Irregular block turbo codes for communication systems.

Sholiyi, Abiodun Olugbenga

FOR COMMUNICATION SYSTEMS

Swansea University
Prifysgol Abertawe

ABIODUN OLUGBENGA SHOLIYI

COLLEGE OF ENGINEERING

SWANSEA UNIVERSITY

Submitted to Swansea University in fulfillment of the requirement for the degree

Doctor of Philosophy (Ph.D)

April 2011

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Acknowledgements

All the glory, power and majesty be unto God for having granted me the grace to complete this thesis successfully in good health. Thank you Jesus!

This thesis is based on research partly funded by my family and the Overseas Research Award (ORS) to whom I am very grateful.

I would like to thank Professor Timothy O'Farrell for his help and guidance during this research. I would also like to thank Dr. Pavel Loskot, my second supervisor for his help at the completion stage of this thesis.

I would like to give special thanks to Mr. Akin Bamgbelu for his financial and moral support he gave me since my primary school to the successful completion of this Ph.D. Your labour of love will not be forgotten. God bless you abundantly. I would also like to give special thanks to my lovely and exceptionally understanding wife Abimbola Sholiyi and my daughter Moyinoluwalogo Sholiyi for having endured the time I spent away from them during this research both at home and away from home. Thank you very much for your powerful encouraging words and the prayers. God bless you big time.

Special thanks also go to my mom Mrs. Aderonke Sholiyi who on her knees, through the nights for more than 3 years interceded for the successful completion of this research and with her financial resources fully supporting this research. Thank you very much for all the powerful prayer points and encouragements as well. Your labour of love will not be forgotten. God bless you abundantly. Special thanks also go to Professor Sola Adeyemi and Mrs. Bola Adeyemi for their financial and moral support. It is marvelous in our eyes. Thank you for your words of encouragement as well and your prayers. God bless you richly. Special thanks also go to Mrs. Olubukola Ehinmowo, Mr. Akinloye Sholiyi for their financial and moral support during this research. I would also like to give special thanks to Dr. Bill Cox, Dr. Anna Cox for their support during this research as well as Dr. R. Maunder of the school of Electronics and Computer Science University of Southampton for his help in understanding the EXIT chart which was important to

this research. I would also like to specially thank late Kehinde Ayodeji, for his helped during this research.

Special thanks also go to Suleiman Adegoke, Oladapo Adeeko, Gboyega Adeloye, Ibukun Adeloye and Richard Olaiya for your support and kind words of encouragement. God bless you all big time. During my time at Swansea, I have made lots of friends. They have made my time during this research very enjoyable. In particular I would like to mention Adebayo Aderonmu, Bamidele Adenipekun, Olugbenga Olubodun, Obinna Anyadike, Ayoagun Oluwole, Salim Abukharis, Keneth Nwizege, Jafar Alzubi and Saif Alnawayseh. Thank you guys.

This thesis is dedicated to late Simeon Oluyemi Sholiyi (my lovely dad) and late Ayodele Temitope Sholiyi (my lovely younger brother)

Declarations and statements

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed.. ..... (candidate)

Date.....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated.

Signed ..... (candidate)

Date.....

STATEMENT 2

I hereby give consent to my thesis, if accepted, to be available for photocopying, inter-library loan and for the title and summary to be made available to outsider organizations.

Signed ..... (candidate)

Date.....

Abstract

This thesis proposes new algorithms for and studies the bit error rate (BER) performance of iterative decoders used as channel codecs in today's communication systems. A characterization of high speed downlink packet access (HSDPA) capacity using the EXIT chart technique is examined, with the Convolutional Turbo Code (CTC) as its channel codec. This showed that the summation of the area underneath the upper and lower decoders of a CTC equals the capacity of the channel. The bit error rate (BER) performance of the CTC and the Block Turbo Code (BTC) were evalauted and compared for four different modulations schemes (BPSK, QPSK, 16QAM and 64QAM) with their corresponding distances to channel capacity.

Furthermore, a near capacity irregular Turbo Code (I-TC) is trialed as the channel codec in an HSDPA system. The designed I-TCs were used in higher order modulation schemes, unlike in previous versions of the I-TC designed by previous authors using BPSK. Previously designed I-TCs had long frame size which increases the system's complexity. In this thesis, short frame size I-TCs (5012 bits per frame unlike in previous versions by MacKay et. al. and Boutros et. al. where 2^{17} bits frame size were used) were designed using new puncturing patterns and new bit degree selection process. This impacted largely on the system's complexity (frame size reduction) as well the number of iterations required to achieve a low BER (an average of 25 iterations was used in this thesis unlike the 100 iterations required by Boutros and Sawaya). In comparison to the regular TC, the I-TC in general uses a single recursive Convolutional code and a single soft-input soft-output decoder. Also, the designed I-TC performs no worse than the regular TC and frequently better, for example, it is capable of achieving a coding gain of 1.3 dB over its corresponding regular TC, when used in an AWGN channel with 64QAM modulation and a 1.4 dB coding gain in a 3GPP multipath channel with 16QAM modulation unlike in previous versions of I-TCs by other authors with coding gain less than 0.8dB for an AWGN channel.

Iterative decoding of serially concatenated two-dimensional systematic block codes has been termed Block Turbo Code (BTC). This thesis describes an irregular Block Turbo Code (I-BTC) with code rates comparable to those of a Block Turbo Code when using the BCJR algorithm. In "regular Block Turbo Codes", the horizontal (or vertical) block is first encoded followed by the vertical (or horizontal) encoding. We construct a new and flexible channel codec, termed irregular Block Turbo Code, with information bits that participate in varying numbers of trellis sections which is then encoded horizontally (vertically) without vertical (horizontal) encoding. The decoding requires only one soft-input soft-output (SISO) decoder. In general, a reduction in the number of operations required to implement an I-BTC was achieved in comparison to the regular Block Turbo Code for the same BER of 10^{-5}. The performance of different Bose-Chaudhuri-Hocquenghem (BCH) I-BTCs for different modulation orders is given in an AWGN channel. The results are presented for a BPSK, QPSK, 16QAM and 64QAM modulation schemes using different block sizes. Performance in the AWGN channel shows that the I-BTC is capable of achieving a significant coding gain of 1.28 dB at a bit error rate (BER) of 10^{-5} over the regular block turbo code for 64QAM modulation. Simulation results also show that some of these codes perform within 0.49 dB of capacity for binary transmission over an AWGN channel. A significant 41.3% reduction in the number of operations required to achieve a low BER of 10^{-5} was achieved in the QPSK (255, 247)I-BTC coupled with a coding gain of 0.98 dB over its corresponding BTC. The achieved coding gain in the I-BTC is larger than the coding gain of any irregular code over its regular equivalent by previous authors. Potential applications of the I-BTCs are in digital communication channels with Gaussian models such as fixed Broadband (point to point and point to multi-point), VSAT modems (data and voice), optical fiber communication systems and other microwave point to point links.

The performance of these codes (the I-TCs and the I-BTCs designed in this thesis) are the best to date to the best of my knowledge in respect of smallest distance to capacity, coding gain, number of iterations, frame sizes and complexity.

List of Figures

1.1 A communication model as proposed by Shannon for an unreliable channel.....2

2.1 A CTC encoder structure.....17

2.2 Construction of a two dimensional Turbo Product Code.....19

4.2.1 Schematic of HSDPA system model.....32

4.2.2 The UMTS Turbo Code encoder structure.....34

4.2.3 Schematic of Turbo Code decoding structure 38

4.2.4 Example of a trellis structure for a systematic Convolutional code.....39

4.4.1 BER performance of CTC in BPSK.....44

4.4.2 BER performance of CTC in QPSK.....44

4.4.3 BER performance of CTC in 16QAM.....44

4.4.4 BER performance of CTC in 64QAM.....44

4.4.5 BER performance of CTC in a 3GPP multipath Vehicular A channel for QPSK.....48

4.4.6 BER performance of CTC in a 3GPP multipath Vehicular A channel for 16QAM.....48

4.4.7 BER performance of CTC in a 3GPP multipath Vehicular A channel for 64QAM.....48

4.4.8 (a) EXIT chart of a CTC at an E_b/N_0 of 0.4dB rate 1/3 with BPSK.....54

(b) EXIT chart of a CTC at an E_b/N_0 of 1.1dB rate 1/2 with BPSK.....54

(c) EXIT chart of a CTC at an E_b/N_0 of 0.6dB rate 1/3 with QPSK.....54

(d) EXIT chart of a CTC at an E_b/N_0 of 0.7dB rate 1/3 with QPSK.....54

(e) EXIT chart of a CTC at an E_b/N_0 of 1.18dB rate 1/2 with QPSK.....54

(f) EXIT chart of a CTC at an E_b/N_0 of 2.6dB rate 1/3 with 16QAM.....54

(g) EXIT chart of a CTC at an E_b/N_0 of 3.5dB rate 1/2 with 16QAM.....54

(h) EXIT chart of a CTC at an E_b/N_0 of 5.5dB rate 1/3 with 64QAM.....54

(i) EXIT chart of a CTC at an E_b/N_0 of 7.5dB rate 1/2 with 64QAM.....54

4.4.9 Schematic of a serially concatenated code.....59

4.5.1 BER performance for binary BTCs in BPSK.....59

4.5.2 BER performance for binary BTCs in QPSK.....59

4.5.3BER performance for binary BTCs in 16QAM.....60

4.5.4BER performance for binary BTCs in 64QAM.....60

4.6.1(a) EXIT chart for binary (15, 11)BTC at an E_b/N_0 of 4.5dB with BPSK.....63

(b) EXIT chart for binary (15, 11)BTC at an E_b/N_0 of 4.5dB with QPSK.....63

(c) EXIT chart for binary (15, 11)BTC at an E_b/N_0 of 7.7dB with 16QAM.....63

(d) EXIT chart for binary (15, 11)BTC at an E_b/N_0 of 13.0dB with 64QAM.....63

(e) EXIT chart for binary (31, 26)BTC at an E_b/N_0 of 3.5dB with BPSK.....63

(f) EXIT chart for binary (31, 26)BTC at an E_b/N_0 of 3.4dB with QPSK.....63

(g) EXIT chart for binary (31, 26)BTC at an E_b/N_0 of 7.1dB with 16QAM.....63

(h) EXIT chart for binary (31, 26)BTC at an E_b/N_0 of 12.4dB with 64QAM.....63

(i)EXIT chart for binary (63, 57)BTC at an E_b/N_0 of 3.55dB with BPSK.....63

(j)EXIT chart for binary (63, 57)BTC at an E_b/N_0 of 3.55dB with QPSK.....63

(k)EXIT chart for binary (63, 57)BTC at an E_b/N_0 of 7.05dB with 16QAM.....63

(l)EXIT chart for binary (63, 57)BTC at an E_b/N_0 of 12.55dB with 64QAM.....63

(m)EXIT chart for binary (127, 120)BTC at an E_b/N_0 of 3.69dB with BPSK.....63

(n)EXIT chart for binary (127, 120)BTC at an E_b/N_0 of 5.1dB with QPSK.....63

(o)EXIT chart for binary (127, 120)BTC at an E_b/N_0 of 7.78dB with 16QAM.....63

(p)EXIT chart for binary (127, 120)BTC at an E_b/N_0 of 12.48dB with 64QAM.....63

4.7.1 BER performance for Reed Solomon BTC in BPSK/QPSK modulation scheme..68
4.7.2 BER performance for Reed Solomon BTC in a 16QAM modulation scheme.....68
4.7.3 BER performance of Reed Solomon BTC in a 64QAM modulation scheme.....69
5.1(a) Equivalent encoding structures for a regular Turbo Code.....75
5.1(b) Encoding structure for an Irregular Turbo Code.....75
5.2 Iterative decoding structure of an Irregular Turbo Code.....77
5.3 BER performance of ITC and TC in an AWGN channel with a frame size of 1003 bits in four different modulation schemes, code rate 1/3.....79
5.4 BER performance of ITC and TC in an AWGN channel with a frame size of 5012 bits in four different modulation schemes, code rate 1/3.....79
5.5 BER performance of ITC and TC in an AWGN channel with a frame size of 10016 bits in four different modulation schemes, code rate 1/3.....79
5.6 BER performance of ITC and TC in an AWGN channel with a frame size of 20072 bits in four different modulation schemes, code rate 1/3.....79
5.7 BER performance comparisons between TCs and ITCs in an AWGN channel with BPSK.....81
5.8 BER performance comparisons between TCs and ITCs in an AWGN channel with QPSK.....81
5.9 BER performance comparisons between TCs and ITCs in an AWGN channel with 16QAM.....81
5.10 BER performance comparisons between TCs and ITCs in an AWGN channel with 64QAM.....81
5.11 Throughput versus SNR for TCs and ITCs in an AWGN channel for different modulation schemes.....84
5.12 BER performance comparisons between TCs and ITCs in a 3GPP Vehicular A multipath channel for QPSK.....85
5.13 BER performance comparisons between TCs and ITCs in a 3GPP Vehicular A multipath channel for 16QAM.....85

5.14 BER performance comparisons between TCs and ITCs in a 3GPP Vehicular A multipath channel for 64QAM.....85
6.1(a) Encoding structure for a TPC.....91
6.1(b) An equivalent encoding structures for a TPC.....91
6.1(c) Encoding structure for an Irregular Block Turbo Code.....91
6.1.1 Encoding a (67, 60)I-BTC (degree 2) from a (1 by 60) Information vector.....93
6.1.2 Iterative decoding structure of an Irregular Block Turbo Code.....95
6.1.3 Example of trellis structure of a systematic BCH.....96
6.3.1 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.....108
6.3.2 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK.....108
6.3.3 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.....108
6.3.4 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.....108
6.3.5 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.....109
6.3.6 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK.....109
6.3.7 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.....109
6.3.8 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.....109
6.3.9 (127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.....110
6.3.10 (127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK.....110
6.3.11 (127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.....110
6.3.12 (127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.....110
6.3.13 (255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.....111
6.3.14 (255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK.....111
6.3.15 (255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.....111
6.3.16 (255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.....111
6.3.17(63, 57)TPC with different I-BTC array sizes, BER versus E_b/N_0 for QPSK.....111
6.3.18(120, 127)TPC with different I-BTC array sizes, BER versus E_b/N_0 for QPSK..111
6.3.19(255, 247)TPC with different I-BTC array sizes, BER versus E_b/N_0 for QPSK..111
6.3.20(63, 57)TPC and equivalent I-BTC throughput per channel use versus SNR.....114
6.3.21(127, 120)TPC and equivalent I-BTC throughput per channel use versus SNR..114
6.3.22(255, 247)TPC and equivalent I-BTC throughput per channel use versus SNR..115

List of Tables

4.1 CTC Converging E_b/N_0 values 46
4.2 EXIT chart area properties of the UMTS Convolutional Turbo Code.....56
4.3 Various BTCs with different modulation schemes showing their code rates, converging E_b/N_0 values, throughput and the required number of iterations.....61
4.4 EXIT chart and area properties of various binary BTCs at their converging E_b/N_0 values 64
4.5 EXIT chart and area properties of various non-binary BTCs at their converging E_b/N_0 values for $p = 4$ bits.....70
5.1 I-TC and TC at different frame sizes with their converging E_b/N_0 values in AWGN.....80
5.2 I-TC at different code rates with their converging E_b/N_0 in AWGN.....82
5.3 Number of iterations required for convergence in an ITC and the TC.....83
5.4 I-TC at different code rates with their converging E_b/N_0 in the 3GPP multipath Vehicular A channel.....87
6.1 Number of operations required to achieve a low BER in the TPCs & the I-BTCs.104
6.2 Comparison between the required E_b/N_0 for TPC and I-BTC to achieve a low BER in an AWGN channel.....105
6.3 Distances to Capacity for TPCs and the I-BTCs.....107
6.4 Distance to capacity in bits per channel use for TPC and I-BTC.....116

List of Abbreviations

3.5G	3rd Generation cellular network
3GPP	3rd Generation Partnership Project
APP	*a posteriori*
AWGN	Additive White Gaussian Noise
B	Bandwidth
BCH	Bose-Chaudhuri-Hocquenghem
BCJR	Bahl, Cocke, Jelinek and Ravi
BER	bit error rate
BLER	block error rate
BPSK	Binary phase shift keying
BTC	Block Turbo Code
CCMC	Continuous-input Continuous-output Memoryless Channel
CDMA	Code Division Multiple Access
CTC	Convolutional Turbo Code
DCMC	Discrete-input Continuous-output Memoryless Channel
DMC	Discrete Memoryless Channel
E_b/N_0	signal energy to noise energy ratio per bit
EXIT	Extrinsic Information Transfer
FER	frame error rate
GPRS	General Radio Packet Services
GSM	Global System for Mobile Telecommunications
HSDPA	High Speed Downlink Packet Access
I-BTC	Irregular Block Turbo Code
IrCC	Irregular Convolutional Code
IrMapper	Irregular Mapper
I-TC	Irregular Turbo Code
IrURC	Irregular Unitary Rate Code
LAN	Local Area Network
LDPC	Low Density parity Check
LLR	Log likelihood ratio
Log-MAP	Logarithmic Maximum *a posteriori*
LTE	Long term Evolution
MAP	Maximum *a posteriori*
MATLAB	Matrix Laboratory
Max	maximum
MI	Mutual information
NSC	Non systematic convolutional code
PCC	Parallel Concatenated Code
PSK	Phase shift keying

QAM	Quadrature amplitude modulation
QPSK	Quadrature phase shift keying
RS	Reed Solomon
RSC	Recursive systematic convolutional code
S	Signal power
SCC	Serially Concatenated Code
SIHO	soft input-hard output
SISO	soft input- soft output
SMS	Short Message Services
SNR	signal to noise ratio
TC	Turbo Code
TPC	Turbo Product Code
TTI	Transmission Time Interval
UMTS	Universal Mobile Telecommunication Systems
VSAT	Very Small Aperture Terminal
W-CDMA	Wideband Code Division Multiple Access
WIMAX	Wireless Interoperability for Microwave Access

List of mathematical symbols

$\mathbf{A}$	(Bold capitals represents vectors)
a	fading amplitude
A_{in}	area underneath the EXIT function of an inner code
A_u	area underneath the EXIT function of an upper code
A_l	area underneath the EXIT function of a lower code
C	channel capacity
C_A	attainable capacity
C_{DCMC}	discrete continuous modulated channel capacity
d	data
E_b	transmitted energy per bit
E	expected value
f(x)	function of x
$H(X)$	entropy of X
$H(X/Y)$	conditional entropy of X given Y
I	in phase
I_a	input mutual information
I_e	output mutual information
$I(X, Y)$	mutual information between X and Y
L	capacity loss in bits per channel use
L_c	channel reliability
$L(u_k)$	log likelihood of u_k
$L(u_k \mid y_k)$	conditional log likelihood of u_k given y_k
N_0	noise spectral density
p	probability
Q	quadrature phase
q	number of bits in a symbol
R	code rate
S	throughput
$\sum_{i=1}^{n}$	summation from i to n
π	interleaving
π'	*de*interleaving
δ^2	noise variance
%Δ	percentage change
Φ	number of iterations
$\| \ \|^2$	squared norm
$\approx$	approximation

$\alpha_k(s_k)$	BCJR forward recursion from s'_{k-1} state to s_k state
$\beta_{k-1}(s'_{k-1})$	BCJR backward recursion from s_k state to s'_{k-1} state
$\gamma_k(s'_{k-1}, s_k)$	BCJR transition probability from s'_{k-1} state to s_k state

Contents

Chapter 1 Introduction 1

1.1 Channel coding and decoding complexities 1

1.2 Near capacity operation with low bit error rate 4

1.3 Objective of the Study and key novel contributions 5

1.4 Thesis layout 7

Chapter 2 Background studies 9

2.1 Introduction 9

2.2 Information Theory 10

2.3 Turbo Code (convolutional and Block Turbo Codes) 16

2.3.1 Convolutional Turbo Code 16

2.3.2 Block Turbo Code 18

2.4 Conclusion 20

Chapter 3 Literature review 21

3.1 Introduction 21

3.2 Channel coding 21

3.3 Regular coding 22

3.4 Irregular coding 25

3.5 Iterative decoding and EXIT charts 27

3.6 Conclusion 30

Chapter 4 BER performance of Turbo Code and Block Turbo Code (binary and non-binary).....31

4.1 Introduction.....31

4.2 Turbo Code in high speed downlink packet access.....32

4.2.1 Log-likehood equations.....35

4.2.2 Iterative decoding.....37

4.3 HSDPA system evaluation.....42

4.4 Bit error rate performance of Turbo Code in BPSK, QPSK, 16QAM and 64QAM modulation schemes.....43

4.4.1 Convolutional Turbo Code in 3GPP wireless multipath channel.....47

4.5 EXIT charts of Turbo Codes and their area properties.....49

4.5.1 Extrinsic transfer characteristics using iterative decoder for parallel concatenated scheme.....50

4.5.2 Extrinsic transfer characteristics using iterative decoder for serially concatenated scheme.....51

4.5.3 Parameters influencing the transfer characteristics of an EXIT chart...53

4.6 Block Turbo Code.....57

4.6.1 BER performance of Block Turbo Code 58

4.6.2 EXIT charts of the binary Block Turbo Code and its area properties...62

4.7 Reed Solomon Block Turbo Code.....65

4.7.1 Iterative decoding of Reed Solomon Block Turbo Code.....66

4.7.2 Bit error rate performance of Reed Solomon Block Turbo Code.....68

4.8 Conclusions.....71

Chapter 5 BER performance of the Irregular Turbo Code...73

5.1 Introduction.....73

5.2 Design and construction of an Irregular Turbo Code.....75

5.2.1 System parameters.....78

5.3 Bit error rate and throughput performance of Irregular Turbo Code in

AWGN.....78
5.4 Bit error rate and throughput performance of Irregular Turbo Code in a 3GPP multipath channel.....85
5.5 Conclusions.....87

Chapter 6 Near Capacity Irregular Block Turbo Code (I-BTC) with low complexity.....88

6.1 Introduction.....88
6.2 Design and construction of an I-BTC.....90
6.2.1 Bit degree combination.....100
6.2.2 The EXIT function of an I-BTC and its area property.....101
6.3 Bit error rate performance of I-BTC in BPSK, QPSK, 16QAM and 64QAM modulation schemes.....102
6.4 Key advantages of the I-BTC over the TPC and the potential applications of the I-BTC.....117
6.6 Conclusion.....117

Chapter 7 Conclusions and future work.....119

7.1 Introduction.....119
7.2 Summary of the thesis.....119
7.3 Suggestions For Future Research.....123
Appendix A.....124
Appendix B.....146
References.....147
Publications.....157

Chapter 1

Introduction

Contemporary research results in the field of communication systems have culminated in diverse and expanded possibilities for improving both the transmitter and the receiver section for two way communications. Recent rapidity in technological development and the current state of art in channel coding platforms, makes communications systems to be better positioned for greater improvement. Forward error correction i.e. channel coding has played a significant role in today's communication systems and of importance is the complexity it introduces to the system and hence the need to ameliorate this complexity.

1.1 Channel coding and decoding complexities

Channel coding theory and related information theory made impressive progress in the late 90's and early 2000's towards accomplishing Shannon's seminal work [1] published in 1948. One of Shannon's recommendations was to approach the signal to noise ratio capacity limit with an infinitesimal error probability, i.e. finding an error correcting code that performs as close as possible to channel capacity with an infinitesimal probability of error. In Shannon's paper [1] regarding the mathematical theory of communication, the author showed that communication over noisy channels imposed uncertainty on their received signals. He then showed that, information transmitted over a noisy channel at a rate (expressed in bits per second – bit/s) that does not exceed the channel's capacity,

could theoretically be reconstructed with an infinitesimally low probability of error. This motivated the employment of channel coding which introduced redundancy into the transmitted signal in a deterministically designed manner. This redundancy may be exploited in the receiver to correct any channel-induced errors within the original non-redundant information. This led to the introduction of a channel codec when transmitting messages over noisy channels. The message to be sent is encoded with a channel code before transmission on the channel. At the receiving end, the output from the channel is decoded back to a message, which theoretically, is the same as the original one. A fundamental property of Shannon's work [1] is his channel coding theorem, which states that the probability of error in a communication channel vanishes to zero (or to an infinitesimal value) as long as the information rate does not exceed the "capacity" of the channel. The term capacity in this study refers to the highest rate of transmission in bits per channel use at which information can be sent over a channel with arbitrarily low probability of error at the lowest possible signal to noise ratio. Extensive treatments of this concept can be found in many literatures [2], [13], [48]. Figure 1.1 gives a simple elucidation of Shannon's hypothesis of how information could be transmitted over a communication channel.

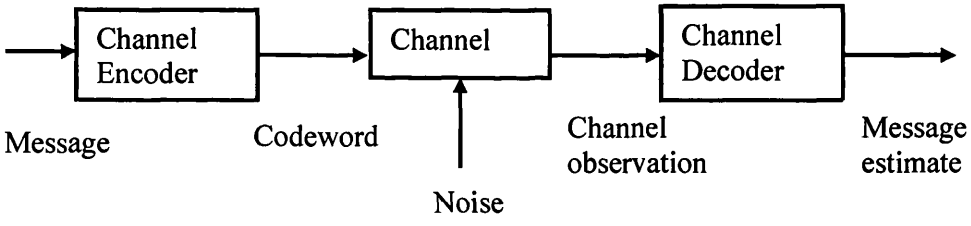


Figure 1.1 A communication model as proposed by Shannon for an unreliable channel

A key parameter in designing a channel codec is the complexity versus its performance. Over time, long codes have been known to perform better in terms of Shannon's limit compared to short codes, but the major challenge is that the former poses a very high decoding complexity. In an attempt to meet the problem of decoding complexity of long codes, Forney [3] in his thesis introduced the concatenated or modularized codes by

breaking the required computation of long codes into manageable segments and easy to debug routines. This resulted in many new channel codecs with an acceptable complexity in comparison with its error correcting power. This is similar to an earlier work by Elias [4] where powerful codes called product codes were constructed from simple linear block or modular codes. However, the decoding performance was poor at that time because of the use of hard-input–hard-output decoding.

Until recently, the most important decoding algorithm has been the Viterbi algorithm [5]. The major reason for its success is that it is optimal, in the sense that it minimizes the probability of decoding error for a given code. However, its major demerit is the computation complexity, which is very high for good codes. Also the number of operations in a Viterbi algorithm grows exponentially with the minimum distance of the code, as demonstrated by Muder [6].

The introduction of iterative decoding by Berrou et. al. [7] in their paper titled "Near Shannon limit error-correcting coding and decoding", gave rise to a new development platform in the world of channel coding. The new channel codes introduced by Berrou were called Turbo Codes. A key feature of Turbo Codes was that, up to date, they are the closest known channel codec for low rate codes to Shannon's limit, in addition to their characteristic of moderate complexity.

Recently, Pyndiah [8] introduced a novel efficient decoding algorithm called the "Pyndiah-Chase" decoding algorithm for product codes. It implements iterative decoding of product codes using a soft-input soft-output (SISO) decoder based on the Chase algorithm [9], followed by reliability calculations to obtain soft decisions from the hard output of the decoder. Thus, product codes were called Turbo Product Codes (TPCs) or Block Turbo Codes (BTC). The Block Turbo Code is the closest high rate channel codec to Shannon's limit.

Designing a high rate channel codec closer to the Shannon bound with a lower complexity compared to existing channel codecs in the previous paragraph is a challenge

to any modern day communication researcher. This was the major motivation for embarking on this research, with some significant results addressing the challenges stated above.

1.2 Near capacity operation with low bit error rate

Diverse channel coding schemes over the years have brought about negligible bit error rate (BER) at low transmit power in recent times in transmission links. The introduction of Hamming codes, Block codes, Convolution codes and Forney's "Concatenated codes" [3] which brought about new developments in channel coding in the late 1960's, as explained earlier, have significantly improved the BER performance in communication systems. In recent times, the most efficient coding schemes in terms of low BER performance versus E_b/N_0 ratio, i.e. the Low Density Parity Check Codes (LDPC) and the Turbo Codes where good BER performances have been recorded, still give room for improvement with regards to a nearer capacity operation and lower implementation complexity. With recent improvements in Turbo Codes (TC) and Block Turbo Codes (BTC), communication channels can now send signals at high bit rates giving rise to video transmission, teleconferencing and real-time video. The goal in this thesis is to design a channel codec operating near the Shannon bound at low bit error rate and most importantly with a lower decoding complexity.

Near-capacity operation is of particular interest in digital communications applications such as VSAT modems (data and voice), optical fiber communication systems and other microwave point to point links. This is because the channel's capacity reduces with the channel's signal energy to noise energy ratio (E_b/N_0), which depends on the transmit power, the distance to the remote receiver and the amount of noise encountered in the channel, Gaussian in our case, that is imposed upon the signal. Hence, digital communication schemes possess particular information transmission rates that are associated with a minimum threshold E_b/N_0 at which the channel's capacity drops below the particular information transmission rate to be maintained [14]. The capacity of a

continuous channel is mathematically and theoretically given as C_c, where $C_c = B \log (1 + S/N)$, with B as available bandwidth, S as signal power and N as average noise power over a given noisy channel. Similarly, for a discrete memoryless channel, the capacity of the channel is given as C_d, where $C_d = \text{Max } I(X, Y)$, that is, the maximum mutual information in any single use of the channel, measured in bits per channel use or bits per transmission [13]. Of importance to this project is the channel capacity for a discrete memoryless channel C_d. Maunder showed in his work [14] that near capacity operation could be achieved in an iterative decoding scheme simply by ensuring that the tunnel gap between the EXIT (extrinsic information transfer) functions of each decoder of an iterative decoding scheme should be as small as possible at the same time not making any intersection before the point (1,1) in the EXIT chart is reached. The point (1,1) is fully explained in Chapter 4 when explaining the EXIT chart itself, as well as its uses for iterative decoding. The point (1,1) also indicates that the iterative decoding scheme will operate at a low bit error rate or a negligible low probability of error. Maunder in [14] also showed that the tunnel gap between the EXIT functions is synonymous to capacity discrepancy as fully explained in Chapter 4 as well. Therefore, ensuring a very narrow tunnel gap between the two EXIT functions during the designing of an iterative decoding scheme is very important for a near capacity operation.

1.3 Objectives of the study and key novel contributions

The main objectives of this study are to design a new channel codec for future communication systems with the following properties:

1. Closer to its channel capacity;
2. Low implementation complexity;
3. Possesses a reducible BER;
4. Possess a mathematically traceable design;

The key significant contributions are as follows;

- A new Irregular Block Turbo Code (I-BTC) with a new encoding and decoding technique was developed. Previous authors in recent years have developed an irregular LDPC and an Irregular Turbo Code.

The I-BTC which is an iteratively decoded channel codec is encoded horizontally without vertical encoding (or vice versa), with the decoding requiring only one soft-input soft-output (SISO) decoder. The SISO decoding in the I-BTC was performed using Log-MAP decoding.

The I-BTC is a high rate channel codec with high flexibility in terms of its design and construction. A significant feature of the novel I-BTC is its low complexity with a significant reduction of up to 46% in the number of operations required for an I-BTC in some cases to achieve a negligible BER in comparison to an equivalent existing BTC.

A coding gain of 1.28 dB was achieved in the I-BTC over its corresponding BTC for 64QAM modulation in an AWGN channel. This is the largest coding gain ever recorded in an irregular code over its corresponding regular code in a Gaussian channel. Previous versions of irregular codes were able to achieve a coding gain less than 0.8dB as seen in the work of MacKay et. Al [39], and Boutros et. al [40].

These features together with the record coding gain in our view makes the I-BTC an ideal replacement as a channel codec for high speed communications on Gaussian noise channels.

- Irregular Turbo Codes (I-TCs) were designed in higher order modulation schemes with a small frame size unlike in previous I-TC versions by MacKay et.al. and Boutros et. Al., where large frame size of 2^{17} were used to achieve a low BER.

The I-TCs designed uses a single encoder and a single decoder as previously designed by other authors, but with a significant reduction in the frame size and the number of iterations required to achieve a negligible BER. This significantly reduces the complexity in designing an I-TC in comparison to existing I-TCs.

The I-TC is capable of achieving a coding gain of 1.3 dB and 1.4 dB in a Gaussian and a 3GPP, multipath channel, respectively, for a higher order modulation scheme (16QAM).

- This thesis records the largest coding gain achieved between an irregular code and a regular code in a Gaussian noisy channel, thereby resulting in a large power savings during system operation.

- The application of EXIT charts to the design of an I-BTC.

The area underneath the EXIT function of an I-BTC was shown to be equal to the channel capacity of the system. This was then used to determine the capacity loss in an I-BTC in bits per channel use.

- The application of EXIT chart in calculating systems capacity

The EXIT chart has been used to calculate the channel capacity of the Turbo Code and the Block Turbo Code, consequently determining the distance to capacity in bits per channel use.

1.4 Thesis layout

The thesis has been divided into seven Chapters. Chapter 1 contains the introduction which delves into pre-background discussion of coding issues and problems. It also highlights some key concepts in the field of channel coding. Chapter 2 gives a discourse

of the background studies on information theory including the classical Turbo Code and the Block Turbo Code theories as well as their applications relevant to this thesis. Chapter 3 contains the literature review of developments regarding channel coding technologies relevant to this thesis. Chapter 4 discusses the bit error rate (BER) performance of a Turbo Code in the high speed downlink packet access (HSDPA) framework at different code rates. The BER performance of various Block Turbo Code sizes for the binary and non-binary cases, using various modulation schemes from BPSK to 64QAM is also presented with the aim of determining their distance to capacity. In addition, the theory of EXIT chart technique and its area properties are examined, where EXIT charts of Turbo Codes and Block Turbo Codes are plotted at various code rates using different modulation schemes. The area property of the EXIT chart is then used in calculating the capacity of the communication channel for the Turbo Code and the Block Turbo Code. In Chapter 5, the BER performance of an Irregular Turbo Code is evaluated for various modulation schemes in AWGN and 3GPP multipath channels. Also in Chapter 5, the throughput analysis of the Irregular Turbo Code is evaluated in comparison with the regular Turbo Code. The novel Irregular Block Turbo Code is discussed in Chapter 6 with all its features and characteristics while Chapter 7 gives a summary of the thesis, and the key conclusions. Future work with regards to this thesis is also discussed in Chapter 7.

An overview of information theory, channel capacity and iterative decoding relevant to the thesis is examined in the next Chapter.

Chapter 2

Background studies

2.1 Introduction

In this Chapter, background studies relevant to this thesis on information theory as related to Shannon's original work, entropy, mutual information and channel capacity is reviewed. Of major significance are the classical Turbo Code and the Block Turbo Code. The encoding structure of these two channel codecs is examined as well. It is necessary to define what performance means for a particular channel codec because, performance may mean different things from different views. In this thesis, "performance" measurements centre mainly on the bit error rate (BER) and system throughput derived from the system's frame error rate (FER) or Block error rate (BLER) as the case may be, while the system complexity and latency tend to be secondary performance concerns. Complexity and latency are then examined as separate channel codec features along with performance. The BER performance in this thesis is approached in two ways. Firstly, by comparing the relative energy efficiency of particular codes using BER versus E_b/N_0 curves. However, this performance metric does not take into account bandwidth efficiencies at specific code rates but only makes known the energy related performance together with the achieved coding gain or loss as the case may be. This basically shows how the transmit power is used with no suggestion on bandwidth usage. Secondly, the BER performance is evaluated by distance to capacity in bits per channel use versus the

average bit energy to noise energy ratio (E_b/N_0). In other words, how close to channel capacity i.e. Shannon's bound, does a given channel codec operate at a very low BER. Unlike the first BER performance approach, channel capacity considers bandwidth efficiency as well as power efficiency [15].

Feasibility of implementation is another vital aspect of a channel codec because it can affect the cost of the equipment and the power requirement of these devices. Complexity could be hard to quantify since various implementations of the same solution may differ in various parameters such as power requirement, cost, etc [15]. In this thesis, complexity is assessed on the number of mathematical operations required in the decoding of a particular codec.

Latency is usually quantified by interleaver depth (i.e. block/frame size) and the number of iterations in an iterative decoder. In this thesis, latency is quantified by the block/frame size together with the number of iterations used to converge to a low BER for a particular channel codec. Chapters, 4, 5 and 6 gives in details how these performance metrics have been used in this thesis.

2.2 Information theory

Reliable data transmission over noisy communication channels requires the use of forward error correcting codes known as channel coding. Originally and theoretically introduced by Shannon, information theory is a fundamental theory in channel coding and unequivocally the most original and important scientific basis of communications. It is also important in theory and in practice because transmitting with a very low probability of error or no error is highly desirable.

Shannon's channel coding theorem states that any communication channel is completely characterized by a single number C, the channel capacity, in the manner that R random

binary digits (or bits) per second can be sent reliably through the channel only if $R < C$, but cannot be so sent reliably if $R > C$ [16]. The impact of the channel coding theorem was so important that Massey [16] commented, "the fact that the rate R of a transmission system need not be reduced to achieve increased reliability must have been as incredible to the communications engineers of 1948 as had been the heliocentric theory to the astronomers of 1543 as both results clashed with all previous experience and theory relating to them". This is more so since bit error rate is inversely proportional to signal to noise ratio and everyone before 1948, knew that to reduce errors, one must increase the signal to noise ratio. However, Shannon demonstrated that what counted was not the signal to noise ratio (as long as $R < C$), but how the information was coded [16].

In [17], Pierce gave one of the first simple mathematical insights into the channel coding theorem by Shannon. For a bandwidth B, the channel capacity C is given by (2.1)

$$C = B\log_2\left(1+\frac{P}{N}\right) \quad bis/s \tag{2.1}$$

where P is the average power of a signal and N is the average noise power, the noise being additive and Gaussian. Before going further on what channel capacity is in digital systems, it is worth examining entropy and mutual information. When an operation is independent of the origin of time, the average quantity of information produced by a source and carried by a message that is transmitted is called its entropy. In information theory, entropy or information entropy is a measure of the uncertainty associated with a random variable. It quantifies the information contained in a message, usually in bits or bits per symbol. It is the minimum message length necessary to communicate information in a given time epoch. Assuming a random variable X with outcomes x_1, x_2, x_3 and x_4 having probabilities $p_1= 1$, $p_2= 0$, $p_3=0$ and $p_4=0$, it is easy to know what the outcome would be in any selection process about the random variable X, as such there is no information from that activity. But suppose their probabilities are $p_1=0.25$, $p_2=0.25$, $p_3=0.25$ and $p_4=0.25$ then there is no idea what the outcome of a selection process about X would be. In other words, there is some measure of information to be gained at every selection process. Similarly, a fair coin when tossed could either be a head or a tail with

complete uncertainty of the outcome. However, if the coin is not fair, then the uncertainty is lowered (with lower information) and thus the entropy is lower. Entropy in information theory then is simply a measure of the average information content per source symbol.

For a discrete memoryless source, the entropy $H(x)$ of such a source is bounded as follows:

1. $0 \leq H(x) \leq \log_2 K$

where K is the radix (number of symbols) of the alphabet of the source.

2. $H(x) = 0$, if and only if; the probability $x_k = 1$ for some K, and the remaining probabilities in the set are all zero. This is the lower bound and corresponds to no uncertainty.

3. $H(x) = \log_2 K$, if and only if $x_k = \frac{1}{K}$ for all K (i.e. all the symbols in the alphabet are equiprobable). This is the upper bound and corresponds to maximum uncertainty.

The entropy of a discrete random variable X, denoted $H(X)$, that can take on possible values $\{x_1, x_2, x_3, \ldots\ldots x_{n-1}, x_n\}$ is given by [13];

$$H(X) = E[-\log_2[p(X)]] = E[I(X)] = -\sum_{i=1}^{n} p(x_i).\log_2[p(x_i)] \quad bits \quad (2.2)$$

Where $I(X)$ is the information content or self-information of X, which in itself is a random variable and $p(x_i) = p(X = x_i)$ is the probability mass function of X. $H(X)$, the entropy of X, is simply the uncertainty of X. It also represents the uncertainty there is about X, prior to its measurement. Equivalently, it is the amount of information gained when X is measured as the uncertainty equals information.

Now suppose that $Y = f(x)$ for some probabilistic function $f(x)$. Let Y be the output of a noisy channel that has X as the input. Let A_y denote the set of possible outcomes from a transmission set A_x. $H(X/Y)$, is the uncertainty remaining about X when Y is measured.

$$H(X/Y) = \mathrm{E}[-\log_2 p_{X/Y}(X/Y)] = -\sum_{x\in A_x} p_{x/y}(x/y)\log_2 p_{x/y}(x/y)\ \ bits \quad (2.3)$$

Then the average uncertainty in X, averaged over the outcome Y is called the conditional entropy; H(X/Y), and by definition computed as

$$H(X/Y) \square \sum_{y\in A_y} H(X/Y)p_Y(Y) = -\sum_{y\in A_y}\sum_{x\in A_x} p_{x/y}(x/y)p(y).\log_2 p_{x/y}(x/y)$$

$$= -\sum_{y\in A_y}\sum_{x\in A_x} P_{x,y}(x,y).\log_2 P_{x/y}(x/y) \quad bits \qquad (2.4)$$

Since $p_{x/y}(x/y)p_y(y) = p_{x/y}(x,y)$ from Bayes' rule.

These definitions then lead us to the mutual information between 2 random variables. The Mutual information between 2 random (events) variables X and Y, is the information content provided by the occurrence of say event Y=y_i about the event $X = x_i$, defined by

$$I(X,Y) = H(X) - H(X/Y) = H(Y) - H(Y/X) \qquad (2.5)$$

also given by (2.6).

$$I(x_i,y_i) = \sum p(x_i)\log_2\frac{1}{P(x_i)} - \sum\sum p(x_i,y_i)p(y_i)\log_2\frac{1}{p(x_i,y_i)} \qquad (2.6)$$

Since $p(x) = \sum p(x/y)p(y)$ then,

$$I(x_i,y_i) = \sum\sum p(x_i/y_i)p(y_i)\log_2\frac{1}{P(x_i)} - \sum\sum p(x_i/y_i)p(y_i)\log_2\frac{1}{p(x_i/y_i)}$$

$$= \sum\sum p(x_i/y_i)p(y_i)\left\{\log_2\frac{1}{p(x_i)} - \log_2\frac{1}{p(x_i/y_i)}\right\}$$

$$= \sum\sum p(x_i/y_i)p(y_i)\log_2\left\{\frac{p(x_i/y_i)}{p(y_i)}\right\}$$

$$= \sum\sum p(x_i, y_i) \log_2 \left\{ \frac{p(x_i / y_i)}{p(y_i)} \right\} \qquad (2.7)$$

Properties of mutual information

- Mutual information (MI) of a channel is symmetric; that is $I(X,Y) = I(Y,X)$

- Mutual information is always non-negative; that is $I(X, Y) \geq 0$; that is on the average, information cannot be lost by observing the output of a channel. Moreover, mutual information is zero if and only if the input and output symbols of the channel are statistically independent.

- Mutual information is related to the joint entropy of the channel input and channel output by ;

  $I(X,Y) = H(X) + H(Y) - H(X, Y)$ (2.8)

  Where $H(X, Y)$ is the joint entropy of the channel input and output.

Channel capacity

The mutual information of a channel does not depend only on the channel characteristics, but also on the way in which the channel is used. The input probability distribution $P(x_i)$ is independent of the channel. The mutual information $I(X,Y)$ of a channel can then be maximized with respect to $P(x_i)$. Hence the channel capacity C of a discrete memoryless channel (DMC) with input X and an output Y can be defined as the

maximum mutual information in any single use of the channel between X and Y taken over all possible input distributions measured in bits per channel use or bits per transmission [13]. The channel capacity referred to here is the discrete memoryless channel (DMC).

$$C = \max_{P_x(x)} I(X,Y) \qquad \text{bits / channel use} \qquad (2.9a)$$

Bits per channel use is the average number of correctly received bits at the receiver section of a two way communication system for a single use of the channel.

Channel characteristics determine the transition probability $p(y_i / x_i)$, but the probabilities of the input symbols are under the control of the discrete channel encoder. The value of $I(X,Y)$ maximized over the set of input symbol probabilities $p(x_i)$, is a quantity that depends only on the characteristics of the Discrete-input Continuous-output Memoryless Channel (DCMC) through the conditional probabilities $p(y_i / x_i)$. Maximizing over a set of input symbol probabilities $p(x_i)$ means, the probabilities of every element x_i in the discrete random variable X should be the same so as to produce maximum entropy.

Shannon's capacity in this thesis is measured first in dB, which is the minimum E_b/N_0 required to achieve a low BER for a Discrete-input Continuous-output Memoryless Channel (DCMC), and is a function of the modulation scheme and code rate [1]. Secondly, capacity is measured by the maximum possible throughput in bits per channel use at a particular E_b/N_0 value at which information can be sent with arbitrarily low probability of error [13], i.e. the maximum mutual information per single channel use, and is also a function of the modulation scheme and code rate, denoted as C_{DCMC} in this thesis. The C_{DCMC} is obtained by simulating the maximum mutual information depending on the modulation scheme and channel used. C_{DCMC} as a measure of distance to capacity, takes into account the system's spectral efficiency as well as the energy

efficiency unlike the E_b/N_0 in dB which only takes into account the energy efficiency, making C_{DCMC} a better option.

In Chapters 4, 5 and 6, the concept of attainable capacity denoted as C_A measured in bits per channel use is also used in determining the distance to capacity. Attainable capacity is the channel capacity derived from the EXIT chart of a channel codec. The DCMC capacity of a channel equals the attainable capacity of the same channel if the code rate of the channel codec equals one. For code rates less than one, the attainable capacity is slightly lower than the DCMC capacity, i.e. $C_A \leq C_{DCMC}$ depending on the code rate.

2.3 Turbo Code (Convolution and Block)

In this section, a brief explanation of the encoding structure of the Convolutional Turbo Code (CTC) and the Block Turbo Code (BTC) or Turbo Product Code (TPC) is given.

2.3.1 Convolutional Turbo Code

Iteratively decoded codes also known as Turbo Codes have the most powerful error correcting capability in today's communication systems ([7], [10]). In general, Turbo Codes can be grouped into two classes according to their constituent encoders, such as the CTC and the BTC. Due to their near Shannon capacity performance, Turbo Codes have received a lot of attention since their introduction in 1993 [7]. The CTC is particularly attractive for wireless communication systems and has been in the specification for both WCDMA and CDMA 2000 third generation cellular standards. HSDPA, the 3.5 G system, also adopted the CTC as it's channel codec. The recently developed Long Term Evolution (LTE) standard has also adopted the CTC as it channel codec due to its BER performance and closeness to capacity. Following the introduction

of the CTC, came the BTC, specially designed for high rate codes. An interesting aspect of Turbo Codes is that it's not just a single code, but rather a combination of two codes that work together to achieve a synergy that would not have been possible with the use of one code alone.

A Convolutional Turbo Code consists of a parallel concatenation of recursive systematic Convolutional (RSC) encoders separated by a random interleaver (other interleavers could also be used such as a block interleaver) [7]. In the CTC, the information bit is fed directly into the first or upper RSC without any modification. The second RSC or lower RSC is then fed with an interleaved version of the original information bit as can be seen in figure 2.1.

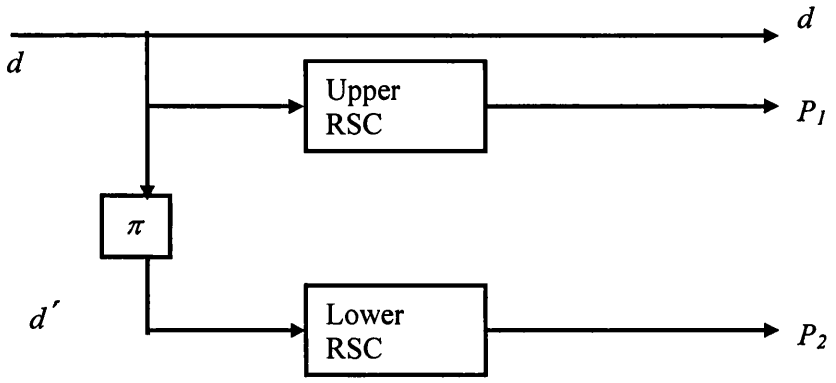


Figure 2.1 A CTC encoder structure

In figure 2.1, an RSC of rate 1/2 is assumed. This gives an overall rate of 1/3 for the Turbo Code which is the natural rate for a Turbo Code, as each RSC outputs a parity bit (P_1 and P_2) for each input information bit d. The rate can be relatively increased by puncturing the parity bits. The modified Bahl, Cocke, Jelinek and Raviv (BCJR) algorithm is used to provide a soft-input soft-output (SISO) mapping or decision between two decoders (one for each RSC) at the receiver section. Given the received symbol sequence y, for each decoded bit U_k, the *a posteriori* log likelihood ratio (*LLR*) over a Gaussian channel is given by (2.10).

$$L(U_k \mid y) = \log_2 \left[\frac{P(U_k = +1 \mid y)}{P(U_k = -1 \mid y)} \right]$$

$$= L(U_k) + L_{cy} + L_e(U_k) \qquad (2.10)$$

Where $L(U_k)$ is the *a priori LLR*, provided by a component decoder, L_{cy} is the channel observation and $L_e(U_k)$ is the extrinsic *LLR* for the bit U_k. $L_e(U_k)$, an important aspect in turbo decoding is fully explained in Chapters 4 and 6, where its mathematical derivation is explained during the decoding of the CTC, BTC and the irregular BTC. The optimal trellis based iterative MAP decoding or Log-MAP decoding method presented in [18] and [19] is applied in the decoding. During the iterative decoding, extrinsic information gleaned from each component decoder is exchanged.

2.3.2 Block Turbo Code

TPCs are built from the serial concatenation of linear block codes, such as the Bose Chaudhuri Hocquenghem (BCH) codes, separated by a block interleaver [8], and then decoded iteratively by its component decoders. Turbo Product Codes can be multi-dimensional, ranging from two dimensions to three, four etc. In a two-dimensional TPC (which happens to be the most popular as found in the literature), the information bits or symbols are first encoded horizontally or row-wise and then encoded vertically or column-wise. The vertical encoding can be thought of as an interleaved version of the horizontally encoded bits or symbols. In the literatures, the word interleaving does not appear in a TPC simply because encoding is done horizontally and then vertically. But in the real sense, the vertical encoding of the information bits or symbols proceeding the horizontal encoding is the same as a block interleaved version of the horizontal bits or symbols. Figure 2.2 shows a block representation of a TPC.

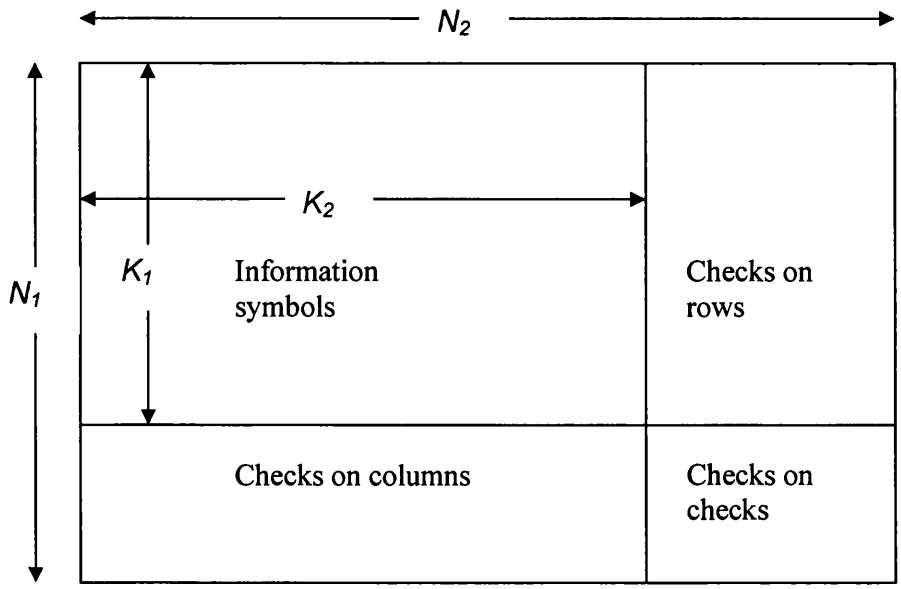


Figure 2.2 Construction of a two dimensional Turbo Product Code

Consider two systematic linear block codes C^1 and C^2 with parameters (n_1, k_1, ∂_1) and (n_2, k_2, ∂_2) respectively, where n is the code length, k, the number of information symbols and ∂ the minimum hamming distance. The product code $P = C^1 \times C^2$ is obtained by first placing information symbols in a $k_1 \times k_2$ matrix as seen in figure 2.2. Then the k_1 rows are encoded by C^1 followed by encoding the n_2 columns using C^2. The parameters of the resulting product code is given by $n = n_1 n_2$; $k = k_1 k_2$; $\partial = \partial_1 \partial_2$, with the code rate $R = R_1 R_2$ [8]. The decoding of a Turbo Product Code can be done in two ways. Firstly, is the "Pyndiah-Chase" algorithm presented in [8]. Secondly, the same trellis-based iterative BCJR decoding method could be used as implemented in [18]. The two decoding algorithms have been used in this thesis although under different conditions. The trellis-based iterative BCJR decoding method was used for the binary Turbo Product Code while the "Pyndiah-Chase" algorithm was used in the non-binary case e.g. Reed Solomon.

2.4 Conclusion

In this Chapter, an overview of the concepts of channel capacity, entropy and mutual information have been examined. These important elements have been instrumental in the design of channel codecs in time past and are currently in use in the design of modern channel codecs such as the Turbo Code.

Mutual information is a key concept in designing channel codecs operating near Shannon bound. This introduces the concept of EXIT chart techniques which are fully discussed in the next Chapter along with the current state of the art in channel coding.

Also in this Chapter, the performance measurements used for the various channel codecs were stated which centers mainly on the bit error rate (BER), system throughput, system complexity and latency.

Chapter 3

Literature Review

3.1 Introduction

This Chapter is an exploration and survey of important theories and developments in the field of information theory pertaining to channel coding as well as recent techniques used in enhancing existing channel codecs.

3.2 Channel coding

The speed or data rate in digital communication systems partly depends on the error correcting power of the channel codec used. This is so because, the reliability of transmission systems depend on the number of correctly received information bits at the receiver, which is a function of the channel codec. In the 1970's to late 1980's, the speed used in communication systems was limited resulting in a fixed amount of data transfer during transmission. This was because of the current state of the art in channel coding that was used then. Improvements were made in channel coding technologies that brought about short message services (SMS) and then the General Radio Packet Services (GPRS). At this point Convolutional coding was used as the channel codec in standard networks such as the Global System for Mobile telecommunications (GSM). This was made possible due to the high error correcting power of the Convolutional code, thereby

increasing the capacity and reliability of communication channels which amounts to an increase in communication speed.

The strength of a channel codec is a function of two key elements. First, is its bit error rate performance, i.e. how many errors detected in a transmission system can it correct. Secondly, is the amount of signal power required by the channel codec to ensure that the information sent over a noisy channel is correctly received, otherwise known as the signal to noise ratio. A combination of these two elements shows how close a channel codec is to the Shannon bound. However, the feasibility of implementing the channel codec, i.e. complexity, as well as the amount of delay, i.e. latency it introduces to the system, are also important in determining the strength of channel codecs. Over time, long codes have been known to perform better in terms of Shannon's limit compared to short codes, but the major challenge is that the former poses a very high decoding complexity. In an attempt to meet the problem of decoding complexity of long codes, Forney [3] introduced the concatenated codes by breaking the required computation of long codes into manageable segments. This resulted into many new channel codecs with an acceptable complexity in comparison with its error correcting power. However, the decoding performance was poor at that time because of the use of hard-input–hard-output decoding, leading to the neglect of concatenated codes for some years. The introduction of the Viterbi algorithm in [5], which involves the use of a trellis in the channel codec, helped introduce the concept of soft decision in channel codec decoding as would be examined in the next section together with the evolution of other strong channel coding concepts.

3.3 Regular Coding

The introduction of Hamming codes, Block codes and Convolution codes significantly improved the BER performance in communication systems from the 1960's to early 1990's. This can be explained by their good error correcting performance as well as their

moderate decoding complexity. The performance of Convolutional coding in particular was enhanced by the concept of soft decision as introduced by Viterbi. The Viterbi algorithm was developed by Andrew Viterbi in 1967 [5] for the decoding of Convolutional codes. It is a dynamic algorithm used to discover the most likely sequence called the "Viterbi path" from a received corrupted sequence, with respect to the originally transmitted sequence. The algorithm assumes that the received sequence corresponds to time, i.e. the Viterbi algorithm operates like a state machine. At any given time (t), the system is in a particular state which is finite in number. The larger the number of states, the more complex the system becomes. In the Viterbi algorithm, a received sequence could lead to any state, with at least one of them leading to the most likely path, called the "survivor path". This is a major assumption of the algorithm because the algorithm examines all possible paths leading to a state and keeps the most likely path. This way the algorithm does not have to keep track of all possible paths, rather it keeps only one path at any given time (t), throwing away all other paths. The algorithm has found universal application in decoding the Convolutional codes used in IS95 (i.e. Code division multiple access standard), GSM, dial-up modems, deep-space communications, and 802.11 wireless LANs. However, the complexity of a Viterbi decoder grows exponentially as the number of states increases.

As demand for higher data throughput and speed increased by the need for real time data services such as video transmission, teleconferencing and other real-time video applications etc, the need for stronger and more powerful channel codecs emerged to meet these new challenges. This led to the discovery of new and more efficient coding schemes in terms of low BER performance versus signal to noise ratio.

The introduction of iterative decoding by Berrou et. al. in 1993 brought about a new class of codes called Turbo Code. These codes are called Turbo Codes, due to their inherent iterative strength in error correction. The Convolutional Turbo Code (CTC) developed by Berrou, places two Convolutional codes in parallel with the upper code receiving the information bits, while the lower code receives an interleaved version of the information bits. The basic concept in interleaving the information bits prior to being sent into the lower decoder was to create diversity. The strength of a Turbo Code as stated earlier, is its turbo property which is derived from a new concept introduced by

Berrou called the extrinsic information. This extrinsic information is passed from one decoder to the other until a stop criterion is reached. This process is called "turbo decoding". The decoder receiving this extrinsic information as *a priori* processes them together with the received observation on a probabilistic level. Berrou later gave credit to the previous work of "G. Battail, J. Hagenauer and P. Hoeher, who, in the late 1980's, highlighted the interest of "probabilistic processing". The CTC is today used as the standard in many telecommunication systems such as the 3G network, High Speed Packet Access (HSPA) network and the LTE. Also new NASA missions such as Mars Reconnaissance Orbiter make use of the CTC, as an alternative to concatenated Reed Solomon-Viterbi codes. CTC is currently in use as in the IEEE 802.16 (WiMAX), a wireless metropolitan network standard.

The introduction of the CTC led to the development of the Block Turbo Code (BTC) by Pyndiah in 1994. The concept of the BTC also known as the Turbo Product Code (TPC) is the same as the CTC with the exception that Block codes have been serially concatenated in the case of the BTC while Convolutional codes are concatenated in parallel for the CTC. Before the development of the BTC, the decoding algorithms for Block codes were hard decisions, thereby limiting the performance of Block codes. The Block Turbo Code also achieved a very low BER at low signal to noise ratio close to Shannon's bound. Pyndiah also introduced a new decoding scheme, "the Chase-Pyndiah" soft decoder, where a selected list of codewords are used to produce soft-outputs from the Chase decoder as Chase is a hard decision decoder. Of importance in this channel codec is its high code rate, making it very useful for high speed communication systems. Today, the BTC is popularly used in satellite communications as well as in optical communication systems.

The LDPC code, originally developed by Gallager in 1963, was neglected, but rediscovered in 1996. An LDPC code is a linear error correcting code constructed using a sparse bipartite graph. They are capacity-approaching codes i.e. very close to the Shannon bound in terms of BER performance and make use of iterative belief propagation techniques. An LDPC code is currently used as the channel codec for digital transmission in the 10GBase-T Ethernet standard, which sends data at 10 gigabits per

second over twisted-pair cables. In 2003, an LDPC code was adopted as the channel codec in the DVB-S2 standard for the satellite transmission of digital television. Also, in 2008, LDPC was adopted as the channel codec for the ITU-T home network standard. Other applications where the LDPC is currently being used are the Digital Terrestrial Multimedia Broadcast (DMB-T/H) and the Worldwide Interoperability for Microwave Access (WiMAX i.e. IEEE 802.16e standard for microwave communications).

The complexity in decoding the CTC and the BTC has been shown to be moderate. The CTC uses a modified version of the BCJR algorithm [7], a trellis based algorithm originally developed in [19]. The BTC on the other hand uses either the "Chase-Pyndiah" algorithm or the BCJR algorithm as implemented in [18].
The channel codecs mentioned in this section could be classified into a class of codes called regular codes. This is so because, each information bit in a sequence, block or frame is equally protected from corruption by noisy channels, pointing us to the unequal protection of information bits termed irregular coding as an alternative approach.

3.4 Irregular Coding

The unequal protection of information bits has recently been of interest, from the design of an Irregular LDPC code to the design of an Irregular Turbo Code. Performance benefits identified so far from the use of irregular coding stem from the extra protection on some of the transmitted bits leading to an improved channel codec in terms of BER performance as well as their closeness to the Shannon bound. An Irregular LDPC code was designed and constructed by Richardson et. al. in their paper titled "Design of Capacity Approaching Irregular Low Density Parity Check Codes" in 2001. The best irregular LDPC code exhibited a performance very close to Shannon bound in a noisy Gaussian channel, with simulation results in a BPSK system showing a distance of 0.13 dB from Shannon's capacity at a BER of 10^{-6}, with a code length of one million bits.

This is about 0.82 dB improvement in the performance of the LDPC codes, however, this irregular LDPC code led to an increase in decoding complexity. In 1999, Brendan Frey and D. MacKay presented the first ever Irregular Turbo Code at the proceedings of the 37th Allerton Conference on Communication, Control and Computing, in their paper titled "Irregular Turbo Codes" [38] where they showed that by making the original rate 1/2 Turbo Code of Berrou et al slightly irregular, a coding gain of 0.15 dB was obtained in an AWGN channel using BPSK modulation, and bringing the Irregular Turbo Code within 0.3 dB of capacity at a bit error rate of 10^{-4}. The then Irregular Turbo Code performed in the same regime as the best known irregular Gallager code [38] in terms of bit error rate performance. This was also demonstrated in their paper titled "Irregular Turbo-like Codes" [39]. Their code can be decoded by the iterative application of the sum-product algorithm (a low-complex, more general form of the turbo coding algorithm). Making their code irregular required an increase in the rates of the constituent codes, leading to an increase in the number of low-weight codewords [39]. Just like regular Turbo Codes, Irregular Turbo Codes are also linear-time encodable. Also in [40], an Irregular Turbo Code achieved a coding gain of 0.24 dB at 10^{-6} BER in comparison with the regular Turbo Code in AWGN with a BPSK modulation scheme. An important concern in the field of channel coding is basically the problem of decoding complexities. The Irregular Turbo Code described in [40] utilized a single recursive systematic Convolutional (RSC) encoder and a single SISO decoder. This reduces the complexities in the decoding compared to the regular Turbo Code except for the very large frame size and the high number of iterations required (about 100) to achieve a very low bit error rate as reported in [40]. The problem of large frame size was also encountered in [38] and [39], with no information on the number of iterations required to converge to a low BER.

Irregular coding has been shown to improve the BER performance of channel codecs and are also closer to Shannon bound but at the cost of complexity. This was the motivation for this study, i.e. to develop a new high speed irregular coding scheme, the Irregular Block Turbo Code (I-BTC). The key challenges in developing this new channel codec were for the codec to produce a better BER performance, closer to Shannon bound

and at the same time possessing a lower decoding complexity in comparison to the existing BTC. To do this, certain state of the art technologies used in channel coding were examined to understand the behaviour of iterative decoding.

3.5 Iterative decoding and EXIT charts

In recent times, the convergence behaviour of iterative decoding was investigated after the discovery of Turbo Codes by Berrou et. al. in [7]. Richardson et. al. in [20] proposed a density evolution algorithm to calculate convergence thresholds for low-density parity check codes (LDPC) in an AWGN channel. Density evolution is a message passing algorithm used in predicting iterative decoding such as the LDPC code by analyzing the distribution of message exchange during decoding. The prediction performs well in very long codes. This is so because, as the codeword length tends to infinity, the codebook will be more and more likely to be cycle free, thereby ensuring that the input messages of each node during decoding are independent and could be calculated iteratively.

Another group of authors in [21] and [22] studied the convergence of iterative decoders based on signal to noise ratio (SNR) measures, while a third group, Peleg et. al. [23], examined a combination of SNR measures and mutual information for the description of inner rate one codes. A novel method for visualizing the convergence behaviour of iterative decoding schemes was then proposed by Stephan [24] in his electronics letter titled "Convergence of iterative decoding", where each constituent decoder was represented by a mutual information transfer characteristic, which described the flow of extrinsic information through the soft in/soft out (SISO) decoder. The exchange of extrinsic information between constituent decoders is plotted in an extrinsic information transfer chart (EXIT Chart). The concept, which was new, was then illustrated for an iterative demapping and decoding scheme [24]. Stephan did not claim to present a rigorous proof of stability and convergence of iterative decoding, however, his simulation results suggested that the EXIT chart accurately predicted the convergence

behaviour of the iterative decoding for large interleaver depths. There are several reasons for choosing EXIT charts as found in [26]:

- Mutual information seems to be the most accurate statistics in predicting the likelihood of decoders in correcting noise corrupted bits ([27, sec. IV], [28]), depending on the SNR value used.

- Mutual information is the most robust statistic, i.e., they accurately predict decoding performances in any channel condition and modulation scheme used. For instance, EXIT functions apply to erasure channels without change. They further apply to symbol-based decoders [29] and to suboptimal decoders such as hard-decision decoders.

In his work, Stephan went on to describe and predict the iterative convergence behaviour for both parallel [30] and serially concatenated codes [31] with the use of an EXIT chart. EXIT chart has since then been widely used for the prediction and design of iterative decoding schemes by various authors as seen in the work of Michael Tuchler " Design of serially concatenated systems for long or short block length" [32], where the EXIT chart was used to construct simple irregular codes, which can significantly improve the convergence behaviour of iterative decoding. To predict if an iterative decoding scheme would converge (i.e. give negligible BER) at a particular signal to noise ratio, an EXIT chart is plotted at that signal to noise ratio value. An open tunnel gap in the EXIT chart suggests convergence, while a closed or crossed over tunnel gap suggest no convergence. In an EXIT chart, the narrower the tunnel gap, the closer the iterative decoding scheme is to capacity. This is the key parameter used in designing and improving iterative decoding schemes. In a thought-provoking paper, Eiko Seidel et. al. [33], studied 2 different parallel concatenated coding schemes by way of EXIT chart prediction. One of the schemes was the classical Turbo Code (TC) [7], while the other was the concatenation of regular LDPC codes. A simplified simulation was set up to obtain the EXIT characteristics for the constituent encoders and applied to parallel concatenated codes for both LDPC and TC. Predictions to know if the LDPC and the

TC would converge were then made with the use of an EXIT chart in iterative decoding, showing good performance of the parallel concatenated code (PCC) schemes [33] at particular SNR values. From their chart, it was possible to see that the Turbo Code showed a better performance than the concatenated regular LDPC codes in terms of cliff E_b/N_0, which is usually the dominant criterion for selection of coding schemes in communication systems. Eiko Seidel et. al. also showed in [33] that, the LDPC parallel concatenated code on the other hand converges faster to low bit error rates once the cliff E_b/N_0 is exceeded. Hanzo et. al. [34] in the paper titled "Near capacity irregular bit-interleaved coded modulation ", were able to exploit the use of an EXIT chart in the design of a joint inner-outer EXIT matching algorithm in order to design an Irregular Convolution Code (IrCC), Irregular Unitary Rate Code (IrURC) and Irregular Mapper (IrMapper) arrangement for obtaining a narrow but still open EXIT tunnel, which indicated a near-capacity operation as explained further in Chapter 4 of this thesis. The works of Maunder in [14] fully exploits the use of an EXIT chart in designing a near-capacity operation for a joint source channel coding.

In general, EXIT chart has been used for the design and construction of irregular codes also seen in the work of Michael Tuchler and Joachim Hagenauer [76] titled "Exit charts of irregular codes" by varying the degree of irregularities of constituent encoders with a view of ensuring that, a narrow tunnel gap exists in their EXIT charts.

In the late 90's when the EXIT chart was introduced, several authors managed to increase the coding gain of iteratively decoded Gallager (LDPC codes) codes by varying the number of parity check equations in which each codeword bit participated. Earliest work on irregular Gallager codes showed that by making the codeword bits participate in varying numbers of parity check equations, significant coding gains can be achieved [35][36][37]. For example in [35], the authors constructed new families of low-density parity-check codes, by using codes based on irregular random bipartite graphs which they called irregular codes (the first irregular code, which was an irregular LDPC code). Their codes could correct more errors than previously known low-density codes with a coding gain of 0.2 dB at a BER of 10^{-5} over a corresponding LDPC code. This improved

performance came from using codes based on irregular random bipartite graphs. Further work in [36] and [37] showed improved performance in irregular Gallager codes with codes, operating at 0.13 dB from the Shannon bound at a BER of 10^{-6}. EXIT charts have been used in this thesis to evaluate the performance of channel codecs as stated in Chapter 2.

3.6 Conclusion

In general, the unequal protection of information bits has been of interest, from the design of an irregular LDPC code to the design of an irregular TC. The significant performance benefits in these irregular codes stemmed from the extra protection on some of the transmitted bits. These recent and newer publications have focused on irregular turbo coding such as in [41] and [42] to achieve a very low bit error rate with no attention to a possible design of an Irregular Block Turbo Code. The challenges in designing a new irregular coding scheme would be its BER performance versus complexity. Designing an irregular code introduces some level of complexity in processing the extra protection given to certain bits, thereby altering the uniformity in their decoding. This pertains to the degrees used in varying the constituent encoder at the transmitter section. The performance of regular codes in comparison to their level of complexity as explained earlier makes them valid in today's communication system. The BER performance of the Turbo Code and the Block Turbo Code is examined in the next Chapter.

Chapter 4

BER performance of Turbo Code and Block Turbo Code (binary and non-binary)

4.1 Introduction

This Chapter discusses the BER performance of modern channel codecs in AWGN and wireless systems. In addition, the characterisation of the High Speed Downlink Packet Access (HSDPA) capacity using the EXIT chart technique at different code rates and with different modulation schemes is studied and shown. Furthermore, the EXIT charts of various binary Block Turbo Codes are plotted at different code rates and modulation schemes showing their distance to capacity in bits per channel use. The BER performance of the Turbo Code and the Block Turbo Code is studied with a view of examining how close they are to capacity. Distance to capacity is measured in two ways. First, is the E_b/N_0 distance to capacity and the other is the distance in bits per channel use.

4.2 Turbo Code in High Speed Downlink Packet Access (HSDPA)

In many of today's wireless systems, the Turbo Code is the standard channel codec in use for reliable information transmission. Examples of these are CDMA2000, Universal Mobile Telecommunication System (UMTS i.e. 3G cellular standard), the HSDPA (3.5G cellular standard) and the Long Term Evolution (LTE). Also by puncturing or inserting bits, Turbo Codes can be designed to operate at any code rate with its natural rates as 1/2, 1/3, 2/3 and 3/4. The development of the Turbo Code in [7] brought about a new concept in channel coding, which is the exchange of extrinsic information between decoders. This obviously seems to be the strength of a Turbo Code which is also the source of the name "Turbo Code". Today's HSDPA system uses the UMTS Turbo Code as depicted in figure 4.2.1.

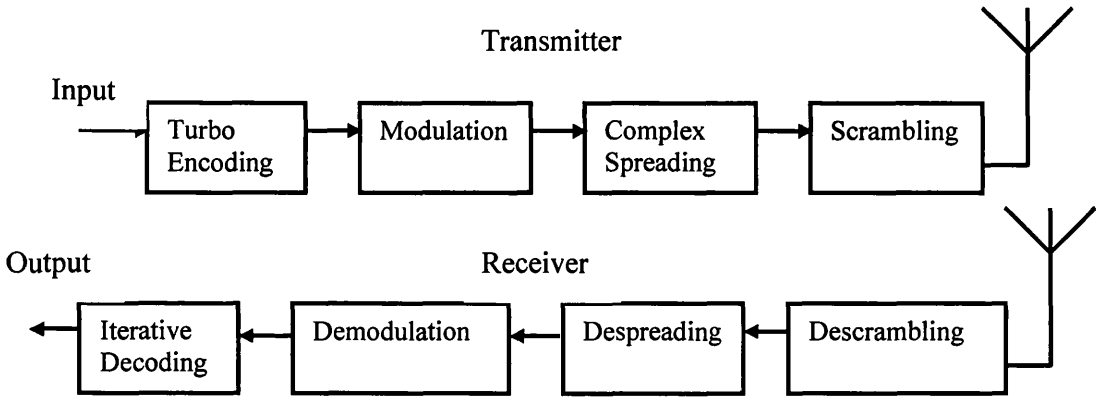


Figure 4.2.1 Schematic of HSDPA system model

The 3GPP W-CDMA system as described in the standard [43], but adapted for the High Speed Downlink Packet Access (HSDPA) was simulated in this study, as this was the current standard in use at the start of this study. At the transmitter, a binary stream is segmented into blocks encoded with a UMTS Turbo Code having octal 13 as the feedback and octal 15 as the feed forward taps. The symbols are then modulated and spread by Walsh codes with a spreading factor of 16, by assigning 1 modulation symbol per Walsh code as seen in [61]. The I and Q symbols are spread separately. The spread signals are then multiplied with the complex scrambling code at the same chip rate,

before transmitting over the I and Q channels, so that components from the I and Q channels cross over. The scrambling codes are essentially segments of long Gold codes, combined to create complex sequences, as described in [43]. In this thesis, the I and Q spread symbols are scrambled separately before transmission. In other to return the symbols to their normal order at the receiver, the complex conjugate of the same scrambling code is applied. Normal order here refers to descrambling and restoring the data bits into the originally spread signal for despreading. The despreader does the exact opposite of the complex spreading using the original spreading factor to despread the descrambled signal. After descrambling, despreading and demodulation, the received bit symbols are passed into an iterative decoder for decoding. This scenario is for a Gaussian channel and as such does not require a channel interleaver. The multipath channel is explained in sections 4.3 and 4.4.1. In the wireless multipath scenario, the wireless channel was assumed stationary i.e. no Vehicular speed. Also, ideal channel estimates are provided at the soft output of the systems demodulator for the 6 multipath taps.

As mentioned earlier in Chapter 2 of this thesis, the Convolutional Turbo Code (CTC) consists of two Convolutional codes in parallel, with the original data fed as input to the upper Convolutional code, while the lower Convolutional code receives an interleaved version of the original data. These Convolutional codes could be either recursive systematic Convolutional codes or non-systematic Convolutional codes. The recursive systematic Convolutional (RSC) code is usually preferred for use in a Turbo Code to the non-systematic Convolutional (NSC) code basically because in the latter, the input bits do not appear at its output. This simply means the codeword contains merged data and parity bits and cannot be separated into distinct regions of data and parity bits as desired in a Turbo Code. Hence, the RSC ensures that there is a clear distinction between the input data bits and the parity bits at the output of the encoder. Another key feature of the RSC is the feedback line from its output to the input. This feedback line in RSCs has been shown in [44] to increase the minimum weight of codewords which becomes dependent on the interleaver depth when used to construct Turbo Codes. This is not so

for an NSC. Hence the RSC has been used in the construction of Turbo Codes such as the UMTS Turbo Code which is used in this study.

The octally represented UMTS generator is shown in figure 4.2.2 with a feedback and feed forward polynomial of 13 and 15, respectively, for the RSCs. In figure 4.2.2, *X*, represents the information bit, *P1* and *P2* represents the punctured parity bits from the upper and lower RSCs, respectively, π represents the interleaver, while *D* represents the memory registers. The modulo 2 addition is used as the numeric system for the CTC.

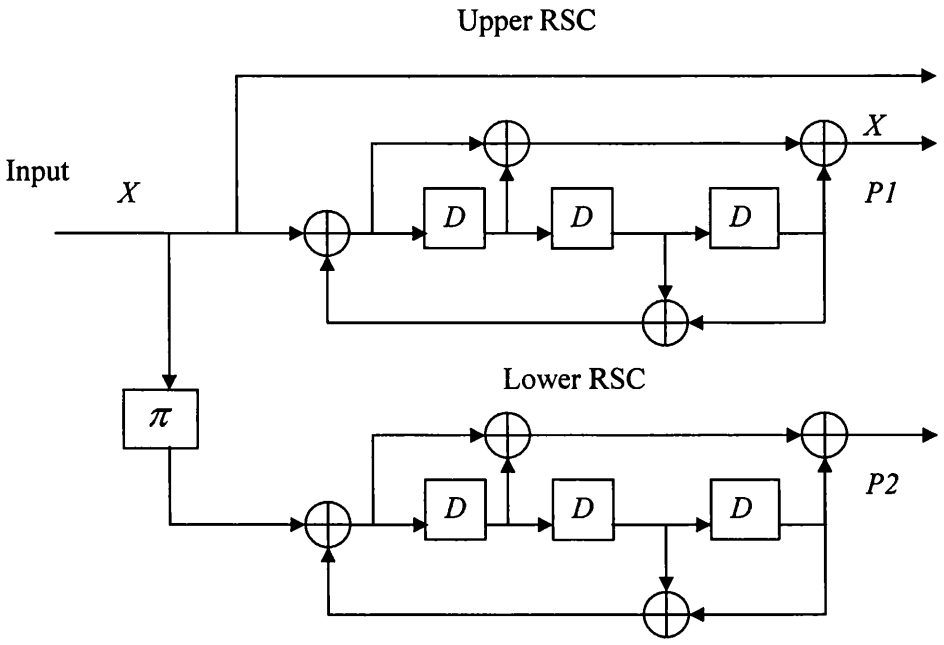


Figure 4.2.2 The UMTS Turbo Code encoder structure

The Log-MAP version of the BCJR (forward, backward decoding algorithm) [19] has been used as the soft-input soft-output decoder which uses the log likelihood ratios of the received channel information to correct as many noise corrupted bits as possible.

4.2.1 Log-likelihood equations

For a given transmitted bit u_k at time k, the log-likelihood ratio $L(u_k)$, is defined by (4.1).

$$L(u_k) = \ln\left[\frac{p(u_k = +1)}{p(u_k = -1)}\right] \tag{4.1}$$

since $$p(u_k = +1) = 1 - p(u_k = -1), \tag{4.2}$$

then $$L(u_k) = \ln\left[\frac{p(u_k = +1)}{1 - p(u_k = +1)}\right] \tag{4.3}$$

therefore

$$e^{L(u_k)} = \frac{p(u_k = +1)}{1 - p(u_k = +1)} \tag{4.4}$$

It can also be deduced that

$$\begin{aligned} p(u_k = +1) &= \frac{e^{L(u_k)}}{1 + e^{L(u_k)}} \\ &= \frac{1}{1 + e^{-L(u_k)}} \\ &= \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}\right).e^{L(u_k)/2} \end{aligned} \tag{4.5}$$

and

$$\begin{aligned} p(u_k = -1) &= 1 - \frac{1}{1 + e^{-L(u_k)}} \\ &= \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}} \\ &= \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}\right).e^{-L(u_k)/2} \end{aligned} \tag{4.6}$$

Equations (4.5) and (4.6) can be combined and represented by (4.7)

$$p(u_k = \pm 1) = \frac{e^{\pm L(u_k)}}{1 + e^{\pm L(u_k)}}$$
$$= \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}} \right) . e^{\pm L(u_k)/2}$$
$$= A_k . e^{\pm L(u_k)/2} \qquad (4.7)$$

Where $A_k = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}} \right)$ is common to the numerator and the denominator.

If the information bit u_k is conditioned on a received bit y_k, over a channel with a fading amplitude of a in an AWGN channel, then a conditional log-likelihood ratio $L(u_k \mid y_k)$ can be defined given by (4.8) and (4.9).

$$L(u_k \mid y_k) = \ln \left[\frac{p(u_k = +1 \mid y_k)}{p(u_k = -1 \mid y_k)} \right]$$

$$= \ln \left[\frac{p(y_k \mid u_k = +1).p(u_k = +1)}{p(y_k \mid u_k = -1).p(u_k = -1)} \right]$$

$$= \ln \left[\frac{p(y_k \mid u_k = +1)}{p(y_k \mid u_k = -1)} \right] + \ln \left[\frac{p(u_k = +1)}{p(u_k = -1)} \right]$$

$$= L(y_k \mid u_k) + L(u_k) \qquad (4.8)$$

Also we can write:

$$L(u_k \mid y_k) = \ln \left[\frac{p(y_k \mid u_k = +1).p(u_k = +1)}{p(y_k \mid u_k = -1).p(u_k = -1)} \right] \qquad (4.9)$$

$$p(y_k \mid u_k = +1) = \frac{1}{\sqrt{2\pi}\delta} . e^{\left(-\frac{E_b}{2\delta^2} . (y_k - a)^2 \right)} \quad \text{for } u_k = (+1).a \qquad (4.10a)$$

$$p(y_k \mid u_k = -1) = \frac{1}{\sqrt{2\pi}\delta} . e^{\left(-\frac{E_b}{2\delta^2} . (y_k + a)^2 \right)} \quad \text{for} \quad u_k = (-1).a \qquad (4.10b)$$

Where E_b and δ^2 are the transmitted energy per bit and the noise variance, respectively. Dividing equation (4.10a) by (4.10b) gives (4.10c).

$$\ln\left[\frac{p(y_k \mid u_k = +1)}{p(y_k \mid u_k = -1)}\right] = \ln\left[\frac{e^{\left(-\frac{E_b}{2\delta^2}.(y_k-a)^2\right)}}{e^{\left(-\frac{E_b}{2\delta^2}.(y_k+a)^2\right)}}\right] \qquad (4.10c)$$

Equation (4.9) can then be re-written as (4.11).

$$\begin{aligned}
&= \ln\left[\frac{e^{\left(-\frac{E_b}{2\delta^2}.(y_k-a)^2\right)}}{e^{\left(-\frac{E_b}{2\delta^2}.(y_k+a)^2\right)}}\right] + \ln\left[\frac{p(u_k = +1)}{p(u_k = -1)}\right] \\
&= \ln\left(e^{\left(-\frac{E_b}{2\delta^2}(y_k^2-2.a.y_k+a^2-y_k^2-2.a.y_k-a^2)\right)}\right) + L(u_k) \\
&= L_c.y_k + L(u_k) \\
&= 4.a.\frac{E_b}{2\delta^2}.y_k + L(u_k) \qquad (4.11)
\end{aligned}$$

Where $L_c = 4.a.\frac{E_b}{2\delta^2}$ is the channel reliability factor used in the simulation or analysis and $L_c.y_k$ is the channel observation. In an AWGN channel a=1 is used while in a flat faded channel, a takes the value of the fading amplitude.

4.2.2 Iterative decoding

The decoding structure for a Turbo Code depicted in figure 4.2.3 consists of two SISO (soft input soft output) decoders.

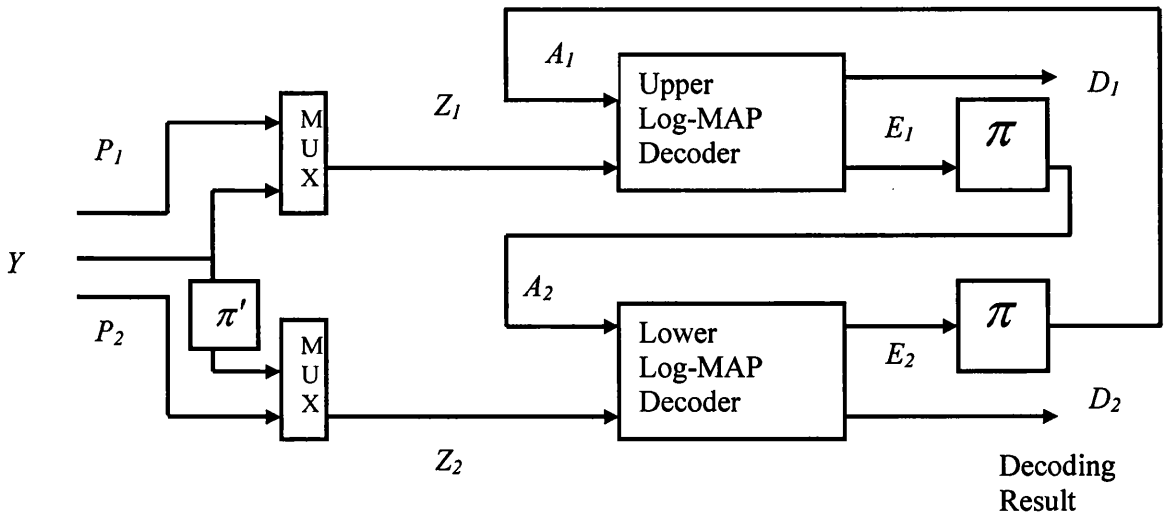


Figure 4.2.3 Schematic of Turbo Code decoding structure

In many cases, these decoders are Maximum *A Posteriori* (MAP) decoders derived from the BCJR algorithm [19]. In our decoding structure, the Log-MAP version of the BCJR has been used due to its near optimal performance but lower decoding complexity in comparison with the MAP since the Log-MAP makes use of the Jacobian approximation (explained later in this sub-section) and operates in the log likelihood domain.

In a given systematic Convolutional code, the soft output from the BCJR decoder is defined as the *a posteriori* log-likelihood ratio. For a received bit sequence y the probability that a decoded bit u_k is a +1 or a -1 which is the same as *a posteriori* value is given by (4.12).

$$L(\hat{u}_k) = L(u_k \mid y) = \ln\left[\frac{p(u_k = +1 \mid y)}{p(u_k = -1 \mid y)}\right] \tag{4.12}$$

This can be represented as the sum of three terms.

$$L(\hat{u}_k) = L_c.y_k + L(u_k) + L_e(\hat{u}_k) \tag{4.13}$$

where $L_c.y_k$ is the channel observation, $L(u_k)$ is the *a priori* value and $L_e(\hat{u}_k)$ is a new value gleaned from the decoder called the extrinsic information. The trellis of a binary Convolutional encoder is shown in figure 4.2.4. The UMTS Turbo Code encoder uses systematic encoding. Figure 4.2.4 illustrates the trellis of a systematic Convolutional code where s'_{k-1} and s_k denotes the trellis states at times $k-1$ and k, respectively. The branch connecting time $k-1$ to k on the trellis is labelled ± 1, where ± 1 i.e. u_k are the information bits. The following mathematical derivations are presented in [18] for systematic Convolutional trellis decoding and also then used in the Log-MAP trellis decoding of the UMTS Turbo Code simulated in this Chapter.

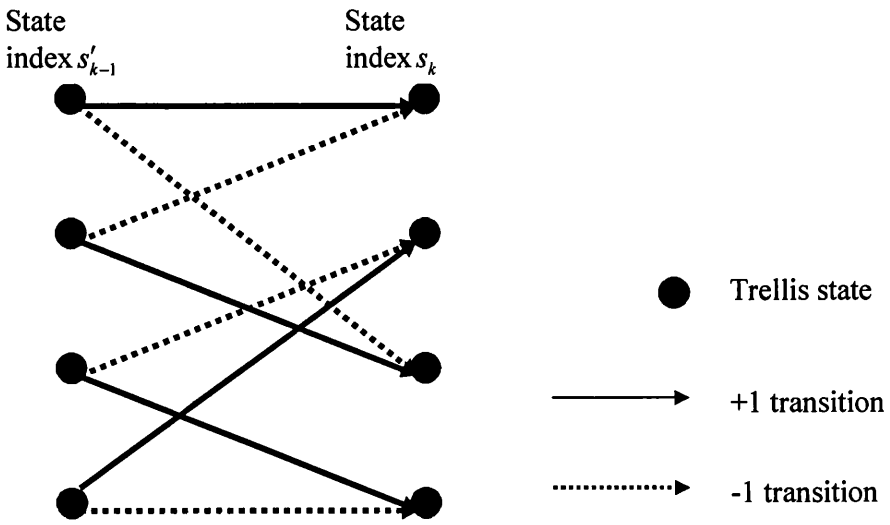


Figure 4.2.4 Example of a trellis structure for a systematic Convolutional code

Firstly, for every received bit sequence y, the transition from state s'_{k-1} to s_k in the trellis means equation (4.12) can be re-written as (4.14). These transitions are mutually exclusive thereby making the probability that any of them occurred as the sum of their individual probabilities. The range of the $\sum$ (summation sign) in the following equations equals the length of the received bit sequence.

$$L(\hat{u}_k) = \ln\left[\frac{p(u_k = +1 \mid y)}{p(u_k = -1 \mid y)}\right] = \ln\left[\frac{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k = +1}} p(s'_{k-1}, s_k, y)}{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k = -1}} p(s'_{k-1}, s_k, y)}\right] \tag{4.14}$$

The term $p(s'_{k-1}, s_k, y)$ in (4.14) is the sum of the joint probabilities of the received sequence in transition from one state to another and can be re-written as (4.15).

$$\begin{aligned} p(s'_{k-1}, s_k, y) &= p(s'_{k-1}, y_{j<k}).p(s_k, y_k \mid s'_{k-1}).p(y_{j>k} \mid s_k) \\ &= p(s'_{k-1}, y_{j<k}).p(s_k \mid s'_{k-1}).p(y_k \mid s'_{k-1}, s_k).p(y_{j>k} \mid s_k) \\ &= \alpha_{k-1}(s'_{k-1}).\gamma_k(s'_{k-1}, s_k).\beta_k(s_k) \end{aligned} \tag{4.15}$$

i.e. $p(s'_{k-1}, y_{j<k}) = \alpha_{k-1}(s'_{k-1})$, $p(s_k \mid s'_{k-1}).p(y_k \mid s'_{k-1}, s_k) = \gamma_k(s'_{k-1}, s_k)$ and $p(y_{j>k} \mid s_k) = \beta_k(s_k)$

In equation (4.15), $y_{j<k}$ represents the sequence of received bits y_j from the start of the trellis up to time $k-1$, while $y_{j>k}$ corresponds to the sequence of received bits y_j from time $k+1$ to the end of the trellis. From the foregoing equations, the forward and backward recursions of the BCJR algorithm are,

$$\alpha_k(s_k) = \sum \gamma_k(s'_{k-1}, s_k).\alpha_{k-1}(s'_{k-1}) \tag{4.16}$$

$$\beta_{k-1}(s'_{k-1}) = \sum \gamma_k(s'_{k-1}, s_k).\beta_k(s_k) \tag{4.17}$$

Equation (4.12) can also be re-written as (4.18).

$$L(\hat{u}_k) = \ln\left[\frac{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k = +1}} \gamma_k(s'_{k-1}, s_k).\alpha_{k-1}(s'_{k-1}).\beta_k(s_k)}{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k = -1}} \gamma_k(s'_{k-1}, s_k).\alpha_{k-1}(s'_{k-1}).\beta_k(s_k)}\right] \tag{4.18}$$

A transition from s'_{k-1} to s_k with $p(s_k \mid s'_{k-1}) = p(u_k)$, where u_k is an information bit corresponding to the transition from s'_{k-1} to s_k, has a branch transition probability given by (4.19).

$$\begin{aligned}\gamma_k(s'_{k-1}, s_k) &= p(s_k \mid s'_{k-1}).p(y_k \mid s'_{k-1}, s_k) \\ &= p(y_k \mid u_k).p(u_k)\end{aligned} \tag{4.19}$$

Further mathematical derivations in [19] show that the branch transition probability from one state to another is given by (4.20), assuming n transmitted bits and $x_{k1} = u_k$.

$$\gamma_k^{(e)}(s'_{k-1}, s_k) = e^{\left(\frac{1}{2}\sum_{v=2}^{n} L_c \cdot y_{k,v} \cdot x_{k,v}\right)} \tag{4.20}$$

The log-likelihood ratio in equation (4.13) is given by (4.21).

$$L(\hat{u}_k) = L_c.y_k + L(u_k) + \ln \frac{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k=+1}} \gamma_k^{(e)}(s'_{k-1}, s_k).\alpha_{k-1}(s'_{k-1}).\beta_k(s_k)}{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k=-1}} \gamma_k^{(e)}(s'_{k-1}, s_k).\alpha_{k-1}(s'_{k-1}).\beta_k(s_k)} \tag{4.21}$$

From (4.21) the extrinsic information is then given by (4.22),

$$L_e(\hat{u}_k) = \ln \frac{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k=+1}} \gamma_k^{(e)}(s'_{k-1}, s_k).\alpha_{k-1}(s'_{k-1}).\beta_k(s_k)}{\sum_{\substack{(s'_{k-1}, s_k) \\ u_k=-1}} \gamma_k^{(e)}(s'_{k-1}, s_k).\alpha_{k-1}(s'_{k-1}).\beta_k(s_k)} \tag{4.22}$$

The BCJR algorithm can then be modified using some approximations to give a Log-MAP by using $\ln(e^{L_1} + e^{L_2}) \approx \max(L_1, L_2)$ in the computation of the extrinsic information as used in this study and equation (4.21), since $\ln \frac{A}{B} = \ln[A + (-B)] = \max[A, (-B)]$.

Extensive treatments of the BCJR concept can be found in many literatures such as [18], [19] and [83].

In figure 4.2.3, Y, P_1 and P_2, are the received soft-value channel observations corresponding to the information bits, parity bits for the upper decoder and parity bits for

the lower decoder, respectively. Z_1, is composed of the received information bits Y and its corresponding parity P_1, which goes into the upper decoder , while Z_2, composed of the received information bits Y and its corresponding parity P_2, goes into the lower decoder. A_1, A_2 are *a priori* inputs to the upper and lower decoder respectively, with E_1 and E_2 as part of the *a posteriori* outputs called extrinsic information, gleaned from the upper and lower decoders respectively. D_2 represents the final decoded output with D_1 a part of the *a posteriori* output from the upper decoder. The variables Z_1, A_1, D_1, E_1, Z_2, A_2, D_2, and E_2 all denote log-likelihood ratios (i.e. *LLR*-values with Y corresponding to the received bit $L(u_k)$). In the iterative decoder, the upper decoder takes the channel observation Z_1 on the received systematic (information) bits Y and respective parity P_1 together with an initial *a priori* input A_1 (A_1 is zero on the first iteration) obtained from the lower decoder i.e. $A_1 = E_2 = D_2 - A_2 - Z_2$. The upper decoder then outputs an *a posteriori* D_1. An extrinsic information E_1 is gleaned from the upper decoder i.e. $E_1 = D_1 - A_1 - Z_1$, which is then passed through an interleaver to become the *a priori* input A_2 for the lower decoder. The lower decoder then takes the interleaved channel observations Z_2 on the systematic bits Y and respective parity bits P_2 together with the *a priori* A_2 and outputs an *a posteriori* D_2. A second extrinsic information E_2 is then gleaned from the lower decoder i.e. $E_2 = D_2 - A_2 - Z_2$, which is fed back into the upper decoder via an interleaver thus becoming the *a priori* knowledge A_1 for the upper decoder. This iteration process is repeated until a stop criterion set by the decoder is reached. In many cases the stop criterion is determined by the value of the bit error rate. These log-likelihood variables,(D_1, E_1, Z_2, A_2, D_2, and E_2) which are soft information values, are eventually used to determine the originally transmitted bits by estimating the *LLR* values below zeros as bit 0 and the values above zero as bit 1.

4.3 HSDPA system evaluation.

The 3GPP W-CDMA system for the HSDPA model described in section 4.2 above and depicted in figure 4.2.1 was modelled by computer simulation using MATLAB. Walsh codes with a spreading factor of 16 and Gold sequences were used as the spreading code

and scrambling code in the HSDPA system, respectively. A random interleaver with a frame size of 5012 bits per frame was used. A frame consists of 5012 bits. The simulation was performed for a 3GPP multipath vehicular A channel [43] including AWGN. The data was generated in packets, once every transmission time interval (TTI). Each packet (or TTI) was comprised of 8 frames in accordance to the standard in [43]. A chip rate of 3.84Mchip/s was also used. RAKE receivers (6 fingers) were used for the slowly faded multipath channel. The octally represented UMTS generator and feedback polynomial of the RSC were 15 and 13, respectively. UMTS Turbo Code rates used were 1/2 and 1/3 and the simulations were performed in the BPSK, QPSK, 16QAM and the 64QAM modulation schemes in the AWGN channel, while QPSK, 16QAM and 64QAM were evaluated in the 3GPP multipath channel. The log-MAP version of the BCJR algorithm was used as the decoder in all cases. For each simulation, a curve showing the BER versus the energy per bit to noise energy ratio (E_b/N_0) is graphed, where the noise energy is the single-sided noise spectral density N_0 of the channel. The performance of the UMTS Turbo Code was evaluated so as to establish a benchmark for comparison with the designed Irregular Turbo Code in Chapter 5.

4.4 Bit error rate performance of Turbo Code in BPSK, QPSK, 16QAM and 64QAM modulation schemes

In this section, the BER performance of the Convolutional Turbo Code (CTC) is evaluated via simulations in four modulation schemes at code rates 1/2 and 1/3 to determine their distances to channel capacity in terms of E_b/N_0 value. In section 4.5, the distance to capacity in bits per channel use was also evaluated. To determine their distance to capacity in bits per channel use, the EXIT charts of the CTCs were plotted and the area properties of their respective EXIT charts were then used to determine the capacity of the channel in bits per channel use. The evaluation was examined in an AWGN channel and then in a 3GPP Vehicular A multipath wireless channel. The evaluation also takes into account the number of iterations required to achieve a very low bit error rate in the region of 10^{-5}. The simulations were done using MATLAB. Random bits of zeros and ones were generated and encoded with the CTC. The encoded

bits were then sent through the channels named above. The received corrupted bits were then sent into the turbo decoder described in sub-section 4.2.2, where the ratio of the number of bits in error to the total number of randomly generated bits was computed. Gray mapping was used for the QPSK, 16QAM and 64QAM modulation schemes.

Figures 4.4.1 to 4.4.4 illustrate the BER versus E_b/N_0 performance of a CTC as used in an HSDPA system model.

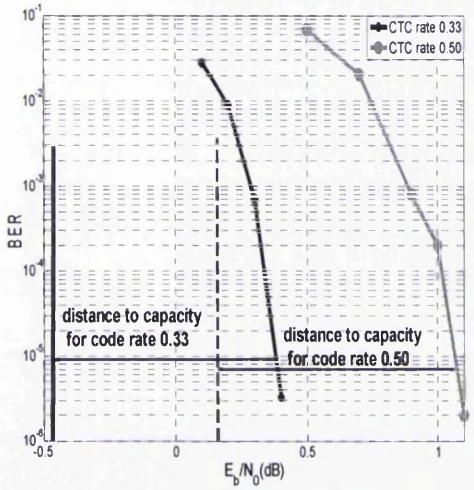


Figure 4.4.1 BER performance of CTC in BPSK.

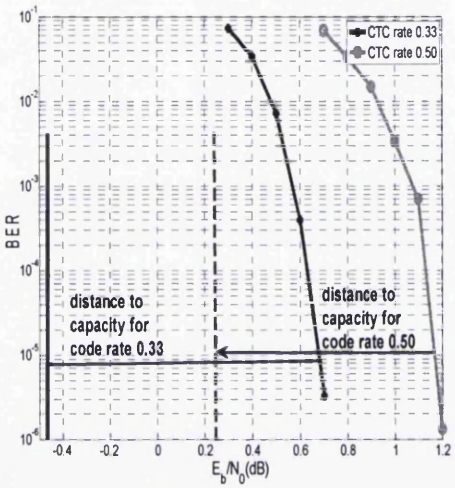


Figure 4.4.2 BER performance of CTC in QPSK.

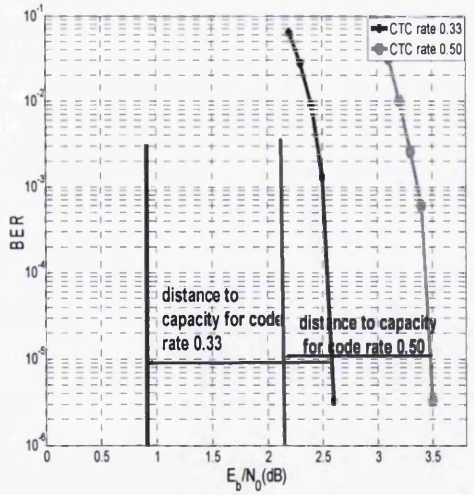


Figure 4.4.3 BER performance of CTC in 16QAM.

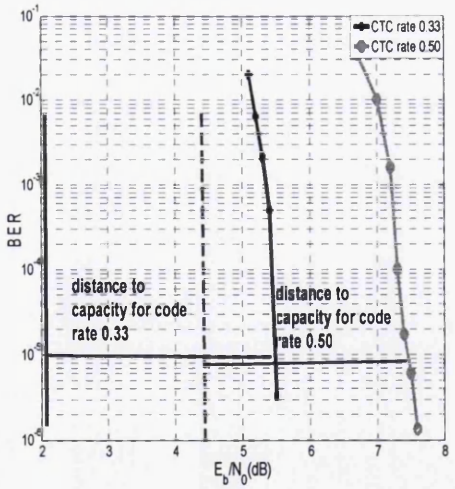


Figure 4.4.4 BER performance of CTC in 64QAM.

The BER curve results for the WCDMA (UMTS) 3GPP Turbo Code in [77],[78],[79]and [80] are the same as the BER curves shown in this section.

The BER curves shown in figures 4.4.1 to 4.4.4 have the characteristic of a waterfall region, which is typical of a Turbo Code as the probability of error falls sharply as the E_b/N_0 increases. This characteristic of a Turbo Code makes it one of the reasons it has been used as a channel codec in modern communication systems besides its closeness to the Shannon bound as illustrated in figures 4.4.1 to 4.4.4. Table 4.1 shows the performance of the CTC tabulated together with their corresponding throughput per channel use which is given by $S = R \times (\log_2 M)(1 - FER)$, where R is the rate of the code, FER the frame error rate and M the M-ary order of the modulation scheme. The frame error rate is computed by dividing the number of frames in error after decoding by the total number of frames used in the simulation. In determining the *FER*, the number of frames corresponding to at least ten frames in errors per E_b/N_0 value was used. A frame in error means, a single or more bits in the frame is received in error after decoding. Table 4.1 also shows converging E_b/N_0 values for the CTCs at different code rates and modulation schemes. It can be deduced from table 4.1 that the higher the modulation order, the further their distances to Shannon's capacity.

Shannon's capacity in this thesis is measured first in dB, which is the minimum E_b/N_0 required to achieve a low BER for a Discrete-input Continuous-output Memoryless Channel (DCMC), and is a function of the modulation scheme and code rate [1]. Secondly, capacity is measured by the maximum possible throughput in bits per channel use at a particular E_b/N_0 value at which information can be sent with arbitrarily low probability of error [13], i.e. the maximum mutual information per single channel use, which is also a function of the modulation scheme and code rate denoted as C_{DCMC} in this thesis. However, C_{DCMC} as a measure of distance to capacity, takes into account the system's spectral efficiency as well as the energy efficiency unlike the E_b/N_0 in dB which only takes into account the energy efficiency, making C_{DCMC} a better option.

Table 4.1: CTC Converging E_b/N_0 values for AWGN

Modulation Scheme	Code rate	E_b/N_0 (dB)	S	Number of iterations	Shannon's Capacity E_b/N_0 (dB)
BPSK	1/3	0.400	0.329	11	-0.430
BPSK	1/2	1.100	0.490	15	0.210
QPSK	1/3	0.700	0.665	15	-0.41
QPSK	1/2	1.180	0.997	11	0.230
16QAM	1/3	2.600	1.328	14	0.750
16QAM	1/2	3.500	1.993	21	2.300
64QAM	1/3	5.500	1.996	24	2.300
64QAM	1/2	7.500	2.960	19	4.500

* S in table 4.1 stands for the throughput value in bits per channel use.

The performance of the CTC in a 64QAM modulation scheme is worse than other modulation schemes in terms of throughput value against required E_b/N_0 to achieve a BER of 10^{-5}. This is mainly due to the dense constellation cluster (together with the number of bits per symbol) required for a 64QAM modulation scheme, thereby requiring a far higher E_b/N_0 value to achieve a very low probability of error. As an illustration, a CTC in a 64QAM modulation scheme at a code rate of 1/3 would give a throughput value of about 1.996 bits per channel use at an E_b/N_0 value of 5.4 dB. In comparison to the CTC in a 16QAM modulation scheme, a bit per channel use of 1.993 requires an E_b/N_0 value of 3.4 dB which is a significant 2 dB less than that required for the CTC in a 64QAM scheme for the same throughput. In this section, the BER performance of the CTC in a Gaussian channel with four different modulation schemes was examined, with a view to determining their distances to their respective E_b/N_0 capacity in dB. In the next section the multipath channel is considered.

4.4.1 Convolutional Turbo Code in 3GPP wireless multipath channel

In this section, the BER performance of the CTC is evaluated via simulations in the 3GPP multipath channel in the HSDPA standard modulation schemes, (i.e. QPSK, 16QAM and 64QAM) at code rates 1/2 and 1/3 with the exception of 64QAM code rate 1/2. The data in the modelled 3GPP multipath channel is generated in packets, once in every TTI with each packet or TTI consists of 8 frames as said earlier in section 4.3. Also 2 interleaving processes are applied to the data as stipulated in [43]. Firstly, the whole 8 frames in each TTI is interleaved so that data is interleaved across frames. The other interleaving process occurs within each frame. The wireless channel used is the 3GPP multipath Vehicular A [84] with 6 paths of various tap weights as used to evaluate the standard. Each of the taps is an independent Rayleigh faded tap with the channel changing once per TTI.
The RAKE receiver used has 6 branches, for each multipath component. The phase correction and weighting of the received signal is carried out within the soft decision metric as described by the HSDPA Technical Report on the soft decoding metric documentation [43]. The received data is sent directly to the RAKE branches.

A wireless channel could be categorised a fast or slow fading channel depending on how fast the channel impulse response changes in comparison to the symbol duration. The minimum time required for a magnitude change of a channel to become uncorrelated from its previous value is called the coherent time [82]. A fast faded channel is one in which the channel impulse response changes rapidly within a symbol duration. Practically, fast fading occurs when the coherent time of the channel is small relative to the delay constraint of the channel [82], i.e. the symbol period is large enough to permit frequent changes to the channel's impulse response within a symbol period. On the other hand, a slow fading channel occurs when the coherent time of the channel is large relative to the delay constraint of the channel [82], i.e. the channel's impulse response changes at a rate slower than the transmitted symbols. This ensures that the impulse response seen during transmission remains the same or constant, resulting in constant

fading amplitude over several symbols. A slow fading channel has been assumed in this thesis. The presence of the multipath requires the use of a RAKE receiver [43] with 6 fingers which uses maximum ratio combining. The BER versus E_b/N_0 curves shown in figures 4.4.5 to 4.4.7 depicts the typical performance of a CTC in the 3GPP Vehicular A multipath channel.

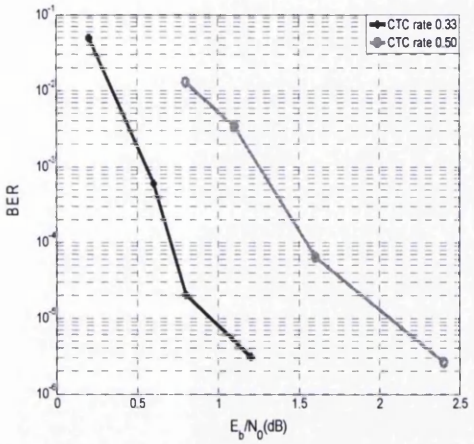


Figure 4.4.5 BER performance of CTC in the 3GPP multipath Vehicular A channel for QPSK.

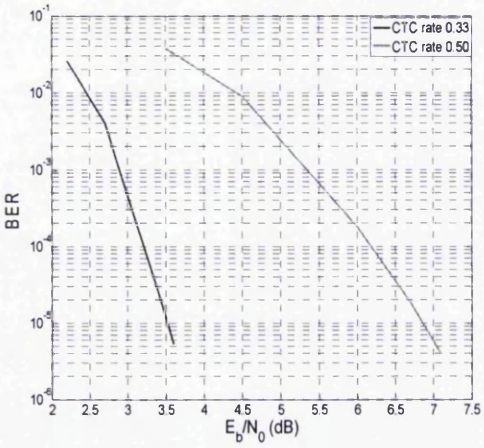


Figure 4.4.6 BER performance of CTC in the 3GPP multipath Vehicular A channel for 16QAM.

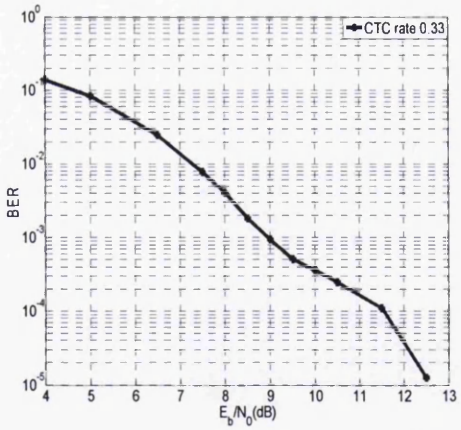


Figure 4.4.7 BER performance of CTC in the 3GPP multipath Vehicular A channel for 64QAM.

The distance to capacity in bits per channel use of a CTC can be evaluated using the area properties of an EXIT chart as said in an earlier part of this Chapter. In the next section, a detailed explanation of the EXIT chart and its area properties is examined.

4.5 EXIT charts of Turbo Codes and their area properties

Since the introduction of channel capacity by Shannon [1], the distance of channel codecs to capacity has been used to determine their efficiency. A tool that can be used to determine the capacity of a channel and the corresponding capacity loss of a channel codec in bits per channel use is the EXIT chart. The EXIT chart is also a tool used to predict the converging E_b/N_0 values of iterative decoding schemes. To predict if an iterative decoding scheme will converge (i.e. give negligible BER) at a particular E_b/N_0, an EXIT chart is plotted at that E_b/N_0 value. An open tunnel gap in the EXIT chart suggests convergence, while closed or crossed-over tunnel gaps suggest no convergence. In an EXIT chart, the narrower the tunnel gap, the closer the iterative decoding scheme is to capacity. This is the key procedure used in designing and improving iterative decoding schemes. The EXIT chart, which is an acronym for extrinsic information transfer chart developed by S. Ten Brink [24], is a graphical description of the convergence process for iterative decoding. In EXIT charts, the transfer characteristics based on mutual information is used to describe the flow of extrinsic information through the soft input soft output constituent decoders of an iterative decoder [27]. It is a tool used for a clearer understanding of the convergence behaviour of iterative decoding schemes. A decoding trajectory is then used to visualize the exchange of extrinsic information between the constituent decoders. The EXIT chart is a tool used to find the least E_b/N_0 to keep the tunnel gap in the EXIT chart open as well as an alternative means to compute the capacity of a channel which is described in details in the next sub-sections.

4.5.1 Extrinsic transfer characteristics using iterative decoder for parallel concatenated scheme

The iterative decoder for a parallel concatenated code (PCC), e.g. a Convolutional Turbo Code, was shown in figure 4.2.3. By definition from figure 4.2.3, Y, P_1 and P_2, are the received channel observations corresponding to the information bits, parity bits for the upper decoder and parity bits for the lower decoder, respectively. Z_1, is composed of the received information bits Y and its corresponding parity P_1, which goes into the upper decoder , while Z_2, composed of the received information bit Y and its corresponding parity P_2, goes into the lower decoder. A_1, A_2 are *a priori* inputs to the upper and lower decoder respectively, with E_1 and E_2 as part of the *a posteriori* outputs called extrinsic information, gleaned from the upper and lower decoders respectively. D_2 represents the final decoded soft output with D_1 a part of the *a posteriori* output from the upper decoder. The variables Z_1, A_1, D_1, E_1, Z_2, A_2, D_2, and E_2 all denote log-likelihood ratios (*LLR*-values). The upper decoder takes the channel observation Z_1 on the received systematic (information) bits Y and respective parity P_1 together with an initial *a priori* input A_1 (A_1 is zero for the first iteration) obtained from the lower decoder. The upper decoder then outputs an extrinsic information E_1 gleaned from itself ($E_1=D_1-A_1-Z_1$) which is then passed through an interleaver to become the *a priori* input A_2 of the lower decoder. The lower decoder then takes the interleaved channel observations Z_2 on the systematic bits Y and respective parity bits P_2 together with the *a priori* A_2 and gleans an extrinsic information $E_2=D_2-A_2-Z_2$, which is fed back into the upper decoder via an interleaver becoming the *a priori* knowledge A_1 for the upper decoder. A plot of the mutual information (I_a) between the *a priori* A and transmitted bits X on the abscissa and the mutual information (I_e) between the extrinsic information E and transmitted bits X of the upper decoder on the ordinate, gives the EXIT function of that decoder. A second plot of an "inverted" EXIT function for the lower decoder is used to explain the convergence of the iterative decoding process. The term "inverted" here simply connotes swapping of the plot's axis, i.e. *a priori* on the ordinate with the extrinsic information on the abscissa. The mutual information $I(X, A)$ between the *a priori* input and transmitted

bits X is given as I_a, while the mutual information $I(X; E)$ between the extrinsic output of a decoder and the transmitted bit X is denoted by I_e. In a typical EXIT chart, I_{a1} denotes the mutual information $I(X, A_1)$ between the *a priori* input to the upper decoder and the transmitted bits X, I_{e1} for the mutual information $I(X; E_1)$ between the extrinsic output of the upper decoder and the transmitted bits X with, I_{a2} and I_{e2} representing the same for the lower decoder respectively. I_{a1}, I_{e1}, I_{a2} with I_{e2} range from 0 to 1 in value. It is important to note that interleaving does not change the inherent mutual information meaning I_{a1} is actually I_{e2} upon de-interleaving as shown in [27]. It should also be noted that, the two EXIT functions in a PCC depends on the E_b/N_0 value being used, simply because the two decoders have inputs from the same channel observation as shown in [26] [45].

4.5.2 Extrinsic transfer characteristics using iterative decoder for a serially concatenated scheme

In this sub-section, a detailed explanation of the EXIT chart of serially concatenated codes (SCC) is explained. In an SCC, the outermost encoder is known as the outer encoder, while the other is called the inner encoder. The corresponding decoder to each of these encoders is also known as the outer and the inner decoders. The EXIT chart of a serially concatenated scheme is similar to that of a parallel concatenated scheme except that, the inner code's EXIT function depends on the E_b/N_0, because it directly receives the channel observations, while the outer code's EXIT function does not depend on the E_b/N_0 as it does not directly receive any channel observations. Mathematically, to measure the information contents of an *a priori* knowledge A, the mutual information $I_a = I(X, A)$ between the transmitted systematic bits X and the LLR values of A as found in [27] are used and given by (4.21).

$$I_a = \frac{1}{2} \cdot \sum_{x=-1,1} \int_{-\infty}^{+\infty} p_a(\xi \mid X = x) \times \log_2 \left[\frac{2.p_a(\xi \mid X = x)}{p_a(\xi \mid X = -1) + p_a(\xi \mid X = +1)} \right] d\xi \qquad (4.23)$$

With $p_a(\xi \mid X = x)$ representing the conditional probability density function belonging to the *LLR* A. The mutual information $I_e = (X, E)$ between the transmitted systematic bits X and the *LLR* values of E (the extrinsic information) as found in [27] is given by (4.24).

$$I_e = \frac{1}{2} \cdot \sum_{x=-1,1} \int_{-\infty}^{+\infty} p_e(\xi \mid X = x) \times \log_2 \left[\frac{2.p_e(\xi \mid X = x)}{p_e(\xi \mid X = -1) + p_e(\xi \mid X = +1)} \right] d\xi \qquad (4.24)$$

with $p_e(\xi \mid X = x)$ representing the conditional probability density function belonging to the *LLR* E. The mutual information for an *LLR* frame at a particular value (say 0, 0.1, 0.2,……., 1.0) depends on the distribution of the *LLR* values in that frame corresponding to zero-valued bits and that of the *LLR* values in the frame pertaining to unity-valued bits. If the distribution of the *LLR* values that corresponds to zero-valued bits equals that of the *LLR* values pertaining to unity-valued bits, then the mutual information will be zero. In this case, the *LLR* values are totally unreliable and the selection of a bit's logical value based upon the sign of the corresponding *LLR* will only give the correct answer 50% of the time [14]. To Compute the mutual information in transmission systems with *a posteriori* (*APP*) decoders, the "averaging method is capable of calculating the mutual information of an *LLR* frame without considering the corresponding originally transmitted bit frame X, provided that the *LLR* is generated by an optimal *APP* decoder. Another method used in calculating mutual information is the histogram method [14][27]. In this thesis, the histogram and the averaging methods were used and found to have very similar results. This illustrates the fact that employing either method would yield a correct EXIT chart. The results generated in this thesis were from the histogram method. One key factor in generating an EXIT Chart is that $I_{e1}=I_{a2}$ must be well interleaved with large interleaver depth to ensure that they are uncorrelated. To plot an

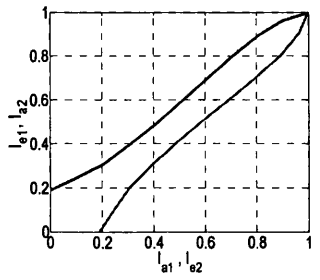
EXIT Chart, all that is required is the mutual information between these *LLR's* i.e. *I(X, A)* and *I(X, E)* with $I(X, A) = H(X) - H(X/A)$ and $I(X, E) = H(X) - H(X/E)$.

4.5.3 Parameters influencing the transfer characteristic curve of an EXIT chart

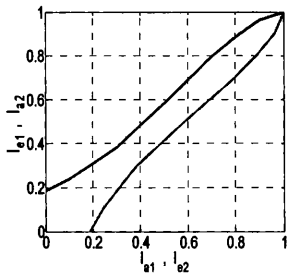
There are some important parameters that impact on the transfer characteristic curve of EXIT charts. These are:

- Constraint length of the code in the case of a Convolutional code;
- Depth of interleaving;
- Different code polynomials ;
- E_b/N_0;
- Shape of the inverted EXIT function.

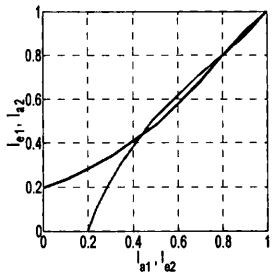
EXIT charts for the UMTS CTCs in an HSDPA model at their converging E_b/N_0 values for various code rates and modulation schemes are shown by way of example in figures 4.4.8(a) to (i).



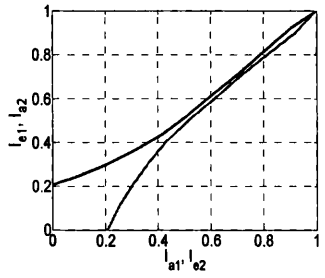
(a) 0.4dB ,BPSK, rate 1/3



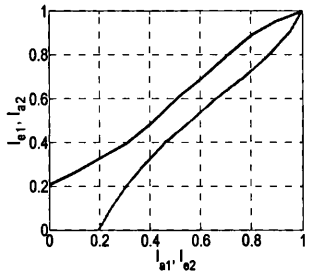
(b) 1.1dB, BPSK, rate 1/2



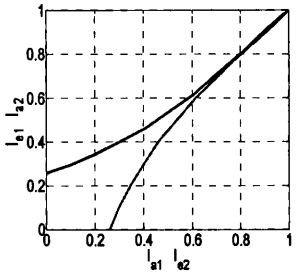
(c) 0.6dB, QPSK, rate 1/3



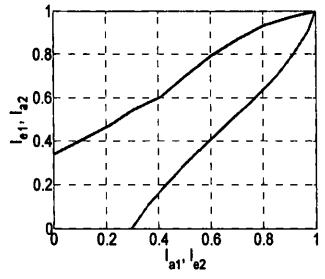
(d) 0.7dB, QPSK, rate 1/3



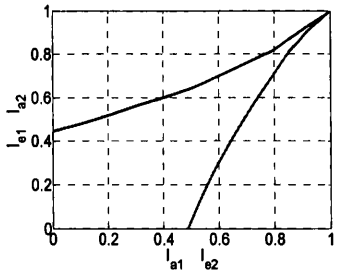
(e) 1.18dB, QPSK, rate 1/2



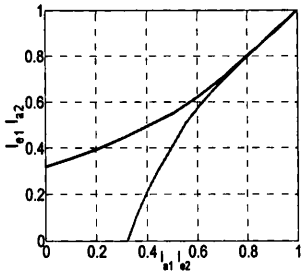
(f) 2.6dB, 16QAM, rate 1/3



(g) 3.5dB, 16QAM, rate 1/2



(h) 5.5dB, 64QAM, rate 1/3



(i) 7.5dB, 64QAM, rate 1/2

Figure 4.4.8 EXIT charts for CTC at code rate 1/3 and 1/2
Legend: 'E_b/N_0 value', 'modulation', 'code rate'
The figures above are labeled in line with the legend, i.e. the E_b/N_0 value, followed by the modulation scheme and then the code rate.

Many of the EXIT charts shown in figure 4.4.8 have an open tunnel gap until the point (1, 1). The open tunnel gap denotes that a very low bit error rate during bit transmission at those E_b/N_0 values will occur. When the gap closes or crosses over before the (1,1) point, the BER does not converge to a negligible value (i.e. the case of figure 4.4.8(c)).

The scenario in figure 4.4.8(d) is the same as that of figure 4.4.8(c) but with a higher E_b/N_0 (0.1 dB difference) giving an open but very narrow tunnel gap. A Parallel concatenated code has two decoders in parallel, which can be called, upper and lower decoders. In the case of a serially concatenated code, the two decoders are serially concatenated and can be called inner and outer decoders. In [47], the authors have shown that the area A_{in} underneath the EXIT function of an inner code for a serially concatenated code can be given by (4.25).

$$A_{in} = \int_0^1 I_e(I_a) dI_a = \frac{I(X,Y)}{R_{in}} \tag{4.25}$$

Where I_e is the extrinsic output values for the inner decoder, I_a the *a priori* input values for the inner decoder, I_e is a function of I_a, i.e. $I_e(I_a)$, $I(X,Y)$ is the maximum mutual information transfer between the transmitted symbol X and received symbol Y, also known as capacity, and R_{in} is the rate of the inner code. I_e which is a function of I_a is integrated with respect to I_a (i.e. dI_a). This implies that for a rate one inner code, the area underneath the inner code equals the capacity of the communication channel and for inner codes with $R_{in} < 1$, the area underneath the inner code is an attainable capacity (a slightly lower capacity bound). For a given modulation scheme, this attainable capacity C_A can be calculated from (4.26).

$$C_A = A_{in} \times R_{in} \times \log_2 M \tag{4.26}$$

Where M is the M-ary order of the modulation scheme.

In the case of the parallel concatenated code, the sum of areas underneath the EXIT functions for the CTCs equals the attainable capacity C_A [26] given by (4.27).

$$A = (A_u + A_l) = \int_0^1 I_e(I_a) dI_a = \frac{I(X,Y)}{R} \tag{4.27}$$

Where A_u is the area underneath the EXIT function of the upper code, A_l as the area underneath the EXIT function of the lower code and R is the code rate of the parallel

concatenated scheme. Table 4.2 shows the use of the area properties of the EXIT chart in the calculation of attainable capacity for a PCC (i.e. a CTC) at their converging E_b/N_0 values. In table 4.2, the various attainable capacities have been used to compute the distance of a parallel concatenated code such as the CTC to capacity in bits per channel use at their converging E_b/N_0 values. This computation was made in four different modulation schemes. C_{DCMC} in table 4.2 is the Discrete-input Continuous-output Memoryless Channel capacity, while L is the capacity loss in bits per channel use (i.e. $C_{DCMC} - S$). The attainable capacity for the BPSK and 16QAM code rate 1/2 in table 4.2 is slightly larger than the DCMC capacity. This is due to some inconsistency in the EXIT charts of short frame size as explained in [46].

Table 4.2: EXIT chart and area properties of the UMTS Convolutional Turbo Code

Modulation Scheme	Code rate	E_b/N_0 (dB)	S	C_{DCMC}	C_A	L	%Δ
BPSK	1/3	0.400	0.329	0.394	0.392	0.065	16.497
BPSK	1/2	1.000	0.490	0.570	0.581	0.080	14.035
QPSK	1/3	0.700	0.665	0.820	0.731	0.155	18.902
QPSK	1/2	1.100	0.997	1.157	1.156	0.160	13.829
16QAM	1/3	2.500	1.328	1.700	1.297	0.372	21.882
16QAM	1/2	3.400	1.993	2.313	2.675	0.320	13.834
64QAM	1/3	5.400	1.996	2.840	2.605	0.844	29.718
64QAM	1/2	7.500	2.960	3.900	3.507	0.940	24.103

* S, C_A, L and C_{DCMC} in the table stands for the throughput value, attainable capacity, capacity loss and Discrete-input Continuous-output Memoryless Channel capacity, respectively. The unit of all these variables is bits per channel use. %Δ is the percentage loss in capacity with respect to C_{DCMC} .

Table 4.2 shows the distance from capacity in bits per channel use for each modulation and code rate for a CTC in HSDPA for an AWGN channel. Table 4.2 shows that the 64QAM modulation scheme (code rate 1/3 and 1/2) is the furthest from the capacity bound in bits per channel use.

4.6 Block Turbo Code

In this section, the BER versus E_b/N_0 performance of the Block Turbo Code (BTC) is examined. In the previous section, the EXIT chart technique was examined with its area properties applied in determining the capacity of a parallel concatenated code such as a CTC. The EXIT chart of serially concatenated schemes such as the BTC is examined in this section, along with its area properties in determining the capacity loss of a serially concatenated code. The performance of the BTC in terms of throughput in bits per channel use is also calculated. In Chapter 6, the performance of the BTC evaluated in this section is compared with the performance of the new Irregular Block Turbo Code (I-BTC) introduced by this research. The I-BTC which will be explained in detail in Chapter 6 is also a high speed channel codec like the BTC.

The Block Turbo Code also known as a Turbo Product Code is basically the iterative decoding of two block codes serially concatenated. Unlike the CTC which is known for its good performance at low code rates (i.e. 1/3, 1/2), the BTC (TPC) is well known for its good performance at high rates, making it a good channel codec for high speed communication systems. The BTC (or TPC) is also a very flexible channel codec that requires no puncturing in attaining a high rate, unlike the CTC which requires puncturing thereby making it suboptimal in terms of BER curve performance. In the next subsection, the BER versus E_b/N_0 performance of the BTC is analysed in an AWGN channel at different code rates and in four modulation schemes. Also, the EXIT chart properties of a BTC together with its area properties are examined.

4.6.1 BER performance of the Block Turbo Code

To computer the BER versus E_b/N_0 performance by MATLAB simulation, random information bits were generated. These bits were then encoded horizontally with a BCH encoder. Various BCH encoders corresponding to various BCH block sizes were used to encode the randomly generated information bits. Following the horizontal encoding of information bits, is the block interleaving of the horizontally encoded bits. The next step is the encoding of the horizontally encoded bits with the same BCH encoder. This encoding can be seen as vertical encoding since the bits have been interleaved with a block encoder. In a block interleaver, the rows become columns while the columns become the rows. The vertically encoded block codes are then sent over a Gaussian noisy channel. At the receiver, the received bits are then passed into the iterative decoders. The BTC is a serially concatenated code, as such the iterative decoder consists of two decoders in series. The first is the inner decoder which is directly connected to the channel, while the other is the outer decoder. The Log-MAP version of the BCJR which is explained later in Chapter 6 has been used in this thesis as the soft input soft output decoder originally proposed in [19] and subsequently used in [18]. The process of iterative decoding in a serially concatenated code is the same as the parallel concatenated code explained earlier with few exceptions. The first is that the decoding process is between two serially concatenated decoders as depicted in figure 4.4.9. The second is that, the *a priori LLR* values of the data bits are summed up together with the received channel information of the data bits which is passed from one decoder to the other. The BER is computed by dividing the total number of bits in error by the total number of randomly generated information bits. The nomenclature of the BTC is given by (n, k)BTC, where n and k denote the block length of the codeword and information bits, respectively. In the BTCs the encoding of information bits was done twice i.e. horizontal and vertical encoding, as required for a BTC. Figures 4.5.1 to 4.5.4 illustrate the system BER versus E_b/N_0 performance of different binary BTCs in an AWGN channel with BPSK, QPSK, 16QAM and 64QAM modulations schemes. The BTC rates are 0.54, 0.70, 0.82 and 0.89 for the (15, 11)BTC, (31, 26)BTC, (63, 57)BTC and (127, 120) BTC, respectively. An important feature in these BER curves is the absence of

early error floors. The Log-MAP version of the BCJR (as explained later in Chapter 6) has been used as the soft input soft output decoder as originally proposed in [19] and subsequently used in [18]. A single iterative decoding is equivalent to a horizontal and a vertical decoding. This implies that a single decoding requires two Log-MAP decoding instances. The BER curves shown in figures 4.5.1 to 4.5.4 are same as the results established in previous published study in [18]. The analysis of the EXIT chart for a binary BTC together with the use of the EXIT chart in the computation of the system's capacity in the next sub-section is a contribution in this thesis.

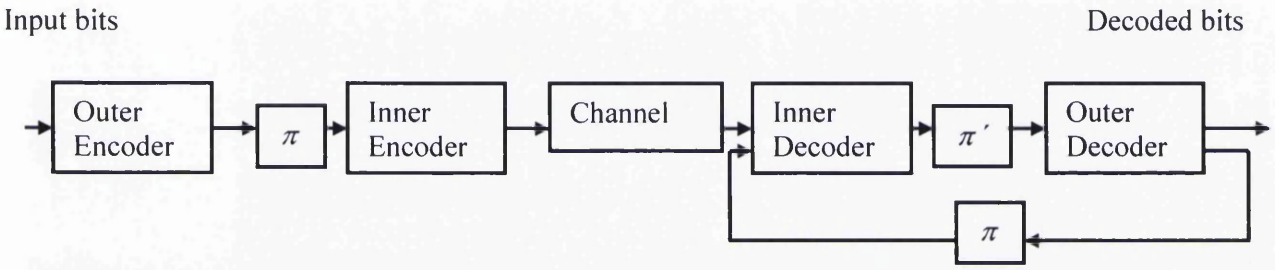


Figure 4.4.9 Schematic of a serially concatenated code

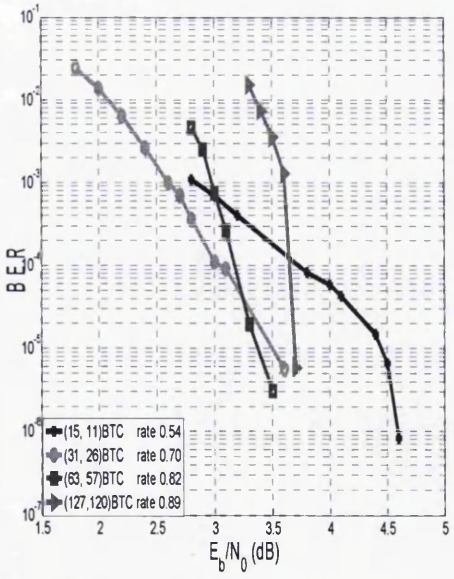


Figure 4.5.1 BER performance for binary BTCs in BPSK.

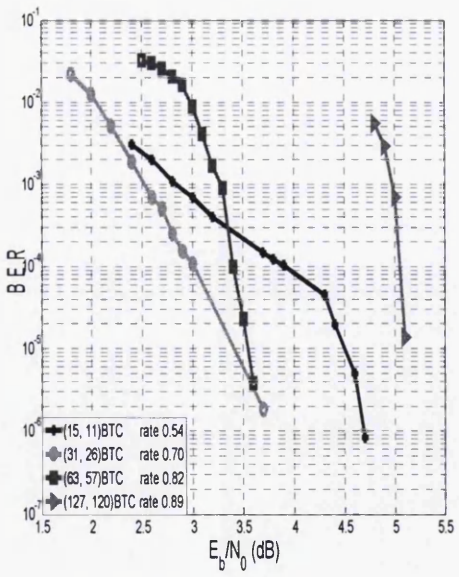


Figure 4.5.2 BER performance for binary BTCs in QPSK.

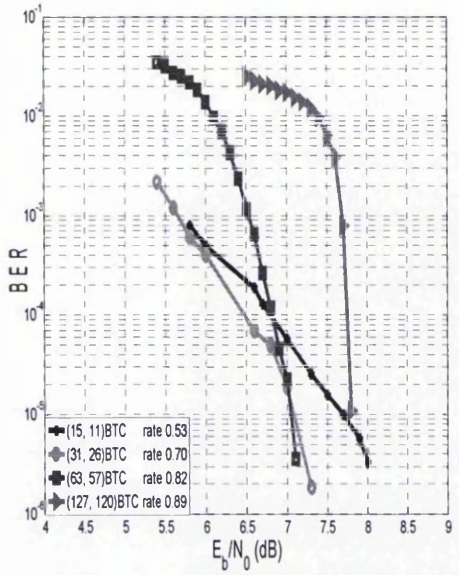


Figure 4.5.3 BER performance for binary BTCs in 16QAM.

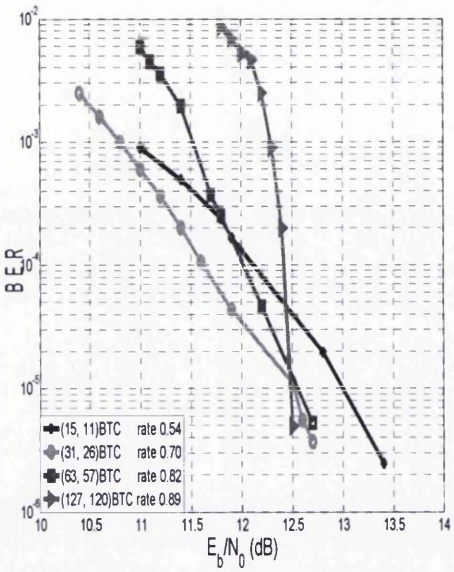


Figure 4.5.4 BER performance for binary BTCs in 64QAM.

Table 4.3 shows the performance of the BTC tabulated together with their corresponding throughput per channel use which is given by $S = R \times (\log_2 M)(1 - BLER)$, where R is the rate of the code, $BLER$ the block error rate and M the M-ary order of the modulation scheme. In this thesis, a block in the BTC is the size of the original information bit (i.e. 11 x 11 and 26 x 26 for the case of the (15, 11)BTC and the (31, 26)BTC, respectively).

Table 4.3: Various BTCs with different modulation schemes showing their code rates, converging E_b/N_0 values, throughput and the required number of iterations

Modulation Scheme	Code & Code rate	E_b/N_0 (dB)	S	Φ
BPSK	(15, 11) 0.54	4.48	0.54	4
BPSK	(31, 26) 0.70	3.50	0.70	5
BPSK	(63, 57) 0.82	3.60	0.81	3
BPSK	(127, 120) 0.89	3.69	0.88	5
QPSK	(15, 11) 0.54	4.50	1.08	2
QPSK	(31, 26) 0.70	3.40	1.40	9
QPSK	(63, 57) 0.82	3.55	1.63	5
QPSK	(127, 120) 0.89	5.10	1.76	10
16QAM	(15, 11) 0.54	7.70	2.16	6
16QAM	(31, 26) 0.70	7.10	2.80	8
16QAM	(63, 57) 0.82	7.05	3.26	3
16QAM	(127, 120) 0.89	7.80	3.51	10
64QAM	(15, 11) 0.54	13.00	3.24	2
64QAM	(31, 26) 0.70	12.40	4.19	8
64QAM	(63, 57) 0.82	12.55	4.91	2
64QAM	(127, 120) 0.89	12.50	5.26	8

* S and Φ in the table stand for the throughput value in bits per channel use and the number of iterations required to achieve a BER of 10^{-5}, respectively.

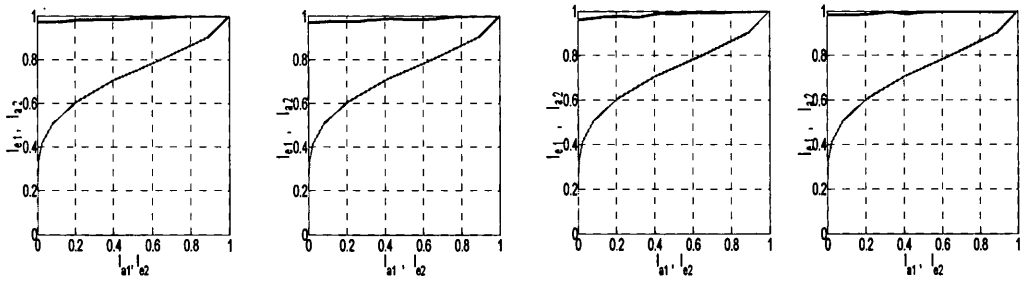
4.6.2 EXIT charts of the Binary Block Turbo Code and its area properties

In this subsection, the EXIT chart of various BTCs and their area properties are examined. Figures 4.6.1 (a) to (d) show the EXIT charts of the (15, 11)BTC for BPSK, QPSK, 16QAM and 64QAM modulation schemes at their converging E_b/N_0. The EXIT charts shown all have an open tunnel gap until the point (1, 1). This shows that a very low bit error rate during bit transmission at these E_b/N_0 values will occur. The same open tunnel gap applies to the EXIT charts of the (31, 26)BTC, (63, 57)BTC and (127, 120) BTC with the BPSK, QPSK, 16QAM and 64QAM modulation schemes, as illustrated in figures 4.6.1 (e) to (p). The area A_{in} underneath the EXIT function of an inner code and the corresponding attainable capacity as explained in section 4.4.3.3 for a serially concatenated code such as a BTC, which is similar to the case of the parallel concatenated code, is given by (4.28).

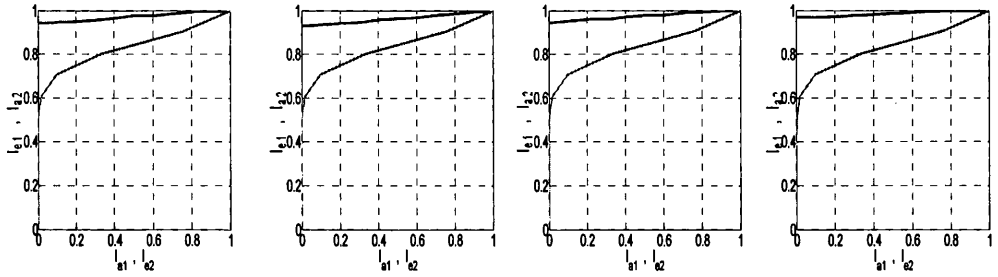
$$A_{in} = \int_0^1 I_e(I_a) dI_a = \frac{I(X,Y)}{R_{in}} \tag{4.28}$$

$$C_A = A_{in} \times R_{in} \times \log_2 M \tag{4.29}$$

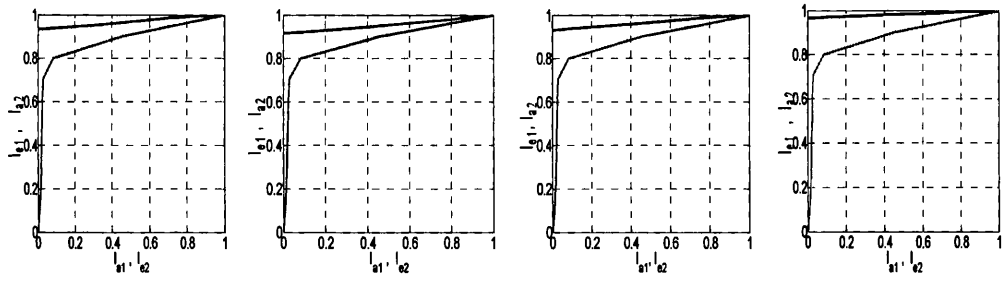
where M is the M-ary modulation order and R_{in} is the code rate of the inner code.



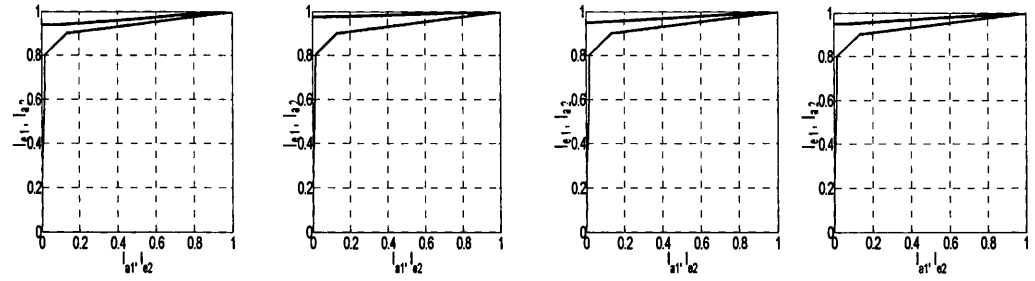
(a) (15, 11), 4.5dB, BPSK (b) (15, 11), 4.5dB, QPSK (c) (15, 11), 7.7dB, 16QAM (d) (15, 11), 13.0dB, 64QAM



(e) (31, 26), 3.5dB, BPSK (f) (31, 26), 3.4dB, QPSK (g) (31,26), 7.1dB, 16QAM (h) (31,26), 12.4dB, 64QAM



(i) (63,57), 3.55dB, BPSK (j) (63,57), 3.55dB, QPSK (k) (63,57), 7.05dB, 16QAM (l) (63,57),12.55dB, 64QAM



(m) (127,120),3.69dB,BPSK (n) (127,120),5.1dB,QPSK (o)(127,120),07.7dB,16QAM (p)(127, 120),12.48dB,64QAM

Figures 4.6.1 EXIT charts for various BTCs using BCH codes.
Legend: 'Code', 'E_b/N_0 of convergence', 'modulation'.
The figures above are labeled in line with the legend, i.e. the code, followed by the E_b/N_0 value for convergence and then the modulation scheme. The y and x axis of the EXIT charts have been labeled (I_{e1}, I_{a2}) and (I_{a1}, I_{e2}), respectively, for clarity.

Table 4.4 shows the use of the area properties of a BTC in calculating the attainable capacity of the various BTCs at their converging E_b/N_0 values with different modulation schemes. Table 4.4 also shows the capacity loss for the various BTCs in bits per channel use, given by, $L = C_{DCMC} - S$.

Table 4.4: EXIT chart and area properties of various binary BTCs at their converging E_b/N_0 values

Modulation Scheme	Code & Code rate	E_b/N_0 (dB)	S	C_{DCMC}	C_A	L	%Δ
BPSK	(15, 11) 0.54	4.50	0.54	0.85	0.73	0.31	36.47
BPSK	(31, 26) 0.70	3.50	0.70	0.86	0.82	0.16	18.60
BPSK	(63, 57) 0.82	3.55	0.81	0.90	0.88	0.09	10.00
BPSK	(127, 120) 0.89	3.69	0.88	0.92	0.91	0.04	4.35
QPSK	(15, 11) 0.54	4.50	1.08	1.70	1.45	0.62	36.47
QPSK	(31, 26) 0.70	3.40	1.40	1.70	1.62	0.30	17.65
QPSK	(63, 57) 0.82	3.55	1.63	1.80	1.73	0.17	9.44
QPSK	(127, 120) 0.89	5.10	1.76	1.93	1.87	0.17	8.81
16QAM	(15, 11) 0.54	7.70	2.16	3.40	2.90	1.24	36.47
16QAM	(31, 26) 0.70	7.10	2.80	3.50	3.27	0.70	20.00
16QAM	(63, 57) 0.82	7.05	3.26	3.62	3.50	0.36	9.94
16QAM	(127, 120) 0.89	7.78	3.51	3.78	3.68	0.27	7.14
64QAM	(15, 11) 0.54	13.00	3.24	5.48	4.37	2.24	40.88
64QAM	(31, 26) 0.70	12.40	4.19	5.61	4.96	1.42	25.31
64QAM	(63, 57) 0.82	12.55	4.90	5.70	5.34	0.80	14.04
64QAM	(127, 120) 0.89	12.48	5.26	5.80	5.52	0.54	9.31

* S, C_A, L and C_{DCMC} in the table stands for the throughput value, attainable capacity, capacity loss and Discrete-input Continuous-output Memoryless Channel capacity, respectively. The unit of all these variables is bits per channel use. %Δ is the percentage loss in capacity with respect to C_{DCMC}.

Table 4.4 illustrates the distance from capacity for each modulation and BTC in an AWGN channel. It can be deduced from table 4.4 that higher rate codes are closer to their C_{DCMC} than the lower code rates in all the modulation schemes. This means that high code rate binary BTCs using BCH codes as its constituent code are more efficient spectrally and in terms of energy consumption.

4.7 Reed Solomon Block Turbo Code

Non-binary Block Turbo Codes have gained importance over time in high speed optical transmission and data storage systems. Non-binary codes such as Reed Solomon (RS) codes are very good in the correction of burst errors in communication systems making them useful error correcting codes in non-binary Block Turbo Codes.

The construction of a RS BTC is similar to the binary BCH BTC with the exception that the code is operational in a Galois field m for various RS BTC sizes. In other words, each non-binary symbol consists of m bits [48]. This can also be interpreted as having $(K_1 \times K_2 \times q)$ information bits, where q corresponds to the Q-ary of the non-binary symbol. K_1 and K_2 are illustrated in figure 2.2 in Chapter 2, showing the construction of a two dimensional Block Turbo Code.
In this section, the system BER performance of various RS codes as the inner and outer codes in a Block Turbo Code scheme is examined in different modulation schemes in an AWGN channel.

4.7.1 Iterative decoding of the Reed Solomon Block Turbo Code

The Chase-Pyndiah decoding algorithm has been used as our soft input soft output decoder, which consists of a soft-input hard-output (SIHO) decoder combined with a

soft-output computation unit as explained and implemented in [11]. In this study, the decoding was performed at the bit level.

Let $r = (r_{1,1}, r_{1,2}, ..., r_{n,q})$ be the received soft information sequence from the channel corresponding to a row or a column, with q representing the number of bits per symbol and n the length of the codeword in symbols. A hard decision Y' is calculated using the sign of the received soft information value for each received bit.

$$Y' = (y_{1,1}, y_{1,2}, ..., y_{n,q}) \qquad (4.30)$$

and

$$y_{i,j} = \begin{cases} +1 & if \quad L(r_{i,j}) > 0 \\ -1 & if \quad L(r_{i,j}) \le 0 \end{cases} \qquad (4.31)$$

The reliability of the decision on the $j-th$ bit in the $i-th$ RS symbol is measured by the magnitude $|r_{i,j}|$ of the corresponding soft input. 2^p error patterns (test patterns) are then generated by considering all possible combinations of 0 and 1 in the p least reliable bit positions. The error patterns are then added to the hard decision Y' to form candidate sequences. Algebraic decoding (such as the Berlekamp Massey decoding algorithm) of each candidate sequence is implemented thereby producing a list containing at most 2^p distinct candidate codewords C. A decision criterion is then used to select one of the codewords as the final hard decision (SIHO) which is the first process in the Chase-Pyndiah decoding algorithm. The decision criterion is given as the candidate codeword C^d at minimum Euclidean distance from the received r sequence given by

$$D = C^d \quad \text{if} \quad |r - C^d| < |r - C^e| \quad \forall \quad d \neq e \qquad (4.32)$$

where D is the selected candidate codeword.

The next step is to compute the soft output from the selected candidate code word D. For a given bit j in the $i-th$ RS symbol in D, the list of candidate codewords is searched for a competing codeword c at minimum Euclidean distance from r such that $c_{i,j} \neq d_{i,j}$ where $d_{i,j}$ is the hard decision for each bit of the selected candidate codeword

D and $c_{i,j}$ the elements of c. If such a codeword exists, the soft output $\hat{r}_{i,j}$ on this bit is given by

$$\hat{r}_{i,j} = \left(\frac{\| r_{i,j} - c_{i,j} \|^2 - \| r_{i,j} - d_{i,j} \|^2}{4} \right) \times d_{i,j} \tag{4.33}$$

where $\| \ \|^2$ denotes the squared norm. Otherwise the soft output is computed as

$$\hat{r}_{i,j} = r_{i,j} + \beta . d_{i,j} \tag{4.34}$$

where $r_{i,j}$ is the received soft information from the channel with β, a fractional predefined positive constant ranging from 0 to 1, that increases as the iteration increases. The values of β used in this study ranges from 0.2 to 0.65 at an interval of 0.1.

In the decoding of the Reed Solomon Block Turbo Code, a horizontal decoding, consisting of a single "Chase-Pyndiah" decoding, is a half iteration, while a horizontal and a vertical decoding, consisting of two "Chase-Pyndiah" decodings is a full iteration. Each element of the β value is used for a half iteration. This implies that a full iteration requires two β values.

The next step is to extract the extrinsic information $w_{i,j}$ which is computed using equation (4.35)

$$w_{i,j} = \hat{r}_{i,j} - r_{i,j} \tag{4.35}$$

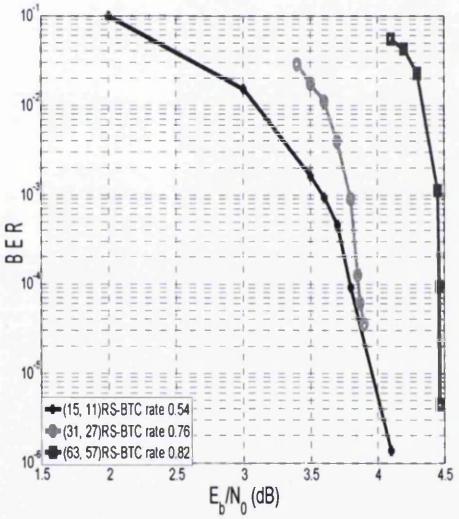
Let v denote a half iteration (i.e. decoding of either a row or column alone), then the SISO decoder's input at half iteration v is given by (4.36),

$$R(v) = R + \alpha(v) W(v) \tag{4.36}$$

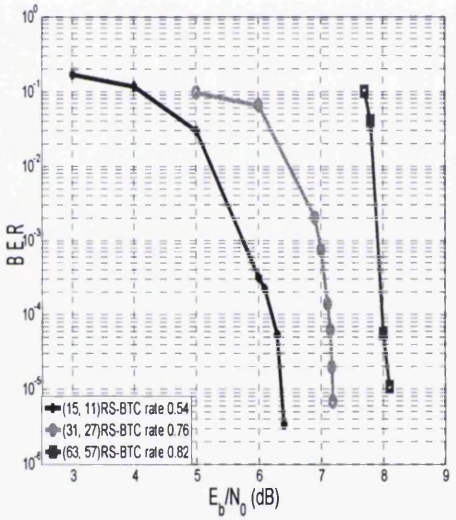
where R is the received soft input going into the decoder and α is a scaling factor used to reduce the influence of extrinsic information delivered by the SISO decoder at the previous half iteration. The values of α used in this study increases with the number of iterations from 0 to 0.7 at an interval of 0.1.

4.7.2 Bit error rate performance of the Reed Solomon Block Turbo Code

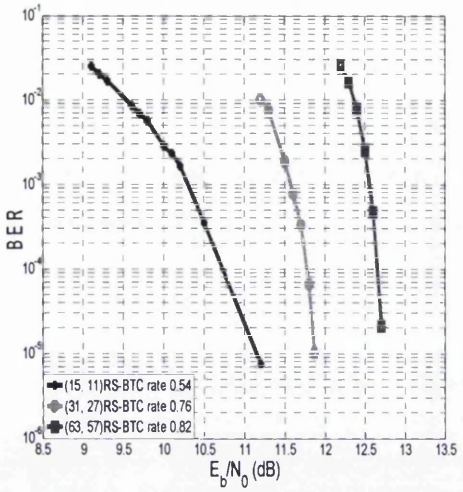
The Chase-Pyndiah decoding algorithm has been used as our soft input soft output decoder with the use of four least reliable bits due to its moderate complexity as explained in [49]. Figures 4.7.1 to 4.7.3 show the BER performance of 3 different Reed Solomon Block Turbo Codes in a QPSK, 16QAM and 64QAM modulation scheme in AWGN. The BER performance in the BPSK and the QPSK modulation schemes are the same.



Figures 4.7.1 BER performance for Reed Solomon BTC in BPSK/QPSK.



Figures 4.7.2 BER performance for Reed Solomon BTC in 16QAM.



Figures 4.7.3 BER performance for Reed Solomon BTC in 64QAM.

The BER curve for the Reed Solomon BTCs shown in figures 4.71 to 4.7.3 have a pattern where the lower rate code converge to a low BER at lower E_b/N_0 values in the modulation schemes. This is so because, the lower code rate Reed Solomon BTCs have lower bits per symbol in comparison to the higher code rate Reed Solomon BTCs. Consequently, each block in the lower code rate Reed Solomon BTCs has a lower number of bits per block. In section 4.7.1 above, a detailed explanation of the "Chase-Pyndiah" algorithm was given with p least reliable bits. In the BER curve for the Reed Solomon BTCs, four least reliable bits were fixed, which means, the lower rate Reed Solomon BTCs have a greater chance of being correctly decoded at lower E_b/N_0 values unlike the higher rate Reed Solomon BTCs because the lower code rates have smaller block lengths. The result in Pyndiah's published work in [11] [12] [49] and [81] verifies this. However, in terms of distance to capacity, the Reed Solomon BTCs with higher code rates were closer to capacity in comparison to the Reed Solomon lower code rates. Table 4.5 shows the converging E_b/N_0 values for the various Reed Solomon BTCs with different modulation schemes, as well as the number of iterations required to converge to a low BER. Also on this table is the throughput value of the various BTCs in comparison to their C_{DCMC} and the capacity loss L in bits per channel use. A similar trend in terms of distance to capacity in bits per channel use seen in table 4.4 is also

4.8 Conclusions

In this Chapter, the bit error rate performance of the Convolutional Turbo Code (CTC) and the Block Turbo Code (BTC) is considered in a Gaussian noisy channel. More specifically, the CTC's BER performance was evaluated using an HSDPA system model with a Gaussian and a 3GPP multipath Vehicular A channel. These BER performances were evaluated in four different modulation schemes; BPSK, QPSK, 16QAM and 64QAM. Additionally, a characterisation of the HSDPA capacity using the EXIT chart technique was given. It was seen in this Chapter, that the EXIT charts of the CTCs at their converging E_b/N_0 values all had open tunnel gaps. Furthermore, it can be deduced that the CTCs code rate 1/3 were further away from capacity in comparison with the CTC code rate 1/2 with the same modulation scheme percentage wise with respect to capacity, making the code rate 1/2 CTC a better choice in terms of spectral efficiency. It was also established in this Chapter that, the sum of areas underneath the EXIT functions of the upper and lower decoders of a CTC sums up to the capacity of the channel.

The BER versus E_b/N_0 performance of various binary BTCs was examined, where the Log-MAP trellis decoding was used in the decoding of the various BTCs. The (31, 26)BTC code rate 0.70 recorded coding gains over the other BTC variants in all the modulation schemes except for the 16QAM modulation scheme where the (63, 57)BTC code rate 0.82 had a coding gain of 0.05 dB over the (31, 26)BTC. This highlights the decoding power of the (31, 26)BTC among the variants shown. Also the EXIT charts of binary BTCs were plotted with the area properties used in determining channel capacity in bits per channel use. The distance between the throughput (in bits per channel use) of the channel codec and the channel capacity were also determined. In terms of distance to capacity, the higher code rate binary BTCs were closer to capacity than the lower code rate binary BTCs for the various modulation schemes. The (127, 120) binary BTC

for the BPSK modulation scheme was the closest to channel capacity at a distance of 0.04 bits per channel use among the BTCs.

The BER performance of the Reed Solomon BTC was also examined in this Chapter as a channel codec in a Gaussian noisy channel. The Reed Solomon Block Turbo Code BER curves had a unique characteristic pattern for the 3 variants of the BTC in that, (15, 11) Reed Solomon BTC converged at a lower E_b/N_0 in comparison to the (31, 27) Reed Solomon BTC, while the (31, 27) Reed Solomon BTC converged at a lower E_b/N_0 value in comparison to the (63, 57) Reed Solomon BTC in all the investigated modulation schemes. As earlier explained, this happens because the same number of least reliable bits was fixed at 4 for all the various Reed Solomon BTCs, with the lower code rates having lower block lengths in comparison to the higher code rates. However, in terms of distance to capacity, the Reed Solomon BTCs with higher code rates were closer to capacity in comparison to the Reed Solomon lower code rates. This establishes the fact that higher code rates are more efficient in terms of bandwidth usage than lower code rates because, more bits per channel use can be transmitted with higher code rate (at rates closer to capacity) than lower code rates which are further away from capacity. The EXIT chart has also been used in this Chapter to establish the convergence of the binary BTCs and the CTC and hence assist in quantifying the codec performances. For systems where signal power conservation is of utmost importance, such as in digital satellite transmission, a high rate code such as the BTC would be efficient to use. In the next Chapter, the application of irregular coding to the CTC is examined with a view of achieving a near capacity operation as well as reduction in the system's complexity.

Chapter 5

BER performance of the Irregular Turbo Code

5.1 Introduction

The introduction of iterative decoding (Turbo codes) by Berrou et. al. in [7] has significantly reduced the transmit power to achieve negligible bit error rate (BER) with moderate to large complexity in digital wireless communication systems. In [7], Turbo Codes (TC) with a large frame size (2^{16} bits) in an AWGN channel at code rate 1/2 have been shown to converge to a low probability of error at an SNR of 0.7 dB from the Shannon limit. Short frame size (2^{12} bits) TC at the same code rate has also been shown to be at about 1.4 dB from the Shannon limit in an AWGN channel [50]. In [79], Valenti used various frame sizes ranging from 40 bits to 5114 bits in a Universal Mobile Telecommunication System (UMTS) TC and showed that the larger the frame size in a TC, the closer the TC gets to the Shannon bound. In general, these results show that performance close to the Shannon bound requires large frame sizes thereby increasing complexity and latency in the system.

The unequal protection of information bits has recently been of interest, from the design of an irregular LDPC codes to the design of an Irregular Turbo Code (I-TC). Significant

performance benefits stem from the extra protection on some of the bits in the frame. An I-TC was first proposed in [38], where a coding gain of 0.23 dB at a BER of 10^{-4} was demonstrated over the corresponding regular TC in an AWGN channel using BPSK modulation and a large frame size (2^{17} bits). In [51], an I-TC with a frame size of 10^4 bits gave a coding gain of about 0.1 dB at a BER of 10^{-5} over the equivalent regular TC for an AWGN channel and 8PSK modulation. Also, in [40] an I-TC of large frame size (2^{17} bits) achieved a coding gain of 0.24 dB at a BER of 10^{-6} in comparison to the regular TC in AWGN with a BPSK modulation scheme. The authors in [51] and [40] have used the same technique to develop their I-TC but have deployed it in different modulation schemes (8PSK and BPSK) leading to different coding gains. This shows that an I-TC with large frame size can achieve a higher coding gain over the regular TC using the same frame size.

In this Chapter, the performance of the I-TC in comparison to the regular TC is examined using different frame sizes and different modulation schemes. Also in this Chapter, a new I-TC is developed, capable of achieving a coding gain of 1.0 dB over the regular TC in an AWGN channel using the HSDPA system model with short frame sizes. This result was achieved using a 64QAM modulation scheme. This shows that large coding gains can be achieved in an I-TC in comparison to the regular TC using short frame sizes. Furthermore, a coding gain of 1.4 dB over the regular TC was achieved in a 3GPP multipath channel using a 16QAM modulation scheme. At the time of writing this report, there are no results for an I-TC with higher order modulations. These results were achieved using one quarter less iterations when compared to other I-TCs.

5.2 Design and construction of an Irregular Turbo Code

A classical parallel TC has two recursive systematic Convolutional (RSC) components, separated by an interleaver [1] with 3 branches: the systematic bits, a first RSC (upper) component and a second RSC (lower) component. Figure 5.1(a) from depicts an equivalent structure of a classical TC which has only one RSC component, where all the information bits have been repeated twice (without puncturing) [40]. **K**, **P** represent the information and parity vectors, respectively, while π denotes interleaving.

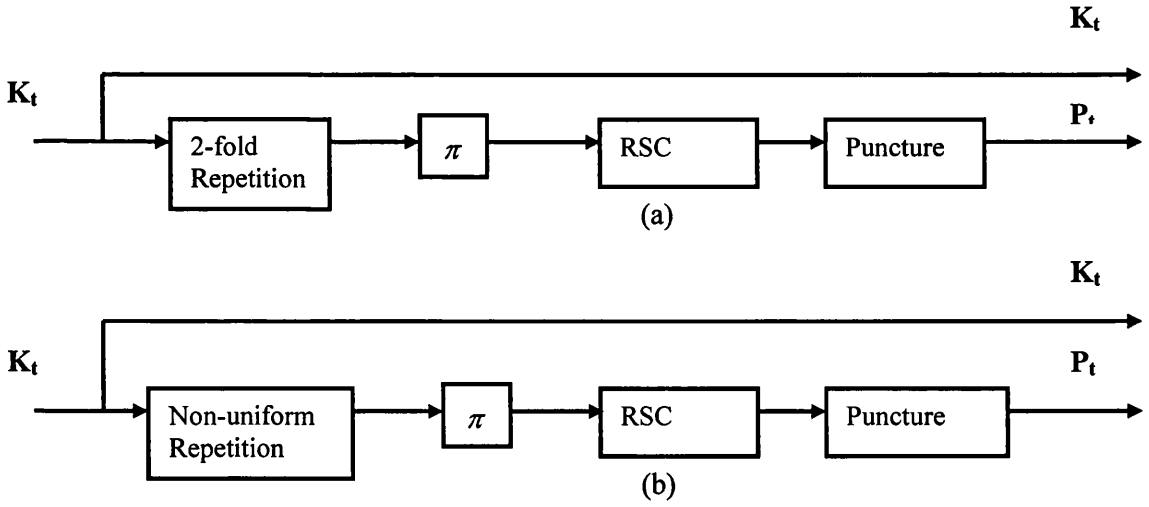


Figure 5.1(a) Equivalent encoding structures for a regular Turbo Code. 5.1(b) Encoding structure for an Irregular Turbo Code. Figures 5.1(a) and (b) is as found in [40].

The performance of the TC in [7] and the equivalent structure in figure 5.1(a) have been shown to be the same with slight differences [51]. To design an irregular TC, the equivalent TC structure is used, where a small fraction of the information bits is repeated d times (called the degree) where $d > 2$, e.g., $d = 4$. The bits of higher degree are well protected because their *a posteriori* values include 4 extrinsic information bits instead of 2, for example. The general structure of an irregular TC is shown in figure 5.1(b). The

non-uniform repetition divides the information bits into groups indexed by $i = 2,3,....$ with each group having a certain number of repetitions d_i where $d_i = 2,3,.....T$, and T is the maximum number of repetitions [51]. The number of bits in a group i gives a fraction f_i of the total number of information bits at the turbo encoder input. The output of the repeater is then randomly interleaved (a random vector interleaver) and passed to the RSC component of the I-TC. The parity bits from the RSC constituent and the original information bits before the non-uniform repetitions are then transmitted. The parity bits are punctured so as to target certain code rates. In designing the I-TC in this thesis, 4 different puncturing patterns were empirically developed. The puncturing pattern 101101110 (the 2nd, 5th and 9th parity bits are punctured for every 9 parity bits), 10110 (the 2nd and 5th parity bits are punctured for every 5 parity bits), 11110 (the 5th parity bit is punctured for every 5 parity bits) and the 10 (the 2nd parity bit is punctured for every 2 parity bits) have been empirically found to be effective. The following equations show the relationship between the irregular TC rate, the puncturing pattern and the various non-uniform repetition degrees.

$$\sum_{i=2}^{T} f_i = 1 \qquad (5.1)$$

$$\sum_{i=2}^{T} i.f_i = \bar{d} \qquad \bar{d} \text{ is the average bit degree} \qquad (5.2)$$

$$R = \frac{1}{1+\bar{d}\left(\frac{1}{\theta}-1\right)} \quad \text{where} \quad \theta = \frac{1}{(2-f_\theta)} \qquad (5.3)$$

In (5.3), f_θ is the fraction of the parity bits that has been punctured.

The decoding of the I-TC requires only one Log-MAP (the modified BCJR algorithm described in Chapter 4) soft-input soft-output (SISO) decoder. Firstly, the received information bits $\mathbf{K_r}$ are repeated (as at the transmitter) then randomly interleaved using the same interleave pattern to give $\mathbf{K'_r}$ and then sent into the SISO decoder together

with the received parity bits $\mathbf{P_r}$ and an initial *a priori* value **A** of equal probability (i.e. zero log likelihood). Secondly, the extrinsic information bits **E** at the output of the SISO decoder are computed by an extrinsic computation block in order to derive new extrinsic information for each information bit. The extrinsic information bits are de-interleaved before being transferred into the extrinsic computation block. Inside the extrinsic computational block, each information bit with degree d_i will then have a new extrinsic information value which is the product (sum when using log likelihood) of the other $d_i - 1$ extrinsic information values. At the output of the extrinsic information block, the new extrinsic values are then interleaved to give the new *a priori* values for the next decoding iteration. The extrinsic computational block is also used in the decoding of an Irregular Block Turbo Code as will be shown later in Chapter 6 of this thesis.

Figure 5.2 depicts the decoding process for an Irregular Turbo Code where π and π' denote the interleaving and deinterleaving functions, respectively. The block S in the schematic diagram is used to select the decoded bits in the format of the originally generated bits to give $\mathbf{K_d}$ (i.e. to remove the non-uniform repetition) for final comparison with the originally generated bits $\mathbf{K_t}$.

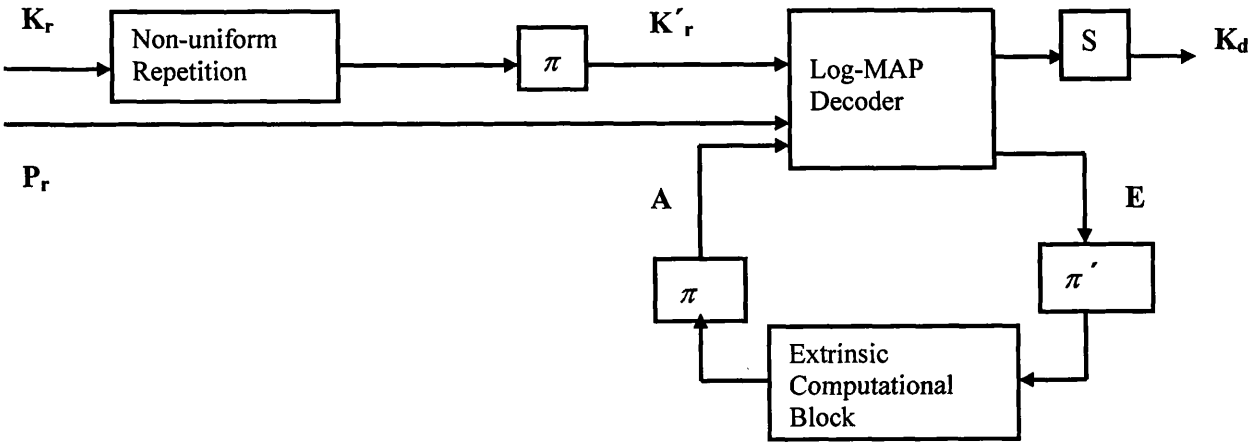


Figure 5.2 Iterative decoding structure of an Irregular Turbo Code from [40].

In the extrinsic computational block depicted in figure 5.2, the extrinsic information at every iteration for the k^{th} bit E_{jk} with a degree of repetition of d_i is recalculated in the log domain using equation (5.4).

$$E_{jk} = \sum_{\substack{l=1 \\ l \neq k}}^{d_i} E_{jl} \qquad (5.4)$$

5.2.1 System parameters

The 3GPP W-CDMA system for the HSDPA model described in sections 4.1 and 4.2 was used for the simulation frame work. Walsh codes and Gold sequences were used as the spreading code and scrambling code in the HSDPA system, respectively. A random Interleaver with different frame sizes ranging from 1003 to 20072 bits was also used. The simulations were performed in a Gaussian and in the 3GPP Vehicular A multipath channel with four different modulation schemes: BPSK, QPSK, 16QAM and 64QAM.

5.3 Bit error rate and throughput performance of Irregular Turbo Code in AWGN.

In this section, the BER performance of the I-TC is examined using frame size of 1003, 5012, 10016 and 20072 bits with different modulation schemes in comparison to the regular TC with the same frame sizes. The simulations were done using MATLAB. The number of iterations required for the I-TC and the regular TC to converge to a low BER was also recorded. Also in this section, the throughput curves for the I-TC and the regular TC have been plotted. Figures 5.3 to 5.6 show the BER performance of an I-TC in comparison to the regular TC in AWGN at code rate 1/3 (with the exception of the QPSK I-TCs at code rate 0.40) with different modulation schemes. Table 5.1 shows the

puncturing patterns in the I-TCs, the number of iterations required to converge to a BER of 10^{-5} in the I-TCs and TCs, the required E_b/N_0 value for convergence in the I-TCs and TCs, as well as the corresponding degree profiles used to achieve this for the I-TCs.

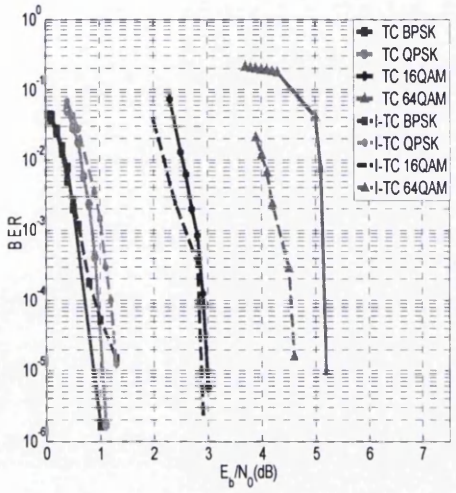


Figure 5.3 BER performance for I-TC and TC in an AWGN channel with a frame size of 1003 bits in four different modulation schemes, code rate 1/3.

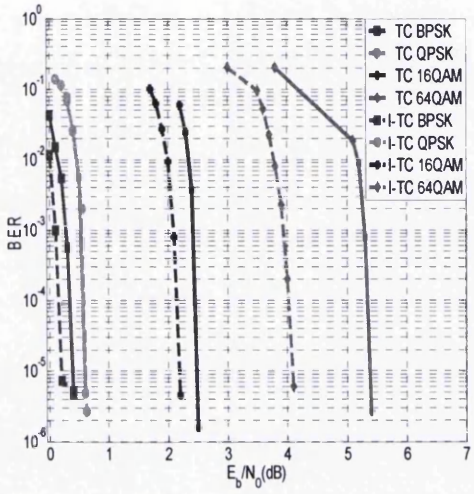


Figure 5.4 BER performance for I-TC and TC in an AWGN channel with a frame size of 5012 bits in four different modulation schemes, code rate 1/3.

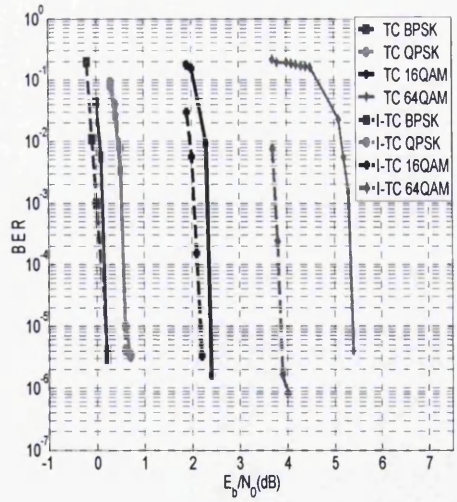


Figure 5.5 BER performance for I-TC and TC in an AWGN channel with a frame size of 10016 bits in four different modulation schemes, code rate 1/3.

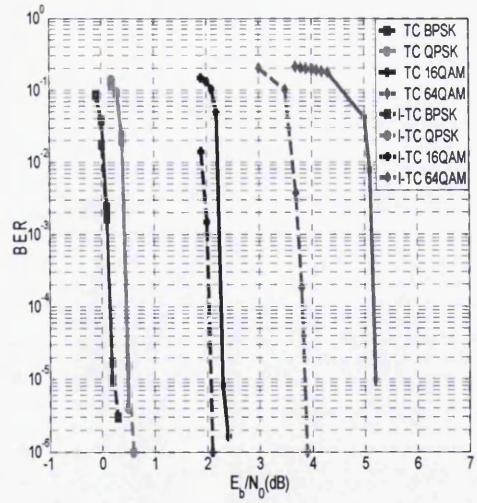


Figure 5.6 BER performance for I-TC and TC in an AWGN channel with a frame size of 20072 bits in four different modulation schemes, code rate 1/3.

The complexity of the I-TC in comparison to the TC is quantified by the number of iterations required to converge to a low bit error rate, as well as the number of decoding components required to do this. It should be noted that a single iteration in the irregular TC consists of only one Log-MAP decoder unlike in the regular TC where a single iteration consists of two Log-MAP decoders. Also previous I-TCs have used about 100 iterations to achieve a low probability of error [40].

Table 5.1: I-TC and TC at different frame sizes with their converging E_b/N_0 in AWGN

Frame Size	Code rate	Modulation Scheme	Converging E_b/N_0 (dB)		Gain (dB)	Number of iterations		Puncture pattern for I-TC	Bit degree d_i (fractions f_i)
			TC	I-TC		TC	I-TC		
1003	0.33	BPSK	0.88	1.30	-0.42	9	21	11101101110	2(0.888), 8(0.06),9(0.052)
	0.33 / 0.40	QPSK	1.04	1.30	-0.26	12	16	11101101110	2(0.96), 6(0.04)
	0.33	16QAM	3.00	2.85	0.15	14	19	Unpunctured	2(0.99), 7(0.01)
	0.33	64QAM	5.90	4.60	1.30	22	24	11101101110	2(0.85), 7(0.15)
5012	0.33	BPSK	0.40	0.20	0.20	11	31	11101101110	2(0.888), 8(0.06),9(0.052)
	0.33 / 0.40	QPSK	0.70	0.62	0.08	15	36	11101101110	2(0.96), 6(0.04)
	0.33	16QAM	2.50	2.20	0.30	14	19	Unpunctured	2(0.99), 7(0.01)
	0.33	64QAM	5.40	4.10	1.30	24	16	11101101110	2(0.85), 7(0.15)
10016	0.33	BPSK	0.19	0.17	0.02	11	26	11101101110	2(0.888), 8(0.06),9(0.052)
	0.33 / 0.40	QPSK	0.60	0.60	0.00	10	22	11101101110	2(0.96), 6(0.04)
	0.33	16QAM	2.30	2.10	0.20	18	17	Unpunctured	2(0.99), 7(0.01)
	0.33	64QAM	5.38	3.86	1.52	7	18	11101101110	2(0.85), 7(0.15)
20072	0.33	BPSK	0.19	0.17	0.02	11	18	11101101110	2(0.888), 8(0.06),9(0.052)
	0.33 / 0.40	QPSK	0.49	0.52	-0.03	10	30	11101101110	2(0.96), 6(0.04)
	0.33	16QAM	2.30	2.10	0.20	13	17	Unpunctured	2(0.99), 7(0.01)
	0.33	64QAM	5.20	3.85	1.35	13	16	11101101110	2(0.85), 7(0.15)

Figures 5.3 to 5.6 show that, the higher the modulation order the higher the coding gain of the I-TCs over the TCs. Also, table 5.1 shows that, the larger the frame size, the larger

the coding gains in the I-TCs and the TCs. This shows that larger frame size codes are closer to Shannon bound than shorter frame sizes as shown in [79].

The BER performance of the I-TC in comparison to the regular TC was then evaluated with four different modulation schemes using a short frame size (5012) for the HSDPA system at code rates 1/3 and 1/2 (code rates 0.40 and 0.41 in the case of QPSK) as shown in figures 5.7 to 5.10.

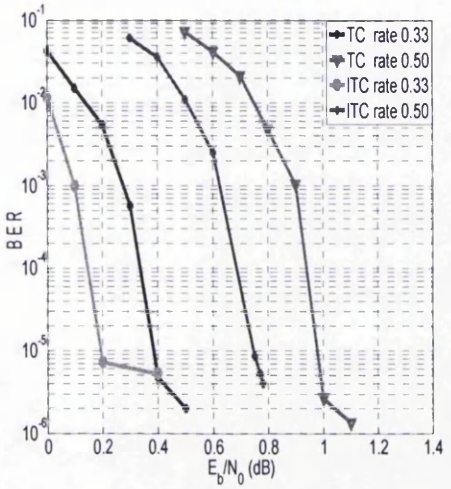


Figure 5.7 BER performance comparisons between TCs and I-TCs in an AWGN channel with BPSK.

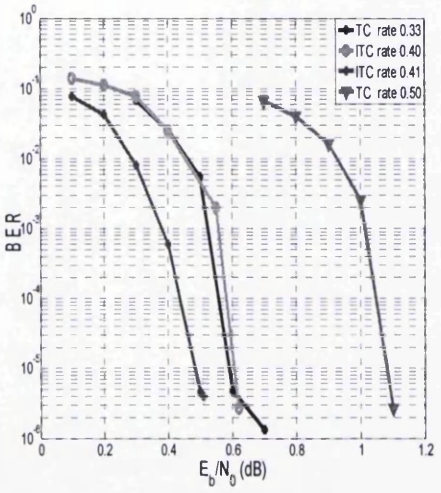


Figure 5.8 BER performance comparisons between TCs and I-TCs in an AWGN channel with QPSK.

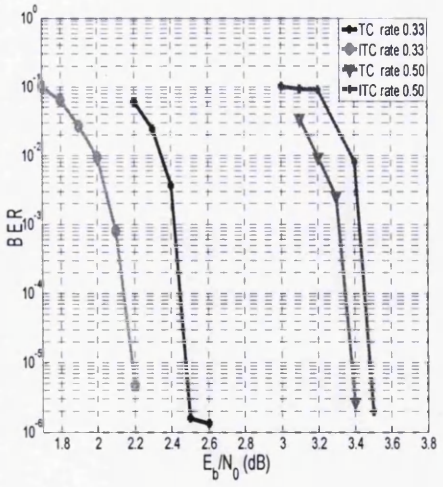


Figure 5.9 BER performance comparisons between TCs and I-TCs in an AWGN channel with 16QAM.

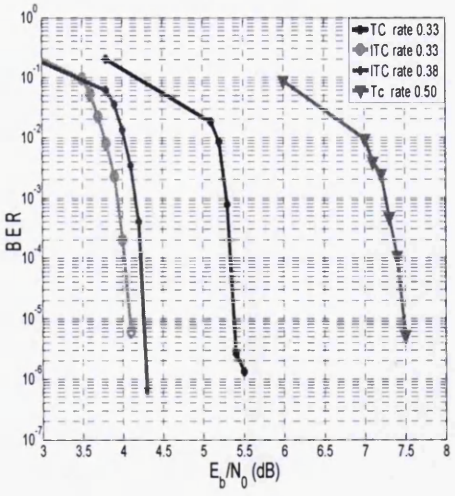


Figure 5.10 BER performance comparisons between TCs and I-TCs in an AWGN channel with 64QAM.

Table 5.2 shows the puncturing pattern, the number of iterations required to converge to a bit error rate of 10^{-5}, the required E_b/N_0 value for convergence and the corresponding degree profiles used to achieve this in the I-TCs.

Table 5.2: I-TC at different codes rates with their converging E_b/N_0 in AWGN

Code rate	Channel	Modulation scheme	E_b/N_0 (dB)	S	Puncture pattern	Bit degree d_i (fractions f_i)	Φ
0.33	AWGN	BPSK	0.20	0.32	11101101110	2(0.888), 8(0.06),9(0.052)	31
0.50	AWGN	BPSK	0.75	0.48	10	2(0.95), 9(0.05)	30
0.40	AWGN	QPSK	0.62	0.76	11101101110	2(0.96), 6(0.04)	36
0.41	AWGN	QPSK	0.51	0.80	11101101110	2(0.95), 5(0.05)	23
0.33	AWGN	16QAM	2.20	1.28	unpunctured	2(0.99), 7(0.01)	19
0.50	AWGN	16QAM	3.50	1.96	10	2(0.94), 3(0.06)	26
0.33	AWGN	64QAM	4.10	1.92	11101101110	2(0.85), 7(0.15)	16
0.38	AWGN	64QAM	4.30	2.22	11101101110	2(0.96), 9(0.04)	23

*S and Φ in the table stand for the throughput in bits per channel use and the number of iterations required to converge to a low BER, respectively.

In the BPSK modulation scheme, the I-TC achieved a minimum coding gain of 0.2 dB over its corresponding regular TC for code rate 0.33 and 0.5 as seen in figure 5.7. In the QPSK modulation scheme, the I-TC rate 0.41 has a slight advantage of about 0.09 dB coding gain over the TC rate 0.33. Also there is a 0.07 bits per channel use increase in the I-TC over the TC at their converging E_b/N_0 values. Progressing to the 16QAM modulation, the I-TC rate 0.33 has a 0.3 dB coding gain over its corresponding TC rate 0.33 as seen in figure 5.9. However, the TC rate 0.5 has a coding gain of 0.1 dB over the I-TC rate 0.5 at a BER of 10^{-5}. The largest coding gain recorded in the I-TC over the

TC is seen in figure 5.10 i.e. the 64QAM modulation scheme, where the I-TC rate 0.33 has a significant coding gain of 1.3 dB over its corresponding TC rate 0.33. Table 5.3 compares the number of iterations required for a low probability of error (10^{-5}) in a regular TC and in the I-TC in an AWGN channel using the 5012 bit frame size.

Table 5.3: Number of iterations required for convergence in an I-TC and the TC

Code rate	Modulation Scheme	Φ_{TC}	Φ_{I-TC}
0.33	BPSK	11	31
0.50	BPSK	15	30
0.33	QPSK	15	-
0.40	QPSK	-	36
0.41	QPSK	-	23
0.50	QPSK	11	-
0.33	16QAM	14	19
0.50	16QAM	21	-
0.33	64QAM	24	16
0.38	64QAM	-	23
0.50	64QAM	19	-

* Φ_{TC} and Φ_{I-TC} in the table stand for the number of iterations required to converge to a low BER in the regular TC and the I-TC, respectively.

Table 5.3 shows that in general, the I-TC requires a higher number of iterations to converge to a low BER in comparison to the regular TC. It should be noted that the I-TC requires only one Log-MAP decoder operations in comparison to the regular TC which requires two Log-MAP decoder operations.

Figure 5.11 shows the throughput versus the signal power to noise power ratio for the TC and the I-TC for the AWGN channel using the 5012 bits frame size.

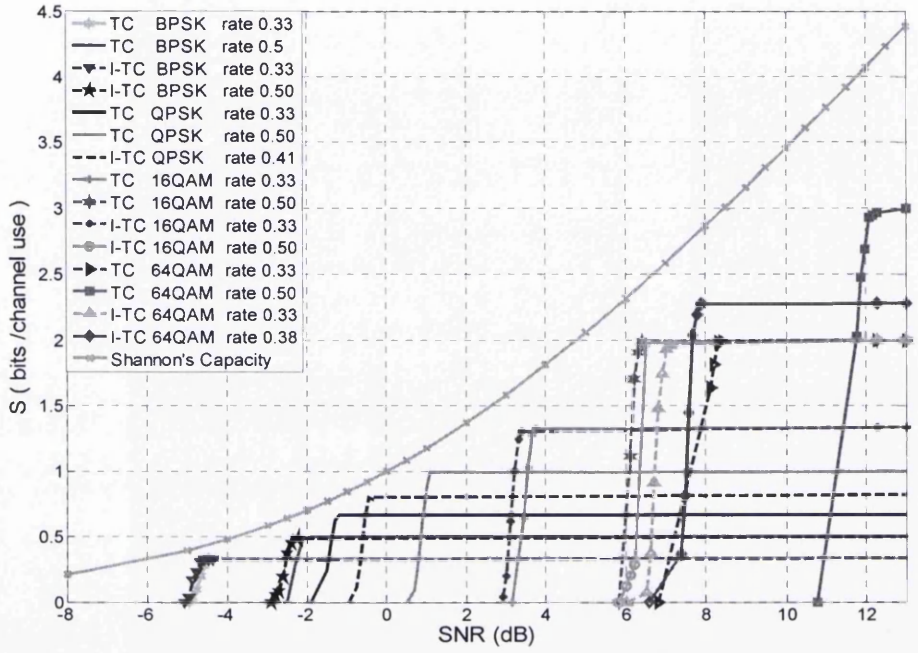


Figure 5.11 Throughput versus SNR for TCs and I-TCs in an AWGN channel for different modulation schemes

The Shannon capacity curve shown in figure 5.11 was calculated by (5.5), i.e. Continuous-input Continuous-output Memoryless Channel (CCMC).

$$\frac{C}{B} = \log_2\left(1 + \frac{P}{N}\right) \quad \text{bit/s/Hz} \qquad (5.5)$$

The throughput per channel use is given by $S = R \times (\log_2 M)(1 - FER)$, where R is the rate of the code, FER the frame error rate and M the M-ary order of the modulation scheme. The frame error rate is computed by dividing the number of frames in error after decoding by the total number of frames used in the simulation. The frame size used is 5012 bits. The SNR difference between Shannon's capacity and the throughput curves

is smaller for the I-TCs in all the modulation schemes and code rates considered than the TCs, except for the 16QAM code rate 0.5 where the I-TC and the TC are of equal distance to Shannon's capacity. The behavioural pattern of the throughput curve for the I-TC and the TC seen above are very similar with the exception that the I-TCs have higher throughput values. An exception to this is in the case of the 64QAM code rate 0.50 where the TC has a higher throughput value but at a significant 4.1dB loss in comparison to the I-TC rate 0.38. In general, the I-TC gives a better performance in comparison to the TC in terms of throughput value at a given SNR value and does not give a worse performance.

5.4 Bit error rate and throughput performance of Irregular Turbo Code in a 3GPP multipath channel.

In this section, the BER performance of the I-TC is examined and then compared with the regular TC in a 3GPP Vehicular A multipath channel [84]. The system model is the same as that described in sections 4.3 and 4.4.1 in Chapter 4 with the simulations done using MATLAB. Figures 5.12 to 5.14 show the BER versus E_b/N_0 performance of the I-TC and the regular TC in the 3GPP multipath Vehicular A channel.

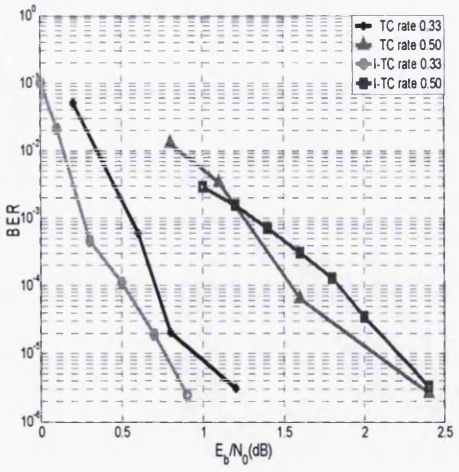


Figure 5.12 BER performance comparisons between TCs and I-TCs in a 3GPP Vehicular A

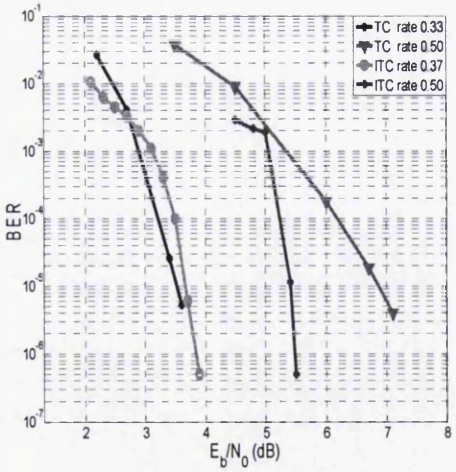


Figure 5.13 BER performance comparisons between TCs and I-TCs in a 3GPP Vehicular A

multipath channel for QPSK. multipath channel for 16QAM.

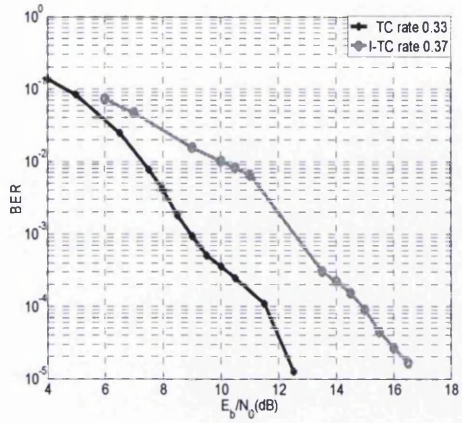


Figure 5.14 BER performance comparisons between TC and I-TC in a 3GPP Vehicular A multipath channel for 64QAM.

There was a problem in the simulation of the I-TC with the QPSK and the 64QAM modulation schemes presented in figures 5.12 and 5.14 which limited the performance of the I-TCs in these schemes. The problem was centred on the extrinsic values in the Log-MAP decoder during iterative decoding. After a certain number of iterations during the decoding, the log likelihood values for the extrinsic information of these I-TCs grow beyond the acceptable or normal range for proper decoding, thereby producing infinite values. This limited the number of iterations in these I-TCs to allow for proper decoding. In the 3GPP Vehicular A multipath channel, the 16QAM I-TC rate 0.37 has a coding loss of about 0.2 dB at a BER lower than 10^{-5} in comparison to the TC rate 0.33, while the 16QAM I-TC rate 0.50 has a significant coding gain of 1.4 dB over a corresponding TC rate 0.50. This makes the I-TC more efficient in terms of power saving but requires more iterations as shown in table 5.4.

Table 5.4 shows the puncturing pattern, the number of iterations required to converge to a BER of 10^{-5}, the required E_b/N_0 value for convergence and the corresponding degree profiles used to achieve this in the I-TC.

Table 5.4: I-TC at different codes rates with their converging E_b/N_0 in the 3GPP multipath Vehicular A channel.

Code rate	Channel	Modulation scheme	E_b/N_0 (dB)	S	Puncture pattern	Bit degree (d_i) and their fractions (f_i)	Φ
0.33	Multipath	QPSK	0.90	0.66	11110	2(0.97), 5(0.03)	6
0.50	Multipath	QPSK	2.20	0.99	10	2(0.98), 4(0.02)	4
0.33	Multipath	16QAM	3.70	1.48	11110	2(0.91), 3(0.09)	12
0.50	Multipath	16QAM	5.40	1.96	10	2(0.98), 4(0.02)	11
0.37	Multipath	64QAM	16.60	2.17	11110	2(0.97), 5(0.03)	13

*S and Φ in the table stand for the throughput in bits per channel use and the number of iterations required to converge to a low BER, respectively

5.5 Conclusions.

In this Chapter, the design and construction of a high performing I-TC has been shown. The BER versus E_b/N_0 performance of the I-TC in comparison to the regular TC using different frame sizes were also shown where larger frame sizes, had larger coding gains in the I-TCs and the TCs. Furthermore, the I-TC in this thesis has been shown to be capable of converging to a low probability of error with lesser complexity in comparison to other I-TCs in terms of the number of iterations and frame sizes. In comparison to the regular TC, the I-TC in general uses a single recursive Convolutional code and a single soft-input soft-output decoder. Also, the designed I-TC performs no worse than the regular TC and frequently better, for example, it is capable of achieving a coding gain of 1.3 dB over its corresponding regular TC, when used in an AWGN channel with 64QAM modulation and a 1.4 dB coding gain in a 3GPP multipath channel with 16QAM modulation, making it an advantageous alternative to the regular TC in future communication systems. In the next Chapter, the application of irregular coding to the Block Turbo Code is done, making it the first designed Irregular Block Turbo Code.

Chapter 6

Near Capacity Irregular Block Turbo Code (I-BTC)

6.1 Introduction

The introduction of iterative decoding (Turbo Codes) by Berrou et. al. in [7] has significantly reduced the transmit power required to achieve negligible BER with moderate to large complexity in digital communication systems. Originally designed for low code rates i.e. code rates 1/3 and 1/2, the Turbo Code achieved a very low BER e.g.10^{-5} with average bit energy to noise energy ratio (E_b/N_0) close to Shannon's theoretical limit in an AWGN channel. For the decoding of the component codes, Berrou used a modified version of the BCJR algorithm [19] known as the maximum *a posteriori* (MAP) algorithm. Due to its moderate complexity and high coding gain, the Turbo Code has been widely adopted as the channel codec in modern wireless systems [43]. Iterative decoding of product codes, also known as Block Turbo Codes, using a soft-input soft-output decoder was then introduced by Pyndiah [8]. The Block Turbo Code achieved a very low BER (10^{-5}) too with an E_b/N_0 close to Shannon's theoretical limit also in an AWGN channel. Pyndiah also introduced a new decoding scheme, "the Chase-Pyndiah" soft decoder, where a selected list of codewords is used to produce soft-outputs from the Chase decoder as Chase is a hard decision decoder. Of importance in this channel codec is its high code rate, making it very useful for high speed communication systems. In the

channel codecs described above, two soft-input soft-output (SISO) decoding components are required to achieve a very low BER.

The unequal protection of information bits has recently been of interest, from the design of an irregular LDPC code to the design of an Irregular Turbo Code. Performance benefits stem from the extra protection on some of the transmitted bits. An Irregular Turbo Code was first proposed by Brendan and McKay in [38], where a coding gain of 0.23 dB at a BER of 10^{-4} was demonstrated over the corresponding regular Turbo Code in an AWGN channel using BPSK modulation. In [40] an Irregular Turbo Code achieved a coding gain of 0.24 dB at 10^{-6} BER in comparison to the regular Turbo Code in AWGN with a BPSK modulation scheme. An important concern in the field of channel coding is the problem of decoding complexities. The Irregular Turbo Code described in [40] utilized a single recursive systematic Convolutional (RSC) encoder and a single SISO decoder. This reduces the complexity of the decoder compared to the regular Turbo Code except for the very large frame size and the high number of iterations required (about 100) to achieve a very low BER as reported in [40].

Recent publications have focused on irregular turbo coding such as in [41] and [42] to achieve a very low BER but no attention has been given to the design of an Irregular Block Turbo Code. In this Chapter, a new irregular coding scheme, an Irregular Block Turbo Code with a greater design flexibility and improved performance in comparison to the best known Block Turbo Codes. This new irregular coding scheme utilizes only a horizontal encoding with no vertical encoding (or vice versa). The regular Block Turbo Code of Pyndiah is in fact a regular turbo product code because both row and column coding / decoding is used. To avoid confusion, in this Chapter, we refer to the Block Turbo Code of Pyndiah as a regular Turbo Product Code (TPC) to distinguish its structure from the new Block Turbo Code proposed here. Our new Irregular Block Turbo Code (I-BTC) only utilizes row encoding. Also, only one SISO decoder is required together with a computational block that executes a slightly higher number of iterations (in comparison to the Pyndiah Turbo Product Code) in order to achieve a low BER (10^{-5}), giving a coding gain of about 1.28 dB for 64QAM modulation in an AWGN channel. Furthermore, the I-BTC is closer to Shannon's theoretical limit than the "regular Turbo Product Code". An important advantage in the design feature of the

I-BTC is its lower complexity in comparison to the "regular Turbo Product Code". The area property of the EXIT function of the I-BTC has been used to calculate the Shannon capacity of the communication channel. This shows how close the new I-BTC is to the capacity of the channel.

The structure of the I-BTC is described in detail in section 6.2 of this Chapter. This is followed by a description of the irregular block turbo decoder also in section 6.2, where a detailed explanation of the iterative decoding of the I-BTC with the exchange of extrinsic information between a SISO decoder and an extrinsic computation block is explained. An EXIT function of the I-BTC and its area property is examined in section 6.3 together with the simulation results i.e. the BER versus E_b/N_0 system performance. Section 6.4 highlights the key advantages of I-BTC over BTC. Potential applications of the I-BTC are summarized in section 6.5. This Chapter ends with a conclusion on the I-BTC in section 6.6

6.2 Design and construction of an I-BTC

In a regular TPC, we encode horizontally (or vertically) followed by a vertical (or horizontal) encoding of the generated information bits in a block format. It can be inferred that a block interleaver exists between the horizontal and the vertical encoders. Figure 6.1(a) shows the encoding structure of a regular TPC. Figure 6.1 (b) depicts a re-designed equivalent structure of the TPC which has only one encoding component, where all the information bits have been repeated twice (this applies only to TPCs with an even number of information bits) as an example. This is so because, each information bit in the degree 2 I-BTC codeword has two extrinsic information values in the decoding process which is similar to the regular TPC with two extrinsic information values per information bit (one from the horizontal decoding and the other from the vertical decoding). It will be shown in section 6.3 of this Chapter that the performance of the

TPC in [10] and the equivalent structure in figure 6.1(b) (a degree 2, uniformly repeated TPC) have the same BER versus E_b/N_0 performance for large block sizes in an AWGN channel with slight differences. It should be noted that the I-BTCs for small block sizes (as shown later in this Chapter) have poor BER performances in comparison to their equivalent TPCs. Figues 6.1(a), (b) and (c) is presented for the first time.

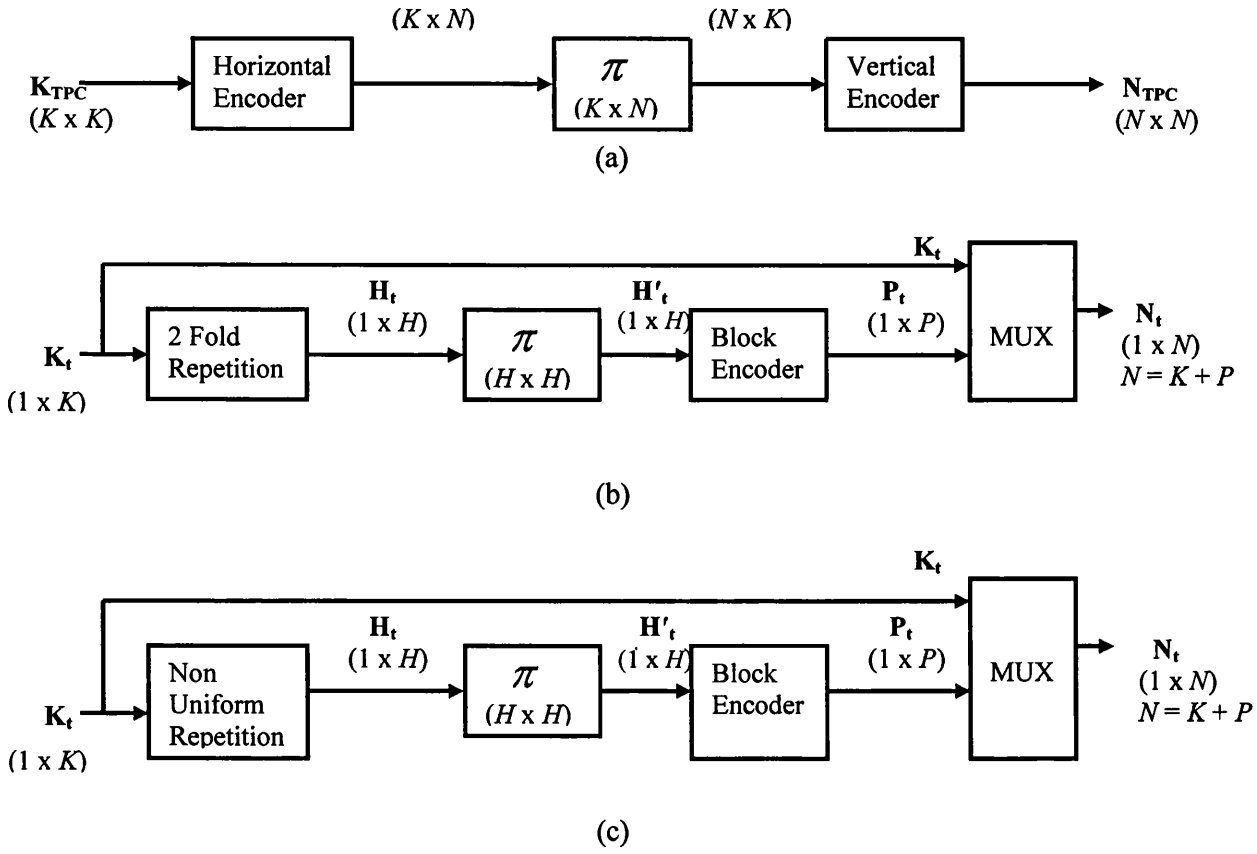


Figure 6.1 (a) Encoding structure for a TPC. (b) An equivalent encoding structure for a TPC. (c) Encoding structure for an Irregular Block Turbo Code.

The general structure of an I-BTC is shown in figure 6.1(c). To design an I-BTC, the equivalent TPC structure is used, where a fraction of the information bits is repeated d times (called the degree) where $d > 2$, for example, with $d = 5$ the bits of higher degree are well protected because their *a posteriori* values include 5 extrinsic information bits instead of 2. The non-uniform repetition divides the information bits into groups indexed

by $i = 1, 2, 3, ...$ with each group having a certain number of repetitions d_i where $d_i = 2, 3, ...T$, and T is the maximum number of repetition. The number of bits in a group i is a fraction f_i of the total number of information bits. The bits belonging to group i are repeated d_i times by the non-uniform repetition. The output of the repeater is then randomly interleaved and passed onto an encoder. Information on the random interleaver, bit degree selection and their corresponding fractions used in this thesis is explained in section 6.2.1. Vectors are represented with bold capital letters. The parity vector $\mathbf{P_t}$ from the block encoder and the original $\mathbf{K_t}$ information vector before the non-uniform repetition are then transmitted in a block format, with the parity vector appended to the information vector for systematic encoding. If $\mathbf{K_t}$ is the information vector, $\mathbf{N_t}$ the transmitted vector and $\mathbf{P_t}$ the parity vector, then the following equations show the relationship between the average bit degree, the fractions f_i and their various non-uniform repetition degrees d_i.

$$\sum_{i=2}^{T} f_i = 1 \qquad (6.1)$$

$$\sum_{i=2}^{T} i.f_i = \bar{d} \qquad \bar{d} \text{ is the average bit degree} \qquad (6.2)$$

$$R = \frac{K}{K + P} \qquad (6.3)$$

In (6.3) R is the code rate of the I-BTC while K and P represent the length of vectors $\mathbf{K_t}$ and $\mathbf{P_t}$, respectively. It should be noted that, the degrees (d_i) used in constructing an I-BTC are limited by the block size of its corresponding TPC. Let us consider a systematic BCH code (n, k, ∂) where n is the codeword length, k the number of information bits and ∂ the minimum hamming distance. In generating the $\mathbf{K_t}$ information vector, there is a limit to the dimension of $\mathbf{K_t}$ that could be generated and grouped before being repeated to give the required length for a given systematic BCH code. For example, an I-BTC derived from a (127, 120, 4) systematic BCH code could have its $\mathbf{K_t}$

information vector as 1 by 60 bits, i.e. one row and 60 columns. Using a degree 2 repetition on vector $\mathbf{K}_t$ produces $\mathbf{H}_t$, a 1 by 120 vector (i.e. fraction $f_i = 1$ and degree $d_i = 2$) with i indicating the group index. In this case we have only one group with its fraction f_i (i.e. $i = 1$ and $f_1 = 1$). This ensures that $\mathbf{H}_t$ retains the original dimension of the TPC information block before encoding. Before interleaving, H (i.e. the length of repeated information bits) number of rows is stored with each row a 1 by 120 vector as explained earlier. In this study, the number of rows of data bits stored equals the number of rows of data bits in a regular TPC for fair comparison. More details on this are given in section 6.3 of this Chapter, showing the sensitivity of the I-BTC to the number of rows chosen. This then produces an array of dimension 120 by 120. Random interleaving of this array is then performed. Each row of the interleaved (120 by 120) array is individually read out as vector $\mathbf{H'}_t$ and separately encoded using the (127, 120, 4) systematic BCH code. The parity vector $\mathbf{P}_t$ from each encoding is then attached to the originally generated $\mathbf{K}_t$ information bit vector for transmission. Before transmission, H (i.e. length of repeated information bits) number of rows is again stored with each row a 1 by 67 vector. This example produces a high rate (60 / 67 = 0.896) I-BTC of degree 2 with the information vector $\mathbf{K}_t$, a 1 by 60 vector and the transmit vector $\mathbf{N}_t$, a 1 by 67 vector (parity vector $\mathbf{P}_t$ is 1 by 7), as depicted in figure 6.1.1.

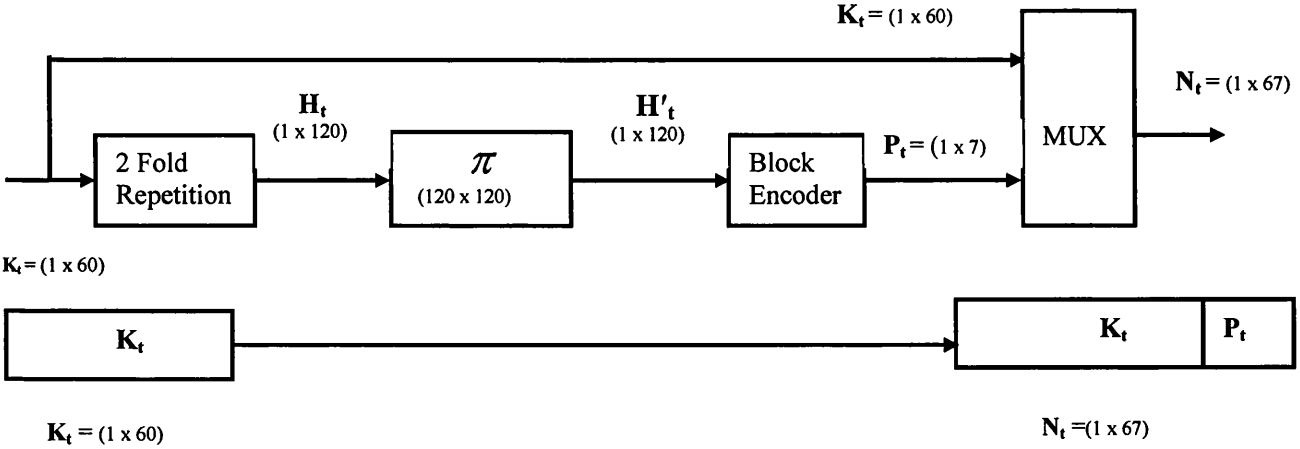


Figure.6.1.1 Encoding a (67, 60)I-BTC (degree 2) from a (1 by 60) Information vector

In general, the repeated $\mathbf{H_t}$ vector should be equal to the original dimensions of the TPC information block before encoding. In terms of block sizes, a (127,120)TPC i.e. a TPC encoded horizontally and vertically with a (127, 120, 4) BCH encoder (i.e. each information bit is encoded twice), would require a block size of (127 x 127) with a code rate of 0.89 while a corresponding I-BTC to the (127,120)TPC described above would require a transmit block size of (120 x 67) with code rate 0.896 (i.e.~0.9), where H equals 120 in this case.

In this study, a modified BCJR algorithm, the Log-MAP algorithm has been used as the decoder for the regular TPC and the I-BTC as originally proposed in [19] and subsequently used in [18]. Firstly, the received $\mathbf{N_r}$ vector is demultiplexed into $\mathbf{K_r}$ and $\mathbf{P_r}$ vectors. Secondly, the $\mathbf{K_r}$ information vector is repeated (as at the transmitter) to give $\mathbf{H_r}$. Before interleaving $\mathbf{H_r}$, a number of rows equal to the length of $\mathbf{H_r}$ (i.e. *H)* is stored and then randomly interleaved (as at the transmitter) to give $\mathbf{H'_r}$. Each row of $\mathbf{H'_r}$ is then attached to its corresponding parity vector $\mathbf{P_r}$ before being sent to the Log-MAP decoder together with an initial *a priori* value of equal probability i.e. zero log likelihood. Thirdly, the extrinsic information **E** (with the same dimension as the repeated information vector $\mathbf{H_r}$) gleaned from the Log-MAP decoder is computed. H number of rows for the extrinsic information is stored before de-interleaving. The extrinsic information bits are then de-interleaved before being passed into the extrinsic computational block. At every iteration, each information bit of degree d_i will then have a new extrinsic information value which is the product, or sum when using the log likelihood value, of the other $d_i - 1$ extrinsic information values. The new extrinsic information values are then interleaved to give new *a priori* values **A** (same dimension as **E**) which is read out individually for the next decoding iteration. After the final iteration, H rows of the decoded vectors are stored, and then de-interleaved, with each row subsequently selected in the format of the originally generated information vector pattern (i.e. $\mathbf{K_t}$) for comparison with the originally generated information vector $\mathbf{K_t}$. Figure 6.1.2 depicts the decoding process where π and π' denote the interleaving and deinterleaving functions, respectively. In this thesis, a different random interleaver pattern has been used for each (H x K) block of data to be interleaved. This implies that the random interleaver pattern used for the first data block differs from the random

interleaver used for the second data block and so on, i.e., the random interleaver itself is seeded randomly. Figure 6.1.2 is presented for the first time.

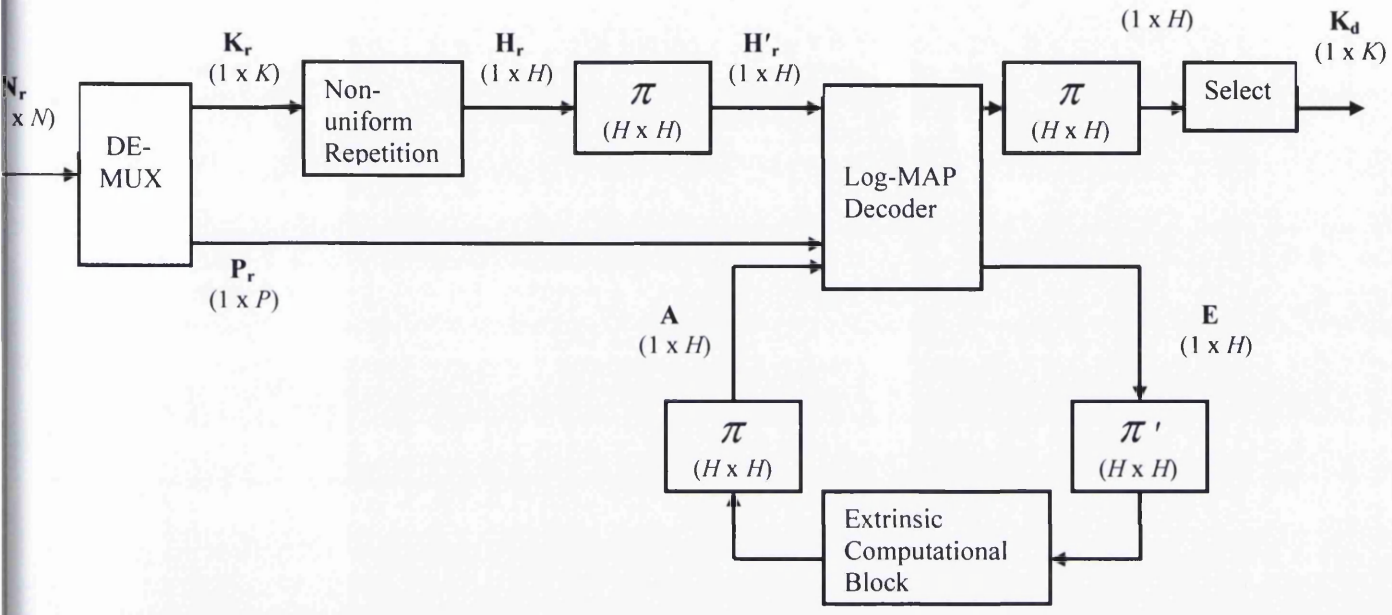


Figure 6.1.2 Iterative decoding structure of an Irregular Block Turbo Code.

The following mathematical derivations as presented in [18] are used here for the systematic BCH trellis decoding. Figure 6.1.3 illustrates the binary trellis of a block code where s' and s denotes the trellis states at times m -1 and m, respectively. The branch connecting time m -1 to m on the trellis is labelled x_m, where x_m is a coded bit. The soft output $L(\hat{x}_m)$ for a given information bit x_m from the Log-MAP decoder can be defined as the conditional *a posteriori* log-likelihood ratio for a received sequence Y. This soft output is given by (6.4).

$$L(\hat{x}_m) = L(x_m \mid Y) = \ln \frac{p(x_m = +1 \mid Y)}{p(x_m = -1 \mid Y)} = \ln \frac{\sum_{\substack{(s',s) \\ x_m=+1}} p(s'_{m-1}, s_m, Y)}{\sum_{\substack{(s',s) \\ x_m=-1}} p(s'_{m-1}, s_m, Y)} \qquad (6.4)$$

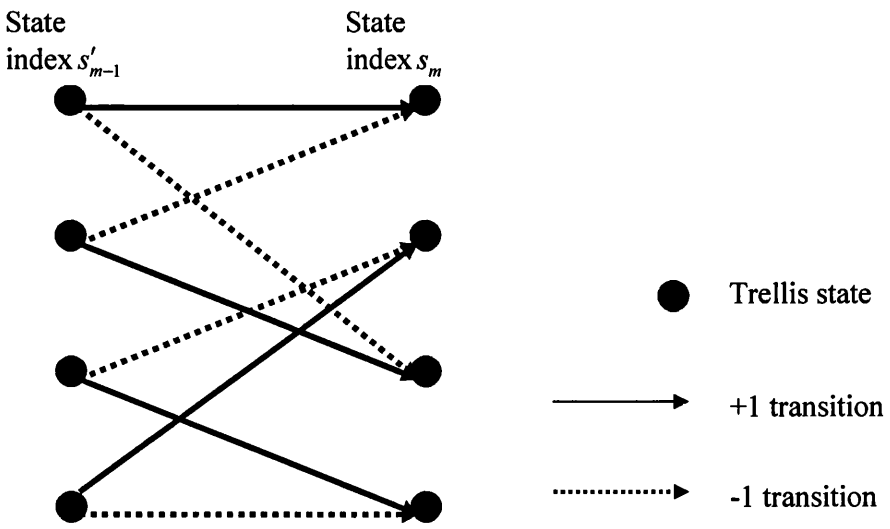


Figure 6.1.3 Example of a trellis structure of a systematic BCH

Assuming a memoryless channel, the joint probability $p(s'_{m-1}, s_m, Y)$ can be written as follows:

$$\begin{aligned} p(s'_{m-1}, s_m, Y) &= p(s'_{m-1}, Y_{j<m}) \cdot p(s_m, Y_m \mid s'_{m-1}) \cdot p(Y_{j>m} \mid s_m) \\ &= p(s'_{m-1}, Y_{j<m}) \cdot p(s_m \mid s'_{m-1}) \cdot p(Y_m \mid s'_{m-1}, s_m) \cdot p(Y_{j>m} \mid s_m) \\ &= \alpha_{m-1}(s'_{m-1}) \cdot \gamma_m(s'_{m-1}, s_m) \cdot \beta_m(s_m) \end{aligned} \tag{6.5}$$

where $Y_{j<m}$ denotes the portion of the received sequence from bit 0 up to bit m-1 [9]. Likewise, the received sequence from bit m up to bit n-1 is denoted by $Y_{j>m}$. $\alpha_m(s_m)$ and $\beta_m(s_m)$ can also be defined as the forward and backward recursions of the MAP decoding, respectively,

$$\alpha_m(s_m) = \sum_{s'} \gamma_m(s'_{m-1}, s_m) \cdot \alpha_{m-1}(s'_{m-1}) \tag{6.6}$$

$$\beta_{m-1}(s'_{m-1}) = \sum_{s} \gamma_m(s'_{m-1}, s_m).\beta_m(s_m) \qquad (6.7)$$

where $\alpha_0(0) = 1$ and $\beta_n(0) = 1$. The branch transition probability is also given by (6.8)

$$\gamma_m(s'_{m-1}, s_m) = p(s_m \mid s'_{m-1}).p(y_m \mid s'_{m-1}, s_m) = p(x_m; y_m) \qquad (6.8)$$

The information bits are assumed to be statistically independent. In an (*n, k*) systematic block code, the transition probability is given by (6.9).

$$\left.\begin{array}{ll} p(x_m; y_m) = p(y_m \mid x_m).p(x_m) & for \quad 1 \le m \le k \\ p(x_m; y_m) = p(y_m \mid x_m) & for \quad k+1 \le m \le n \end{array}\right\} \qquad (6.9)$$

The log likelihood ratio of the received information bit x_m can also be give by (6.9a).

$$L(\hat{x}_m) = \ln \frac{p(x_m = +1)}{p(x_m = -1)} \qquad (6.9a)$$

From equation (6.9a), we derive equation (6.9b), (6.9c) and (6.9d).

$$e^{L(\hat{x}_m)} = \frac{p(x_m = +1)}{1 - p(x_m = +1)}, \quad \text{since } p(x_m = -1) = 1 - p(x_m = +1) \qquad (6.9b)$$

Therefore,

$$p(x_m = +1) = \frac{e^{L(\hat{x}_m)}}{1 + e^{L(\hat{x}_m)}} = \frac{1}{1 + e^{-L(\hat{x}_m)}} \qquad (6.9c)$$

Similarly:

$$p(x_m = -1) = \frac{1}{1+e^{+L(\hat{x}_m)}} = \frac{e^{-L(\hat{x}_m)}}{1+e^{-L(\hat{x}_m)}} \qquad (6.9d)$$

The terms $p(x_m)$ and $p(y_m \mid x_m)$ in (6.9) are the *a priori* and the conditional probabilities, respectively, and are given by (6.10) and (6.11), respectively, using equations (6.9a) to (6.9d)

$$p(x_m = \pm 1) = \left(\frac{e^{-L(x_m)/2}}{1 + e^{-L(x_m)}} \right) . e^{\pm L(x_m).x_m/2}$$

$$= A_m . e^{\pm L(x_m).x_m/2} \qquad (6.10)$$

The same analysis goes for equation (6.11) where L_c is the received channel observation.

$$p(y_m \mid x_m = \pm 1) = \left(\frac{p(x_m).(1 + e^{-L(x_m)}).e^{-L_c.y_m/2}}{1 + e^{-(L(x_m)+L_c.y_m)}} \right) . e^{\pm L_c.y_m.x_m/2}$$

$$= B_m . e^{\pm L_c.y_m.x_m/2} \qquad (6.11)$$

A_m and B_m in equation (6.10) and (6.11) are equal for all transitions from time m-1 to time m and hence cancel out in equation (6.4) when used. As a result of the cancelation, the branch transition probability is reduced to

$$\gamma_m(s'_{m-1}, s_m) = e^{\left((L_c.y_m + L(x_m)) \right) x_m/2)} \qquad (6.12).$$

In a systematic block code such as the BCH used in this thesis, *a priori* probability $L(x_m)$ is equal to zero if x_m is a parity bit, since the *a priori* values are *LLR* values which give the likelihood of an information bit being a +1 or a -1 [18].

The Log-MAP soft output of equation (6.4) is then given by (6.13).

$$L(\hat{x}_m) = L_c y_m + L(x_m) + \ln \frac{\sum_{\substack{(s',s),\\ x_m=+1}} \alpha_{m-1}(s'_{m-1}).\beta_m(s_m)}{\sum_{\substack{(s',s),\\ x_m=-1}} \alpha_{m-1}(s'_{m-1}).\beta_m(s_m)} \tag{6.13}$$

Where $L_c y_m$, $L(x_m)$ and the last term in equation (6.13) are the channel observation, *a priori* and the extrinsic information for bit x_m. An approximation of the extrinsic information can be obtained using $\log_e(e^{L_1}+e^{L_2}) \approx \max(L_1, L_2)$ and equation (6.13) is then simplified to (6.14).

$$\begin{aligned} L(\hat{x}_m) = {} & L_c y_m + L(x_m) \\ & + \max_{(s'_{m-1},s_m),x_m=+1} \left\{ \log_e[\alpha_{m-1}(s'_{m-1})] + \log_e[\beta_m(s_m)] \right\} \\ & - \max_{(s'_{m-1},s_m),x_m=-1} \left\{ \log_e[\alpha_{m-1}(s'_{m-1})] + \log_e[\beta_m(s_m)] \right\} \end{aligned} \tag{6.14}$$

In the extrinsic computational block depicted in figure 6.1.2, the extrinsic information (*E*, the last term in equation (6.13)) at every iteration for the m^{th} bit E_{jm} with a degree of repetition of d_i is recalculated in the log domain using equation (6.15).

$$E_{jm} = \sum_{\substack{l=1 \\ l \neq m}}^{d_i} E_{jl} \tag{6.15}$$

6.2.1 Bit degree combination

Selecting an appropriate bit degree profile d_i and its corresponding fraction f_i is none trivial since they depend on the block length of the code. At present, there is no algorithmic means to derive optimum bit degree combinations. However, from observation it is highly recommended to have a fraction f_i of the information bits repeated once (degree 2 bits) as a substantial part of the bit degree profile. This fraction f_i for the degree 2 bits should also be within a certain neighbourhood range for optimum performance. Also for good performance, the number of groupings i (see section 6.2) should not exceed three. A limiting factor to the number of bit degrees (apart from degree 2) and their frequency of repetition is the number of information bits in the linear code also explained in section 6.2. In general, it was observed that the degree 2 profile must have its fraction f_i between 75% and 99% of the original information bits while the remaining fraction is shared between degrees which vary depending on the modulation scheme. With respect to the above explanation a good bit degree combination would have 90% of the information block matrix with a degree profile of 2, another 6% of the information block matrix with a degree profile of 3, while the remaining 4% of the information block matrix would have a degree profile of 4. A degree 2 repetition for say 60% of the information block matrix has been observed to produce a BER performance in the I-BTC worse than its equivalent TPC. Table 6.2 in section 6.3 shows the various I-BTCs used in this thesis together with their degree profiles and their corresponding fractions f_i. The search for an algorithmic means of determining f_i is a topic of future research.

6.2.2 The EXIT function of an Irregular Block Turbo Code and its area property

The distance to capacity for a channel codec determines how spectrally efficient the channel codec is. In this section, a description of the computation for the area underneath an EXIT function is given. This would be used in determining how close the I-BTC is to capacity in bits per channel use. The area underneath the EXIT functions of the I-BTCs and their corresponding TPCs are calculated in section 6.3 together with their corresponding attainable capacities. These values were then used to compute the distance between the attainable capacities and their corresponding throughput in bits per channel use.

In an EXIT chart, transfer characteristics based on mutual information are used to describe the flow of extrinsic information through the soft in/soft out (SISO) constituent decoders of an iterative decoder. A decoding trajectory is then used to visualize the exchange of extrinsic information between the constituent decoders (i.e. the EXIT functions) [27]. In an I-BTC, the constituent code is a single BCH encoder which can be seen as the inner encoder in a serially concatenated code. In [47], the authors have shown that the area A underneath the EXIT function of an inner code using BPSK modulation scheme is given by (6.16).

$$A = \int_0^1 I_e(I_a)\,dI_a = \frac{I_{max}(X;Y)}{R_{in}} \tag{6.16}$$

where I_e is the extrinsic output values, I_a the *a priori* input values, I_e a function of I_a i.e. $I_e(I_a)$, $I_{max}(X,Y)$ the maximum mutual information transfer between the transmitted symbol X and received symbol Y, also known as the capacity, and R_{in} is the rate of the inner code. I_e which is a function of I_a is integrated with respect to I_a (i.e. dI_a). This implies that for a rate one inner code, the area underneath the inner code equals the capacity (C) of the communication channel. In cases where the inner code has a code rates less than unity i.e. $R_{in}< 1$, the area underneath the inner code is an attainable capacity (C_A) (a slightly lower capacity bound) which is slightly lower than the capacity

of the communication channel. In higher order modulation schemes, the attainable capacity (C_A), is given by (6.17) where M is the M-ary modulation order.

$$C_A = A \times R_{in} \times \log_2 M \tag{6.17}$$

$$\text{Also, } C_A = C = A \times \log_2 M \quad \text{when } R_{in} = 1 \tag{6.18}$$

The attainable capacity is the upper capacity bound in bits per channel use for any inner code with its code rate less than one in a Gaussian modelled digital communication channel. The TPCs and the I-BTCs used in this thesis all posses' inner code rates less than one. In section 6.3, the effective throughputs in bits per channel use for each channel codec are compared to their corresponding attainable capacities to determine the capacity loss for each channel codec for different modulation schemes.

6.3 BER performance of I-BTC in BPSK, QPSK, 16QAM and 64QAM modulation schemes

This section illustrates and discusses the system BER performance and the throughput curve of the I-BTC in comparison to the regular TPC. Simulation of the system was done using MATLAB for an AWGN channel. In this study, a single decode iteration of an I-BTC consists of only one Log-MAP calculation whereas in a TPC a single decode iteration consists of two Log-MAP calculations. Thus two decode iterations in an I-BTC is equivalent to a single decode iteration in the TPC. In this study, the length of $\mathbf{K_{TPC}}$ (for the regular TPC) is twice the length of $\mathbf{K_t}$ for the degree 2 I-BTC, which is equivalent in structure to the regular TPC having an even length for its information vector $\mathbf{K_{TPC}}$. As stated earlier in section 6.2 of this Chapter, the performance in AWGN of the Turbo Product Code in [10] and its equivalent degree 2 structure illustrated in figure 6.1 (b) is only the same for large block lengths ($\geq$ the (47, 40)I-BTC). Simulation results showing the BER versus E_b/N_0 curve performance have been plotted for an AWGN channel using the BPSK, QPSK, 16QAM and 64QAM modulation schemes as well as using different I-BTC and TPC block sizes to illustrate the coding gain of large

block sized I-BTCs over its equivalent TPCs as well as the coding loss of small block sized I-BTCs in comparison to their equivalent TPCs. Also, plotted is the throughput (S) in bits per channel use for the various I-BTCs and TPCs in different modulation schemes (BPSK, QPSK, 16QAM and the 64QAM) versus the signal power to noise power ratio (SNR), illustrating a more efficient bandwidth usage in the I-BTC over its equivalent TPC. In respect of computational complexity, the number of operations required to achieve a low BER in the TPC and the I-BTC are compared. Each bit in the decoding Log-MAP trellis requires 23 mathematical operations. Firstly, an operation is required to add the received channel information to the *a priori* values. This is followed by a single operation required to calculate the branch transition probability for each bit. Seven operations are then required for the forward recursion calculation per bit. This includes the exact Jacobian logarithm for the Log-MAP approximation. The same number of operations is also required for the backward recursion. A single operation is then required to calculate the *a posteriori* transition log-confidences. Lastly six operations are required to calculate the *a posteriori* log-likelihood values which also include a Jacobian operation (five operations per bit). On the other hand, the extrinsic computational block requires four mathematical operations per bit, i.e., two operations to calculate the original length of the repeated bits, an operation to group the bits into the various groups i, and an operation to sum them. The above explanation then gives a total of 713, 1449, 2921 and 5865 operations for a Log-MAP decoding in a (31, 26), (63, 57), (127, 120) and (255, 247)TPCs, respectively. Table 6.1 shows the number of decoding operations required for convergence in the various I-BTCs and TPCs taking into account the number of iterations.

Table 6.1 shows that, twelve I-BTCs require fewer operations in comparison to the their corresponding TPCs, with a significant 41.3% reduction in the number of operations required in the QPSK (108, 100)I-BTC coupled with a coding gain of up to 0.98 dB over its corresponding TPC (see Table 6.2). Also, two I-BTCs require about the same number of operations in comparison to their corresponding TPCs, with six I-BTCs requiring a higher number of operations in comparison to their corresponding TPCs. This shows that a good number of the I-BTCs require fewer operations to converge on a low BER.

Table 6.1: Number of operations required to achieve a low BER in the TPCs and the I-BTCs

Code	Modulation Scheme	Number of Log-MAP decodings		Number of iterations		Number of operations		%Δ
		TPC	I-BTC	TPC	I-BTC & code rate	TPC	I-BTC	
(31,26)	BPSK	2	1	5	9 0.67	7130	7533	-5.65%
			1		11 0.72		9207	-29.13%
	QPSK	2	1	9	11 0.67	12834	9207	28.26%
			1		7 0.72		5859	54.35%
	16QAM	2	1	8	6 0.67	11408	5022	55.98%
			1		7 0.72		5859	48.64%
	64QAM	2	1	8	5 0.67	11408	4185	63.32%
			1		9 0.72		7533	33.97%
(63,57)	BPSK	2	1	3	6 0.77	8694	10206	-17.3%
	QPSK	2	1	7	9 0.77	20286	15309	24.5%
	16QAM	2	1	8	7 0.77	23184	11907	48.6%
	64QAM	2	1	2	4 0.77	5796	6804	-17.3%
(127,120)	BPSK	2	1	5	17 0.90	29210	55233	-89.0%
			1		10 0.88		32490	-11.2%
			1		6 0.85		19494	33.2%
	QPSK	2	1	10	11 0.90	58420	35739	38.8%
			1		38 0.88		123462	-111.3%
			1		18 0.85		58482	-0.1%
	16QAM	2	1	10	14 0.90	58420	45486	22.1%
			1		13 0.88		42237	27.7%
			1		16 0.85		51984	11.0%
	64QAM	2	1	7	9 0.90	40894	29241	28.4%
			1		13 0.88		42237	3.2%
			1		14 0.85		45486	-11.2%
(255,247)	BPSK	2	1	11	10 0.93	129030	68850	46.6%
	QPSK	2	1	15	15 0.93	175950	103275	41.3%
	16QAM	2	1	10	17 0.93	117300	117045	0.2%
	64QAM	2	1	8	11 0.93	93840	75735	19.2%

*The (31, 26), (63, 57), (127, 120) and (255, 247) TPCs have fixed code rates of 0.70, 0.82, 0.89 and 0.94, respectively. %Δ in the table stands for the percentage change in the decoding operations with respect to TPC.

Table 6.2 shows the length of information vector $\mathbf{K}_t$ and the transmit vector $\mathbf{N}_t$ used for the various fractions f_i and degrees d_i for the I-BTCs together with the minimum E_b/N_0 in dB required to achieve a probability of error $\leq 10^{-5}$ for the I-BTC in comparison to

the TPC. The various fractions f_i and degrees d_i used for the I-BTCs in this study are the most efficient combinations found in terms of BER performance.

Table 6.2: Comparison between the required E_b/N_0 for TPC and I-BTC to achieve a low BER in an AWGN channel.

Code	Modulation Scheme	TPC E_b/N_0 (dB)	I-BTC E_b/N_0 (dB)	Gain of I-BTC Over TPC (dB)	K , N for I-BTC	Bit degree(d_i) & their fractions (f_i) for I-BTC
(31, 26)	BPSK	3.40	4.60	-1.20	10 , 15	2(0.9) & 8(0.1)
			4.60	-1.20	13 , 18	2(1)
	QPSK	3.40	4.60	-1.20	10 , 15	2(0.9) & 8(0.1)
			4.60	-1.20	13 , 18	2(1)
	16QAM	7.10	8.10	-1.00	10 , 15	2(0.9) & 8(0.1)
			8.40	-1.30	13 , 18	2(1)
	64QAM	12.50	12.50	0.00	10 , 15	2(0.9) & 8(0.1)
			12.30	0.02	13 , 18	2(1)
(63, 57)	BPSK	3.35	3.60	-0.25	20 , 26	2(0.9),10(0.05)& 11(0.05)
	QPSK	3.55	3.50	0.05	20 , 26	2(0.85), 5(0.1) & 13(0.05)
	16QAM	7.04	6.40	0.64	20 , 26	2(0.85), 7(0.05) & 8(0.1)
	64QAM	12.55	11.18	1.37	20 , 26	2(0.80), 7(0.15) & 4(0.05)
(127,120)	BPSK	3.69	3.80	-0.11	60 , 67	2(1)
			3.60	0.09	50 , 57	2(0.9) & 6(0.1)
			3.40	0.29	40 , 47	2(0.9), 7(0.05) & 17(0.05)
	QPSK	5.10	5.15	-0.05	60 , 67	2(1)
			4.80	0.30	50 , 57	2(0.9) & 6(0.1)
			4.30	0.80	40 , 47	2(0.9), 7(0.05) & 17(0.05)
	16QAM	7.78	7.90	-0.12	60 , 67	2(1)
			7.30	0.48	50 , 57	2(0.96) & 12(0.04)
			7.10	0.68	40 , 47	2(0.9) & 12(0.1)
	64QAM	12.48	12.10	0.38	60 , 67	2(1)
			11.50	0.98	50 , 57	2(0.98) & 22(0.02)
			11.20	1.28	40 , 47	2(0.9), 9(0.05) & 15(0.05)
(255, 247)	BPSK	4.24	4.16	0.08	100 , 108	2(0.97),18(0.02)& 17(0.01)
	QPSK	8.13	7.15	0.98	100 , 108	2(0.97),19(0.02)& 15(0.01)
	16QAM	10.06	9.45	0.61	100 , 108	2(0.97),19(0.02)& 15(0.01)
	64QAM	14.47	13.59	0.88	100 , 108	2(0.97),19(0.02)& 15(0.01)

*The (31, 26), (63, 57), (127, 120) and (255, 247)TPCs have fixed code rates of 0.70, 0.82, 0.89 and 0.94, respectively

In section 6.2 of this Chapter, it was stated that the BER versus E_b/N_0 performance of the I-BTC for small block sizes was poor in comparison to their corresponding TPC. This is basically due to the low depth of the random interleaver used in the I-BTCs. The (31, 26) I-BTC family (i.e. the (18, 13)I-BTC and the (15, 10)I-BTC) was used as an example of a small block size I-BTC for comparison with its TPC counterpart. Figures 6.3.1 to 6.3.4 shows the BER performance of the (31, 26) I-BTC family in comparison to its corresponding TPC in four different modulation schemes. It was observed that the BER curve performance of the I-BTC performed worse in terms of coding gain in comparison to the TPC in the BPSK, QPSK and 16QAM modulation scheme. An exception to this was in the 64QAM modulation scheme, where the BER curve performance of the I-BTC was similar to its TPC equivalent.

The performance of the larger block size TPCs and their equivalent I-BTCs were evaluated in terms of BER versus E_b/N_0. Also the throughput in bits per channel use versus SNR was calculated for the larger block size TPCs and I-BTCs. These metrics have been evaluated for four different digital modulation schemes, namely; BPSK, QPSK, 16QAM and 64QAM. Figures 6.3.5 to 6.3.12 show the BER performance of the (63, 57)TPC in comparison to its equivalent I-BTC for BER values up to 10^{-5}. The results show that the (63, 57)TPC rate 0.82 code performs slightly better than the (26, 20)I-BTC rate 0.77 code by 0.25 dB at a BER of 10^{-5} in a BPSK scheme. In the case of higher order modulation schemes e.g. QPSK and 16QAM, the coding gain of the (26, 20)I-BTC increases to 0.05 dB and 0.64 dB in the QPSK and 16QAM modulation schemes, respectively, over the rate 0.82 TPC. Early error floors in the rate (26, 20)I-BTC are observed in the BER plots of figures 6.3.5 to 6.3.7. These early error floors do not occur in larger block length I-BTCs as illustrated in figures 6.3.9 to 6.3.12 which show the BER performance of the (127, 120)TPC in comparison to its equivalent I-BTCs. For this larger block size, the rate 0.85 and 0.88 I-BTCs have higher coding gains than the rate 0.89 TPC for all four modulation schemes investigated. The largest recorded coding gain over the TPC is in the (47, 40)I-BTC for a 64QAM modulation scheme, with a significant 1.28 dB coding gain at a BER of 10^{-5} over the (127, 120)TPC as shown in figure 6.3.12. The degree 2 I-BTC with a code rate of 0.90 has a very similar performance in comparison to the rate 0.89 TPC with respect to distance to

capacity as shown in Table 6.3. In the case of the (108, 100)I-BTC and TPC BER performances shown in figures 6.3.13 to 6.3.16, the I-BTC with code rate 0.93 records as much as 1 dB coding gain over its equivalent rate 0.94 TPC with a QPSK modulation scheme.

Table 6.3: Distances to Capacity for TPCs and the I-BTCs

Code	Modulation Scheme	D(TPC) (dB)	D(I-BTC) (dB)	I-BTC Code rate
(63, 57)	BPSK	1.30	1.58	0.77
	QPSK	1.34	1.42	0.77
	16QAM	1.71	1.99	0.77
	64QAM	3.64	3.66	0.77
(127,120)	BPSK	0.61	0.49	0.90
			0.60	0.88
			1.46	0.85
	QPSK	2.02	2.00	0.90
			1.93	0.88
			1.74	0.85
	16QAM	1.35	1.41	0.90
			1.05	0.88
			1.30	0.85
	64QAM	2.25	1.82	0.90
			1.58	0.88
			1.78	0.85
(255, 247)	BPSK	0.28	0.53	0.93
	QPSK	4.26	3.53	0.93
	16QAM	2.70	2.41	0.93
	64QAM	3.47	2.64	0.93

*The (63, 57), (127, 120) and (255, 247)TPCs have fixed code rates of 0.82, 0.89 and 0.94, respectively. D(TPC), D(I-BTC) in the table represents, the distance to capacity for the TPC and the I-BTC in dB, respectively.

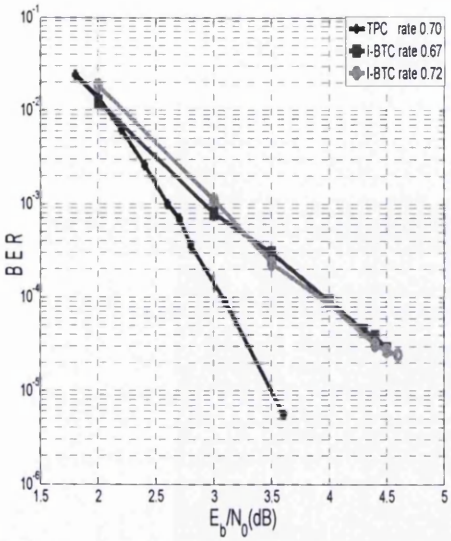


Figure 6.3.1 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.

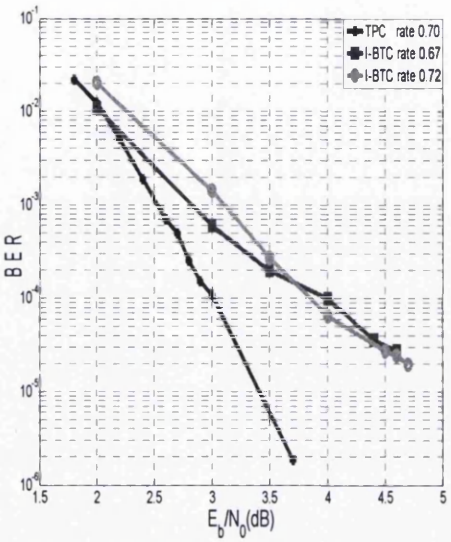


Figure 6.3.2 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK

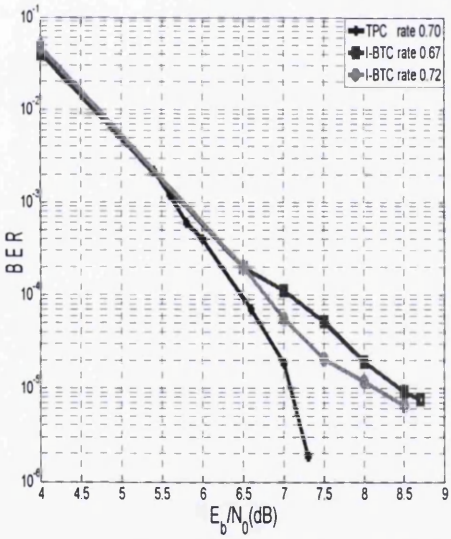


Figure 6.3.3 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.

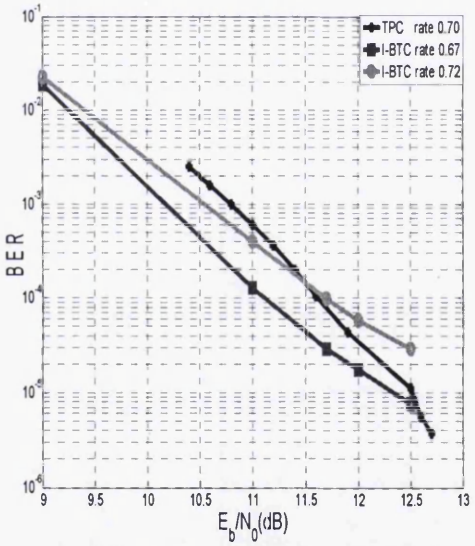


Figure 6.3.4 (31, 26)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.

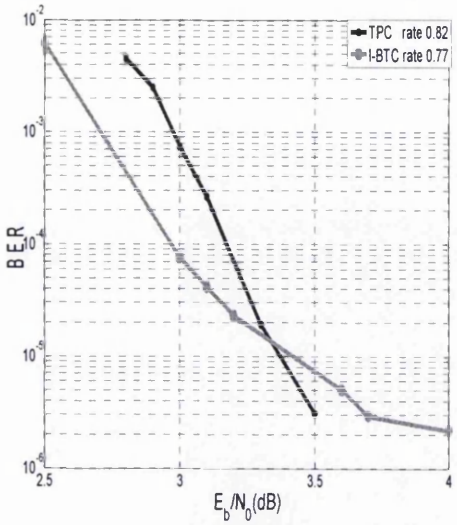


Figure 6.3.5 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.

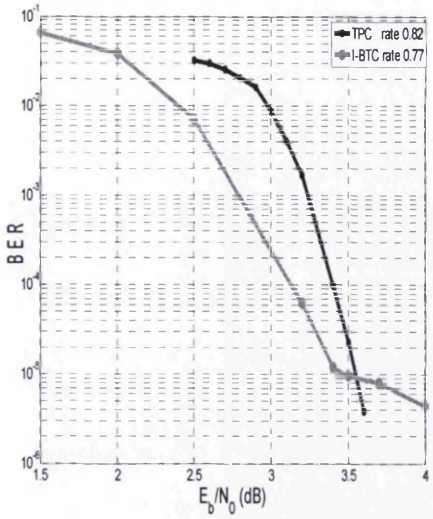


Figure 6.3.6 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK.

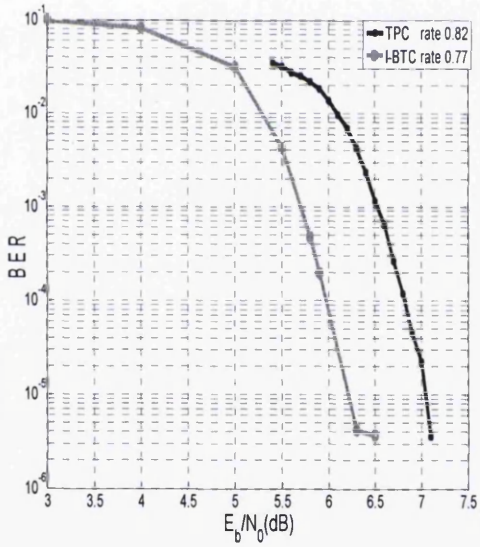


Figure 6.3.7 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.

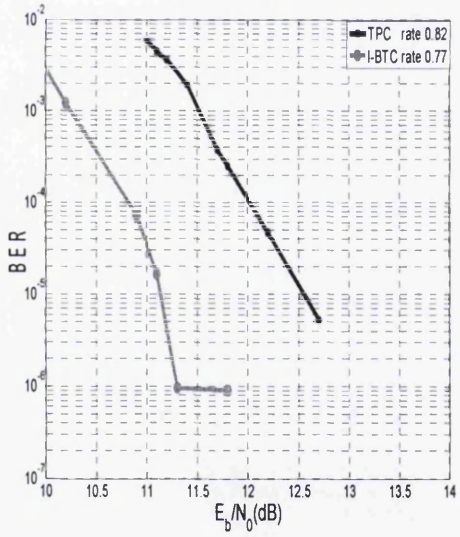


Figure 6.3.8 (63, 57)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.

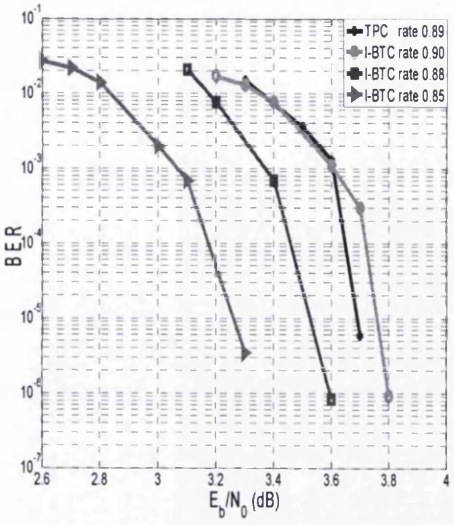


Figure 6.3.9(127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.

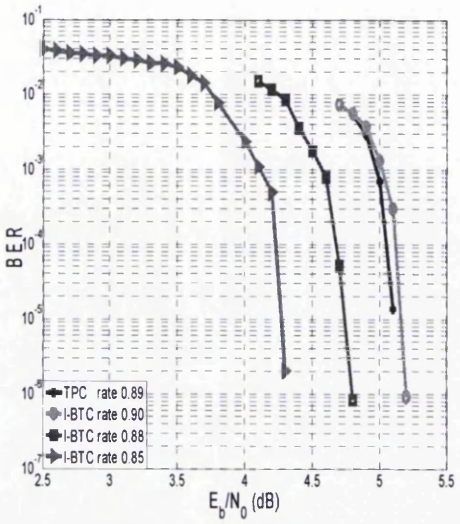


Figure 6.3.10(127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK.

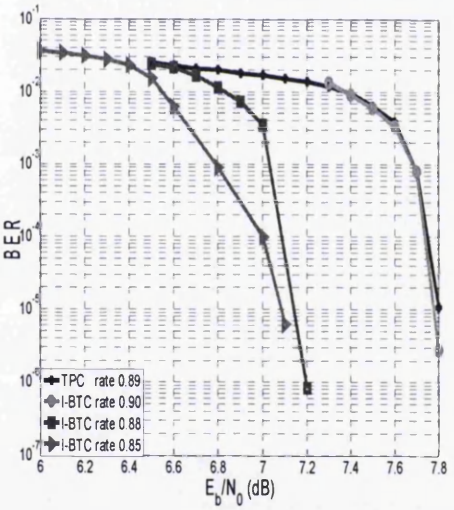


Figure 6.3.11(127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.

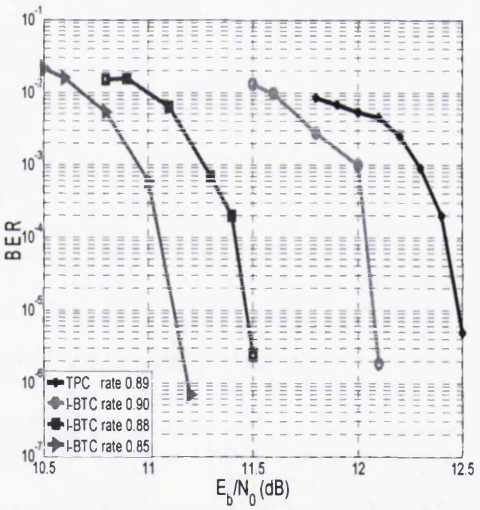


Figure 6.3.12(127, 120)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.

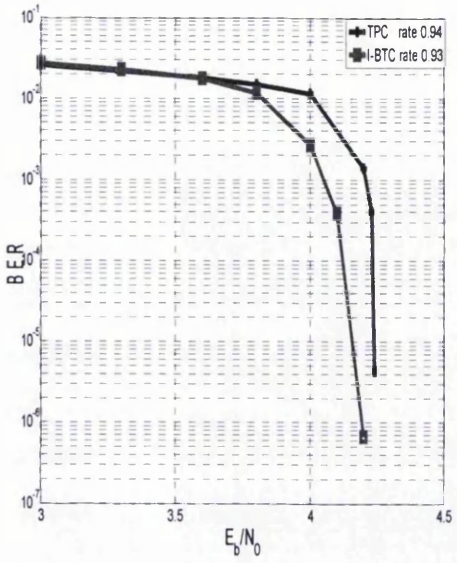


Figure 6.3.13(255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for BPSK.

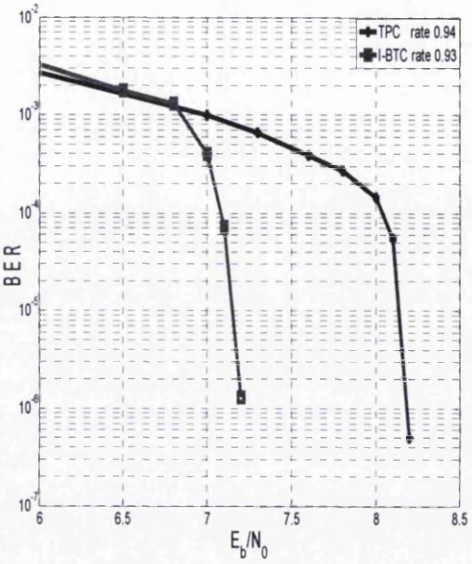


Figure 6.3.14(255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for QPSK.

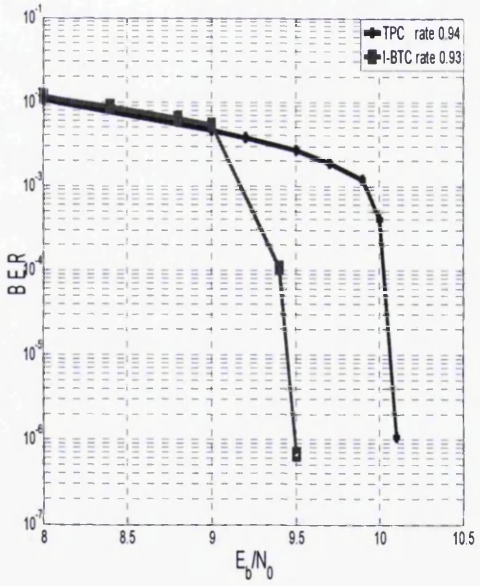


Figure 6.3.15(255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for 16QAM.

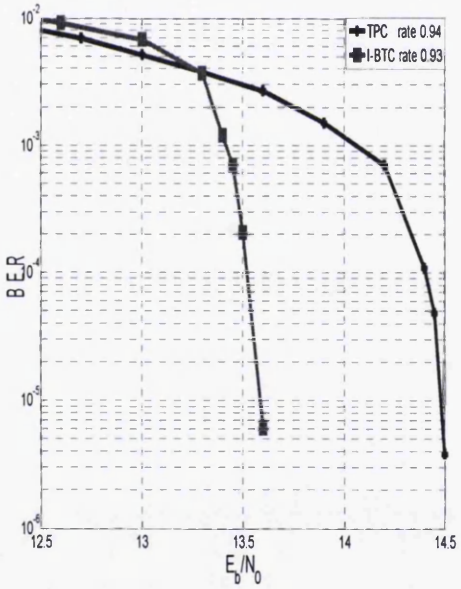


Figure 6.3.16(255, 247)TPC and equivalent I-BTC BER versus E_b/N_0 for 64QAM.

The BER versus E_b/N_0 sensitivity to interleaver depth for the I-BTC is examined, with the view of comparing the BER performance of the I-BTCs for different interleaver depth. The comparison is made in the QPSK modulation scheme. The investigation was performed on the current interleaver depth of (57 by 20), (120 by 40) and (247 by 100) corresponding to the (26, 20), (47, 40) and the (108, 100)I-BTC, respectively. This was done by increasing their interleaver depths by 25% and then decreasing them by 25% and 50%. Figures 6.17, 6.18 and 6.19 illustrate the BER performance of the I-BTCs with changes to their interleaver depth. The results show that an increase in the interleaver depth for the (47, 40)I-BTC from (120 by 40) to (150 by 40) (i.e. 25% increment) resulted to a coding gain slightly larger than the (120 by 40). On the other hand, a reduction in the interleaver depth from (120 by 40) to (90 by 40) and (60 by 40) (i.e. 25% and 50% reduction) resulted to an appreciable coding loss with respect to the (120 by 40) interleaver depth. The same analysis can be said for the (108, 100) and the (26, 20)I-BTCs where the larger interleaver depth resulted into a slightly larger coding gain with the smaller interleaver depth resulting to an appreciable coding loss with respect to their original interleaver depth. An exception to this is the (108, 100)I-BTC, where the larger interleaver depth has a slightly lower coding gain in comparison to the (247 by 100) interleaver depth. This relatively makes the original interleaver depth a better option in terms of BER performance versus complexity since complexity is also a function of the interleaver depth.

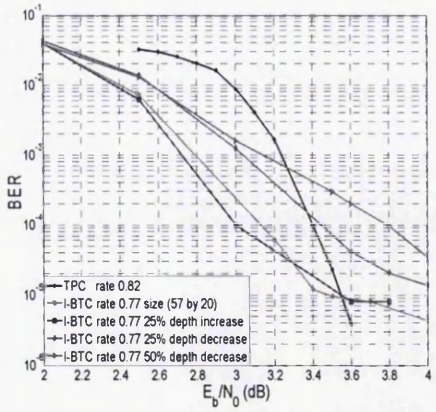


Figure 6.3.17 (63, 57)TPC with different I-BTC array sizes, BER versus E_b/N_0 for QPSK.

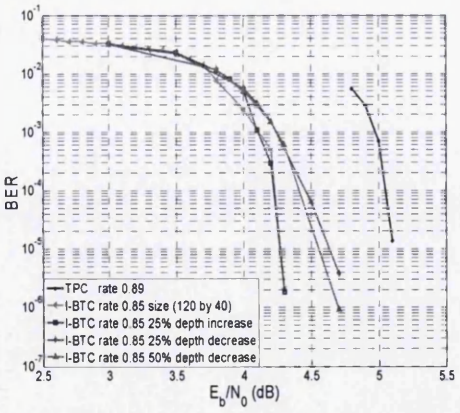


Figure 6.3.18 (127, 120)TPC with different I-BTC array sizes, BER versus E_b/N_0 for QPSK.

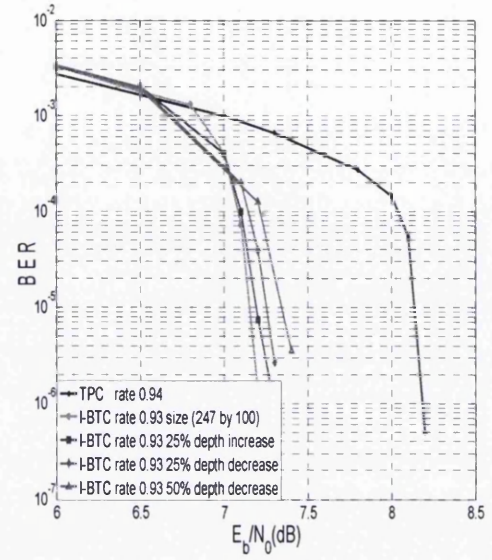


Figure 6.3.19 (255, 247)TPC with different I-BTC array sizes, BER versus E_b/N_0 for QPSK.

The throughput per channel use which is given by $S = R \times (\log_2 M)(1 - BLER)$, where R is the rate of the code, $BLER$ the block error rate and M the M-ary order of the modulation scheme. A block in the TPC is the size of the original information bits i.e. K x K from $\mathbf{K}_{\mathbf{TPC}}$, while a block in the I-BTC consists of H rows of K information bits i.e. H x K. The behavioural patterns of the throughput curves in figures 6.3.20, 6.3.21 and 6.3.22 for the I-BTC and the TPC in an AWGN channel show a continuous coding gain of the I-BTCs over their corresponding TPCs in all four modulation schemes investigated. This is so because the I-BTCs have converged to a very low probability of error before the TPCs can achieve any appreciable throughput values. An exception to this is the degree 2 I-BTC with the same throughput performance as the TPC as shown in figure 6.3.21. The Shannon capacity curve shown in figures 6.3.20, 6.3.21 and 6.3.23 was calculated by (6.19), i.e. CCMC.

$$\frac{C}{B} = \log_2\left(1 + \frac{P}{N}\right) \quad \text{bits/s/Hz} \qquad (6.19)$$

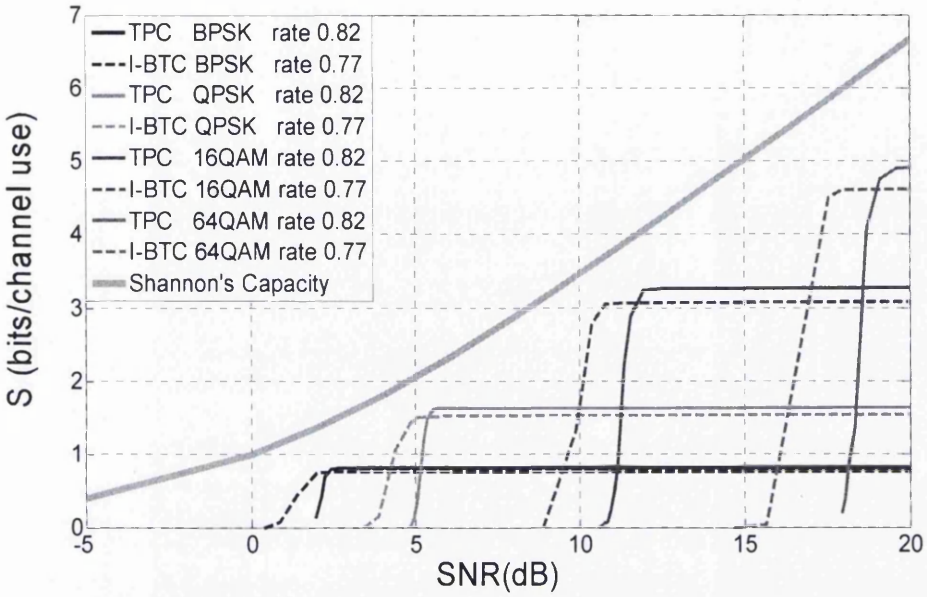


Figure 6.3.20 (63, 57)TPC and equivalent I-BTC throughput per channel use versus SNR

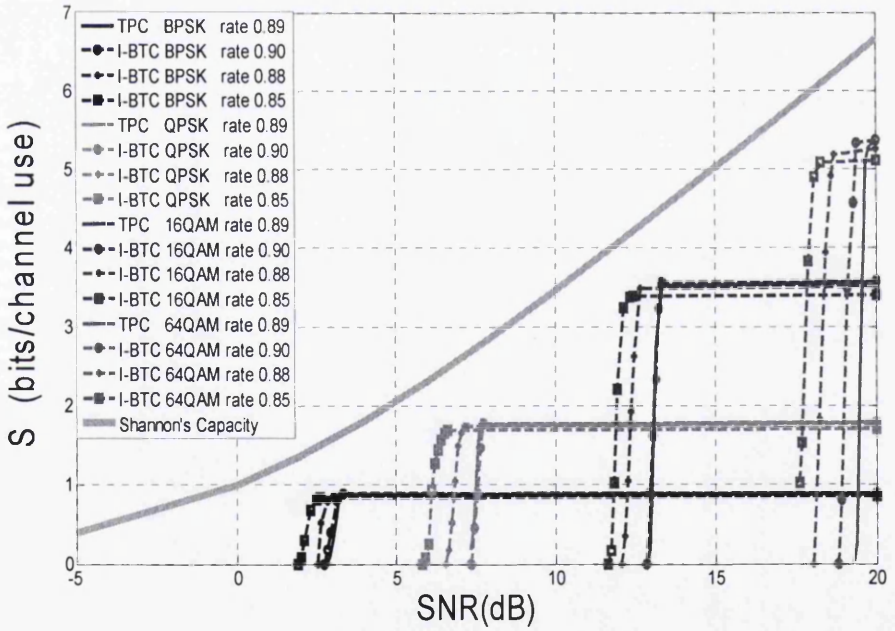


Figure 6.3.21 (127,120)TPC and equivalent I-BTC throughput per channel use versus SNR

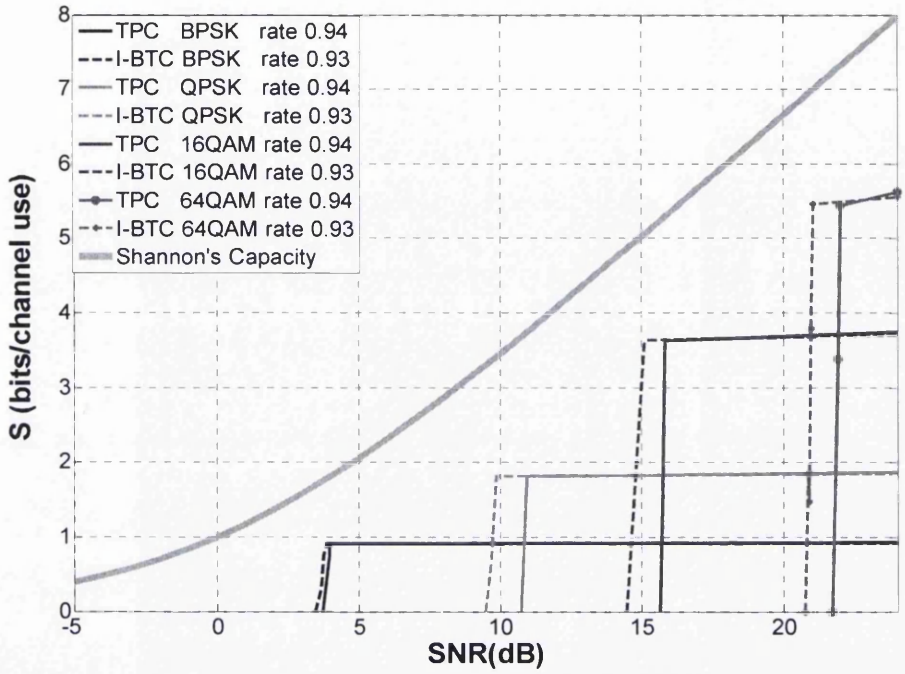


Figure 6.3.22 (255, 247)TPC and equivalent I-BTC throughput per channel use versus SNR

In section 6.2.2, equation (6.17) was used to compute the attainable capacity C_A of the channel for a TPC where R_{in} is the rate of the inner code, which is also the code rate seen by the Log-MAP decoder at the receiver section, i.e. (n, k) (where n is the block length and k, the information bit length, making the code rate seen by the Log-MAP decoder n/k). In the case of the I-BTC, there is no inner code as such, hence the rate R_{in} used in equation (6.17) for the case of the I-BTC is the same as the code rate seen by the Log-MAP decoder for the I-BTC i.e. n/k. Table 6.4 below shows the distance in bits per channel use between the various codes (TPC and I-BTC) and their corresponding attainable capacities computed from their EXIT functions. From table 6.4, the I-BTC with larger block sizes are closer to channel capacity in comparison to their corresponding TPCs (an exception is the (108, 100)I-BTC in a BPSK modulation scheme). The (127, 120) family of I-BTCs and the (108, 100)I-BTC gives good performance due to their large block sizes and the extra protection offered to each

information bit by the repetition code. The new I-BTC, like the TPC, is a high rate code, thereby saving bandwidth during transmission and ensuring efficient bandwidth utilization. This is so because, for every channel use, the parity bits present in high rate codes are low in comparison to the information bits, thereby ensuring maximum utilization of the channel for information purposes.

Table 6.4: Distance to capacity in bits per channel use for TPC and I-BTC

Code	Modulation Scheme	D(TPC)	D(I-BTC)
(63, 57)	BPSK	0.05	0.10
	QPSK	0.10	0.20
	16QAM	0.22	0.31
	64QAM	0.44	0.54
(127, 120)	BPSK	0.04	0.03
			0.04
			0.05
	QPSK	0.10	0.09
			0.10
			0.13
	16QAM	0.16	0.12
			0.16
			0.20
	64QAM	0.24	0.15
			0.27
			0.20
(255, 247)	BPSK	0.04	0.10
	QPSK	0.91	0.40
	16QAM	1.15	0.22
	64QAM	0.96	0.47

*D(TPC), D(I-BTC) in the table represents, the distance to capacity for the TPC and the I-BTC in bits per channel use, respectively.

6.4 Key advantages of the I-BTC over the TPC and the potential applications of the I-BTC

The I-BTC as shown in the previous sections has the capability of achieving coding gains over the regular Block Turbo Code with lower complexity. This coding gain could be as much as 1.28 dB, which is a huge saving in terms of power over a long period of time. The I-BTC interestingly also performs better than the TPC in terms of throughputs. At certain SNR values on the throughout curve, the I-BTC can be used as a channel codec where the TPC is yet to record any significant throughput value. There is also a higher flexibility in designing an I-BTC in comparison to the TPC, because the I-BTC has more than one variant in some cases for every TPC. An example is the (127, 120) family of I-BTC with variants (47, 40), (57, 50) and (67, 60)I-BTCs.

The new I-BTC is a high rate code just like the TPC, thereby saving bandwidth during transmission and ensuring efficient bandwidth utilization. Potential applications of the I-BTCs are in digital communication channels with Gaussian noise such as fixed Broadband (point to point and point to multi-point), VSAT modems (data and voice), optical fiber communication systems and other microwave point to point links.

6.6 Conclusions

In this Chapter an Irregular Block Turbo Code (I-BTC) for Gaussian noise systems, is presented for the first time. The new channel codec is a high speed channel codec. The I-BTC is also a flexible channel codec in terms of construction. The I-BTC is closer to Shannon's capacity in comparison to the existing TPC and capable of converging to a very low probability of error (10^{-5}) at a significant coding gain of 1.28 dB for the (47, 40)I-BTC over its corresponding TPC for 64QAM modulation. This ensures a large energy savings over a long period of time with the use of the I-BTC over its corresponding TPC. Equally important is the lower complexity of the I-BTC over the

TPC, as the I-BTC in some cases requires 46.6% fewer operations to achieve a low BER. In general, the proposed I-BTC due to its extra protection of information bits performs no worse and frequently much better than the existing TPC.

Chapter 7

Conclusions and future work

7.1 Introduction

The research conducted in this thesis primarily focused on designing a new high rate channel codec with lower complexity and nearer performance to the Shannon bound. The use of irregular coding was also another key focus with a view of designing new channel codecs closer to the Shannon bound in comparison to existing channel codecs. Firstly, irregular coding was applied to the already existing Convolutional Turbo Code (CTC), thereby getting it closer to channel capacity. Secondly, irregular coding was also applied to the already existing Turbo Product Code (TPC), making it the first ever Irregular Block Turbo Code (I-BTC) and also getting it closer to channel capacity. This Chapter presents the main conclusions of the research work, and suggests suitable topics for future work.

7.2 Summary of the thesis

In Chapter 4, an iterative decoding "Turbo Code" was examined with an overview of the encoding and decoding structure of the CTC, with particular reference to the UMTS Turbo Code. The strength of the Turbo Code is basically in its extrinsic information, gleaned from individual decoders which is passed from one decoder to the other during

decoding. This was followed by the BER versus E_b/N_0 performance of the CTC in a noisy Gaussian channel and a 3GPP multipath Vehicular A channel using an HSDPA system model. The use of EXIT chart technique in the characterisation of the HSDPA capacity was then examined. This showed the capacity losses in the HSDPA system for four modulation schemes (BPSK, QPSK, 16QAM and the 64QAM). Using the EXIT chart technique, the capacity loss in each of the modulation scheme examined for the CTC at their converging E_b/N_0 value, showed that the further the CTC was from capacity (in bits per channel use), the larger the tunnel gap between the 2 EXIT functions of the EXIT chart. This shows that despite the fact that an open EXIT chart tunnel gap is desired, a narrow EXIT chart tunnel gap indicates a more efficient CTC closer to capacity. It was then observed in Chapter 4, that the EXIT charts of the CTCs at their converging E_b/N_0 values all had open tunnel gaps confirming that the CTCs have converged to a low BER. Also, the CTCs code rate 1/3 were further away from capacity (in bits per channel use) in comparison to the CTCs code rate 1/2 using the same modulation scheme percentage wise with respect to capacity. It was also established in Chapter 4 that, the sum of areas underneath the EXIT functions of the upper and lower decoders of the CTC sums up to the attainable capacity of the channel.

Furthermore, the BER versus E_b/N_0 performance of the binary TPC was analysed in a noisy Gaussian channel. The binary TPC's BER performance was then characterized using four different variants of the TPC, namely; (15, 11)TPC, (31, 26)TPC, (63, 57)TPC and the (127, 120)TPC in four different modulation schemes (BPSK, QPSK, 16QAM and the 64QAM). This surveyed a broad range of the capabilities of the different TPC variants in different modulation schemes. In general, the (31, 26)TPC BER performances in all the four modulation schemes requires the lowest E_b/N_0 value to achieve a very low BER except in the 16QAM modulation scheme. However, the (31, 26)TPC has a lower code rate (i.e. 0.7) when compared to the (63, 57)TPC and the (127, 120)TPC, both having code rates 0.82 and 0.89, respectively. In terms of distance to capacity i.e DCMC capacity (in bits per channel use) for the binary TPCs, the higher code rates were closer to capacity than the lower code rates for the various modulation schemes. This could be attributed to the block size of the higher code rates. In BTCs, the higher code rates possess larger block sizes. Consequently, the higher code rates are

closer to capacity due to their large interleaving depths. The characterisation of the TPC capacity was then evaluated in all the modulation schemes with the use of the EXIT chart technique. The (127, 120)TPC for the BPSK modulation scheme was the closest to attainable channel capacity at a distance of 0.04 bits per channel use among the TPCs. Finally in Chapter 4, the BER versus E_b/N_0 performance of the Reed Solomon TPC in four different modulation schemes was examined. The Reed Solomon TPCs BER curves had a unique characteristic pattern in that, the (15, 11) Reed Solomon TPC converged at a lower E_b/N_0 in comparison to the (31, 27) Reed Solomon TPC, while the (31, 27) Reed Solomon TPC converged at a lower E_b/N_0 value in comparison to the (63, 57) Reed Solomon TPC in all the modulation schemes investigated. This is contrary to the binary case. This is so because, in the Reed Solomon case, the lower code rates have a smaller number of bits per symbol in comparison to the higher code rates. Consequently, each block in the lower code rates possess a smaller number of bits per block with the higher code rates having larger number of bits per block. In the detailed explanation of the "Chase-Pyndiah" algorithm in section 4.7.1 of Chapter 4, a fixed number of least reliable bits p was used in decoding the Reed Solomon TPCs (p was 4 in this case as proposed and used by Pyndiah in [11] and [81] for optimum results with moderate complexity). In the BER curves shown in section 4.7.1 of Chapter 4, four least reliable bits were used, hence, the lower code rates have a greater chance of being correctly decoded at lower E_b/N_0 values. However, in terms of distance to capacity (in bits per channel use), the Reed Solomon TPCs with higher code rates were closer to capacity in comparison to the lower code rates.

Irregular codes were then introduced in Chapter 5, with the design and construction of the I-TC. In Chapter 5, it was observed that the higher the modulation order, the higher the coding gain of the I-TCs over the TCs. This observation was made with the use of 4 different frame sizes (1003, 5012, 10016 and 20072) with the BPSK, QPSK, 16QAM and 64QAM modulation schemes. Also in Chapter 5, it was shown that, the larger the frame size, the larger the coding gains in the I-TCs and the TCs in any of the 4 modulation schemes. This shows that larger frame size codes are closer to Shannon

bound than shorter frame sizes codes due to the large interleaving size of large frame size codes.

The I-TCs designed in this thesis (i.e. in Chapter 5) were shown to be capable of converging to a low probability of error with lesser complexity in comparison to other researchers' I-TCs in terms of the number of iterations and frame size. Also, the designed I-TC in this thesis performed no worse than the regular Turbo Code and frequently better in terms of BER versus E_b/N_0 value. For instance, it is capable of achieving a significant coding gain of 1.3 dB at a BER of 10^{-5} over its corresponding regular Turbo Code, when used in an AWGN channel with 64QAM modulation and a significant 1.4 dB coding gain at a BER of 10^{-5} in a 3GPP multipath channel with 16QAM modulation, making it an advantageous alternative to the regular Turbo Code in future versions of the HSDPA system. Also in Chapter 5, the throughput performance versus SNR of the I-TC was analysed in comparison to the regular Turbo Code. The SNR difference between Shannon's capacity curve and the throughput curves was smaller for the I-TC in all the modulation schemes and different code rates than the regular Turbo Codes, except for the 16QAM code rate 0.5 where the I-TC and the TC were of equal distance to Shannon's capacity.

Chapter 6 introduced a new channel codec, termed I-BTC. The I-BTC is a flexible, high speed channel codec for communication systems over noisy Gaussian channels. The I-BTC is closer to Shannon's capacity in comparison to the existing TPC. The I-BTC is also capable of converging to a very low probability of error (BER of 10^{-5}) at a significant coding gain of 1.28 dB over its corresponding TPC for 64QAM modulation. This is so due to the extra protection of some bits in the I-BTC block, set in a specific manner as explained in section 6.2.1 of Chapter 6. Equally important is the lower complexity in the I-BTC in terms of the number of decoding operations required to converge to a low BER of 10^{-5} in comparison to the number of decoding operations required in the TPC. In some cases, the I-BTC requires almost 47% fewer decoding operations to achieve a low BER of 10^{-5} in comparison to the TPC. The novel I-BTC is well positioned as a future high speed channel codec as it performs no worse and frequently much better than the existing TPCs with lower complexity.

7.3 Suggestions For Future Research

Future work beyond this thesis should be primarily on the non-binary TPC. Firstly, the symbol decoding of the non-binary TPC could be exploited so as to increase the coding gain of the non-binary TPC. In the bit level decoding currently in use for non-binary TPC, the non-binary symbols are converted to bits before the implementation of the "Chase-Pyndiah" decoding algorithm, where the least reliable bits being flipped and new test patterns formed. The proposed symbol level decoding would not be decoded at the bit level, but decoded at the symbol level. In this regard, the future work here is the metric that would be used to form new test patterns. Unlike in the bit level decoding, where there are only 2 symbol levels (of which one could flip the bits binary-wise i.e. only between a 1 and a 0) , the symbol level decoding can possess as many as 16 levels, thereby creating the uncertainty of which symbol level to be used in place of another while forming test patterns. The application of irregular coding to the non-binary TPC together with the design of irregular coding to symbol level decoding of non-binary TPC for the Gaussian is an area for future research.

The performance of the binary I-BTC and non-binary I-BTC in wireless and Rayleigh channels is also another area for further research. TPCs are known to have poor BER versus E_b/N_0 performance in wireless multipath channels. The key to developing an I-BTC (binary or the non-binary) for the Rayleigh and the wireless multipath would be in the degree profile selection. In other words, future research as to how to determine the bit degree profiles for optimal BER curve performance for future I-BTCs depending on the system's channel is an important step forward.

Another future work beyond this thesis should be on the design of higher order I-TC in wireless channels. Recall in section 5.4 of Chapters 5 in this thesis that there was a limitation to the design of I-TCs, which was in the extrinsic information. Future research in line with this should also involve how to determine the bit degree profile for optimal BER curve performance. These are contextual green fields for future research with regards to this thesis.

Appendix A

Matlab sample code for the irregular Turbo Code

```
% Vehicular A rate 1/3

%HSDPA for UMTS WCDMA  with Spreading factor 32 + 1/2 TURBO Encoding
using log-MAP decoding
%Uses multipath channel soft metric
%64QAM over 4 CDM Channels
%QPSK users on 0 Channels
%Outer (inter-frame) Interleaver & Inner (Intra-frame) Interleaver

clear all
tic;
swt=1;

%-------------------------------------------------------------------------
----------------------
% System Parameters
%-------------------------------------------------------------------------
----------------------

N = 32;
HDR_channels    = 4;
Std_channels    = 0;
CDM_codes = hadamard(N);
CDM_codes_used = HDR_channels + Std_channels;
WCsymbolsFrame     = 625;
usedWCsymbolsFrame = WCsymbolsFrame+1;

%-------------------------------------------------------------------------
----------------------
% 16-QAM Input Parameters
%-------------------------------------------------------------------------
----------------------

k_Symbol_qam    = 6;
P_av_qam        = 20;
A_branch_qam    = 1;
No_qam_sym      = HDR_channels;
tot_Symbol_qam  = k_Symbol_qam*No_qam_sym;


%-------------------------------------------------------------------------
----------------------
% Allocate masking Sequence
```

```
%-------------------------------------------------------------------------
----------------------

maskI = Gold_I;
maskQ = Gold_Q;
masking_seqI = reshape(maskI,1,38400);
masking_seqQ = reshape(maskQ,1,38400);
pn = length(masking_seqI);

%-------------------------------------------------------------------------
----------------------
%Coding Parameters
%-------------------------------------------------------------------------
----------------------

g = [ 1 0 1 1;   % feedback loop
      1 1 0 1 ]; % feed forward
[n,K] = size(g);
m = K - 1;
nstates = 2^m;
NTAILBITS = 0;
puncture = 1;       %puncture = 0, puncturing into rate 1/2; %puncture
= 1, no puncturing
a = 1;
dec_alg = 0;
niter = 30;
rate = 1/(2+puncture);

%-------------------------------------------------------------------------
----------------------
% Frame Sizes
%-------------------------------------------------------------------------
----------------------

enc_qambits   = WCsymbolsFrame*No_qam_sym*k_Symbol_qam;
unenc_qambits = enc_qambits*rate;


ferror = 0;
err   = zeros(1,niter);
ferr  = zeros(1,niter);
errs  = zeros(1,niter);
ferrs = zeros(1,niter);


nEN = 0;

inc_dB = 0.1;

min_dB = input(' Enter minimum Eb/No value (dBs)  :   ');
if isempty(min_dB)
   min_dB = 5;
end
max_dB = input(' Enter maximum Eb/No value (dBs)  :   ');
if isempty(max_dB)
```

```
   max_dB = min_dB;
end
TTI = input(' Enter number of packets (1 per TTI) to be transmitted  :
');
if isempty(TTI)
   TTI = 1;
end
framesTTI    = 8;
totalFrames  = TTI*framesTTI
L_total    = WCsymbolsFrame * rate * No_qam_sym*k_Symbol_qam ;
%(960info bits) Input Frame

for eb_n0_dB = min_dB : inc_dB : max_dB

  rate1 = 5000/13360;
   eb_n0_dB=eb_n0_dB
   nEN = nEN+1;
   L_c=1;
   mask_index = 0;
   eb_n0_linear = 10^(eb_n0_dB/10);
   noiseStdDev_qam = sqrt( P_av_qam   /(k_Symbol_qam  * rate1 *
eb_n0_linear));

   errs(nEN,1:niter) = zeros(1,niter);
   ferrs(nEN,1:niter) = zeros(1,niter);

   for numberTTIs = 1 : TTI;


      numberTTIs=numberTTIs
      fErrorsd = 0;

%     ------------------------------------------------------------
%     Turbo Encode Frame Data
%     ------------------------------------------------------------

      frameData_qam = round(rand(framesTTI, L_total-NTAILBITS));
% info. bits (1/0)


       mm =length(frameData_qam);
    s = struct('percentage', 0, 'frequency', 0);
    B = repmat(s, 1, 2);


    B(1).percentage = 97 ;
    B(1).frequency = 2 ;
    B(2).percentage = 3 ;
    B(2).frequency = 5 ;

    for i =1:8
         resultArray(i,:) = checkout(frameData_qam(i,:), B);
     end
```

```
    L_info = length(resultArray);
    newL_total = L_info  ;

    bbb = floor(newL_total/5) ;

      [temp, alpha] = sort(rand(1,newL_total)); % random interleaver
mapping


      for numberCWord = 1 : framesTTI
           en_output3(numberCWord,:)  =
encoderm(resultArray(numberCWord,:) , g ,alpha, puncture );
      end

      en_output2 = (en_output3+1)/2;


      en_output = [frameData_qam   en_output2] ;


      KK= length( en_output );
      kk= KK - length(frameData_qam);

      actualframesize = length(en_output);
%     ------------------------------------------------------------
%     Outer Interleaver - block interleaver across 8 frames
%     ------------------------------------------------------------

       input_Int1 = reshape( en_output', 1 , actualframesize *
framesTTI );
      turboCword_tot_Int1 = interleaver1(input_Int1);

      for numberframe = 1 : framesTTI

          turboCword_qam = turboCword_tot_Int1(1 + (numberframe-
1)*actualframesize : actualframesize + (numberframe-
1)*actualframesize);

%         ------------------------------------------------------------
%         Inner Interleaver - block interleaver across 1 frame
%         ------------------------------------------------------------
%         Padding for 30 column interleaver is required as number of
columns in interleaver is set,
%         so length of frame needs to be multiple of 30. The posistions
of the extra bits are recorded
%         so they can be removed at the interleaver output and re-
inserted at the receiver before de-interleaving

          ipint2        = length(turboCword_qam);
          No_columns    = 30;
          temp1         = ceil(ipint2/No_columns);
          padding_int   = ones( temp1*No_columns - ipint2 , 1 )*5;

          input_Int2    = [ turboCword_qam  padding_int' ];
```

```
          turboCword_qam_Int2 = interleaver2(input_Int2);

          padded_length = length(turboCword_qam_Int2);

%         remove interleaver padding
          countdata = 0;
          countpad  = 0;
          for count_p = 1 : padded_length
              if turboCword_qam_Int2(count_p) == 1
                  countdata = countdata + 1;
                  turboCword_qam(1,countdata) =
turboCword_qam_Int2(count_p);
              else if turboCword_qam_Int2(count_p) == 0
                  countdata = countdata + 1;
                  turboCword_qam(1,countdata) =
turboCword_qam_Int2(count_p);
              else
                  countpad = countpad + 1;
                  position_pad(numberframe,countpad) = count_p;
              end
            end
          end


%-------------------------------------------------------------
%         Modulate, Spread and Scramble Encoded Frame
%-------------------------------------------------------------

%         padding for whole number of Walsh Code symbols
          extrabits       = (ceil(actualframesize/tot_Symbol_qam) *
tot_Symbol_qam)-(actualframesize);
          padding         = ones(1,extrabits);
          turboCword_qam  = [turboCword_qam padding];
          lengthop        = length(turboCword_qam);

          usedWCsymbolsFrame1 = lengthop/( No_qam_sym*k_Symbol_qam );


%         64-QAM Modulate
          for numbersymbols = 1 : usedWCsymbolsFrame1;

            for ix_01 = 1 : tot_Symbol_qam
                inputSymb_qam(ix_01) = turboCword_qam( 1,
((numbersymbols-1) * tot_Symbol_qam + ix_01) );
%               previous Symbol
                if ((numbersymbols-1)*tot_Symbol_qam + ix_01 -
(tot_Symbol_qam)) > 0
                   prevSymb_qam(ix_01) =
turboCword_qam(1,((numbersymbols-1) * tot_Symbol_qam + ix_01) -
(tot_Symbol_qam));
                else
                   prevSymb_qam(ix_01) = 0;
                end
%               next Symbol
```

```
                if ((numbersymbols-1)*tot_Symbol_qam + ix_01 +
(tot_Symbol_qam)) < lengthop  - tot_Symbol_qam + 1;
                   nextSymb_qam(ix_01) =
turboCword_qam(1,((numbersymbols-1) * tot_Symbol_qam + ix_01) +
(tot_Symbol_qam));
                else
                   nextSymb_qam(ix_01) = 0;
                end
            end

%           current previous symbol
            for sym_No2 = 1 : HDR_channels
                qamSymbol(sym_No2,:) = inputSymb_qam((sym_No2 -
1)*k_Symbol_qam + 1 : (sym_No2 - 1)*k_Symbol_qam + k_Symbol_qam);
                inputSeq2(sym_No2,:)    = A_branch_qam *
enc_64qam2(qamSymbol(sym_No2,:));
            end

%           generate previous symbol
            for sym_No1 = 1 : HDR_channels
                qamSymb_prev(sym_No1,:) = prevSymb_qam((sym_No1 -
1)*k_Symbol_qam + 1 : (sym_No1 - 1)*k_Symbol_qam + k_Symbol_qam);
                inputSeq1(sym_No1,:)    = A_branch_qam *
enc_64qam2(qamSymb_prev(sym_No1,:));
            end

%           generate next symbol
            for sym_No3 = 1 : HDR_channels
              qamSymb_next(sym_No3,:) = nextSymb_qam((sym_No3 -
1)*k_Symbol_qam + 1 : (sym_No3 - 1)*k_Symbol_qam + k_Symbol_qam);
              inputSeq3(sym_No3,:)    = A_branch_qam *
enc_64qam2(qamSymb_next(sym_No3,:));
            end

            inputSeqI1 = inputSeq1(:,1);
            inputSeqQ1 = inputSeq1(:,2);
            inputSeqI2 = inputSeq2(:,1);
            inputSeqQ2 = inputSeq2(:,2);
            inputSeqI3 = inputSeq3(:,1);
            inputSeqQ3 = inputSeq3(:,2);

%           CDM Channels
            CDM_signal_I1   =
inputSeqI1*ones(1,N).*CDM_codes(1:CDM_codes_used,:)*1/sqrt(N);
            CDM_signal_Q1   =
inputSeqQ1*ones(1,N).*CDM_codes(1:CDM_codes_used,:)*1/sqrt(N);
            CDM_signal_I2   =
inputSeqI2*ones(1,N).*CDM_codes(1:CDM_codes_used,:)*1/sqrt(N);
            CDM_signal_Q2   =
inputSeqQ2*ones(1,N).*CDM_codes(1:CDM_codes_used,:)*1/sqrt(N);
            CDM_signal_I3   =
inputSeqI3*ones(1,N).*CDM_codes(1:CDM_codes_used,:)*1/sqrt(N);
            CDM_signal_Q3   =
inputSeqQ3*ones(1,N).*CDM_codes(1:CDM_codes_used,:)*1/sqrt(N);

%           Scrambling Code
```

```
            if mask_index >= pn/(3*N)
                mask_index = 0;
            end

            for y = 1 : CDM_codes_used
                CDM_signalI_PNI(y,:) = [CDM_signal_I1(y,:)
CDM_signal_I2(y,:)
CDM_signal_I3(y,:)].*masking_seqI(mask_index*3*N+1:mask_index*3*N+(3*N)
);
                CDM_signalI_PNQ(y,:) = [CDM_signal_I1(y,:)
CDM_signal_I2(y,:)
CDM_signal_I3(y,:)].*masking_seqQ(mask_index*3*N+1:mask_index*3*N+(3*N)
);
                CDM_signalQ_PNI(y,:) = [CDM_signal_Q1(y,:)
CDM_signal_Q2(y,:)
CDM_signal_Q3(y,:)].*masking_seqI(mask_index*3*N+1:mask_index*3*N+(3*N)
);
                CDM_signalQ_PNQ(y,:) = [CDM_signal_Q1(y,:)
CDM_signal_Q2(y,:)
CDM_signal_Q3(y,:)].*masking_seqQ(mask_index*3*N+1:mask_index*3*N+(3*N)
);
            end

%           sum CDM signals
            TotalsignalI_PNI = sum(CDM_signalI_PNI);
            TotalsignalI_PNQ = sum(CDM_signalI_PNQ);
            TotalsignalQ_PNI = sum(CDM_signalQ_PNI);
            TotalsignalQ_PNQ = sum(CDM_signalQ_PNQ);

            Transmit_Sig_Real = (1/sqrt(2))*(TotalsignalI_PNI -
TotalsignalQ_PNQ);
            Transmit_Sig_Imag = (1/sqrt(2))*(TotalsignalI_PNQ +
TotalsignalQ_PNI);


%-------------------------------------------------------------
%           Channel
%-------------------------------------------------------------
            %Multipath Fading Channel
            i=sqrt(-1);
            P1 = 10^(0)      ;
            P2 = 10^(-0.1)   ;
            P3 = 10^(-0.9)   ;
            P4 = 10^(-1)     ;
            P5 = 10^(-1.5)   ;
            P6 = 10^(-2)     ;

            if numbersymbols == 1    %slow fading - only changes at
beginning of each frame
               c1= sqrt(P1/2)*(randn(1)+i*randn(1));
               c2= sqrt(P2/2)*(randn(1)+i*randn(1));
               c3= sqrt(P3/2)*(randn(1)+i*randn(1));
               c4= sqrt(P4/2)*(randn(1)+i*randn(1));
               c5= sqrt(P5/2)*(randn(1)+i*randn(1));
               c6= sqrt(P6/2)*(randn(1)+i*randn(1));
            end
```

```
            multi_channel   = [c1 c2 c3 c4 c5 c6];
            channel_length  = length(multi_channel);

            Transmit_Sig = Transmit_Sig_Real + i*Transmit_Sig_Imag;

            Total_MP_signal_I = real( conv ( Transmit_Sig ,
multi_channel ) );
            Total_MP_signal_Q = imag( conv ( Transmit_Sig ,
multi_channel ) );

%           Add AWGN noise
            noiseI_qam   = randn(1, 3*N + channel_length-1) *
noiseStdDev_qam;
            noiseQ_qam   = randn(1, 3*N + channel_length-1) *
noiseStdDev_qam;
            receiveI_qam = Total_MP_signal_I  + (noiseI_qam*swt);
            receiveQ_qam = Total_MP_signal_Q  + (noiseQ_qam*swt);

%-------------------------------------------------------------
%           Despread and Demodulate
%-------------------------------------------------------------

%          Despread
           receiveI_qam_branch1 =   receiveI_qam ( N + 1 : 2 * N );
           receiveQ_qam_branch1 =   receiveQ_qam ( N + 1 : 2 * N );
           receiveI_qam_branch2 =   receiveI_qam ( N + 2 : (2 * N) + 1
);
           receiveQ_qam_branch2 =   receiveQ_qam ( N + 2 : (2 * N) + 1
);
           receiveI_qam_branch3 =   receiveI_qam ( N + 3 : (2 * N) + 2
);
           receiveQ_qam_branch3 =   receiveQ_qam ( N + 3 : (2 * N) + 2
);
           receiveI_qam_branch4 =   receiveI_qam ( N + 4 : (2 * N) + 3
);
           receiveQ_qam_branch4 =   receiveQ_qam ( N + 4 : (2 * N) + 3
);
           receiveI_qam_branch5 =   receiveI_qam ( N + 5 : (2 * N) + 4
);
           receiveQ_qam_branch5 =   receiveQ_qam ( N + 5 : (2 * N) + 4
);
           receiveI_qam_branch6 =   receiveI_qam ( N + 6 : (2 * N) + 5
);
           receiveQ_qam_branch6 =   receiveQ_qam ( N + 6 : (2 * N) + 5
);


           mask_sectionI =
masking_seqI(mask_index*3*N+1:mask_index*3*N+(3*N));
           mask_sectionQ =
masking_seqQ(mask_index*3*N+1:mask_index*3*N+(3*N));

%          RAKE Branch 1
           for z = 1 : N
```

```
            despread_I_PNI_branch1(z) =
sum(receiveI_qam_branch1.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_I_PNQ_branch1(z) =
sum(receiveI_qam_branch1.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
            despread_Q_PNI_branch1(z) =
sum(receiveQ_qam_branch1.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_Q_PNQ_branch1(z) =
sum(receiveQ_qam_branch1.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
         end
         despreadI_branch1 = 1/sqrt(2)*(despread_I_PNI_branch1 +
despread_Q_PNQ_branch1);
         despreadQ_branch1 = 1/sqrt(2)*(despread_Q_PNI_branch1 -
despread_I_PNQ_branch1);

%        RAKE Branch 2
         for z = 1 : N
            despread_I_PNI_branch2(z) =
sum(receiveI_qam_branch2.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_I_PNQ_branch2(z) =
sum(receiveI_qam_branch2.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
            despread_Q_PNI_branch2(z) =
sum(receiveQ_qam_branch2.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_Q_PNQ_branch2(z) =
sum(receiveQ_qam_branch2.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
         end
         despreadI_branch2 = 1/sqrt(2)*(despread_I_PNI_branch2 +
despread_Q_PNQ_branch2);
         despreadQ_branch2 = 1/sqrt(2)*(despread_Q_PNI_branch2 -
despread_I_PNQ_branch2);




         % RAKE Branch 3
         for z = 1 : N
            despread_I_PNI_branch3(z) =
sum(receiveI_qam_branch3.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_I_PNQ_branch3(z) =
sum(receiveI_qam_branch3.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
            despread_Q_PNI_branch3(z) =
sum(receiveQ_qam_branch3.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_Q_PNQ_branch3(z) =
sum(receiveQ_qam_branch3.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
```

```
          end
          despreadI_branch3 = 1/sqrt(2)*(despread_I_PNI_branch3 +
despread_Q_PNQ_branch3);
          despreadQ_branch3 = 1/sqrt(2)*(despread_Q_PNI_branch3 -
despread_I_PNQ_branch3);

%         RAKE Branch 4
          for z = 1 : N
             despread_I_PNI_branch4(z) =
sum(receiveI_qam_branch4.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
             despread_I_PNQ_branch4(z) =
sum(receiveI_qam_branch4.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
             despread_Q_PNI_branch4(z) =
sum(receiveQ_qam_branch4.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
             despread_Q_PNQ_branch4(z) =
sum(receiveQ_qam_branch4.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
          end
          despreadI_branch4 = 1/sqrt(2)*(despread_I_PNI_branch4 +
despread_Q_PNQ_branch4);
          despreadQ_branch4 = 1/sqrt(2)*(despread_Q_PNI_branch4 -
despread_I_PNQ_branch4);




%         RAKE Branch 5
          for z = 1 : N
             despread_I_PNI_branch5(z) =
sum(receiveI_qam_branch5.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
             despread_I_PNQ_branch5(z) =
sum(receiveI_qam_branch5.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
             despread_Q_PNI_branch5(z) =
sum(receiveQ_qam_branch5.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
             despread_Q_PNQ_branch5(z) =
sum(receiveQ_qam_branch5.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
          end
          despreadI_branch5 = 1/sqrt(2)*(despread_I_PNI_branch5 +
despread_Q_PNQ_branch5);
          despreadQ_branch5 = 1/sqrt(2)*(despread_Q_PNI_branch5 -
despread_I_PNQ_branch5);



%         RAKE Branch 6
          for z = 1 : N
```

```
            despread_I_PNI_branch6(z) =
sum(receiveI_qam_branch6.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_I_PNQ_branch6(z) =
sum(receiveI_qam_branch6.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
            despread_Q_PNI_branch6(z) =
sum(receiveQ_qam_branch6.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionI(N+1:2
*N));
            despread_Q_PNQ_branch6(z) =
sum(receiveQ_qam_branch6.*CDM_codes(z,:)*1/sqrt(N).*mask_sectionQ(N+1:2
*N));
          end
          despreadI_branch6 = 1/sqrt(2)*(despread_I_PNI_branch6 +
despread_Q_PNQ_branch6);
          despreadQ_branch6 = 1/sqrt(2)*(despread_Q_PNI_branch6 -
despread_I_PNQ_branch6);



%         Output of 6 RAKE branches
          despreadI_branch1 = despreadI_branch1(1:HDR_channels);
          despreadQ_branch1 = despreadQ_branch1(1:HDR_channels);
          despread_branch1  =
[reshape(despreadI_branch1,HDR_channels,1)
reshape(despreadQ_branch1,HDR_channels,1)];

          despreadI_branch2 = despreadI_branch2(1:HDR_channels);
          despreadQ_branch2 = despreadQ_branch2(1:HDR_channels);
          despread_branch2  =
[reshape(despreadI_branch2,HDR_channels,1)
reshape(despreadQ_branch2,HDR_channels,1)];




          despreadI_branch3 = despreadI_branch3(1:HDR_channels);
          despreadQ_branch3 = despreadQ_branch3(1:HDR_channels);
          despread_branch3  =
[reshape(despreadI_branch3,HDR_channels,1)
reshape(despreadQ_branch3,HDR_channels,1)];

          despreadI_branch4 = despreadI_branch4(1:HDR_channels);
          despreadQ_branch4 = despreadQ_branch4(1:HDR_channels);
          despread_branch4  =
[reshape(despreadI_branch4,HDR_channels,1)
reshape(despreadQ_branch4,HDR_channels,1)];


          despreadI_branch5 = despreadI_branch5(1:HDR_channels);
          despreadQ_branch5 = despreadQ_branch5(1:HDR_channels);
```

```
          despread_branch5  =
[reshape(despreadI_branch5,HDR_channels,1)
reshape(despreadQ_branch5,HDR_channels,1)];


          despreadI_branch6 = despreadI_branch6(1:HDR_channels);
          despreadQ_branch6 = despreadQ_branch6(1:HDR_channels);
          despread_branch6  =
[reshape(despreadI_branch6,HDR_channels,1)
reshape(despreadQ_branch6,HDR_channels,1)];



%         'sd_16qam_mp.m' provides a soft decision output using
multipath soft metric
%         'quantiz.m' quantizes soft decisions into finite number of
integer levels from 0 to 2^nsdec-1, nsdec = 4

%         Estimates of complex channel fading
          est_w1 = (c1);
          est_w2 = (c2);
          est_w3 = (c3);
          est_w4 = (c4);
          est_w5 = (c5);
          est_w6 = (c6);

              for output_No = 1 : HDR_channels
                    [output_qam_sd(output_No,:)]  =
sd_64qam_mp_ice(despread_branch1(output_No,:),despread_branch2(output_N
o,:), despread_branch3(output_No,:),despread_branch4(output_No,:),
despread_branch5(output_No,:), despread_branch6(output_No,:),
est_w1,est_w2,est_w3,est_w4,est_w5, est_w6);
                    output_r_sd = reshape(output_qam_sd(output_No,:),
k_Symbol_qam, 1);
                    outputSymbolsd((output_No - 1)*k_Symbol_qam + 1 :
(output_No - 1)*k_Symbol_qam + k_Symbol_qam,:) = output_r_sd;
              end

              for ix_03 = 1 : tot_Symbol_qam
                   rx_sdCword_tot(((numbersymbols-1) * tot_Symbol_qam
+ ix_03) , numberframe ) = outputSymbolsd(ix_03,:);
              end

              mask_index = mask_index + 1;

            end   %numbersymbols

            input_deint2_tempsd ( : , numberframe)  =  rx_sdCword_tot(
1 : (length(rx_sdCword_tot)-extrabits), numberframe );

        end %numberframe

%       ------------------------------------------------------------
```

```
%        Inner De-interleaver - across each frame
%        ----------------------------------------------------------

%        Re-insert extra bits for de-interleaving
         for NCword = 1 : framesTTI
             position1 = 0;
             for count_insert = 1 : length(padding_int)
                  position2 = position_pad(NCword, count_insert);
                  input_deint2sd( ( (position1 + 1) : (position2 - 1) )
, NCword ) = input_deint2_tempsd( (position1 - (count_insert-2) ) :
(position2 - count_insert) , NCword);
                  input_deint2sd( position2 , NCword ) =
padding_int(count_insert);
                  position1 = position_pad(NCword,count_insert) ;
             end
             count_insert = count_insert+1;
             input_deint2sd((position1 + 1 ) : padded_length , NCword )
= input_deint2_tempsd( (position1 - (count_insert-2)) :
length(input_deint2_tempsd) , NCword );

%            Deinterleaver each received frame (Inner deinterleaving)
             rx_sdCword_deint2_t(NCword,:) =
deinterleaver2(input_deint2sd( : , NCword)');

         end

%       Remove deinterleaved extra bits
        rx_sdCword_deint2 = rx_sdCword_deint2_t(: , 1 :
actualframesize)';

%-------------------------------------------------------------
%       De-interleave Frames
%-------------------------------------------------------------

        input_deint1 = reshape(rx_sdCword_deint2 , 1 ,
length(rx_sdCword_deint2) * framesTTI);
        rxsdCword_deint1 = deinterleaver1(input_deint1);

%-------------------------------------------------------------
%       Decode
%-------------------------------------------------------------

%       check length of reconstructed Code Word

        for numberCWord = 1 : framesTTI

          rx_sdCword_qam1 = rxsdCword_deint1(1 + (numberCWord-
1)*actualframesize : actualframesize + (numberCWord-
1)*actualframesize);
           rx_length = length(rx_sdCword_qam1);


           rx_sdCword_qam = rx_sdCword_qam1 / sqrt(42)  ;
```

```
        fErrorsd = 0;


         paritynoise2 = rx_sdCword_qam(1,mm+1:KK);
        resultArraynoise2 = rx_sdCword_qam(1,1:mm);

     paritynoise = great(paritynoise2,bbb,newL_total);


     q = struct('percentage', 0, 'frequency', 0);
     C = repmat(q, 1, 2);


     C(1).percentage = 97 ;
     C(1).frequency = 2 ;
     C(2).percentage = 3 ;
     C(2).frequency = 5 ;

    resultArraynoise22 = checkout(resultArraynoise2, C);

     for i = 1:L_info
        resultArraynoise(1,i) = resultArraynoise22(1,alpha(i));
     end


    % Scale the received bits
     rec_s1 = 0.5*L_c*resultArraynoise  ;
     rec_s2 = 0.5*L_c*paritynoise ;



     rec_s = zeros(0);

     for i = 1:newL_total
       rec_s =[rec_s,rec_s1(i),rec_s2(i)];
     end
     rec_s  ;

      yk = rec_s;



 % Initialize extrinsic information
      L_e(1:newL_total) = zeros(1,newL_total);



%            % to get intra & inter row permutation pattern for
sequence of length colyk/2
%            [interleavedummy, U, T, extraele] =
interleaver(L_e(1:length(frameData_qam)));
```

```
          for iter = 1:niter


              L_a  = L_e(alpha) ;  % a priori info.


            % Decoder one

            if dec_alg == 0
               L_all = logmapo(yk(1,:), g, L_a, 2);                %
complete info.
            else
               L_all = sova(yk(1,:), g, L_a, 2);                  %
complete info.
            end
            L_e1 = L_all - 2*yk(1:2:2*newL_total) - L_a ;           %
extrinsic info.



            L_e_new(alpha) = L_e1;                  % extrinsic info.

         u = struct('percentage', 0, 'frequency', 0);
         A=repmat(u,1,2);


          A(1).percentage = 97 ;
          A(1).frequency = 2 ;
          A(2).percentage = 3 ;
          A(2).frequency = 5;


         L_e_n = Group2(A, L_e_new);

         L_e = L_e_n ;


  % Estimate the info. bits
   xhat(alpha) = (sign(L_all)+1)/2;

            AA = xhat(1:2:9700);
            AB = xhat(9701:5:newL_total) ;


       yes = [AA AB];


     % Number of bit errors in current iteration
     err(iter) = length(find(yes~=frameData_qam(numberCWord,:))) ;

          end   %iter
```

```
            for count_iter = 1 : niter
               if err(count_iter) > 0
                   ferror(count_iter) = 1;
               else
                   ferror(count_iter) = 0;
               end
            end

            % Total number of bit errors for all iterations
            errs(nEN,1:niter) = errs(nEN,1:niter) + err(1:niter)
            ferrs(nEN,1:niter) = ferrs(nEN,1:niter) + ferror(1:niter);

         end  %numberCWord

      end %TTI

      % Bit error rate
      ber(nEN,1:niter) = errs(nEN,1:niter)/totalFrames/(L_total-NTAILBITS)
      fer(nEN,1:niter) = ferrs(nEN,1:niter)/totalFrames



end  %eb_n0_dB



a=['*' '+' '<' 's' '>' 'o' '^' 'v' 'h' 'd' '*' '+' '<' 's' '>' 'o' '^'
'v' '*' '+' '<' 's' '>' 'o' '^' 'v' 'h' 'd' '*' '+' '<' 's' '>' 'o' '^'
'v' '*' '+' '<' 's' '>' '*' '+' '<' 's' '>' ];
for KK = 1:niter
xx= min_dB:0.5:max_dB;
semilogy(xx,ber(1:nEN,KK),a(KK));
hold on;
end
```

Matlab sample code for the irregular Block Turbo Code

```
clear all
block_count = 26;

 k_Symbol_qam     = 6;
 P_av_qam         = 20;

 niter = 15 ;



 EbN0db = input(' Please enter Eb/N0 in dB : default [2.0]     ');
if isempty(EbN0db)
   EbN0db = [2.0];
end

no_frame = input(' Enter number of frames to be transmitted  :    ');
if isempty(no_frame)
   no_frame= 1;
end



for nEN = 1:length(EbN0db)


a = 1;
rate = (13)/(18) ;
en = 10^(EbN0db(nEN)/10);
noiseStdDev_qam = sqrt( P_av_qam   /(k_Symbol_qam  * rate * en));
Lc = 1 ;


   err = zeros(1,niter);
   errs(nEN,1:niter) = zeros(1,niter);
   ferrs(nEN,1:niter) = zeros(1,niter);

   for nframe=1 : no_frame

kk  = round(rand(block_count,13));
encoded_bits = zeros(block_count,31);



s = struct('percentage', 0, 'frequency', 0);
     B = repmat(s, 1, 1);

     B(1).percentage = 100 ;
     B(1).frequency  = 2  ;
```

```
for i =1:block_count
    Irr_Infobits(i,:) = checkout(kk(i,:), B);
end


% Interleave Infobits

[temp, alpha] = sort(rand(1,676));

new_info = reshape(Irr_Infobits,1,676) ;

Int_Infobits = new_info(alpha);

new_info2 = reshape(Int_Infobits,26,26) ;


% BCH encoderI

for block_index = 1:block_count
    encoded_bits(block_index,:) = bch_31(new_info2(block_index,:));
end

parity = encoded_bits(1:26,27:31) ;

encoded_bits2 = [kk parity] ;

[AA GG] = size(encoded_bits2) ;



for n = 1 :block_count
%     padding
      actualframesize = GG ;
      extrabits = (ceil(actualframesize/k_Symbol_qam) * k_Symbol_qam)-
(actualframesize);
      padding   = ones(1,extrabits);
      en_output_bin(n,:) = [encoded_bits2(n,:) padding];
      [Y Z] =size(en_output_bin);
      lengthop = Z ;
end



      usedWCsymbolsFrame = block_count ;
      U = lengthop / k_Symbol_qam ;



 %     16QAM Modulate
```

```
     for numbersymbols = 1 : usedWCsymbolsFrame;


           inputSymb_qam(numbersymbols,:) = en_output_bin(
numbersymbols, : );


         for sym_No2 = 1 : U
               qamSymbol(sym_No2,:) =
inputSymb_qam(numbersymbols,(sym_No2 - 1)*6 + 1 : (sym_No2 - 1)*6 + 6);
               inputSeq2(sym_No2,:) = enc_64qam2(qamSymbol(sym_No2,:));
           end

           inputSeqI2 = inputSeq2(:,1);
           inputSeqQ2 = inputSeq2(:,2);




%-------------------------------------------------------------
%          Channel
%-------------------------------------------------------------

           real_sig = inputSeqI2 ;
           imag_sig = inputSeqQ2 ;

%          Add AWGN noise
           noiseI_qam   = randn(1, U ) * noiseStdDev_qam;
           noiseQ_qam   = randn(1, U ) * noiseStdDev_qam;
           receiveI_qam = real_sig  + (noiseI_qam)';
           receiveQ_qam = imag_sig  + (noiseQ_qam)';

%-------------------------------------------------------------
%         Demodulate
%-------------------------------------------------------------



despread_symbol = [reshape(receiveI_qam,U,1)
reshape(receiveQ_qam,U,1)];

         for output_No = 1 : U
                 [output_qam_sd(output_No,:)]  =
sd_hd_64qam(despread_symbol(output_No,:));
                 output_r_sd = reshape(output_qam_sd(output_No,:), 6,
1);
                 outputSymbolsd1((output_No - 1)*6 + 1 : (output_No -
1)*6 + 6,:) = output_r_sd;
         end

          outputSymbolsd =  outputSymbolsd1' ;
```

```
        rx_sdCword_qam( numbersymbols,: )   = outputSymbolsd ;

    end


%    check length of reconstructed Code Word

     [T M] = size(rx_sdCword_qam);
     r = length( 1 : M-extrabits);


    rx_a =  rx_sdCword_qam(:,1:r) ;



   rx_new = rx_a(1:26,1:13) ;
   pp     = rx_a(1:26,14:18) ;



s = struct('percentage', 0, 'frequency', 0);
    C = repmat(s, 1, 1);

    C(1).percentage = 100 ;
    C(1).frequency  = 2  ;



for i =1:block_count
    rx_new2(i,:) = checkout(rx_new(i,:), C);
end


rx_new3 = reshape(rx_new2,1,676) ;

rx_new4 = rx_new3(alpha);

rx_new5 = reshape(rx_new4,26,26) ;


rx = [rx_new5 pp] ;

   apriori_encoded_llrs = rx*Lc ;

   apriori_uncoded_llrs = zeros(26,26);


   for iter =1:niter
```

```
   for block_index = 1:26
      [aposteriori_uncoded_llrs(block_index,:),
aposteriori_encoded_llrs(block_index,:)] =
bcjr_decoder(apriori_uncoded_llrs(block_index,:),
apriori_encoded_llrs(block_index,:));
   end


      extrinsic_uncoded_llrs = aposteriori_uncoded_llrs -
apriori_encoded_llrs(1:26,1:26) - apriori_uncoded_llrs ;


     ext_1 = reshape(extrinsic_uncoded_llrs,1,676) ;

     ext_2(alpha)  = ext_1 ;

     ext_3 = reshape(ext_2,26,26) ;



        u = struct('percentage', 0, 'frequency', 0);

         A=repmat(u,1,1);
         A(1).percentage = 100 ;
         A(1).frequency  = 2  ;



  for i =1:block_count
      L_e(i,:) = Group2(A,  ext_3(i,:)  );
  end


  L_e1 = reshape(L_e,1,676) ;

  L_e2 = L_e1(alpha) ;

  L_e3 = reshape(L_e2,26,26) ;


  apriori_uncoded_llrs =L_e3 ;


  yes3 = reshape(aposteriori_uncoded_llrs,1,676) ;

  yes2(alpha)  = yes3 ;

  yes1 = reshape(yes2,26,26) ;
```

```
      yes = yes1(:,1:2:26) ;


     % Estimate the info. bits
        xhat = (sign(yes)+1)/2 ;


    % Number of bit errors in current iteration
         err(iter) = length(find(xhat~=kk)) ;

    end % iter

     for count_iter = 1 : niter
         if err(count_iter) > 0
             ferror(count_iter) = 1;
         else
             ferror(count_iter) = 0;
         end
     end


       errs(nEN,1:niter) = errs(nEN,1:niter) + err(1:niter)
       ferrs(nEN,1:niter) = ferrs(nEN,1:niter) + ferror(1:niter);

    end %nframe

     ber(nEN,1:niter) = errs(nEN,1:niter)/(338*nframe)
     fer(nEN,1:niter) = ferrs(nEN,1:niter)/nframe

     end %nEN

a=[ '*' '>' 'o' 'v' '+' 'x' '^' 's' 'd' '<' '*' '>' 'o' 'v' '+' 'x' '^'
's' 'd' '<' '*' '>' 'o' 'v' '+' 'x' '^' 's' 'd' '<' '*' '>' 'o' 'v' '+'
'x' '^' 's' 'd' '<'];
for KK = 1:niter
xx= [2.5:0.1:2.9] ;
semilogy(xx,ber(1:nEN,KK),a(KK));
hold on;
end
```

Appendix B

Channel parameters for the Vehicular A ITU / UMTS channel

Tap	Average power (dB)
1	0.0
2	-1.0
3	-9.0
4	-10.0
5	-15.0
6	-20.0

References

[1] C. E. Shannon, "The mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[2] T. Cover and J. Thomas "Elements of information theory"

[3] G.D. Forney, "Concatenated codes". *Technical Report 37, MIT, 1965*

[4] Elias, "Error free coding", *Information theory, IRE*, Vol.4 Issue: 4, pg 29-37, Sept 1954.

[5] A. J. Viterbi, "Error bounds for Convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory,* vol. IT–13, pp. 260–269, April 1967

[6] D. J. Muder, "Minimal trellises for block codes", *IEEE Trans. Inform. Theory*, vol. IT–34, pp. 1049–1053, Sept. 1988

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes (1)," *in Proceedings of the IEEE International Conference on Communications*, vol. 2, Geneva, May 1993, pp. 1064–1070

[8] R. Pyndiah, A. Glavieux, A. Picart, S. Jacq, "Near optimum decoding of product codes" Global Telecommunications conference, 1994. *GLOBECOM '94 Communications: The Global Bridge., IEEE*

[9] D. Chase, "Class of algorithms for decoding block codes with channel

measurement information", *Information Theory, IEEE Transaction on*, vol.18, pp. 170-182, 1972.

[10] R. Pyndiah, "Near optimum decoding of product codes: Block turbo codes" IEEE Transaction on Communications, vol. 46, No. 8, Aug. 1998.

[11] O. Aitsab, R. Pyndiah, " Performance of Reed Solomon block turbo Codes Global Telecommunications conference, 1996. Globecom '96. Vol.1, pg 121-125.

[12] R. Zhou; L. Bidan, R. Pyndiah, A. Goalic, " Low-complexity High rate Reed Solomon Block Turbo Code" *Communications , IEEE Transactions on*, Vol.55, Issue:9, pg 1656-1660.

[13] Haykin, Simon S, Communication systems. *New York: Wiley, c2001.*

[14] Robert G. Maunders "Irregular variable length coding" PhD *Thesis Dec.2007, University of Southampton, U.K.*

[15] IEEE 802.16 broadband wireless access working group

[16] J.L. Massey, Information theory. "The Copernican system of communications"

[17] J.R. Pierce, "The early days of information theory"

[18] Hagenauer, J., Offer, E., Papke, L., "Iterative decoding of binary block and Convolutional codes", IEEE Transaction on information theory, Vol. 42, No. 2, March 1996.

[19] Bahl, L.R., Cocke, J., Jelinek, F., Raviv, J., "Optimal decoding of linear

codes for minimizing symbol error rate,", IEEE Transaction on Information Theory, Vol. IT-20, pp. 284-287, Mar. 1974.

[20] T.J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding", *IEEE Trans. Inform. Theory,* vol. 47, pp. 599-618, Feb. 2001

[21] H. El Gamal and A. R. Hammons, "Analyzing the turbo decoder using the Gaussian approximation," *IEEE J. Select. Areas Commun.* , vol. 47, pp. 671–686, Feb. 2001.

[22] D. Divsalar, S. Dolinar, and F. Pollara, "Low complexity turbo-like codes," in *Proc. 2nd Int. Symp. Turbo Codes*, Sept. 2000, pp. 73–80.

[23] M. Peleg, I. Sason, S. Shamai, and A. Elia, "On interleaved, differentially encoded Convolutional codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2572–2582, Nov. 1999.

[24] S. ten Brink, "Convergence of iterative decoding," *Electronics Letters*, vol.35, no. 10,pp. 806–808, May 1999.

[25] S. ten Brink, "Designing Iterative Decoding Schemes with the Extrinsic Information Transfer Chart "*AE¨U Int. J. Electron. Commun.54 (2000)* No.6, 389–398

[26] A. Ashikhmin, G. Kramer, S. ten Brink "Extrinsic information transfer function: Model and erasure channel properties" *Information Theory, IEEE Transactions*

[27] S. ten Brink, "Convergence behaviour of iteratively decoded parallel concatenated codes "*IEEE Trans. communications*, vol. 49,

No. 10, October 2001

[28] M. Tüchler, S. ten Brink, and J. Hagenauer, "Measures for tracing Convergence of iterative decoding algorithms," in *Proc. 4th Int. ITG Conf. Source and Channel Coding*, Berlin, Germany, Jan. 2002.

[29] B. Scanavino, G. Montorsi, and S. Benedetto, "Convergence properties of iterative decoders working at bit and symbol level," in *Proc.2001 IEEE Global Telecommunications Conf. (GLOBECOM '01)*, vol. 2, 2001, pp. 1037–1041

[30] S.ten Brink, S.: Iterative decoding trajectories of parallel concatenated codes. *Proc. 3rd IEEE/ITG Conference on Source and Channel Coding (2000)*, 75–80

[31] Ten Brink, S.: Design of serially concatenated codes based on iterative decoding convergence. *Proc. 2nd International Symp. on Turbo Codes (2000)*, 319–322

[32] M.Tuchler, "Design of serially concatenated systems for long or short block length "*Communications, 2003. ICC '03. IEEE International Conference,* Volume 4, 11-15 May 2003 Page(s):2948 - 2952 vol.4

[33] Alexander Golitschek Edler von Elbwart', Joachim Lohr', Eiko Seidel' "Performance comparison of parallel concatenated codes using EXIT chart prediction" Personal, *Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium*

[34] Tee, R.Y.S.; Maunder, R.G.; Wang, J.; Hanzo, L.; "Near-Capacity Irregular Bit-Interleaved Coded Modulation" *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*

[35] M.G. Luby, M.Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation", *in proceedings of IEEE International Symposium on Information Theory, 1998.*

[36] D.J.C.Mackay, S.T.Wilson, and M.C.Davey, "Comparison of Constructions of irregular Gallager codes", *IEEE Transactions on Communications, Vol.47, October 1999.*

[37] T. Richardson, A. Shokrollahi and R.Urbanke, "Design of provably good low density parity check codes", *Submitted to IEEE Transactions on information Theory, July 1999.*

[38] B. Frey and D. MacKay,"Irregular Turbo Codes" *In proceedings of the 37th Allerton conference on communication, control and computing 1999, Allerton House , Illinois*

[39] B. Frey and D. MacKay, "Irregular turbo-like codes," in *Proc. 2nd Int. Symp. Turbo Codes, Brest, France, Sept. 2000, pp. 67–72.*

[40] H. E. Sawaya, J. J. Boutros : "Irregular Turbo Codes with symbol-based iterative decoding," *Proceeding of the 3rd International Symposium on Turbo Codes and Related Topics,* ISTC, Brest, France, September 2003.

[41] X. Jasper, L. Vandendorpe, " Joint source-channel codes based on Irregular Turbo Codes and variable length codes", *IEEE Transaction on communications*, Vol. 56, no. 11, pg. 1824-1835, November 2008.

[42] G.M. Kraidy, V. Savin. "Irregular Turbo Codes design for binary erasure Channel" *5th International Symposium on Turbo Codes and related topics*, September 2008.

[43] Universal Mobile Telecommunications System (UMTS); Multiplexing and channel coding (TDD) (3GPP TS 25.222 version 7.11.0 Release 7)

[44] S.Benedetto and G. Montorsi, "Role of recursive Convolutional codes in Turbo Code", *IEEE Electronics letters*, vol.31, issue 11:,pg 858-859, 25th May, 1995.

[45] A.O. Sholiyi, B. Badic, R.M. Joyce, T. O'Farrell, "A characterisation of HSDPA Capacity using EXIT chart techniques" *in proceedings of the IEEE NGAMAST, Cardiff, Wales, September 2009.*

[46] J.W. Lee and R.E Blahut, "Generalized EXIT chart and BER analysis of finite-length Turbo Codes" in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM '03), vol 4, pg 2067 – 2072, Dec 2003.*

[47] A. Ashikhmin, G. Kramer, and S. ten Brink, "Code rate and the area under extrinsic information transfer curves," *in Proceedings of the IEEE International Symposium on Information Theory, Lausanne, Switzerland, June 2002, p. 115*

[48] Benard Sklar " Principles of digital communications "

[49] R. Zhou; A. Picart, R. Pyndiah, A. Goalic "Reliable Transmission with low complexity Reed-Solomon Block Turbo Codes" *In proceedings IEEE 1st International symposium on wireless communication systems, 2004,*

[50] M.C.VALENTI, 'Turbo Codes and iterative decoding processing,' *in Proceedings IEEE New Zealand Wireless Communication Symposium, (Auckland, New Zealand), Nov. 1998.*

[51] Sawaya, H.E., '8-PSK combined to regular and irregular symbol based

Turbo Codes,' in 1st international symposium on Control, Communications and Signal Processing, 2004, pp. 163–166.

[52] Richard W. Hamming "Coding and information theory "

[53] R. Pyndiah, A. Picart, A. Glavieux "Performance of Block Turbo Coded 16QAM and 64QAM modulations " *IEEE Global Telecommunications Conference, 1995. Globecom*

[54] D. Chase, " A class of algorithms for decoding block codes with channel measurement information" *IEEE transaction on information theory, vol.18, Issue:1 ,1972 pages 170 – 182.*

[55] J.E.M Nilsson, R. Kotter, " Iterative decoding of product code Construction" *in proceedings of the international symposium on Information theory and its applications, 1994 pages 1059 – 1064*

[56] S. Benedetto, G. Montorsi, " Iterative decoding of serially concatenated Convolutional codes" *IET electronics letters vol. 32, Issue:13, 1996, pages 1186 – 1188.*

[57] M.C. Valenti, B.D. Woerner, " Variable latency Turbo Codes for wireless Multimedia applications" *in proceedings of international symposium on Turbo Codes, Best, France, 1997.*

[58] A. Burr, " Modulation and coding"

[59] P. Robertson, E. Villebrum, P. Hoerher "A comparison of optimal and suboptimal MAP decoding algorithms operating in the log domain" *IICC Seattle, gateway to globalization, IEEE international conference 1995 Vol.2 , 1995, pages 1009 – 1013*

[60] H.H. Manoukian, B. Honary, " BCJR trellis construction for binary linear Block codes" *IEE proceedings, vol.144, Issue:6, 1997, pages 367 -371*

[61] A. Richardson, " WCDMA design handbook"

[62] R. Tanner, J. Woodard, "WCDMA requirements and practical design"

[63] J.K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis" *IEEE transaction on information theory, vol 24, Issue: 1, 1978, pages 76 -80.*

[64] R.J. McEliece," On the BCJR trellis for linear block codes" *IEEE Transaction on Information theory, vol 42, Issue: 4, 1996, pages 1072 -1092*

[65] F. Zesong, W. K. Jingming, " Improved binary Turbo Coded modulation with 16QAM in HSDPA" *in proceedings of wireless communications and networking, WCNC 2003, IEEE. Vol.1 pages 322 – 325.*

[66] J.G. Proakis, " Digital communications"

[67] Z. Peterson, "Introduction to digital communications"

[68] J. Hagenauer, " The EXIT chart – Introduction to extrinsic information transfer in iterative processing" *in proceedings of the 12th Europ.signal processing conference, 2004, pages 1541 – 1548.*

[69] T.K Moon, "Error correction coding mathematical methods and algorithms"

[70] B. Vucetic, J. Yuan, "Turbo Codes: principles and applications"

[71] J.J. Boutros, "Asymptotic behaviour study of irregular Turbo Codes" *in proceedings of the 7th International workshop on digital signal processing techniques for space communications, Sesimbra, Portugal, October 2001.*

[72] J. Boutros, G. Caire, E. Viterbo, H. Sawaya, S. Vialle, "Turbo Code at 0.03dB from capacity limit codes" *in proceedings of IEEE international symposium on Information theory, 2002.*

[73] V.D. Trajkovic, M. Fu, P.J. Schreier, " Turbo equalization with irregular Turbo Codes" *in proceedings of 4th international symposium on wireless communication systems, ISWCS 2007, pages 153 – 157.*

[74] A. Huebner, J. Freudenberger, R. Jordan, M. Bossert, "Irregular turbo codes and unequal error protection" *in proceedings of international symposium on information theory, 2001.*

[75] X. Jasper, L. Vandendorpe, " Joint source channel codes based on irregular Turbo Codes and variable length codes" *IEEE transaction on communications, vol 56. Issue: 11, 2008, pages 1824 -1835.*

[76] M. Tuchler, J. Hagenauer, "EXIT charts of Irregular codes" *in proceedings of Conference on Information Science and systems. Princeton University, March 2002.*

[77] M.C. Valenti, J. Sun, "The UMTS Turbo Code and an efficient decoder Implementation suitable for software-defined Radios", *International Journal Of Wireless Information Networks, Vol. 8, No. 4, October 2001*

[78] M.C. Valenti "An efficient Software Radio Implementation of the UMTS Turbo codec",

[79] M.C. Valenti "Turbo Codes" *Handbook of RF and wireless techniologies, pages 357 to 400*

[80] T. Iliev, "A study of Turbo Codes for UMTS third generation cellular Standard", *International Conference on Computer systems and Technologies- CompSys 2007*

[81] R. Zhou; A. Picart, R. Pyndiah, A. Goalic, "Potential Applications of Low complexity non-binary high code rate Block Turbo Code" *In proceedings IEEE Military Communications conference, 2004, MILCOM*

[82] T.S. Rappaport, *Wireless Communications: Principles and practice*, Second Edition, Prentice Hall, 2002.

[83] Hanzo, L; T.H. Liew, B.L. Yeap, "Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Wireless Channels.

[84] R. Jain, "Channel Models, a Tutorial", *February 2007.*

Publications

[1] A. O Sholiyi, B. Badic, R. Joyce, T. O'Farrell, "A Characterisation of HSDPA using EXIT chart technique" *IEEE 3rd Int. Conf. on Next Generation Mobile Applications, Services and Technologies, Cardiff, Sept. 2009.*

Submitted

[2] A.O Sholiyi, T. O'Farrell, "Irregular Block Turbo Code with low complexity" *IEEE Transaction on Communication.* Manuscript identification number TCOM-TPS-11-0292.

[3] A.O Sholiyi, T. O'Farrell, "Near Capacity Irregular Turbo Code" *EURASIP Journal on Wireless Communications and Networking, November 2011.*