mathbf{y} = \{y_0, y_1, \ldots, y_n\}$ denote an $n-$tuple vector at the output of the MAP decoder, that is the a-posteriori probability (APP) of the

MAP decoder. Let $\mathbf{l} = \{l_0, l_1, \ldots, l_n\}$ denote the reliability sequence of $\mathbf{y}$, where $l_i = \log(\Pr(y_i| + 1)/\Pr(y_i| - 1))$. Assume that $i_{\max}$ is an integer where $1 \leq i_{\max} \leq n$ and $\mathbf{p} \subseteq \{0, 1, \ldots, n-1\}$ is a vector of length $i_{\max}$.

Step 1. Store the vector $\mathbf{l}$, let the integer i be initialised to 0.

Step 2a. Set $l'_{\mathbf{p}_i} = l_{\mathbf{p}_i}$ and $l_{\mathbf{p}_i} = -\infty$. Restart the iterative decoder, store the decoded vector ($\mathbf{d}^-_{\mathbf{p}_i}$).

Step 2b. Set $l_{\mathbf{p}_i} = +\infty$. Restart the iterative decoder, store the decoded vector ($\mathbf{d}^+_{\mathbf{p}_i}$) and restore $l_{\mathbf{p}_i}$, i.e. $l_{\mathbf{p}_i} = l'_{\mathbf{p}_i}$.

Step 3. If $i < i_{\max}$ then set $i = i + 1$ and continue to **Step 2**. Otherwise, stop the algorithm and from the list of all decoded vectors $\mathbf{d}^-_{\mathbf{p}_i} \bigcup \mathbf{d}^+_{\mathbf{p}_i}, \forall i \in \{0, 1, \ldots, i_{\max} - 1\}$, choose a decoded vector that has the minimum euclidean distance. It is assumed that the iterative decoder always outputs a codeword.

From the steps above, it is clear that the complexity of the algorithm depends on $i_{\max}$.

3.2. Critical Symbols

One of the major obstacles concerning the RVCM algorithm is the difficulty in finding the symbol(s) that, if modified, can converge the iterative decoder to the maximum-likelihood solution [3], [4]. These symbols are referred as the critical symbols and their distribution is uniform with no sign of vulnerable or favourite symbol positions. On the other hand, due to their uniform distribution, it is likely that we can find one of the critical symbols if we confine our search to a small group, i.e. keeping the value of $i_{\max}$ low. In this way, we can reduce the computational complexity for the price of sub-optimum performance. As we will show later that, the gain obtained by confining $i_{\max}$ to a small value is still significant compared to the performance of the standard iterative decoder.

There are various methods for selecting the critical symbols, see [6] for details. In this paper, we restrict the selection to one method only, that is the reliability of the APP at the output of the MAP decoder.

4. RVCM Performance

We evaluate the performance of the RVCM algorithm on some short-block length turbo and LDPC codes. Figure 2 and 3 show the error rate performance of the turbo code under the standard iterative and RVCM decoders. The turbo code considered is the $b/b+1$ tail-biting turbo code[1], where $b = 3$, $k = 198$ and $n = 278$. The interleaver used is an $S-$interleaver. Significant improvement is noticed and the increase in performance gets better as SNR increases. It is worth noting that the results in Figure 3 were obtained using SNR mismatch technique. With this technique, the improvement in performance over standard iterative decoder is even greater as SNR increases. In the presence of media noise, SNR mismatch methods do not provide the same performance as observed with electronics noise only [9], however better targets for media noise are being investigated by the authors.

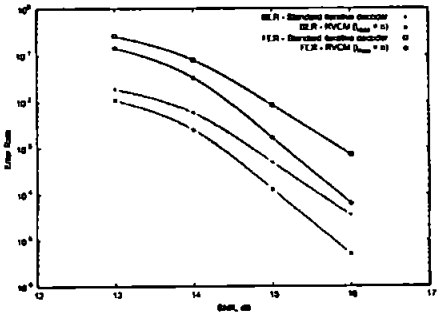


Figure 2. Error performance of RVCM decoder on the turbo code in the presence of electronic noise only

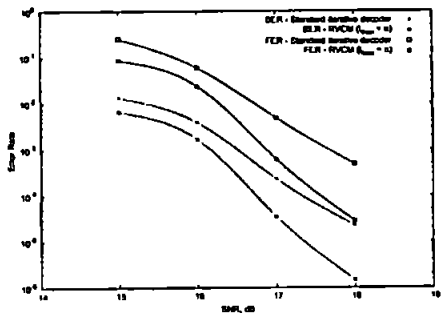


Figure 3. Error performance of RVCM decoder on the turbo code in the presence of electronic and media noise

Similar performance improvement is observed for the LDPC codes, see Figure 4. The [127, 84] cyclic LDPC code, which has minimum-distance of 10, was constructed using a method described in [12]. As mentioned earlier, the RVCM algorithm allows one to trade off the performance against the computational complexity. From Figure 4, despite the performance obtained by setting $i_{\max} = 10$ is approximately 0.3dB inferior to that by setting $i_{\max} = n$, the coding gain from the standard iterative decoding is significant. For the case of $i_{\max} = 10$, we select the critical symbols based on the reli-

[1] C. Berrou once referred this code as duo-binary turbo code.

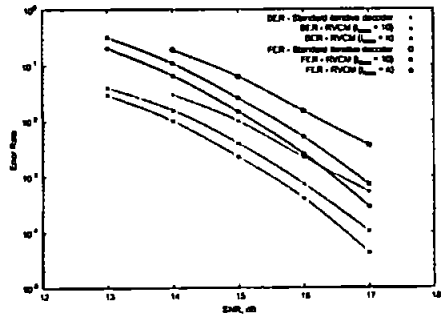


Figure 4. Error performance of RVCM decoder on the [127, 84] cyclic LDPC code in the presence of electronic and media noise

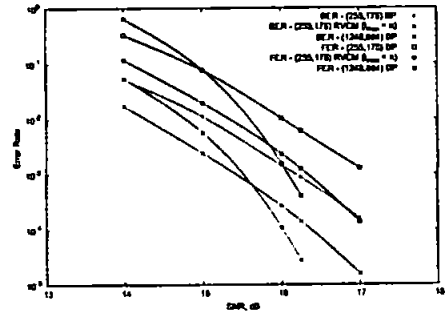


Figure 5. Performance of the [255, 175] cyclic and [1248, 864] codes in the presence of electronic and media noise

ability measure at the output of the MAP decoder. From the vector l, we construct a vector p of length $i_{\max}$ such that $l_{p_0} < l_{p_1} < l_{p_2} < \ldots < l_{p_{n-1}}$. In Figure 5, we compare the performance of the [255, 175] cyclic code and that of the [1284, 864] quasi-cyclic code. We can see that the RVCM algorithm provides significant gain, within one order of magnitude improvement, over the BP algorithm. At approximately 3×10^{-5} BER, the performance of the cyclic code with RVCM is within 0.6dB away from the longer code under BP decoding.

5. Conclusions

We have shown that the application of the RVCM algorithm to perpendicular magnetic recording shows promising results. Simulation results show that improvement of within one order of magnitude is possible. Short block length offers an attractive error-correction scheme in which RVCM algorithm can be fully exploited by setting $i_{\max} = n$. A bank consisting of $2i_{\max}$ parallel RVCM decoders can be built on chips and the decoding of short-block length data has low latency. The performance of longer block-length codes, up to a certain error-rate, can be outperformed by the application of RVCM algorithm to shorter codes. The exact point, at which the longer codes start to perform better, depends on the code structure.

We also extended our investigations on using some non binary cyclic LDPC codes [13] and we observe similar improvement as in the binary cases. Further investigations in identifying the critical symbols will allow the application of RVCM algorithm to long powerful codes.

Acknowledgement

This work was partly funded by the Overseas Research Students award and the Data Storage Network, UK. The authors would like to thank Prof. Barry K. Middleton of University of Manchester for the channel noise discussion.

References

[1] R. Wood, "The feasibility of magnetic recording at 1 Terabit per square inch," *IEEE Trans. Magn.*, vol. 36, pp. 36–42, Jan. 2000.

[2] E. Papagiannis, M. A. Ambroze, and M. Tomlinson, "Received Vector Codeword Modification (RVCM)." UK Patent Application 0320126.6, Jul. 2003.

[3] E. Papagiannis, M. A. Ambroze, and M. Tomlinson, "Analysis of non convergence blocks at low and moderate snr in scc turbo schemes," *SPSC 2003 8th International worksop on Signal Processing for Space Communications*, Catania, Italy, pp. 121–128, Sep. 2003.

[4] E. Papagiannis, M. A. Ambroze, and M. Tomlinson, "Approaching the ml performance with iterative decoding," *International Zurich Seminar on Communications, Zurich, Switzerland*, pp. 220–223, Feb. 2004.

[5] C. J. Tjhai, E. Papagiannis, M. Tomlinson, M. A. Ambroze, and M. Z. Ahmed, "Improved iterative decoder for LDPC codes with performance approximating to a maximum likelihood decoder." UK Patent Application 0409306.8, Apr. 2004.

[6] E. Papagiannis, M. Ambroze, and M. Tomlinson, "Improved Decoding of Low-Density Parity-Check Codes with Low, Linearly Increased Added Complexity." Submitted to ISIT 2005, Jan. 2005.

[7] H. Pishro-Nik and F. Fekri, "Improved Decoding Algorithms for Low-Density Parity-Check Codes," *Proc. 3rd Intl. Symp. on Turbo Codes, Brest, France*, pp. 117–120, 2003.

[8] N. Varnica and M. Fossorier, "Belief Propagation with Information Correction : Imporved Near Maximum-Likelihood Decoding of Low-Density Parity-Check Codes," *Proc. of IEEE Intl. Symp. Inform. Theory (ISIT), Chichago, USA*, p. 343, July 2004.

[9] W. Tan and J. Cruz, "Signal-to-Noise Ratio Mismatch for Low-Density Parity-Check Coded Magnetic Recording Channels," *IEEE Trans. Magn.*, vol. 40, pp. 498–506, Mar. 2004.

[10] H. Sawaguchi, Y. Nishida, H. Takano, and H. Aoi, "Performance Analysis of Modified PRML Channels for Perpendicular Recording Systems," *J. Magn. Magn. Mater.*, vol. 235, pp. 265–272, 2001.

[11] M. Madden, M. Öberg, Z. Wu, and R. He, "Read Channel for Perpendicular Magnetic Recording," *IEEE Trans. Magn.*, vol. 40, pp. 241–246, Jan. 2004.

[12] R. Horan, C. Tjhai, M. Tomlinson, M. Ambroze, and M. Ahmed, "A Finite-Field Transform Domain Construction of Binary Low-Density Parity-Check Codes." to be presented in ITW 2005, New-Zealand, 2005.

[13] C. Tjhai, M. Tomlinson, R. Horan, M. Ambroze, and M. Ahmed, "GF(2^m) Low-Density Parity-Check Codes Derived from Cyclotomic Cosets." Submitted to ISIT 2005, Jan. 2005.

IMPROVED DECODING OF LOW-DENSITY PARITY-CHECK CODES BY REDUCTION OF PSEUDOCODEWORDS

Evangelos I. Papagiannis
Communications Research Group
University of Plymouth
Drake Circus PL4 8AA
Plymouth, UK
email: epapagiannis@plymouth.ac.uk

Adrian Ambroze, Martin Tomlinson, and Mohammed Z. Ahmed
Communications Research Group
University of Plymouth
Drake Circus PL4 8AA
Plymouth, UK
email: (mambroze,mtomlinson,mahmed)@plymouth.ac.uk

ABSTRACT
The paper starts with an analysis of iterative decoding based on an implementation of the Sum-Product algorithm for LDPC codes. It is shown that the iterative decoder performs optimum decoding but in a vector space different than this defined by the codebook. The crucial role of the correlation between the received vector and the pseudocodewords is justified by simulation results. Based on these results we present an algorithm which can deliver significant improvement to the iterative performance of LDPC codes, with relatively low complexity. It is shown that maximum likelihood performance can be achieved for a certain range of codes.

KEY WORDS
Coding and Modulation Techniques, LDPC codes, iterative decoding, pseudocodewords

1 Introduction

LDPC codes [1] can operate very close to channel capacity. At practical block-lengths and code rates though, the Belief Propagation algorithm only approximates the maximum likelihood performance. Iterative decoding of powerful codes with high minimum distance often fails to settle to a solution that is a valid codeword. For weaker codes the iterative decoder does provide a valid solution but this is not always the maximum likelihood one which means that the decoder is suboptimal. The sub-optimality of iterative decoders has been investigated by various authors [2][3] and has been attributed to the presence of pseudocodewords induced by small stopping sets within the structure of the code. The challenge here is to reduce, if not eliminate, the sub-optimality of iterative decoding with the minimum amount of additional processing complexity.

2 Generation of pseudocodewords during iterative decoding

In the case of optimal Maximum A Posteriori (MAP) decoding the decisions for each bit j is calculated by the sum of the probabilities (with respect to the received vector r) of codewords $\mathbf{c} \in C$ where $c_j = x \in \{0, 1\}$.

$$APP_j(x) = \sum_{\mathbf{c} \in C, c_j = x} P(\mathbf{c}|\mathbf{r}) \tag{1}$$

In the case of iterative decoding though its vector space $\mathcal{U}$ is different than the code space C ($\mathcal{U} \neq C$), which means that part or all of the vectors $\mathbf{u} \in \mathcal{U}$ do not represent valid codewords. This is due to the local operations of the Sum-Product algorithm [3]. In this paper it is shown that the distortion of the iterative vector space $\mathcal{U}$ might appear as an *unequal scaling*, as an additional *offset* [4], or as both in the iterative decoder decisions.

Example 1: Consider the Hamming (7,4) code with parity check matrix H given below.

$$\begin{array}{c} 0\ 0\ 1\ 0\ 1\ 1\ 1 \\ 0\ 1\ 0\ 1\ 0\ 1\ 1 \\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \end{array}$$

Using the Sum-Product algorithm, the decision for each bit j at any iteration would be calculated by combining the expressions obtained by the parity check equations where bit j participates. Considering the case of bit 3 (first column is bit 0) the local extrinsic information (as local is defined the extrinsic information obtained by each individual parity check equation) $e_{l_{i3}}(1)$ at iteration 0 as obtained by the parity check equations 0 and 2 can be expressed as

$$\textit{parity check 0: } e_{l_{03}}(1) = [p_1 \cdot q_5 \cdot q_6] + [q_1 \cdot p_5 \cdot q_6] + [q_1 \cdot q_5 \cdot p_6] + [p_1 \cdot p_5 \cdot p_6] \tag{2}$$

$$\textit{parity check 2: } e_{l_{13}}(1) = [p_0 \cdot q_4 \cdot q_6] + [q_0 \cdot p_4 \cdot q_6] + [q_0 \cdot q_4 \cdot p_6] + [p_0 \cdot p_4 \cdot p_6] \tag{3}$$

Where $p_j = p(1|r_j)$ and $q_j = 1 - p_j = p(0|r_j)$ for $j = 0, 1, ..., 6$. The overall extrinsic of 0 for bit 3 would be the product of the two local extrinsics as this is shown in equation 4. From equations 4 and 5 it can be seen that none of the terms of the iterative decoder output represents a valid codeword $\mathbf{c} \in C$. The vectors $\mathbf{u} \in \mathcal{U}_3$ that account for the decision of bit 3 can be considered as pseudocodewords. The decision by the iterative decoder will be based

$$
\begin{aligned}
e_3(1) = &[p_0 \cdot p_1 \cdot q_4 \cdot q_5 \cdot q_6^2] + [q_0 \cdot p_1 \cdot p_4 \cdot q_5 \cdot q_6^2] + [q_0 \cdot p_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + [p_0 \cdot p_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + \\
&[p_0 \cdot q_1 \cdot q_4 \cdot p_5 \cdot q_6^2] + [q_0 \cdot q_1 \cdot p_4 \cdot p_5 \cdot q_6^2] + [q_0 \cdot q_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [p_0 \cdot q_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] + \\
&[p_0 \cdot q_1 \cdot q_4 \cdot q_5 \cdot q_6 \cdot p_6] + [q_0 \cdot q_1 \cdot p_4 \cdot q_5 \cdot q_6 \cdot p_6] + [q_0 \cdot q_1 \cdot q_4 \cdot q_5 \cdot p_6^2] + [p_0 \cdot q_1 \cdot p_4 \cdot q_5 \cdot p_6^2] + \\
&[p_0 \cdot p_1 \cdot q_4 \cdot p_5 \cdot q_6 \cdot p_6] + [q_0 \cdot p_1 \cdot p_4 \cdot p_5 \cdot q_6 \cdot p_6] + [q_0 \cdot p_1 \cdot q_4 \cdot p_5 \cdot p_6^2] + [p_0 \cdot p_1 \cdot p_4 \cdot p_5 \cdot p_6^2]
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
e_{3,ml}(1) = &\underbrace{[p_0 \cdot p_1 \cdot q_2 \cdot q_4 \cdot q_5 \cdot q_6]}_{ML[1]} + \underbrace{[q_0 \cdot p_1 \cdot p_2 \cdot p_4 \cdot q_5 \cdot q_6]}_{ML[2]} + \underbrace{[p_0 \cdot q_1 \cdot p_2 \cdot q_4 \cdot p_5 \cdot q_6]}_{ML3]} + \underbrace{[q_0 \cdot q_1 \cdot q_2 \cdot p_4 \cdot p_5 \cdot q_6]}_{ML[4]} + \\
&\underbrace{[q_0 \cdot q_1 \cdot p_2 \cdot q_4 \cdot q_5 \cdot p_6]}_{ML[5]} + \underbrace{[p_0 \cdot q_1 \cdot q_2 \cdot p_4 \cdot q_5 \cdot p_6]}_{ML[6]} + \underbrace{[q_0 \cdot p_1 \cdot q_2 \cdot q_4 \cdot p_5 \cdot p_6]}_{ML[7]} + \underbrace{[p_0 \cdot p_1 \cdot p_2 \cdot p_4 \cdot p_5 \cdot p_6]}_{ML[8]}
\end{aligned} \tag{5}
$$

$$
e_{it_3}(1) = \underbrace{\frac{q_6}{q_2} \cdot (ML[1] + ML[4]) + \frac{p_6}{q_2} \cdot (ML[6] + ML[7]) + \frac{q_6}{p_2} \cdot (ML[2] + ML[3]) + \frac{p_6}{p_2} \cdot (ML[5] + ML[8])}_{scaling\ part} + offset \tag{6}
$$

on a pseudo space $\mathcal{U}_3 \not\subset C$, where $\mathcal{U}_3$ is a subset of the overall pseudo space $\mathcal{U}_j \subset \mathcal{U}$.

$$
\mathcal{U} = \mathcal{U}_0 \bigcup \mathcal{U}_1 \bigcup \ldots \bigcup \mathcal{U}_{N-1} \tag{7}
$$

The size $\Psi(j)$ of the pseudo-space $\mathcal{U}_j$ can be obtained from the structure of the parity check matrix H as

$$
\Psi(j) = 2 \cdot \left(\prod_{\forall h_{ij}=1} 2^{(w_{r_i}-2)} \right) \tag{8}
$$

Where w_{r_i} is the row weight of the parity check equation that bit j participates. The number of product operations is equal to the column weight of j. Clearly when $\Psi(j) > 2^k$, vectors which are not part of the codebook C will be involved in this decision, thus adding an offset to the iterative decoder's expressions.

The output of the iterative decoder (equation 4) can be alternatively expressed as in equation 6. A quick check on the H matrix can verify that bit 3 does never occur in the same equation with bit 2, thus bit 2 does not appear in any of the iterative decoder expressions for the extrinsic information of bit 3, which is why its effect is cancelled out from all terms of equation 6. On the other hand bit 6 forms a cycle of length 4 with bit 3 and that is the reason for the appearance of the squared components in equation 4, and equivalently, the reason of the extra multiplicative factors in equation 6. This provides an explanation why the presence of short cycles within the H matrix structures limits the performance of LDPC codes.

The structure of the offset is slightly more complicated. It includes all the remaining vectors of equation 4 that have been induced by the multiplication of the local sequences and are not associated with the ML solution. Notice that bit 6 appears twice in each of these vectors.

The decision of the iterative decoder for bit 3 at iteration 0, will be the result of an optimal MAP operation on the vector space $\mathcal{U}_3$. Generally

$$
APP_j(x) = \sum_{\mathbf{u} \in \mathcal{U}_j, u_j = x} p(\mathbf{u}|\mathbf{r}) \tag{9}
$$

At later iterations the iterative decoder will perform MAP decoding on the same set $\mathcal{U}$ but based on the updated received vector.

Example 2: The (7,3) code, whose decoding parity check matrix is shown below, is a perfect difference set cyclic code [5] derived from cyclotomic cosets [6]. The special structure of these kind of codes permits the use of an extended (N equations) parity check matrix at the decoder side.

$$
\begin{array}{c}
1\ 1\ 0\ 1\ 0\ 0\ 0 \\
0\ 1\ 1\ 0\ 1\ 0\ 0 \\
0\ 0\ 1\ 1\ 0\ 1\ 0 \\
0\ 0\ 0\ 1\ 1\ 0\ 1 \\
1\ 0\ 0\ 0\ 1\ 1\ 0 \\
0\ 1\ 0\ 0\ 0\ 1\ 1 \\
1\ 0\ 1\ 0\ 0\ 0\ 1
\end{array}
$$

From its H matrix we can deduce the following characteristics

- The H matrix is regular so $w_{r_j} = w_{r_i} \forall i, j < N$. Moreover, $w_{c_j} = w_{c_i} \forall i, j < N$ and $w_{r_j} = w_{c_j} \forall j < N$. That effectively means that for the decision of any bit j the vector space $\mathcal{U}_j$ involves $\Psi(j) = 2 \cdot \left(\prod_{\forall h_{ij}=1} 2^{(w_{r_i}-2)} \right)$ terms.
- $\Psi(j) = 16 > 2^k, \forall j$. Thus, there will be $16 - 2^k$ additional terms involved in the decision of any bit j.
- Every bit j neighbours with every other bit at most once. Thus, the absence of cycles of length 4 is guaranteed and for every pair of bits j and m there is an equation i for which $h_{ij} = 1$ and $h_{im} = 1$.

$$e_0(1) = \underbrace{[p_1 \cdot q_2 \cdot q_3 \cdot p_4 \cdot q_5 \cdot p_6]}_{ML[1]} + [p_1 \cdot p_2 \cdot q_3 \cdot p_4 \cdot q_5 \cdot q_6] + [p_1 \cdot q_2 \cdot q_3 \cdot q_4 \cdot p_5 \cdot p_6] + \underbrace{[p_1 \cdot p_2 \cdot q_3 \cdot q_4 \cdot p_5 \cdot q_6]}_{ML[2]} + [q_1 \cdot q_2 \cdot p_3 \cdot p_4 \cdot q_5 \cdot p_6] + \underbrace{[q_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot q_5 \cdot q_6]}_{ML[3]} + \underbrace{[q_1 \cdot q_2 \cdot p_3 \cdot q_4 \cdot p_5 \cdot p_6]}_{ML[4]} + [q_1 \cdot p_2 \cdot p_3 \cdot q_4 \cdot p_5 \cdot q_6] \quad (10)$$

$$\mathcal{O}_0 = \left\{ \underbrace{1110100, 1100011, 1001101, 1011010}_{\text{associated with bit 0 decision equal to 1}}, \underbrace{0010001, 0000110, 0101000, 0111111}_{\text{associated with bit 0 decision equal to 0}} \right\} \quad (11)$$

Using the previous analysis for bit 0, it turns out that the output of the iterative decoder at iteration 0 includes the ML terms undistorted, plus an offset since $\Psi(0) > 2^k$ for every j. The analytic output for $e_0(1)$ is shown in equation 10. Analogous results can be derived for $e_0(0)$ and for every $e_j(x)$ due to the uniform structure of the H matrix. The remaining $\Psi(0) - 2^k$ vectors that have emerged from the multiplication of the local extrinsic expressions form the offset (the non-underlined terms of equation 10). Thus the vector space $\mathcal{U}_j$ for every j at the end of iteration 0 is

$$\mathcal{U}_j = \mathcal{C} \bigcup \mathcal{O}_j \quad (12)$$

The vectors $\mathbf{o} \in \mathcal{O}_j \subset \mathcal{U}_j$ are listed in 11. The set of vectors $\mathbf{o} \in \mathcal{O}$ do not represent valid codewords but pseudocodewords. However, they are equal contributors to the *a posteriori* decisions of the iterative decoder. The APP decision of the iterative decoder for any bit j at iteration 0 is in this case expressed by

$$APP_j(x) = \sum_{\mathbf{u} \in \mathcal{U}_j, u_j = x} p(\mathbf{u}|\mathbf{r}) = \sum_{\mathbf{c} \in \mathcal{C}, c_j = x} p(\mathbf{c}|\mathbf{r}) + \sum_{\mathbf{o} \in \mathcal{O}_j, o_j = x} p(\mathbf{o}|\mathbf{r}) \quad (13)$$

3 Correlation of pseudocodewords with the received vector

Up to this point it has been shown that the iterative decoder performs optimal MAP decoding in the pseudo space $\mathcal{U}$ instead of the code space $\mathcal{C}$. As a consequence, the iterative decoder does not always converge to the optimal ML solution and is considered sub-optimal. Expanding the previous analysis to further iterations would involve an exponential increase in the number of terms since the extrinsic information that is fed back to update the received vector includes all the contributions from all the neighbouring nodes of the updated position. Simulations can easily verify that the output of the iterative decoder at iteration t is equivalent to MAP decoding (with respect to the received vector $\mathbf{r}$) on the complete set of vectors $\mathbf{u}$ that have emerged at that iteration. Alternatively, the same outcome can be obtained by applying MAP decoding to the exact set of vectors that have emerged during iteration 0, but with respect to the updated received vector.

In [3] the authors have presented similar conclusions but through the code's computation graph that is used to specify the pseudocode [7]. The authors claim that iterative decoding is optimal but on the pseudocode $\mathcal{C}'$ and not the actual code $\mathcal{C}$. They categorise the pseudocodewords as "good" and "bad" based on whether their component u_j is in favour or not of the correct decoding.

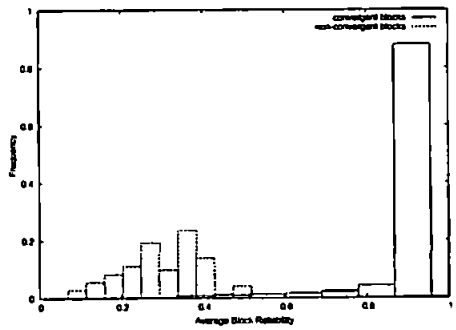


Figure 1. Histogram of the average reliabilities of the optimal MAP decoder APP decisions for blocks that converge and not converge to the ML solution when iteratively decoded by the Sum-Product algorithm. Results based on the perfect difference set (7,3) cyclic code

It can be deduced from all these that during the iterative procedure, codewords are competing with "good" and "bad" pseudocodewords. Those that exhibit higher correlation with the received vector will dominate. As long as the codewords are strongly correlated to the received vector, the iterative decoder should be able to converge to the ML solution. In the opposite case the chances that pseudocodewords are dominating increase. The histogram of figure 1 proves that case. The iterative decoder fails to converge to the ML solution in those cases where the ML decisions are not reliable enough, thus when none of the codewords $\mathbf{c} \in \mathcal{C}$ is strongly correlated to the received vector. The position with the least reliable APP decision of the ML decoder can be considered as the weakest contributor to the correlation of the codewords with the received vector. Similarly, it can be considered as the strongest contributor to the correlation of pseudocodewords with the received vector. The immediate question is if and by how much the performance of the iterative decoder would improve if the correlation of this specific bit with "good" pseudocode-

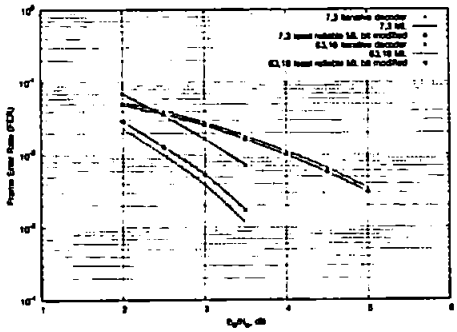


Figure 2. Improvement of iterative decoding performance by modifying the position with the least reliable ML APP decision

words and codewords that are associated with the transmitted symbol at that position, is maximised. That could be achieved by modifying the received vector at that position to one of the possible transmitted values (± 1 for BPSK modulation).
The results of figure 2 show that the improvement achieved by modification of just one of the received vector coordinates is remarkable. For the case of the (7,3) code the iterative decoder performance becomes identical to the ML. The same method applied to the (63,16) code improves the iterative performance by almost an order of magnitude and approaches the optimal ML. These results were the inspiration for the development of an algorithm that is presented in the next section.

4 The Received - Vector - Coordinate - Modification (RVCM) algorithm

4.1 Description of the algorithm

Assume an AWGN channel with BPSK modulation and output $\mathbf{r} = \{r_0, r_1, .., r_N\}$. Consider $L = \{l_0, l_1, .., l_N\}$ as the log-likelihood-ratios vector. Given that the decoder fails to satisfy the set stopping criteria, the RVCM algorithm commences. The latter can be summarised as follows

- Step1: Vector $\mathbf{L}$ is stored
- Step2: Define parameter β_{max}, $0 < \beta_{max} \leq N$
- Step3: l_β is substituted by $l'_\beta = +\infty$ if $l_\beta > 0$ and by $l'_\beta = -\infty$ if $l_\beta < 0$
- Step4: Decoding restarts
- Step5: We store the Euclidean distance $Euc(\beta)$ of the decoder output d_β from the received vector $\mathbf{r}$ when bit β has been modified and the associated output of the decoder as a candidate solution. If $\beta < \beta_{max}$ we restore the original l_β value, increment β and return to step3. If $\beta >= \beta_{max}$ we proceed to the next step
- Step6: Choose the decoder output which is associated with the minimum $Euc(\beta)$

The above algorithm can in many cases offer improvement of an order of magnitude or more in terms of FER and BER. Remarkable improvement has been also obtained by application of the algorithm in turbo code schemes [8][9][10]. For many short codes, application of RVCM achieves maximum likelihood performance as it will be shown in later section. Note that the complexity added is at most $2N$ additional decoding operations when $\beta_{max} = N$, and it increases linearly with the block length N since only one coordinate is modified at any time. However, this is just the worse case scenario since with just a fraction of modifications ($\beta \ll \beta_{max}$) we can get performance arbitrarily close to the optimum especially in the high SNR region. For convenience, the bit positions that are capable of correcting a prior non-convergent block if modified will be called *critical bits* throughout the rest of the paper.

4.2 Selection methods of critical positions

Since the positions of the least reliable ML APP decisions are unknown, various selection methods have been investigated so that the positions of critical bits are accurately identified. Generally, a large number of bits are critical for each block and that increases the chances of identifying a critical position with the minimum effort. Target of any of the selection methods is to achieve the maximum performance while the number of candidate bits β is minimised. Below, two selection criteria are presented that increase the throughput of the algorithm.

4.2.1 Method A: Channel output unreliability (UNR criterion)

The UNR criterion provides the simplest and most straight forward method of selecting candidate critical bits. The selection is based on the information content of the received vector coordinates. The coordinates with the highest entropy (or equivalently with the lowest information content) are selected first, where the entropy vector $\mathcal{H}$ for each of the components is defined as

$$\mathcal{H}_j = q_j \cdot \log_2\left(\frac{1}{q_j}\right) + p_j \cdot \log_2\left(\frac{1}{p_j}\right) \qquad (14)$$

This method narrows down the number of potential critical bits and improves the throughput of RVCM.

4.2.2 Method B: Average participation in unsatisfied equations (UNSEQ criterion)

The UNSEQ criterion is basically a combination of two measures which gather information in parallel at all iterations of standard decoding.

part1 : The first measure deals with the participation of the bits in unsatisfied equations and bases its results on the standard decoder's output at the end of each iteration. Let $\mathbf{D}^t$ be the set of thresholded APP decisions of the standard decoder at iteration t. Assuming non convergence there will be a set of equations $\mathbf{Y}^t$ that remain unsatisfied. The number of occurrences of each bit j at the set of unsatisfied equations $\mathbf{Y}^t$ at iteration t can be easily computed as

$$\mathcal{K}^t(j) = \sum_{j \in Y^t} h_{ij} \tag{15}$$

Where h_{ij} is the i_{th} row and j_{th} column component of the H matrix. At the end of the standard decoding procedure the records of the participation of each bit j in unsatisfied equations per iteration $F_1(j)$ can be obtained by

$$F_1(j) = \frac{\sum_{t=0}^{T} \mathcal{K}^t(j)}{T} \tag{16}$$

where T the number of iterations performed by the standard decoder.
The information from $F_1(j)$ can be combined with a second measure and considerably enhance the performance of the UNSEQ selection criterion.

part2 : This second measure deals with the set of the updated probabilities obtained at the end of each iteration of the Sum-Product algorithm. Considering the row updating implementation, the local extrinsic information $e_{l_{ij}}$ of each equation participant j is calculated for all those positions where h_{ij}=1. When all parity checks have been processed, the overall extrinsic information e_j for each bit j is calculated as the product of all local extrinsics for this specific bit.

$$e_j(x) = \prod_{\forall h_{ij}=1} e_{l_{ij}}(x) \tag{17}$$

The updated probabilities upd_{ij} $\forall h_{ij} = 1$ can be then calculated for all individual participants by multiplication of the channel probability $p_j(x|r_j)$ with the overall extrinsic information $e_j(x)$, omitting though the corresponding local extrinsic value to prevent positive feedback.

$$upd_{ij}(x) = \frac{p_j(x|r_j) \cdot e_j(x)}{e_{l_{ij}}(x)} \tag{18}$$

Consider now a thresholded version $\mathbf{upd}'$ of the updated matrix. Since the updated values are locally optimised it is guaranteed that they satisfy all parity check equations ($\mathbf{Y} = \emptyset$), but on the other hand it is not guaranteed that

$$upd'_{ij|h_{ij}=1} = upd'_{mj|h_{mj}=1} \tag{19}$$

In other words we have the case that a certain bit j satisfies two equations with two different values, and that points to the presence of even error equations that would not be detected if only the $F_1(j)$ measure of the first part of the UNSEQ criterion had been used. To decide which of the two updated values for j is the erroneous one a minority decision is applied. The value of j that appears with lowest frequency is considered as erroneous and the number of occurrences can be used as a measure $F_2(j)$ of the participation of bit j in even error equations.
Thus, the total $F(j)$ parameter of the UNSEQ criterion which forms the basis for the selection of the candidate critical bits, is given by the sum of $F_1(j)$ and $F_2(j)$.

$$F(j) = F_1(j) + F_2(j) \tag{20}$$

Like in the UNR criterion case, selecting the bits with the highest $F(j)$ values narrows down the choice for the potential critical bits with minimum losses from the optimum RVCM performance.
In many cases the UNSEQ criterion can deliver considerably better performance when β is kept very low, especially for codes with low row weight w_r. For higher values of β the UNR criterion performs slightly better in most of the times.

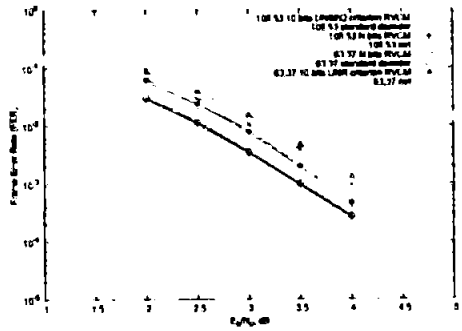


Figure 3. RVCM performance of the cyclic codes (105,53) and (63,37)

4.3 The effect of RVCM in the reduction of pseudocodewords

The effect of RVCM can be easily demonstrated in cases where unequal scaling occurs during iterative decoding. Consider the analytical output of the Hamming(7,4) code at iteration 0 (equation 4). Modification of bit 6 would eliminate most of the pseudocodewords and would minimise the distortion of the vector space $\mathcal{U}$ since if p_6=1 then $p_6^2 = p_6$. Equivalently if q_6=1 then $q_6^2 = q_6$ and the scaling problem disappears. Moreover, since every term of the offset includes both p_6 and q_6 modification of bit 6 would set offset to zero. So the $e_0(1)$ at this case would be as in equation 21. Note that scaling problems will occur for all codes after iteration 0 because of the update operation from the overall extrinsic vector $e_j(x)$ which contains information from a large span of nodes.

4.4 RVCM performance

Figures 3 and 4 show typical performances of codes when RVCM is applied. Note that for the case of the short

$$e_3(1) = \frac{ML[1]+ML[4]}{q_2} + \frac{ML[6]+ML[7]}{q_2} + \frac{ML[2]+ML[3]}{p_2} + \frac{ML[5]+ML[8]}{p_2} \quad (21)$$

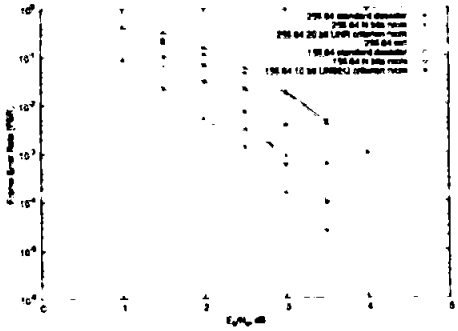


Figure 4. RVCM performance of the cyclic (255,64) and the Tanner(155,64) codes

weak codes of figure 3, application of the RVCM algorithm achieves performance identical to ML, even when $\beta_{max} \ll N$. The ML performance is approximated by the mrl [11] curves which represent a practical lower bound. Let $\mathbf{R}$ be the received, $\mathbf{Tx}$ be the transmitted and $\mathbf{D}$ the error decoded vectors in the euclidean space. Then a block is considered as mrl when

$$Euc(\mathbf{D},\mathbf{R}) < Euc(\mathbf{Tx},\mathbf{R}) \quad (22)$$

In this case we know for sure that the correct codeword does not coincide with the closest to the received vector since at least one error codeword is closer (is more likely) than that and the ML decoder would fail. The RVCM improvement is more spectacular in the case of the longer and higher d_{min} codes of figure 4 where for $\beta_{max} = N$ the reduction in Frame Error Rate (FER) exceeds the two orders of magnitude at higher signal to noise ratios.

5 Conclusion

This paper deals with the reasons for the sub-optimality of iterative decoding. The problem was approached from a point of view that relates the convergence problem to the generation and interaction of pseudocodewords with the valid codewords of the code. The analysis was carried out through simple examples on short codes, and was based on the row-update implementation of the Sum-Product algorithm. It was shown that the iterative decoder performs optimum MAP decoding on a pseudo-space and the chances that the decoder does not converge to the ML solution increase when pseudocodewords are highly correlated to the received vector. The histogram of figure 1 strengthened that view since for blocks that the iterative decoder failed to converge their actual correlation with the available codewords was low (average APP decisions reliability of ML decoder was low). Modification in the received vector value of the coordinate with the least reliable APP decision at the output of the ML decoder was shown to improve dramatically the iterative performance. This approach was based on the observation that a coordinate that is weakly correlated to the codewords can be considered as a major contributor to the correlation of specific pseudocodewords with the received vector.
The effect of changing a single bit on the outcome of the iterative decoder, inspired the creation of a novel algorithm that exploits this characteristic. Simulations proved that the RVCM algorithm is capable of providing significant coding gain and even eliminate the gap between iterative and ML decoding in some cases. Results that show reductions in terms of FER that exceed two orders of magnitude have been presented.

References

[1] R.G.Gallager,"Low-density parity-check codes", PhD, MIT, 1963.

[2] T.Richardson and R.Urbanke, "Efficient encoding of low-density parity-check codes",*IEEE Tansactions on Information Theory*,vol.47, pp.638-656, feb. 2001.

[3] B.Frey, R.Koetter, and A.Vardy, "Signal-space characterization of iterative decoding",*IEEE Tansactions on Information Theory*,vol.47, pp.766-780, feb. 2001.

[4] M.Z.Ahmed, "Crosstalk-resiliant coding for high density digital recordings",PhD, University of Plymouth, UK, Mar.2003.

[5] E.Weldon, "Difference-set cyclic codes", *The Bell System Technical Journal*,vol. XXXVIII, no.45, pp. 1045-1055, Sept. 1966.

[6] C.Tjhai, M.Tomlinson, R.Horan, M.Ambroze. and M.Z.Ahmed, "GF(2^m) low-density parity-check codes derived from cyclotomic cosets", *preprint*,2005.

[7] N.Wiberg, "Codes and decoding on general graphs",PhD, Linkoping University, Sweden, Apr. 1996.

[8] E.Papagiannis, M.Ambroze, and M.Tomlinson, "Analysis of non convergent blocks at low and moderate SNR in SCCC turbo schemes", *Proc. SPSC 2003* 8^{th} *International workshop on Signal Processing for Space Communications*, Catania, Italy, pp.121-128, Sept.2003.

[9] E.Papagiannis, M.Ambroze, and M.Tomlinson, "Approaching the ML performance with iterative decoding", *Proc. International Zurich Seminar on Communications*, Zurich, Switzerland, pp.220-223, Feb.2004.

[10] E.Papagiannis, M.Ambroze, and M.Tomlinson, "UK patent 0320126.6."

[11] M.Ferrari, S.Bellini, M.Tomlinson and, M.Ambroze, "Backbone interleaver designfor multi-binary turbo codes",*Proc.* 3^{rd} *International Symposium on Turbo Codes*, Brest, France, Sept.2003.

References and Bibliography

[1] D. Agrawal. *GMD decoding of Euclidean-space codes and iterative decoding of turbo codes.* PhD thesis, University of Illinois, Urbana, USA, 1999.

[2] D. Agrawal and Vardy A. The turbo decoding algorithm and its phase trajectories. *IEEE Transactions on Information Theory*, 47(2):699–722, February 2001.

[3] M.Z. Ahmed. *Crosstalk-resiliant coding for high density digital recording.* PhD thesis, University of Plymouth, Plymouth, UK, March 2003.

[4] E.F. Assmus, J.M. Goethals, and H.F. Mattson Jr. Generalised t-designs and majority decoding of linear codes. Inform. Contr. vol.32, pp. 43-60, 1976.

[5] L.R. Bahl, J. Cocke, F. Jelenik, and J. Raviv. Optimal decoding of linear codes for minimising sybmol error rate. *IEEE Transactions on Information Theory*, 20:284–287, March 1974.

[6] S. Benedetto and M. Montorsi. Unveiling turbo codes: some results on parallel concatenated coding schemes. *IEEE Transactions on Information Theory*, 42(2):409–429, March 1996.

[7] S. Benedetto, M. Montorsi, D. Divsalar, and F. Pollara. Serial concatenation of interleaved codes: Performance analysis, design and iterative decoding. *JPL TDA Progress Report*, pages 42–126, August 1996.

[8] E. Berlekamp. Non-binary bch decoding. Institute of Statistics, 1966. University of North Carolina, Chapel Hill, N.C.

[9] C. Berrou, P. Thitimajshima, and A. Glavieux. Near Shannon limit error correcting coding and decoding: turbo codes. In *Proc. IEEE International Conference on Communications*, pages 1064–1070, Geneva, Switzerland, May 1993.

[10] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error correcting binary group codes. Inform. Contr., 1960. Vol.3 68-79.

[11] D. Chase. A class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory*, pages 170–182, 1972.

[12] Y.S. Chung. On the design of low-density parity-check codes within 0.0045db of the shannon limit. IEEE Commun. Lett., Feb. 2001. vol.5, pp.58-60.

[13] D.J. Costello Jr., J. Hagenauer, H. Imai, and S.B. Wicker. Applications of error control coding. *IEEE Transactions on Information Theory*, 44(6):2531–2560, October 1998.

[14] D. Divsalar, S. Dolinar, and F. Pollara. Iterative turbo decoder analysis based on density evolution. *tdapr*, (42-124):1–33, feb 2001.

[15] D. Divsalar and F. Pollara. Hybrid concatenated codes and iterative decoding. ISIT '97.

[16] D. Divsalar and F. Pollara. Serial and hybrid concatenated codes with applications. Brest, France, September 1997.

[17] B.G. Dorsch. A decoding algorithm for binary block codes and j-ary output channel. *IEEE Transactions on Information Theory*, IT-20(3):391–394, 1974.

[18] R.M. Fano. A heuristic discussion of probabilistic decoding. *IEEE Transactions on Information Theory*, IT-9:64–74, April 1963.

[19] J. Feldman and D.R. Karger. Decoding turbo like codes via linear programming. Proc. 43rd annual IEEE Symposium on Foundations of Computer Science (FOCS). November 2002.

[20] J. Feldman, M.J. Wainwright, and D.R. Karger. Using linear programming to decode linear codes. 37th annual Conference on Information Sciences and Systems (CISS'03). May 2003.

[21] M. Ferrari, S. Bellini, M. Tomlinson, and M. Ambroze. Backbone interleaver design for multi-binary turbo codes. Brest, France, September 2003.

[22] G.D. Forney. *Concatenated codes.* Cambridge, MA: M.I.T. Press, 1966.

[23] G.D Forney Jr. The viterbi algorithm. Proc. IEEE, Mar.1973. vol.61, pp.268-278.

[24] M. Fossorier and S. Lin. Chase-type and gmd coset decodings. *IEEE Transactions on Communications*, 48(3):345–350, 2000.

[25] M.P.C. Fossorier and S. Lin. Soft decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, IT-41(5):1379–1396, 1995.

[26] M.P.C. Fossorier and S. Lin. Computationally efficient soft decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, IT-42(3):738–750, 1996.

[27] B.J. Frey, R. Koetter, and A. Vardy. Signal-space characterization of iterative decoding. *IEEE Transactions on Information Theory*, 47(2), February 2001.

[28] R.G. Gallager. *Low-Density Parity-Check codes.* MA:MIT Press, Cambridge, 1963.

[29] D. Gnaedig, E. Boutillon, M. Jezequel, V. Gaudet, and P. Gulak. On multiple slice turbo codes.

[30] M.J.E. Golay. Notes on digital coding. 37:657, 1949.

[31] R.W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.

[32] R.V.L. Hartley. Transmission of information. *The Bell System Technical Journal*, 7(3):535–563, July 1928.

[33] J. Hokfelt, O. Edfors, and T. Maseng. Turbo codes: Correlated extrinsic information and its impact on iterative decoding performance.

[34] C. Kelley and D. Shridkara. Structure of pseudo-codewords in tanner graphs. Proc. 2004 International Symposium on Information Theory and its Applications. Parma, Italy, Oct 2004.

[35] P. Koetter and P. Vontobel. Graph covers and iterative decoding. Brest, France, September 2003.

[36] Yu Kou, Shu Lin, and Marc Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Transactions on Information Theory*, 47(7):2711–2736, November 2001.

[37] J.L. Massey. Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory*, IT-15:122–127, 1969.

[38] D.J.C. McKay and R.M. Neal. Near shannon limit performance of low-density parity-check codes. *Electronics Letters*, 32:1645–1646, 1996.

[39] D.E. Muller. Application of boolean algebra to switching circuit design and to error detection. IRE Trans. Electron. Computers, 1954. EC-3, 6-12.

[40] H. Nyquist. Certain factors affecting telegraph speed. *The Bell System Technical Journal*, 3(2):324–346, April 1924.

[41] E. Offer and E. Soljanin. An algebraic description of iterative decoding schemes. Codes, Systems, and Graphical Models, IMA Volumes in Mathematics and its Applications, v.123, Springer-Verlag, 2001.

[42] E. Papagiannis, A. Ambroze, M. Tomlinson, and M.Z. Ahmed. Improved decoding of low-density parity-check codes by reduction of pseudo-codewords. The 4th IASTED International Conference on Communication Systems. Benidorm, Spain, Sept.2005.

[43] E. Papagiannis, M.A. Ambroze, and M. Tomlinson. Analysis of non convergent blocks at low and moderate snr in sccc turbo schemes. Catania, Italy, sep 2003. ESA.

[44] E. Papagiannis, M.A. Ambroze, and M. Tomlinson. Approaching the ml performance with iterative decoding. Zurich, Switzerland, February 2004. IEEE International Zurich Seminar.

[45] E. Papagiannis, M.A. Ambroze, and M. Tomlinson. Improved iterative decoder with performance approximating to a maximum likelihood decoder. Granted UK Patent GB2405562, March 2006.

[46] Lance C. Perez, Jan Seghers, and Daniel J. Costello. A distance spectrum interpretation of turbo codes. *IEEE Transactions on Information Theory*, 42(6):1698–1709, November 1996.

[47] F. Perez-Gonzalez, J. Hernandez, and F. Balado. Approaching the capacity limit in image watermarking: a perspective on coding techniques for data hiding applications. *Signal Processing*, 81(6):1215–1238, June 2001.

[48] H. Pishro-Nik and F. Fekri. Improved decoding algorithms for low-density parity-check codes. Brest, France, September 2003.

[49] E. Prange. Cyclic error-correcting codes in two symbols. Rept. AFCRC-TN-57-103, 1957. USAF Cambridge Research Centre, Bedford, Mass.

[50] I.S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *IEEE Transactions on Information Theory*, PGIT-4:38–49, 1954.

[51] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math, 1960. 8, 300-304.

[52] A.C. Reid, T.A. Gulliver, and D.P. Taylor. Convergence and errors in turbo-decoding. *IEEE Transactions on Communications*, 49(12):2045–2051, December 2001.

[53] T. Richardson, M. Shokrolari, and R. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47:638–656, February 2001.

[54] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of provably good low-density parity-check codes. *IEEE Transactions on Information Theory.*

[55] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message passing decoding. *IEEE Transactions on Information Theory*, 47:599–618, February 2001.

[56] T.J. Richardson. The geometry of turbo decoding dynamics. *IEEE Transactions on Information Theory*, 46:9–23, January 2000.

[57] A. J. Rogers, M. Tomlinson, M. A. Ambroze, and M. Z. Ahmed. Enhancement of turbo codes using cyclic redundancy check with interleaver optimisation. Chicago, USA, jun 2004. IEEE International Symposium on Information Theory.

[58] W. Sawyer. *Numerical functional analysis.* Oxford University Press, 1978.

[59] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[60] C.E. Shannon. Communication in the presence of noise. 37(1):10–21, January 1949.

[61] E. Soljanin and E. Offer. Bit optimal decoding of codes whose tanner graphs are trees. *IEEE Transactions on Information Theory*, 2003. preprint.

[62] T.A. Summers and S.G. Wilson. Snr mismatch and online estimation in turbo decoding. *IEEE Transactions on Communications*, 46(4), April 1998.

[63] Y.O. Takeshita, O. Collins, P.C. Massey, and D.J. Costello Jr. On the frame-error rate of concatenated turbo codes. *IEEE Transactions on Communications*, 49(4):602–608, April 2001.

[64] R.M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, IT-27(5):533–547, September 1981.

[65] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

[66] T.M. Thompson. *From error-correcting codes through sphere packings to simple groups.* The mathematical association of America, 1983.

[67] T. Tian, C. Jones, J.D. Villasenor, and R.D. Wesel. Construction of irregular ldpc codes with low error floors. IEEE, 2003.

[68] C. Tjhai, M. Tomlinson, R. Horan, A. Ambroze, and M.Z. Ahmed. Low-density parity-check codes derived from cyclotomic cosets. preprint, 2005.

[69] G. Ungerboeck. Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, IT-28(1):55–67, January 1982.

[70] N. Varnica and M. Fossorier. Belief propagation with information correction: improved near maximum likelihood decoding of low-density parity-check codes. ISIT 2004. Chicago, USA.

[71] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, April 1967.

[72] P.O. Vontobel, R. Smarandache, N. Kiyavash, J. Teutsch, and Vukobratovic D. On the minimal pseudo-codewords of codes from finite geometries. Proc. Intern. Symp. on Inform. Theory. Adelaide, Australia, Sept 2005.

[73] E. Weldon. Difference-set cyclic codes. *The Bell System Technical Journal*, XXXVIII(45):1045–1055, September 1966.

[74] N. Wiberg. *Codes and decoding on general graphs.* PhD thesis, Linkoping University, Linkoping, Sweden, April 1996.

[75] A. Worm, P. Hoeher, and N. Wehn. Turbo decoding without snr estimation. IEEE Communications Letters. vol.4, no.6, June 2000.

[76] J. Wozencraft and I. Jacobs. *Principles of communication engineering.* John Wiley and Sons, Inc., 1965.

[77] Weiss Y. Correctness of local probability propagation in graphical models with loops. Neural Computation. 12(1):1-41, 2000.