NOVEL METHODS IN THE IMPROVEMENT OF TURBO CODES AND THEIR DECODING

A. J. Rogers

Ph.D. February 18, 2013

NOVEL METHODS IN THE IMPROVEMENT OF TURBO CODES AND THEIR DECODING

by

ANDREW JOHN ROGERS

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

DOCTOR OF PHILOSOPHY

School of Computing and Mathematics
Faculty of Science and Technology

February 18, 2013

Abstract

The performance of turbo codes can often be improved by improving the weight spectra of such codes. Methods of producing the weight spectra of turbo codes have been investigated and many improvements were made to refine the techniques. A much faster method of weight spectrum evaluation has been developed that allows calculation of weight spectra within a fews minutes on a typical desktop PC. Simulation results show that new high performance turbo codes are produced by the optimisation methods presented. The two further important areas of concern are the code itself and the decoding. Improvements of the code are accomplished through optimisation of the interleaver and choice of constituent coders. Optimisation of interleaves can also be accomplished automatically using the algorithms described in this work.

The addition of a CRC as an outer code proved to offer a vast improvement on the overall code performance. This was achieved without any code rate loss as the turbo code is punctured to make way for the CRC remainder. The results show a gain of 0.4dB compared to the non-CRC (1014,676) turbo code.

Another improvement to the decoding performance was achieved through a combination of MAP decoding and Ordered Reliability decoding. The simulations show a performance of just 0.2dB from the Shannon limit. The same code without ordered reliability decoding has a performance curve which is 0.6dB from the Shannon limit. In situations where the MAP decoder fails to converge ordered reliability decoding succeeds in producing a codeword much closer to the received vector, often the correct codeword. The ordered reliability decoding adds to the computational complexity but lends itself to FPGA implementation.

Table of Contents

List of Acronyms

2-PAM	2 level Pulse Amplitude Modulation
4-PAM	4 level Pulse Amplitude Modulation
ACM	Adaptive Coding and Modulation
AM	Amplitude Modulation
APP	A Posteriori Probability
ASK	Amplitude Shift Keying
AWGN	Additive White Gaussian Noise
BCH	Bose, Chaudhuri and Hocquenghem
BCJR	Bahl, Cocke, Jelinek and Raviv
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CRC	Cyclic Redundancy Check
dB	decibel
DC	Direct Current
DVB	Digital Video Broadcasting
DVB-RCS	Digital Video Broadcasting - Return Channel via Satellite
DVB-S2	Digital Video Broadcasting - Satellite 2^{nd} Generation
FER	Frame Error Rate
FM	Frequency Modulation
GF	Galois Field
KJV	King James Version
LDPC	Low Density Parity Check
MAP	Maximum A posteriori Probability
MLA	Maximum Likelihood Asymptote
MRL	More Likely
ORLD	Ordered Reliability List Decoder
PC	Personal Computer
PCCC	Parallel Concatenated Convolutional Code
PCM	Pulse Coded Modulation
PEG	Progressive Edge Growth
PM	Phase Modulation
QPSK	Quadrature Phase Shift Keying
RSC	Recursive Systematic Convolutional
Rx	Receiver
SNR	Signal to Noise Ratio
TV	Television
Tx	Transmitter
UB	Union Bound
VCM	Variable Coding and Modulation

List of Tables

List of Figures

Acknowledgements

Many thanks to the Engineering and Physical Sciences Research Council (EPSRC) for their support by means of grant funding. Funding has allowed many academic and personal achievements to be made since 2002.

Throughout my research period I have received much valued support. In particular I would like to thank Professor Martin Tomlinson, Dr. Marcel Ambroze and Dr. Mohammed Ahmed for the supervisory support, encouragement and guidance they have given me. Thanks also go to Xin Xu, Cen Jung Tjhai, Jing Cai, Evangelos Papagiannis, Purav Shah, Li Yang, Ismail Isnin, Kieko Takeuchi and Andrea Tieghi for their friendship and great parties. I would like to acknowledge the support of my parents for their unlimited encouragement and patience, without which I would not have reached this stage. I would also like to acknowledge the friendly support given by Professor Mick Fuller and the Graduate School, David Evans (coordinating Chaplain and especially for the great Chaplaincy walks) and John Peel (my pastor). Thanks must also go to Waterfront City Church, Plymouth, and to my Christian family who have provided endless prayer to encourage me.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Graduate Committee.

This study was financed with the aid of a studentship from the Engineering and Physical Sciences Research Council.

During this study many contributions to the field of turbo coding theory have been made. The addition of a CRC outer-code vastly improved performance of the turbo code and decoding. This work was presented at the world class ISIT2004 conference, also a patent was awarded. The combination of ORD and MAP decoding also gave good decoding improvement.

Conference presentations:

2004 IEEE International Symposium on Information Theory, Chicago, IL USA

EPSRC/IEE/IEEE sponsored PREP 2005 conference.

Patents:

A. Rogers, M. Ambroze, and M. Tomlinson "Cyclic redundancy check for error correcting codes with minimal code rate loss", UK Patent GB2407945

Word count of main body of thesis: 26,219

Signed Date

1 Introduction and Literature Review

1.1 Thesis Structure and Findings

Chapter 1 introduces communications and in particular digital communications. Section 1.2.10 gives an introduction to turbo codes. A background and literature review in given in Sections 1.2.20 and 1.2.21. Although constructed from convolutional encoders the turbo code can be analysed as a block code because of the framing necessary for interleaving. This is discussed in Chapter 2. The choice of interleaver was found to have a significant impact upon the performance of the turbo code. New interleaver design methods are introduced in Chapter 3. The ultimate test for performance used in this work is the computer simulation of the effect an AWGN channel has on the code. However, it is shown that weight spectrum analysis can give a prediction of performance prior to running lengthy simulations. This is discussed in Chapter 4 along with interleaver modification based on codeword weights.

1.2 General Introduction to Communications

A very obvious characteristic that distinguishes man from other animals is his ability to communicate complex concepts and at high speed. Even before the advent of technology assisted communications man has benefited from his ability to communicate in a manner far more advanced than in any other species. Along with audible communications (primitive noises, groanings and speech) man has used forms of communication that are visible where audible forms are not suited. Cave paintings and early forms of Chinese characters have been used for longer term storage of information and for cases where both parties

have not been simultaneously present. Smoke signals and fire have also been used for communications as these can be seen over distances far greater than possible with audible communications. Raised fire baskets placed around the coast warned of the approach of the Spanish Armada 1588.

Dynamic development has risen out of our ability to communicate and record information so that it can be passed down through our generations for their benefit. Discoveries and theories are continuously being developed by the next generation leading to our current-day standard of living. Nearly every development in electrical technology has been exploited and fashioned into some form of communications system; from the simple battery cell up to the latest multi-core microprocessor, all have played a major role in communications. In most cases our desire for more efficient communications has driven this technology.

The use of electrical signals for the transmission of information has replaced traditional methods of communication. In the past, messages have been carried by runners, carrier pigeons, drum beats, and torches[2, pg 1]. Electrical communication systems allow the transmission of information over much larger distances at speeds approaching the speed of light. Early systems provided for human to human communications and carried telegrams or voice. Communications systems make use of many fields such as electronics, electromagnetics, computing and statistics to name just a few. The fundamental purpose of a communication system is to carry information from the source to the destination some distance away. Many sources of information exist but are broadly categorised into two types, analogue and digital. An analogue source produces a smooth continuously varying signal, examples include speech, temperature and light intensity. The communications system must be able to carry and reproduce this signal with a certain degree of fidelity. Digital sources have discrete values like the letters that make up this text. There are 26 letters to the English alphabet and there are no valid values between consecutive letters. The advances in digital computers, particularly small embedded microcontrollers,

has led to increased use of digital communication channels. A digital channel must be able to convey information at an acceptable error rate.

1.2.1 Electricity and it's use in communication

Today's young experimenters would most likely demonstrate a simple electrical communications system by laying a length of wire between a battery and a torch light bulb. Touching the wires on the battery terminals would cause a current to flow to the light bulb to produce light, the light can be turned on and off by connecting and disconnecting the wire to/from the battery terminal.

Samuel F. B. Morse

In the 1830s light bulbs were not readily available when Samuel F. B. Morse developed the first electric telegraph system. Morse used an electromagnetic transducer in his experiments. Samuel F. B. Morse, in 1838, demonstrated his telegraph in New York. The message "Attention, the Universe, by kingdoms right wheel." was transmitted over ten miles using Morse code. Later, in 1844, Morse transmitted the bible passage (Numbers 23:23, KJV) "What hath God wrought!" chosen by Annie Ellsworth who had brought Morse the news that his bill was passed allowing him to place a wire between Baltimore and Washington, a distance of forty miles.

Morse's system used a key (form of on-off switch) to connect and interrupt the current to the electromagnetic receiver. In this system the current has two values determined by the position of the key. The key was operated for two distinct periods of times, the shorter time is referred to as a *dot* and the longer time as a *dash*. The dot and dash form the two symbols of a binary system and thus Morse had developed the first digital electrical telecommunications system. In the Morse Code letters of the alphabet are represented by a variable length sequence of dots and dashes. The most common letters have the shortest sequences. Morse, not only provided a means of communication over a distance, he also

provided a means of data compression. Morse had addressed two fundamental topics in communication theory. The first is encoding the most used letters into the shortest sequences, a form of data compression. The second topic he addressed was reliably getting the information to the receiver. In today's teaching of digital communications these two topics have become distinct and are referred to as *source coding* and *channel coding*, these are described later.

1.2.2 Elements of a Communication System

There are three main elements that every useful communication system must have. These elements are shown in Figure 1.1. Many forms of information source exist and are

Figure 1.1: Basic elements of a communication system

categorised into two main types; analogue and digital. Analogue sources are usually translated into a continuous voltage signal before being transmitted over the channel. The most prevalent example of an analogue source is speech. Digital sources include Morse code used in early telegraph systems. More recently the information age has presented many forms of digital sources due to the wide-spread use of computers. Digital sources are non-continuous and are represented by a sequence of discrete values.

The channel is the medium that connects the source to the receiver. In the case of radio we are concerned with electro-magnetic channels. However, before radio, Morse used a simple electrical circuit and switched on and off a current. However many forms of channel exist, for speech air is compressed and rarefied in a longitudinal wave. Two cans, one tied on each end of a taut string, can be used as a communication link. The taut string becomes the communications channel. As nearly all channels are analogue we

must convert digital sources to analogue for transmission, see Section 1.2.5.

The role of the receiver is to receive the signal and recover the information that was sent at the source. For Morse's telegraphy system this was a tape which was slowly moved at a steady speed and a pencil that was connected to the armature of a solenoid. The solenoid converts the current into a mechanic force that drives the pencil into contact with the moving tape. In a wired telephony system the speaker or earpiece performs the role of the receiver converting transitions in electrical current into sound waves.

1.2.3 More Complete Communications System Model

The simple communications model described above only shows the outline. In order to understand the theory and accommodate radio communication systems we need to develop the model. Perhaps the most significant addition to the model is *noise*. One of the biggest challenges to a successful communications system is the way it operates in the presence of noise. If you are having a conversation with a colleague on a building site where there is a lot of noise you may find yourself shouting. What you are doing is increasing the signal power to overcome the noise. Radio communications is susceptible to electrical noise and in early broadcast systems noise was overcome by using very high transmitter (Tx) powers, they shouted. Figure 1.2 shows the elements of a communications system.

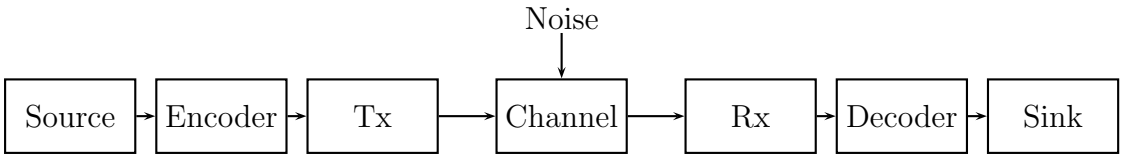

Figure 1.2: More complete model of a communication system

The encoder encodes the message into a format that can be transmitted and compatible with the receiver. We saw earlier in the discussion of Morse Code how letters were

mapped into a series of dots and dashes, this is a form of encoding. The sequences of dots and dashes are referred to as codewords. It is this part of the communications link that this thesis is most concerned with. The encoder can also map letters so that the codeword has redundancy and this can be used to improve performance in the presence of noise. This can be easily demonstrated in Morse Code by transmitting each sequence three times instead of just once. The added redundancy allows the receiver to compare the three received sequences and if one doesn't match choose the two that match as the valid codeword. Of the two matching codewords either can then be decoded to the correct letter. This type of decoding is known as 'majority logic' decoding.

The *transmitter* (Tx) processes the codeword to produce a transmission signal that is appropriate for the characteristics of the channel. This almost always involves modulation of a carrier as the source signal itself is rarely of a suitable form for transmission over a distance. Both analogue and digital forms of modulation are discussed later in Sections 1.2.4 and 1.2.5. The *channel* is the medium that constitutes the transmission path and connects the transmitter and *receiver* (Rx) that are some distance apart. This medium might be a wire, a beam of light or radio. The channel with have an associated loss, not all the transmitted energy gets to the receiver. In the case of a radio channel nearly all the energy is lost and only a small fraction is available to the receiver. Generally the larger the transmission distance the greater the loss or attenuation. The receiver amplifies the received signal to compensate for the loss of energy in the channel. It then demodulates the transmission signal to recover the signal at the source. It performs the reverse function of the transmitter.

The channel is subject to many unwanted electrical effects such as attenuation, distortion, interference and noise, particularly for radio channels. Distortion is often caused by non-linear behaviour of the components of the system. Most transistorised circuits are not perfectly linear and will introduce some distortion. Interference may occur due to other transmitters, power distribution networks, motors, switches, etc. Filtering can help

eliminate interference that has a frequency band outside of the desired signal frequency band. Noise is the random electrical energy that is naturally present, thermal noise is one example. Thermal noise occupies all range of frequencies. Filtering eliminates a large amount of this noise but leaves some noise present at the desired signal frequency band. However, the noise is superimposed on the carrier and can corrupt the message. Noise therefore is a fundamental system limitation.

1.2.4 Analogue communication

Whilst this thesis is concerned with digital communications it is worthwhile to give analogue communications a mention as it will help us gain a better understanding of the benefits of digital. Analogue communications is also a good place to start our discussion of modulation.

The first analogue electrical communication system was the telephone patented by Alexandra Graham Bell in March 1876. It was a wired system that consisted simply of a microphone, speaker, battery and wire. No electronic amplifiers were available at this time and the sound heard at the receiver was faint even when the person who was speaking at the transmitter was shouting. In 1876, sometime after he invented the microphone, Bell invented his "electrical speech machine" which we now know as the telephone. In 1878 Bell had set up the first telephone exchange in New Haven, Connecticut. In 1884 long distance connections were made between Boston, Massachusetts and New York City.

Wire can carry all frequencies of speech and can even carry DC current. Thus wired telephony is possible without modulation. However, wireless channels cannot adequately carry speech frequencies, they require a transmitter that performs modulation. Essentially converting speech frequencies to higher frequencies that can be carried over a wireless channel. The wireless receiver has to perform the reverse operation, demodulation.

Analogue Modulation

Where a channel cannot directly carry the information signal a carrier signal is generated than can be carried over the channel. The carrier signal is nearly always higher in frequency than the information signal. The carrier is a repetitive waveform such as a sine wave that has three main parameters; *amplitude*, *frequency* and *phase*. A sinusoidal carrier signal can be defined as

$$v_c = E_c \cos(\omega_c t + \phi_c)$$
$$\begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ \textbf{AM} & \textbf{FM} & \textbf{PM} \end{array}$$

Any one of three parameters can be varied with time to modulate the carrier, thus the information signal can be used to modulate the carrier. The demodulator in the receiver (called a detector is some texts) detects the variation in the carrier and recovers the information signal. By varying E_c the amplitude of the carrier signal can be made to follow the information signal, this is known as *Amplitude Modulation (AM)*. Similarly ω_c and ϕ_c can be varied to provide *Frequency Modulation (FM)* and *Phase Modulation (PM)* respectively.

1.2.5 Digital Communication

Digital communication is involved with the transmission of information in a digital form. The sources themselves need not be digital as the can be digitised as in the case with digital cellular phone networks. With the boom in the digital era you could be excused for thinking that digital communications is a more recent technology. However, as we have seen, the first telegraph system developed by Morse was digital. Telegraph systems required letters to be encoded into Morse Code and this could only be done by trained operators. Bell's telephony system was quickly adopted as it could carry speech and removed the manual encoding operation. This system and many others following it

were analogue. It wasn't until computers were available that the digital communications revival began. Computers allowed encoding to be done automatically and digital systems no longer needed trained operators to encode and decode messages.

Émile Baudot, in 1875, developed a system of coding where every letter was encoded into a fixed-length 5-bit binary codeword[3, pg 13][4, pg 12]. His work found applications in telegraph systems. As newer telegraph systems emerged many used modified versions of the Baudot code.

Nyquist

The early theoretical work of Harry Nyquist in [5] determined the maximum signalling rate that could be used over a channel of a given bandwidth without intersymbol interference. Nyquist's work was the precursor to our understanding of modern digital communications systems. His model of a telegraph system modelled the transmitted signal as

$$s(t) = \sum_n a_n g(t - nT) \tag{1.1}$$

where $g(t)$ represents the basic pulse shape and a_n are the ± 1 mappings of the binary data sequence transmitted periodically with the interval T seconds per bit[3, pg 13][4, pg12]. Nyquist found that the optimum pulse shape, band-limited to B Hz and maximise the bit rate, to be $g(t) = \sin(2\pi Bt)/2\pi Bt$. He concluded that the maximum pulse rate is $2B$ pulses per second, widely known as the *Nyquist rate*. For each sampling point the corresponding pulse shape has an amplitude defined by the sample value, a_n, for other sampling points the pulse shape value is zero and thus does not interfere with other samples. This is shown graphically in Figure 1.3.

$$s(t) = \sum_n a_n \frac{\sin[2\pi B(t - nT)]}{2\pi B(t - nT)} \tag{1.2}$$

Equation 1.2 allows a signal of bandwidth B to be reconstructed by summation of

the pulses centred on the sample values. The reconstructed signal is shown by the bold curve in Figure 1.3. Note that it passes through the sample points without error.

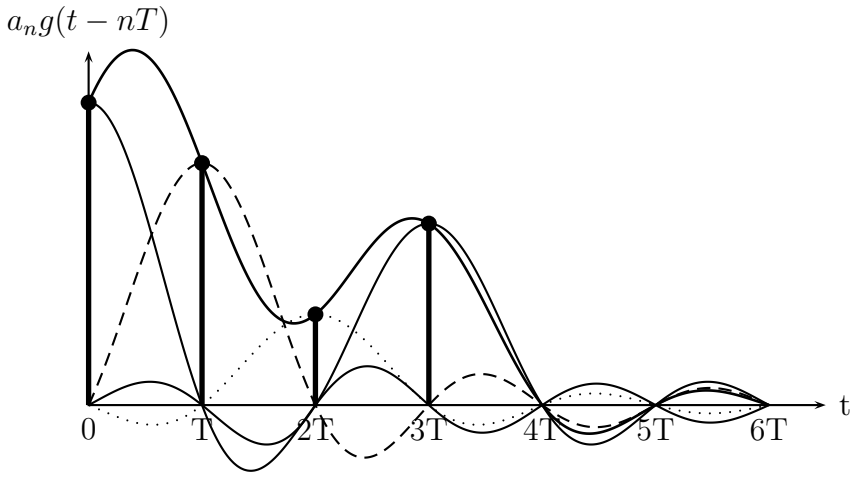

Figure 1.3: Band-limited pulses with no inter-symbol interference (only four pulses shown for clarity)

Hartley's Law

In his publication in 1928[6], Hartley investigates the amount of data that can be transmitted reliably using a number of distinct pulses over a band-limited channel. He argued that the dynamic range of the signal amplitude and the precision of the receiver in detecting amplitude levels limited the maximum number of distinct pulses that could be reliable transmitted over a channel. The number of distinguishable levels, M, is stated as $M = 1 + A/\Delta V$, where A is the signal amplitude voltage (limited by a power constraint) and ΔV is the precision of the detector. Drawing on Nyquist's theory that the maximum pulse rate is twice the channel's bandwidth, $R \leq 2B$, combined with the number of bits per pulse being $\log_2(M)$ capacity can be expressed as in Equation 1.3.

$$C = 2B \log_2(M) \tag{1.3}$$

Shannon Hartley theorem

Shannon's channel capacity theorem [7] is concerned with achieving reliability through error correction coding rather than through distinguishing pulse levels. M in Hartley's formula is specified in terms of signal-to-noise ratio.

$$C = B \log_2 \left(1 + \frac{S}{N}\right) \tag{1.4}$$

It can be seen that Equation 1.4 can readily be obtained by substituting $M = \sqrt{(1 + S/N)}$ in Equation 1.3.

Sampling and Pulse Coded Modulation (PCM)

For digital encoding and transmission of an analogue source it is first necessary to digitise (to make digital) the source. It is possible to transmit very short intervals of a analogue signal with large gaps between the intervals and to be able to reconstruct the signal at the receiving end. The short intervals of the analogue signal are called samples. The only condition is that these samples must be taken at a minimum rate - the **Nyquist rate** - which is twice the highest frequency component of the signal. An application of sampling is in time division multiplexing (TDM) where the gaps can be filled with samples from other source signals and carried on the same wire. TDM is used by telephone networks to simultaneously carry more than one voice channel on just one line.

To be transmitted digitally the samples must undergo a further process called quantisation. This involves numbering the amplitude of the samples with finite precision. For digital speech telephony systems that employ 8-bit PCM 256 different levels are used to represent the sample amplitudes.

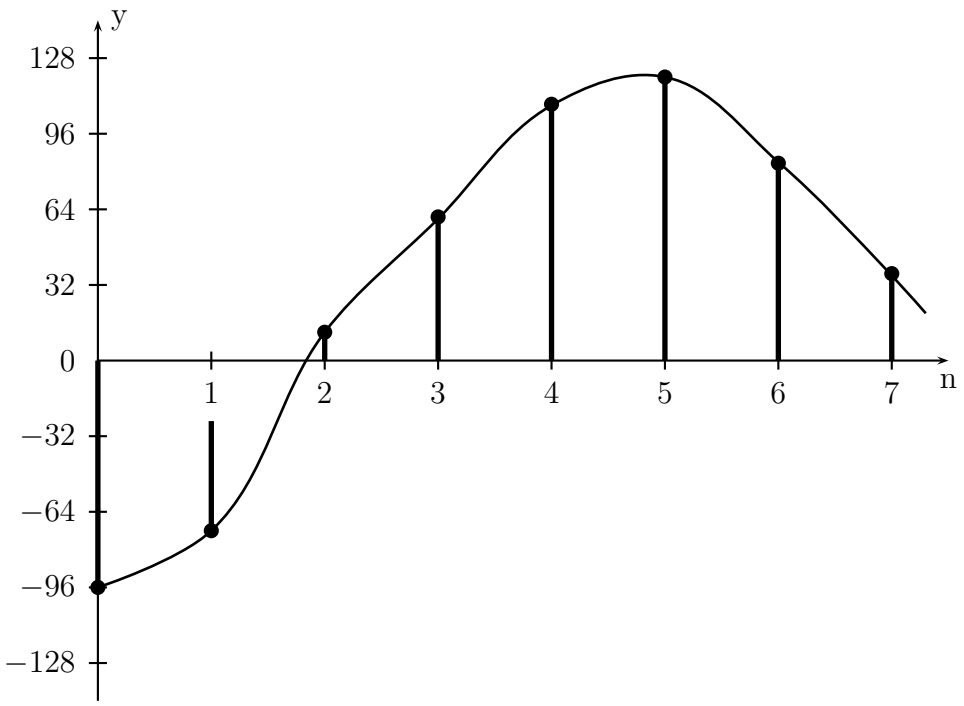

Figure 1.4: PCM modulation showing sampling

Digital Modulation

Each form of analogue modulation has a digital equivalent. The modulating signal, instead of being a continuously varying value, has discrete values. In digital modulation the term 'shift keying' is used in place of the word 'modulation'. AM becomes *amplitude shift keying (ASK)*. The reference to 'keying' comes from the use of a Morse Key used in early telegraph systems.

Analogue	**Digital**	**Extensions**
AM	ASK	
FM	FSK	DTMF
PM	PSK	BPSK,QPSK,8-PSK

Table 1.1: Digital modulation types shown with the analogue equivalent

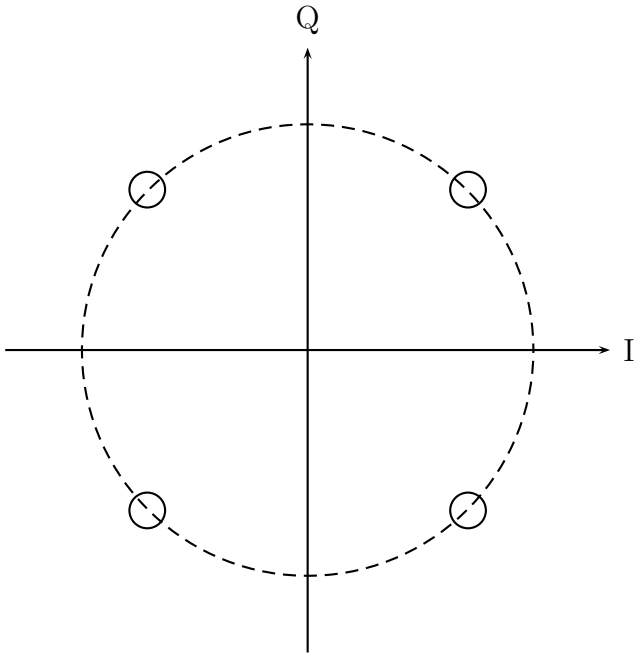

Figure 1.5: QPSK Constellation.

1.2.6 Noise

In early communication systems the human operator would notice noise at the receiver, much like the often heard hiss on an AM radio. The human operator had the job of understanding the information even in the presence of this noise. The brain is remarkably effective at this job of recovering the original message, sometimes even if a whole word is lost. In these early systems the human brain performed the role of *error correction*. Prior to the introduction of FM, AM broadcast radio transmitted music over the air. To overcome the hiss caused by electrical noise high transmission power was used, very much like having to shout to a friend in a noisy place. Transmitting at a higher power increases the *signal to noise ratio*.

1.2.7 Bandwidth

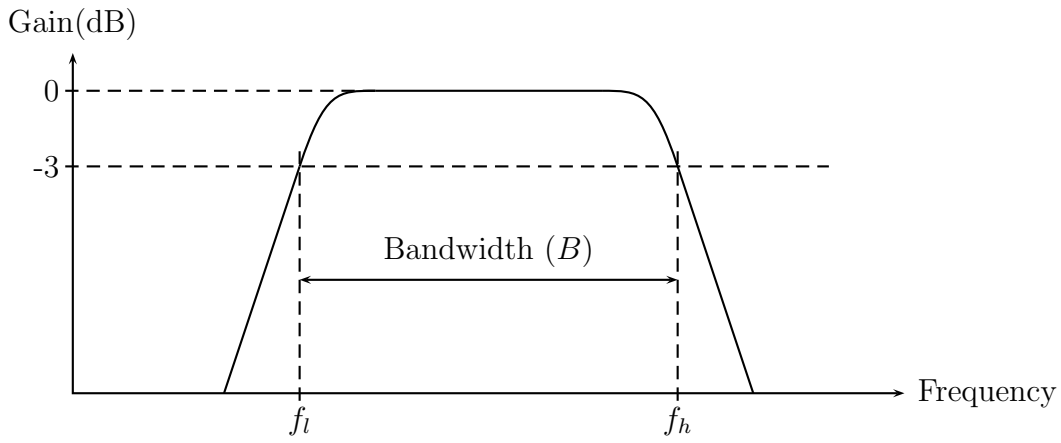

Figure 1.6: 3dB Bandwidth

For a communications system to operate in the presence of other communications systems, each system requires its own frequency band, otherwise interference is going to occur. Often broadcast radio programmes are unidentified by their carrier frequency or wavelength. The user select what programme they want to listen to by tuning their receiver to a particular carrier frequency. A rapidly changing signal will occupy a large

range of frequencies and is said to have a high bandwidth. Broadcast radio programmes are separated so that there is no overlap in their frequency spectra, otherwise interference will result. Each system is given only a finite amount of bandwidth B and this becomes the second fundamental system limitation. A communications channel requires sufficient bandwidth to accommodate the spectrum of the signal. Information sources have varying bandwidth requirements, TV video signal requires a 6MHz bandwidth and voice a much lower bandwidth of 3kHz. For digital signals with symbol rate r, the bandwidth must be $B \geq r/2$.

1.2.8 Coding

Coding forms an essential part of any digital communications system. Noise present in communications systems have the potential to distort any information that is conveyed over them. In most systems it is necessary to devise coding schemes which will allow some tolerance to distortion caused by noise.

Block codes

Codes are constructed from fields with a finite number of elements. These finite fields are called *Galois fields* in honour of Évariste Galois. A Galois field having q elements is referred to as GF(q). A field must have at least two elements; a *zero* element and a *one* element. Thus the GF(2) is the simplest Galois field. Table 1.2 shows that addition in GF(2) can be achieved using modulo-2 arithmetic. Multiplication in GF(2) can also

+	0	1
0	0	1
1	1	0

Table 1.2: Addition over GF(2)

be performed by modulo-2 arithmetic, this is shown in Table 1.3 In general fields can be created with any value of q that is either a prime or an integer power of a prime. Where

$\cdot$	0	1
0	0	0
1	0	1

Table 1.3: Multiplication over GF(2)

q is prime, modulo-q arithmetic can be used for addition and multiplication. If q is a power of a prime, $q = p^m$, then addition and multiplication are performed using modulo-p arithmetic. GF(p^m) is known as the *extension field* of GF(p)[3, pg 418][4, pg 404]. The elements of GF(2) are simply 0 and 1. An extension field is created from a primitive polynomial and the primitive element, α. For example, the elements of GF(2^3) can be generated using the primitive polynomial, $p(x) = x^3 + x + 1$ by substituting x with α such that $p(\alpha) = \alpha^3 + \alpha + 1 = 0$. Adding elements in GF($2^m$) the vector representation allows

Power	Polynomial	Vector
0	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	$1 + \alpha$	011
α^4	$\alpha + \alpha^2$	110
α^5	$1 + \alpha + \alpha^2$	111
α^6	$1 + \alpha^2$	101

Table 1.4: GF(2^3)

the exclusive-or operation available on nearly all processors to be used. Multiplication is achieved using the power representation, multiplication is done by adding the powers modulo $2^m - 1$.

G Matrix

A codeword, $\mathbf{C}$, for a block code, can be generated by multiplication of the information vector, $\mathbf{D}$, and a matrix, $\mathbf{G}$, called the *generator matrix*.

$$\mathbf{C} = \mathbf{G} \cdot \mathbf{D} \tag{1.5}$$

H Matrix

Received codewords can be checked for errors by multiplying them by a matrix called the *parity check matrix*, **H**. Without any errors the result should be the zero matrix, **0**.

$$\mathbf{C} \cdot \mathbf{H}^T = \mathbf{0} \tag{1.6}$$

A non-zero result would indicate that the codeword has errors caused by noise.

Hamming Codes

In 1950 Richard Hamming published his work on the Hamming Code. Hamming was frustrated by the errors produced by the card reader of the computer he was using a Bell labs. He created his code to solve this problem. The (7,4) (the codeword is seven bits and the information has four bits) Hamming code is able to correct an error in four bits of data due to the redundancy of an additional three bits called parity bits.

1.2.9 Convolutional codes

Block codes, discussed earlier, are decoded using hard decision decoders. The use of convolutional coding allows soft decision decoding so that performance approaches the channel capacity [4, pg 491]. In convolutional codes, however, the codeword bits are generated by a different process. Information bits are shifted along a tapped shift register, connected to the taps are a number of modulo-2 adder. This arrangement shown in Figure 1.7 performs convolution on the information and produces the output bit sequence required. The polynomial representation for the convolutional encoder of Figure 1.7 is $1 + x + x^3$.

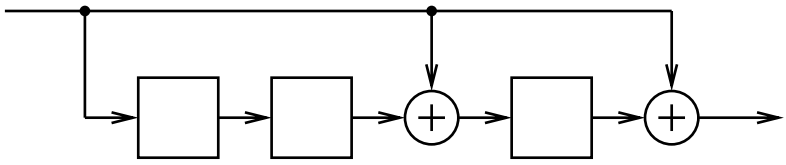

Figure 1.7: An example of a convolutional encoder.

1.2.10 Turbo codes

Turbo codes were introduced in 1993 by Berrou, et. al. [8] and achieve a performance that is very near to the Shannon limit. The name 'turbo' comes about from the method of decoding were two or more (usually limited to two) *maximum a posteriori probability* (MAP) decoders pass extrinsic information between each other and this extrinsic information completes a cycle through all the MAP decoders in each iteration. This cycling of information is likened to a turbo of an internal combustion engine where the pressure of the exhaust gases is used to increase the inlet manifold pressure of the engine. The literature on turbo codes will be discussed in Sections 1.2.20 and 1.2.21.

1.2.11 Parallel concatenation

The structure of a turbo encoder is simply a concatenation of two (sometimes more) convolutional or block encoders. The information bits are passed to both constituent encoders; directly to the first and interleaved before entering the second. Each encoder operates on an input of k bits and each produces a number of parity bits. If each encoder produces one parity bit for each input bit then a total of two parity bits will be generated for each information bit given rise to a rate $\frac{1}{3}$ code. The code rate can be increased by removing some of the bits from the parity output, this is known as *puncturing*. In this work multi-binary, rate $\frac{b}{b+1}$, encoders are used to increase code rate. Multi-binary encoders have two or more convolution polynomials and are an extension of duo-binary

codes.

The diagram in Figure 1.8 shows the structure of a turbo code, a parallel concatenated convolutional code (PCCC), note the parallel nature of the information flow paths into RSC1 and RSC2. The PCCC encodes a length k binary information sequence $\mathbf{i} = (i_0, i_1, \ldots, i_{k-1}) \in [\mathrm{GF}(2)]^k$, where GF(2) represents the binary Galois field. There exists three parallel paths through the PCCC as shown in Figure 1.8. The uppermost path is clearly the information sequence passing through to the output forming the systematic portion of the code. The middle path provides parity, p_1, which is produced by a convolutional encoder. The lower most path is similar to the middle path in that it also produces parity, p_2, using a convolutional encoder. However, the information sequence is interleaved prior to the input to this path's coder. The interleaver produces a permutation of the information sequence that is supplied to the lower encoder, thus the lower encoder produces a parity output that is uncorellated with the parity output of the upper encoder. The output from the three paths are combined to produce the codeword $\mathbf{c} = (\mathbf{i}, \mathbf{p_1}, \mathbf{p_2})$.

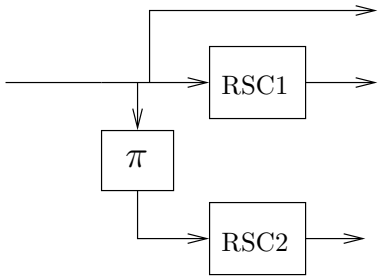

Figure 1.8: Structure of a parallel concatenated convolutional code encoder showing information flow to two parallel branches.

1.2.12 Serial concatenation

Serially concatenated codes described by Benedetto and Montorsi[9] are a serial concatenation of two (sometimes more) constituent coders separated by an interleaver. Again, as

for parallel concatenation, the constituent coders maybe block or convolutional coders. In the case of serial concatenation the k information bits are encoded by the first encoder to produce parity bits. In contrast to parallel concatenation, the parity output from the first encoder is also interleaved to give an intermediate codeword of N bits. This intermediate codeword is then input to the second encoder which produces the final codeword of n bits. Using rate $\frac{1}{2}$ constituent encoders gives rise to a rate $\frac{1}{4}$ code.

1.2.13 Interleaver

In order for the iterative decoder to be effective it is important to reduce as much as possible the cross-correlation of the extrinsic information from one of the MAP decoders to the others. Therefore an interleaver permutes the information sequence entering the second encoder. Due to the interleaving the second encoder will produce a parity stream that is different to that produced by the first encoder. Events that are localised in the first encoder are spread out by the interleaver before being encoded by the second encoder. This reduces the cross-correlation and improves the performance of the iterative decoder. Much of the work presented herein is focused on designing good interleavers that give the best performance for these decoding schemes.

1.2.14 Tailbiting

Tailbiting is used in this work to avoid the necessity of trellis (a form of state transitional diagram that shows the change of state for each time step) termination. A starting state is chosen such that the encoder's state finishes in the same state that it started in. This means that the trellis ends can be connected to form a circular trellis[10].

1.2.15 Both a block code and a convolutional code!

Since the turbo encoder employs an interleaver the information must be 'framed' so that it can be permuted as required. This framing means that the turbo code can be thought of as a block code. As already stated above it can be a concatenation of two convolutional codes. Thus, it fits into both block code and convolutional code classifications. In Chapter 6.1 the turbo code is treated as a block code and decoded using a form of list decoder. In this work the generator matrix is used to derive the decoding equations.

1.2.16 Decoding

Since turbo codes have two (or more) constituent encoders it lends itself to being decoded iteratively. The likelihood of the individual bits from the decoding of the first constituent decoder is, after interleaving, fed into the second decoder. Thus it is important that the decoding algorithms used provide likelihood information for each individual bit. As the Viterbi algorithm makes a maximum-likelihood decision on a *sequence* of bits and does not provide individual bit likelihood it is not suitable for use in the iterative decoding of a turbo code. The soft-output Viterbi algorithm can be used, with less complexity than the MAP decoder, but has degraded performance[11, pg 642]. Whatever algorithm is chosen the likelihoods computed by the first decoder are passed to the second decoder. Then the second decoder produces bit likelihoods that are passed, after de-interleaving, back to the first decoder. This process is repeated for a number of iterations until the decoding converges.

MAP Decoding Using the BCJR Algorithm

A commonly used MAP algorithm is the method developed by *Bahl, Cocke, Jelinek and Raviv (BCJR)*[12] from now on referred to as the BCJR algorithm. The BCJR algorithm provides probabilities for the individual bits that are used in the iterative decoding of

turbo codes. From our discussion on convolutional codes we know that a trellis diagram can be used to represent the state transitions with respect to time. Here it is used to illustrate the operation of the BCJR algorithm.

1.2.17 Why error correction?

The signal-to-noise ratio (SNR) of a communications channel determines the number of transmission errors that occur in a digital communications system. For a system which has a fixed SNR an alternative means must be found to increase the reliability in order to reduce the number of transmission errors. Error correction is most often the best solution. Using error correction it is possible to achieve a lower bit error rate for the same SNR and hence communications becomes that much more reliable.

1.2.18 Coding gain

An alternative method of reaching the required bit error rate is to increase the transmission power, directly increasing the SNR. So to achieve a certain bit error rate either the transmission power can be increased or an error correcting scheme can be introduced. The utilisation of the error correcting solution as opposed to the increased power solution leads to the concept of 'coding gain'. Proakis[3, pg 442][4, pg 426] defines *coding gain* as the gain of the coded system compared to the uncoded system.

1.2.19 Before Turbo Codes

Prior to the introduction of turbo codes[8] many error correcting coding techniques had been developed. The performance of these codes was far removed from the theoretical limits described by Shannon[7, 13]. These error correcting codes could be classified as either *block codes* or *convolutional codes*. As you will read later a turbo code can partially fit into both categories.

Berrou introduced turbo codes in 1993 [8] and are reviewed in [14, 15, 16, 17, 6, 18].

1.2.20 Interleaver design

It is known that the choice of interleaver has a significant influence on the performance of a turbo code[19, 20, 21]. The work in [21], although originally applied to trellis terminated codes, can be adapted and applied to tailbiting schemes.

The diagram in Figure 1.8 shows the structure of a turbo code, a parallel concatenated convolutional code (PCCC). The PCCC encodes a length k binary information sequence $\mathbf{i} = (i_0, i_1, \ldots, i_{k-1}) \in [\mathrm{GF}(2)]^k$, where GF(2) represents the binary Galois field. There exists three parallel paths through the PCCC as shown in Figure 1.8. The uppermost path is clearly the information sequence passing through to the output forming the systematic portion of the code. The middle path provides parity, p_1, which is produced by a recursive convolutional encoder. The lower most path is similar to the middle path in that it also produces parity, p_2, using a recursive convolutional encoder. However, the information sequence is interleaved prior to the input to this path's coder. The interleaver produces a permutation of the information sequence. The output from the three paths are combined to produce the codeword $\mathbf{c} = (\mathbf{i}, \mathbf{p_1}, \mathbf{p_2})$. The permutations can be described by a permutation vector $\pi = (\pi_0, \pi_1, \ldots, \pi_{k-1})$ applied to the information sequence to give the permuted information sequence $i'_j = i_{\pi_j}$, $j = 0, 1, \ldots, k-1$.

The interleaver permutes the information sequence so that when two consecutive non-zero symbols are presented to the upper RSC they are spread out by the interleaver to ensure that they are not consecutive in the second RSC. The interleaver ensures that the non-zero symbols are spread out for at least one of the RSCs. For all possible sequences of two or more non-zero symbols it is possible to find the minimum spread. If an interleaver is entirely random then it is quite likely that this minimum spread will be very small and will have an undesirable effect on the weight spectrum and hence performance of the overall code. It is known that structured interleavers can be developed with good

spreading capabilities[22]. However, it has been found that these interleavers produce very high multiplicities in the terms close to the d_{min}term of the weight spectrum, this leads to poor performance[22]. The union bound frame error rate is proportional to the multiplicities.

1.2.21 Weight Spectrum Analysis

Turbo codes have excellent performance at low signal-to-noise ratios due to the relatively low number of low weight codewords. Using primitive feedback polynomials and large interleavers performance is improved through spectral thinning. However, the error floor occurs due to the low free distance of the turbo code[15, 23, 24]. Later in the chapter the union bound will be introduced. The union bound can be calculated from the weight spectrum and used to predict the asymptotic performance. The results from many simulations show that error floor performance approaches the union bound at moderate signal-to-noise ratios. The introduction of the interleaver makes weight spectrum computation much more complex. This is because shifted information patterns when interleaved can create different patterns which then lead to codewords with different weight. All shifts of a pattern have to be evaluated.

1.2.22 Techniques

Existing Weight Spectrum Evaluation procedures

Weight-two Input Sequence Algorithm

This procedure involves an information input sequence generator that generates all the $\binom{k}{2}$ possible weight-two information sequences. These are then encoded by the turbo encoder (PCCC) to produce codewords and the weights of these are evaluated. It is assumed in this method that the lowest Hamming weight $d_{min}^{(2)}$ codewords produced will give the d_{min} of the code. In a paper by Garello et al[25], it is proven that this method does not

always produce codewords that have a Hamming weight equal to the minimum distance, $d_{min}^{(2)} \geq d_{min}$. This procedure for producing d_{min} codewords becomes even less reliable for short interleaver lengths ($k < 1000$, for example). When this method was tried with codes having interleaver length $k = 676$ the results showed that many d_{min} and other low weight codewords were not found. Simulation of these codes showed that often a d_{min} codeword can be produced from a weight four information sequence. This procedure has to be discarded as extending this procedure to weight-four produces an unreasonable computational complexity. Further analysis of the importance of information sequences of weight greater than two can be found in[26].

Error Event Algorithm

A initial limit is set on the maximum weight of codeword to be produced, d^*. This should be greater than the expected minimum distance, $d^* > d_{min}$. The set S of all information sequences that force the RSC to leave the zero state at time zero and produce a combined weight (information plus parity from the RSC) less than d^* are turbo coded and their codeword weights evaluated. Is is important to consider not only error events (sequences forcing the RSC to leave state zero and return to state zero) but also the concatenation of these error events. This procedure must be repeated with all shifted versions of the information sequences in set S along the interleaver length k. Garello et al[25] state that this method works well for codes with small minimum distance ($d_{min} < 10$). The computational complexity becomes prohibitive with larger d_{min}.

In this work there is a requirement to analyse turbo codes with duo-binary[27] and multi-binary[28] constituent coders. This gives rise to a much larger number of possible error events which is due to the increase in the number of possible state transitions. In a binary RSC there are only two possible exit paths from any given state. For a duo-binary there are four possible exit paths from any given state. And for the multi-binary RSC with rate $b/(b+1)$ there are 2^b possible exit paths from any given state. The effect of

this is that for duo-binary and multi-binary turbo codes the complexity of the error event method becomes prohibitive.

1.2.23 Current Standards Utilising Turbo Codes

Turbo codes have been deployed in many emerging standards. The most significant of these being Digital Video Broadcast (DVB)[29], 3G and 4G LTE mobile phone standards[30] and WirelessMAN, a metropolitan area network[31].

1.3 Thesis Aims

The overall aim of this work is to advance the field of turbo codes. This work presents many advances made to the field, backed up by the filing of a patent and paper being presented at ISIT2004. The objectives will be summarised below.

1.3.1 Objectives

Improvements to a turbo code based communications system are to be made by investigating improvements to both the encoder and the decoder. Below the objectives are listed for both the encoder and decoder.

Objectives for the Improvement of the Encoder

- Develop new interleaver design techniques.
- Produce new structures for the constituent RSC encoders.
- Investigate the advantages of outer codes.

Objectives for the Improvement of the Decoder

- Investigate the possibility of combining decoding techniques into a hybrid decoder.

- Can an outer code provide information back to the inner (turbo-MAP) decoder?
- Understand better the turbo-MAP non-convergence behaviour.

2 Turbo Code as a Block Code

2.1 Introduction

It is generally recognised that cycles of length four in LDPC codes (and any other type of code if using MAP decoding) limit convergence[32]. There are well performing designs of LDPC codes which minimise cycles of length six and eight. An example of such codes are the Progressive Edge Growth (PEG) LDPC codes[33], so called because they are formed by randomly adding edges to the Tanner graph ensuring that short cycles are not produced. The idea being explored in this section is whether turbo codes designed to have a reduced number of cycles are able to achieve higher performance.

2.2 Definition of a 'Cycle'

The rows of the H matrix correspond to parity check equations for each parity bit. This is clearly seen when the H matrix is in its reduced echelon form(having an identity matrix in it's left portion). There are $n-k$ parity check equations corresponding to the $n-k$ rows of the H matrix. Reduced echelon form can be obtained from linear combinations of the rows of the H matrix, coupled with column swapping as necessary to form an identity matrix for the parity bits. How to obtain a reduced echelon form is discussed later. Parity check equations are derived from the H matrix by representing the codeword as a vector, $\mathbf{v} = \{d_0, d_1, \ldots, d_{k-1}, p_0, p_1, \ldots, p_{n-k-1}\}$ and multiplying this by $\mathbf{H}$ to give zero syndrome as in equation 2.1.

$$\mathbf{H} \cdot \mathbf{v}^{\mathbf{T}} = \mathbf{0} \tag{2.1}$$

In belief propagation[34] each parity check equation is used to transmit extrinsic information to each bit in the equation and to receive extrinsic information from all other equations involving that bit. With reference to Figure 2.1, cycles of length four occur when the same two bits occur in any two parity check equations. For a cycle of length six to occur three parity check equations and three bits must be involved. Such cycle occurs when for example, i_2 and i_4 belong to the first parity check equation, i_4 and i_5 belong to the second and the third has i_2 and i_5. Similarly a cycle of length eight must involve four parity check equations and four bits. Example cycles in the H matrix are clearly illustrated in Figure 2.1 for cycles of length four, six and eight.

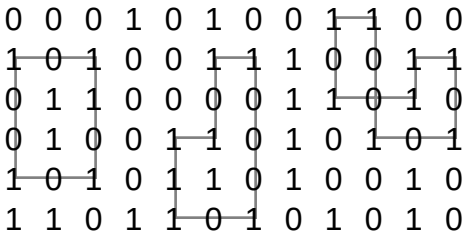

Figure 2.1: Cycles of length 4, 6 and 8.

From Figure 2.1 is can be seen that the cycle of length four has four line segments or corners, cycles of length six have six segments and cycles of length eight have eight segments. These cycles can also be shown using a Tanner graph[1]. Figure 2.2 shows a partial Tanner graph for the H matrix in Figure 2.1, some nodes and edges are not shown for clarity. The dotted lines show a cycle of length four and the dashed line show a cycle of length six.

2.3 Derivation of H Matrix of a Turbo Code

The H matrix of a parallel concatenated convolutional code (PCCC turbo code) is formed from the joining of the H matrices of the constituent coders. The H matrix can be considered as a list of parity check equations. For a parallel concatenation of two coders

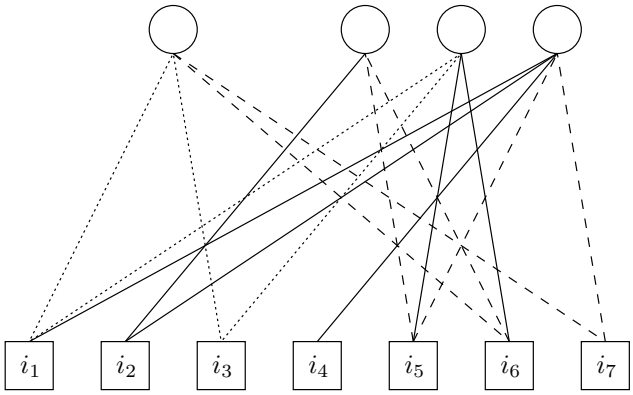

Figure 2.2: Tanner graph[1] representation; cycle length 4 (dotted) and 6 (dashed).

the parity from both coders is used to form the codeword and so the parity check equations associated with both coders can be combined into a H matrix that describes the turbo code. When producing the turbo code's H matrix the effect of the interleaver must be considered for the parity check equations associated with any constituent coder that has interleaved input. The H matrix will be derived for PCCC having two recursive systematic coders (RSCs), where the input to the second RSC is interleaved as shown in figure 1.8. For a recap on the structure of the encoder refer back to Section 1.2.11.

2.3.1 H Matrix of the Recursive Systematic Coder (RSC)

The input, output and coefficients of the RSC can be described in polynomial form. The input, $d(x)$, is convolved with the feed-forward polynomial, $b(x)$. The result is then deconvolved by the feedback polynomial, $a(x)$, to give the output, $p(x)$. The polynomials for the RSC shown in figure 2.3 are $b(x) = 1 + x^2$ and $a(x) = 1 + x + x^2$. We will continue to use these polynomials to illustrate this procedure by way of example.

$$p(x) = \frac{d(x) \cdot b(x)}{a(x)} \tag{2.2}$$

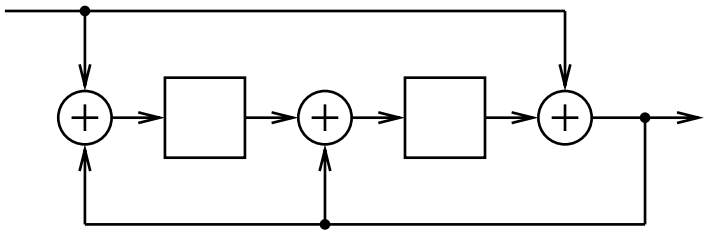

Figure 2.3: An example of a constituent encoder.

$$a(x) \cdot p(x) = b(x) \cdot d(x) \tag{2.3}$$

$$a(x) \cdot p(x) + b(x) \cdot d(x) = 0 \tag{2.4}$$

$$\begin{aligned}
0 &= a(x) \cdot p(x) + b(x) \cdot d(x) \\
&= \left[1 + x^2\right] \cdot \left[d_0 + d_1 x + d_2 x^2 + d_3 x^3 + d_4 x^4 + d_5 x^5 + d_6 x^6\right] \\
&+ \left[1 + x + x^2\right] \cdot \left[p_0 + p_1 x + p_2 x^2 + p_3 x^3 + p_4 x^4 + p_5 x^5 + p_6 x^6\right] \qquad (2.5) \\
&= \left(d_0 + d_5 + p_0 + p_6 + p_5\right) x^0 \\
&+ \left(d_1 + d_6 + p_1 + p_0 + p_6\right) x^1 \\
&+ \left(d_2 + d_0 + p_2 + p_1 + p_0\right) x^2 \\
&+ \left(d_3 + d_1 + p_3 + p_2 + p_1\right) x^3 \\
&+ \left(d_4 + d_2 + p_4 + p_3 + p_2\right) x^4 \\
&+ \left(d_5 + d_3 + p_5 + p_4 + p_3\right) x^5 \\
&+ \left(d_6 + d_4 + p_6 + p_5 + p_4\right) x^6 \qquad (2.6)
\end{aligned}$$

In this example a tailbiting scheme is used, the information length, k, is chosen to be 7. The information length is kept small so that the follow equations can be easily written. Information length must not be a factor of three as this is the period of the RSC and tailbiting would not be possible. We can define $x^{i+7} = x^i$ for number of information bits,

$k = 7$. From equation 2.6 terms in $x^0, x^1, x^2, \ldots, x^6$ are collected to produce the seven parity check equations required for the H matrix.

$$\begin{aligned} d_0 + d_5 + p_0 + p_6 + p_5 &= 0 \\ d_1 + d_6 + p_1 + p_0 + p_6 &= 0 \\ d_2 + d_0 + p_2 + p_1 + p_0 &= 0 \\ d_3 + d_1 + p_3 + p_2 + p_1 &= 0 \\ d_4 + d_2 + p_4 + p_3 + p_2 &= 0 \\ d_5 + d_3 + p_5 + p_4 + p_3 &= 0 \\ d_6 + d_4 + p_6 + p_5 + p_4 &= 0 \end{aligned} \tag{2.7}$$

$d(x)$ and $p(x)$ are combined systematically to produce a codeword $v(x) = d(x) + p(x) \cdot x^{n-k}$. $v(x)$ can be represented as a vector $\mathbf{v} = \{d_0, d_1, d_2, d_3, d_4, d_5, d_6, p_0, p_1, p_2, p_3, p_4, p_5, p_6\}$, then a matrix can be found that when multiplied by $\mathbf{v}$ gives a zero vector and reproduces the parity check equations 2.7.

$$\mathbf{H} \cdot \mathbf{v^T} = \mathbf{0} \tag{2.8}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.9}$$

It turns out that each row in the H matrix is a cyclic shift of its preceding row. The H matrix is easily determined from the coefficients of the RSC polynomials.

2.3.2 Parallel Concatenation of two RSCs

The turbo code in this example is a parallel concatenation of two RSCs. The information supplied to the lower RSC is first interleaved. For this example let the interleaver be described by the permutation, $\Pi = \{4, 6, 1, 3, 5, 0, 2\}$, and both RSCs have the same polynomials as in figure 2.3. The H matrix for the lower RSC can be derived from the H Matrix of the upper RSC by permuting the information columns according to the

interleaver description.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.10}$$

For the lower RSC let the parity be called $q(x)$ so that the codeword vector for the turbo code is $\mathbf{v} = \{d_0, d_1, d_2, d_3, d_4, d_5, d_6, p_0, p_1, p_2, p_3, p_4, p_5, p_6, q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$. For the complete turbo code then $\mathbf{H_{turbo}} \cdot \mathbf{v^T} = \mathbf{0}$ and the H matrix for the turbo code is simply

a combination of the H matrices of both the RSCs.

$$\mathbf{H} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{bmatrix} \tag{2.11}$$

2.4 Analysis of Cycles from H Matrix

An investigation was carried out to count the number of cycles in the H matrix of a typical turbo Code. An algorithm was developed so that this search could be carried out with the use of a computer. Prior to the development of a procedure to derive the H matrix of the Turbo code, this algorithm was applied to the H matrix of a low density parity check (LDPC) code. It was later applied to the H matrix of a turbo code.

2.4.1 Algorithm for searching for cycles in a H Matrix

The search for cycles in based on a tree search method. First a search is made in each row of the H matrix until a 1 is found and its column is recorded as the starting column.

When a 1 is found its row is searched for another 1, if found its column is searched for another 1. For this 1 its row is searched for another 1 and if this 1 is in the same column as the starting column then a cycle of length four has been found. Row and column searches are made alternatively at each level of the tree search. With reference to Figure 2.1, when searching for cycles of length greater than four it is important to check that no more than one horizontal line occurs in any row, and that no more than one vertical line occurs in any column, otherwise the effect is of a shorter cycle.

2.4.2 Cycles in the H Matrix of a Turbo Code

For a convolutional code that is decoded using a MAP decoder the short cycles may not become an issue since the MAP decoder is not iterative. However, in a turbo decoder consisting of two convolutional coders, the cycles between the constituent coders becomes important. This is due to the iterative nature of the turbo decoder. The cycles that need investigating are those which cross from the upper half of the H matrix to the lower half. Remember that the upper half of the H matrix forms the parity check equations for the first constituent RSC and the lower half forms the parity check equations for the second constituent RSC. This requires a modification to the algorithm that was originally applied to the H matrix of the LDPC code. After cycles are detected in the H matrix they are analysed to check that each row in the cycle comes from alternating halves of the H matrix, ie. cycles between the constituent RSC coders. Cycles that fall entirely with one half of the H matrix are not a problem since these are localised to just one constituent RSC whose MAP decoder is immune to such cycles. Since the only cycles counted are those which cross the boundary between the constituent RSCs, these do not involve parity bits because of the structure of the matrix. Each constituent RSC produces its own set of parity equations.

2.4.3 Cycle Counting for the (1014,676,13) Turbo Codes

Later in Section 4.1.3 an interleaver modification algorithm will be described that improves the performance of the turbo code by swapping entries of the interleaver. The interleaver modification algorithm provides optimisation of the weight spectrum by retaining those swaps which reduce the number of low weight codewords. During the development of the interleaver modification algorithm many interleavers were produced for the (1014,676) turbo code. These interleavers vary in terms of the level of optimisation and lend themselves to this investigation. Table 2.1 lists these interleavers showing the number of d_{min} codewords (multiplicity) and number of cycles having a length of four. It is very interesting to observe that interleavers that were more heavily optimised had

$\mathbf{d_{min}}$	**Multiplicity**	**Cycles of length 4**
13	71	8
13	6	11
13	5	13

Table 2.1: Counting cycles of length 4 for interleavers of increasing optimisation.

more cycles of length four. It has now become apparent that the number of length four cycles does not indicate how well a turbo code will perform. Some information weight two sequences are capable of producing a very short parity burst as the RSC goes from the zero state on the first information bit and then gets reset to the zero state on the second information bit. The problem is when these information weight two sequences are interleaved such that both RSCs produce very short parity bursts. It was originally thought that since the optimisation process removed these cycles, that the cycles count would decrease. However, it is probably the case that these cycles have their bit indexes interleaved differently such that both RSCs do not produce short parity bursts simultaneously. The optimisation produces swaps adjacent entries in the interleaver so it is likely that cycles remain due to this localised swapping.

2.4.4 The connection between interleaver spread and length four cycles

The S-Interleaver design algorithm guarantees to spread information weight two sequences, whose 1 bits are less than S bits apart, so that the information bits are spaced by at least R bits apart after interleaving. There is a spread associated with the taps of the constituent RSCs and this spread can be seen from the H matrix of the turbo code. The indexes of 1s from the fourth row of the H matrix of the (1014,676) turbo code are $\{0, 1, 3, 339, 340, 341, 342, 345, 346, 348, 349, 350, 351, 352, 353\}$. The indexes up to and including 337 are the 338 parity bits. The information indexes range from 338 to 1013. It can be seen that the range of spread for the information taps of the RSC is $S_t = 353 - 339 = 14$. Therefore it is necessary to produce an interleaver with a spread, $S > S_t$. The minimum value required for S in this case is $S = 15$ in order to guarantee no cycles of length four exist. In order to confirm this theory a number of interleavers

Spread	**Cycles of length 4**
10	228
12	146
14	36
15	0

Table 2.2: Counting cycles of length 4 for interleavers of increasing optimisation.

were produced with varying spread and each one was analysed for cycles of length four. The results shown in Table 2.2 show that a design spread of 15 was required in order to eliminate cycles of length four. However, it is conceivable that an interleaver with a spread of less than 15 could be designed that would not generate cycles of length four. For a cycle to exist all its vertices must fall upon a non-zero tap of the RSC.

2.5 Conclusions

- The number of length four cycles is not the only factor indicating the performance of a turbo code, since only a small percentage of information weight two sequences cause a short parity burst from the RSCs.

- Cycles of length four cannot occur when the spread of an interleaver is greater than the tap spread of the constituent RSC coders.

- It is possible that an interleaver with a spread less that the RSC coder tap spread would not lead to cycles of length four. A cycle is only produced when all of its vertices fall upon a non-zero tap of the RSC.

3 Interleaver design

The best interleavers are those which compromise spread with randomness. Structured interleavers can guarantee to spread sequences with two non-zero symbols. A problem arises when there are four or more non-zero symbols in a sequence. Randomness is introduced by swapping elements of the permutation vector whilst retaining the minimum required spread. Research into interleaver design based upon the iterative decoder showed little improvement in performance [35, 36, 14]. Iterative decoder convergence problems are addressed by using lower order feedback polynomials in the constituent recursive convolutional encoders (Section 4.3.2).

Described in [28] is a method of producing an interleaver based upon the RSC structure. There are input-weight-2 sequences that can produce a low parity weight when the second input bit causes the state of the RSC to become zero. For the turbo coding scheme to produce a high parity weight then these sequences must be interleaved such that for the second RSC a high parity weight is produced. This is achieved by either generating an interleaver that spreads these bits or by ensuring that the interleaved sequence does not terminate the second RSC. Sequences that terminate the RSC are carefully analysed and are broken by the interleaver so that the second RSC can produce a higher weight [21] [37].

3.1 Structured interleaver design - arithmetic sequence

An interleaver is designed such that a group of R neighbouring elements have a minimum difference S between each other. The interleaver is designed by producing an arithmetic series. The constant difference, S_t between two consecutive elements in the series is chosen to be in the region of $\sqrt{k}$, where k is the information length of an (n, k) code. Then the value of the element in the i^{th} position, E_i, is given by:

$$E_i = S_t i \mod k \tag{3.1}$$

This method works particularly well when S_t is a factor of $k-1$. For an illustrative case $k = 16$ is chosen. The factors of $k-1$ are 1,3,5 and 15. $S_t = 5$ is chosen since this is closest to $\sqrt{k}$.

$$\begin{aligned} E_i &= 5i \mod 16 \\ &= \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75\} \mod 16 \\ &= \{0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12, 1, 6, 11\} \end{aligned}$$

In this case the values of S_t and R are optimal. The product $S \times R$ is $k-1$, the maximum possible theoretical value[21]. For any case where $\sqrt{k}$ is an integer a difference of $\sqrt{k} \pm 1$ can be chosen. This can be shown as follows:

$$\left(\sqrt{k}+1\right)\left(\sqrt{k}-1\right) \tag{3.2}$$

$$= k + \sqrt{k} - \sqrt{k} - 1 \tag{3.3}$$

$$= k - 1 \tag{3.4}$$

It is important to note that S_t or R should not be defined as being equal to $\sqrt{k} \pm 1$ as this is only the case for values of k where $\sqrt{k}$ is an integer. It is important to use a difference that does not share a common divisor with k, otherwise the above algorithm will repeat the numbers after the first $k/\gcd(k, S_t)$ entries. A computer program has been written that can easily generate these interleavers. The computer program can overcome the problem with repeated entries for cases where $k/\gcd(k, S_t)$ is greater than 1. This is achieved by checking to see if each new value is equal to the starting value, if it is then the new value is decremented and the algorithm continues. However, this reduces the S_t value to $S_t - 1$.

3.2 S-Random interleaver design

Structured interleavers are limited to spreading just information-weight-two sequences. This is because when spreading an information-weight-four sequence, although could be broken into information-weight-two sequences, the interleaved pairs may combine or overlap and be less than the required distance R apart. Random swapping of interleaver elements has proven to be a popular partial solution to this problem[21]. In this work swapping is performed, by a computer, to ensure that if $|i - j| < R$ then $|E_i - E_j| \geq S$.

3.3 Conclusions

- Structured interleavers are simple to produce and can guarantee the maximum possible spread of information-weight-two sequences.
- Structured interleavers produce high multiplicities in the weight spectrum, particularly near the d_{min}terms. Structure interleaver design methods to not always effectively spread information-weight-four sequences.
- No guarantee is given that cycles created by information weight four (and above)

sequences can be broken. However, the S-Random design discussed in Section 3.2 offers a partial solution.

- Random swapping helps break information weight four (and above) sequences.
- Provides a good starting point for the interleaver optimisation algorithm.

4 Weight Spectrum Analysis

4.1 Techniques

4.1.1 Weight Spectrum Evaluation of Turbo Codes Based On Two Identical Recursive Systematic Coders

Using Two Identical Recursive Systematic Coders the Computational Complexity is Reduced

All input sequences must be coded such that the codeword produced has a weight greater than, or equal to, the required minimum distance of the codebook. This means that any input sequence that has a low information weight must produce a codeword that has a high parity weight. For example, if a $d_{min} = 12$ is required and the input sequence has a weight of four, the required minimum parity weight is $12 - 4 = 8$. The parity bits are produced by both RSCs. A certain sequence, S, produces a parity sequence, P_u, at the output of the upper RSC. S is interleaved to produce S_i which the lower RSC uses to generate a different parity sequence, P_l. Each sequence generated is both interleaved and de-interleaved. For the purposes of weight spectrum evaluation only codewords that have a weight less than some target weight require evaluation. The sequences generated to give a low parity weight of the upper RSC will match interleaved sequences that produce a low parity weight in the lower RSC since both RSCs are identical. If a target codeword weight is set, then all sequences that produce a codeword weight less than this target must be found. A hypothetical sequence, $S = \{1, 4, 5, 8\}$, which has an information weight of four, might produce an upper RSC parity weight of five and a lower RSC parity

weight of three to give a codeword weight of twelve. It at first appears that sequences must be selected to produce the full parity weight in the upper RSC in case the lower RSC produces a parity weight of zero. The lower RSC gave a lower parity weight than the upper RSC because S is interleaved to give S_i prior to coding by the lower RSC. We could have chosen to generate the interleaved sequence S_i and to de-interleave S_i to give S. Since S_i, when coded, gives a parity weight of three then S_i also has to be evaluated and hence the sequence generator generates both S and S_i. If the sequence generator only produces sequences that when coded give up to half of the required parity weight, four in this case, then S would not be generated and S_i would have been generated. S can now be produced by inverse interleaving S_i. This can be extended to all the other possible sequences and the sequence generator is only required to generate sequences that produce up to half the target parity weight.

Algorithm 1 Generate sequence

Ensure: valid codeword
  $t \Leftarrow 0$
  state, s at time, t, $s_t \Leftarrow 0$
  while $t \geq 0, W_p \leq W_{p_{max}}$ **do**
    $d \Leftarrow 0$
    $s \Leftarrow rsc(s, d)$
    $p \Leftarrow par(s, d)$
    if parity, $p = 1$ **then**
      $W_p \Leftarrow W_p + 1$
    end if
    if $s = 0$ **then**
      encode
      update weight spectrum
      $t \Leftarrow t - 1$
    else
      $t \Leftarrow t + 1$
    end if
  end while

Sequence Generation

While it is not practical to produce a trellis for the turbo coder, it is possible to draw a trellis for just one of the RSCs. For the purposes of sequence generation only one RSC is considered. The sequences are generated by tracing paths through the trellis and accumulating parity and information weight. All possible paths that produce a combined weight which is less than some target are generated. The combined weight is the sum of the information weight and twice the parity weight as mentioned previously.

Algorithm 2 Interleaver

for $i = 0$ to k **do**
  $\mathbf{Z_i} \Leftarrow \mathbf{I_{\Pi_i}}$
end for

Interleaved Sequence Evaluation

Each sequence is interleaved (or inverse interleaved) before entering the second RSC. The interleaving process is a reasonably straight forward process, so it is implemented as a lookup table. Algorithm 2 shows how interleaving is performed and is the standard method used in the literature, it is shown here for convenience. A code that has a block length of 951 has 951 entries in the table. The value of each entry is unique and determined initially by an S-Random interleaver design algorithm described is Section 3.2. All entries have values ranging from 0 to 950 for a code having a block length of 951. The interleaved sequence is produced from the values of the lookup table indexed by the input sequence.

De-interleaving is the process by which the interleaved sequence is processed to give the input sequence. Another lookup table is used for de-interleaving. Table 4.1 shows an interleaver and the corresponding de-interleaver table for a fictitious code length of ten.

i	$v(i)$
0	9
1	2
2	5
3	8
4	1
5	4
6	7
7	3
8	0
9	6

(a)

i	$v_i(i)$
0	8
1	4
2	1
3	7
4	5
5	2
6	9
7	6
8	3
9	0

(b)

Table 4.1: (a) interleaver lookup table. (b) inverse interleaver lookup table.

4.1.2 Computer program details: Weight spectrum analysis

A program was written that computes the weight spectrum of Parallel Concatenated Convolutional Coders (PCCCs). The encoder is defined by its polynomials and its interleaver.

The minimum hamming distance, d_{min}, of a code is a key metric in the code design. The weight spectrum of a few terms starting with the d_{min} term also can be used to estimate the code performance at the region of the error floor. The program described below calculates the first few terms, including the d_{min} term, of the weight spectrum. The program was extended by the addition of an optimisation algorithm that significantly increases the d_{min}. Previously this was done manually but the complexity made achieving a high d_{min} an impractical task. A high performance PC can now be used to achieve these higher d_{min} values.

Sequence Generation

A fast algorithm has been developed which is based on factoring the overall codeword weight into partitions which may be computed separately. Firstly the maximum codeword weight required from the weight spectrum analysis is defined. This in turn defines

the maximum parity bit weight of interest, $w_{p(target)}$, since the information weight is given by the number of information bits in each codeword. All possible information sequences are generated up to a maximum user defined information weight using the tree search. The sequences are then interleaved and the corresponding parity output sequences are evaluated and punctured. We retain those sequences output from RSC1 with parity weight greater than $\frac{w_{p(target)}}{2}$. The procedure is repeated starting with RSC2 and the corresponding RSC1 parity sequences are then evaluated by inverse interleaving the information sequences input to RSC2. The algorithm is as follows:

1. Generate next information sequence using tree search, input to RSC1, if there are no more sequences goto 7
2. Calculate CRC and append to information sequence
3. Input to RSC1, evaluate parity sequence weight
4. Interleave sequence, input to RSC2, evaluate parity sequence weight
5. Record total weights for each codeword
6. Goto 1
7. Generate next information sequence using tree search, input to RSC2, puncture output
8. Inverse interleave information sequence input to RSC2
9. Calculate CRC and append to information sequence
10. Calculate RSC1 parity weight
11. Interleave sequence and calculate RSC2 parity weight
12. Record total weights for each codeword

13. If there are more sequences goto 7

14. Save weight spectrum to file and exit

Tail-biting

In order to correctly produce a codeword it is necessary to ensure that the state of the constituent RSCs returns to its initial state. A traditional approach is to append additional bits to force the state to zero. This has the disadvantage of increasing the codeword size. An alternative solution is to choose an initial state such that the final state is the same as the initial state. This is normally achieved by first encoding the information sequence using an initial zero state, after the codeword has been produced the resulting state of the RSC is used to index a lookup table. The entries of the lookup table give the initial state required so that when the sequence is encoded a second time the resultant state is the same as the initial state. This approach means that each codeword is encoded twice. For the purposes of weight spectrum computation a much faster method was developed.

The "Biggest Gap" Approach

A low weight codeword has its information bits clustered into short subsequences of at least two bits. For a low weight codeword to be produced the state between these subsequences must return to zero. If there is a large separation between two non-zero information bits and the state has not returned to zero after encoding the first bit the codeword produced will have a high weight. So it sufficient to find the largest gap between the non-zero bits and set an initial state of zero and encode from this gap. The encoding must continue past he the end of the sequence by wrapping around until the gap is reached. If the state is non-zero at this point then the codeword has a high weight and is beyond our range of interest.

Weight Evaluation

The sequences that are produced by the sequence generator are shifted into the RSCs so that the parity weight can be obtained. The parity weight is produced from the output of two RSCs. Since one RSC has been used in the sequence generation above it is only necessary to shift the sequence into the other RSC. The overall codeword weight is the sum of the sequence weight, the upper RSC parity weight and the lower RSC parity weight. Codewords that have a weight less than some user defined value are recorded and a weight spectrum is produced.

CRC and Parity Puncturing

To obtain space for additional CRC bits we remove some parity bits from the codeword. The puncturing is surprisingly more implementationally complex than the addition of the CRC, this due to the complex nature of the sequence generator. The puncturing effects the sequence generator since we must consider the parity weight after puncturing. At the sequence generation stage we cannot derive the weight of the CRC as we have not yet produced the entire information sequence. All that can be done is to assume the worst case, all the punctured parity bits will be replaced with zero bits. Potentially there are more sequences generated due to the parity weight loss, that is, the sequences have a lower weight and are therefore output from the sequence generator.

The generated information sequences are then applied to a CRC evaluator, the CRC is appended to the information sequence. The information and CRC are then encoded by the Turbo coder with puncturing applied, this time it is necessary to produce both parity outputs as the sequence is now different from the sequence produced by the sequence generator above. Remember that without the puncturing the sequence generator had already given us one of the parity output weights, we only had to find the other.

4.1.3 Interleaver Modification

The performance of the turbo coder can be improved by modifying the interleaver. The modified interleaver works in the same way as the interleaver described above in Algorithm 2 but has a modified permutation resulting from the method explained below.

Entry Swapping

Sequences that produce a total weight less than the target are recorded. The recorded sequences are used as the basis from which the interleaver is modified. The sequence contains the positions of the 1s in the information input. For each of the elements in the sequence the corresponding entry in the interleaver lookup table can be swapped either up or down. For each entry in the lookup table there is a corresponding flag that indicates that the corresponding entry has been modified. This prevents entries being swapped back when the same index occurs in another sequence. Using this method alone caused the interleaver modifier algorithm to lock up once all the flags are set. An adjustment to the algorithm was made so that if a lockup is detected the flags are cleared to allow modification. Allowing further modification means that an entry can be swapped more than once in the same direction, but this is at the expense of damaging previous modification attempts.

Re-evaluation of Codeword Weight After Interleaver Modification

The interleaver is modified according to the elements of each bad sequence. After the interleaver has been modified for a particular sequence the codeword weight of that sequence is calculated to check that an improvement has been made. When subsequent sequences are used to modify the interleaver it is possible that modifications made for previous sequences are damaged. After modifications have been made for all sequences the process of generating and evaluating sequences must be restarted. On this rerun any damaged modifications will be detected and further modifications to the interleaver will

be made as required. This cycle may repeat many times and currently there is no way to determine the number of cycles required for any given coder and interleaver.

Algorithm 3 Interleaver Modification

```
for i = 0 to k − 1 do
    flag_i ⇐ false
end for
for j = 0 to m_total − 1 do
    if weight(C_i) < w_target then
        swapped ⇐ 0
        for i = 0 to k − 1, swapped = false do
            if C_j(i) = 1 then
                if flag_{i−1} = false then
                    flag_{i−1} ⇐ true
                    swap(Π_{i−1}, Π_i)
                    swapped ⇐ true
                else if flag_{i+1} = false then
                    flag_{i+1} ⇐ true
                    swap(Π_{i+1}, Π_i)
                    swapped ⇐ true
                end if
            end if
        end for
    end if
end for
```

4.1.4 Computer program details: Interleaver modification algorithm

After all the above processes we now have identified all information sequences that give rise to a low codeword weight. By modification of the interleaver it is possible to improve the interleaver so that a higher codeword weight is produced. The works by modifying elements of the interleaver whose indexes correspond with the indexes of the non-zero information bits. The modification made is to swap the element with its nearest neighbour either way depending on which modification improves the codeword weight. Once an interleaver modification has been made a flag is set so that that element is not modified

later by another information sequence. It was found that this approach would lock up when all the flags were set and no modifications were made. The solution was to detect this and then clear the flags and continue the modification process. The algorithm proves to be very successful and results in a good interleaver design that is tailored to the specified encoder design.

4.2 BPSK Turbo codes

4.2.1 Rate 2/3 Turbo code

A d_{free} of 14 was achieved for an interleaver length of 676 using a (1014,676) parallel concatenated turbo code. Both RSCs are identical and have eight states. The coder structure is shown in Figure 1.8. The d_{free} of 14 was not achieved immediately, but was achieved after first reaching 13. As the multiplicities in Turbo10 were much lower for terms upto a weight of 13 a target d_{free} of 13 whose chosen. The algorithm soon achieved this target so it was decided that the algorithm should be made to work harder and try to produce a d_{free} of 14. A software problem was later found that meant that a very few sequences weren't being evaluated correctly. After this bug was fixed it was found that Turbo14 also has a d_{free} of 13. Although the algorithm did not achieve a d_{free} of 14 the multiplicity of the d_{min} term was reduced from 71 down to 5.

4.2.2 Weight Spectra and Performance of $k = 676$ codes

The (1014,676) codes whose weight spectra are shown in Table 4.3, Table 4.4 and Table 4.5 show near identical performance as illustrated in the frame error rate plots in Figure 4.1. The frame error rate plot also shows the Shannon limit curve for comparison, the results are within 0.6dB of Shannon's limit at the frame error rate given for $\frac{E_b}{N_o} = 2.5dB$. The bit error rate plots are shown in figure 4.2.

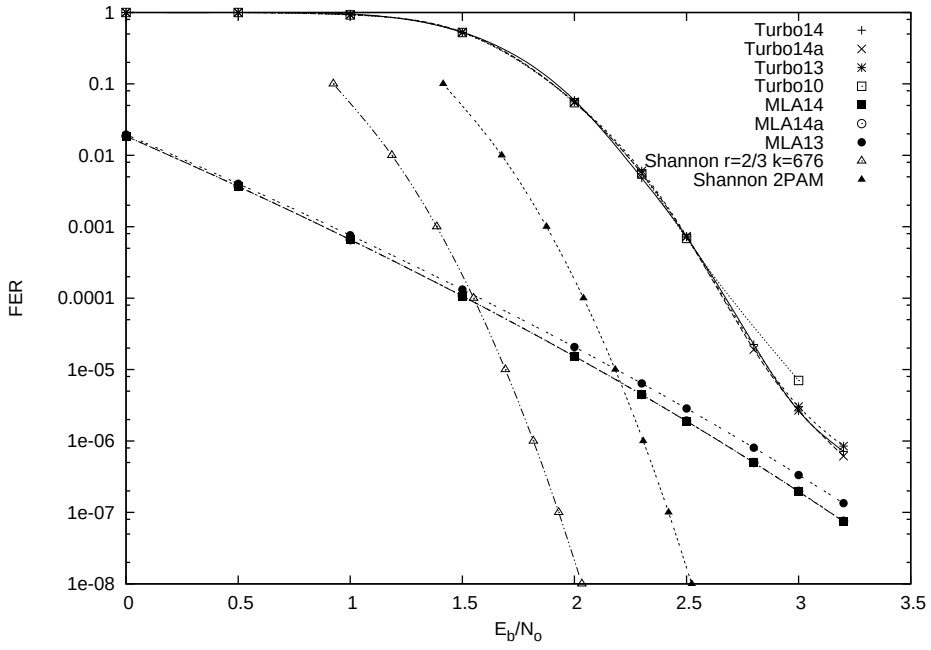

Figure 4.1: Frame error rate curves for four different 2/3 rate codes, k=676 bits.

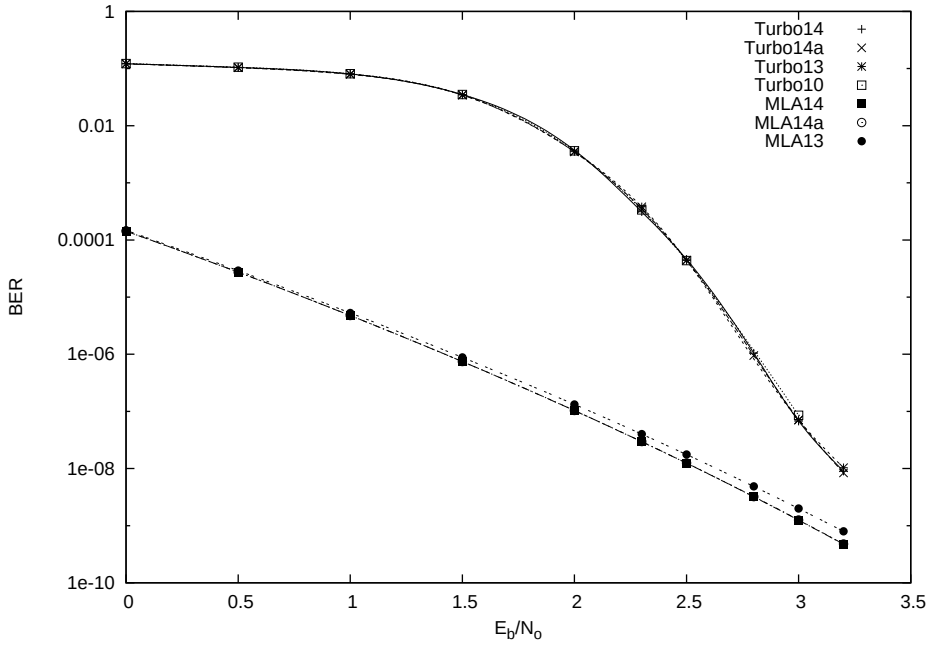

Figure 4.2: Bit error rate curves for four different 2/3 rate codes, k=676 bits.

Union Bound

Codes Turbo14($d_{min} = 13$) and Turbo14a($d_{min} = 13$) have very similar weight spectra as can be seen from Table 4.4 and Table 4.5. The code Turbo13 also has a similar weight spectrum if the $d = 13$ entry is ignored and only $d \geq 14$ entries are considered. The effect of the $d = 13$ entry can just be seen in the frame error rate plot of Figure 4.1. Turbo14($d_{min} = 13$) and Turbo14a($d_{min} = 13$) have near identical frame error rate curves as may be expected from the near identical weight spectra. The union bound calculated from the weight spectra also show these codes to have very similar performance.

Union Bound Calculation

For BPSK the probability of decoder error for a codeword of distance d is given by

$$P_e(d) = \frac{1}{2} M_d \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_o} d\frac{k}{n}}\right) \tag{4.1}$$

The overall probability of decoder error is calculated for the entire range of distance from d_{min} to n.

$$P_e = \sum_{d=d_{min}}^{n} \frac{1}{2} M_d \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_o} d\frac{k}{n}}\right) \tag{4.2}$$

In cases where the entire weight spectrum is not known the union bound can be evaluated from the first N terms of the weight spectrum to approximate frame error rate.

$$\mathrm{FER}_{\mathrm{UB}} = \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2} M_d \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_o} d\frac{k}{n}}\right) \tag{4.3}$$

Similarly the bit error rate is given by

$$\mathrm{BER}_{\mathrm{UB}} = \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2} \frac{w_d}{k} \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_o} d\frac{k}{n}}\right) \tag{4.4}$$

For the weight spectra shown in Tables 4.2, 4.3, 4.4 and 4.5 the union bound has

been calculated for a range of E_b/N_0. The tables illustrate how each term of the weight spectrum influences the union bound calculated for $E_b/N_0 = 3dB$. As the union bound is the summation of the terms derived from the weight spectrum the contribution each term has can be illustrated as bars in the right-most column of the tables. From the bars the effect each term has can be visualised. See Appendix B for the derivation of equation (4.3).

Weight	**Multiplicity**	**Info. Weight**	**UB Contrib.**	$E_b/N_0 = 3db$
9	1	2	4.81678×10^{-7}	
10	4	9	4.83513×10^{-7}	
11	19	49	5.78915×10^{-7}	
12	31	89	2.38967×10^{-7}	
13	66	231	1.2912×10^{-7}	
14	166	673	8.26416×10^{-8}	
15	362	1552	4.59673×10^{-8}	
16	753	3549	2.44382×10^{-8}	
17	1727	8582	1.4351×10^{-8}	
18	3626	18551	7.72735×10^{-9}	
19	7310	38596	4.00087×10^{-9}	
20	13764	74598	1.93721×10^{-9}	
21	25594	141090	9.27412×10^{-10}	
22	45595	254231	4.2581×10^{-10}	
23	78176	440953	1.88347×10^{-10}	
24	128807	732854	8.01303×10^{-11}	
25	207889	1191509	3.34209×10^{-11}	

Table 4.2: Computed weight spectrum for Turbo10.

Weight	Multiplicity	Info. Weight	UB Contrib.	$E_b/N_0 = 3db$
13	71	257	1.38902×10^{-7}	
14	163	650	8.1148×10^{-8}	
15	364	1573	4.62212×10^{-8}	
16	739	3452	2.39839×10^{-8}	
17	1693	8366	1.40685×10^{-8}	
18	3574	18367	7.61654×10^{-9}	
19	7241	38250	3.96311×10^{-9}	
20	13800	74843	1.94228×10^{-9}	
21	25264	139275	9.15454×10^{-10}	
22	45140	251730	4.21561×10^{-10}	
23	76904	434206	1.85282×10^{-10}	
24	127480	725521	7.93048×10^{-11}	
25	204024	1169786	3.27995×10^{-11}	
26	320469	1847259	1.33239×10^{-11}	
27	486695	2818462	5.23677×10^{-12}	
28	727545	4229474	2.02727×10^{-12}	
29	1062896	6198778	7.67454×10^{-13}	

Table 4.3: Computed weight spectrum for Turbo13.

Weight	Multiplicity	Info. Weight	UB Contrib.	$E_b/N_0 = 3db$
13	6	29	1.17382×10^{-8}	
14	150	566	7.46761×10^{-8}	
15	368	1602	4.67292×10^{-8}	
16	803	3772	2.6061×10^{-8}	
17	1646	8154	1.3678×10^{-8}	
18	3528	18130	7.51851×10^{-9}	
19	7160	37711	3.91877×10^{-9}	
20	13865	74938	1.95142×10^{-9}	
21	25795	142300	9.34695×10^{-10}	
22	45586	254250	4.25726×10^{-10}	
23	77447	437249	1.8659×10^{-10}	
24	129211	735973	8.03816×10^{-11}	
25	205761	1179845	3.30788×10^{-11}	
26	322611	1859864	1.34129×10^{-11}	
27	490693	2842251	5.27979×10^{-12}	
28	731543	4253628	2.03841×10^{-12}	
29	1070463	6242890	7.72918×10^{-13}	

Table 4.4: Computed weight spectrum for Turbo14a.

Weight	**Multiplicity**	**Info. Weight**	**UB Contrib.**	$E_b/N_0 = 3db$
13	5	23	9.78186×10^{-9}	
14	151	574	7.51739×10^{-8}	
15	365	1595	4.63482×10^{-8}	
16	811	3808	2.63206×10^{-8}	
17	1643	8147	1.3653×10^{-8}	
18	3542	18203	7.54834×10^{-9}	
19	7147	37637	3.91166×10^{-9}	
20	13886	75052	1.95438×10^{-9}	
21	25858	142665	9.36978×10^{-10}	
22	45443	253394	4.2439×10^{-10}	
23	77227	435978	1.8606×10^{-10}	
24	129128	735412	8.033×10^{-11}	
25	205530	1178682	3.30416×10^{-11}	
26	322601	1859747	1.34125×10^{-11}	
27	490340	2840123	5.27599×10^{-12}	
28	731136	4251321	2.03728×10^{-12}	
29	1069842	6239221	7.7247×10^{-13}	

Table 4.5: Computed weight spectrum for Turbo14.

Error Frames Recorded From Code Simulation

Weight spectra were also produced from the results of the simulation with $E_b/N_0 = 3dB$. It is important to note that the simulation gives a good estimate of the code's performance and is not intended to produce a complete weight spectrum like those discussed above. These weight spectra shown in Tables 4.6, 4.7, 4.8 and 4.9 are plotted in Figures 4.3, 4.4, 4.5 and 4.6 as histograms of the number of codewords having a certain weight.

Inspecting the errored frames produced from the simulation will help identify a connection between the d_{min} of a code and performance. Whilst the Turbo10 code has a $d_{min} = 9$ Table 4.6 shows that most of the error frames have a codeword weight of 11. This coincides with the maximum union bound contribution from the $d = 11$ term of the weight spectrum in Table 4.2. From both the union bound contribution and the simulation results it can be seen that the $d = 11$ term causes the most loss of performance for the Turbo10 code. Thus the $d_{min} = 9$ parameter does not have the largest influence on code performance. However if the Turbo13 code is now considered we see that the $d_{min} = 13$ term has the largest influence on the codes performance as shown in Table 4.7. The largest contribution to the union bound is due to the $d = 13$ term as can be seen in Table 4.3. Generally it has been observed that it is the combination of the multiplicities of the first few non-zero terms of the weight spectrum that have the greatest impact on the performance of a code. In many cases the d_{min} term alone does not determine the performance of a code.

Weight	Multiplicity	Info. Weight
10	4	8
11	10	25
12	2	7
13	2	10
14	3	15
15	0	0
16	1	6
17	0	0
18	2	11
19	0	0
20	1	7
21	0	0
22	1	4
23	0	0
24	2	23

Table 4.6: Recorded error frames for Turbo10.

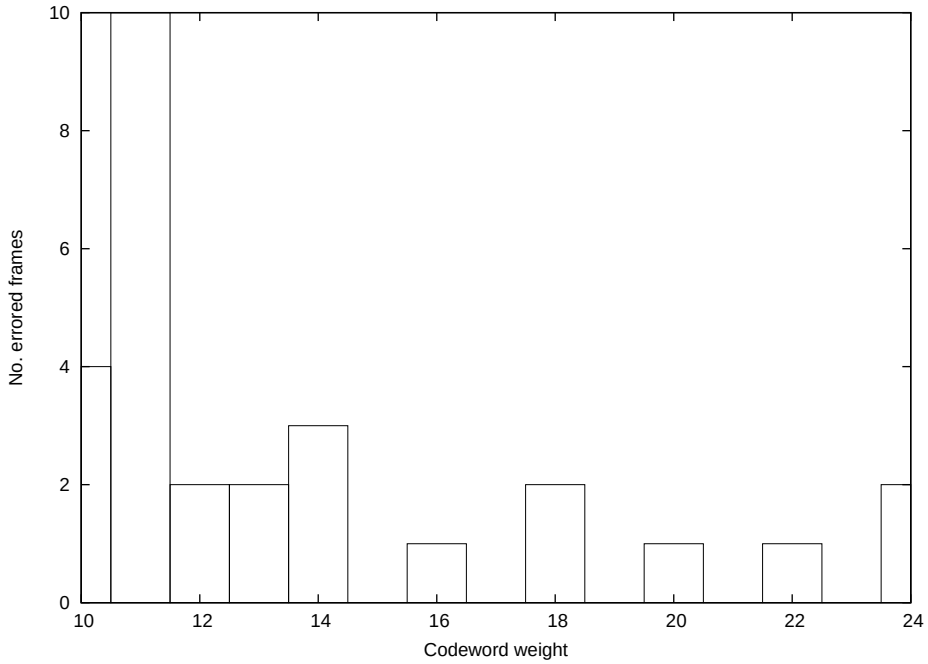

Figure 4.3: Recorded error frames for Turbo10.

Weight	Multiplicity	Info. Weight
13	51	202
14	38	161
15	28	139
16	19	112
17	13	75
18	14	90
19	10	70
20	8	61
21	7	61
22	4	29
23	4	43
24	2	18
25	3	29
26	1	11
27	3	34

Table 4.7: Recorded error frames for Turbo13.

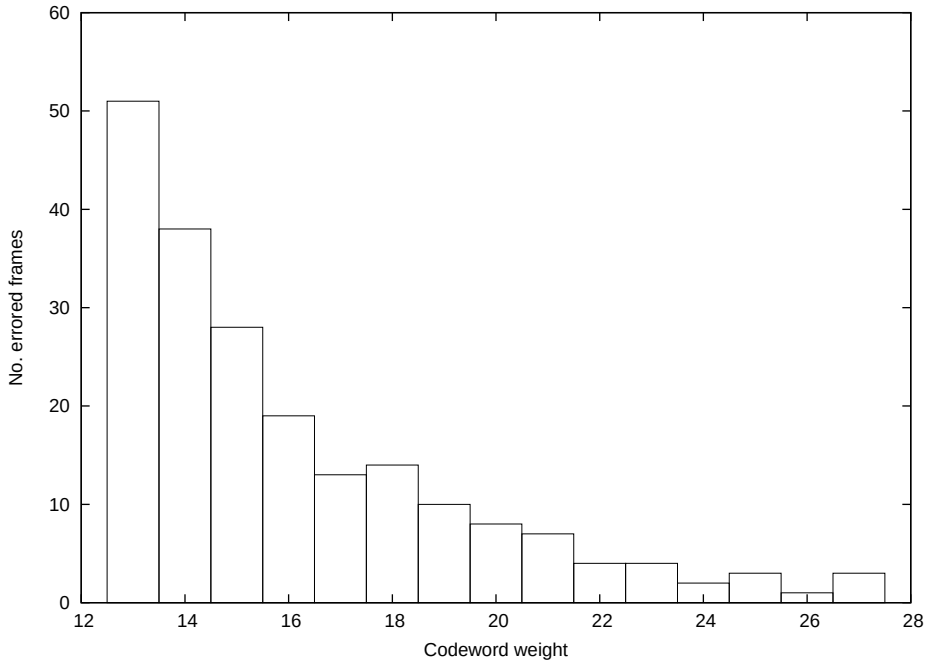

Figure 4.4: Recorded error frames for Turbo13.

Weight	Multiplicity	Info. Weight
13	3	18
14	19	89
15	15	84
16	12	65
17	14	89
18	6	44
19	6	44
20	7	55
21	5	47
22	3	27
23	2	19
24	2	22
25	2	21
26	1	13
27	4	45

Table 4.8: Recorded error frames for Turbo14a.

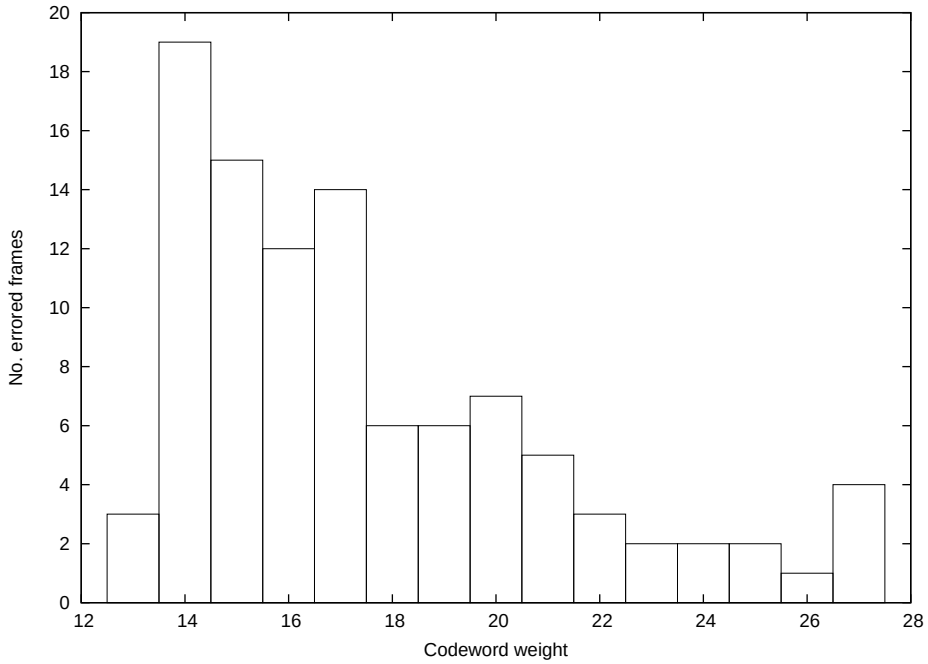

Figure 4.5: Recorded error frames for Turbo14a.

Weight	Multiplicity	Info. Weight
14	14	60
15	21	121
16	5	31
17	3	18
18	8	50
19	3	19
20	3	22
21	4	34
22	2	18
23	4	44
24	3	32
25	0	0
26	0	0
27	2	23
28	1	16

Table 4.9: Recorded error frames for Turbo14.

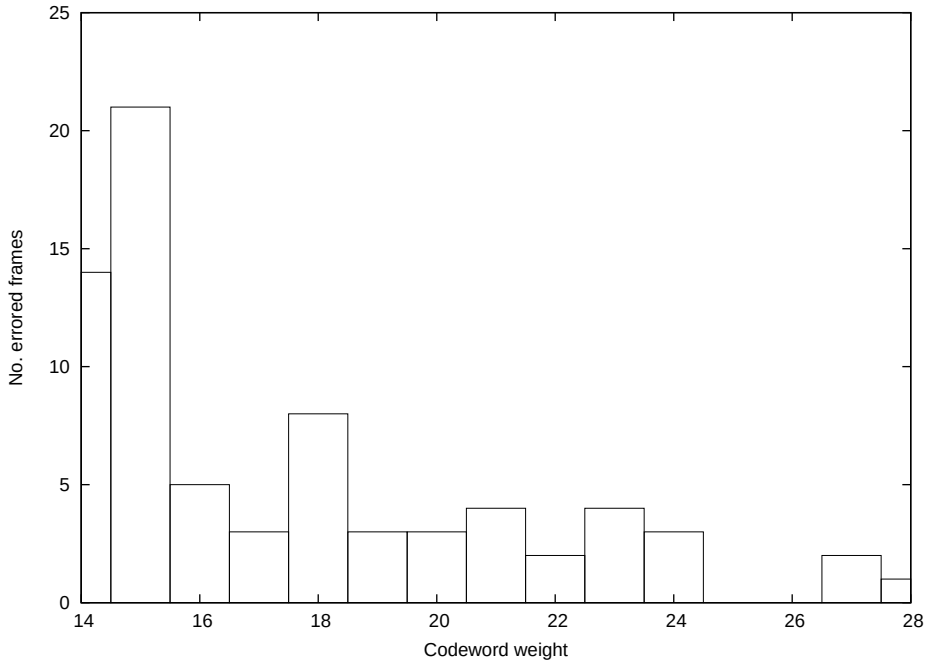

Figure 4.6: Recorded error frames for Turbo14.

4.2.3 Rate 4/5, (4050,3240) Turbo code

It is generally known that a code with higher block length performs better than a code with a low block length. To apply this knowledge to a turbo code the block length was increased from $n = 1014$ to $n = 4050$, giving a (4050,3240) code. Due to the higher code rate the optimised interleaver achieves $d_{min} = 10$ for sequences of up to information weight four. The weight spectrum was evaluated for sequences with information weight up to six and this is shown in Table 4.10.

Weight	**Multiplicity**	**Info. Weight**
8	1	5
9	10	54
10	329	1274
11	775	3368

Table 4.10: Computed weight spectrum for (4050,3240) code

Results in Figure 4.7 show the frame error rate to be within 0.5dB of Shannon's limit constrained to a BPSK modulation. The bit error rate is shown in Figure 4.8. At a frame error rate of 10^{-3} the $k = 676$ code requires an $\frac{E_b}{N_o}$ of approximately 2.5dB whilst the $k = 3240$ code requires an $\frac{E_b}{N_o}$ of 2.8dB. The code rate is 0.8 compared to 0.667, a rate gain of 0.8dB, therefore the $k = 3240$ code has a net gain of 0.5dB. As the $k = 676$ code was within 0.6dB of the Shannon limit, the higher block length code has the better performance as predicted confirming the well accepted trend.

Weight	**Multiplicity**	**Info. Weight**
9	10	54
10	45	247
11	814	3502
12	2034	9507

Table 4.11: Computed weight spectrum for k=3240, $d_{min} = 11$ @ iw=4

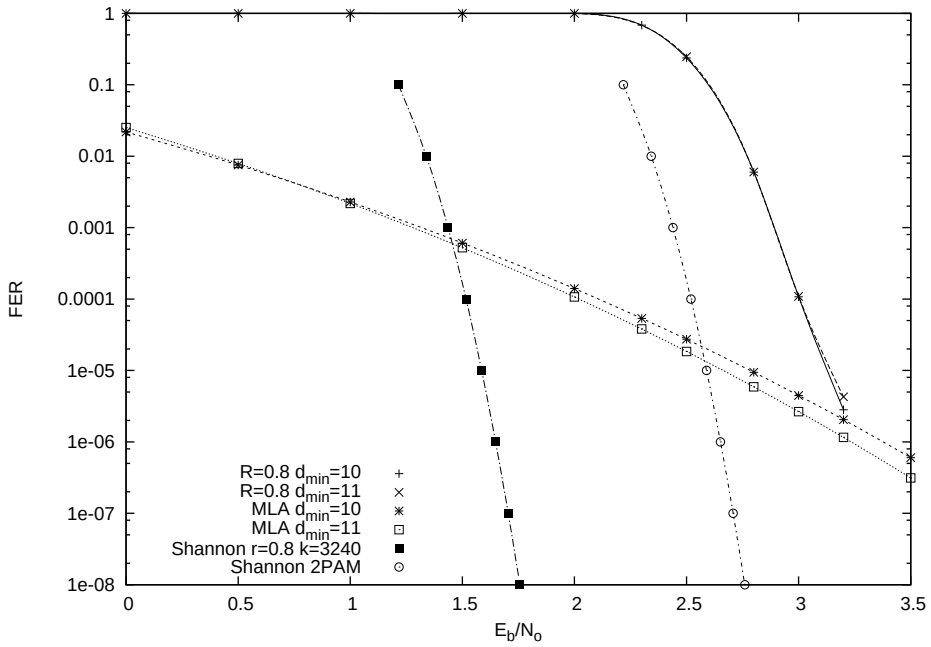

Figure 4.7: Frame error rate curves for four different 0.8 rate codes, k=3240 bits.

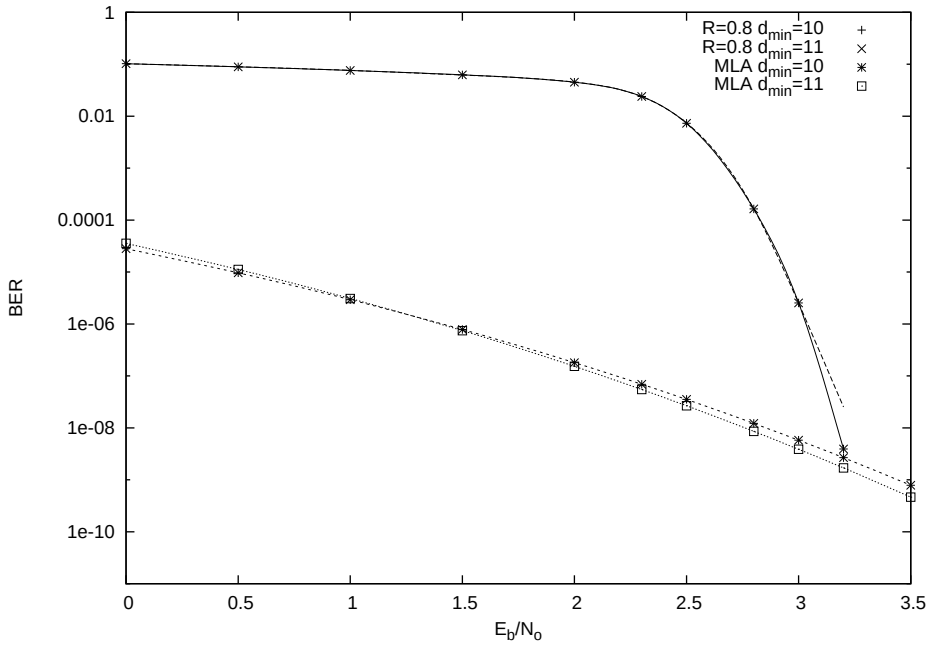

Figure 4.8: Bit error rate curves for four different 0.8 rate codes, k=3240 bits.

4.3 4-PAM code performance

In this section we discuss the comparison of BPSK and 4-PAM. For the 4-PAM scheme the symbols are mapped to the constellation of $-3d$, $-d$, d and $3d$. Appendix B explains this in more detail if required.

4.3.1 (8100,6480) Turbo code

A block length of $n = 8100$ was chosen since this would give the same number of symbols per frame as the $n = 4050$ codes give for BPSK. The code was optimised for for d_{min} produced an improvement from $d_{min} = 8$ to $d_{min} = 32$. The frame error rate performance is plotted in Figure 4.9 in comparison with the Shannon limit for the block length and rate.

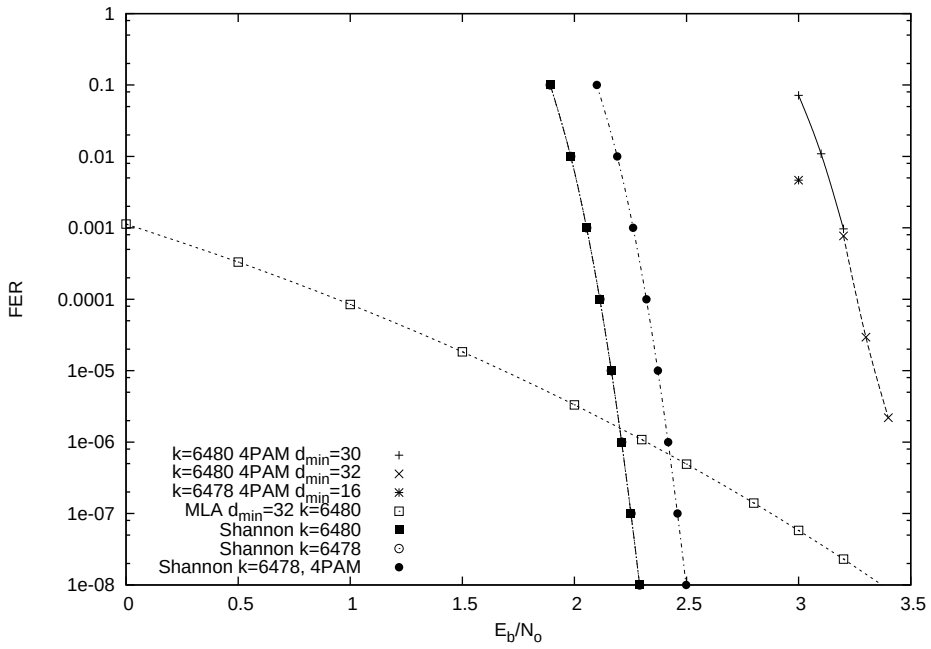

Figure 4.9: Frame error rate curves for four different 0.8 rate codes, k=6480 bits.

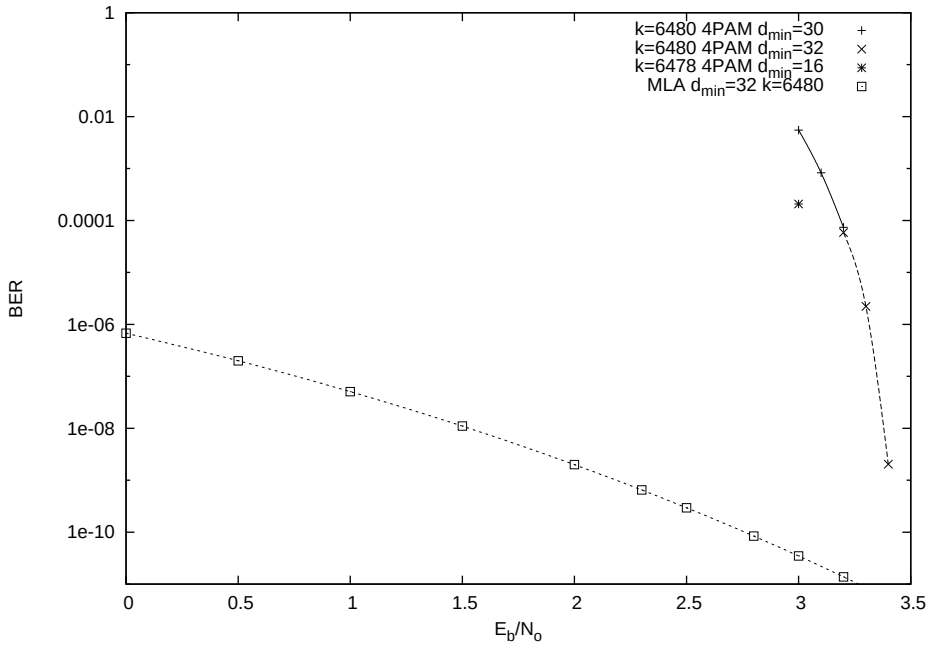

Figure 4.10: Bit error rate curves for four different 0.8 rate codes, k=6480 bits.

4.3.2 k=6478 Turbo code

Decoder convergence problems led to the investigation of codes with a lower order feedback (divisor) polynomial, the order was reduced from three to two. Since the period of the constituent RSCs was reduced to three from seven an information bit length of $k = 6480$ could no longer be used since 6480 is divisible by three and a tail biting code is not available in this circumstance. A new information bit length of 6478 was chosen.

To investigate the performance two different sets of polynomials were used; first set $13_8, 15_8, 7_8$ will be compared against the second $15_8, 17_8, 7_8$. The performance of the $k = 6478$ code is shown in Figure 4.9 and Figure 4.10.

4.3.3 Comparing BPSK and 4-PAM

The modulation constraint of 0.49dB is applied to the Shannon limit for a rate 2/3 code when BPSK modulated. For 4-PAM the modulation constraint is reduced to 0.13dB

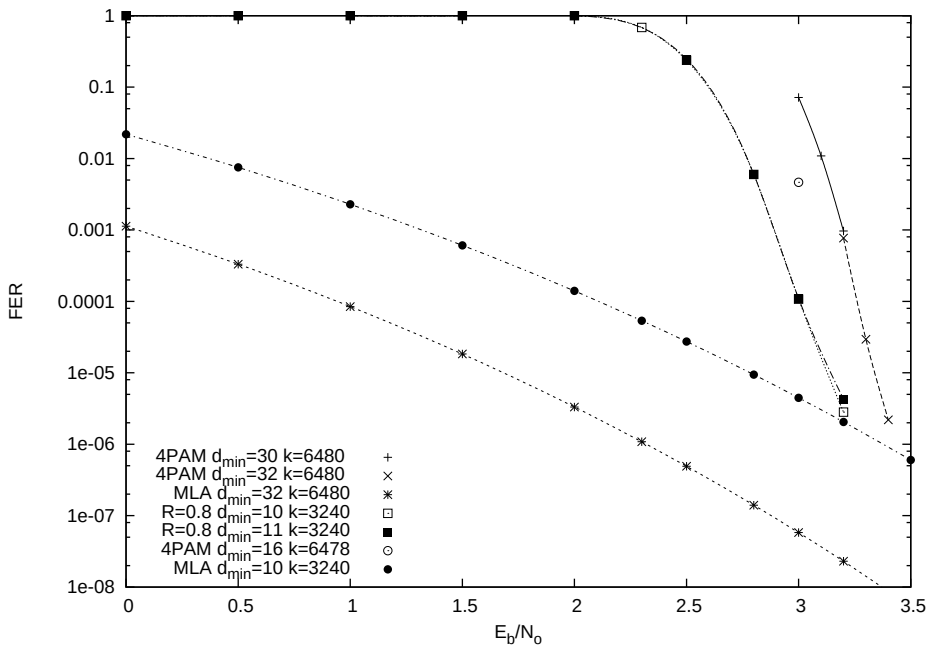

Figure 4.11: Comparison of Frame error rate for 2-PAM and 4-PAM codes

for the same code rate. From this we could hope to realise a 0.36dB improvement in performance from using 4-PAM coding. Firstly the union bound for 4-PAM was derived in order to compare the actual performance of the codes against the best performance obtainable from a code with this weight spectrum.

4.3.4 Union Bound for 4-PAM

For 4-PAM the union bound can be calculated from the first N terms of the weight spectrum.

$$\mathrm{FER} = \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2} N_d \operatorname{erfc}\left(\sqrt{\frac{3}{5}\frac{E_b}{N_o} d\frac{k}{n}}\right) \tag{4.5}$$

This bound is very loose due to the choice of worst case Euclidean distance. See Appendix B for the derivation.

Weight	Multiplicity	Info. Weight
13	5	23
14	151	574
15	365	1595
16	811	3808
17	1643	8147
18	3542	18203
19	7147	37637
20	13886	75052
21	25858	142665
22	45443	253394
23	77227	435978
24	129128	735412
25	205530	1178682
26	322601	1859747
27	490340	2840123
28	731136	4251321
29	1069842	6239221

Table 4.12: Computed weight spectrum for Turbo14.

4.4 Using The Union Bound as a Tool for Selecting Good Performance Codes

In order to determine just how significant the value of d_{min} is on our code's performance the union bound was initially calculated using the first seventeen terms of the weight spectrum as shown in table 4.12. The union bound was then calculated excluding the d_{min} term and as the plot in figure 4.12 shows there was no significant effect on the union bound. Further investigation shows that only the first eight or nine terms are required to calculate the union bound for $E_b/N_o > 2dB$. Figure 4.12 tells us that improving the d_{min} from 13 to 14 will give negligible code improvement, whereas improving the d_{min} to 15 or above would lead to improved performance. The bit error rate plot in Figure 4.13 shows the same characteristics as the frame error rate plot shown in Figure 4.12.

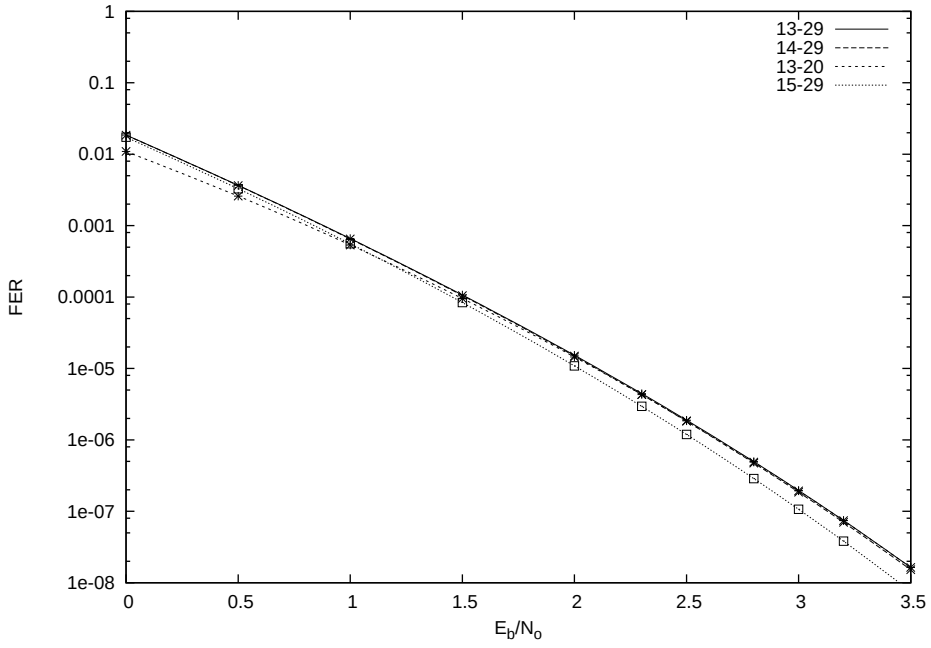

Figure 4.12: Frame error rate union bound computed from subsets of the weight spectrum

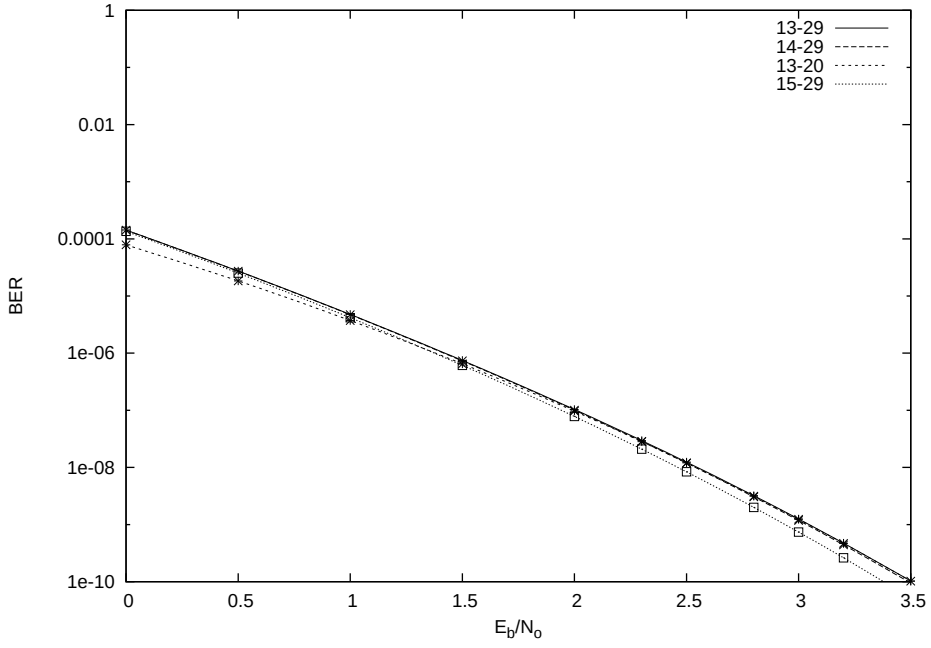

Figure 4.13: Bit error rate union bound computed from subsets of the weight spectrum

5 CRC Outer Code

Work presented in this chapter has been submitted and presented at the IEEE ISIT2004 conference[38], see the paper at the end of the thesis.

5.1 Introduction

There are several advantages in using a CRC or an outer BCH code[39, 40, 41] in conjunction with a turbo code[8], iteratively decoded. It provides a useful stopping criterion for the decoder in conjunction with any other stopping criteria and also strengthens codeword verification further discussed in Section 5.5. It is shown below that a CRC may be used without associated code rate loss and without loss in performance by optimised puncturing of some of the parity bits. Only high weight codewords are punctured so that the d_{min} of the code is not changed. The weight spectrum of the code is evaluated and checked after introduction of the CRC and the punctured parity bits. Surprisingly, a short CRC of a few bits duration produces a dramatic improvement due to a substantial thinning of the weight spectrum. The approach used is to evaluate the weight spectrum of the punctured code in conjunction with optimisation of the interleaver design so that the weight spectrum is better than that of the original code. The fast algorithm for evaluation of the weight spectrum of a turbo code with CRC and arbitrary parity bit puncturing is described as well as the simulation results of the optimised code.

5.2 Encoder Structure

The encoder shown in Figure 5.1 consists of an inner code and an outer code. The outer code is the CRC and the inner code is a PCCC turbo code. When all k information

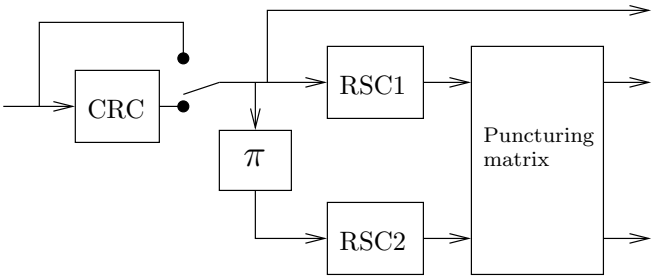

Figure 5.1: The CRC is shown as an outer code. The parity output from the RSCs is punctured.

bits have entered the encoder the CRC remainder is then input into the Turbo encoder. To maintain the code rate it is necessary to puncture m parity bits to make room for the CRC remainder. It is also necessary to puncture a further $m\left(\frac{1}{R}-1\right)$ bits to make room for the parity bits associated with the CRC remainder. A total of $\frac{m}{R}$ parity bits are punctured in order to maintain a code rate that is equivalent to the non-CRC code.

5.3 Weight Spectrum Analysis

Although some methods have been given for weight spectrum analysis for turbo codes in the literature[25, 37], these do not cover the case for a turbo code with CRC as well as arbitrary parity bit puncturing. The fast algorithm has been developed which is based on factoring the overall codeword weight into partitions which may be computed separately, see Section 4.1.2. Firstly the maximum codeword weight required from the weight spectrum analysis is defined, d_{max}. This in turn defines the maximum parity bit weight of interest, $w_{p(target)}$, since the information weight is given by the number of information bits in each codeword. All possible information sequences are generated up to a maximum

user defined information weight using a tree search associated with the trellis of RSC1. The sequences are then interleaved and the corresponding parity output sequences are evaluated and punctured. Sequences that produce a parity sequence weight more than $\frac{w_{p(target)}}{2}$ are discarded. Only sequences with parity weight less than or equal to $\frac{w_{p(target)}}{2}$ are further processed by obtaining the RSC2 parity sequence weight. The procedure is repeated starting with RSC2 retaining output sequences with a parity sequence weight less than or equal to $\frac{w_{p(target)}}{2}$. The corresponding RSC1 parity sequences are then evaluated by de-interleaving the information sequences input to RSC2 and supplying them to RSC1. Sequences that have a total codeword weight less than or equal to d_{max} are recorded in the weight spectrum. The algorithm is as follows:

1. Generate next information sequence with required parity weight using tree search of RSC1 trellis, if there are no more sequences go to 7
2. Calculate CRC and append to information sequence
3. Re-input to RSC1, evaluate parity sequence weight after puncturing
4. Interleave information plus CRC sequence, input to RSC2, evaluate parity sequence weight after puncturing
5. Record total weights for each codeword generated
6. Go to 1
7. Generate next information sequence with required parity weight using tree search of RSC2 trellis
8. Inverse interleave information sequence input to RSC2
9. Calculate CRC and append to information sequence
10. Calculate RSC1 parity weight after puncturing

11. Interleave sequence and calculate RSC2 parity weight after puncturing

12. Record total weights for each codeword generated

13. If there are more sequences go to 7

14. Evaluate the weight spectrum from all retained codewords

5.4 Weight Spectrum of Non-CRC and CRC Turbo Codes

The probability of the decoder making an error in decoding a codeword transmitted over a channel with AWGN is bounded by the classic equation, Equation (5.1), where m_d is the number of codewords with a Hamming weight of d.

$$\mathrm{P_e} \approx \frac{1}{2} \sum_{d=d_{min}}^{d_{max}} M_d \,\mathrm{erfc}\left(\sqrt{\frac{E_b}{N_o}\, d\frac{k}{n}}\right) \tag{5.1}$$

A full weight spectrum consists of values for m_d where $d = d_{min}, d_{min}+1, \ldots, d_{max}$. When a full spectrum is available it is possible to upper bound and lower bound P_e using Equation (5.1) by changing the range of d.

In order to decrease the probability of decoder error it is necessary to decrease the multiplicities, particularly for the first few terms, in the weight spectrum. A procedure for improving the weight spectrum beyond that produced by the CRC is based upon interleaver optimisation and is presented in Section 5.8.

Weight	**Multiplicity**	**Info. Weight**
11	7	29
12	132	393
13	282	1011

Table 5.1: Partial weight spectrum of a $k = 676$, $rate = \frac{2}{3}$ code.

Weight	Multiplicity	Info. Weight
11	5	11
12	13	39
13	9	26

Table 5.2: Partial weight spectrum of a $k = 676$, $rate = \frac{2}{3}$ code, 6 parity bits punctured, 4 bit CRC.

Table 5.1 is the first few terms of the weight spectrum for a k=676, rate $\frac{2}{3}$ code without the benefit of a CRC. Table 5.2 is the weight spectrum for a similar code using a CRC of length 4 bits.

The CRC used has a degree four primitive polynomial of 23_8 and has sixteen possible states. We can therefore expect that for a large fraction (15/16) of codewords will result in a non-zero CRC remainder producing high codeword weight after turbo encoding. The effect of this is that fifteen out of sixteen codewords would benefit from an increase in weight due to the contribution of the CRC remainder bits and can be removed from the weight spectrum in Table 5.1. This spectral thinning effect is visible when Table 5.2 is compared to Table 5.1.

5.5 Decoding improvement due to CRC

The turbo MAP decoder is iterative and can fail to converge to one particular codeword. Often the decoder will alternate between two, sometimes more, solutions. With the CRC being used as an outer code, it can be utilised in the decoder to verify that the decoder has reached the correct solution by performing a simple CRC evaluation.

5.6 Maximum Likelihood Asymptote (MLA)

The MLA shown in Figure 5.2 is calculated from the weight spectrum of Table 5.1. Using Equation (5.1) for $\frac{E_b}{N_o} = 2.5dB$ with the weight spectrum shown in Table 5.2:

$$\begin{aligned} \mathrm{P_e} &\approx \frac{1}{2}\sum_{d=11}^{13} m_d \operatorname{erfc}\left(\sqrt{1.778 \cdot\ d \cdot \frac{676}{1014}}\right) \\ &= \frac{1}{2}\left[1.6366 \times 10^{-6} + 1.2484 \times 10^{-6} + 2.5433 \times 10^{-7}\right] \\ &= 1.5697 \times 10^{-6} \end{aligned}$$

This value, along with other values, is plotted in the graph of Figure 5.2. The MLA gives an approximation to the error floor for $\frac{E_b}{N_o} > 2.8dB$. Figure 5.2 for the simulation results can be seen to approach the plotted MLA. The MLA may be used to estimate the performance of this code and the performance of any new code before lengthy simulations are carried out. Since the simulation results can take several weeks to compute, the MLA saves much time allowing an earlier indication of the codes performance. With the weight spectrum available the MLA may be calculated immediately.

5.7 Simulation Results

Figure 5.2 shows the simulation results for the CRC and non-CRC codes. When a low frame error rate is required the CRC code has better performance than the non-CRC code. This can be seen at the lower frame error rates in Figure 5.2. The CRC code has a lower error floor confirming the predictions from the MLA and weight spectrum analysis.

The results from the simulation show the performance of the CRC code to be within 0.36dB from Shannon's Sphere Packing Bound limit [13, 42] constrained for binary transmission. As the MLA is a union bound (upper bound) code performance can be better as shown by the CRC curve in Figure 5.2.

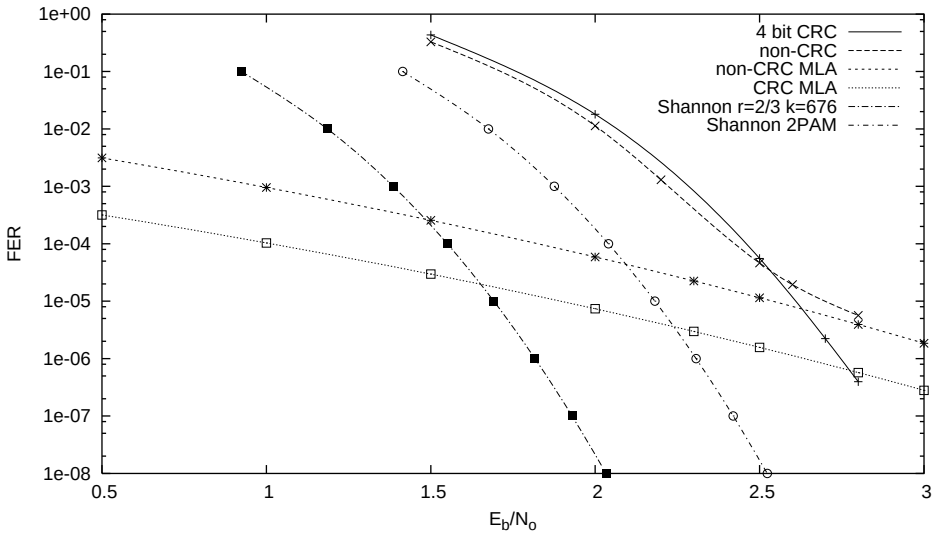

Figure 5.2: Simulation of a $k = 676$, $rate = \frac{2}{3}$ code. Comparing non-CRC code with CRC code.

5.8 Weight Spectrum Improvement Through Optimisation of Interleaver

The weight spectrum is heavily dependent upon the selection of a suitable interleaver. As the weight spectrum is evaluated the interleaver may be modified to improve the minimum distance, d_{min}, to a target value, $d_{min(target)}$. For all codewords that have a distance less than the required target, $d_{min(target)}$, a modification is made to the interleaver. Each of these codewords may be represented by a vector of indices to the non-zero information bits in each codeword. The corresponding elements of the interleaver are swapped with their neighbour (to maintain a reasonable S value), starting with the first information index then taking each index in turn until the codeword weight is raised to $d_{min(target)}$ or higher. Several iterations of weight spectrum evaluation and interleaver modification are necessary as each modification made has an effect on other codewords.

5.9 Conclusion

The results of many code designs of different lengths and code rates, together with the introduction of the CRC, with parity bit puncturing and interleaver optimisation show that the CRC code has a significant performance gain particularly in the region of the error floor, compared to the non-CRC code. This improvement is evident in the weight spectrum as well as in the simulation results.

6 Ordered Reliability List Decoder

6.1 Abstract

Interactive satellite terminals allow satellite links using Variable Coding and Modulation (VCM), as featured in the DVB-S2 standard, to achieve high levels of bandwidth and power efficiency. One of the limitations of efficiency is the coding/modulation used for the return links relies upon an iterative decoder. A new type of decoder suitable for next generation DVB-RCS systems is presented. It is based upon an ordered reliability decoder in conjunction with a Turbo code/ MAP decoder. The decoder algorithm is described and analysed and some results are presented. For the example code cited, it is demonstrated that the hybrid decoder gives an improvement of $0.45dB$ compared to the standard Turbo MAP decoder currently specified in the DVB-RCS standard and is about $0.3dB$ from the sphere packing bound.

6.2 Introduction

The DVB-S2 broadcasting standard [43] allows for a transmission mode in which Variable Coding and Modulation (VCM) and Adaptive Coding and Modulation (ACM) can be employed, whereby time division multiplexed datagrams can use different modulation and coding formats so that the average data throughput can be maximised to classes of receiving terminals with different $\frac{G}{T}$ performances [44]. Significant improvements in data throughput can be realised particularly when a return channel is available[45]. The satellite return channel standard DVB-RCS standard[46]was finalised in 2004 before the DVB-S2 standard and it is highly likely in the near future that a second generation DVB-

RCS standard will be considered that will feature VCM. The DVB-RCS standard features parallel concatenated Turbo codes and iterative decoding and was originally proposed by Berrou et al[47]. Improvements to these original codes, using higher memory for the constituent recursive encoders have been proposed by Berrou at al [48, 49]. The use of higher memory tends to improve the minimum Hamming distance of the Turbo code at the expense of iterative decoder convergence.

It has been noted by several researchers that the iterative decoder limits the performance achievable for Turbo codes[50, 24]. It has been observed in many cases where the Turbo MAP decoder fails to converge, that usually only a few bits remain in error[51]. In this work it is proposed that Ordered Reliability List Decoding is used to correct the remaining bits in error. Analysis is provided of the decoder algorithm and results for an example code which show the effectiveness of the decoder. Ordered reliability techniques (also known as G-space techniques) have previously been applied to BCH codes by Wu and Hadjicostis[52]. Their work shows simulation results for a (128, 64, 22) extended BCH code.

6.3 An Algorithm for a Hybrid Ordered Reliability List Decoder / Turbo MAP decoder

In the proposed scheme an Ordered Reliability List Decoder (ORLD) is provided with extrinsic probability outputs plus the channel values after each MAP decoding cycle in the Turbo decoder. The ORLD takes as input the A Posteriori Probabilities (APP) from the output of the MAP decoder. In many cases the MAP decoder can converge to just a few bits in error in about four iterations. It has been observed that in many cases, when the Turbo MAP decoder fails, that further iterations are unable to alter the output because the extrinsic information saturates or shows evidence of unstable exchanges of extrinsic

information between the constituent decoders. The ORLD is able to break this cycle by forcing the constituent decoders' output to best match possible transmitted codewords.

After each Turbo MAP decoder iteration the output APP's are supplied to the ORLD which then firstly orders them in terms of their reliability, and then generates a list of candidate codewords. Each candidate codeword is correlated with the received vector which is denoted as **R**. The modulation is assumed to be BPSK or QPSK. After demodulation the values of the received vector are ± 1 with added noise (AWGN). Note that the candidate codewords are not correlated with the extrinsic information from the Turbo MAP decoder since these may have become saturated during the Turbo MAP decoding. Also note that the Turbo MAP decoder is not constrained to produce a codeword, this being a feature of convergence failure[51]. However, the ORLD has the advantage that it always produces codewords.

Advantages of Ordered Reliability List Decoding

- Overcomes non-convergence of Turbo MAP decoder.
- Always results in a valid codeword where the Turbo MAP decoder doesn't make this promise.
- Many codewords could be analysed simultaneously, an advantage on massively paralleled architectures, eg. FPGA.

6.3.1 Detailed Operation of the Ordered Reliability List Decoder

In the standard manner first proposed by Dorsch [53], the received vector, **R**, is first ordered according to the reliability of each bit. From the received vector $n - k$ erasures are made to the least reliable bits. The erased bits are then solved using a Gaussian elimination process applied to the parity check matrix, **H**, of the Turbo code. It is

relatively simple to obtain the parity check matrix of the Turbo code from the parity check matrices of the component codes[18]. It is sometimes not possible to solve for a few of the erased bits, in this case the bits are un-erased and the next bits erased. The Gaussian elimination process is then continued until there are $n-k$ erasures that can be solved. The process ensures that all of the erased bits are solved, least reliable first.

It is sometimes the case that errors occur in the k un-erased bits, in this case the solved codeword will not correspond to the transmitted codeword. Many of the solved bits will be in error as they are derived from the erroneous bit. This situation can easily be detected since the derived codeword will have a large Euclidean distance from the received vector due to the many solved bits being incorrect. A standard solution to this problem is to systematically guess bit values in the un-erased bit positions. A fast incremental, correlation approach is proposed below, so that the likely error positions are eliminated first. The decoding procedure stops when the codeword, $S_{best}(x)$, is produced which gives the lowest Euclidean distance from the received vector or the highest cross-correlation, see Section 6.4. This codeword is either not in error or is an MRL error.(A codeword closer to the received vector than the transmitted codeword, so that a maximum likelihood decoder would also fail to decode correctly this received vector). Whilst the ORLD is not limited by convergence problems, it is limited in practical applications by the computational complexity involved in guessing more than a fews bits in error.

6.4 Cross-correlation of Codewords to Determine the Most Likely Codeword

In decoding the codeword with the smallest Euclidean distance from the received vector is the optimum choice having the maximum likelihood of being the correct codeword. This optimal choice will correlate stronger than other codeword choices to the received vector. The cross-correlation is used in this section in order to simplify the mathematical

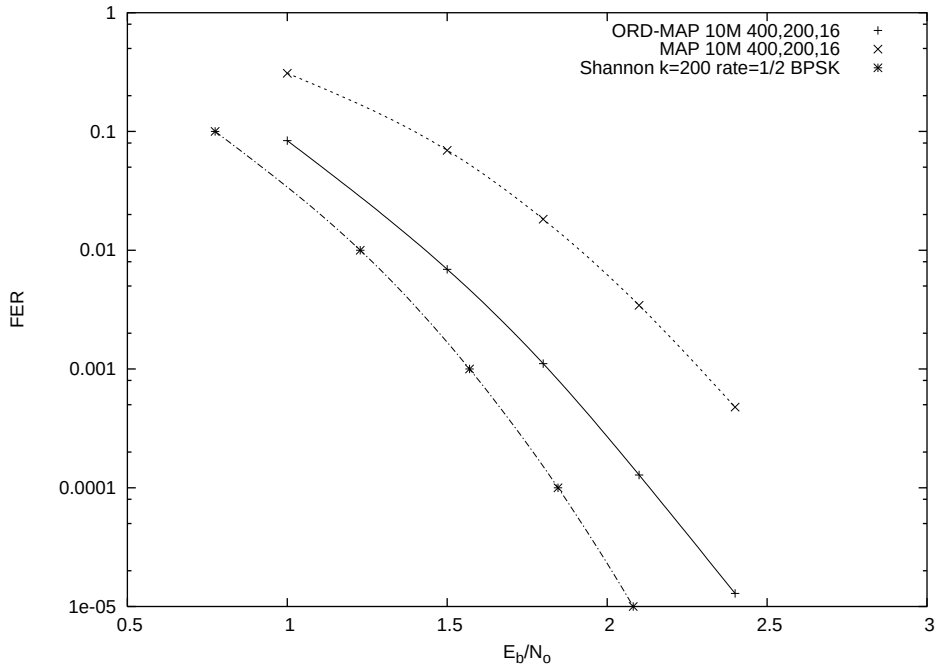

Figure 6.1: Performance of a (400,200), $\frac{1}{2}$ rate, Turbo code comparing the iterative MAP decoder with the hybrid MAP-ORLD decoder.

description. The MAP log-likelihood outputs, $\mathbf{A}$, are permuted by an ordering $\mathbf{a}$ such that

$$|A_{a_0}| \geq |A_{a_1}| \geq \cdots \geq |A_{a_n-1}| \tag{6.1}$$

$\mathbf{\Pi}$ is a permutation vector defined as $\Pi_j = a_{i_j}$ where $i_0, i_1, \ldots, i_{k-1}$ are the smallest index values from $\{0, 1, \ldots, n-1\}$ such that

$$\text{rank} \left[\begin{array}{cccc} g_{\Pi_0} & g_{\Pi_1} & \cdots & g_{\Pi_{k-1}} \end{array} \right] = k \tag{6.2}$$

where g_i is the i^{th} column in the generator matrix, $\mathbf{G}$. The remaining elements of $\mathbf{\Pi}$ are derived from the remaining indexes such that

$$|A_{\Pi_i}| \geq |A_{\Pi_i}| \geq \cdots \geq |A_{\Pi_i}| \qquad i = 0, 1, \ldots, k-1 \tag{6.3}$$

$$|A_{\Pi_i}| \geq |A_{\Pi_i}| \geq \cdots \geq |A_{\Pi_i}| \quad i = k, k+1, \ldots, n-1 \tag{6.4}$$

The received vector, $\mathbf{R}$, is permuted using the same permutation. The k most reliable bits of the permuted received vector, $\mathbf{S}$, are defined as

$$S_i = \frac{1 - \text{sign}(R_{\Pi_i})}{2} \quad i = 0, 1, \ldots, k-1 \tag{6.5}$$

Note that $\mathbf{S}$ takes on values of 0 or 1 and is de-mapped from the BPSK levels of ± 1. The columns of the $\mathbf{G}$ matrix can also be permuted to give $\mathbf{G}'$.

$$G'_{i,j} = G_{i,\Pi_j} \quad i = 0, 1, \ldots, k-1 \quad j = 0, 1, \ldots, n-1 \tag{6.6}$$

An initial permuted codeword is produced from $\mathbf{S}$ and $\mathbf{G}'$.

$$\mathbf{C} = \mathbf{S} \cdot \mathbf{G}' \tag{6.7}$$

Correlation of the initial codeword is given by

$$X = \sum_{i=0}^{n-1} (1 - 2C_i) R_{\Pi_i} \tag{6.8}$$

$$= \sum_{i=0}^{k-1} |R_{\Pi_i}| + \sum_{i=k}^{n-1} (1 - 2C_i) R_{\Pi_i} \tag{6.9}$$

Later the correlation partial products will be used, these are represented by the vector, $\mathbf{V}$

$$V_i = (1 - 2C_i) R_{\Pi_i} \quad i = 0, 1, \ldots, n-1 \tag{6.10}$$

Modification codewords are generated from the 'G' matrix of the code

$$\mathbf{M} = \mathbf{Q} \cdot \mathbf{G}' \mod 2 \tag{6.11}$$

$$M_j = \sum_{i=0}^{k-1} Q_i \cdot G'_{i,j} \mod 2 \quad j = 0, 1, \ldots, n-1 \tag{6.12}$$

Where Q is a vector containing a few non-zero bits placed at indexes where modification is desired based on the bit reliability at that index being low. Since there are only a few non-zero bits in Q it is more efficient to select corresponding rows from the 'G' matrix. A record of the guessed bits is kept in a list, f.

$$M_j = \sum_{q=0}^{p-1} G_{f_q,j} \mod 2 \quad j = 0, 1, \ldots, n-1 \tag{6.13}$$

The correlation of the new codeword is given by

$$\begin{aligned} Y &= \sum_{j=0}^{n-1} V_j \left(1 - 2M_j\right) && (6.14)\\ &= \sum_{j=0}^{n-1} V_j - 2\sum_{j=0}^{n-1} V_j M_j && (6.15)\\ &= X - 2\sum_{j=0}^{n-1} V_j M_j && (6.16) \end{aligned}$$

Typically there are only about $\frac{n-k}{2}$ positions in $\mathbf{M}$ that are non-zero, thus the term of the right of Equation(6.16) involves few calculations. The evaluation of modified codewords only requires summation of the few non-zero elements of this term: $\sum_{j=0}^{n-1} V_j M_j$. The best codeword has the largest negative value for $\sum_{j=0}^{n-1} V_j M_j$ and with this approach the most likely candidate codeword can be quickly determined. Moreover the procedure lends itself to a fast, parallel implementation of the decoder.

6.5 Results

In many applications, satellite return links are used to transmit short packets of data of 200 bits or less. An optimised Turbo code of length 400 bits has been used to evaluate the performance improvement of the hybrid decoder. The Turbo code that been used has been optimised in respect to the multiplicity and minimum Hamming distance of

the weight spectrum which is 16. The code features memory 4 recursive encoders and has a code matched interleaver. Fig. 6.1 shows the performance of the Hybrid decoder in comparison to the MAP decoder and to the sphere packing bound [13]. The Hybrid decoder has an advantage of $0.45dB$ compared to the standard MAP decoder and is $0.3dB$ only, from the best achievable performance for (400,200) codes, as represented by the sphere packing bound.

6.6 Conclusions

It has been shown that a hybrid MAP and ORLD decoder can provide improved performance for short Turbo codes. With the likelihood of a second generation DVB-RCS standard being considered soon, the hybrid decoder is a promising candidate for further evaluation. Future work will consider a range of codes, code rates and higher order modulation formats using this decoder for next generation DVB-RCS.

7 Conclusion

The methods developed in this work have helped in developing new Turbo codes and good interleavers. The results show that the interleaver optimisation program considerably improves the performance of the code. Calculation of the MLA can be used as a guide to the performance of the Turbo codes in the region of the error floor, thus saving time simulating each code. A particularly poor design can be rejected based upon its MLA and weight spectrum, without lengthy simulations. Many codes could be produced and short-listed using the MLA calculation to select the best of a set of codes. It is only necessary to simulate the selected codes, thus saving much time. Decoder convergence problems have also been identified, when these problems are investigated it will be possible to modify the interleaver design programs taking into account the decoders convergence performance.

The addition of a CRC outer code has shown that significant improvements are possible. The CRC does not necessarily increase the d_{min} of the code but does lead to a reduction in the multiplicities of the first few non-zero terms of the weight spectrum. It was previously shown in Section 4.2.2 that reduction of the multiplicities improves code performance without necessarily increasing the d_{min}. Not only does the CRC help the decoder it also provides a mechanism for reducing the multiplicities beyond that possible with interleaver modification alone.

Issues of MAP decoder stability and convergence were successfully improved through the addition of the Ordered Reliability Decoder which decodes the turbo code as a block code. The combination of both decoding schemes proved to have the best performance. Results presented in Section 6.5 are a testimony to the overall combined decoding capability.

7.1 Major Findings

The major contributions this work has made to the field of turbo codes are in the weight spectrum analysis, interleaver design, code design and decoding methods.

7.1.1 New Weight Spectrum Analysis Technique

In Chapter 4 a new weight spectrum analysis method was developed to allow faster evaluation of weight spectra. This meant that new codes could be evaluated much more quickly than by previous methods.

7.1.2 Improved Interleaver Designs

A new interleaver design method was presented in Section 3.1 to work with tailbiting codes. Subsequent improvements where made to the interleaver design firstly by random-swapping and secondly by weight spectrum optimisation.

7.1.3 Improved Code Design

The addition of the CRC outer code, described in Chapter 5, gave significant performance improvement. Not only was the codes performance improved by reduction of the multiplicities in the weight spectrum but also the CRC provides a stopping criteria for the iterative turbo MAP decoder.

7.1.4 Improved Decoding Performance

In addition to the performance obtained from the use of the CRC outer code, further decoder performance was obtained by the development of a hybrid decoding technique introduced in Chapter 6.

7.1.5 Emerging Standards Relating to This Work

Combined CRC and turbo codes are being developed for Digital Video Broadcast (DVB)[29] and 3G and 4G LTE mobile phone standards[30].

7.2 Further or Future work

Throughout the period of this study there has been plenty of branches that could have been explored further. One area that is of particular interest is the possibility of fixing bits in the turbo-MAP decoder so that they cannot alternate, the ORLD can be employed to select with bits to fix. An observation made from the CRC outer coding is that the CRC introduces in most cases extra bits into the input of the turbo encoder. This leads to increased codeword weight in many cases. Can the CRC or some other form of outer code be applied that can lead to an even greater increase in codeword weight?

8 References and Bibliography

[1] R. Michael Tanner. A recursive approach to low complexity codes. *IEEE Transactions in Information Theory*, 27(5):533–547, September 1981.

[2] B. P. Lathi. *Modern Digital and Analog Communication Systems*. Oxford University Press, third edition, 1998. ISBN 0 19 511009 9.

[3] John G. Proakis. *Digital Communications*. McGraw-Hill, fourth edition, 2001. ISBN 0 07 118183 0.

[4] John G. Proakis and Masoud Salehi. *Digital Communications*. McGraw-Hill, fifth edition, 2008. ISBN 0 07 126378 0.

[5] H. Nyquist. Certain factors affecting telegraph speed. *The Bell System Technical Journal*, 3:324–346, April 1924.

[6] H. Herzber. Multilevel turbo coding with short interleavers. *IEEE Journal on Selected Areas in Communication*, 16(2):303–309, Feb 1998.

[7] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[8] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. *IEEE International Conference on Communication*, pages 1064–1070, Geneva, Switzerland, May 1993.

[9] S. Benedetto and G. Montorsi. Serial concatenation of block and convolutional codes. *Electronic Letters*, 32(10):887–888, May 1996.

[10] I.E. Bocharova, R. Johannesson, B.D. Kudryashov, and P. Stahl. Tailbiting codes: Bounds and search results. *IEEE Transactions in Information Theory*, 48(1):137–148, January 2002.

[11] John G. Proakis and Masoud Salehi. *Communication Systems Engineering*. Prentice Hall, second edition, 2002. ISBN 0 13 095007 6.

[12] L.R. Bahl, J. Cocke, F. Jelenik, and J. Raviv. Optimal decoding of linear codes for minimising sybmol error rate. *IEEE Transactions in Information Theory*, 20:284–287, March 1974.

[13] C. E. Shannon. Probability of error for optimal codes in a gaussian channel. *The Bell System Technical Journal*, 38(3):611–656, May 1959.

[14] J. Hokfelt, 0. Edfors, and T. Maseng. A turbo code interleaver design criterion based on the performance of iterative decoding. *IEEE Communications Letters*, 5(2):52–54, Feb 2001.

[15] Oscar Y. Takeshita, Marc P. C. Fossorier, and Daniel J. Costello Jr. A new technique for computing the weight spectrum of turbo-codes. *IEEE Communications Letters*, 3(8):251–253, August 1999.

[16] Jason P. Woodard and Lajos Hanzo. Comparative study of turbo decoding techniques: An overview. *IEEE Transactions on Vehicular Technology*, 49(6):2208–2233, November 2000.

[17] S. Benedetto and G. Montorsi. Design of parallel concatenated convolutional codes. *IEEE Transactions on Communication*, 44(5):591–600, May 1996.

[18] G. Colavolpe. Design and performance of turbo gallager codes. *IEEE Transactions on Communication*, 52(11):1901–1908, Nov 2004.

[19] M. H. Baligh and A. K. Khandani. Asymptotic effect of interleaver structure on the performance of turbo-codes. *Conference on Information Sciences and Systems, Princeton University*, 20^{th}-22^{nd} March 2002.

[20] Kenneth S. Andrews, Chris Heegard, and Dexter Kozen. Interleaver design methods for turbo codes. *IEEE International Symposium on Information Theory*, Cambridge, MA, USA, 16^{th}-21^{st} August 1998.

[21] H. R. Sadjadpour, N. J. A. Sloane, M. Salehi, and G. Nebe. Interleaver design for turbo codes. *IEEE Journal on Selected Areas in Communication*, 19(5):831–837, May 2001.

[22] S. N. Crozier. New High-Spread High-Distance Interleavers for Turbo-Codes. *Proceedings of the 20th Biennial Symposium on Communications, Queen's University, Kingston, Ontario, Canada*, pages 3–7, May 2000.

[23] Ping-Cheng Yeh, Özgür Yilmaz, and Wayne E. Stark. On the error floor analysis of turbo codes: Weight spectrum estimation (WSE) scheme. *IEEE International Symposium on Information Theory*, page 439, Yokohama, Japan, June 2003.

[24] L. C. Perez, J. Seghers, and D. J. Costello Jr. A distance spectrum interpretation of turbo-codes. *IEEE Transactions in Information Theory*, 42(6):1698–1709, June 1996.

[25] R. Garello, P. Pierleoni, and S. Benedetto. Computing the free distance of turbo codes and serially concatenated codes with interleavers, algorithms and applications. *IEEE Journal on Selected Areas in Communication*, 19:800–812, May 2001.

[26] M. Breiling and J. Huber. Combinatorial analysis of the minimum distance of turbo codes. *IEEE Transactions in Information Theory*, March 2000.

[27] C. Doulliard, M. Jezequel, and C. Berrou. The turbo code standard for DVB-RCS. 2^{nd} *Int. Symp. on Turbo Codes & Related Topics*, pages 535–538, September 2000.

[28] M. Ferrari, S. Bellini, M. Tomlinson, and M. Ambroze. Backbone interleaver design for multi-binary turbo codes. 3^{rd} *Int. Symp. on Turbo Codes & Related Topics*, pages 443–446, September 2003.

[29] Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems: ETSI EN 301 790 V1.5.1. May 2009.

[30] LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding: ETSI TS 136 212 V10.6.0. July 2012.

[31] IEEE 802.16m System Description Document (SDD). Dec 2010.

[32] G. Lechner. Convergence of Sum-Product Algorithm for Finite Length Low-Density Parity-Check Codes. *Winter School on Coding and Information Theory*, Monte Verità, Ascona, Switzerland, 24^{th}-27^{th} Feb 2003.

[33] H.Xiao and A. H. Banihashemi. Improved Progressive-Edge-Growth (PEG) construction of irregular LDPC codes. *IEEE Communications Letters*, 8(12):715–717, Dec 2004.

[34] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding Belief Propagation and its Generalizations. *MITSUBISHI ELECTRIC RESEARCH LABORATORIES*, Jan 2002.

[35] J. Hokfelt, 0. Edfors, and T. Maseng. Turbo codes: Correlated extrinsic information and its impact on iterative decoding performance. *Proc. IEEE VTC*, 1999.

[36] J. Hokfelt, 0. Edfors, and T. Maseng. Interleaver design for turbo codes based on the performance of iterative decoding. *IEEE International Conference on Communication*, 1999.

[37] S. Dolinar and D. Divsalar. Weight distribution for turbo codes using random and non-random permutations. *JPL Progress Report 42-122*, pages 56–65, August 1995.

[38] A. J. Rogers, M. Tomlinson, M. A. Ambroze, and M. Z. Ahmed. Enhancement of turbo codes using cyclic redundancy check with interleaver optimisation. *IEEE International Symposium on Information Theory*, Chicago, June 2004.

[39] Shu Lin and Daniel J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, October 1982. ISBN 0 1328 3796 X.

[40] A. Shibutani, H. Suda, and F. Adachi. Decoding-complexity reduction on turbo-crc concatenated code in w-cdma mobile radio. *Proc. IEICE General Conf.*, 1(B-5-109):541, March 1999. (in Japanese).

[41] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello. On the frame-error rate of concatenated turbo codes. *IEEE Transactions on Communication*, 49(4), April 2001.

[42] M. Tomlinson, G. Wade, P. Van Eetvelt, and A. Ambroze. Bounds for finite block-length codes. 149(2), April 2002.

[43] The DVB-S2 standard: ETSI EN 302 307.

[44] D. Breynaert. Analysis of the bandwidth efficiency of DVB-S2 in a typical data distribution network. *CCBN2005*, March 2005.

[45] R. Pieck. Implementation of DVB-S2 into DVB-RCS systems. *Futurecom DVB-RCS Symposium,Florianopolis,Brazil*, Oct 2005.

[46] The DVB-RCS standard: ETSI TR 101 790.

[47] C. Douillard and C. Berrou. Turbo codes with rate-m/(m+1) constituent convolutional codes. *IEEE Transactions on Communication*, 53(10):1630–1638, Oct 2005.

[48] C. Berrou. Designing good permutations for turbo codes: Towards a single model. *IEEE International Conference on Communication*, pages 341–345, June 2004.

[49] S. Benedetto, R. Garello, G. Montorsi, C. Berrou, C. Douillard, D. Giancristofaro, A. Ginesi, L. Giugno, and M. Luise. Mhoms:high-speed acm modem for satellite applications. *Wireless Communications, IEEE*, 12(2):66–77, April 2005.

[50] A. C. Reid, T. A. Gulliver, and D. P. Taylor. Convergence and errors in turbo-decoding. *IEEE Transactions on Communication*, 49(12):2045–2051, Dec 2001.

[51] S. ten Brink. Convergence of iterative decoding. *Electronic Letters*, 35(10):806–808, May 1999.

[52] Yingquan Wu and Christoforos Hadjicostis. Soft-decision decoding of linear block codes using efficient iterative g-space encodings. *IEEE Globecom*, 2:921–925, 2001.

[53] B. G. Dorsch. A decoding algorithm for binary block codes and j-ary output channels. *IEEE Transactions in Information Theory*, 20:391–394, May 1974.

[54] E. Rosnes and Ø. Ytrehus. Turbo stopping sets: the uniform interleaver and efficient enumeration. *IEEE International Symposium on Information Theory*, 20:1251–1255, Sept 2005.

[55] S. Dolinar, D. Divsalar, and F. Pollara. Code performance as a function of block size. *JPL, TMO Progress Report 42-133*, May 1998.

A Computer Programs for the Evaluation of Shannon Limit Curves

A.1 Shannon's Sphere Packing Bound Limit

Shannon's limit gives the theoretical limit of the codes performance at a given signal to noise ratio (SNR). A computer program has been written to calculate this limit for a range of SNR values. The program implements many of the equations in[55]. Shannon showed that an optimum code needs to be received with a minimum SNR in order to acheive a given probability of error[13], this is commonly known as Shannon's limit since it is not possible to produce a code with a performance better than it.

A.2 Degradation to Shannon Limit due to Modulation

When a code is constrained to a digital modulation scheme there is a degradation to the Shannon limit. The degradation is a function of the number of levels used in the modulation and the code rate. For BPSK and a code rate of $\frac{2}{3}$ the Shannon limit is degraded by $0.49dB$. A program was written to calculate the degradation for any given code rate and number of modulation levels.

B Derivation of the Union Bound

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{-x^2}{2\sigma^2}} \tag{B.1}$$

$$M = \frac{1}{\sqrt{\pi}} \int_d^\infty e^{\frac{-x^2}{2\sigma^2}}\, \frac{dx}{\sqrt{2\sigma^2}} \tag{B.2}$$

Let $z = \frac{x}{\sqrt{2\sigma^2}}$ and $\sqrt{2\sigma^2}\, dz = dx$

$$\begin{aligned} M &= \frac{1}{\sqrt{\pi}} \int_{\frac{d}{\sqrt{2\sigma^2}}}^\infty e^{-z^2}\, dz && \text{(B.3)} \\ &= \frac{1}{2} \operatorname{erfc}\left(\frac{d}{\sqrt{2\sigma^2}}\right) && \text{(B.4)} \\ &= \frac{1}{2} \operatorname{erfc}\left(\sqrt{\frac{d^2}{2\sigma^2}}\right) && \text{(B.5)} \end{aligned}$$

An approximation to frame error rate for a coded signal can be calculated from the first N terms of the weight spectrum

$$\text{FER} = \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2}\, M_d \operatorname{erfc}\left(\sqrt{\frac{S}{N_o}}\right) \tag{B.6}$$

where N_o is the noise energy and S is the signal energy defined as

$$S = \frac{d_e^2 R}{4} \tag{B.7}$$

where d_e is the Euclidean distance and R the rate.

M-PAM uses M different amplitudes to represent M different symbols. Adjacent signal amplitudes are seprated by a distance of $2d$[3, pp 169–171][4, pp 98–100].

$$A_m = (2m - 1 - M)d, m = 1, 2, \ldots, M \tag{B.8}$$

2-PAM has a constellation of $A_1 = -d$ and $A_2 = d$. The average energy per symbol is given by

$$\begin{aligned} E_s &= E_b = \frac{(-d)^2 + d^2}{2} && \text{(B.9)} \\ &= d^2 && \text{(B.10)} \end{aligned}$$

The symbols are separated by $2d$ and the minimum Euclidean distance squared is calculated as

$$\begin{aligned} d_e^2 &= d_{min}(2d)^2 && \text{(B.11)} \\ &= d_{min}4d^2 && \text{(B.12)} \\ &= d_{min}4E_b && \text{(B.13)} \end{aligned}$$

$$\begin{aligned} S &= \frac{d_e^2 R}{4} && \text{(B.14)} \\ &= \frac{1}{4} d_{min} 4 E_b \frac{k}{n} && \text{(B.15)} \\ &= d_{min} E_b \frac{k}{n} && \text{(B.16)} \\ \text{FER} &= \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2} M_d \operatorname{erfc}\left(\sqrt{\frac{S}{N_o}}\right) && \text{(B.17)} \\ \text{FER} &= \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2} M_d \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_o} d \frac{k}{n}}\right) && \text{(B.18)} \end{aligned}$$

4-PAM has a constellation of $-3d$, $-d$, d and $3d$. The average energy per symbol is given by

$$\begin{aligned} E_s &= 2E_b = \frac{(-3d)^2 + (-d)^2 + d^2 + (3d)^2}{4} && \text{(B.19)} \\ &= \frac{9d^2 + d^2 + d^2 + 9d^2}{4} && \text{(B.20)} \\ &= 5d^2 && \text{(B.21)} \end{aligned}$$

A sequence of three non-zero symbols is $01, 10, 11$ and has a hamming weight of four. For the worst case the symbols are separated by $2d$ and the worst case minimum Euclidean distance squared is calculated as

$$\begin{aligned} d_e^2 &= d_{min}\,\frac{3}{4}(2d)^2 && \text{(B.22)} \\ &= d_{min}\,\frac{3}{4}4d^2 && \text{(B.23)} \\ &= d_{min}\,3d^2 && \text{(B.24)} \\ &= d_{min}3\frac{2E_b}{5} && \text{(B.25)} \\ &= d_{min}\frac{6}{5}E_b && \text{(B.26)} \end{aligned}$$

$$S = \frac{d_e^2 R}{4} \tag{B.27}$$

$$= \frac{1}{4}\, d_{min} \frac{6}{5} E_b \frac{2k}{n} \tag{B.28}$$

$$= \frac{3}{5}\, d_{min} E_b \frac{k}{n} \tag{B.29}$$

$$\text{FER} = \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2}\, M_d \operatorname{erfc}\left(\sqrt{\frac{S}{N_o}}\right) \tag{B.30}$$

$$\text{FER} = \sum_{d=d_{min}}^{d_{min}+N} \frac{1}{2}\, M_d \operatorname{erfc}\left(\sqrt{\frac{3}{5}\,\frac{E_b}{N_o}\, d\frac{k}{n}}\right) \tag{B.31}$$

C Publications

Enhancement of Turbo codes using Cyclic Redundancy Check with Interleaver Optimisation was published in the proceedings of the 2004 IEEE International Symposium on Information Theory, Chicago, IL USA.

`http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1365544`

Enhancement of Turbo codes using Cyclic Redundancy Check with Interleaver Optimisation

Andrew John Rogers, Martin Tomlinson, Marcel Adrian Ambroze, Mohammed Zaki Ahmed
Digital Communications Research, University of Plymouth, UK
{A.J.Rogers, M.Tomlinson, M.Ambroze, M.Ahmed}@plymouth.ac.uk

Abstract — **A method is presented where a parallel concatenated convolutional code is preceded with a short cyclic redundancy check (CRC) without code rate loss. Parity bits are punctured and their place taken by the CRC. The positions of the punctured parity bits are optimised with reference to the weight spectrum of the overall code in order to produce an overall improvement in code performance. Results are given for the weight spectrum of an optimised code, rate $\frac{2}{3}$, codeword length, n=1014 in comparison with an optimised, non-CRC code using the same rate and codeword length. Frame error rate performance achieved is within 0.36dB of the Sphere Packing Bound constrained for binary transmission.**

I. Introduction

There are several advantages in using a CRC or an outer BCH code[1, 2, 3] in conjunction with a Turbo code[4], iteratively decoded. It provides a useful stopping criterion for the decoder in conjunction with any other stopping criteria and also strengthens codeword verification. It is shown below that a CRC may be used without associated code rate loss and without loss in performance by optimised puncturing of some of the parity bits. Only high weight codewords are punctured so that the d_{min} of the code is not changed. The weight spectrum of the code is evaluated and checked after introduction of the CRC and the punctured parity bits. Surprisingly, a short CRC of a few bits duration produces a dramatic improvement due to a substantial thinning of the weight spectrum. The approach used is to evaluate the weight spectrum of the punctured code in conjunction with optimisation of the interleaver design so that the weight spectrum is better than the original code. A fast algorithm for evaluation of the weight spectrum of a Turbo code with CRC and arbitrary parity bit puncturing is described as well as simulation results of the optimised code.

II. Encoder Structure

The encoder shown in Figure 1 consists of an inner code and an outer code. The outer code is the CRC and the inner code is a PCCC Turbo code. When all k information bits have entered the encoder the CRC remainder is then input into the Turbo encoder. To maintain the code rate it is necessary to puncture m parity bits to make room for the CRC remainder. It is also necessary to puncture a further $m\left(\frac{1}{R}-1\right)$ bits to make room for the parity bits associated with the CRC remainder. A total of $\frac{m}{R}$ parity bits are punctured in order to maintain a code rate that is equivalent to the non-CRC code.

III. Weight Spectrum Analysis

Although some methods have been given for weight spectrum analysis for Turbo codes in the literature[5, 6], these do not cover the case for a Turbo code with CRC as well as arbitrary parity bit puncturing. A fast algorithm has been developed which is based on factoring the overall codeword weight into partitions which may be computed separately. Firstly the maximum codeword weight required from the weight spectrum analysis is defined, d_{max}. This in turn defines the maximum parity bit weight of interest, $w_{p(target)}$, since the information weight is given by the number of information bits in each codeword. All possible information sequences are generated up to a maximum user defined information weight using a tree search associated with the trellis of RSC1 shown in Figure 1. The sequences are then interleaved and the corresponding parity output sequences are evaluated and punctured. Sequences that produce a parity sequence weight more than $\frac{w_{p(target)}}{2}$ are discarded. Only sequences with parity weight less than or equal to $\frac{w_{p(target)}}{2}$ are further processed by obtaining the RSC2 parity sequence weight. The procedure is repeated starting with RSC2 retaining output sequences with a parity sequence weight less than or equal to $\frac{w_{p(target)}}{2}$. The corresponding RSC1 parity sequences are then evaluated by inverse interleaving the information sequences input to RSC2 and supplying them to RSC1. Sequences that have a total codeword weight less than or equal to d_{max} are recorded in the weight spectrum. The algorithm is as follows:

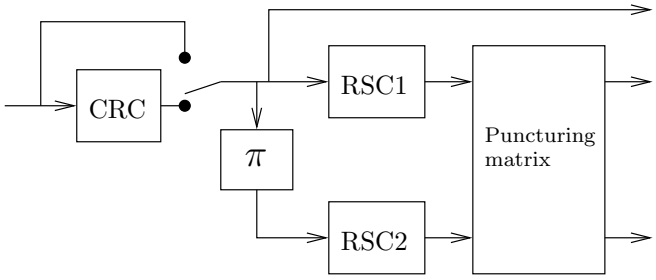

Figure 1: The CRC is shown as an outer code. The parity output from the RSCs is punctured.

1. Generate next information sequence with required parity weight using tree search of RSC1 trellis, if there are no more sequences go to 7
2. Calculate CRC and append to information sequence
3. Re-input to RSC1, evaluate parity sequence weight after puncturing
4. Interleave information plus CRC sequence, input to RSC2, evaluate parity sequence weight after puncturing
5. Record total weights for each codeword generated

6. Go to 1
7. Generate next information sequence with required parity weight using tree search of RSC2 trellis
8. Inverse interleave information sequence input to RSC2
9. Calculate CRC and append to information sequence
10. Calculate RSC1 parity weight after puncturing
11. Interleave sequence and calculate RSC2 parity weight after puncturing
12. Record total weights for each codeword generated
13. If there are more sequences go to 7
14. Evaluate the weight spectrum from all retained codewords

IV. Weight Spectrum of Non-CRC and CRC Turbo Codes

The probability of the decoder making an error in decoding a codeword transmitted over a channel with AWGN is bounded by the classic equation, Equation (1), where m_d is the number of codewords with a Hamming weight of d.

$$\mathrm{P_e} \approx \frac{1}{2} \sum_{d=d_{min}}^{d_{max}} m_d \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_o} d \frac{k}{n}}\right) \qquad (1)$$

A full weight spectrum consists of values for m_d where $d = d_{min}, d_{min}+1, \ldots, d_{max}$. When a full spectrum is available it is possible to upper bound and lower bound P_e using Equation (1) by change the range of d.

In order to decrease the probability of decoder error it is necessary to decrease the multiplicities, particularly for the first term, in the weight spectrum. A procedure for improving the weight spectrum beyond that produced by the CRC is based upon interleaver optimisation and is presented in Section VII.

Weight	Multiplicity	Info. Weight
11	7	29
12	132	393
13	282	1011

Table 1: Partial weight spectrum of a $k = 676$, $rate = \frac{2}{3}$ code.

Weight	Multiplicity	Info. Weight
11	5	11
12	13	39
13	9	26

Table 2: Partial weight spectrum of a $k = 676$, $rate = \frac{2}{3}$ code, 6 parity bits punctured, 4 bit CRC.

Table 1 is the first few terms of the weight spectrum for a k=676, rate $\frac{2}{3}$ code without the benefit of a CRC. Table 2 is the weight spectrum for a similar code using a CRC of length 4 bits.

The CRC used has a degree four primitive polynomial of 23_8 and has sixteen possible states. We can therefore expect that for a large fraction of codewords $\frac{15}{16}$ will result in a non-zero CRC remainder producing high codeword weight after Turbo encoding. The effect of this is that fifteen out of sixteen codewords would benefit from an increase in weight due to the contribution of the CRC remainder bits and can be removed from the weight spectrum in Table 1. This spectral thinning effect is visible when Table 2 is compared to Table 1.

V. Maximum Likelihood Asymptote (MLA)

The MLA shown in Figure 2 is calculated from the weight spectrum of Table 1. Using Equation (1) for $\frac{E_b}{N_o} = 2.5dB$ with the weight spectrum shown in Table 2:

$$\begin{aligned} \mathrm{P_e} &\approx \frac{1}{2} \sum_{d=11}^{13} m_d \operatorname{erfc}\left(\sqrt{1.778 \cdot d \cdot \frac{676}{1014}}\right) \\ &= \frac{1}{2}\left[1.6366 \times 10^{-6} + 1.2484 \times 10^{-6} + 2.5433 \times 10^{-7}\right] \\ &= 1.5697 \times 10^{-6} \end{aligned}$$

This value, along with other values, is plotted in the graph of Figure 2. The MLA gives an approximation to the error floor for $\frac{E_b}{N_o} > 2.8dB$. Figure 2 for the simulation results can be seen to approach the plotted MLA. The MLA may be used to estimate the performance of this code and the performance of any new code before lengthy simulations are carried out. Since the simulation results can take several weeks to compute, the MLA saves much time allowing an earlier indication of the codes performance. With the weight spectrum available the MLA may be calculated immediately.

VI. Simulation Results

Figure 2 shows the simulation results for the CRC and non-CRC codes. When a low frame error rate is required the CRC code has better performance than the non-CRC code. This can be seen at the lower frame error rates in Figure 2. The CRC code has a lower error floor confirming the predictions from the MLA and weight spectrum analysis.

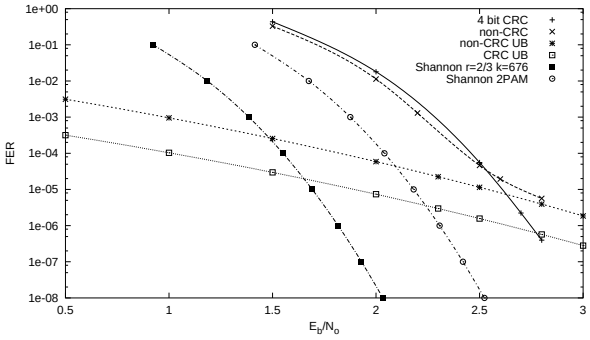

Figure 2: Simulation of a $k = 676$, $rate = \frac{2}{3}$ code. Comparing non-CRC code with CRC code.

The results from the simulation show the performance of the CRC code to be within 0.36dB from Shannon's Sphere Packing Bound limit [7, 8] constrained for binary transmission. As the MLA is a union bound (upper bound) code performance can be better as shown by the CRC curve in Figure 2.

VII. Weight spectrum improvement through optimisation of interleaver

The weight spectrum is heavily dependent upon the selection of a suitable interleaver. As the weight spectrum is evaluated the interleaver may be modified to improve the minimum distance, d_{min}, to a target value, $d_{min(target)}$. For all codewords that have a distance less than the required target, $d_{min(target)}$, a modification is made to the interleaver. Each of these codewords may be represented by a vector of indices to the non-zero information bits in each codeword. The corresponding elements of the interleaver are swapped with their neighbour (to maintain a reasonable S value), starting with the first information index then taking each index in turn until the codeword weight is raised to $d_{min(target)}$ or higher. Several iterations of weight spectrum evaluation and interleaver modification are necessary as each modification made has an effect on other codewords.

VIII. Conclusion

The results of many code designs of different lengths and code rates, together with the introduction of the CRC, with parity bit puncturing and interleaver optimisation show that the CRC code has a significant performance gain particularly in the region of the error floor compared to the non-CRC code. This improvement is evident in the weight spectrum as well as the simulation results.

References

[1] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications,* Prentice Hall, New Jersey, 1983,

[2] A. Shibutani, H. Suda, and F. Adachi, "Decoding-complexity Reduction On Turbo-CRC Concatenated Code in W-CDMA Mobile Radio," *Proc. IEICE General Conf.,* B-5-109, vol. 1, pp. 541, March 1999 (in Japanese)

[3] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, "On the Frame-Error Rate of Concatenated Turbo Codes" *IEEE Transactions On Communications,* Vol. 49, No. 4, April 2001

[4] C. Berrou, P. Thitimajshima, and A. Glavieux, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes," *Proc. IEEE International Conference on Communications,* Geneva, Switzerland, May 1993, pp. 1064-1070.

[5] R. Garello, P. Pierleoni and S. Benedetto, "Computing the Free Distance of Turbo Codes and Serially Concatenated Codes with Interleavers, Algorithms and Applications," *IEEE Journal Selected Areas in Communications,* Vol 19, pp 800-812, May 2001.

[6] S. Dolinar and D. Divsalar, "Weight Distribution for Turbo Codes Using Random and Non-Random Permutations," *JPL Progress Report 42-122,* pp 56-65, August 15 1995.

[7] C. E. Shannon, "Probability of Error for Optimal Codes in a Gaussian Channel," *Bell System Technical Journal,* Vol 38, No. 3, pp 611-656, May 1959.

[8] M. Tomlinson, G. Wade, P. Van Eetvelt and A. Ambroze, "Bounds for finite block-length codes," *IEE Proc. Commun.,* Vol 149, No. 2, April 2002.

Hybrid Decoding of Turbo Codes using Ordered Reliabilities for Next Generation DVB-RCS Interactive Satellite Terminals

Andrew John Rogers, Marcel Adrian Ambroze, Martin Tomlinson
Digital Communications Research, University of Plymouth, UK
{A.J.Rogers, M.Ambroze, M.Tomlinson}@plymouth.ac.uk

I. Abstract

Interactive satellite terminals allow satellite links using Variable Coding and Modulation (VCM), as featured in the DVB-S2 standard, to achieve high levels of bandwidth and power efficiency. One of the limitations of efficiency is the coding/modulation used for the return links relies upon an iterative decoder. A new type of decoder suitable for next generation DVB-RCS systems is presented. It is based upon an ordered reliability decoder in conjunction with a Turbo code/ MAP decoder. The decoder algorithm is described and analysed and some results are presented. For the example code cited, it is demonstrated that the hybrid decoder gives an improvement of $0.45dB$ compared to the standard Turbo MAP decoder currently specified in the DVB-RCS standard and is about $0.3dB$ from the sphere packing bound.

II. Introduction

The DVB-S2 broadcasting standard [1] allows for a transmission mode in which Variable Coding and Modulation (VCM) and Adaptive Coding and Modulation (ACM) can be employed, whereby time division multiplexed datagrams can use different modulation and coding formats so that the average data throughput can be maximised to classes of receiving terminals with different $\frac{G}{T}$ performances [2]. Significant improvements in data throughput can be realised particularly when a return channel is available[3]. The satellite return channel standard DVB-RCS standard[4]was finalised in 2004 before the DVB-S2 standard and it is highly likely in the near future that a second generation DVB-RCS standard will be considered that will feature VCM. The DVB-RCS standard features parallel concatenated Turbo codes and iterative decoding and was originally proposed by Berrou et al[5]. Improvements to these original codes, using higher memory for the constituent recursive encoders have been proposed by Berrou at al [6], [7]. The use of higher memory tends to improve the minimum Hamming distance of the Turbo code at the expense of iterative decoder convergence.

It has been noted by several researchers that the iterative decoder limits the performance achievable for Turbo codes[8], [9]. It has been observed in many cases where the Turbo MAP decoder fails to converge, that usually only a few bits remain in error[10]. In this paper it is proposed that Ordered Reliability List Decoding is used to correct the remaining bits in error. Analysis is provided of the decoder algorithm and results for an example code which show the effectiveness of the decoder.

III. An Algorithm for a Hybrid Ordered Reliability List Decoder / Turbo MAP decoder

In the proposed scheme an Ordered Reliability List Decoder (ORLD) is provided with extrinsic probability outputs plus the channel values after each MAP decoding cycle in the Turbo decoder. The ORLD takes as input the A Posteriori Probabilities (APP) from the output of the MAP decoder. In many cases the MAP decoder can converge to just a few bits in error in about four iterations. It has been observed that in many cases, when the Turbo MAP decoder fails, that further iterations are unable to alter the output because the extrinsic information saturates or shows evidence of unstable exchanges of extrinsic information between the constituent decoders. The ORLD is able to break this cycle by forcing the constituent decoders' output to best match possible transmitted codewords.

After each Turbo MAP decoder iteration the output APP's are supplied to the ORLD which then firstly them in terms of their reliability, and then generates a list of candidate codewords. Each candidate codeword is correlated with the received vector which is denoted as $\mathbf{R}$. The modulation is assumed to be BPSK or QPSK. After demodulation the values of the received vector are ± 1 with added noise (AWGN). Note that the candidate codewords are not correlated with the extrinsic information from the Turbo MAP decoder since these may have become saturated during the Turbo MAP decoding. Also note that the Turbo MAP decoder is not constrained to produce a codeword, this being a feature of convergence failure[10]. However, the ORLD has the advantage that it always produces codewords.

A. Detailed Operation of the Ordered Reliability List Decoder

In the standard manner first proposed by Dorsch [11], the received vector, $\mathbf{R}$, is first ordered according to the reliability of each bit. From the received vector $n-k$ erasures are made to the least reliable bits. The erased bits are then solved using a Gaussian elimination process applied to the parity check

matrix, $\mathbf{H}$, of the Turbo code. It is relatively simple to obtain the parity check matrix of the Turbo code from the parity check matrices of the component codes[12]. It is sometimes not possible to solve for a few of the erased bits, in this case the bits are un-erased and the next bits erased. The Gaussian elimination process is then continued until there are $n-k$ erasures that can be solved. The process ensures that all of the erased bits are solved, least reliable first.

It is sometimes the case that errors occur in the k un-erased bits, in this case the solved codeword will not correspond to the transmitted codeword. Many of the solved bits will be in error as they are derived from the erroneous bit. This situation can easily be detected since the derived codeword will have a large Euclidean distance from the received vector due to the many solved bits being incorrect. A standard solution to this problem is to systematically guess bit values in the un-erased bit positions. A fast incremental,correlation approach is proposed below, so that the likely error positions are eliminated first. The decoding procedure stops when the codeword, $S_{best}(x)$, is produced which gives the lowest Euclidean distance from the received vector or the highest cross-correlation, see Section IV. This codeword is either not in error or is an MRL error.(A codeword closer to the received vector than the transmitted codeword, so that a maximum likelihood decoder would also fail to decode correctly this received vector). Whilst the ORLD is not limited by convergence problems, it is limited in practical applications by the computational complexity involved in guessing more than a fews bits in error.

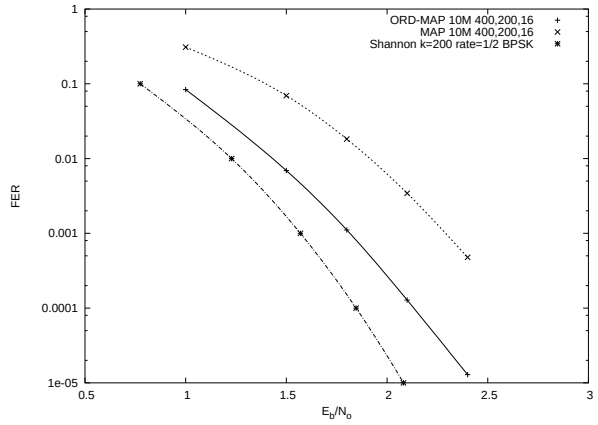

Fig. 1. Performance of a (400,200), $\frac{1}{2}$ rate, Turbo code comparing the iterative MAP decoder with the hybrid MAP-ORLD decoder.

IV. Cross-correlation of Codewords to Determine the Most Likely Codeword

The MAP log-likelihood outputs, $\mathbf{A}$, are permuted by an ordering $\mathbf{a}$ such that

$$|A_{a_0}| \geq |A_{a_1}| \geq \cdots \geq |A_{a_n-1}| \tag{1}$$

$\mathbf{\Pi}$ is a permutation vector defined as $\Pi_j = a_{i_j}$ where $i_0, i_1, \ldots, i_{k-1}$ are the smallest index values from $\{0, 1, \ldots, n-1\}$ such that

$$\text{rank}\left[\begin{array}{cccc} g_{\Pi_0} & g_{\Pi_1} & \cdots & g_{\Pi_{k-1}} \end{array}\right] = k \tag{2}$$

where g_i is the i^{th} column in the generator matrix, $\mathbf{G}$. The remaining elements of $\mathbf{\Pi}$ are derived from the remaining indexes such that

$$|A_{\Pi_i}| \geq |A_{\Pi_i}| \geq \cdots \geq |A_{\Pi_i}| \qquad i = 0, 1, \ldots, k-1 \tag{3}$$

$$|A_{\Pi_i}| \geq |A_{\Pi_i}| \geq \cdots \geq |A_{\Pi_i}| \qquad i = k, k+1, \ldots, n-1 \tag{4}$$

The received vector, $\mathbf{R}$, is permuted using the same permutation. The k most reliable bits of the permuted received vector, $\mathbf{S}$, are defined as

$$S_i = \frac{1 - \text{sign}(R_{\Pi_i})}{2} \quad i = 0, 1, \ldots, k-1 \tag{5}$$

Note that $\mathbf{S}$ takes on values of 0 or 1 and is de-mapped from the BPSK levels of ± 1. The columns of the $\mathbf{G}$ matrix can also be permuted to give $\mathbf{G}'$.

$$G'_{i,j} = G_{i,\Pi_j} \quad i = 0, 1, \ldots, k-1 \quad j = 0, 1, \ldots, n-1 \tag{6}$$

An initial permuted codeword is produced from $\mathbf{S}$ and $\mathbf{G}'$.

$$\mathbf{C} = \mathbf{S} \cdot \mathbf{G}' \tag{7}$$

Correlation of the initial codeword is given by

$$X = \sum_{i=0}^{n-1} (1 - 2C_i) R_{\Pi_i} \tag{8}$$

$$= \sum_{i=0}^{k-1} |R_{\Pi_i}| + \sum_{i=k}^{n-1} (1 - 2C_i) R_{\Pi_i} \tag{9}$$

Later the correlation partial products will be used, these are represented by the vector, $\mathbf{V}$

$$V_i = (1 - 2C_i) R_{\Pi_i} \quad i = 0, 1, \ldots, n-1 \tag{10}$$

Modification codewords are generated from the 'G' matrix of the code

$$\mathbf{M} = \mathbf{Q} \cdot \mathbf{G}' \mod 2 \tag{11}$$

$$M_j = \sum_{i=0}^{k-1} Q_i \cdot G'_{i,j} \mod 2 \quad j = 0, 1, \ldots, n-1 \tag{12}$$

Since there are only a few non-zero bits in Q it is more efficient to select corresponding rows from the 'G' matrix. A record of the guessed bits is kept in a list, f.

$$M_j = \sum_{q=0}^{p-1} G_{f_q,j} \mod 2 \quad j = 0, 1, \ldots, n-1 \tag{13}$$

The correlation of the new codeword is given by

$$Y = \sum_{j=0}^{n-1} V_j \left(1 - 2M_j\right) \tag{14}$$

$$= \sum_{j=0}^{n-1} V_j - 2 \sum_{j=0}^{n-1} V_j M_j \tag{15}$$

$$= X - 2 \sum_{j=0}^{n-1} V_j M_j \tag{16}$$

Typically there are only about $\frac{n-k}{2}$ positions in $\mathbf{M}$ that are non-zero, thus the term of the right of Equation(16) involves few calculations. The evaluation of modified codewords only requires summation of the few non-zero elements of this term: $\sum_{j=0}^{n-1} V_j M_j$. The best codeword has the largest negative value for $\sum_{j=0}^{n-1} V_j M_j$ and with this approach the most likely candidate codeword can be quickly determined. Moreover the procedure lends itself to a fast, parallel implementation of the decoder.

V. Results

In many applications, satellite return links are used to transmit short packets of data of 200 bits or less. An optimised Turbo code of length 400 bits has been used to evaluate the performance improvement of the hybrid decoder. The Turbo code that been used has been optimised in respect to the multiplicity and minimum Hamming distance of the weight spectrum which is 16. The code features memory 4 recursive encoders and has a code matched interleaver. Fig. 1 shows the performance of the Hybrid decoder in comparison to the MAP decoder and to the sphere packing bound [13]. The Hybrid decoder has an advantage of $0.45dB$ compared to the standard MAP decoder and is $0.3dB$ only, from the best achievable performance for (400,200) codes, as represented by the sphere packing bound.

VI. Conclusions

It has been shown that a hybrid MAP and ORLD decoder can provide improved performance for short Turbo codes. With the likelihood of a second generation DVB-RCS standard being considered soon, the hybrid decoder is a promising candidate for further evaluation. Future work will consider a range of codes, code rates and higher order modulation formats using this decoder for next generation DVB-RCS.

References

[1] The DVB-S2 standard: ETSI EN 302 307.

[2] D. Breynaert. Analysis of the bandwidth effi ciency of dvb-s2 in a typical data distribution network. *CCBN2005*, March 2005.

[3] R. Pieck. Implementation of dvb-s2 into dvb-rcs systems. *Futurecom DVB-RCS Symposium,Florianopolis,Brazil*, Oct 2005.

[4] The DVB-RCS standard: ETSI TR 101 790.

[5] C. Douillard and C. Berrou. Turbo codes with rate-m/(m+1) constituent convolutional codes. *IEEE Transactions on Communication*, 53(10):1630–1638, Oct 2005.

[6] C. Berrou. Designing good permutations for turbo codes: Towards a single model. *IEEE International Conference on Communication*, pages 341–345, June 2004.

[7] S. Benedetto, R. Garello, G. Montorsi, C. Berrou, C. Douillard, D. Giancristofaro, A. Ginesi, L. Giugno, and M. Luise. Mhoms:high-speed acm modem for satellite applications. *Wireless Communications, IEEE*, 12(2):66–77, April 2005.

[8] A. C. Reid, T. A. Gulliver, and D. P. Taylor. Convergence and errors in turbo-decoding. *IEEE Transactions on Communication*, 49(12):2045–2051, Dec 2001.

[9] L. C. Perez, J. Seghers, and D. J. Costello Jr. A distance spectrum interpretation of turbo-codes. *IEEE Transactions in Information Theory*, 42:1698–1709, June 1996.

[10] S. ten Brink. Convergence of iterative decoding. *Electronic Letters*, 35(10):806–808, May 1999.

[11] B. G. Dorsch. A decoding algorithm for binary block codes and j-ary output channels. *IEEE Transactions in Information Theory*, 20:391–394, May 1974.

[12] G. Colavolpe. Design and performance of turbo gallager codes. *IEEE Transactions on Communication*, 52(11):1901–1908, Nov 2004.

[13] C.E. Shannon. Probability of error for optimal codes in a gaussian channel. *The Bell System Technical Journal*, May 1959.

[14] E. Rosnes and Ø. Ytrehus. Turbo stopping sets: the uniform interleaver and effi cient enumeration. *IEEE International Symposium on Information Theory*, 20:1251–1255, Sept 2005.