Repeat-Punctured Turbo Codes and Superorthogonal Convolutional Turbo Codes

Narushan Pillay

Department of Electrical, Electronic and Computer Engineering
University of Kwa-Zulu Natal
Durban, South Africa

November 2007

A thesis submitted to the University of Kwa-Zulu Natal in fulfillment of the requirements for the degree of Master of Science in Engineering

Dedication

To,

My family, and in loving memory of my dad, Ronald Pillay

and

The Supreme Personality of Godhead that is the cause of all causes

Abstract

The use of error-correction coding techniques in communication systems has become extremely imperative. Due to the heavy constraints faced by systems engineers more attention has been given to developing codes that converge closer to the Shannon theoretical limit. Turbo codes exhibit a performance a few tenths of a decibel from the theoretical limit and has motivated a lot of good research in the channel coding area in recent years.

In the under-mentioned dissertation, motivated by turbo codes, we study the use of three new error-correction coding schemes: Repeat-Punctured Superorthogonal Convolutional Turbo Codes, Dual-Repeat-Punctured Turbo Codes and Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes, applied to the additive white Gaussian noise channel and the frequency non-selective or flat Rayleigh fading channel. The performance of turbo codes has been shown to be near the theoretical limit in the AWGN channel. By using orthogonal signaling, which allows for bandwidth expansion, the performance of the turbo coding scheme can be improved even further. Since the resultant is a low-rate code, the code is mainly suitable for spread-spectrum modulation applications. In conventional turbo codes the frame length is set equal to the interleaver size; however, the codeword distance spectrum of turbo codes improves with an increasing interleaver size. It has been reported that the performance of turbo codes can be improved by using repetition and puncturing. Repeat-punctured turbo codes have shown a significant increase in performance at moderate to high signal-to-noise ratios. In this thesis, we study the use of orthogonal signaling and parallel concatenation together with repetition (dual and single) and puncturing, to improve the performance of the superorthogonal convolutional turbo code and the conventional turbo code for reliable and effective communications.

During this research, three new coding schemes were adapted from the conventional turbo code; a method to evaluate the union bounds for the AWGN channel and flat Rayleigh fading channel was also established together with a technique for the weight-spectrum evaluation.

"Never discard information prematurely that may be useful in making a decision until all decisions related to that information have been completed."

Andrew James Viterbi

Statement of Originality

I declare that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that it does not contain any material previously published or written by another person except where due reference is made in the text.

Narushan Pillay

Acknowledgements

Firstly and most importantly I would like to express my gratitude to my Supervisor Professor HongJun Xu for his constant motivation, support and guidance throughout my endeavour for MSc Eng. I would also like to thank my co-supervisor Professor Fambirai Takawira for his support and guidance.

I want to express my sincere gratitude to the CSIR for financing my research and making it possible for me to attend two international conferences.

I would like to thank my mum for her constant motivation and support.

I want to express my sincere appreciation to the following people for proof reading my thesis

- Professor HongJun Xu
- Dr Narushni Pillay
- Pragashini Naidoo

Contents

List of Figures

List of Tables

Glossary of Acronyms

DCS	Digital Communication System
VoIP	Voice over Internet Protocol
PGP	Pretty Good Privacy
AWGN	Additive White Gaussian Noise
SNR	Signal-to-Noise Ratio
NLOS	Non-Line of Sight
LOS	Line of Sight
SCC	Serial concatenated code
PCCC	Parallel concatenated convolutional code
HCC	Hybrid concatenated code
RPSCTC	Repeat-Punctured Superorthogonal Convolutional Turbo Code
SCTC	Superorthogonal Convolutional Turbo Code
DRPTC	Dual-Repeat-Punctured Turbo Code
RPTC	Repeat-Punctured Turbo Code
DRPSCTC	Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Code
UMTS	Universal Mobile Telecommunication System
WCDMA	Wideband Code Division Multiple Access
CDMA2000	Code Division Multiple Access 2000
3GPP/3GPP2	3rd Generation Partnership Project
NASA	National Aeronautics and Space Administration
CCSDS	Consultative Committee for Space Data Systems
DVB-T	Digital Video Broadcasting-Terrestrial
IIR	Infinite impulse response
RSC	Recursive systematic code
NSC	Non-recursive systematic code
FIR	Finite impulse response
PHY	Physical layer of open systems interconnection model
MAP	Maximum a-posteriori
BCJR	Bahl, Cocke, Jelinek, Raviv

APP	A-posteriori probability
pdf	probability density function
cvs	Continuous valued signal
LLR	Log-likelihood ratio
ISI	Inter-symbol interference
FRF	Flat Rayleigh Fading channel
BPSK	Binary phase shift keying
SI	Side information
BER	Bit error rate
QoS	Quality of Service
SSM	Spread-spectrum modulation
DSSS	Direct sequence spread-spectrum
CDMA	Code division multiple access
ARQ	Automatic Repeat-Request
TC	Turbo Code
PCC	Parallel concatenated code
VA	Viterbi Algorithm
SOVA	Soft-output Viterbi algorithm
WEF	weight enumerating function

Glossary of Symbols

m_i	Message symbol for $i = 1,...,M$ members
u_i	Channel symbol for $i = 1,...,M$ members
R	Data rate
P_E	Probability of error
C	Channel capacity
B_w	Channel bandwidth
E_s	Average signal energy
T	Time period/interval
N_0	Noise power spectral density
P_E	Probability of error
K	Constraint length
$g(D)$	Generator polynomials
E_b	Energy per bit
n	Number of bits in the codeword
m	Memory depth
k	Number of message sequence bits
d_k	$k-th$ message bit
u_k	systematic bit
v_k	parity bit
G_1, G_2	Generator polynomials
a_k	encoder state bit
π	interleaver
π^{-1}	deinterleaver
d	message sequence
P	permutation mapping
dP	interleaved message sequence

P^T	transpose permutation
N	Input frame length
x^s	transmitted systematic bit
x^p	transmitted parity bit
x_k^s	$k-th$ transmitted systematic bit
y_k^s	$k-th$ received systematic bit
x_k^{p1}	$k-th$ transmitted first parity bit
x_k^{p2}	$k-th$ transmitted second parity bit
y_k^{p1}	$k-th$ received first parity bit
y_k^{p2}	$k-th$ received second parity bit
L_{e1}	extrinsic log-likelihood ratio from first decoder
L_{e2}	extrinsic log-likelihood ratio from second decoder
$L(d_k)$	soft log-likelihood ratios
a_1, a_2	bits in shift register
S_k, S_{k-1}	current state, next state
β_k	reverse state metric
y_k^N	$k-th$ received bit of length N
ϕ	any arbitrary transition from trellis
δ	branch metric
q_k	received bit
σ	standard deviation
T_{spread}	spreading interval for fading
f_n	Doppler shift
f_d	Doppler frequency
β_n	for zero cross correlation in Jakes model
Q	Q-function
A_d	multiplicity
L_{e12}, L_{e21}	log-likelihood extrinsic information

B	interleaver amplitude/length
P_b	probability of bit error
∂_1, ∂_2	intersection points of two-signal class
φ_0	decision line in the two signal class case

Chapter 1

Introduction

Presently, and in the recent years, digital communications has been attracting wide span attention because of the exponential increase in the demand for data communications, and because of the data processing techniques and flexibilities allowed, in comparison to its counterpart analog communications [1]. There are many reasons why both military and commercial communication systems have "gone digital". One of the primary advantages is the ease with which digital signals are regenerated, compared to analog signals. Digital circuits are also known to exhibit higher immunities to distortion and interference, which are both deleterious inputs to the efficient performance and operation of a communication system.

Digital techniques also lend themselves to signal processing functions that protect against interference and signal jamming, or that provide signal encryption and user privacy [1], [9]. To counteract the unavoidable noisy channel problem, information needs to be protected prior to transmission. This protection is known as channel coding, or more commonly as error-control coding. Convolutional codes, a well-established type of error-correcting code, were introduced to the coding community in 1955 by Peter Elias [1], [9]. These codes have often been used for performance improvement in digital radio, mobile phones, satellite links, Bluetooth implementations and even deep-space communications, for example, the Mars Pathfinder or the Cassini probe to Saturn [1], [31]. These codes are now giving way to turbo codes, a new class of iterated convolutional codes that closely approaches the theoretical limits imposed by Shannon's theorem with much less decoding complexity, than the Viterbi algorithm for original convolutional codes that would be required for the same performance.

1.1 Structure of the Digital Communication System

The generic structure of a digital communication system (DCS) is shown in Figure 1.1A.

This block diagram represents the basic elements that are of interest in this dissertation and excludes other elements e.g. synchronization, etc. The upper half of the block diagram, i.e., information source, source encoding, encryption, channel coding and modulation, represents the transmission of data from the source to the transmitting antenna. The lower half of the diagram, i.e., demodulation, channel decoding, source decoding, decryption and estimation of the information represents the reception of data from the receiving antenna to the user/operator. The transmission and receiving parts are separated by a link or communications channel.

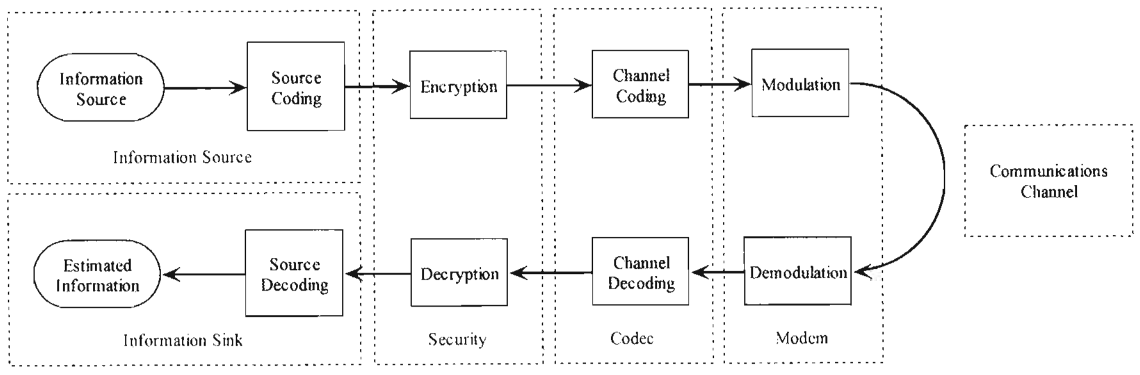


Figure 1.1A Generic structure of a Digital Communications System

The signal processing steps that are performed prior to transmission, are for the most part, reversed in the receiver. In a digital communication system, the information source is converted to binary bits, which are then grouped to form message symbols. Each message symbol m_i, $i = 1,...,M$, can be regarded as a member of a finite alphabet set containing M members. Source coding is used to convert analog signals to digital format (digitizing), and also serves to remove redundant or unneeded information from the original message sequence. Compression, either lossless or lossy, is also done within the source coding block.

To provide communication privacy, encryption is used. Encryption prevents unauthorized users from tapping into messages and from injecting erroneous or false messages into the system.

Channel coding introduces redundancy and transforms a sequence of message symbols into a sequence of channel symbols u_i for error correction or control. For a given coding rate R, channel coding can reduce the probability of error, P_E in the received sequence, or reduce the required signal-to-noise ratio (SNR) to achieve a desired P_E, at the expense of transmission bandwidth or alternatively decoder complexity [1], [9].

Table 1.1. Examples for transmitter constituents of the digital communication system [1]

Information Source
• Imaging Satellite – capturing images in deep space that are sent back to earth • Television broadcasting - camera • VoIP – Voice over Internet Protocol – microphone
Source Coding
• Predictive coding • Block coding • Lossless/Lossy compression • Variable length coding
Encryption
• PGP – Pretty Good Privacy
Channel Coding
• Convolutional coding • Block coding • Concatenated coding
Modulation
• Phase, frequency, amplitude shift keying

Table 1.1 gives examples for each of the upper component blocks in the DCS.

1.2 The Shannon Limit

Shannon showed that the capacity C of a channel perturbed by additive white Gaussian noise (AWGN), is a function of the average received signal power E_s and the transmission bandwidth [1], [9]. Shannon's theorem is stated in Theorem 1.1 below [1], [9].

Theorem 1.1:
Consider an AWGN channel with channel capacity C. There exist error control coding techniques such that information can be transmitted via the channel at code rates less than C with an arbitrarily low bit error rate.

The above theorem states what Claude Shannon presented in his paper, "A Mathematical theory of Communication", where he defined the capacity of a channel as

$$C = B_w \log_2\left(1+\frac{E_s}{N_0}\right) \textit{ bits per second} \tag{1.2a}$$

where B_w is the bandwidth of the channel in Hertz, and E_s / N_0 is the average signal-to-noise energy ratio in each signaling interval of duration T.
Shannon stated that for code rates $R \leq C$, it was theoretically possible to transmit with an arbitrarily small bit-error probability, or that for an information code rate $R > C$ it is not possible to achieve a code that could exhibit an arbitrarily low error probability. Shannon's paper showed that the values of signal-to-noise ratio and bandwidth set a limit on the transmitting rate and not on the error probability. The Shannon limit is known to be at $E_b / N_0 = -1.6$ dB corresponding to a probability of bit-error P_b [1], [9]. It is impossible in practice to reach this limit because bandwidth requirements and implementation or computational complexities increase without bound [1]. Shannon then proceeded to prove the noisy Channel Coding theorem, thereby launching the field of error-control coding on its quest. The fact remained, however, that forty five years after

Shannon's findings were presented, a gap in coding gain of approximately 2dB continued to separate the performance of the most advanced error-control systems from the theoretical limit. This gap vanished overnight with the advent of turbo coding in the year 1993 by Berrou *et. al.*.

1.3 Noise in Communication Systems

One of the major problems faced by communication systems is noise, which is unwanted electrical signals that lead to distortion in the communication link. The presence of noise superimposed on transmitted signals tends to obscure the signal, and thus limit the receiver's ability to make correct decisions that are sent to the user/operator. Noise is created by both natural and man-made means. Spark-plug ignition noise, switching transients and electromagnetic radiating signals are some of the examples of man-made noise sources amongst others, while natural means of noise includes the atmosphere, sun and a variety of galactic sources.

The additive white Gaussian noise (AWGN) channel produces simple, tractable models which are useful for understanding the underlying behavior of a communication system (both analog and digital). The only data impairment produced by AWGN is the superposition of wideband or white noise with uniform power spectral density and Gaussian distributed amplitudes. AWGN assumes a zero mean and a variance $N_0/2$, where $N_0/2$ is the two-sided noise spectral density. In mobile communications, more complex channel models need to be investigated to gain insight for fading channels. Fading refers to the distortion faced by a carrier modulated signal over certain propagation media. In wireless communications, fading is due to multipath fading and is often referred to as multipath induced fading. Figure 1.3A illustrates how multipath induced fading is manifested. Multipath refers to the various paths signals take, on their route to the receiver, shown by the solid lines. As a consequence of these multiple paths and the presence of obstructions, as shown in Figure 1.3A, atmospheric ducting, refraction, diffraction and reflection all contribute to the changes in phase, constructive or destructive superposition

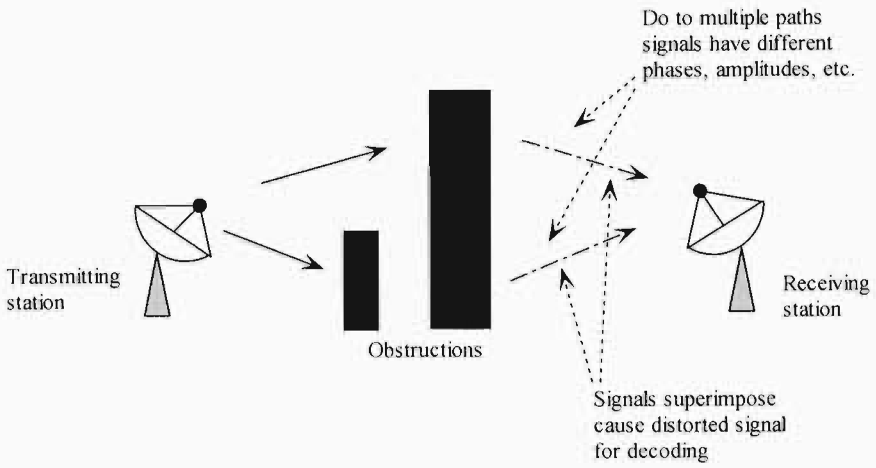

Do to multiple paths signals have different phases, amplitudes, etc.
Transmitting station
Obstructions
Receiving station
Signals superimpose cause distorted signal for decoding

Figure 1.3A Manifestation of multipath induced fading

of the final signals, depicted by the dash-dotted lines. The Rayleigh distribution or Rayleigh fading model is used for non-line-of-sight (NLOS), (obstructed path from transmitter to receiver), fading in communication links. However, if a strong line-of-sight (LOS), (unobstructed path from transmitter to receiver) component is present, then the fading is modeled by a Rician distribution.

1.4 Concatenation Schemes

Serial concatenated codes (SCC) were proposed by Forney in 1966 and were the first examples of error-control codes that utilize polynomial-time decoding to correct linearly increasing rations of errors [5]. In the 1990s, various versions of concatenation emerged with the advent of turbo codes and other iteratively decodable families.

Conventional turbo codes (parallel concatenated codes), represent a breakthrough in coding theory, and have inspired and motivated a large amount of new research. These codes are parallel concatenated convolutional codes (PCCC), whose encoder is typically formed by two constituent systematic encoders separated by an interleaver.

Parallel concatenated convolutional codes yield very large coding gains (10-11 dB) at the expense of bandwidth expansion or very small data reductions [5]. Figure 1.4A shows the structure of the parallel concatenated code (PCC). Two constituent codes are separated by an interleaving mechanism and joined in parallel.

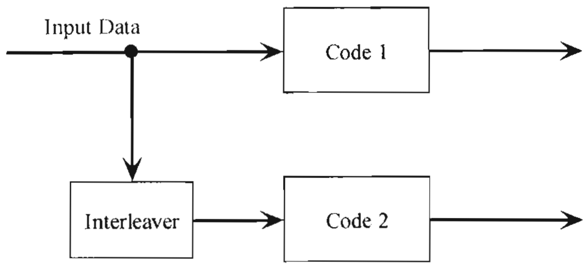


Figure 1.4A   Structure of the Parallel Concatenated Code

The structure of the serial concatenated code (SCC) is depicted in Figure 1.4B. Here there is an outer code and an inner code joined by an interleaving mechanism. The input information bits, feed the outer encoder, and after being permuted by an interleaver, they enter the inner encoder. Other concatenation schemes have also been proposed, viz. Hybrid concatenated codes (HCC) and Self-concatenated codes. In this document, parallel concatenated codes are most important, since this determines the primary structure of turbo codes.

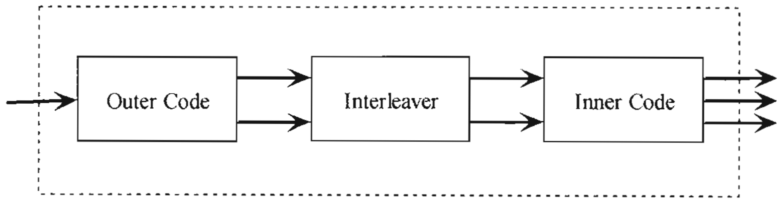


Figure 1.4B   Structure of the Serial Concatenated Code

1.5 Motivation and Research Objective

Turbo codes are a powerful branch of error-control coding that has exhibited performances close to the theoretical Shannon limit (approximately a few tenths of a dB from the limit), whilst offering a reasonable level of computational complexity. The performance of turbo codes has been shown to increase with increasing block length. The major factors that contribute to the performance and the computational complexity of the conventional turbo code are [1], [9], [24], [32]:

- The constraint length, K, of each component code, and consequently the memory
  - The larger the constraint length, the higher the complexity and the more stringent the memory requirements. For the turbo code usually a constraint length of $K=3$ to $K=5$ is chosen for best performance [1].
- The generator polynomials, $g(D)$, for the component codes
  - These are chosen in accordance with the signal-to-noise ratio. They affect the waterfall region and the error floor region of the performance characteristic curve.
- The decoding algorithm utilized to yield an estimate of the message sequence
  - This affects the waterfall region and the complexity of the code
- Internal decoding iterations
  - The number of iterations determines how good the performance is. Increasing the number of iterations also incurs a delay and a greater computational complexity.
- Interleaving technique employed
  - Various designs for interleaving are available, the more elegant the design the better the performance and the higher the complexity depending on how well the interleaver is designed.
- The amplitude of the frame length or the interleaver
  - As the frame length, and consequently the interleaver size, is increased a more superior performance is yielded.
- Puncturing of the output codewords
  - Puncturing serves to increase the code rate but consequently decreases the performance of the code.

The second last point conveys to us that the larger the frame amplitude or the interleaver amplitude the better the performance that can be achieved. However, by using larger frame sizes, larger delays are created. Large delays cannot be tolerated in all applications, for example, real-time telephony or voice applications.

In this thesis we will examine a novel technique for exploiting this factor by increasing just the size of the interleaver and using small frame sizes.

1.6 Research Contributions

The major contributions of this thesis can be summarized as follows:

i. Repeat-Punctured Superorthogonal Convolutional Turbo Codes (RPSCTC), a scheme that uses repetition and puncturing to improve the performance of Superorthogonal Convolutional Turbo Codes (SCTC) for AWGN and Rayleigh fading channels.
ii. A method to evaluate the performance of the new scheme, RPSCTC, in both AWGN and Rayleigh fading channels.
iii. A distance spectrum evaluation of the RPSCTC scheme.
iv. Dual-Repeat-Punctured Turbo Codes (DRPTC), a scheme that improves the existing performance of Repeat-Punctured Turbo Codes (RPTC).
v. A method to evaluate the performance of DRPTC.
vi. A distance spectrum evaluation of the DRPTC coding scheme.
vii. Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes (DRPSCTC), a scheme that improves the performance even further in AWGN channels.
viii. A method to evaluate the performance of DRPSCTC in AWGN channels.
ix. A distance spectrum evaluation of the DRPSCTC scheme.

1.7 Overview of Thesis Structure

In Chapter 2, the structure of the conventional turbo encoder and decoder, and the application of the algorithm to the transmitter and receiver sections of a communication system, is presented. Simulation results and the analytical performance evaluation are presented and explained.

Chapter 3 explains the concept of applying orthogonal signaling, repetition and puncturing, to yield superior performances, compared to the conventional turbo code performances. Simulation results, together with analytical bounds, are presented and explained for Repeat-Punctured Turbo Codes and Superorthogonal Convolutional Turbo Codes.
In Chapter 4, three new derivative schemes are introduced, viz. Repeat-Punctured Superorthogonal Convolutional Turbo Codes (RPSCTC), Dual-Repeat-Punctured Turbo Codes (DRPTC) and Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes (DRPSCTC). Simulation results and analytical bounds are presented for each of the schemes and motivated.

Finally, in Chapter 5, the thesis is concluded with a detailed description of what was achieved and points the reader to possible future work.

Chapter 2
Turbo Codes

Turbo codes are currently being utilized in commercial third-generation UMTS/W-CDMA/CDMA2000 (3GPP/3GPP2) base station equipment and wireless receiver portable devices, and provide the best performance in terms of bit-error rate and data throughput [1]. They have also been incorporated into standards used by NASA for deep-space communications (CCSDS) and terrestrial digital video broadcasting (DVB-T) [1].

Turbo codes are a class of error-correction codes used in applications where engineers seek to achieve maximum information transfer over a limited bandwidth communication link, in the presence of information-corrupting noise. Error-correction codes are most commonly used to improve the energy efficiency of wireless communication systems [9]. On the transmitter side of the communication system, after source coding or formatting, channel coding is needed to add redundancy to the data in the form of parity information [1], [9].

At the receiver section of the communication system, an error correction decoder is needed, which will exploit the redundancy in such a way that a reasonable amount of channel errors can be corrected. Coded communications systems can afford to operate with a lower transmit power, since more channel induced errors can be tolerated with, than without an error-correction code [9].

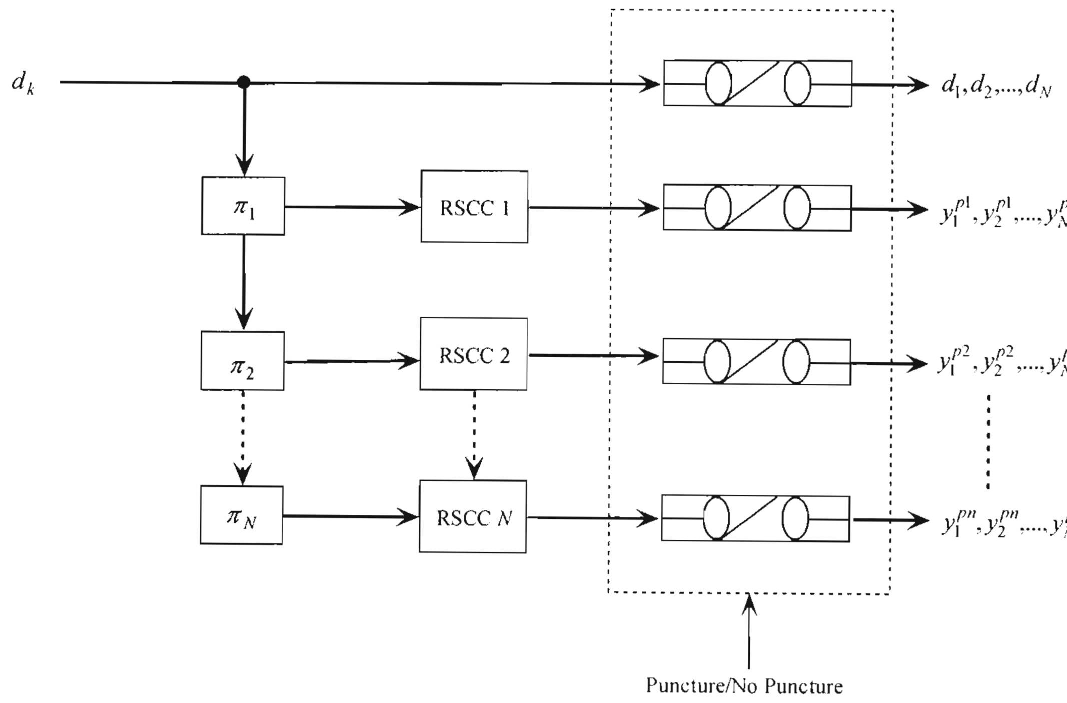
Turbo codes were discovered by Berrou *et. al.* [4], [12], and presented to the coding community in 1993 in their groundbreaking paper, "Near Shannon limit error-correcting coding and decoding: Turbo-codes" [4]. Initially, the results reported were met with much skepticism, but as soon as other researchers began to validate the results, a research effort was soon underway with the aim of explaining and, better yet, improving the astounding performance of turbo codes.

Turbo codes have exhibited performances a few tenths of a decibel from Shannon's theoretical limit. For example, for a frame length of $N = 16384$ and with iterative decoding E_b / N_0 values of $-0.15\,\text{dB}$ at a bit-error rate level of 10^{-3} were reported [1-7],

[9], [12]. Additionally, the use of turbo codes also makes it possible to increase the data rate, without the need to increase transmission power, or alternatively can be used to lower the amount of power needed to transmit at a specific data rate.

2.1 Encoding with Turbo Codes

Turbo coding consists of the parallel concatenation of infinite impulse response (IIR) or recursive systematic constituent codes. The general structure of the turbo code encoder is shown below in Figure 2.1A.



d_k
π_1
RSCC 1
π_2
RSCC 2
π_N
RSCC N
$d_1, d_2, ..., d_N$
$y_1^{p1}, y_2^{p1}, ..., y_N^{p1}$
$y_1^{p2}, y_2^{p2}, ..., y_N^{p2}$
$y_1^{pn}, y_2^{pn}, ..., y_N^{pn}$
Puncture/No Puncture

Figure 2.1A Generic structure of the turbo encoder

The encoder has a systematic output branch, and up to n parity output branches. However, two parity outputs are most commonly used in turbo encoding. The typical structure of the turbo encoder is shown in Figure 2.1B. Each of the parity branches constitutes an interleaver or permuter, and a recursive systematic convolutional (RSC) encoder. These RSC encoders are component encoders of the overall turbo encoder. Furthermore, a puncturing mechanism can be used at the output to control the code rate. Puncturing serves to increase the code rate, but with a drop in performance. However, more extravagant puncturing patterns could be used to render a minimal drop in performance.

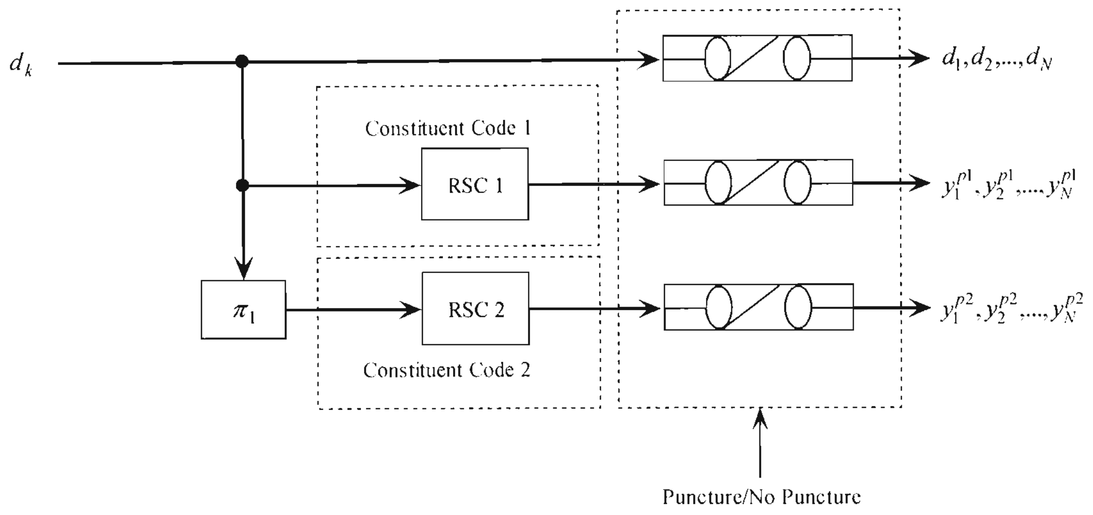


Figure 2.1B Typical structure of the turbo encoder

2.1.1 Encoding with Recursive Systematic Codes

Turbo coding involves the parallel concatenation of component recursive systematic convolutional codes [1-4]. Let us first consider a nonsystematic convolutional (NSC) code. Figure 2.1.1A depicts the structure of the NSC code. This is a simple binary rate ½ convolutional encoder, with constraint length $K = 3$ and memory $m = K - 1 = 2$. At time k a bit d_k is sent to the encoder. The corresponding output is the bit pair u_k, v_k and is defined by the modulo-2 equations (2.1.1a) and (2.1.1b) [1].

$$u_k = \sum_{i=0}^{K-1} g_{1i} d_{k-i} \qquad \text{where } g_{1i} \in \{0,1\} \qquad (2.1.1a)$$

$$v_k = \sum_{i=0}^{K-1} g_{2i} d_{k-i} \qquad \text{where } g_{2i} \in \{0,1\} \qquad (2.1.1b)$$

The code generators are given by $G_1 = \{g_{1i}\} = \{111\}$ and $G_2 = \{g_{2i}\} = \{101\}$, i.e., $\{7,5\}$ in octal. From the structure of the NSC, it can be seen that the response would be that of a discrete-time finite impulse response (FIR) linear system. At large signal-to-noise ratios the performance of NSC encoders is better than a systematic code, and the converse would be true for small SNRs [1].

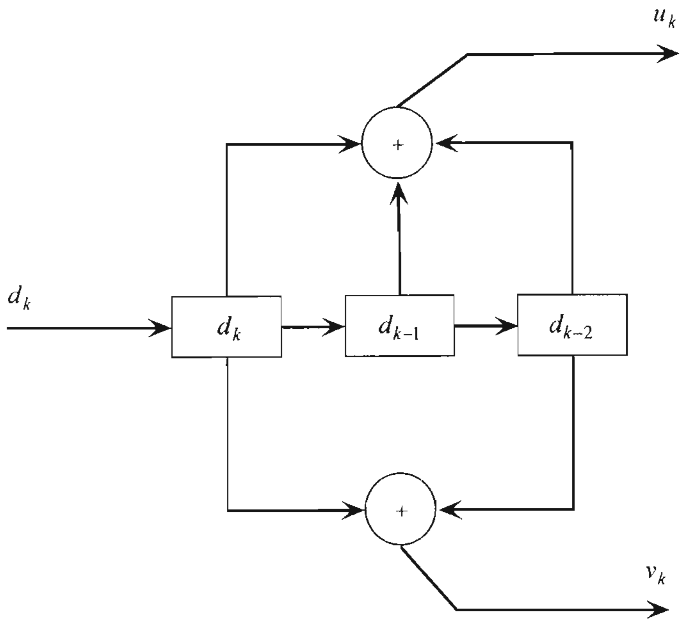


Figure 2.1.1A Typical structure of a nonsystematic convolutional encoder [1]

For the conventional turbo encoder a class of infinite impulse response (IIR) constituent codes had been proposed by Berrou *et. al.* [4]. These constituent codes can also be referred to as recursive systematic convolutional (RSC) codes, since previously encoded information bits are being continually fed back to the input of the encoder. RSC codes also result in superior error performance than the best NSC codes at all values of SNR [1]. The RSC code can be derived from the NSC code simply by creating a feedback loop and allowing one of the two outputs branches to equal to d_k. The structure of a RSC code is shown in Figure 2.1.1B. Here the memory depth is increased to $m = 3$. Two feedback loops have been introduced to create the infinite impulse response and a systematic output branch is introduced. a_k can be recursively calculated from the modulo-2 equation (2.1.1c) [1],

$$a_k = d_k + \sum_{i=1}^{K-1} g_i' a_{k-i} \qquad (2.1.1c)$$

where g_i' is the new generator polynomial.

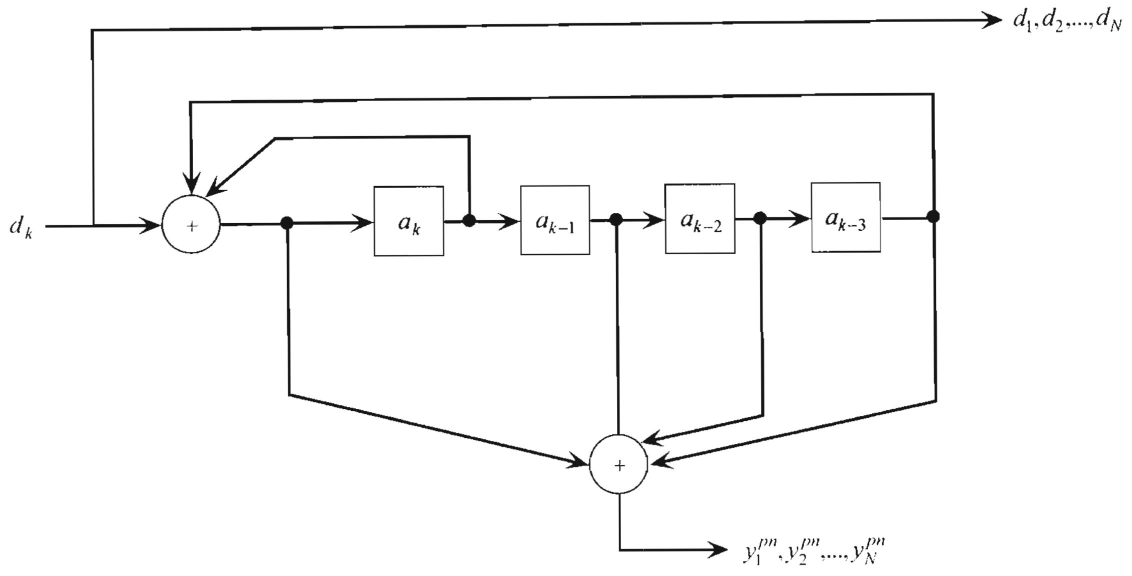


Figure 2.1.1B Structure of a recursive systematic convolutional encoder [1]

If identical code generators were used, then the output codewords weight distribution from the RSC encoder will not differ, in comparison to the weight distribution of the NSC encoder [1]. The only change that occurs is the mapping between input information sequences and output codeword sequences.

2.1.2 Interleaving for Turbo Codes

Interleaving is the process of scrambling or permuting the information symbols prior to it being sent as input to the second component encoder. The primary function of the interleaver, π, is in achieving good codeword distance properties, which can not be accomplished by interleaving alone, but with an accompanying recursive constituent encoder. The objective is to achieve at least one codeword of high Hamming weight. The interleaver decreases the probability of both constituent encoders producing low weight codewords at their outputs simultaneously. This is referred to as spectral thinning.

Associated with the interleaver is the deinterleaver π^{-1}, which applies the inverse permutation to reconstruct the original sequence. The two major criteria in the design of the interleaver is the weight spectrum of the code, and the correlation between the input information data and the soft-decision output of each decoder corresponding to its parity bits [8]. The interleaver size determines the tradeoff between time delay and performance, since both these factors are proportional to size. The performance of the code is determined by the interleaver mapping that is chosen. Interleaving is applied to most applications for spreading burst errors. However, in application to turbo coding, it has many more functions. Firstly, it is used to inject the constituent encoder with permutations so that the generated redundancy sequences can be assumed independent, which is a function of the particular interleaver used. This automatically excludes interleavers that generate regular sequences, for example cyclic shifts. In addition, a turbo code, unlike convolutional codes, gives the weight distribution higher priority than the minimum distance, so the other critical function of the interleaver would be that of spectral thinning [6], [7], as pointed out earlier. There are two kinds of commonly used interleavers for turbo codes: uniform or random interleaving, and the more recent two step S-random interleaver [8]. [8] shows how S-random interleavers can be designed for use in turbo codes and also presents the simulation results, showing the performance improvement over uniform interleavers.

Typically, the input sequence $d = \{d_1, d_{2,\dots,} d_N\}$ and P the interleaver mapping or matrix, yields the interleaved sequence dP. The deinterleaving matrix would then be the transpose of the interleaving matrix P^{T}, where T represents the transpose operation.

A random or uniform interleaver is simply a random permutation. This type of interleaver performs well for fairly large frame lengths, however, for very small frame lengths the performance degrades down to a point where convolutional codes become more suitable for the same computational complexity [8]. The research presented in this thesis was simulated using the random interleaver.

2.1.3 Puncturing

Puncturing is the process used to remove specifically chosen bits from the error-correction coded output sequence, and is used in many common communication systems, for example, UMTS for rate matching and the 802.11a PHY standard to achieve the $3/4$ rate. This is tantamount to encoding with an error-correction code of a higher rate or lower redundancy. Puncturing considerably increases the flexibility of the system without significantly increasing its complexity, since the same decoder can be utilized regardless of the number of bits punctured prior to transmission [1], [9]. The main issue to consider when puncturing turbo codes is to minimize the puncturing of systematic bits. Since these bits are more important than parity bits and puncturing a systematic bit leads to more performance degradation than puncturing a parity bit. The second issue is to prevent the puncturing of termination bits. Termination bits or tail bits are used to ensure that the ending states of the encoder and decoder are zero state. Puncturing termination bits is tantamount to non-termination of the turbo code, and can result in serious performance degradation as pointed out in [1].

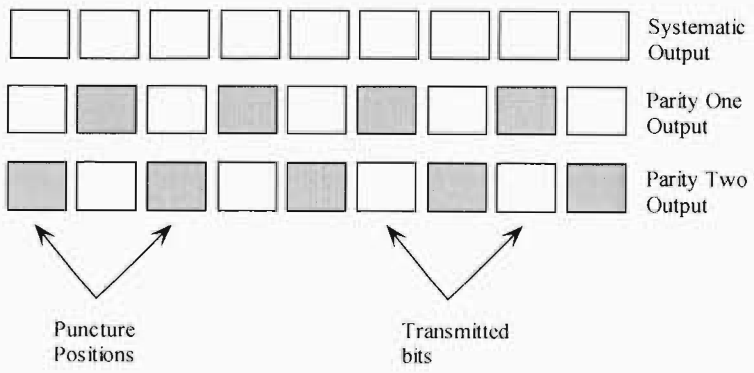


Figure 2.1.3A Puncturing to achieve $R = 1/2$ from $R = 1/3$ Turbo Code

Another important factor to consider during the puncturing of turbo codes is to provide approximately equal puncturing of parity bits at the output of the two constituent encoders. This factor determines the effectiveness of the error correction capability of the punctured sequences. Figure 2.1.3A illustrates how the puncturing of turbo codes is undertaken prior to transmission. The tail bits required for returning the encoder to zero state are not included in the illustration. The systematic output represented by the top layer of blocks is left un-punctured. Now to increase the rate to $R = \frac{1}{2}$ the first parity sequence is punctured at 4 positions leaving 5 bits, and the second parity sequence is punctured at 5 positions leaving 4 untouched bits. At the decoder section, the punctured bits, represented by the dark-shaded blocks, need to be replaced by dummy bits '1'. This process is referred to as reconstruction of the parity sequences. The constituent decoders require input sequences of a specific length for processing.

2.1.4 Trellis Termination

Termination of the trellises for turbo codes requires a different technique, compared to the conventional method of padding with m zeros to flush a convolutional encoder to the all-zero state [1]. This is due to the recursive nature of the turbo code component encoders. The tail bits required depend on the state of the encoder after encoding N message bits. This makes it impossible for the same tail bits to be utilized for flushing the encoders.

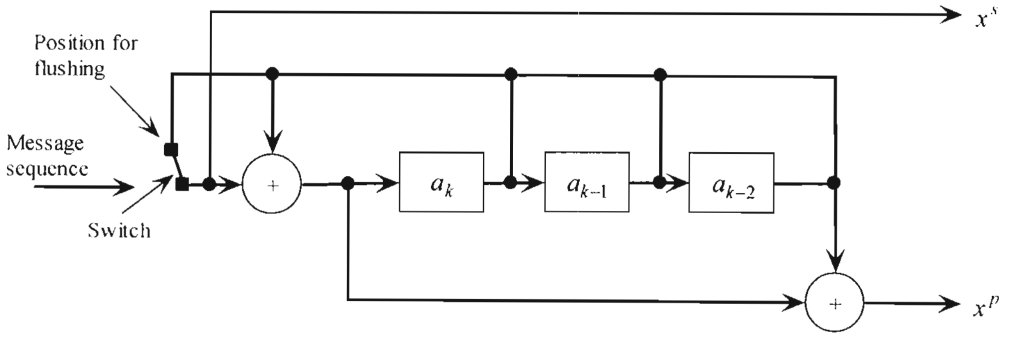


Figure 2.1.4A Method used for terminating trellis

Figure 2.1.4A illustrates a simple method for trellis termination for turbo codes [3], [5], [15]. The switch at the input first allows the message sequence to pass through for N message bits. After the message block is sent through the switch moves to the position indicated in the diagram, and allows the feedback bits to flush the memory elements to the all-zero state. The decoder is then at liability to assume initiation from the all-zero state.

2.2 Decoding Turbo Codes

The structure of the turbo decoder is shown in Figure 2.2A below. Figure 2.2B shows the convention that we use for the sequences that are transmitted after turbo encoding and the sequences that are received prior to decoding. The systematic sequence x_k^s becomes the corrupted systematic output y_k^s. Similarly the first and second parity sequences x_k^{p1} and x_k^{p2} become the corrupted parity sequence one y_k^{p1} and two y_k^{p2}, respectively.

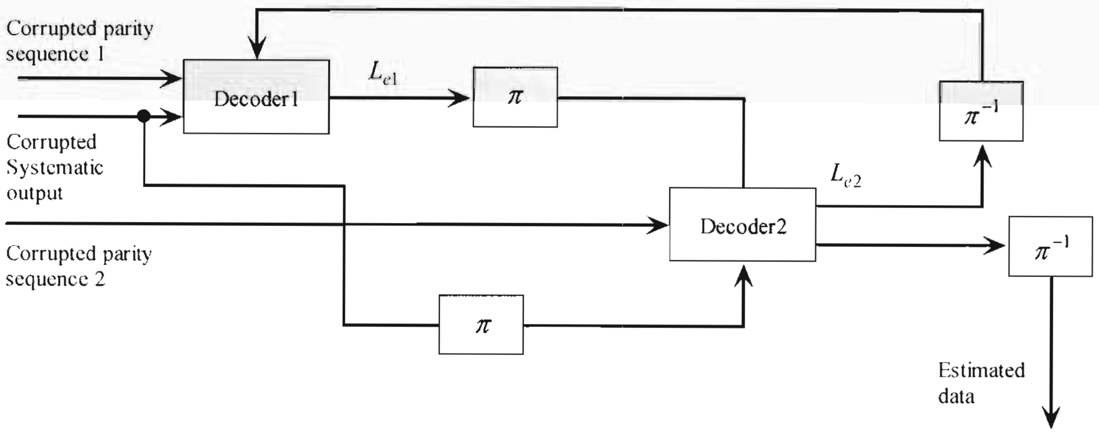

Corrupted parity sequence 1
Decoder1
L_{e1}
π
π^{-1}
Corrupted Systematic output
L_{e2}
Decoder2
π^{-1}
Corrupted parity sequence 2
π
Estimated data

Figure 2.2A Structure of the Turbo decoder

The turbo decoder consists of two component decoders to handle the two encoders that were utilized at the transmitter section. In addition, interleavers π and deinterleavers π^{-1} are used. The interleaver mapping or matrix is the same as that which was used at the turbo encoder. In the first iteration, Decoder 1 has two inputs, viz. the corrupted parity sequence one y_k^{p1} and the corrupted systematic output y_k^s. Decoder 1 then uses the maximum a-posteriori (MAP) algorithm to produce soft decisions $L(d_k)$.

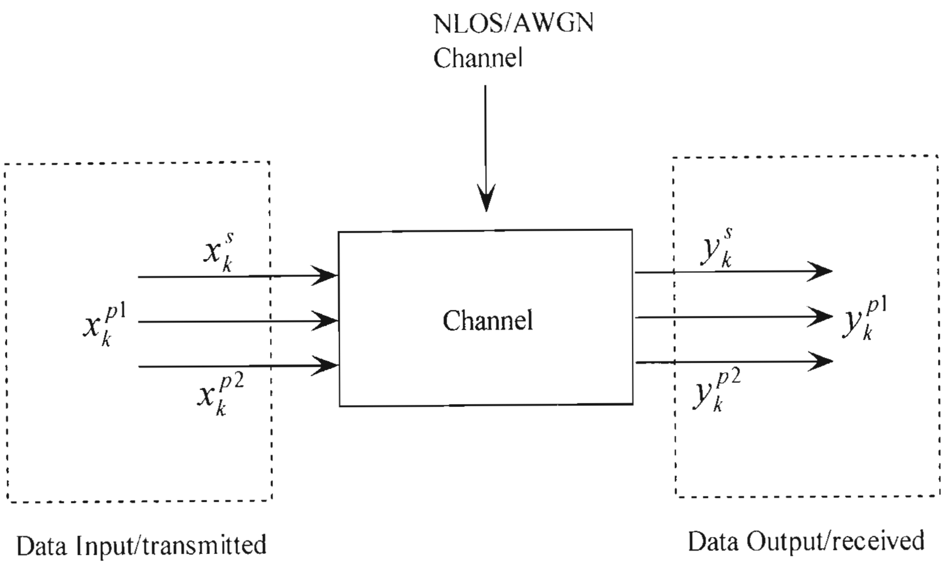


Figure 2.2B Convention for transmitted and received sequences

These soft decisions are then converted into soft extrinsic decisions taking into account the channel measurement, either AWGN or non-line of sight (NLOS) channels and a-priori information, if any. These extrinsic decisions L_{e1} are then interleaved by the encoder mapping and sent to Decoder two as a-priori information. Decoder 2 has three inputs, viz. the corrupted parity sequence two y_k^{p2}, the interleaved corrupted systematic output, i.e., $y_k'^s = \pi(y_k^s)$, and the a-priori information supplied by Decoder one's extrinsic soft decisions L_{e2}. Decoder 2 in turn produces soft decisions which are converted into soft extrinsic information. This information is then deinterleaved $\pi^{-1}(L_{e2})$ and sent back to Decoder one as assist information. In the second iteration Decoder 1 has three inputs, viz. y_k^{p1}, y_k^s and the a-priori information from Decoder two. After a set number of iterations, the output from Decoder two, i.e., the soft decisions, are deinterleaved, then converted into hard decisions to yield the estimate of the original message sequence d_k. The main features of the turbo decoder can be summarized as follows:

- Serial concatenation of component decoders
- Use of the MAP algorithm
- Extrinsic information gleaned from decoding process
- Iterative decoding – turbo decoding

The MAP algorithm also known as the modified BCJR algorithm which, is a prerequisite to the extrinsic information and the principle of iterative decoding, and will be presented in the next subsection with the BCJR algorithm presented in [10].

2.2.1 The MAP Algorithm

The Viterbi algorithm is a well-known technique employed in the decoding of convolutional codes, and it is optimal in the sense of minimizing the probability of word

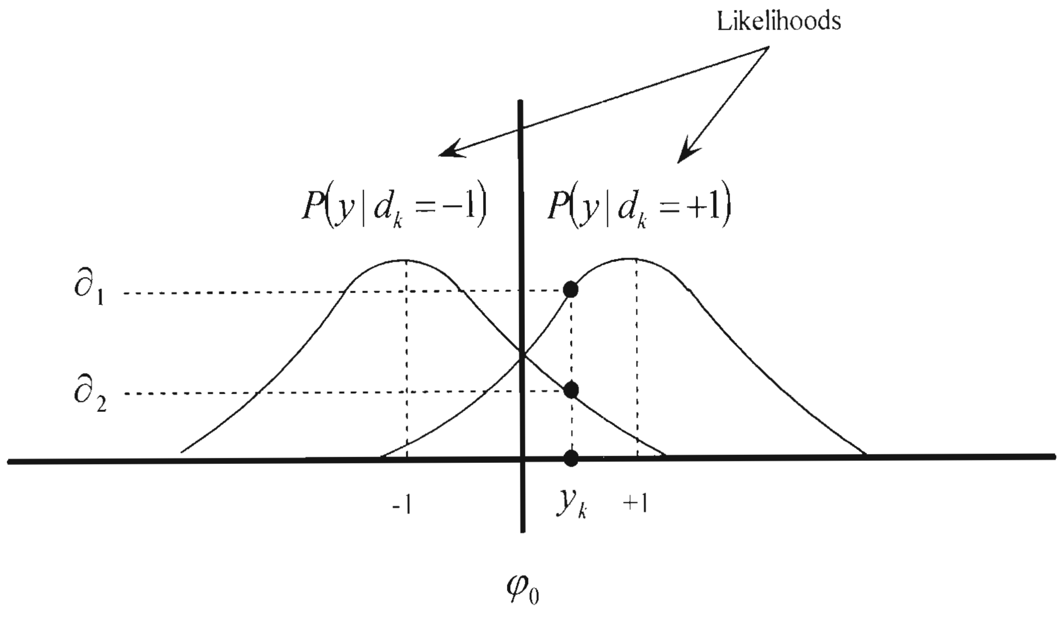


Figure 2.2.1A The two-signal class case

error, but it is unable to yield a-priori probabilities (APP) for every bit that is decoded [1], [11]. In 1974, the BCJR algorithm was proposed by Bahl *et. al.* which works in a forward-backward recursive manner and minimizes the bit-error probability [10], [11]. In 1993, Berrou, Glavieux and Thitimajshima used a slightly modified BCJR algorithm in the decoding of turbo codes, but the forward recursion was unnecessarily complicated [1], [11], [12]. In 1996, Berrou and Glavieux proposed a modified version of the BCJR algorithm, which promised to reduce a considerable amount of computational complexity

[12]. A summarized version of the MAP algorithm will be presented in this section, with a more detailed version presented in Appendix A.

Utilizing Bayes theorem and a-priori probabilities,

$$P(d_k = i \mid y) = \frac{P(y \mid d_k = i)P(d_k = i)}{P(y)} \quad \text{for } i = 1, \ldots, M \tag{2.2.1a}$$

where
$$P(y) = \sum_{i=1}^{M} P(y \mid d_k = i)P(d_k = i) \tag{2.2.1b}$$

$P(d_k = i \mid y)$ is the APP and $d_k = i$ represents data belonging to the i^{th} signal class from a set of M classes.
$P(y \mid d_k = i)$ is the pdf for a data plus noise continuous valued signal (cvs) conditioned in signal class $d_k = i$, and $P(d_k = i)$ is the a-priori probability.

For an AWGN channel consider the conditional probability density functions (pdf) or likelihood functions as shown in Figure 2.2.1A. The function on the right of the graph represents the pdf of the random variable y conditioned on $d_k = +1$ being transmitted, and the pdf on the left similarly represents the random variable y conditioned on $d_k = -1$ being transmitted. The line drawn from y_k on the abscissa axis intersects both functions at positions ∂_1 and ∂_2. By using the maximum likelihood decision rule, one would choose the larger of the two intercepts to determine if a '+1' or a '-1' had been transmitted. The MAP algorithm is a similar decision rule. In this decision rule, equations (2.2.1c) and (2.2.1d), a-posteriori probabilities are taken into account.

$$P(d_k = +1 \mid y) > P(d_k = -1 \mid y) \text{ then } d_k = +1 \tag{2.2.1c}$$

$$P(d_k = +1 \mid y) < P(d_k = -1 \mid y) \text{ then } d_k = -1 \tag{2.2.1d}$$

If we take the logarithm of the conditions above this would represent a useful metric called the log-likelihood ratio (LLR). Equation (2.2.1e) starts to define the LLR used for the decoding of turbo codes using APPs.

$$L(d_k \mid y) = \log\left[\frac{P(d_k = +1 \mid y)}{P(d_k = -1 \mid y)}\right] \qquad (2.2.1e)$$

$$= \log\left[\frac{P(y \mid d_k = +1)P(d_k = 1)}{P(y \mid d_k = -1)P(d_k = -1)}\right] \qquad (2.2.1f)$$

Now equation (2.2.1f) can be separated into two terms, the likelihood probability ratio, and the a-priori probability ratio.

$$L(d_k \mid y) = \log\left[\frac{P(y \mid d_k = +1)}{P(y \mid d_k = -1)}\right] + \log\left[\frac{P(d_k = +1)}{P(d_k = -1)}\right] \qquad (2.2.1g)$$

(2.2.1g) can be written in a simpler form by

$$L(d_k \mid y) = L(y \mid d_k) + L(d_k) \qquad (2.2.1h)$$

and

$$L(d_k \mid y) = L_{channel}(y) + L(d_k) \qquad (2.2.1i)$$

where $L_{channel}$ represents the channel measurement and is dependent on the channel model being investigated, and $L(d_k)$ represents the a-priori probabilities.

Now a term for the extrinsic information from the previous decoder needs to be added into (2.2.1i) to yield the final expression for the soft LLR values [1].

$$L(d_k \mid y) = L_{channel}(y) + L(d_k) + L_e(\hat{d}) \qquad (2.2.1j)$$

The extrinsic information L_e represents information gleaned from the decoding process and forms the a-priori information for the next decoder within the iteration.

Since, in turbo decoding, information is continuously being passed between the decoders, the log-likelihood ratio would serve useful as a soft-decision metric. It has been reported in [1] that using hard decisions as an input for a decoder can degrade performance up to 0.7 dB.

Referring to Appendix A.1, the LLR can be further defined in terms of likelihood ratios, λ, as

$$L(d_k \mid y) = \log\left[\frac{\sum_m \lambda_k^{i=1}(v)}{\sum_m \lambda_k^{i=0}(v)}\right], \qquad \forall\ k = [1,\ldots,N] \tag{2.2.1k}$$

with the likelihood ratio defined by

$$\lambda_k^i(v) = \sum_{m'} \delta_i(y_k, v', v)\alpha_{k-1}(v')\beta_k(v) \tag{2.2.1l}$$

where $\delta_i(y_k, v', v)$ is the branch metric between states v and v', $\alpha_{k-1}(v')$ is the forward state metric (future state) and $\beta_k(v)$ is the reverse/backwards state metric (past state). Substituting (2.2.1l) into (2.2.1k), then the soft LLR is given by

$$L(d_k \mid y) = \log\left[\frac{\sum_v \sum_{v'} \delta_1(y_k, v', v)\beta_k(v)\alpha_{k-1}(v')}{\sum_v \sum_{v'} \delta_0(y_k, v', v)\beta_k(v)\alpha_{k-1}(v')}\right] \tag{2.2.1m}$$

2.2.1.1 The forward state metric

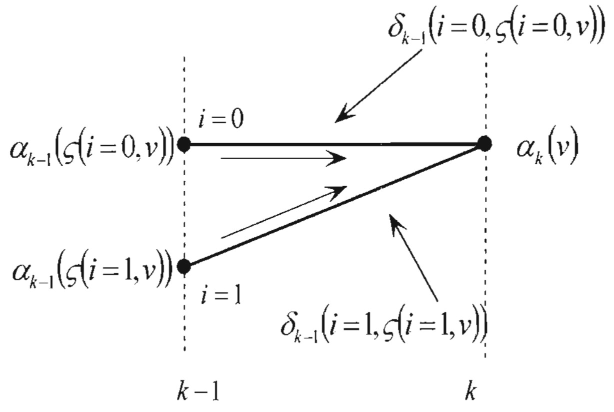
In order to apply equation (2.2.1m), from Section 2.2.1, to the decoding of the turbo code the forward state metric $\alpha_{k-1}(v')$ needs to be computed. During the calculation of the forward state metric the trellis is traversed in the forward direction.
Figure 2.2.1.1A graphically shows the use of equation (2.2.1.1a) to calculate the forward state metric.

$$\alpha_k(v) = \sum_{i=0}^{1} \alpha_{k-1}(\varsigma(i,v))\delta_{k-1}(i, \varsigma(i,v)) \tag{2.2.1.1a}$$

$\alpha_k(v)$ at trellis depth k is the summation of the product of the previous forward state metrics and its branch metrics at trellis depth $k-1$. Since the encoder was initiated on the all-zero state, the following boundary conditions for the trellis can be assumed when decoding is undertaken.

$$\alpha_{k-1}(\varsigma(i, v=00)) = 1.0,$$
$$\alpha_{k-1}(\varsigma(i, v=01)) = 0,$$
$$\alpha_{k-1}(\varsigma(i, v=10)) = 0,$$
$$\alpha_{k-1}(\varsigma(i, v=11)) = 0.$$

Assuming, that a code with memory depth $m = 2$ was implemented, resulting in four states $\{00,01,10,11\}$. A more detailed presentation of the forward state metric can be obtained in Appendix A.2.



$\delta_{k-1}(i=0, \varsigma(i=0, v))$
$i=0$
$\alpha_{k-1}(\varsigma(i=0, v))$
$\alpha_k(v)$
$\alpha_{k-1}(\varsigma(i=1, v))$
$i=1$
$\delta_{k-1}(i=1, \varsigma(i=1, v))$
$k-1$
k

Figure 2.2.1.1A Calculation of the forward state metric

2.2.1.2 The reverse state metric

Equation (2.2.1m), from Section 2.2.1, requires that the reverse state metric $\beta_k(v)$ be calculated from the trellis. During the computation of the reverse/backward state metric the trellis is traversed in the backward direction.

Figure 2.2.1.2A shows the use of equation (2.2.1.2a) graphically for calculation of the reverse state metric.

$$\beta_k(v) = \sum_{i=0}^{1} \delta_k(i,v)\beta_{k+1}(\phi(i,v)) \qquad (2.2.1.2a)$$

$\beta_k(v)$ at trellis depth k is computed by the summation of the product of the future state metrics and its branch metrics at trellis depth $k+1$. Since, the first encoder was flushed to the all-zero state after encoding, by the tail bits, the following boundary conditions can be assumed during decoding for the first decoder.

$$\beta_{k+1}(\phi(i, v=00)) = 1.0,$$
$$\beta_{k+1}(\phi(i, v=01)) = 0,$$
$$\beta_{k+1}(\phi(i, v=10)) = 0,$$
$$\beta_{k+1}(\phi(i, v=11)) = 0.$$

Once again assuming, that a code with memory depth $m = 2$ was implemented, this results in four states $\{00,01,10,11\}$. It is important to note that due to the interleaver, the end state of the second constituent encoder cannot be assumed as the all-zero state. Thus the boundary conditions presented above do not hold for Decoder 2.

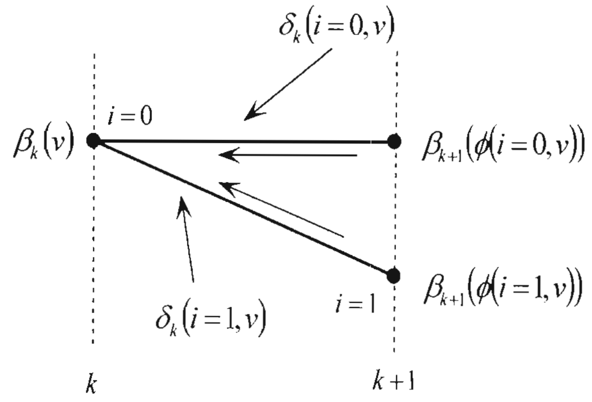


Figure 2.2.1.2A Calculation of the reverse state metric

In this scenario the boundary conditions are given by

$$\beta_{k+1}(\phi(i, v = 00,01,10,11)) = \frac{1}{2^m} \tag{2.2.1.2b}$$

for Decoder 2 for each of the four states $\{00,01,10,11\}$.

A more detailed presentation of the reverse state metric can be obtained in Appendix A.3.

2.2.1.3 The branch metric

The branch metric, as shown in Appendix A.4, is defined by

$$\delta_k(i,v) = A_k \pi_k^i \exp\left[\frac{1}{\sigma^2}\left(p_k a_k^i + q_k b_k^{i,v}\right)\right] \tag{2.2.1.3a}$$

where A_k is a constant, and can be assumed as unity, and π_k is known as the a-priori probability ratio, i.e., $\pi_k = \frac{\pi_k^{i=1}}{\pi_k^{i=0}}$.

The branch metric is a function of the bit pairs transmitted a_k and b_k, and the bit pairs received p_k and q_k.

For the AWGN channel the channel measurement $L_{channel}$ is defined by the logarithm of the ratio of likelihood functions

$$L_{channel} = \log\left(\frac{P(y_k \mid d_k = +1)}{P(y_k \mid d_k = -1)}\right) \tag{2.2.1.3b}$$

Now substituting into the Gaussian distribution function and simplifying

$$L_{channel} = -0.5\left(\frac{y_k - 1}{\sigma}\right)^2 + 0.5\left(\frac{y_k + 1}{\sigma}\right)^2 = \frac{2}{\sigma^2} y_k \tag{2.2.1.3c}$$

2.2.2 Turbo Codes in Rayleigh Fading Channel

Due to the multiple paths taken by a signal to the receiver, distortion is imminent due to the changes in phase, attenuation and delay suffered by the signal. If there is no direct line-of-sight component, then the fading is usually modeled by a Rayleigh distributed function, otherwise, a Rician model is suitable.

The effect of a fading channel on a transmitted signal is a function of the choice of the signal bandwidth B_w, and the time duration of the signal chosen.

In [29], detailed theory for utilizing the Rayleigh probability distribution to model the NLOS fading in communication systems can be obtained. In [30], the application of the turbo code for a Rayleigh fading channel was presented. Appendix B also presents the application of the turbo code to a NLOS channel in detail.

2.2.3 Iterative Decoding for Turbo Codes

The decoding of the turbo code utilizes an iterative technique. Figure 2.2.3A shows a detailed diagram of the decoding process for turbo codes. Decoder one takes two inputs in the first iteration, viz. the noise corrupted parity sequence one, $y_1^{p1}, y_2^{p1}, ..., y_N^{p1}$ and the corrupted systematic output, $y_1^s, y_2^s, ..., y_N^s$, where N is the message sequence frame length. The third input as shown in Figure 2.2.3A is the a-priori input, this is assumed as zero since the probability of receiving a '+1' or '-1' is the same for the first iteration of decoding. Decoder 1 then uses the MAP algorithm as explained in Section 2.2.1 to produce soft log-likelihood soft decisions $L_1(d_k \mid y)$. Equation (2.2.3a) defines how the soft decisions from Decoder 1 are converted into extrinsic decisions by taking the channel measurement and a-priori information into account.

$$L_{e12} = L_1(d_k \mid y) - L_{channel} - L_a \tag{2.2.3a}$$

For this extrinsic information to be of any use to Decoder 2, the extrinsic information L_{e12} is first interleaved by the same interleaver mapping at the encoder. The resulting sequence L'_{e12} is sent to Decoder 2 as a-priori information and also sent to the subtractor block. Decoder 2 takes three inputs, viz. the second corrupted parity sequence $y_1^{p2}, y_2^{p2}, \ldots, y_N^{p2}$, the interleaved systematic output $y_1'^s, y_2'^s, \ldots, y_N'^s$ and the permuted a-priori information supplied by Decoder 1. Again soft LLR decisions are produced by the second decoder and sent to the subtractor operation block. Using equation (2.2.3b) extrinsic information is produced once again. This extrinsic information is scrambled by the interleaver matrix, and is thus denoted by L'_{e21}. To make L'_{e21} of use to Decoder 1, the information needs to be first deinterleaved then sent to Decoder 1 as a-priori information.

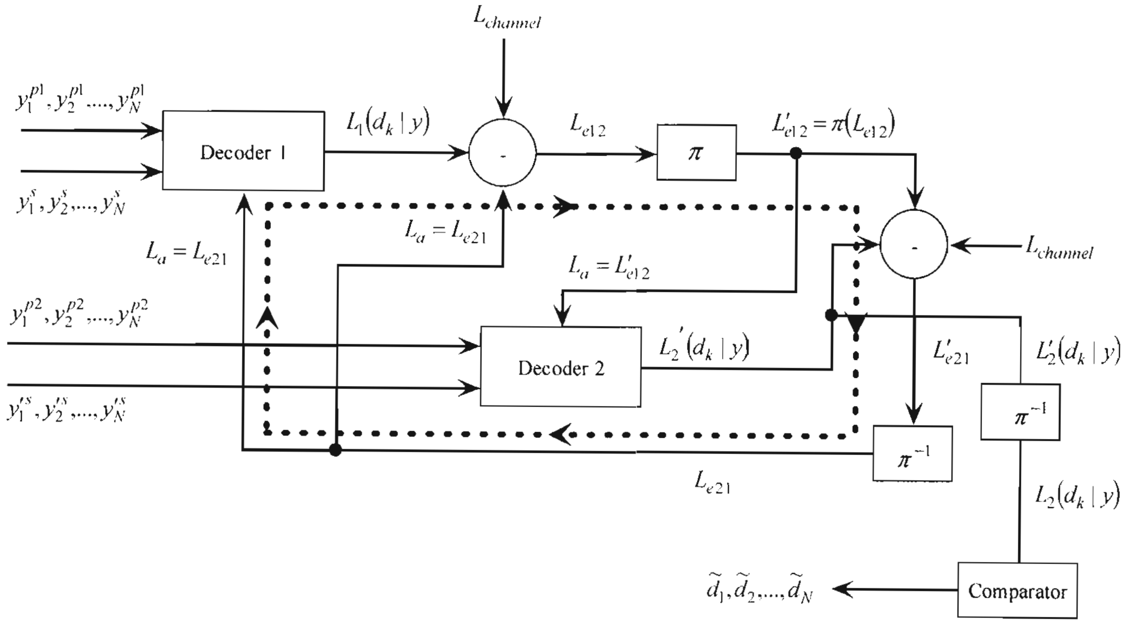


Figure 2.2.3A Iterative decoding for Turbo Codes

$$L'_{e21} = L_2'(d_k \mid y) - L_{channel} - L'_{e12} \tag{2.2.3b}$$

The bold, dotted line in Figure 2.2.3A depicts the path of iterations during the turbo decoding process. After a set amount of iterations, usually 18 iterations yield maximum performance [1], the output of Decoder 2 $L_2'(d_k \mid y)$ is sent to the deinterleaver to unscramble the soft decisions. These resulting soft decisions $L_2(d_k \mid y)$ are then compared to a zero threshold to yield an estimate of the message sequence $\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_N$.

2.2.4 Performance Evaluation and Simulation Results for Turbo Codes

There are two main tools for the performance evaluation of turbo codes: computer simulation and the analytical bounds. Computer simulation generates reliable probability of error estimates as low as 10^{-6} and is useful for rather low SNRs since the error probabilities for larger SNRs are difficult to simulate.
[27] and [28] explain the use of the transfer function bounding technique to evaluate the union bounds for turbo codes. Starting with the state diagram [27], [28], for example consider Figure 2.2.4A, the state transition matrix can be evaluated. The state transition matrix is given by

$$A(L,I,D) = \begin{pmatrix} L^l I^i D^d{}_{0,0} & \cdots & L^l I^i D^d{}_{0,j} \\ \vdots & \ddots & \vdots \\ L^l I^i D^d{}_{z,0} & \cdots & L^l I^i D^d{}_{z,j} \end{pmatrix} \qquad (2.2.4a)$$

where, in a monomial $L^l I^i D^d$, l is equal to 1 for a single bit input, and i and d are either zero or one, depending on the input and output weights respectively for the z^{th} to the j^{th} state of the state diagram.
For an encoder with 2^m states with the last m edges as termination edges, the generating function is given by

$$T(L,I,D) = \sum_{l \geq 0} \sum_{i \geq 0} \sum_{d \geq 0} L^l I^i D^d t(l,i,d) \qquad (2.2.4b)$$

Now since $I + A + A^2 + A^3 + ... = (I - A)^{-1}$. Then the transfer function for each constituent encoder can be expressed in the form of

$$T(L,I,D) = [(I - A(L,I,D))^{-1}]_{0^m,0^m} \qquad (2.2.4c)$$

Denoting the probability of producing a codeword fragment of weight *d* given a randomly selected input sequence of weight *i* by

$$p(d_1 \mid i) = \frac{t(N,i,d_1)}{\sum_{d_1} t(N,i,d_1)} \approx \frac{t(N,i,d_1)}{\binom{N}{i}} \qquad (2.2.4d)$$

for constituent encoder 1, and

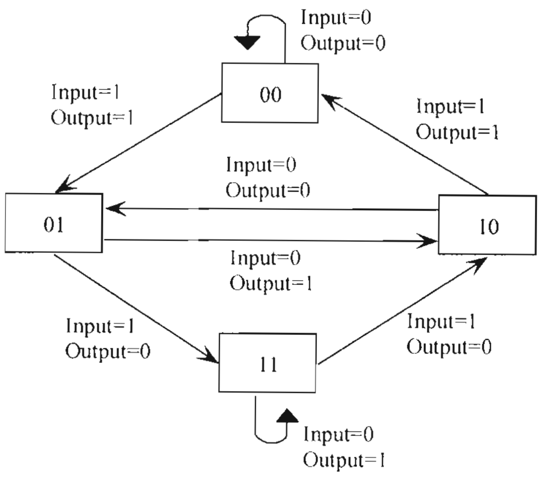


Figure 2.2.4A State Diagram for a {111},{101} code

$$p(d_2 \mid i) = \frac{t(N,i,d_2)}{\binom{N}{i}} \qquad (2.2.4e)$$

for the second component encoder, where $t(N,i,d_q)$ is the input-output weight enumerator. Next the union bound can be obtained from

$$p_{bit} \le \sum_{d=d_{min}}^{n} \sum_{i} \sum_{d1} \sum_{d2} \frac{i}{N} \binom{N}{i} p(d_1 \mid i) p(d_2 \mid i) p_2(d) \qquad (2.2.4f)$$

where the two-codeword probability is denoted by

$$p_2(d) \le Q(\sqrt{2dRE_b / N_o}) \qquad (2.2.4g)$$

$Q(.)$ is the Q-function and R is the rate of the code, and E_b / N_0 is the signal-to-noise ratio.

The union bound provides an upper bound on the performance of turbo codes with maximum-likelihood decoding averaged over all possible interleavers. From [27], [28] if A_d is the weight enumerator of the code and $p_2(d)$ is the pair-wise error probability between the all-zero codeword and a codeword of weight d, the union bound for an (n,k) block code in terms of the weight enumerator is

$$p_{word} \le \sum_{d=d_{min}}^{n} A_d p_2(d) \qquad (2.2.4h)$$

The only way to calculate A_d is via an exhaustive search involving all possible input sequences. One solution is to calculate an average upper bound by computing an average weight function over all possible interleaving schemes. We define an average weight function as

$$\overline{A}_d = \sum_{i=1}^{k} \binom{k}{i} p(d/i) \qquad (2.2.4i)$$

where $p(d/i)$ is the probability that an interleaving scheme maps an input sequence of weight i to produce a codeword of total weight d and $\binom{k}{i}$ is the number of input frames with weight i. Following this, the number of codewords with Hamming distance d is given by

$$A_d = \sum_{i=1}^{N} \sum_{d_1+d_2=d-i} \frac{A_{i,d_1} A_{i,d_2}}{\binom{N}{i}} \qquad (2.2.4j)$$

where A_{i,d_1} is the conditional weight enumerating function (CWEF) for constituent code 1, A_{i,d_2} is the CWEF for constituent code 2 and d_1 and d_2 refer to the first and second parity sequences respectively.

All simulations were undertaken with the C++ Compiler. Figure 2.2.4B shows simulations for a $\{7,5\}_{octal}$ turbo code for various frame lengths, viz. $N = 200$, $N = 400$ and $N = 800$, for the AWGN channel. The stopping criterion for each of the simulations was 150 frame errors. The effect of increasing the message sequence frame length utilizing a random interleaver is evident in the graph. For $N = 200$ a bit-error rate level of approximately 6×10^{-4} is achieved at a SNR of 1.5 dB. For a frame length of $N = 400$, and at the same signal-to-noise ratio, a BER of approximately 10^{-4} is obtained, and for a frame length of $N = 800$, a BER level of 8.5×10^{-6} is achieved. Also at a BER of 10^{-3} a coding gain of approximately 0.25 dB is achieved.

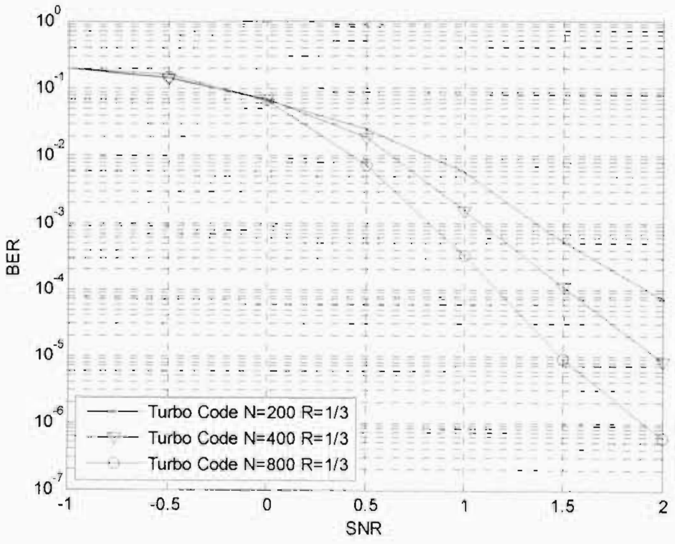


Figure 2.2.4B Simulation for Turbo Codes in the AWGN channel, N=200, 400, 800, R=1/3

Figure 2.2.4C shows the quality of service (QoS) in terms of the BER versus SNR for a turbo code with input frame length, $N = 150$, for the AWGN channel. The union bound for the turbo code is also plotted here. Only one scenario is considered with the bound to show how the simulation converges to the bound. More scenarios are unnecessary in this case since the reader can be referred to other literatures [1-7], [15], [27].

Figure 2.2.4D illustrates the effect of the number of iterations of the decoder, in the AWGN channel, or the number of times extrinsic information is exchanged mutually by the component decoders. The upper plot represents a frame of length $N = 150$ encoded, then decoded with a single iteration. The plot below this one, made up of crosses and dashes, represents seven iterations of the constituent decoders. The solid-cross line plot is simulated for 18 iterations of the decoder. Clearly, as the iterations are increased, the performance curve exhibits a greater gradient. It is important to note that this effect starts to saturate as the iterations are increased to higher values.

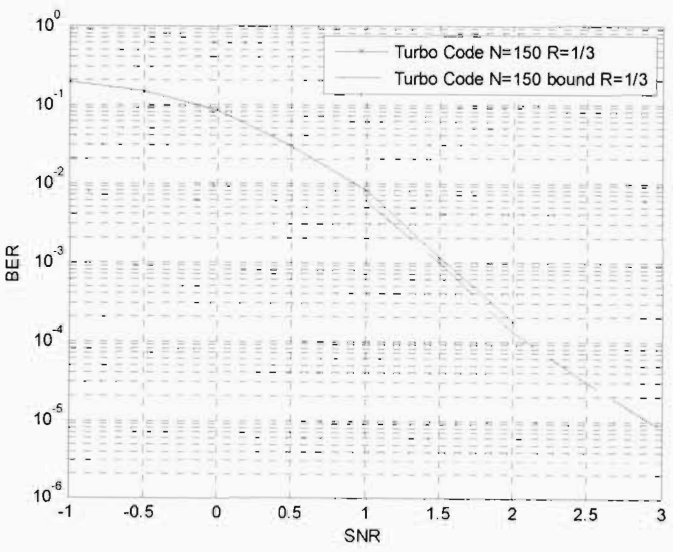


Figure 2.2.4C Simulation for Turbo Codes in the AWGN channel, N=150, R=1/3, together with the union bound

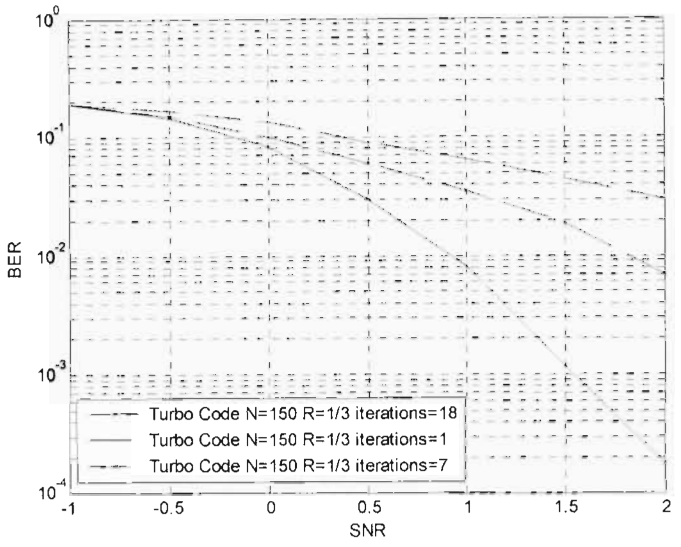


Figure 2.2.4D Simulation for Turbo Codes in the AWGN channel, N=150, R=1/3, for different numbers of iterations, 1, 7, 18

Finally Figure 2.2.4E depicts the codeword distance spectrum for a frame length $N = 128$. The ordinate axis represents the codeword multiplicity A_d, i.e., the number of codewords of weight d, and the abscissa axis represents the Hamming distance d.

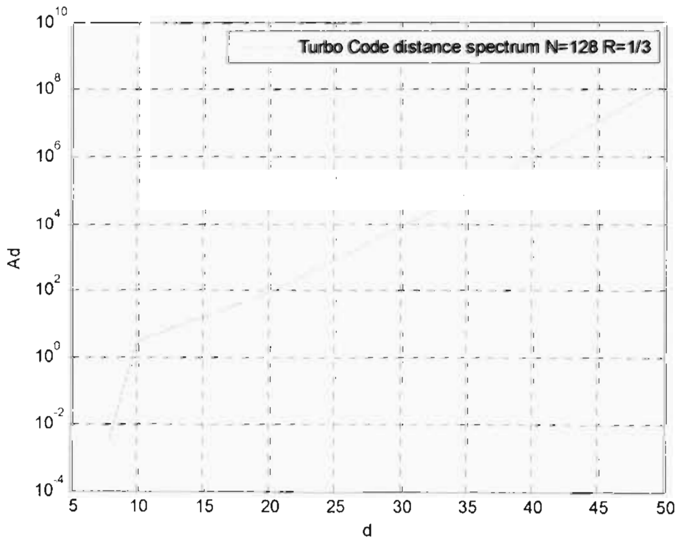


Figure 2.2.4E Distance spectrum for Turbo Codes N=128, R=1/3

From the distance spectrum, one can conclude that a small percentage of codewords are produced with low Hamming distances. This is the effect of spectral thinning, and is primarily responsible for the superior performance of turbo codes.

In this chapter we have reiterated the technique for turbo coding. Simulation results for a rate, $R = 1/3$ code, with frame lengths, $N = 200$, $N = 400$ and $N = 800$ for an AWGN channel was presented. In addition, and very importantly, the effect of increasing the number of decoder iterations was plotted and discussed, and the distance spectrum for a $N = 128$ code was presented.

Chapter 3

Repeat-Punctured Turbo Codes and Superorthogonal Convolutional Turbo Codes

Repeat-Punctured Turbo Codes (RPTC) presented by Kim *et. al.* in [13] show an improvement in the performance of turbo codes for moderate to high SNRs. The exceptional performance achieved with turbo codes is majorly due to the interleaver which is responsible for spectral thinning of the generated codewords [13], [1]. In the conventional turbo codes the frame length is set identical to the interleaver size. [1], [7] and [15] has shown that increasing the frame length leads to an increase in the performance. However this would not be tolerable in all communication systems, e.g. real-time voice communications, since increasing frame length would result in large encoding/decoding delays [13]. Increasing frame length is tantamount to increasing interleaver size in conventional turbo codes. [13] reports on how this factor can be exploited by increasing just the interleaver size to yield better performances with the frame length remaining constant.

Superorthogonal Convolutional Turbo Codes (SCTC) presented by Komulainen and Pehkonen is a low-rate code that combine the principles of low-rate convolutional coding and parallel concatenation, and outperforms the conventional turbo codes due to bandwidth expansion [14], [17], [19]. For the conventional turbo code, using iterative decoding and long interleavers ($N = 16384$) SNR values of $-0.15\,\mathrm{dB}$ have been reported at a BER level of 10^{-3} [15]. For the same BER level and an interleaver length of $N = 4000$, SNR values of $-0.5\,\mathrm{dB}$ for SCTC have been reported in [14], [17], [19]. Superorthogonal convolutional turbo codes make use of orthogonal signaling, together with parallel concatenation and iterative decoding, to yield better performances with an expense in bandwidth expansion. The resultant code is a low-rate code suitable for spread-spectrum modulation (SSM) applications.

3.1 Encoding with Repeat-Punctured Turbo Codes

Repeat-Punctured Turbo Codes make use of repetition and puncturing to improve the performance of the conventional turbo codes. The key ingredient responsible for the spectral thinning of codewords in turbo codes, is the use of a random or uniform interleaver in the encoder [13]. In the classical turbo coding algorithm the frame length is set identical to the interleaver size. The graph shown in Figure 3.1A shows the BER versus $\frac{E_b}{N_0}$ plots for four different frame lengths, viz. L = 1024, 4096, 16384 and $L = 65536$ for a half rate encoder with constraint length $K = 5$ and 18 iterations of the decoder [16]. It is evident that an increase in the frame length leads to an increase in the code performance.

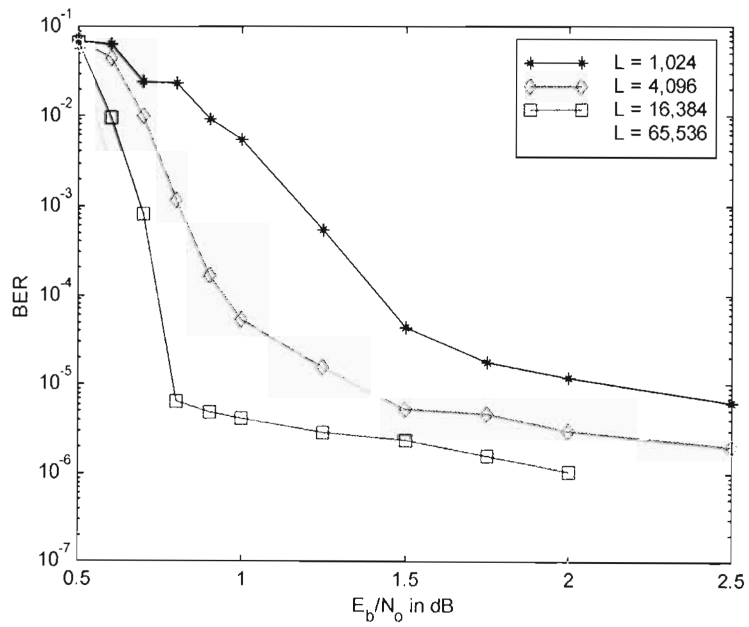


Figure 3.1A Effect of using larger frame/interleaver lengths [16]

A similar performance improvement can be achieved if the interleaver size was increased instead of the frame size, and this would thus result in smaller delays. In [13] Kim *et. al.* reported the use of a repeater-puncturer combination that can be applied to the conventional turbo codes to enable the use of interleavers of sizes larger than the information frame length.

The structure of the encoder used for RPTC is depicted in Figure 3.1B. The information source produces a message frame of length N every T second messaging interval. There are again three parallel outputs for each input frame bit. The first branch is the systematic output $x_1^s, x_2^s, ..., x_N^s$. The first parity sequence $x_1^{p1}, x_2^{p1}, ..., x_N^{p1}$ is produced identically to the conventional turbo code. The message sequence of length N is encoded by a recursive systematic convolutional (RSC) encoder. Prior to interleaving on the third branch of the encoder, the information sequence of length N is repeated L times to yield a sequence of length LN as input to the interleaver. Thus, here we are utilizing a larger interleaver compared to the classical turbo codes. This larger sequence is then encoded by the RSC encoder normally. For a single rate RSC code the output codeword will be of length LN, i.e., $x_1^{p2}, x_2^{p2}, ..., x_{LN}^{p2}$. In order to recover the loss in code rate at the output, a puncturing mechanism needs to be considered. Every l bits are be punctured for every L bits in the codeword sequence. The third parity sequence $x_1^{p2}, x_2^{p2}, ..., x_N^{p2}$ is produced after puncturing, and clearly keeps the overall code rate constant, i.e., $\frac{N}{N+N+N} = \frac{1}{3}$. Puncturing results in the drop of code performance [1], [9] but more extravagant puncturing patterns could be utilized to minimize the performance drop.

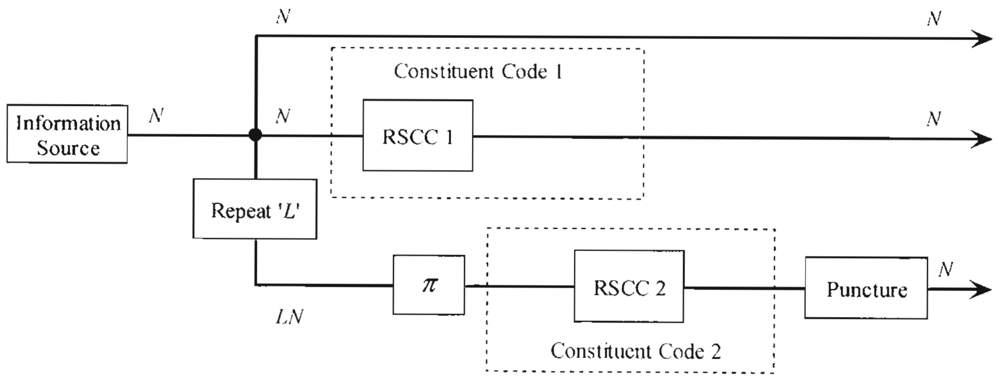
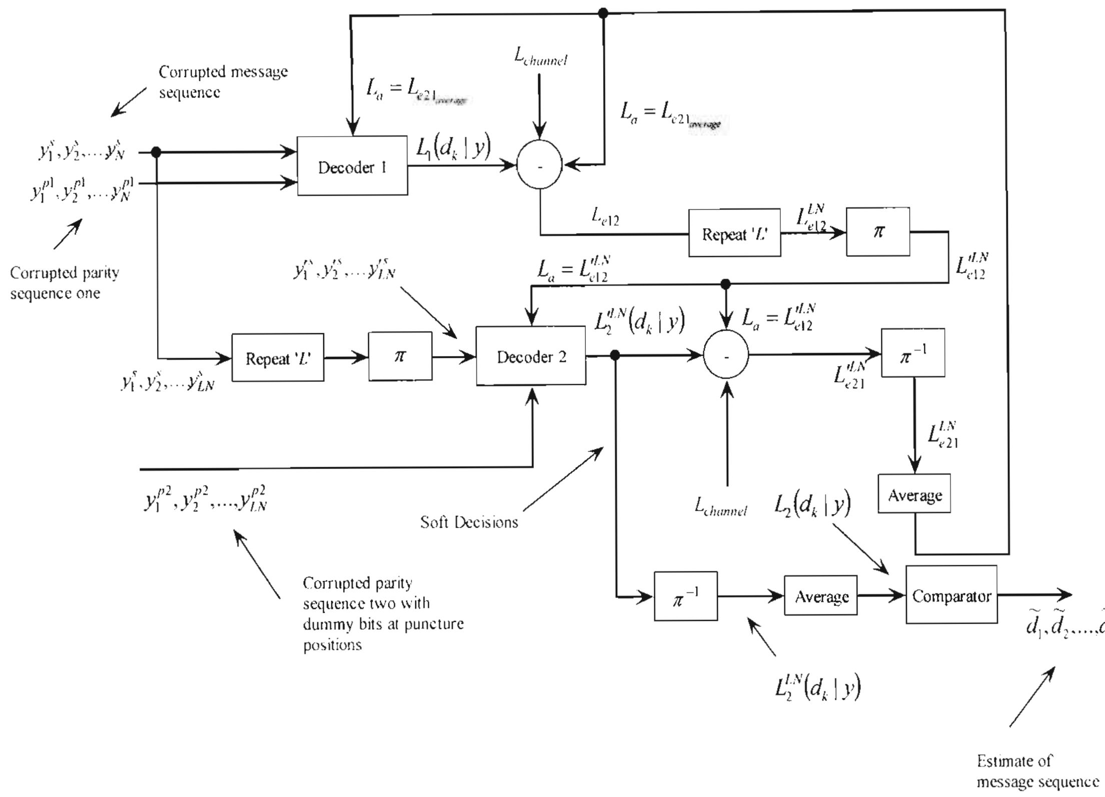


Figure 3.1B Structure of the encoder for Repeat-Punctured Turbo Codes

3.2 Decoding Repeat-Punctured Turbo Codes

The decoding for repeat-punctured turbo codes utilizes an iterative technique similar in the principle to the conventional turbo codes. Figure 3.2A below shows the structure of the RPTC decoder.



Corrupted message sequence
$y_1^s, y_2^s, \dots y_N^s$
$y_1^{p1}, y_2^{p1}, \dots y_N^{p1}$
Corrupted parity sequence one
$L_a = L_{e21_{average}}$
$L_{channel}$
Decoder 1
$L_1(d_k \mid y)$
-
L_{e12}
Repeat 'L'
L_{e12}^{LN}
π
$L_{e12}^{\prime LN}$
$y_1^{\prime s}, y_2^{\prime s}, \dots y_{LN}^{\prime s}$
$L_a = L_{e12}^{\prime LN}$
$L_2^{\prime N}(d_k \mid y)$
Decoder 2
π^{-1}
$L_{e21}^{\prime LN}$
L_{e21}^{LN}
Average
$y_1^s, y_2^s, \dots y_{LN}^s$
$y_1^{p2}, y_2^{p2}, \dots, y_{LN}^{p2}$
Soft Decisions
Corrupted parity sequence two with dummy bits at puncture positions
$L_{channel}$
$L_2(d_k \mid y)$
Comparator
$\tilde{d}_1, \tilde{d}_2, \dots, \tilde{d}_N$
$L_2^{LN}(d_k \mid y)$
Estimate of message sequence

Figure 3.2A Structure of the decoder for Repeat-Punctured Turbo Codes

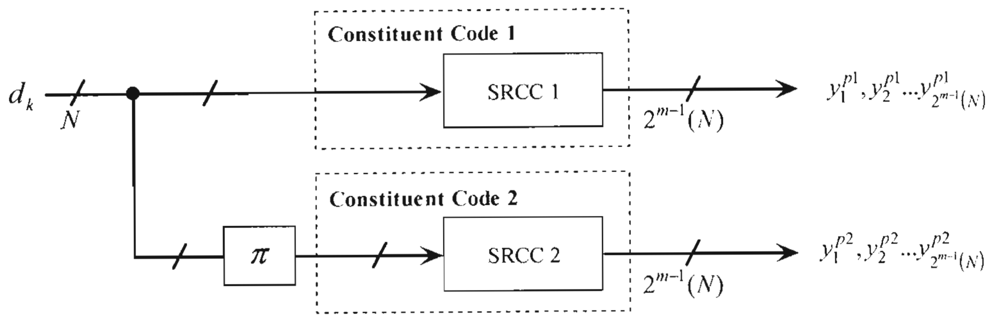
In the first iteration, Decoder one has two inputs, viz. the corrupted message sequence $y_1^s, y_2^s, \dots, y_N^s$ and the corrupted first parity sequence $y_1^{p1}, y_2^{p1}, \dots, y_N^{p1}$. Utilizing these two inputs and zero a-priori information, the decoder uses the MAP algorithm to produce LLR (soft decisions) $L_1(d_k \mid y)$ at its output. These LLRs are then sent to the subtraction block. Both the channel measurement $L_{channel}$ and the a-priori information, which are assumed as zero in first iteration, are subtracted from $L_1(d_k \mid y)$. At the output of the subtractor the extrinsic log-likelihood ratios L_{e12} are produced. This operation follows equation (2.2.2a) in Section 2.2.2. To render this soft extrinsic information of use to Decoder two, (Decoder two is operating on sequences of length LN and the interleaver matrix used at the encoder is of length LN), these extrinsic decisions need to be first repeated L times. The resulting sequence of extrinsic LLRs of length LN is denoted as L_{e12}^{LN}. Prior to supplying the extrinsic LLRs as assist information, the decisions are first interleaved by the same mapping at the encoder. The interleaved sequence $L_{e12}^{\prime LN}$ is then sent as a-priori information to Decoder two. Decoder two has two other inputs. The corrupted message sequence $y_1^s, y_2^s, \ldots, y_N^s$ has to be repeated L times to yield $y_1^s, y_2^s, \ldots, y_{LN}^s$ and then interleaved to yield $y_1^{\prime s}, y_2^{\prime s}, \ldots, y_{LN}^{\prime s}$ and sent to Decoder two. The outstanding input to Decoder two is now the second parity sequence $y_1^{p2}, y_2^{p2}, \ldots, y_{LN}^{p2}$. Remember that prior to transmission, the second parity sequence of length LN was punctured to length N, in order to control the code rate. Therefore, at the decoder, dummy bits need to be introduced into the puncture positions. A '+1' was inserted into the puncture positions. Decoder two then similarly utilizes the MAP algorithm to produce LN permuted soft log-likelihood ratios $L_2^{\prime LN}(d_k \mid y)$. These soft LLRs are sent to the subtractor, and extrinsic LLRs, $L_{e21}^{\prime LN}$ are produced, taking into account the channel measurement and the assist information supplied by Decoder one. Now to make these extrinsic LLRs of use to Decoder one, the LN extrinsic decisions are deinterleaved, $L_{e21}^{LN} = \pi^{-1}\left(L_{e21}^{\prime LN}\right)$, then averaged according to equation (3.2a).

$$L_{e21_{average}} = \frac{L_{e21,i} + L_{e21,i+1}}{L}, \qquad i = 0,1,\ldots,N-1 \tag{3.2a}$$

The averaged N extrinsic LLRs are then sent back to Decoder one as a-priori information for the second iteration. This mutual exchange of information, known as cooperation, is repeated several times before the LN soft interleaved LLRs from Decoder two are deinterleaved and averaged, then converted into hard decisions to yield an estimate of the initial message sequence, $\widetilde{d}_1, \widetilde{d}_2, \ldots, \widetilde{d}_N$.

3.3 Encoding with Superorthogonal Convolutional Turbo Codes

Superorthogonal Convolutional Turbo Codes (SCTC) is a low-rate code with good distance properties suitable for spread-spectrum applications, and makes use of bandwidth expansion, parallel concatenation, iterative decoding and low-rate convolutional encoding, to outperform classical turbo codes in the AWGN channel, and especially fading channels [14], [17], [19]. The basic structure of the SCTC encoder is shown in Figure 3.3A with a more detailed diagram shown in Figure 3.3C.



d_k
N
Constituent Code 1
SRCC 1
$2^{m-1}(N)$
$y_1^{p1}, y_2^{p1} \dots y_{2^{m-1}(N)}^{p1}$
Constituent Code 2
π
SRCC 2
$2^{m-1}(N)$
$y_1^{p2}, y_2^{p2} \dots y_{2^{m-1}(N)}^{p2}$

Figure 3.3A Structure of the encoder for SCTC

Basically, a message sequence of length N is supplied to two constituent superorthogonal recursive convolutional coders (SRCC), SRCC 1 and SRCC 2. These two encoders produce orthogonal sequences for every input bit d_k. The first parity sequence $y_1^{p1}, y_2^{p1}, \dots, y_{2^{m-1}(N)}^{p1}$ and the second parity sequence $y_1^{p2}, y_2^{p2}, \dots, y_{2^{m-1}(N)}^{p2}$ are both of length $2^{m-1}(N)$.

Orthogonal, bi-orthogonal and superorthogonal signals/codes have been discussed in [14], [17], [18]. Orthogonal codes are used extensively in wireless communications, e.g. spreading in DSSS CDMA systems and ARQ systems, with partial retransmission strategies, and are sets of sequences in which the pair-wise cross-correlations should be zero. For an orthogonal convolutional encoder, all of the possible output sequences per trellis branch are orthogonal to each other. Figure 3.3B shows a section of a trellis branch with two states i and j. From the illustration it is evident that four mutually orthogonal sequences are needed for every two states of the trellis branch, if the binary number system is used. This would imply that a total of 2^{m+1} orthogonal sequences are needed per trellis branch for encoder memory m and 2^m encoder states.

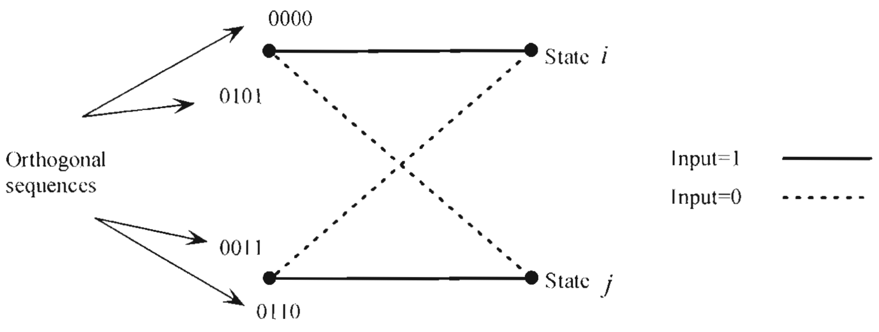


Figure 3.3B Section of trellis for encoding/decoding orthogonal sequences [14]

If the orthogonal signal set is enlarged to include its complementary sequences, then the resulting signal set would be a bi-orthogonal set which has twice as many members for the same sequence length [14].

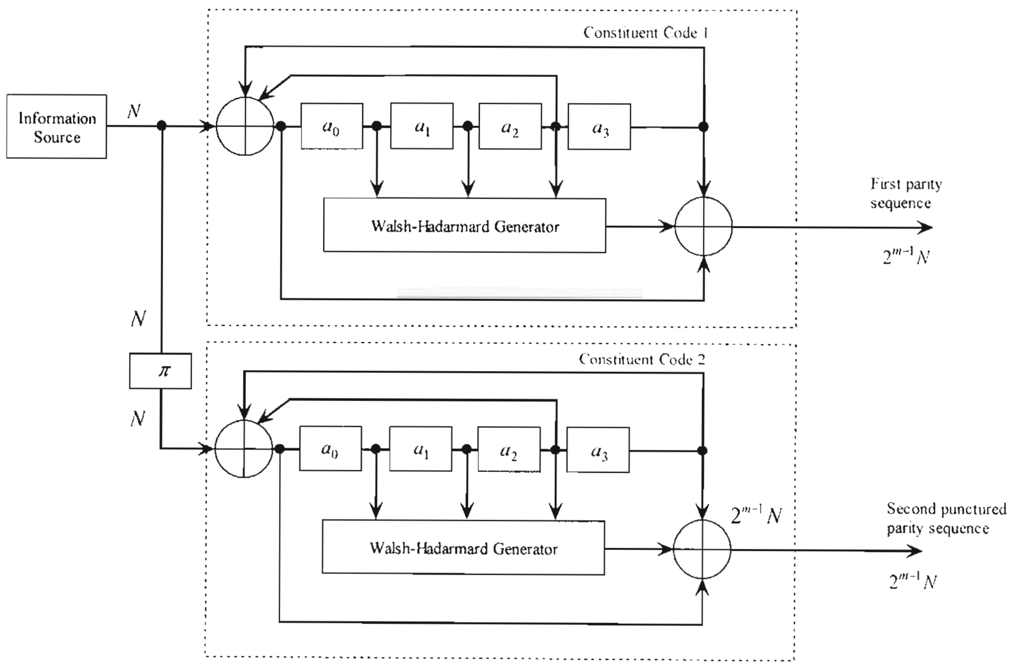


Figure 3.3C Detailed structure of the SCTC encoder [14]

To improve the performance, the memory of the code can be increased by unity without any increase in the bandwidth, but with the expense of an increased complexity. Figure 3.3D illustrates the sequences required for a similar section of the trellis as depicted in Figure 3.3B for bi-orthogonal codes.

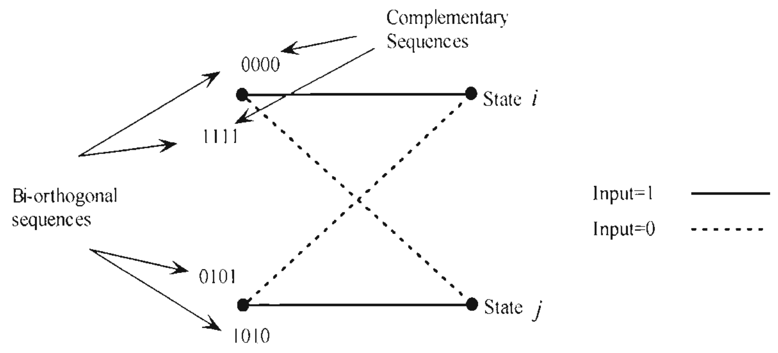


Figure 3.3D Section of trellis for encoding/decoding bi-orthogonal sequences [14]

In Figure 3.3D, it is clear that for this section of the trellis, i.e., two states, only two of the sequences are needed from the orthogonal signal set to fill up the paths, since the transition sequences can include their complementary sequences. This fact allows the memory to be increased by one, without an increase in the bandwidth. From the fact that the length of an orthogonal sequence is given by 2^{m+1} [14], then this implies a code rate of $\frac{1}{2^{m+1}}$ for orthogonal codes. Now since twice as many members are needed for the same sequence length for bi-orthogonal codes then the rate will also be twice that for orthogonal codes, i.e., $\frac{1}{2^{m}}$. Since all the transitions in the trellis that leave at the same state or arrive at the same state are antipodal (Figure 3.3B or Figure 3.3D), it turns out that it is possible to further increase performance by using only one sequence and its complement for every two states of the trellis. Thus, it is relevant to increase the memory of the bi-orthogonal code again, keeping the bandwidth constant. This new signal set is known as the superorthogonal convolutional signal set. Figure 3.3E shows the section of a trellis for the superorthogonal convolutional signal set.

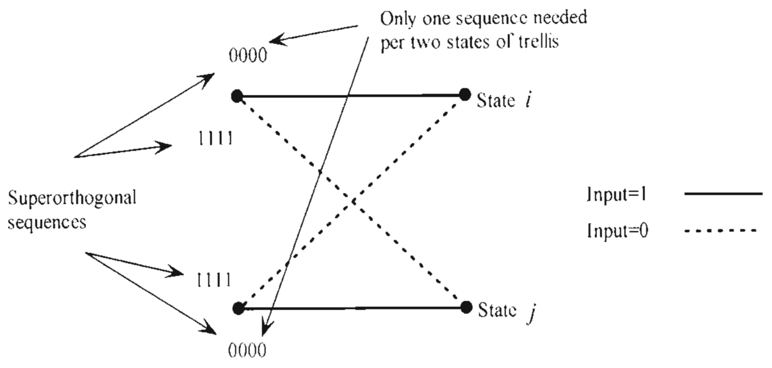

Only one sequence needed per two states of trellis
0000
1111
Superorthogonal sequences
1111
0000
State i
State j
Input=1
Input=0

Figure 3.3E Section of trellis for encoding/decoding superorthogonal sequences [14]

Again the code rate is further increased for the superorthogonal code to $\frac{1}{2^{m-1}}$. The orthogonal sequences required can be generated using many techniques; one of the possible choices for generating orthogonal sequences is utilizing the Walsh functions derived from Hadamard matrices. Recursion can be used to generate a Hadamard matrix of size $2^{\kappa} \times 2^{\kappa}$ where κ is the input alphabet size. The recursion is given by

$$\mathrm{H}_{\kappa} = \begin{bmatrix} \mathrm{H}_{\kappa-1} & \mathrm{H}_{\kappa-1} \\ \mathrm{H}_{\kappa-1} & \overline{\mathrm{H}}_{\kappa-1} \end{bmatrix} \tag{3.3a}$$

were $\mathrm{H}_1 = [1]$.

Figure 3.3F gives an example of a typical Hadamard matrix and its complementary matrix generated for a superorthogonal convolutional encoder with memory $m = 4$. Reverting back to a more detailed diagram of the encoder, shown in Figure 3.3C, the constituent encoders are both superorthogonal recursive convolutional encoders with feedback polynomial $\{10011\} = 23_{octal}$ and constraint length $K = 5$ or memory $m = 4$. The orthogonal sequences are generated by the Walsh-Hadamard generators present in each component encoder. These generators can be simple lookup tables with the relevant output sequences stored. Table 3.3 shows the state table for any of the component encoders. Evidently, there is no explicit systematic output for SCTCs as in the case of the turbo code. However, it turns out that the recursiveness of the code leads to systematic codewords at the output.

$$H_8 = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&1&1&0&0&1&1&0 \\ 0&0&0&0&1&1&1&1 \\ 0&1&0&1&1&0&1&0 \\ 0&0&1&1&1&1&0&0 \\ 0&1&1&0&1&0&0&1 \end{bmatrix} \qquad \overline{H_8} = \begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 1&0&1&0&1&0&1&0 \\ 1&1&0&0&1&1&0&0 \\ 1&0&0&1&1&0&0&1 \\ 1&1&1&1&0&0&0&0 \\ 1&0&1&0&0&1&0&1 \\ 1&1&0&0&0&0&1&1 \\ 1&0&0&1&0&1&1&0 \end{bmatrix}$$

Figure 3.3F Hadamard matrices for memory $m = 4$

Table 3.3 State table for SCTC encoder *m*=4

Starting State	Input	Next State	Output
0000	0	0000	00000000
	1	1000	11111111
0001	0	1000	00000000
	1	0000	11111111
0010	0	1001	10101010
	1	0001	01010101
0011	0	0001	10101010
	1	1001	01010101
0100	0	0010	00110011
	1	1010	11001100
0101	0	1010	00110011
	1	0010	11001100
0110	0	1011	10011001
	1	0011	01100110
0111	0	0011	10011001
	1	1011	01100110
1000	0	0100	00001111
	1	1100	11110000
1001	0	1100	00001111
	1	0100	11110000
1010	0	1101	10100101
	1	0101	01011010
1011	0	0101	10100101
	1	1101	01011010
1100	0	0110	00111100
	1	1110	11000011
1101	0	1110	00111100
	1	0110	11000011
1110	0	1111	10010110
	1	0111	01101001
1111	0	0111	10010110
	1	1111	01101001

3.4 Decoding Superorthogonal Convolutional Turbo Codes

The decoding of superorthogonal convolutional turbo codes, again makes use of the iterative technique employed in the decoding of the ordinary turbo codes. The structure of the decoder is shown in Figure 3.4A. Decoder one has two inputs, viz. the first parity sequence $y_1^{p1}, y_2^{p1}, \ldots, y_{2^{m-1}N}^{p1}$, of length 2^{m-1}, obscured by noise, and the a-priori information supplied by Decoder two. Obviously, here again, the a-priori information in the first iteration is zero for Decoder one, since the probability of receiving either a '+1' or '-1' is the same and the logarithm of unity is zero. Also, there is no explicit corrupted systematic input compared to classical turbo codes. However, as pointed out earlier the recursiveness of the code results in a systematic output. So each parity sequence actually includes the systematic bits. Utilizing

$$L_1(d_k \mid y) = \log\left[\frac{\sum_m \sum_{m'} \delta_1(y_k, m', m)\beta_k(m)\alpha_{k-1}(m')}{\sum_m \sum_{m'} \delta_0(y_k, m', m)\beta_k(m)\alpha_{k-1}(m')}\right] \tag{3.4a}$$

Decoder one then produces the soft log-likelihood decisions $L_1(d_k \mid y)$. The soft extrinsic information is gleaned from the decoding process by

$$L_{e12} = L_1(d_k \mid y) - L_{channel} - L_a \tag{3.4b}$$

Decoder two also has two inputs. Firstly the second parity sequence, $y_1^{p2}, y_2^{p2}, \ldots, y_{2^{m-1}N}^{p2}$, which again has internal systematic bits, but this time in a permuted form, and the a-priori information sent to Decoder two from Decoder one. To make the extrinsic information from Decoder one L_{e12} useful to the second component decoder, the extrinsic LLRs need to be first interleaved before being sent to Decoder two as assist information. Decoder two then uses $L_a = L'_{e12}$ to improve the reliability of its soft LLRs $L'_2(d_k \mid y)$, which is in a scrambled format and given by

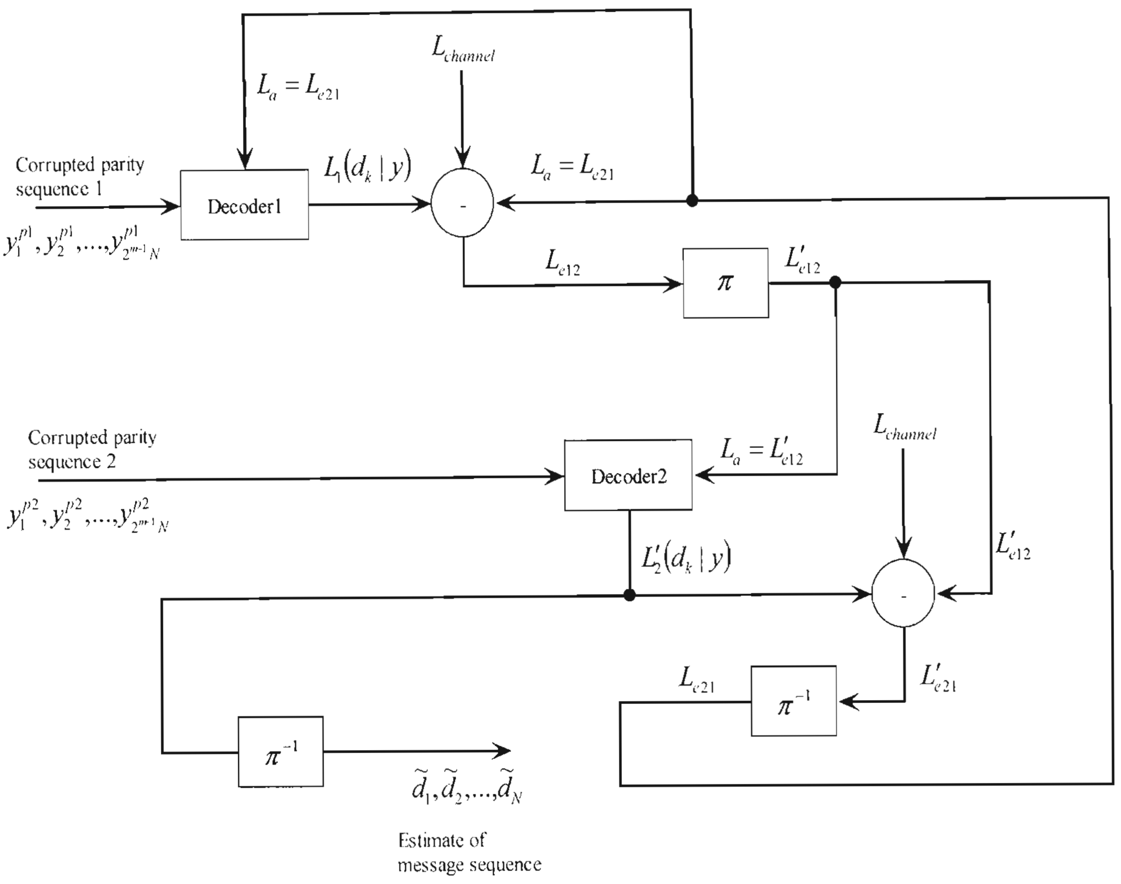

$L_{channel}$
$L_a = L_{e21}$
Corrupted parity sequence 1
$y_1^{p1}, y_2^{p1}, \dots, y_{2^{m-1}N}^{p1}$
Decoder1
$L_1(d_k \mid y)$
-
$L_a = L_{e21}$
L_{e12}
π
L'_{e12}
Corrupted parity sequence 2
$y_1^{p2}, y_2^{p2}, \dots, y_{2^{m-1}N}^{p2}$
Decoder2
$L_a = L'_{e12}$
$L_{channel}$
$L'_2(d_k \mid y)$
-
L'_{e12}
L_{e21}
π^{-1}
L'_{e21}
π^{-1}
$\tilde{d}_1, \tilde{d}_2, \dots, \tilde{d}_N$
Estimate of message sequence

Figure 3.4A Structure of the decoder for SCTC

$$L'_2(d_k \mid y) = \log\left[\frac{\sum_m \sum_{m'} \delta_1(y_k, m', m)\beta_k(m)\alpha_{k-1}(m')}{\sum_m \sum_{m'} \delta_0(y_k, m', m)\beta_k(m)\alpha_{k-1}(m')}\right] \tag{3.4c}$$

Extrinsic information gleaned from the second decoder is then given by

$$L'_{e21} = L_2(d_k \mid y) - L_{channel} - L_a \tag{3.4d}$$

The extrinsic decisions are then deinterleaved and sent back to the decoder and subtractor as a-priori information for the second iteration. After a set number of iterations of both decoders the soft decisions from Decoder two are deinterleaved then sent through a comparator to yield the message sequence estimate. It is important to note that the sign of the extrinsic information is the same as the soft LLRs from the decoder in most instances, so this factor works to improve the final estimate of the message sequence.

3.5 Performance Evaluation and Simulation results for Repeat-Punctured Turbo Codes and Superorthogonal Convolutional Turbo Codes

Transfer function bounding techniques studied in [27] and [28] were used to derive the union bounds for Repeat-Punctured Turbo Codes and Superorthogonal Convolutional Turbo Codes. The technique was also reiterated in Chapter 2. In this section we need only mention the expressions that change with the changing structure. For the RPTC scheme the second constituent encoder takes as input a sequence of length LN thus the probability of producing a codeword fragment of weight d given a randomly selected input sequence of weight i is given by

$$p(d_2 \mid i) = \frac{t(LN, Li, d_2)}{\binom{LN}{Li}} \tag{3.5a}$$

The probability of producing a codeword fragment for the first component encoder is given by

$$p(d_1 \mid i) = \frac{t(N, i, d_1)}{\sum_{d_1} t(N, i, d_1)} \approx \frac{t(N, i, d_1)}{\binom{N}{i}} \tag{3.5b}$$

The union bound is then obtained from

$$p_{bit} \leq \sum_{d=d_{\min}}^{n} \sum_{i} \sum_{d1} \sum_{d2} \frac{i}{N} \binom{N}{i} p(d_1 \mid i) p(d_2 \mid i) p_2(d) \tag{3.5c}$$

where the two codeword probability is given by the expression

$$p_2(d) \le Q(\sqrt{2dRE_b / N_o}) \tag{3.5d}$$

for the Gaussian channel and

$$p_2^{SI}(d) \le \frac{1}{2}\left(\frac{1}{1+E_s / N_o}\right)^d \tag{3.5e}$$

for the Rayleigh fading channel [28], [29].

For SCTC the constituent encoders are defined by the same expression since both encoders take as input, sequences of length N and produce sequences of the same length at their outputs. Expression (3.5b) can be used to define the component encoders of the SCTC scheme remembering that in this case the output sequence is of length $2^{m-1}N$ thus affecting the output weight. The codeword multiplicity is obtained from

$$A_d = \sum_{i=1}^{N} \sum_{d_1+d_2=d-i} \frac{A_{i,d_1} A_{Li,d_2}}{\binom{LN}{Li}} \tag{3.5f}$$

for the RPTC scheme and

$$A_d = \sum_{i=1}^{N} \sum_{d_1+d_2=d-i} \frac{A_{i,d_1}^{SRCC1} A_{i,d_2}^{SRCC2}}{\binom{N}{i}} \tag{3.5g}$$

for the SCTC scheme, were A_{i,d_1}^{SRCC1} is the weight enumerating function for the first superorthogonal component code and A_{i,d_2}^{SRCC2} represents the second constituent superorthogonal encoder.

Computer simulations were carried out for performance evaluation of the RPTC and SCTC schemes of varying complexities.

Figure 3.5A shows the simulation for the $\{7,5\}_{octal}$ turbo code in the AWGN channel, $N = 200$ and $N = 400$, at a code rate of $R = 1/3$. The simulation for the RPTC scheme in AWGN is also plotted here for the same frame lengths and code rate considered for the TC.

This graph serves to verify the performance of RPTC presented in [13]. RPTC achieves better performances at moderate to high SNRs.

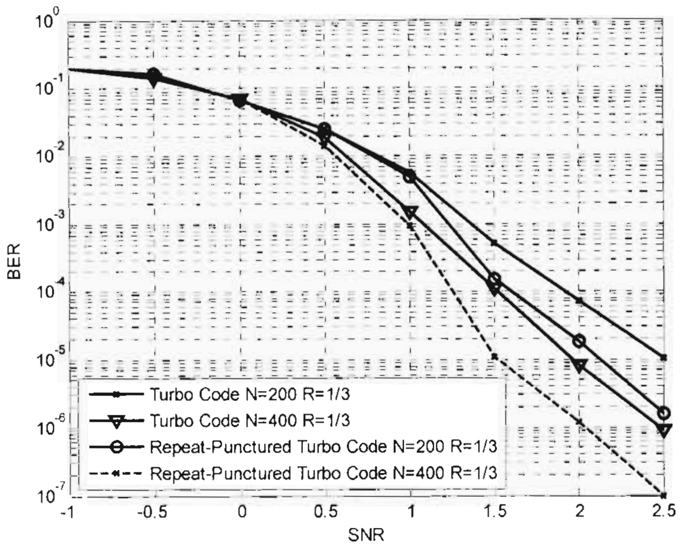


Figure 3.5A Simulation for Turbo Code and Repeat-Punctured Turbo Code in the AWGN channel, N=200, N=400, R=1/3

The turbo code can also slightly outperform RPTCs at low SNRs. At a BER of 10^{-5} RPTC exhibits a coding gain of approximately 0.5 dB for a frame length of $N = 400$ and a coding gain of 0.35 dB for an input frame length $N = 200$.

Figure 3.5B presents the simulation results for SCTC for the AWGN channel. Here memory depths of $m = 2$ and $m = 4$ have been considered. The number of iterations within the decoder is set at 18. At higher SNRs the number of iterations can be decreased, to shorten simulation time or to alternatively shorten decoding time at the receiver, without a compromise on accuracy. Almost 1 dB of coding gain is achieved at a BER level of 10^{-5} by increasing the constraint length by unity.

The simulation results for NLOS channels (modeled by the Rayleigh distribution) are exemplified in Figure 3.5C. This graph includes the simulations for SCTC for memories $m = 2$ and $m = 4$ and is plotted up to 10 dB to show the performance exhibited for low BERs. The analytical bounds derived from transfer function bounding techniques presented in [27] and [28] are also plotted. For frequency non-selective (flat Rayleigh fading) channels with side information (SI) approximately 1 dB of coding gain is achieved at a probability of error of 10^{-4}.

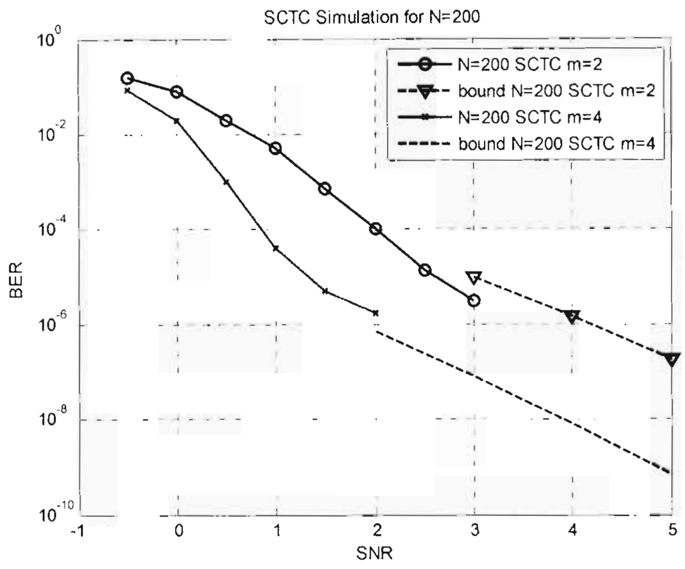


Figure 3.5B Simulations for Superorthogonal Convolutional Turbo Codes in the AWGN channel, N=200, m=2, m=4, R=1/3, R=1/15, together with analytical bounds

The simulations diverge slightly as the SNR level increases. The distance spectrum for RPTC for an input frame length $N = 128$ is presented in Figure 3.5D with multiplicity on the ordinate axis and codeword distance on the abscissa axis.

The distance spectrum verifies the performance improvement shown in the previous graph (Figure 3.5A). It can be seen that the number of codewords with low weights (small distances, *d*) has been decreased and the spectrum has been shifted to the right due to there being a larger amount of high-weight codewords present.

The distance spectrum for SCTC is examined together with the other low rate schemes in Chapter 4.

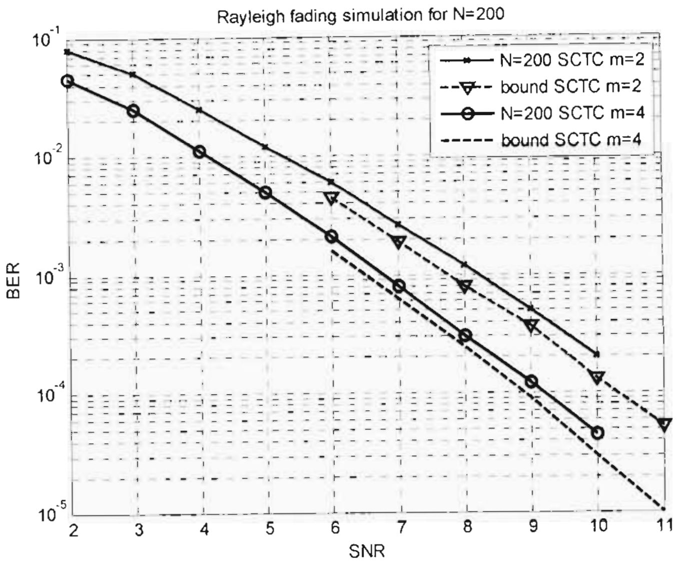


Figure 3.5C Simulations for Superorthogonal Convolutional Turbo Codes in the flat Rayleigh fading channel, N=200, m=2, m=4, R=1/3, R=1/15, including the bounds

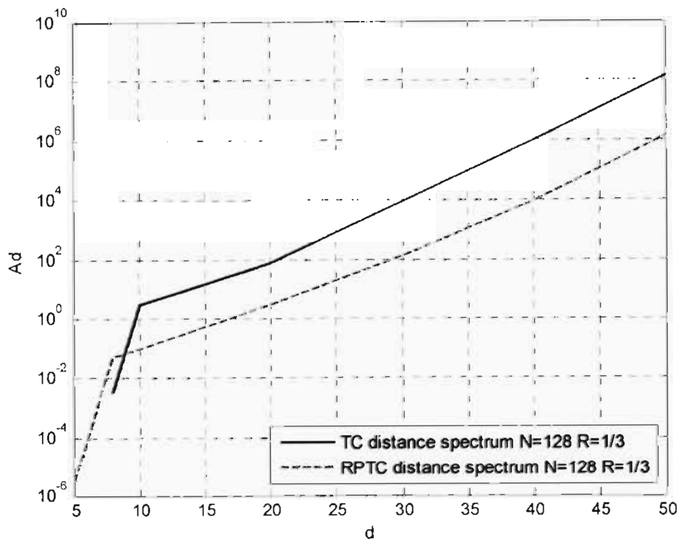


Figure 3.5D Distance spectrum for Turbo Code and Repeat-Punctured Turbo Code N=128, R=1/3

In this chapter, RPTCs and SCTCs were presented. These schemes were previously introduced to the coding community. Simulation results for the AWGN channel and Rayleigh fading channel was presented. The distance spectrum was also considered.

Chapter 4
Repeat-Punctured Superorthogonal Convolutional Turbo Codes, Dual-Repeat-Punctured Turbo Codes and Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes

In Chapter 3 Superorthogonal Convolutional Turbo Codes were introduced. This makes use of superorthogonal signals to create a low-rate code suitable for spread-spectrum applications. SCTCs exhibit an improved performance over the classical turbo coding algorithm. For small frame lengths, e.g. $N = 200$, and a constraint length $K = 4$, E_b / N_o values of approximately 0.7 dB at a BER level of 10^{-3} have been reported [14].

The performance of turbo codes improves with increasing interleaver size, due to the larger interleaver gain [1], [3], [7], [13]. The proposed repeat-punctured turbo codes (RPTC) in [13] not only increases the interleaver size, but also enlarges the low weight codewords. Repeat-Punctured Turbo Codes have shown a significant increase in performance at moderate to high SNRs [13]. For example, for an information frame length $N = 1024$, simulation results have shown that the RPTC scheme has approximately 2 dB coding gain over the conventional turbo codes at a bit-error rate level of 10^{-5} in the AWGN channel. Motivated by RPTC, we applied orthogonal signaling, and together with a repeat-punctured mechanism, simulations have shown much improved performances.

4.1 Encoding with Repeat-Punctured Superorthogonal Convolutional Turbo Codes

Similar to the conventional SCTC, a repeat-punctured superorthogonal convolutional turbo code (RPSCTC) encoder consists of two parallel concatenated superorthogonal recursive convolutional codes as shown in Figure 4.1A. As an example, a more detailed diagram with memory $m = 4$ is depicted in Figure 4.1B. The first parity sequence $y_1^{p1}, y_2^{p1}, \ldots, y_{2^{m-1}N}^{p1}$ is produced from the first constituent encoder with an input data frame of length N. Since the parallel concatenated code (PCC) is made up of superorthogonal recursive convolutional code (SRCC) component codes the length of the first parity sequence is $2^{m-1}N$ [14]. However, unlike the conventional turbo code where the input data frame of length N is directly permuted by an interleaver of length N, the input data frame is repeated L times and interleaved by an interleaver π of length LN. The second parity sequence $y_1^{p2}, y_2^{p2}, \ldots, y_{2^{m-1}LN}^{p2}$ is produced from the second constituent code with an interleaved data frame of length LN as input. To control the code rate at the output the second parity sequence is then punctured and serially transmitted together with the first parity sequence to the receiver front-end. As pointed out earlier in Chapter 3, the superorthogonal recursive constituent encoders turn out to be implicitly systematic due to the infinite impulse response of the code. We thus had the liberty of puncturing one of the bits from the Walsh sequences. In the case of RPSCTC, since repetition is used in the second branch of encoding this leads to a large drop in code rate which has to be dealt with prior to transmission. Figure 4.1C illustrates the method of puncturing employed in the RPSCTC scheme. In the figure, puncturing is undertaken on a superorthogonal recursive encoder with memory depth $m = 4$. This implies a code rate of $R = \frac{N}{2^{m-1}N} + \frac{N}{2^{m-1}LN} = \frac{1}{8} + \frac{1}{16} = \frac{1}{24}$. To draw comparisons between superorthogonal

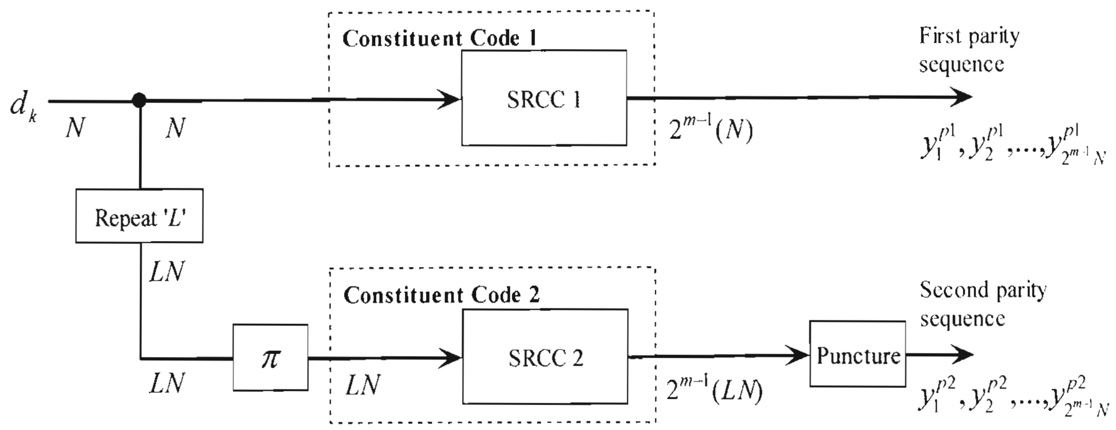


Figure.4.1A Structure of RPSCTC Encoder

convolutional turbo codes and RPSCTC, the code rate was increased to $R = \frac{1}{15}$ by puncturing. For the second parity sequence, all alternate bits were punctured, i.e., puncture every l bits from every L bits, where $l = 1$ and $L = 2$. Only the last bit from the first parity sequence is punctured to bring the code rate to $R = \frac{1}{15}$. The Walsh-Hadamard generator uses Walsh functions obtained from Hadamard matrices to generate the orthogonal sequences. For an encoder with memory $m = 4$, the Hadamard matrices H_8 and its complementary matrix $\overline{H_8}$ need to be considered.

The effect of using larger interleavers and constant frame lengths was pointed out in [13]. Figure 4.1D graphically represents the phenomena known as spectral thinning, which is largely responsible for the remarkable performance of the conventional turbo coding scheme and the schemes discussed in Chapter 3. It can be seen that as the interleaver length B is increased towards infinity for hypothetical distance spectra, the distance spectrum begins to 'thin' for low-weight codewords.

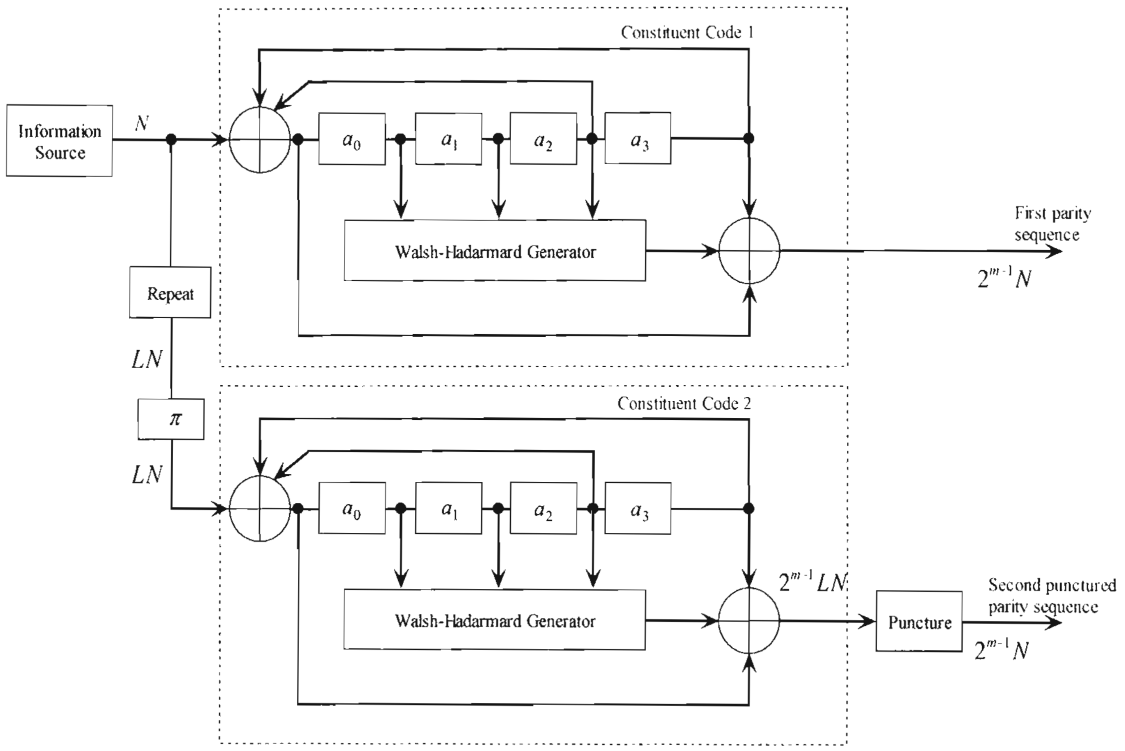


Figure 4.1B Detailed structure of the RPSCTC encoder

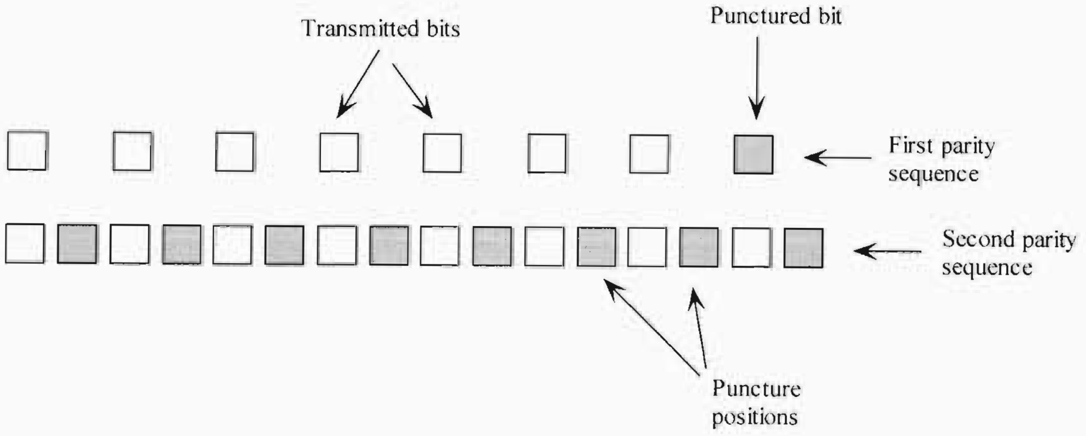


Figure 4.1C Puncturing technique for RPSCTC

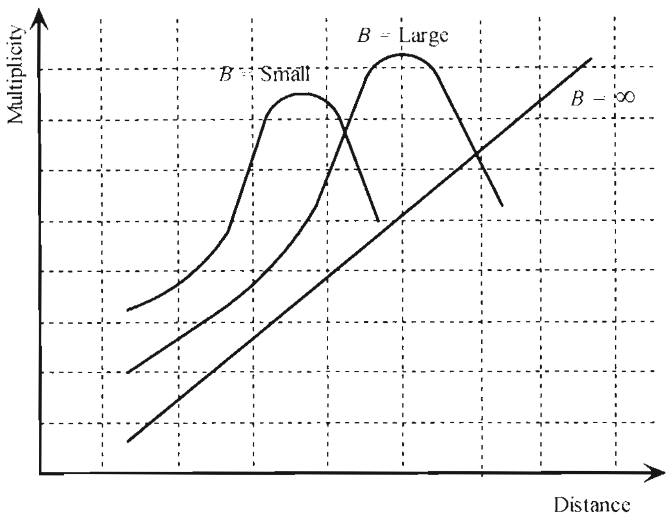
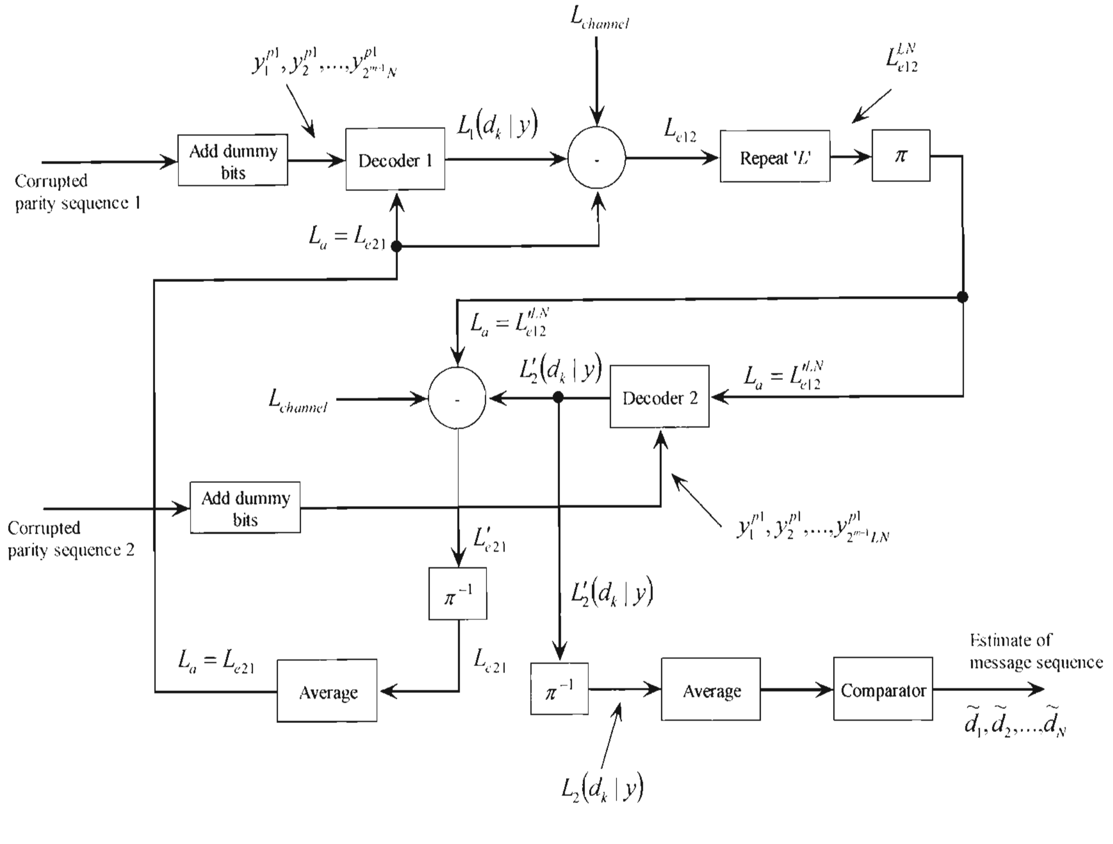


Figure 4.1D Spectral thinning due to increasing interleaver size[25]

It is important to note that this thinning of the codeword distance spectrum allows the free-distance asymptote of a turbo code to dominate the performance for lower SNRs and thus achieve near Shannon-capacity performance [25].

4.2 Decoding Repeat-Punctured Superorthogonal Convolutional Turbo Codes



$L_{channel}$
$y_1^{p1}, y_2^{p1}, \ldots, y_{2^{m-1}N}^{p1}$
L_{e12}^{LN}
Add dummy bits
Decoder 1
$L_1(d_k \mid y)$
-
L_{e12}
Repeat 'L'
π
Corrupted parity sequence 1
$L_a = L_{e21}$
$L_a = L_{e12}^{\prime LN}$
$L_2'(d_k \mid y)$
Decoder 2
$L_a = L_{e12}^{\prime LN}$
$L_{channel}$
-
Add dummy bits
Corrupted parity sequence 2
L_{e21}'
$y_1^{p1}, y_2^{p1}, \ldots, y_{2^{m-1}LN}^{p1}$
π^{-1}
$L_2'(d_k \mid y)$
Estimate of message sequence
$L_a = L_{e21}$
Average
L_{e21}
π^{-1}
Average
Comparator
$\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_N$
$L_2(d_k \mid y)$

Figure 4.2A Structure of the RPSCTC decoder

Similar to the conventional turbo codes, RPSCTC uses the MAP algorithm in each of the component decoders.

The decoding structure as shown in Figure 4.2A makes use of an iterative technique, together with serial concatenation of the constituent decoders. An outer decoder, Decoder 1 and an inner decoder, Decoder 2, both in cooperation, produce soft decisions that are converted into extrinsic log-likelihood ratios, which are mutually exchanged, thus increasing the reliability of the decisions.

In the first step of the decoding for RPSCTC, the punctured corrupted first parity sequence from the channel is sent to Decoder 1. Since the sequence was punctured before transmission, dummy bits now need to be added. This is also the case for the second parity sequence. Figure 4.2B illustrates the manner in which the dummy bits are added at the input of the decoder. The lightly shaded, dash outlined blocks represent positions at which bits were punctured to increase the code rate. Before either corrupted parity sequence is sent to a component decoder, dummy bits, '1', are introduced into these

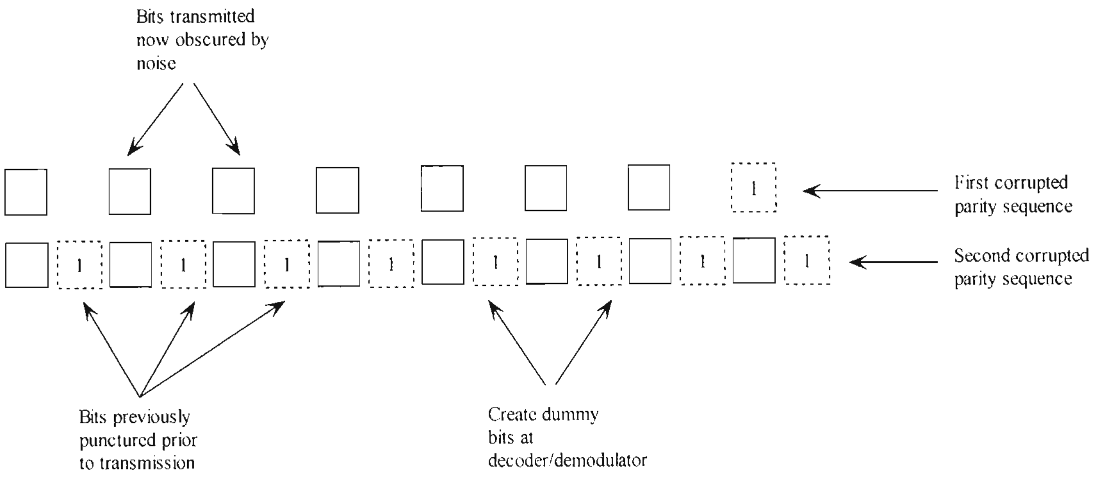

Bits transmitted now obscured by noise
1
First corrupted parity sequence
1
1
1
1
1
1
1
1
Second corrupted parity sequence
Bits previously punctured prior to transmission
Create dummy bits at decoder/demodulator

Figure 4.2B Dummy bits introduced before decoding

positions. This serves to reconstruct the original parity sequences prior to puncturing at the encoder. The reconstructed first parity sequence $y_1^{p1}, y_2^{p1}, \ldots, y_{2^{m-1}N}^{p1}$ is sent to the first decoder. Decoder 1 assumes zero a-priori information and utilizes the MAP algorithm to yield N soft LLRs $L_1(d_k \mid y)$ at its output. This needs to be converted into N extrinsic decisions which can be used to assist Decoder 2 in its LLR computations. Taking into account the channel measurement and zero a-priori information, the extrinsic information is computed in a manner similar to the turbo code. At this point it is important to consider the second decoders initial processing. The reconstructed second parity sequence $y_1^{p2}, y_2^{p2}, \ldots, y_{2^{m-1}LN}^{p2}$ is fed into Decoder 2. Since the second decoder is operating on enlarged sequences, the a-priori information assigned to the first decoder's extrinsic output has to also be enlarged. These N extrinsic decisions from Decoder 1 L_{e12} are first repeated L times to yield L_{e12}^{LN}. This yields the extrinsic decisions of the same form as the original input message sequence. Now, earlier it was pointed out that the second parity sequence from the encoder contained systematic bits, but in a permuted form. It is thus important to interleave, $\pi\left(L_{e12}^{l,N}\right)$, the repeated extrinsic sequence before input to the second decoder. Decoder two now uses this second input to improve the reliability of its computation of $L_2^{'}(d_k \mid y)$. These LN soft LLRs are routinely converted into LN extrinsic decisions $L_{e21}^{'l,N}$. The extrinsic decisions are then deinterleaved $L_{e21}^{l,N}$. This deinterleaved sequence is then averaged to produce N deinterleaved extrinsic decisions L_{e21}. The process of averaging is responsible for a performance improvement in the decoding, since LN decisions from the second decoder are being converted into N decisions and is thus continuously working to improve the reliability of the assist information to Decoder one. The mutual exchange of extrinsic information in the overall decoder is iterated several times. After a sufficient number of iterations the soft output from Decoder 2 $L_2^{'l,N}(d_k \mid y)$ is deinterleaved and averaged to yield N soft decisions. The soft decisions are then converted into hard decisions, i.e., the estimate of the original message sequence.

4.3 Encoding with Dual-Repeat-Punctured Turbo Codes

Repeat-punctured turbo codes have shown improved performances in terms of bit-error rate versus signal-to-noise ratios at moderate to high SNRs [13]. It turns out that by using a dual repeat-punctured mechanism within the encoder, better performances can be obtained. The dual-repeat-punctured turbo code (DRPTC) scheme is an extension of the repeat-punctured turbo codes scheme presented in [13]. The structure of the DRPTC encoder is shown in Figure 4.3A. In each messaging interval an input message sequence of length N is produced at the input. The information frame is sent directly to the output to form the systematic output branch. Prior to processing for the parity sequences, the input frame is repeated L times. This allows for the use of larger interleavers before the constituent encoders process their respective inputs.

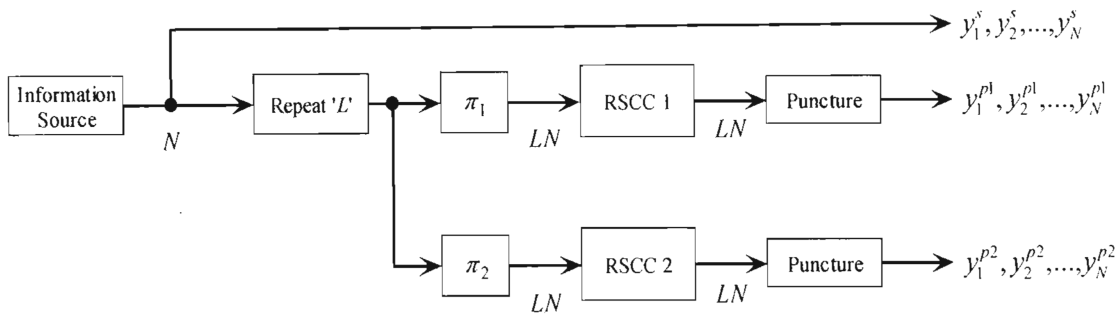

Information Source
N
Repeat 'L'
π_1
LN
RSCC 1
LN
Puncture
π_2
LN
RSCC 2
LN
Puncture
$y_1^s, y_2^s, \ldots, y_N^s$
$y_1^{p1}, y_2^{p1}, \ldots, y_N^{p1}$
$y_1^{p2}, y_2^{p2}, \ldots, y_N^{p2}$

Figure 4.3A Structure of the DRPTC Encoder

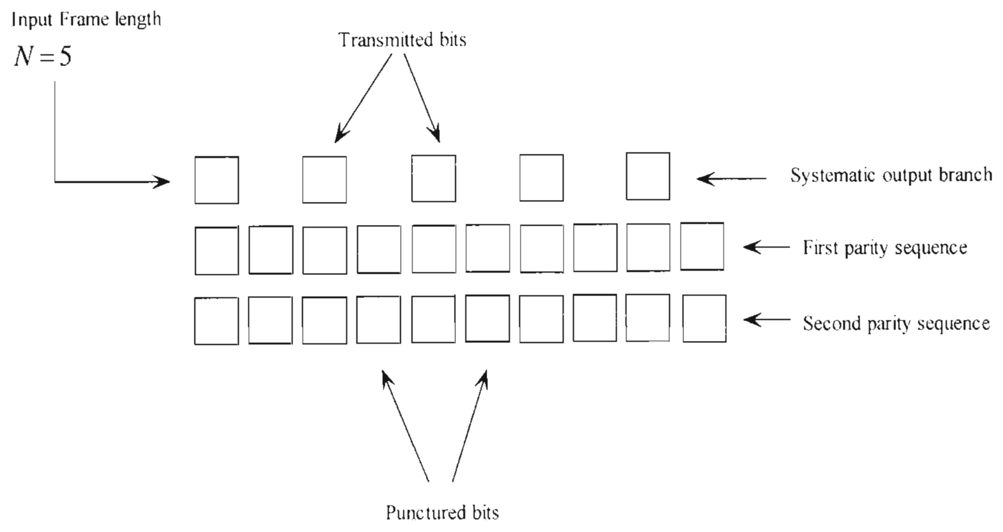


Figure 4.3B Puncturing undertaken at transmitter

Note that there are two interleavers, i.e., two interleaver mappings π_1 and π_2, utilized in this scheme or as many interleavers as parity branches implemented. After interleaving, the enlarged message sequences are encoded by recursive systematic convolutional coders. The output codewords are again of length LN or depending on the code rate of the component encoder if not a single rate one. The resultant parity sequences $y_1^{p1}, y_2^{p1}, \ldots, y_{LN}^{p1}$ and $y_1^{p2}, y_2^{p2}, \ldots, y_{LN}^{p2}$ are now responsible for the decreased code rate, i.e., $\dfrac{N}{N+LN+LN} = \dfrac{N}{2LN+N} = \dfrac{1}{1+2L}$. To counteract this, puncturing is employed to recover this loss in the code rate. Figure 4.3B illustrates how the puncturing is carried out for DRPTC. For the parity sequences every l bits from every L bits are punctured.

Puncturing of the systematic output need not be done, since we don't employ repetition here, which is unnecessary at this point. However these systematic bits need to be repeated again prior to decoding which contributes to an increase in performance exhibited by the code.

4.4 Decoding Dual-Repeat-Punctured Turbo Codes

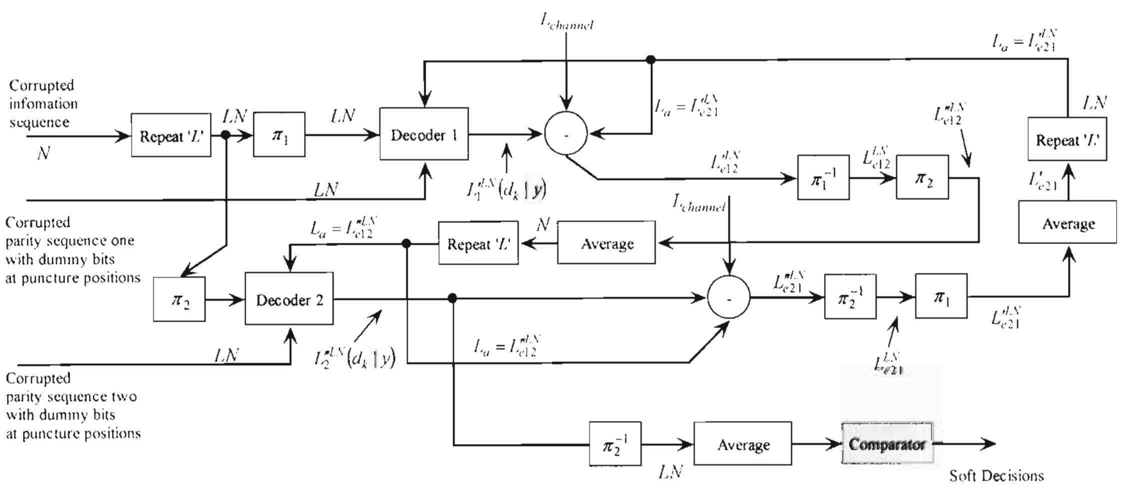


Figure 4.4A Structure of the decoder for DRPTC

The decoding of dual-repeat-punctured turbo codes is of a higher complexity than that of classical turbo codes. The structure of the decoder is presented in Figure 4.4A. Prior to feeding inputs to respective component decoders, the sequences at the receiver need to be reconstructed (due to puncturing at the transmitter) in a manner so as to satisfy the processing needs of the constituent decoders.

Figure 4.4B illustrates how these sequences are prepared for decoding. The corrupted systematic output sequence is of length N. This sequence is first repeated L times. The shaded blocks in the diagram show the repeated systematic bits. For the parity sequences, dummy bits need to be introduced into positions where those bits were previously punctured at the transmitter. In the lower part of Figure 4.4B the parity sequences are depicted, the previously punctured positions are now occupied by dummy bits '1'. These positions are represented by lightly shaded dash outlined blocks.

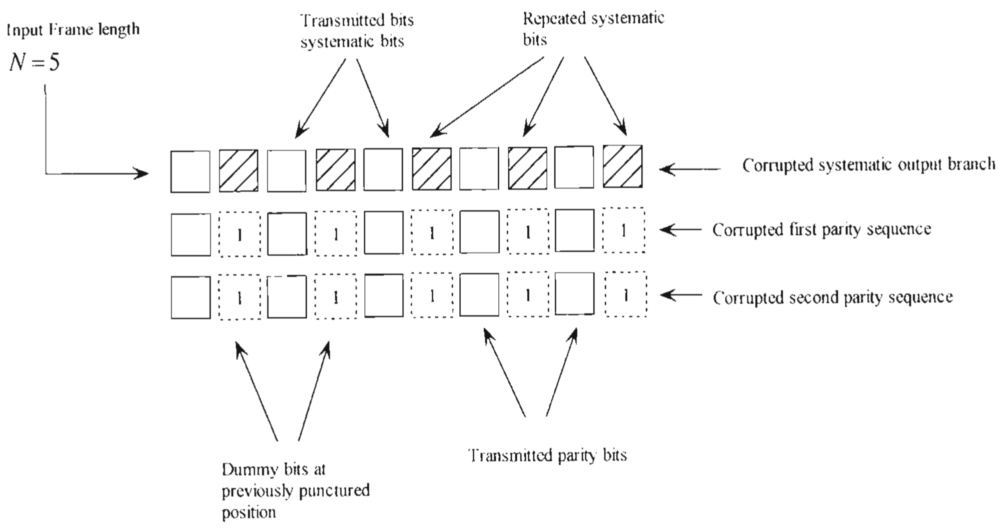

Input Frame length
$N=5$
Transmitted bits systematic bits
Repeated systematic bits
Corrupted systematic output branch
1
Corrupted first parity sequence
Corrupted second parity sequence
Dummy bits at previously punctured position
Transmitted parity bits

Figure 4.4B Dummy bits introduced prior to decoding

Also since two interleavers are being used here it is useful to have a special naming convention. Figure 4.4C shows the type of naming convention that will be used. The input sequence ϑ_1 if interleaved by the first interleaver π_1 will result in an output sequence ϑ_1' and an input sequence ϑ_2 if interleaved by the second interleaver mapping π_2 will result in a sequence ϑ_2''.

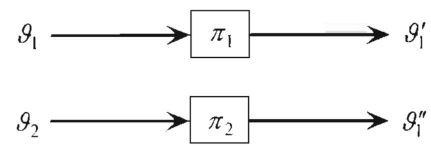


Figure 4.4C Naming convention used when two interleavers are utilized

Reverting back to the structure of the decoder in Figure 4.4A, Decoder one accepts two inputs in the first iteration, viz. the repeated systematic sequence and the corrupted first parity sequence. The repeated systematic sequence $y_1^s, y_2^s, \ldots, y_{LN}^s$ has to be interleaved by the first interleaver mapping π_1 before being sent to Decoder 1. Decoder one then uses these two sequences to generate soft log-likelihood decisions $L_1'^{LN}(d_k \mid y)$. These soft LLRs are next converted into soft extrinsic decisions $L_{e12}'^{LN}$. Prior to supplying this as a-priori information to the second decoder, the extrinsic information is first deinterleaved $\pi_1^{-1}\left(L_{e12}^{\prime\prime LN}\right)$ then interleaved by interleaver mapping two to yield the extrinsic sequence $L_{e12}^{\prime\prime LN}$. Averaging and repetition of the extrinsic decisions can be utilized to improve the reliability of the assist information $L_a = L_{e12}^{\prime\prime LN}$ for Decoder 2.

The second component decoder takes three sequences as input, viz. the assist information supplied from Decoder 1, the repeated systematic sequence which has to be first interleaved by the second interleaver mapping, and the corrupted second parity sequence with dummy bits in puncture positions. Decoder two now uses these inputs to produce soft LLRs $L_2^{\prime\prime LN}(d_k \mid y)$. These soft LLRs are routinely converted into soft extrinsic information $L_{e21}^{\prime\prime LN}$, which has to be deinterleaved and then interleaved in the opposite manner to the extrinsic information from Decoder 1. The extrinsic information $L_{e21}^{\prime\prime LN}$ is deinterleaved by the second interleaver mapping to yield L_{e21}^{LN}. This new sequence is next interleaved by the first interleaver matrix, i.e., $\pi_1\left(L_{e21}^{LN}\right)$ to produce $L_{e21}^{\prime LN}$. Next this new extrinsic information sequence is averaged and repeated to improve the reliability of the a-priori information $L_a = L_{e21}^{\prime LN}$ supplied to Decoder one in the second iteration. This exchange of extrinsic information between the decoders is iterated several times after which the LN soft LLRs from Decoder 2, $L_2^{\prime\prime LN}(d_k \mid y)$ are deinterleaved by π_2^{-1}, then averaged and then converted into hard decisions to yield the message sequence estimate, $\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_N$. The averaging and repetition involved in the mutual exchange of information between the decoders is quite crucial to this scheme and serves to improve the performance by a substantial amount.

4.5 Encoding with Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes

Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes (DRPSCTC) is an extension of the RPSCTC scheme presented in [26] and Section 4.1. It turns out that a simple change in the structure, i.e., dual repetition, can serve to improve the performance further. The structure of the encoder for DRPSCTC is shown in Figure 4.5A.

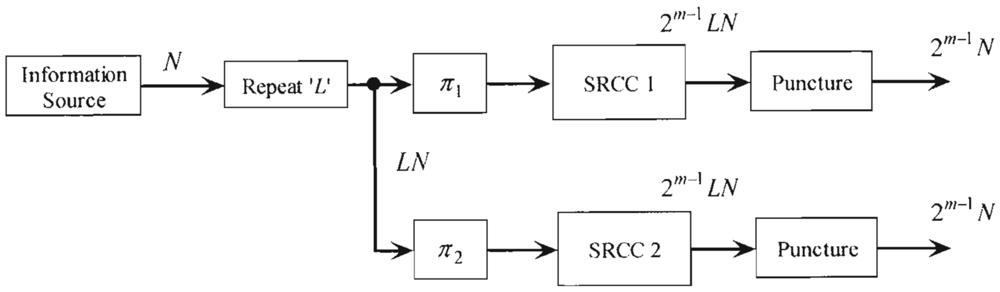


Figure 4.5A Structure of the DRPSCTC encoder

This is almost identical to the encoder used for dual-repeat-punctured turbo codes, with the major difference being the constituent encoders and consequently the length of the sequences being produced at the outputs. The message sequence of length N is first sent through a repeater structure which enlarges the sequence to LN entries. On their respective branches the enlarged sequences are interleaved, then encoded with superorthogonal signals to yield sequences of length $2^{m-1}LN$. Now to control the code rate the sequences need to be punctured. Here a special technique needs to be considered for puncturing.

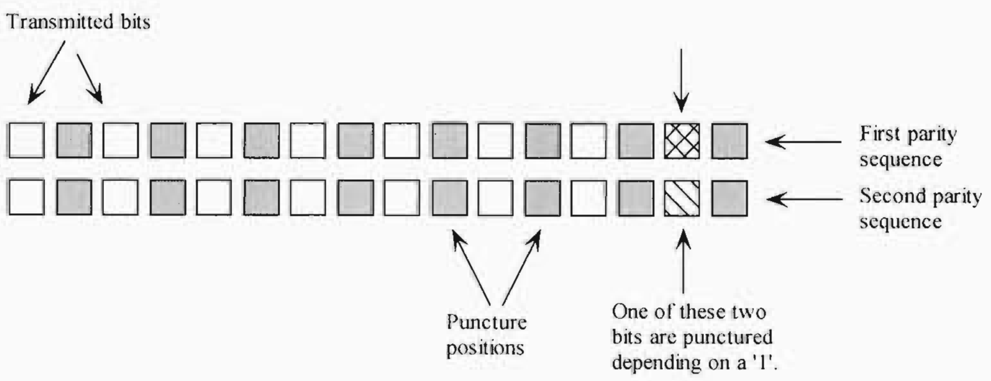
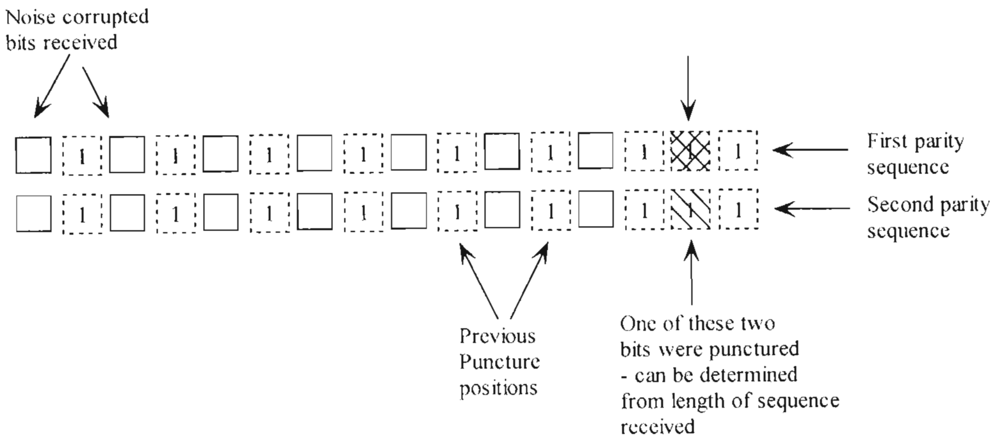


Figure 4.5B Puncturing for the DRPSCTC scheme

The aim here is to puncture the sequences to the same rate that we used to investigate the low rate schemes, i.e., the SCTC scheme and the RPSCTC scheme, so that comparisons can be drawn between these schemes. Figure 4.5B illustrates the technique employed for the puncturing of DRPSCTC coded outputs. Now for a memory $m = 4$ and repetition index $L = 2$, there should be $2^{m-1}LN$ bits at the output of the branch for every input bit, i.e., 16 bits. This translates to a total of 32 bits at the output for two parity sequences, as shown in Figure 4.5B. The completely shaded blocks represent bits that are punctured. One bit still needs to be punctured to increase the rate to $R = \frac{1}{15}$. On the right portion of Figure 4.5B, two bits, represented by the partially shaded blocks, are monitored for a '1'. Any of these two bits are punctured if a '1' is detected. However if a '1' is not found then any of these two bits are punctured for every alternate input frame. This is done so as to effectively ensure a code rate of $R = \frac{1}{15}$.

4.6 Decoding Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes

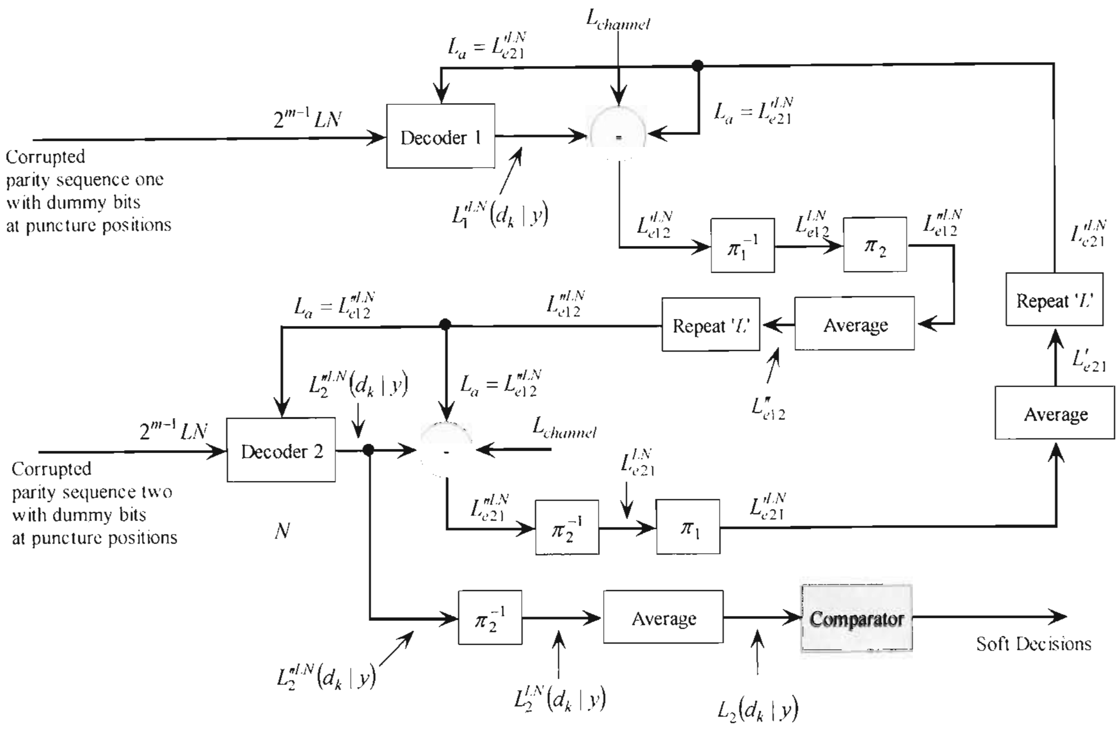
The decoder for DRPSCTC is similar to the structure of the decoder for dual-repeat-punctured turbo codes. However in this case there is no explicit systematic output thus eliminating the need for interleaving at the input. Firstly, the sequences need to be reconstructed before being sent to the constituent decoders. Figure 4.6A illustrates how the sequences are reconstructed. Firstly, on reception of the parity sequences the lengths of each of these sequences need to be checked to determine which sequence was punctured for an extra bit. This position should be occupied by a '1'. These positions are represented by the partially shaded dash outlined blocks in the right portion of the diagram. At this point each of the parity sequences should be of length $2^{m-1}N$. Next dummy bits '1' are inserted into previously punctured bit positions. These positions are shown by the lightly shaded dash outlined blocks in Figure 4.6A.



Noise corrupted bits received
1
First parity sequence
Second parity sequence
Previous Puncture positions
One of these two bits were punctured - can be determined from length of sequence received

Figure 4.6A Sequence reconstruction prior to decoding

The structure of the decoder is shown in Figure 4.6B. In the first iteration, Decoder 1 has a single input, i.e., the corrupted parity sequence one reconstructed to length $2^{m-1}LN$. Using this input the MAP algorithm is utilized to generate soft LLRs $L_1^{\prime LN}(d_k \mid y)$ at the decoder output. Note that the soft LLRs are in a π_1 permuted format and is thus denoted with a single dash. The same convention discussed in Section 4.4 is being used here. $L_1^{\prime LN}(d_k \mid y)$ is next converted into soft extrinsic information $L_{e12}^{\prime LN}$ in a manner similar to the conventional turbo decoding and the ensuing schemes discussed. This extrinsic information is next deinterleaved by the first interleaver mapping, then interleaved by the second interleaver mapping. It is important to note that the order of interleaving and deinterleaving here depends on the order of interleaving done within the encoder. The resulting sequence of scrambled extrinsic decisions is then followed by averaging and repetition to improve the reliability of the assist information to Decoder 1. Decoder 2 takes two inputs, viz. the a-priori information sent from Decoder 1 and the second corrupted parity sequence with dummy bits in previously punctured positions to produce soft LLRs $L_2^{\prime\prime LN}(d_k \mid y)$ in a π_2 permuted format. The extrinsic sequence $L_{e21}^{\prime\prime LN}$ is then deinterleaved by the second interleaver mapping to yield L_{e21}^{LN} which is then permuted by π_1.



$L_{channel}$
$L_a = L_{e21}^{\prime I.N}$
$2^{m-1}LN$
Decoder 1
Corrupted parity sequence one with dummy bits at puncture positions
$L_1^{\prime I.N}(d_k \mid y)$
$L_{e12}^{\prime I.N}$
π_1^{-1}
$L_{e12}^{I.N}$
π_2
$L_{e12}^{\prime\prime I.N}$
$L_{e21}^{\prime I.N}$
Repeat 'L'
$L_a = L_{e12}^{\prime\prime I.N}$
Average
L_{e12}''
L_{e21}'
$L_2^{\prime\prime I.N}(d_k \mid y)$
$2^{m-1}LN$
Decoder 2
Corrupted parity sequence two with dummy bits at puncture positions
N
$L_{e21}^{I.N}$
π_2^{-1}
π_1
Comparator
Soft Decisions
$L_2^{I.N}(d_k \mid y)$
$L_2(d_k \mid y)$

Figure 4.6B Structure of the Decoder for DRPSCTC

These LN elements are then averaged and repeated L times to produce a-priori information suitable for improving the error performance of Decoder 1. After the mutual exchange of information between the decoders are iterated several times, the soft output from Decoder 2 is deinterleaved $\pi_2^{-1}\left(L_2^{\prime\prime I.N}(d_k \mid y)\right)$, then averaged to yield N soft decisions, which are converted into hard decisions to yield an estimate of the original message sequence.

4.7 Performance Evaluation and Simulation results for RPSCTC, DRPTC and DRPSCTC

The union bounds for the RPSCTC, DRPTC and DRPSCTC schemes are all evaluated using the same transfer function method discussed in [27], [28] and Chapter 2, Section 2.2.4. In this section we need only present the expressions for each of the encoders for the abovementioned schemes.

For RPSCTC, the probability of producing a codeword fragment of weight d given a randomly selected input sequence of weight i for the first encoder is given by

$$p(d_1 \mid i) = \frac{t(N,i,d_1)}{\sum_{d_1} t(N,i,d_1)} \approx \frac{t(N,i,d_1)}{\binom{N}{i}} \tag{4.7a}$$

and by

$$p(d_2 \mid i) = \frac{t(LN,Li,d_2)}{\binom{LN}{Li}} \tag{4.7b}$$

for the second encoder.

Due to the dual structure for both DRPTC and DRPSCTC both component encoders take as input a message block of length LN. Thus the probability of producing a codeword of weight d conditioned on a weight i input sequence is given by

$$p(d_q \mid i) = \frac{t(LN,Li,d_q)}{\binom{LN}{Li}} \tag{4.7c}$$

were $q=1$ for the first encoder and $q=2$ for the second encoder.

The codeword distance spectrums are given by

$$A_d = \sum_{i=1}^{N} \sum_{d_1+d_2=d-i} \frac{A_{i,d_1}^{SRCC1} A_{Li,d_2}^{SRCC2}}{\binom{LN}{Li}} \tag{4.7d}$$

for RPSCTC,

$$A_d = \sum_{i=1}^{N} \sum_{d_1+d_2=d-i} \frac{A_{Li,d_1}^{SRCC1} A_{Li,d_2}^{SRCC2}}{\binom{LN}{Li}} \tag{4.7e}$$

for DRPSCTC, and

$$A_d = \sum_{i=1}^{N} \sum_{d_1+d_2=d-i} \frac{A_{L,i,d_1} A_{L,i,d_2}}{\binom{LN}{Li}} \qquad (4.7f)$$

for the DRPTC scheme with individual RSC weight enumerating functions.

All simulations were undertaken in Borland C++. Figure 4.7A shows the simulation of the RPSCTC scheme in AWGN for memory $m = 4$ and an input sequence frame length of $N = 200$. The dashed line represents the analytical bound plotted from SNR=1 dB through to 5 dB. A plot for SCTC scheme presented in Sections 3.3 and 3.4 is also included in this graph for comparative reasons. The rate of the two codes is $R = 1/15$ due to puncturing. At a BER of 10^{-3} a coding gain of approximately 0.5 dB is evident and at a BER level of 10^{-6} a 1.5 dB coding gain is achieved.

Figure 4.7B shows a similar graph for $m = 2$, plots of RPSCTC, $N = 200$ versus the superorthogonal turbo coding scheme at a frame length of $N = 200$, again in the AWGN channel.

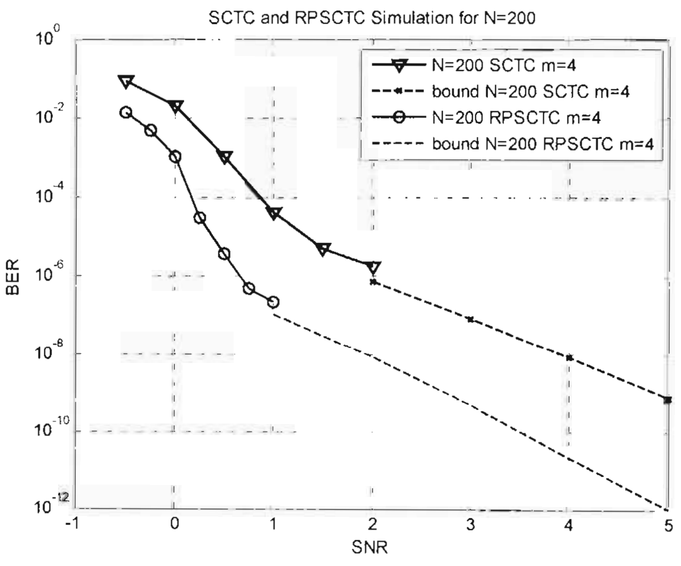


Figure 4.7A Simulation for SCTC and RPSCTC in the AWGN channel, N=200, m=4, R=1/15, including the analytical bounds

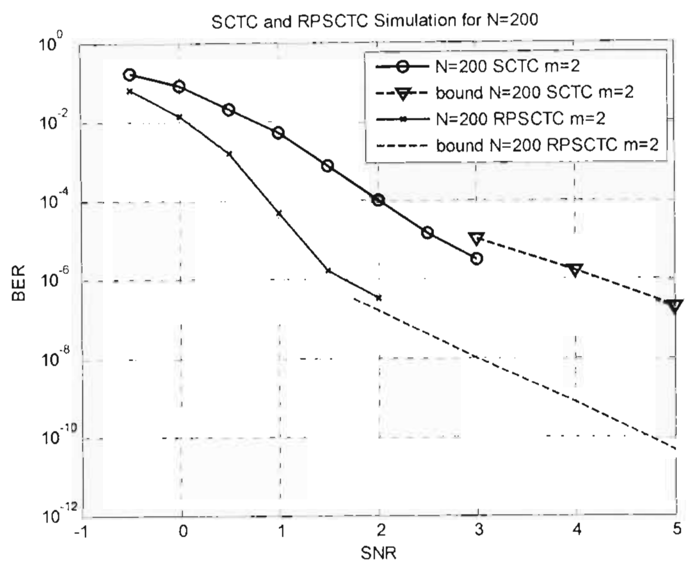


Figure 4.7B Simulation for SCTC and RPSCTC in the AWGN channel, N=200, m=2, R=1/3, including the analytical bounds

The constraint length in this scenario is $K=3$ and 3 bits are being transmitted for every input message bit. Here again similar coding gains are achieved throughout the range of signal-to-noise ratios considered. Figure 4.7C plots the two graphs for repeat-punctured superorthogonal convolutional turbo codes together with their analytical bounds, for the AWGN channel. At a BER of 10^{-3} a coding gain of almost 0.6 dB is achieved and at a bit-error level of 10^{-6} approximately 0.8 dB of coding gain is observed. The superior performance exhibited by the memory $m=4$ code is due to the increase in the number of states. It is important to note that one of the disadvantages offered by the superorthogonal signal-based schemes is the fact that the rate is exponentially related to the memory depth of the code. Best performances are also noted around constraint lengths of $K=3$ to $K=5$ following the conventional turbo codes [1], [9]. Rayleigh fading is a statistical model for the effect of a propagation environment, (or the heavy build up of urban environments), on a radio signal and is a reasonable model for tropospheric and

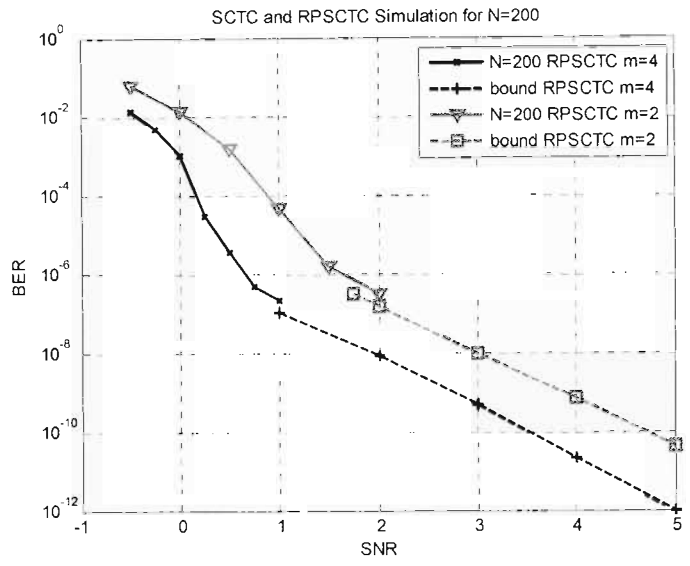


Figure 4.7C Simulation for RPSCTC in the AWGN channel, N=200, m=2, m=4, R=1/3, R=1/15

ionospheric signal propagation.

For non-line-of-sight (NLOS) between the transmitter and receiver, the scheme was also simulated in a Rayleigh fading channel. The results are presented in Figures 4.7D and 4.7E. Again the measure of quality of service (QoS) chosen is the bit-error rate versus SNR plot. 150 frame errors were used as a stopping criterion for the simulations for both SCTC and RPSCTC. The velocity used for the moving receiver was v_c = 50 km/h, and a f_c = 2 GHz carrier frequency was used with Doppler frequency f_d given by (4.7j). The speed of light c is 3×10^8 m/s.

$$f_d = \frac{v_c f_c}{c} \tag{4.7j}$$

Again a frame length of $N = 200$ was considered at a code rate of $R = 1/15$. A SNR range of 2 - 11 dB is considered so that the performance for low probability of errors can be investigated. A coding gain of almost 1 dB is achieved throughout the SNR range of SCTC and RPSCTC. Figures 4.7F and 4.7G present the simulation results for the DRPTC scheme. The first of the graphs presents the plots for the turbo code scheme, $N = 200$ and $N = 400$.

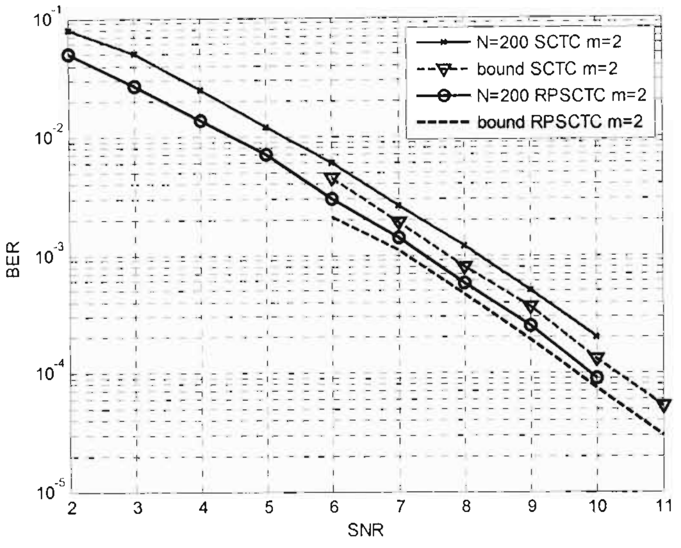


Figure 4.7D Simulation for SCTC and RPSCTC in a flat Rayleigh fading channel N=200, m=2, R=1/3, including the analytical bounds

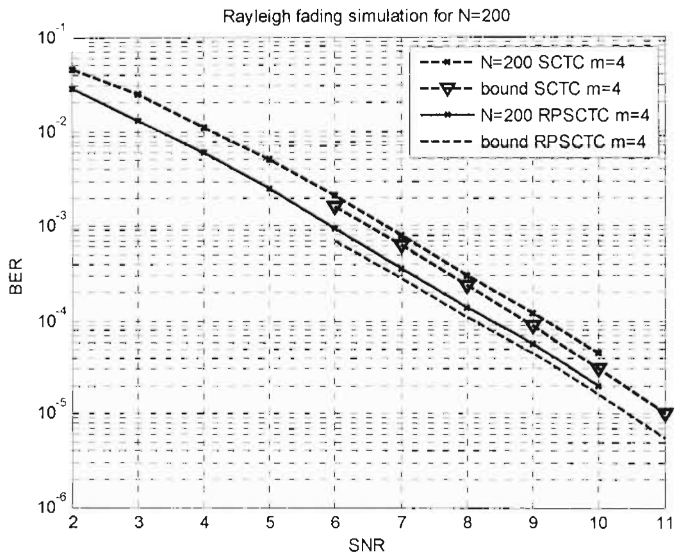


Figure 4.7E Simulation for SCTC and RPSCTC in a flat Rayleigh fading channel N=200, m=4, R=1/15, including the analytical bounds

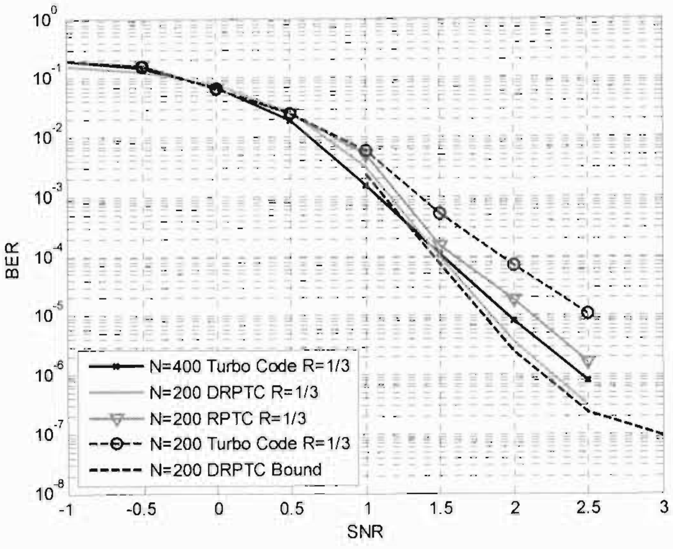


Figure 4.7F Graph comparing the simulations for TCs, N=200 and N=400 with RPTC, N=200 and DRPTC, N=200, plus the analytical bound, for the AWGN channel, at a code rate R=1/3

The RPTC scheme for $N = 200$ is then considered together with the DRPTC scheme and its analytical bound. DRPTC outperforms the $N = 200$ TC and the $N = 200$ RPTC scheme from a SNR of approximately 0.7 dB. The plot exhibits a behaviour similar to that of repeat-punctured turbo codes where, better performances are achieved at moderate to higher SNRs and turbo code type performance is achieved at the lower SNRs. The DRPTC scheme begins to outperform the $N = 400$ TC scheme at a SNR of almost 1.6 dB. Figure 4.7G presents similar results to Figure 4.7F, for AWGN, differing only by the message sequence frame lengths at the inputs. All simulations are undertaken for frame lengths twice those considered in Figure 4.7F. Again a similar type of performance can be noted. The $N = 400$ curve for DRPTC converges to the bound and outperforms all other schemes at higher SNRs.

Figure 4.7H shows the plots for SCTC, RPSCTC and DRPSCTC together with their analytical bounds, for the AWGN channel. These are all low-rate schemes suitable for spread-spectrum applications. The input message sequence frame length is $N = 200$ and the rate of the code is $R = 1/15$ corresponding to a constraint length $K = 5$. DRPSCTC starts to outperform RPSCTC at around 0.25 dB.

At a SNR of 1 dB DRPSCTC is at a BER of approximately 6×10^{-8}.

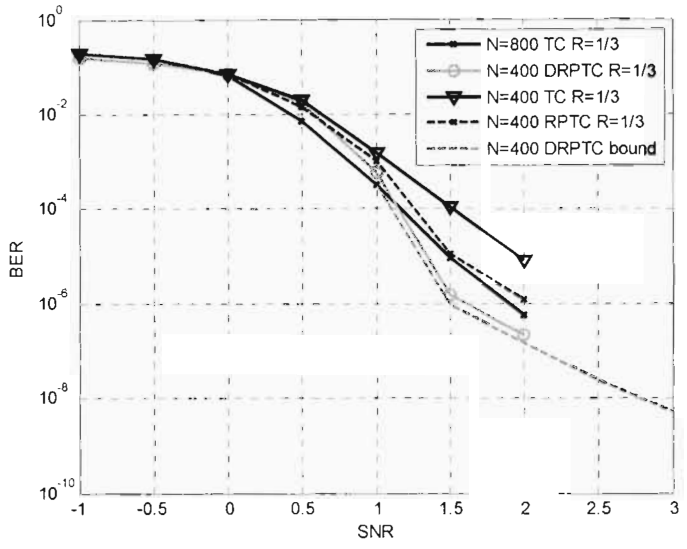


Figure 4.7G Graph comparing the simulations for TCs, N=400 and N=800 with RPTC, N=400 and DRPTC, N=400 plus the analytical bound, for the AWGN channel, at a code rate R=1/3

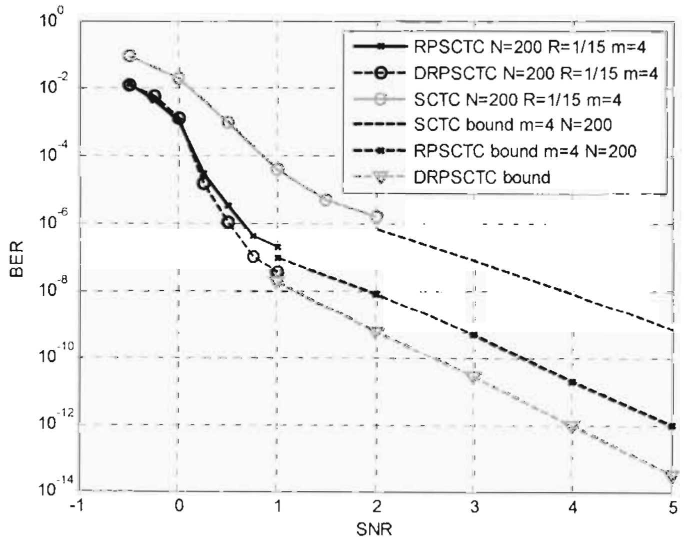


Figure 4.7H Graph comparing the simulations for SCTCs, N=200 with RPSCTC, N=200 and DRPSCTC, N=200 plus the analytical bounds, for the AWGN channel, at a code rate R=1/15

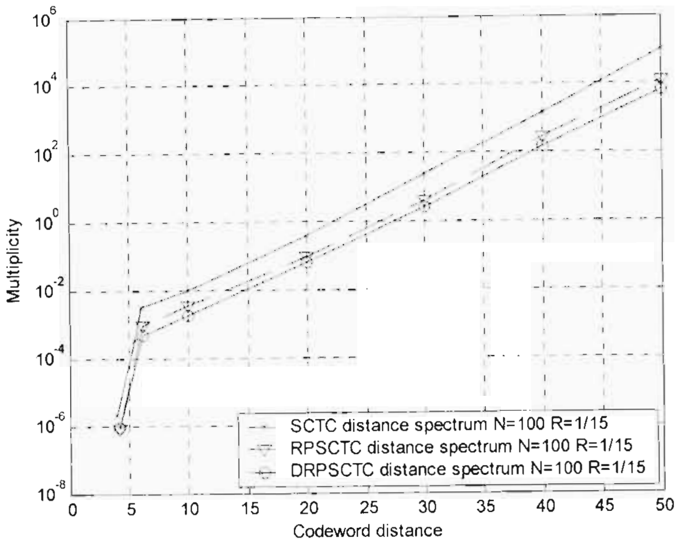


Figure 4.7I Distance spectrum for SCTC, RPSCTC, DRPSCTC, N=100, R=1/15

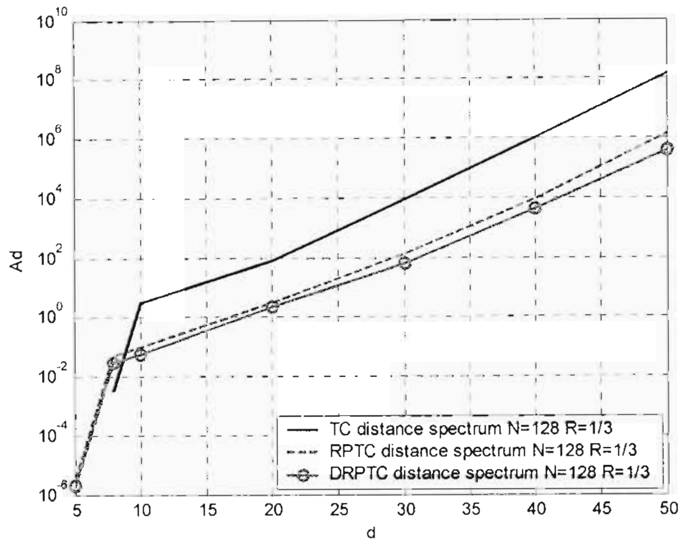


Figure 4.7J Distance spectrum for TC, RPTC, DRPTC, N=128, R=1/3

The bounds show that the coding gain achieved from DRPSCTC continuously increases with an increase in SNR.

Figures 4.7I and 4.7J show the codeword distance spectrums for frame lengths of $N = 100$ and $N = 128$ respectively. In Figure 4.7I the distance spectrums for DRPSCTC and RPSCTC are compared with that of SCTC.

The distance spectrum verifies the performance improvement shown in the previous graphs. It can be seen that the number of codewords with low weights (small distances, d) has been decreased and the spectrum has been shifted to the right due to there being a larger amount of high-weight codewords present.

Figure 4.7J serves to compare the distance spectrums for the classical turbo code, the repeat-punctured turbo code and the dual-repeat-punctured turbo code. The DRPTC spectrum is slightly lower, i.e., lower multiplicities, than RPTC. It can also be seen that there is a smaller number of low Hamming distance codewords and a larger number of codewords at higher multiplicities. As we move to higher weight codewords the multiplicity begins to diverge even further for DRPTC compared to RPTC.

The minimum distance for the single-repeat and dual-repeat schemes is lower than that of the conventional turbo codes. This is due to the puncturing that is required prior to transmission to recover the loss in code rate after repetition and encoding [13].

In this chapter, three new error-correction coding schemes have been introduced, RPSCTCs, DRPSCTCs and DRPTCs. Motivated by the near-Shannon-capacity performance of turbo codes, these schemes were adapted from the parallel concatenated structure of the conventional turbo code. Various simulations were undertaken for each of the schemes. Methods for deriving the union bounds were also presented and supported the BER performances reported. The distance spectrum was evaluated and plotted and served to motivate for the BER performances exhibited by these codes.

Chapter 5
Conclusion

5.1 Research Achievements

In this dissertation three new coding schemes or structures have been presented. These were Repeat-Punctured Superorthogonal Convolutional Turbo Codes, Dual-Repeat-Punctured Turbo Codes and Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes.
These schemes were all researched, motivated by turbo codes which, exhibits performances a few tenths of a decibel from the Shannon theoretical limit. For each of the three schemes the following have been investigated:

Repeat-Punctured Superorthogonal Convolutional Turbo Codes:

1. The QoS in terms of the BER versus SNR for the additive white Gaussian noise channel
2. The BER versus SNR performance in the NLOS or flat Rayleigh fading channel.
3. The codeword distance spectrum.
4. A method for the performance evaluation for the code using transfer function bounding techniques.

Dual-Repeat-Punctured Turbo Codes:

1. The bit-error rate versus signal-to-noise ratio performance curve for the AWGN channel.
2. Evaluation of the codeword distance spectrum using weight enumerating functions (WEFs).

3. A sound method for evaluating the analytical union bounds in the additive white Gaussian noise channel.

Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes:

1. The BER versus SNR performance exhibited in the additive white Gaussian channel.
2. Evaluation of the weight spectrum for particularly smaller Hamming distances.
3. A sound method for the analytical performance evaluation using transfer function bounding techniques for the AWGN channel.

The following table summarizes the performances exhibited at specific signal-to-noise ratios for each of the schemes presented in this thesis.

Table 5.1.1 Summary of simulated results for SNR=2, 2.5

		Bit Error Rate		
	SNR	**Turbo Code**	**Repeat-Punctured Turbo Code**	**Dual-Repeat-Punctured Turbo Code**
$N=200$ $R=1/3$	2	$\approx 7e^{-5}$	$\approx 2e^{-5}$	$\approx 3e^{-6}$
	2.5	$\approx 1e^{-5}$	$\approx 1.5e^{-6}$	$\approx 3e^{-7}$

Table 5.1.2 Summary of simulated results for SNR=0.5, 1 for the low-rate codes

		Bit Error Rate		
	SNR	**Turbo Code**	**Repeat-Punctured Turbo Code**	**Dual-Repeat-Punctured Turbo Code**
$N=200$ $R=1/15$	0.5	$\approx 9.7e^{-4}$	$\approx 3.6e^{-6}$	$\approx 1.1e^{-6}$
	1	$\approx 4e^{-5}$	$\approx 2e^{-7}$	$\approx 3.9e^{-8}$

From the tabulated results it is clear that the new schemes presented in this dissertation exhibit better performances than the conventional schemes discussed in Chapters 1 and 2. However, this is at the expense of computational complexity at the decoder. It is important to note that the improved performance that SCTC and consequently RPSCTC and DRPSCTC exhibit is achieved at the expense of bandwidth expansion and complexity and would therefore be ideal for power-limited, satellite communication links or interference-limited systems.

Another problem with the superorthogonal-based schemes would be the fact that the rate of the code is exponentially related to the memory depth chosen for the constituent encoders. Thus many more extravagant puncturing strategies could be investigated to yield better BER performances.

Table 5.1.3 shows specific input sequence weights and the corresponding output sequence weights yielded by the TC, RPTC and DRPTC schemes. Utilizing a random or uniform interleaver, it is evident from the table, that the RPTC and DRPTC schemes produce at least one parity sequence, for a typical problematic sequence of weight 1 or 2, with high Hamming weight. Thus the schemes employing the repetition technique are prone to produce high Hamming weight codewords more frequently compared to the conventional turbo code structure.

Table 5.1.3 Hamming weights for specific message sequences

	Hamming weight					
Input sequence weight	TC		RPTC		DRPTC	
	i/p	o/p	i/p	o/p	i/p	o/p
1	1	low	1	low	2	low
2	2	low	4	high	4	high

5.2 Future Work

In addition to the work covered in this thesis the following aspects can be extended on:

- Two new low-rate coding schemes have been investigated in this thesis. Their application to a multi-user detection scheme, Interleave Division Multiple Access

(IDMA), an extension of the commonly used Code Division Multiple Access (CDMA) scheme can be investigated to yield the performances achievable.

- The dual schemes, Dual-Repeat-Puncture Turbo Codes and Dual-Repeat-Punctured Superorthogonal Convolutional Turbo Codes can still be investigated for the Non-Line of Sight (NLOS) channels modeled by the Rayleigh probability distribution.
- One of the emerging fields of research in wireless communications is the use of space and time for multiple antenna transmission and co-operative, adaptive multiple receiving networks. The application of the three new coding structures introduced in this thesis can be evaluated for improved performances.
- A thorough method for the complexity analysis of the schemes presented in this thesis.

5.3 Contribution to the Literature

Conference Papers

1. N. Pillay, H. Xu, F. Takawira, "Repeat-Puncture Superorthogonal Convolutional Turbo Codes in AWGN and Flat Rayleigh Fading Channels," SATNAC 2007, Mauritius.
2. N. Pillay, H. Xu, F. Takawira, "Repeat-Puncture Superorthogonal Convolutional Turbo Codes in AWGN," Africon IEEE, Namibia, Sept. 2007.

Journal papers still under the reviewing process

1. N. Pillay, H. Xu, F. Takawira, "Repeat-Punctured Superorthogonal Convolutional Turbo Codes in AWGN and Flat Rayleigh Fading Channels," EURASIP Journal on Wireless Communications and Networking, submitted November 2007.
2. N. Pillay, H. Xu, F. Takawira, "Dual-Repeat-Punctured Turbo Codes in AWGN Channels," Hindawi Research Letters on Communications, submitted November 2007.

References

[1] B.Sklar, Digital Communications. Fundamentals and Applications. Beijing 2001.

[2] S. Benedetto and G. Montorsi, "Performance evaluation of parallel concatenated codes," in *Proc. IEEE Int. Conf. Commun.*, ICC'95, Seattle,WA, June 1995, vol. 2, pp. 663-667.

[3] D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications," The Telecommunications and Data Acquisition Progress Report 42-121, January-March 1995, Jet Propulsion Laboratory, Pasadena, California, May 15, 1995, pp. 66-77.

[4] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. conf. Commun.*, ICC'93, Geneva, Switzerland, May 1993, vol. 2, pp. 1064-1070.

[5] D. Divsalar, F. Pollara, "Serial and Hybrid Concatenated Codes with Applications," Jet Propulsion Laboratory, California institute of Technology.

[6] A. H. Mugaibel, M. A. Kousa, "Understanding Turbo Codes," King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

[7] Benedetto, Sergio and Montorsi, Guido, "Unveiling turbo-codes: some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, No. 2, March 1996, pp. 409-428.

[8] H. R. Sadjadpour, N. J. A. Sloane, M. Salehi, G. Nebe, "Interleaver Design for Turbo Codes," *IEEE Journal on Selected Areas in Communications*, Vol. 19, No. 2, May 2001, pp. 831-837.

[9] H. Taub, D. L. Schilling, Principles of Communication Systems, The City College of New York, 1986.

[10] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, Vol. IT-20, pp. 248-287, Mar. 1974.

[11] G. Zhu, "Performance Evaluation of Turbo Codes," Queen's University, Kingston, Ontario, Canada, Sept. 1998.

[12] C. Berrou, A. Glavieux, " Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. On Communications*, Vol. 44, No. 10, Oct. 1996.

[13] Y. Kim, J. Cho, W. Oh and K. Cheun, "Improving the performance of turbo codes by repetition and puncturing," Division of Electrical and Computer Engineering, Pohang University of Science and Technology, 1996.

[14] P. Komulainen and K. Pehkonen, "Performance evaluation of Superorthogonal Turbo Codes in AWGN and flat Rayleigh fading channels," in *IEEE Journ. On Sel. Areas in Commun.*, February 1998, vol. 16, no.2, pp. 196-205.

[15] D. Divsalar, F. Pollara, "Turbo-codes for PCS applications," in *Proc. IEEE Int. Conf. Commun.*, ICC'95, Seattle, WA, June 1995, vol. 1, pp. 54-59.

[16] M.C. Valenti, "Turbo Codes and Iterative Processing," Mobile and Portable Radio Research Group, Virginia Polytechnic Institute and State University, Blacksburg, USA.

[17] P. Komulainen and K. Pehkonen, "A low-complexity Superorthogonal Turbo-Code for CDMA Applications," in *Proc. IEEE Int. Symp. Personal, Indoor, Mobile Radio Commun.*, PIMRC '96, Taipei, Taiwan, R.O.C.,Oct. 1996, vol. 2, pp. 369-373.

[18] K. Rikkinen, "Comparison of very low rate coding methods for CDMA radio communicatios system," in *Proc. IEEE Int. Symp. Spread Spectrum Techniques Appl., ISSSTA '94,* Oulu, Finland, July 1994, vol. 1, pp. 268-272.

[19] K. Pehkonen, P. Komulainen, "A superorthogonal turbo-code for CDMA applications," in *Proc. IEEE Int. Symp. Spread Spectrum Techniques Appl., ISSSTA '96,* Mainz, Germany, Sept. 1996, vol. 3, pp.580-584.

[20] J. D. Anderson, "Turbo codes extended with outer BCH code," Electronic Letter, Vol. 32, no.22, pp. 2059-2060, Oct. 1996.

[21] K. R. Narayanan and G. L. Stuber, "Selective serial concatenation of turbo codes," *IEEE Commun. Lett.*, Vol/ 1, pp.136-138, Sept. 1997.

[22] I. Chatzigeorgiou, M. R.D. Rodrigues, I.J. Wassell and R. Carrasco, "Pseudo-random Puncturing: A technique to lower the error floor of turbo codes, " ISIT 07, 2007

[23] Carl Fredrik Leanderson, "Low-rate Turbo Codes," Department of Applied Electronics, Lund University, SE-221 00 LUND, Sweden, 1998.

[24] Y. Ould-Cheikh-Mouhamedou and S. Crozier, "Improving the Error Rate Performance of Turbo Codes using the Forced Symbol Method," *IEEE Communications Letters*, Vol. 11, pp. 616-618, No. 7, July 2007.

[25] Lance C. Perez, Jan Seghers, Daniel J. Costello Jr., "A Distance Spectrum Interpretation of Turbo Codes," *IEEE Transactions on Information Theory*, Vol. 42, No. 6, November 1996.

[26] Pillay Narushan, Xu HongJun, Takawira Fambirai, "Repeat-Puncture Superorthogonal Convolutional Turbo Codes in AWGN and Flat Rayleigh Fading Channels," SATNAC 2007.

[27] D. Divsalar, S. Dolinar, and F. Pollara, "Transfer Function Bounds on the Performance of Turbo Codes," TDA Progress Report 42-122, August 15, 1995, Communications Systems and Research Section, R. J. McEliece California Institute of Technology, pp. 44-55.

[28] H. Xu, F. Takawira, "Performance bounds of Turbo Codes with Redundant Input," School of Electrical and Computer Engineering, Inha University, Korea, and Guilin Institute of Electronic Technology, P.R. China, School of Electrical and Electronic Engineering, University of Natal, South Africa.

[29] W. C. Jakes, "Microwave Mobile Communications," New York: John Wiley and Sons Inc, 1994.

[30] J. Qi, "Turbo Codes in IS-2000 Code Division Multiple Access Communications under fading," Thesis submitted to Department of Electrical and Computer Engineering and the Faculty of the Graduate School of Wichita State University, 1999.

[31] C. Heegard, S. B. Wicker, "Turbo Coding," Boston, Kluwer Academic Publishers, 1999.

[32] Y. Ould-Cheikh-Mouhamedou, "On Distance Measurement Methods For Turbo Codes," Ph.D. Thesis, November 2005.

Appendices

Appendix A.1

Derivation of the soft log-likelihood ratio

Consider that the sequence prior to transmission is

$$x_1^N = \{x_1,\ldots,x_N\} \tag{A.1a}$$

and the received sequence at the receiver front-end is

$$y_1^N = \{y_1,\ldots,y_N\} \tag{A.1b}$$

Now the likelihood function can be expressed as

$$\lambda_k^i = P\left(d_k = i \mid y_1^N\right) = \sum_{v=0}^{M} \lambda_k^i(v) \quad \forall\ k = [1,\ldots,N], \forall\ i \in [0,1] \tag{A.1c}$$

Taking into account the states v, the likelihood function can be further written as

$$\lambda_k^i(v) = P\left(d_k = i, S_k = v \mid y_1^N\right) \tag{A.1d}$$

Following from this, the LLR can be defined as

$$L(d_k \mid y) = \log\left[\frac{\sum_m \lambda_k^{i=1}(v)}{\sum_m \lambda_k^{i=0}(v)}\right], \qquad \forall\ k = [1,\ldots,N] \tag{A.1e}$$

Now, using Bayes theorem (A.1d) can be written as

$$\lambda_k^i(v) = \frac{P\left(d_k = i, S_k = v, y_1^k, y_{k+1}^N\right)}{P\left(y_1^N\right)} \tag{A.1f}$$

$$= \frac{P\left(y_{k+1}^N \mid d_k = i, S_k = v, y_1^k\right)P\left(d_k = i, S_k = v, y_1^k\right)}{P\left(y_1^N\right)} \tag{A.1g}$$

then since $S_k = v$ and y_{k+1}^N are independent of d_k and y_1^k

$$\lambda_k^i(v) = \frac{P\left(y_{k+1}^N \mid S_k = v\right)P\left(d_k = i, S_k = v, y_1^k\right)}{P\left(y_1^N\right)} \tag{A.1h}$$

and

$$P(d_k = i, S_k = v \mid y_1^N) = P(d_k = i, S_k = v, y_k, y_1^{k-1}) \tag{A.1i}$$

$$= \sum_{v'} P(d_k = i, S_k = v, y_k, S_{k-1} = v', y_1^{k-1})$$

$$= \sum_{v'} P(d_k = i, S_k = v, y_k \mid S_{k-1} = v', y_1^{k-1}) P(S_{k-1} = v', y_1^{k-1})$$

$$= \sum_{v'} P(d_k = i, S_k = v, y_k \mid S_{k-1} = v') P(S_{k-1} = v', y_1^{k-1})$$

(A.1j)

If we substitute (A.1j) in (A.1h) we get

$$\lambda_k^i(v) = \frac{\sum_{v'} P(d_k = i, S_k = v, y_k \mid S_{k-1} = v') P(y_{k+1}^N \mid S_k = v) P(S_{k-1} = v', y_1^{k-1})}{P(y_1^N)} \tag{A.1k}$$

Next if we consider the forward recursion $\alpha_k(m)$ over the trellis

$$\alpha_k(v) = P(S_k = v \mid y_1^k) \tag{A.1l}$$

And the backward recursion $\beta_k(v)$ defined as

$$\beta_k(v) = \frac{P(y_{k+1}^N \mid S_k = v)}{P(y_{k+1}^N \mid y_1^k)} \tag{A.1m}$$

The branch metric $\delta_i(y_k, v, v')$ is defined as

$$\delta_i(y_k, v, v') = P(d_k = i, S_k = v, y_k \mid S_{k-1} = v') \tag{A.1n}$$

Now if $P(y_1^N)$ was the product of $P(y_1^{k-1})$ and $P(y_{k+1}^N \mid y_1^k)$

Then

$$\lambda_k^i(v) = \sum_{m'} \delta_i(y_k, v', v) \alpha_{k-1}(v') \beta_k(v) \tag{A.1o}$$

However since $P(y_1^N)$ can only be factored as

$$P(y_1^k, y_{k+1}^N) = P(y_{k+1}^N \mid y_1^k) P(y_1^k) \tag{A.1p}$$

Then the LLR can be expressed as

$$L(d_k \mid y) = \log \left[\frac{\sum_v \sum_{v'} \delta_1(y_k, v', v) \beta_k(v) P(S_{k-1} = v', y_1^{k-1}) / P(y_1^k)}{\sum_v \sum_{v'} \delta_0(y_k, v', v) \beta_k(v) P(S_{k-1} = v', y_1^{k-1}) / P(y_1^k)} \right] \tag{A.1q}$$

It is clear that $P(y_1^k)$ does not effect the summation of v and v' so it can be replaced by $P(y_1^{k-1})$ to give

$$L(d_k \mid y) = \log\left[\frac{\sum_{v}\sum_{v'}\delta_1(y_k, v', v)\beta_k(v)\alpha_{k-1}(v')}{\sum_{v}\sum_{v'}\delta_0(y_k, v', v)\beta_k(v)\alpha_{k-1}(v')}\right] \tag{A.1r}$$

Appendix A.2

Derivation of the forward state metric

Consider the expression for the forward state metric

$$\alpha_k(v)=\sum_{m'}\sum_{i=0}^{1}P\left(d_{k-1}=i,S_{k-1}=v',y_1^{k-1}\mid S_k=v\right) \tag{A.2a}$$

Using Bayes theorem and writing y_1^{k-1} as $\{y_1^{k-2},y_{k-1}\}$ then

$$\alpha_k(v)=\sum_{v'}\sum_{i=0}^{1}P\left(y_1^{k-2}\mid S_k=v,d_{k-1}=i,S_{k-1}=v',y_{k-1}\right)P\left(d_{k-1}=i,S_{k-1}=v',y_{k-1}\mid S_k=v\right) \tag{A.2b}$$

$$=\sum_{i=0}^{1}P\left(y_1^{k-2}\mid S_{k-1}=\varsigma(i,v)\right)P\left(d_{k-1}=i,S_{k-1}=\varsigma(i,v),y_{k-1}\right) \tag{A.2c}$$

where $\varsigma(i,v)$ is the past state from state v from the previous branch corresponding to input i. From (A.2c) $\alpha_k(v)$ can be written as follows

$$\alpha_k(v)=\sum_{i=0}^{1}\alpha_{k-1}(\varsigma(i,v))\delta_{k-1}(i,\varsigma(i,v)) \tag{A.2d}$$

Figure A.2A graphically shows the use of equation (A.2d) to calculate the forward state metric.

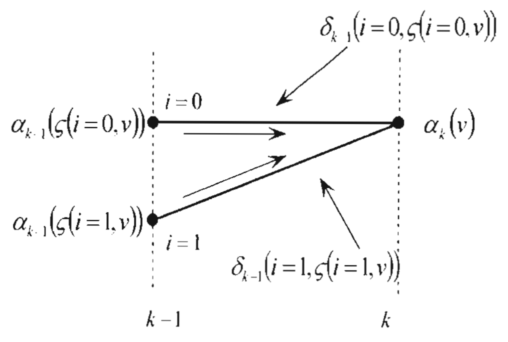

$\delta_{k-1}(i=0,\varsigma(i=0,v))$
$i=0$
$\alpha_{k-1}(\varsigma(i=0,v))$
$\alpha_k(v)$
$\alpha_{k-1}(\varsigma(i=1,v))$
$i=1$
$\delta_{k-1}(i=1,\varsigma(i=1,v))$
$k-1$
k

Figure A.2A Calculation of the forward state metric

Appendix A.3

Derivation of the reverse state metric

The reverse state metric can be written as

$$\beta_k(v) = P\left(y_k^N \mid S_k = v\right) \tag{A.3a}$$

$$= P\left(y_k, y_{k+1}^N \mid S_k = v\right) \tag{A.3b}$$

The reverse state metric can be expressed as a summation of all possible probabilities/transitions.

$$\beta_k(v) = \sum_{v'} \sum_{i=0}^{1} P\left(d_k = i, S_{k+1} = v', y_k, y_{k+1}^N \mid S_k = v\right) \tag{A.3c}$$

Utilizing Bayes theorem again

$$\beta_k(v) = \sum_{v'} \sum_{i=0}^{1} P\left(y_{k+1}^N \mid S_k = v, d_k = i, S_{k+1} = v', y_k\right) P\left(d_k = i, S_{k+1} = v', y_k \mid S_k = v\right) \tag{A.3d}$$

Now consider the next state as $\phi(i,v)$ given an input i and state v. We can now replace $S_{k+1} = v'$ with $S_k = v$ considering the condition that $S_{k+1} = \phi(i,v)$ since $S_k = v$ and $d_k = i$.

$$\beta_k(v) = \sum_{i=0}^{1} P\left(y_{k+1}^N \mid S_{k+1} = \phi(i,v)\right) P\left(d_k = i, S_k = v, y_k\right) \tag{A.3e}$$

which can be written as

$$\beta_k(v) = \sum_{i=0}^{1} \delta_k(i,v) \beta_{k+1}(\phi(i,v)) \tag{A.3f}$$

Figure A.3A shows the use of equation (A.3f) graphically for calculation of the reverse state metric.

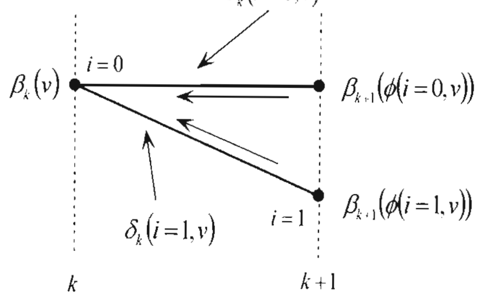


Figure A.3A Calculation of the reverse state metric

Appendix A.4
Derivation of the branch metric

Starting with the expression for the branch metric $\delta_k(i,v)$

$$\delta_k(i,v) = P(d_k = i, S_k = v, y_k) \tag{A.4a}$$

And again using Bayes theorem can be expanded as

$$\delta_k(i,v) = P(y_k \mid d_k = i, S_k = v)P(S_k = v \mid d_k = i)P(d_k = i) \tag{A.4b}$$

The current state can be assumed to be independent of the current input since the noise affecting the data and the parity bits is independent and can be any of 2^m states. m is the memory of the code. So it can be shown that

$$P(S_k = v \mid d_k = i) = \frac{1}{2^m} \tag{A.4c}$$

and following from here

$$\delta_k(i,v) = P(p_k \mid d_k = i, S_k = v)P(q_k \mid d_k = i, S_k = v)\frac{\pi_k^i}{2^m} \tag{A.4d}$$

π_k^i is the a-priori probability of d_k.

From probability theory we know that [1]

$$P(G_k = g_k) = P_{g_k}(g_k)dg_k \tag{A.4e}$$

The branch can now be written as (A.4f) using the Gaussian pdf for AWGN channels.

$$\delta_k(i,v) = \frac{\pi_k^i}{2^m\sqrt{2\pi\sigma}}\exp\left[-0.5\left(\frac{p_k - a_k^i}{\sigma}\right)^2\right]dp_k \frac{1}{\sqrt{2\pi\sigma}}\exp\left[-0.5\left(\frac{q_k - b_k^{i,v}}{\sigma}\right)^2\right]dq_k \tag{A.4f}$$

a_k, b_k represent the data and parity bits sent and p_k, q_k represent the data and parity bits received. σ is the square root of the variance or the standard deviation. Equation (A.4f) can be written as

$$\delta_k(i,v) = A_k \pi_k^i \exp\left[\frac{1}{\sigma^2}\left(p_k a_k^i + q_k b_k^{i,v}\right)\right] \tag{A.4g}$$

The differentials are absorbed into A_k, a constant. π_k is known as the a-priori probability ratio, i.e., $\pi_k = \frac{\pi_k^{i=1}}{\pi_k^{i=0}}$.

And for the AWGN channel the channel measurement is defined by the logarithm of the ratio of likelihood functions

$$L_{channel} = \log\left(\frac{P(y_k \mid d_k = +1)}{P(y_k \mid d_k = -1)}\right) \tag{A.4h}$$

Now substituting into the Gaussian distribution function and simplifying

$$L_{channel} = -0.5\left(\frac{y_k - 1}{\sigma}\right)^2 + 0.5\left(\frac{y_k + 1}{\sigma}\right)^2 = \frac{2}{\sigma^2} y_k \tag{A.4i}$$

Appendix B
Rayleigh Fading for Turbo Codes

Consider a signal duration interval of length T, if an interval $T >> T_{spread}$ is chosen, where T_{spread} is the multipath spread of the channel, then there should be a negligible amount of inter-symbol inference (ISI) which is a deleterious input to the decoder.

For a bandwidth $B_w = \frac{1}{T}$ and for $T >> T_{spread}$ this would imply that $B_w << \frac{1}{T_{spread}}$. This implies a frequency non-selective channel or a flat Rayleigh fading (FRF) channel. In a FRF channel all the frequency components undergo the same attenuation and phase shift in the channel. For a Rayleigh fading channel the transfer function is defined by

$$TF_{rayleigh} = \alpha(t)e^{j\phi(t)} \tag{Ba}$$

were $\alpha(t)$ represents the envelope of the function and the remaining term the phase of the channel.

For a zero-mean complex valued Gaussian random process, $\alpha(t)$ is Rayleigh distributed for fixed values of t and $\phi(t)$. Figure B shows the model for the Rayleigh fading channel. The input signal $x(t)$ is subjected to the effects of fading i.e. attenuation and phase shift, this is denoted by $A = \alpha(t)e^{j\phi(t)}$. The noise component $n(t)$ is Gaussian, i.e., zero mean and a variance of $N_0/2$. The resultant signal is $y(t) = \alpha(t)e^{j\phi(t)}x(t) + n(t)$. For binary phase shift-keying (BPSK) modulation it can be assumed that $\phi(t)$ will be known at the decoder. For a channel with side information, A is known at the decoder.

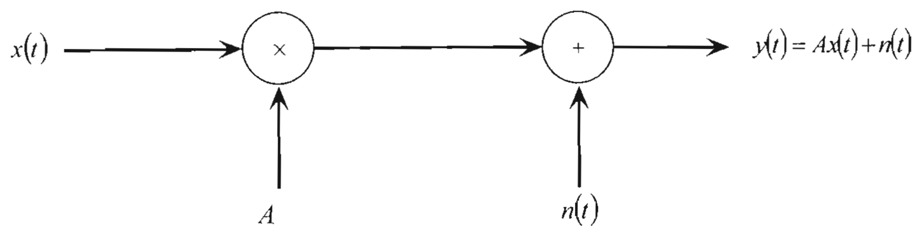


Figure B Model of the Rayleigh fading channel

The Jake's model [29] can be used to generate the fading gain. For a frequency non-selective channel, correlated waveforms are used, referred to as correlated fading. Assume that the scatterers, objects in the line of path, are uniformly distributed around a point at angles α_n with k rays produced by each scatterer. The Doppler shift on each ray n is defined by $f_n = f_d \cos\alpha_n$ and with M scatterers the Rayleigh fading is modeled by

$$\Re = 2\sqrt{2}\left(\sum_{n=1}^{\mathrm{M}}(\cos\beta_n + j\sin\beta_n)\cos(2\pi f_n t + \theta_{n,k}) + \frac{1}{\sqrt{2}}(\cos\alpha + j\sin\alpha)\cos(2\pi f_d t)\right) \tag{Bb}$$

where B_n is defined by

$$\beta_n = \frac{\pi n}{\mathrm{M}+1} \tag{Bc}$$

and is chosen such that there is zero cross-correlation between the real and imaginary parts of (Bb). For a frequency non-selective or flat Rayleigh fading channel, multiple waveforms are not needed, therefore $\theta_{n,k}$ can be assumed to be zero. For an uncorrelated channel or frequency selective channel, multiple waveforms are required and $\theta_{n,k}$ is defined by

$$\theta_{n,k} = \beta_n + \frac{2\pi(k-1)}{\mathrm{M}+1} \tag{Bd}$$

α is also usually set to zero.

$\delta(i,v)$ needs to be changed for decoding 'faded' sequences. Referring to Appendix A, for side information (SI) $\delta(i,v)$ needs to depend on α_k^s, α_k^p. The manipulated branch metric is then given by

$$P(y_k^s \mid d_k = i, \alpha_k^s) P(y_k^p \mid d_k = i, \alpha_k^p, S_k = v, S_{k-1} = v') q(d_k = i \mid S_k = v, S_{k-1} = v') \pi(S_k = v \mid S_{k-1} = v')$$

where

$$P(y_k^s \mid d_k = i, \alpha_k^s) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k^s - \alpha_k^s(2i-1))^2}{2\sigma^2}} \tag{Be}$$

and for the parity bits

$$P(y_k^p \mid d_k = i, \alpha_k^p, S_k = v, S_{k-1} = v') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k^p - \alpha_k^p(2i-1))^2}{2\sigma^2}} \tag{Bf}$$