

Efficient Iterative Decoding Algorithms for Turbo and Low-Density Parity-Check (LDPC) Codes

Stylios Papaharalabos

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey



Centre for Communications Systems Research (CCSR)
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

December 2005

© Stylios Papaharalabos 2005

Summary

Turbo and Low-Density Parity-Check (LDPC) codes are among the two most significant advances in channel coding over the recent years. Their astonishing bit error rate (BER) performance compared to moderate decoding complexity has enabled coding theorists to design practical codecs that can perform within a few tenths of a decibel to the channel capacity limit, exactly as *Shannon* had predicted more than fifty years ago. This research work has been motivated by the recent application of turbo and LDPC codes in many *satellite* standardisation committees, such as ETSI (S-UMTS, DVB-RCS/S2) and NASA (CCSDS), and also in some practical satellite systems, such as INMARSAT (BGAN) and EUTELSAT (Skypex). The use of this kind of capacity-approaching codes is crucial to the power savings of the satellite, which in this case benefits by increasing, for example, the overall system capacity and area of coverage.

In this thesis, we focus our attention on the *decoder design*. This is because, although the encoder is specified by the existing standards and systems, decoding algorithms are left open to the receiver designer. *Efficient* iterative decoding algorithms are proposed that can be applied to the general case of turbo and LDPC codes. It is shown that improvements to the BER performance of these codes are feasible with either the same or reasonable increase in the decoding complexity. In other cases, small BER performance degradation can be observed but with decoding complexity savings. Classical turbo codes (in binary form), duo-binary turbo codes (such as in the DVB-RSC standard) and regular binary LDPC codes are investigated. Mostly binary phase shift keying (BPSK) or quadrature phase shift keying (QPSK) modulation is assumed over either the additive white Gaussian (AWGN) channel or sometimes over an uncorrelated Rayleigh/Rician fading channel.

The original thesis contributions are summarised as following. First, the error floor of the improved SOVA turbo decoder is removed to lower BER values when considering binary turbo codes. This is done by using a simple correcting factor of the extrinsic information that has two steps. Then, the \max/\max^* operation replacement method, already known for the case of improved SOVA turbo decoding, is extended to either the forward/backward recursion or the soft-output computation with Max-Log-MAP and Log-MAP turbo decoding. Good trade-off between BER performance and decoding complexity is observed for both binary and duo-binary turbo codes. This is followed by a novel method of applying the \max/\max^* operation to different *levels*. Similar trade-off for binary turbo codes with Max-Log-MAP and Log-MAP turbo decoding is observed. Furthermore, the Constant Log-MAP decoding for duo-binary turbo codes is improved compared to another previously known algorithm, which has approximately the same computational complexity. In this case, the resulting algorithm can perform very close to Log-MAP turbo decoding. Finally, two modifications to the sum-product algorithm are proposed to reduce the error floor of LDPC codes to lower BER values. Piecewise linear function approximation and quantization table are also applied to reduce further the computational complexity of the hyperbolic tangent ($\tanh(x)$) and inverse (arc) hyperbolic tangent ($\tanh^{-1}(x)$) functions respectively, which are used in check-node update computation.

Key words: Extrinsic information, iterative decoding, low-density parity-check codes and turbo codes.

Email: S.Papaharalabos@surrey.ac.uk (alt. spapaha@ieee.org)

WWW: <http://www.ee.surrey.ac.uk/ccsr>

Acknowledgements

I would like to thank my first supervisor Dr. Peter Sweeney, for the weekly meetings we have had during the three years of my research work, the various suggestions and comments on technical issues he has provided to me and for being able to review all the submitted papers and also my thesis in a very fast response. I am really grateful to the kind of supervision I have received from him.

I am also grateful to my second supervisor Professor Barry G. Evans for the economic support through the EPSRC (UK) sponsorship I have received, until the submission of my thesis. Also, my involvement to the SatNEx EU Project would not happen without his encouragement.

During my involvement to this Project, I had the opportunity to spend two and half months, through a personnel exchange program, at University of Bologna, Italy. I would like to thank my second supervisor for his initial decision. It was a real challenge for me and the experience I have gained on LDPC codes is really great. My kindly thanks to Mr. Massimo Neri, Mr. Gianni Albertazzi, Dr. Alessandro Vanelli-Coralli and Professor Giovanni E. Corazza, for the collaboration and personal interest they have shown to me, since the first day of my arrival in Bologna on October 1, 2004. I think a successful joint task was completed in overall.

Special thanks to Professor Takis P. Mathiopoulos from the National Observatory of Athens, Greece who has shown a particular interest and willing to contribute actively to my research work as soon as I have been involved in the SatNEx EU Project. I am really grateful to him for his personal experience and useful suggestions on our jointly performed research work. Also, my kindly thanks to Dr. George Karagiannidis from the Aristotle University of Thessaloniki, Greece who has been welcome to any future collaboration.

I would also like to thank all the past and present CCSR Secretaries (Mrs. Stephanie J. Evans, Mrs. Liz James, Mrs. Emannuelle Darut and Mrs. Anne Rubin), the Computing Support Staff (Mr. Adam Kirby, Mr. Chris Clack and Miss Hannah Pretifer), the Finance Support Staff (Mr. Andrew Johnson, Mr. Neil Crosswell and Mr. Martyn Simmons), the Technician Mr. Terry Roberts and finally the School Postgraduate Office (Miss Kelly D. Green and Ms. Amanda Ellis).

During the three years of research work, I really appreciate any kind of discussion and help I have received from all my fellows at CCSR. Among them, I would like to thank Dr. Atta Quddus and Mr. Sadeh Fazel for issues related to the computer simulator. Special thanks to Dr. Roshano S. Roberts for the provision of some useful modifications to this \LaTeX file. I would also like to acknowledge Mr. Michael Georgiades and Mr. Qinlin Luo, who have been among my best friends at CCSR.

At the end, and the most important of all, I am grateful to my parents, Panagiotis and Eirini and to my brother, Dimitris for being always next to me, whenever I needed their priceless help and support. They have also pursued me to study for a PhD degree, after completing an MRes degree at CCSR. Let my research work progress over the last three years to be proved as the best example of the respect I pay to them.—

*Αφιερωμένο στους γονείς μου,
Παναγιώτη και Ειρήνη*

(Dedicated To My Parents, Panagiotis and Eirini)

Contents

List of Acronyms	ix
List of Figures	xiii
List of Tables	xxii
1 Introduction	1
1.1 Background	1
1.2 Research Motivation and Objectives	3
1.3 Original Achievements and Personal Contributions	5
1.4 Thesis Outline	7
1.5 Summary	9
2 Turbo Codes, LDPC Codes and Iterative Decoding for Satellite Applications	11
2.1 Satellite Communications Aspects	11
2.1.1 A Brief History of Satellite Communications	12
2.1.2 Some Successful Satellite Systems and Research Projects	14
2.2 Elements from Information Theory	17
2.2.1 Limits to Channel Capacity	17
2.2.2 Error Control Coding	18
2.2.3 A Brief History of Coding Schemes	19
2.3 Soft-Input Soft-Output (SISO) Decoding Based on Trellises	21
2.3.1 SOVA Decoding	24
2.3.2 MAP Decoding	25
2.3.3 Log-MAP and Max-Log-MAP Decoding	27
2.3.4 Decoding Complexity Comparison	28

2.4	Binary Turbo Codes	29
2.4.1	Binary Turbo Encoder	30
2.4.2	Binary Turbo Decoder	32
2.4.3	Binary Turbo Performance Example	33
2.5	Duo-Binary Turbo Codes	34
2.5.1	Duo-Binary Turbo Encoder	35
2.5.2	Duo-Binary Turbo Decoder	37
2.5.3	Duo-Binary Turbo Performance Example	39
2.6	LDPC Codes	40
2.6.1	Factor Graphs	41
2.6.2	LDPC Encoding	43
2.6.3	LDPC Decoding	44
2.6.4	LDPC Performance Example	47
2.7	Computer Simulation Environment	48
2.7.1	Computer Simulated Performance Validation	51
2.7.2	Computer Simulated Performance Comparison	53
2.8	Summary	63
3	Improved SOVA Decoding for Binary Turbo Codes	64
3.1	Introduction	64
3.2	SOVA Turbo Decoder Implementations	65
3.3	Relevant Work on Improved SOVA Turbo Decoder	66
3.3.1	Fundamental Approaches	66
3.3.2	Latest Research Work	70
3.4	Proposed Method	72
3.4.1	Motivation	73
3.4.2	Simple Two-Step Approach	73
3.4.3	Modified Branch Metrics	74
3.5	Computer Simulation Results	75
3.5.1	Best Parameter Values	75
3.5.2	Simulation Performance Comparison	80
3.5.3	More Computer Simulation Results	83
3.5.4	Discussion	86
3.6	Summary	88

4	Improved Max-Log-MAP and Log-MAP Decoding for Binary Turbo Codes	90
4.1	Introduction	90
4.2	Relevant Work on Improved Max-Log-MAP and Log-MAP Turbo Decoder	91
4.3	SISO Algorithms Based on Max/Max* Operation Replacement for Turbo Decoding	94
4.3.1	Motivation	94
4.3.2	Proposed SISO Decoding Algorithms and Complexity Estimation	95
4.3.3	Computer Simulation Results	98
4.4	SISO Algorithms Based on the Application of Max/ Max* Operation on <i>Levels</i> for Turbo Decoding	104
4.4.1	Motivation	104
4.4.2	Proposed SISO Decoding Algorithms and Complexity Estimation	105
4.4.3	Computer Simulation Results	111
4.5	Summary	118
5	Improved Decoding Algorithms for Duo-Binary Turbo Codes	120
5.1	Introduction	120
5.2	DVB-RCS Standard	122
5.3	Relevant Work on Duo-Binary Turbo Codes and Related DVB-RCS Standard Improvements	125
5.4	SISO Algorithms Based on Max/Max* Operation Replacement for DVB-RCS Turbo Code	128
5.4.1	Motivation	128
5.4.2	Proposed SISO Decoding Algorithms and Complexity Estimation	128
5.4.3	Computer Simulation Results	130
5.5	Efficient Constant Log-MAP Decoding Algorithm for DVB-RCS Turbo Code	135
5.5.1	Motivation	136
5.5.2	Proposed Decoding Algorithm and Complexity Estimation . . .	136
5.5.3	Computer Simulation Results	140
5.6	Summary	145

6	Improved Decoding Algorithms for LDPC Codes	148
6.1	Introduction	148
6.2	Relevant Work on Optimum and Reduced Complexity Decoding Algorithms for LDPC Codes	149
6.3	Modified <i>tanh</i> Function in Sum-Product Algorithm for Decoding LDPC Codes	156
6.3.1	Motivation	156
6.3.2	Proposed Method	157
6.3.3	Decoding Complexity Reductions	159
6.3.4	Computer Simulation Results	161
6.4	Modified <i>inverse</i> (arc) <i>tanh</i> Function in Sum-Product Algorithm for Decoding LDPC Codes	168
6.4.1	Motivation	168
6.4.2	Proposed Method	168
6.4.3	Decoding Complexity Reductions	170
6.4.4	Computer Simulation Results	173
6.5	Summary	179
7	Conclusions	182
7.1	Research Work Summary and Contribution	182
7.2	Suggested Future Research Work	186
A	Publications List	190
B	Turbo and LDPC Codes Computer Simulated Performance Validation	192
C	The Effect of Different Parameters to the Simulated Turbo Code Performance	204
	References	213

List of Acronyms

A

ADSL	Asymmetric Digital Subscriber Line
APP	A Posteriori Probability
ASK	Amplitude Shift Keying
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise

B

BCH	Bose-Chaudhuri-Hocquenghem
BEC	Binary Erasure Channel
BER	Bit Error Rate
BGAN	Broadband Global Area Network
BPSK	Binary Phase Shift Keying

C

CC	Convolutional Code
CCSDS	Consultative Committee for Space Data Systems
CDMA	Code Division Multiple Access
CSI	Channel State Information

D

DMB	Digital Multimedia Broadcast
DRP	Dithered Relative Prime
DSP	Digital Signal Processing
DVB	Digital Video Broadcasting
DVB-H	Digital Video Broadcasting for Handheld terminals

DVB-RCS	Digital Video Broadcasting Return Channel over Satellite
DVB-RCT	Digital Video Broadcasting Return Channel over Terrestrial
DVB-S	Digital Video Broadcasting over Satellite
DVB-S2	Second Generation Digital Video Broadcasting over Satellite

E

E_b/N_0	Bit Energy to Noise Power Spectral Density
ESA	European Space Agency
ETSI	European Telecommunications Standards Institute
EUTELSAT	EUropean TELecomunications SATellite organisation
EXIT	EXtrinsic Information Transfer

F

FEC	Forward Error Correcting
FER	Frame Error Rate
FP	Framework Program
FPGA	Field-Programmable Gate Array

G

GAN	Global Area Network
GEO	Geostationary Earth Orbit
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications

H

HAPS	High Altitude Platforms
HCCC	Hybrid Concatenated Convolutional Codes
HEO	Highly Elliptic Orbit

I

INMARSAT	INternational MARitime SATellite organisation
INTELSAT	INternational TELecomunications SATellite organisation
IP	Internet Protocol
ITU	International Telecommunication Union

J

JPL Jet Propulsion Laboratory

L

LDPC Low-Density Parity-Check

LEO Low Earth Orbit

LLR Log-Likelihood Ratio

LLR-SPA Log-Likelihood Ratio Sum-Product Algorithm

LUT Look-Up Table

M

MAP Maximum A posteriori Probability

MBCO Mobile Broadcasting Corporation

MBMS Multimedia Broadcast/Multicast Service

MEO Medium Earth Orbit

MIMO Multiple Input Multiple Output

MLD Maximum Likelihood Decoding

MPEG Moving Picture Experts Group

N

NASA National Aeronautics and Space Administration

P

PCCC Parallel Concatenated Convolutional Codes

PEG Progressive Edge Growth

PSK Phase Shift Keying

Q

QPSK Quadrature Phase Shift Keying

R

RA Repeat-Accumulate

RMSE Root Mean Square Error

RS Reed-Solomon

RSC	Recursive Systematic Convolutional
S	
SCCC	Serial Concatenated Convolutional Codes
S-DAB	Digital Audio Broadcasting Via Satellite
S-DMB	Satellite Digital Multimedia Broadcast
SISO	Soft-Input Soft-Output
SNR	Signal-to-Noise Ratio
SOVA	Soft Output Viterbi Algorithm
(Bi)-SOVA	Bidirectional Soft Output Viterbi Algorithm
(BR)-SOVA	Soft Output Viterbi Algorithm from <i>Battail</i>
(HR)-SOVA	Soft Output Viterbi Algorithm from <i>Hagenauer</i>
SPA	Sum-Product Algorithm
S-UMTS	Satellite Universal Mobile Telecommunication System
T	
TCM	Trellis Coded Modulation
T-UMTS	Terrestrial Universal Mobile Telecommunication System
U	
UMTS	Universal Mobile Telecommunication System
V	
VA	Viterbi Algorithm
VLSI	Very Large Scale Integration
W	
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
1G	First Generation
2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Project

List of Figures

1.1	Thesis organisation.	8
2.1	<i>Shannon's</i> capacity limit for the AWGN channel from <i>Vucetic</i> [1]. The achieved spectral efficiency of various modulation and coding schemes is also shown.	18
2.2	LLR values used in a SISO decoder.	23
2.3	PCCC scheme.	30
2.4	Typical turbo encoder.	31
2.5	Turbo decoder.	32
2.6	Turbo code performance from the JPL web site [2]. Frame size 16384 bits, different coding rates and number of decoding iterations. For coding rate $R = 1/2$, an asymmetrical turbo code is considered with lower complexity.	34
2.7	Duo-binary constituent RSC encoder.	35
2.8	Trellis diagram of duo-binary constituent RSC encoder.	36
2.9	Duo-binary turbo code FER performance from <i>Berrou</i> [3]. 16-states encoder, different coding rates, QPSK modulation, AWGN channel, improved Max-Log-MAP algorithm, 4 bits quantization and 8 decoding iterations. Solid lines-simulation and dashed lines-theoretical limits. (a) ATM frame size, i.e. 424 bits, (b) MPEG frame size, i.e. 1504 bits. . . .	40
2.10	<i>Tanner</i> graph example.	42
2.11	BER performance of different LDPC codes (i.e. regular, irregular, binary and non-binary) and comparison to turbo and convolutional codes, as from <i>Soleymani</i> [4]. Coding rate $R = 1/4$ over the AWGN channel. . .	47
2.12	Computer simulation chain.	49
2.13	BER/FER comparison between binary (solid lines) turbo code with generator polynomials $(1, 13/15)_o$, i.e. 8-states, and duo-binary turbo code (dashed lines), such as in the DVB-RCS standard. ATM frame size, i.e. 424 bits, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations and different coding rates.	55

2.14	BER/FER comparison between binary (solid lines) turbo code with generator polynomials $(1, 13/15)_o$, i.e. 8-states, and duo-binary turbo code (dashed lines), such as in the DVB-RCS standard. MPEG frame size, i.e. 1504 bits, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations and different coding rates.	56
2.15	BER/FER comparison between $(96, 48)$ LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from <i>Gallager's</i> approach and either maximum 10 or 200 decoding iterations. Turbo code, 48 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.	57
2.16	BER/FER comparison between $(504, 252)$ LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from <i>Gallager's</i> approach and either maximum 10 or 50 decoding iterations. Turbo code, 252 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.	58
2.17	BER/FER comparison between $(1008, 504)$ LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from <i>Gallager's</i> approach and either maximum 10 or 80 decoding iterations. Turbo code, 504 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.	59
2.18	BER/FER comparison between $(8000, 4000)$ LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from <i>Gallager's</i> approach and either maximum 10 or 200 decoding iterations. Turbo code, 4000 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.	60
2.19	BER/FER comparison between $(816, 408)$ LDPC code (solid lines), binary turbo code (dashed lines) with different generator polynomials and duo-binary turbo code (dashed-dotted line), such as in the DVB-RCS standard. LDPC code, SPA decoding algorithm from <i>Gallager's</i> approach and either maximum 10 or 200 decoding iterations. Turbo code, 408 bits frame size, Log-MAP algorithm and 10 decoding iterations. Duo-binary turbo code, ATM frame size, i.e. 424 bits, Log-MAP algorithm and 8 decoding iterations. In all cases, coding rate $R = 1/2$ and the AWGN channel.	61

2.20	BER/FER comparison between (4000, 2000) LDPC code (solid lines), binary turbo code (dashed lines) with different generator polynomials and duo-binary turbo code (dashed-dotted line), such as in the DVB-RCS standard. LDPC code, SPA decoding algorithm from <i>Gallager's</i> approach and either maximum 10 or 200 decoding iterations. Turbo code, 2000 bits frame size, Log-MAP algorithm and 10 decoding iterations. Duo-binary turbo code, MPEG frame size, i.e. 1504 bits, Log-MAP algorithm and 8 decoding iterations In all cases, coding rate $R = 1/2$ and the AWGN channel.	62
3.1	Improved (normalised) SOVA turbo decoder.	67
3.2	BR/HR-SOVA graphical comparison from <i>Lin et al</i> [5].	74
3.3	Impact of the parameter x_0 to the turbo code BER performance for different E_b/N_o values. $(1, 15/13)_o$ turbo encoder, coding rate $R=1/3$, 1000 bits frame size, <i>norm1</i> SOVA algorithm and 8 decoding iterations in the AWGN channel.	77
3.4	Normalised SOVA (solid lines) and Log-MAP (dashed lines) iterative decoding performance comparison for different generator polynomials, coding rate $R=1/3$, 1000 bits frame size and 8 decoding iterations in the AWGN channel.	78
3.5	Correlation coefficient between intrinsic and extrinsic information of the second decoder against the E_b/N_o value, using standard SOVA and <i>norm1</i> SOVA. Turbo encoder $(1, 15/13)_o$, coding rate $R=1/3$, 1000 bits frame size in the AWGN channel. (a) 2 decoding iterations, (b) 8 decoding iterations respectively.	80
3.6	BER performance comparison of different normalised iterative SOVA algorithms. Coding rate $R=1/3$, 1000 bits frame size and 8 decoding iterations in the AWGN channel.	81
3.7	BER performance of different turbo encoders using <i>norm1</i> SOVA (solid lines) and reference performance comparison (dashed lines). Coding rate $R=1/2$, 1000 bits frame size and 8 decoding iterations in the AWGN channel.	82
3.8	BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 5114 bits frame size, 3GPP interleaver and 18 decoding iterations in the AWGN channel.	84
3.9	BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 10000 bits frame size and 18 decoding iterations in the AWGN channel.	85
3.10	BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 65536 bits frame size and 18 decoding iterations in the AWGN channel.	86

3.11	BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 10000 bits frame size and 18 decoding iterations in an uncorrelated Rayleigh/Rician fading channel.	87
4.1	BER performance comparison of SISO decoding algorithms based on max/max^* operation replacement. Turbo code generator polynomials $(1,5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.	99
4.2	BER performance comparison of SISO decoding algorithms based on max/max^* operation replacement. Turbo code generator polynomials $(1,21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.	100
4.3	BER performance comparison of SISO decoding algorithms based on max/max^* operation replacement. Turbo code generator polynomials $(1,5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.	101
4.4	BER performance comparison of SISO decoding algorithms based on max/max^* operation replacement. Turbo code generator polynomials $(1,21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.	102
4.5	BER performance comparison of SISO decoding algorithms based on different levels of max/max^* operation. Turbo code generator polynomials $(1,5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.	112
4.6	BER performance comparison of SISO decoding algorithms based on different levels of max/max^* operation. Turbo code generator polynomials $(1,21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.	113
4.7	BER performance comparison of SISO decoding algorithms based on different levels of max/max^* operation. Turbo code generator polynomials $(1,5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.	114
4.8	BER performance comparison of SISO decoding algorithms based on different levels of max/max^* operation. Turbo code generator polynomials $(1,21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.	115
5.1	DVB-RCS turbo encoder.	123
5.2	BER performance comparison of SISO decoding algorithms based on max/max^* operation replacement. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.	131

5.3	FER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.	132
5.4	BER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.	133
5.5	FER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.	134
5.6	BER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.	141
5.7	FER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.	142
5.8	BER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.	143
5.9	FER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.	144
6.1	The effect to the BER performance when approximating the \tanh function with different values. BER with no approximation is shown in dashed lines. (1008,504) LDPC code, coding rate $R = 1/2$, AWGN channel and maximum 80 decoding iterations.	158
6.2	Example of \tanh function (continuous-circle line) and approximations with piecewise linear function (dashed line) and quantization (constant function-solid line).	160
6.3	BER performance with or without modification to the \tanh function and comparison to Gallager's approach (dashed lines). Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.	163
6.4	BER performance with or without modification to the \tanh function and comparison to Gallager's approach (dashed lines). Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel.	164

6.5	BER performance with modified \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.	165
6.6	BER performance with modified \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel.	166
6.7	The effect to the BER performance when approximating the inverse \tanh function with different values. BER with no approximation is shown in dashed lines. (1008, 504) LDPC code, coding rate $R = 1/2$, AWGN channel and maximum 80 decoding iterations.	170
6.8	Example of inverse (arc) \tanh function (continuous-circle line) and approximations with piecewise linear function (dashed line) and quantization (constant function-solid line).	172
6.9	BER performance with or without modification to the inverse \tanh function and comparison to <i>Gallager's</i> approach (dashed lines) and modified \tanh function. Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.	175
6.10	BER performance with or without modification to the inverse \tanh function and comparison to <i>Gallager's</i> approach (dashed lines) and modified \tanh function. Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel.	176
6.11	BER performance with modified inverse \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.	177
6.12	BER performance with modified inverse \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel. . . .	178
B.1	BER comparison with <i>Berrou</i> [6]. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 65536 bits frame size, AWGN channel and different number of decoding iterations. Solid lines-simulation (Log-MAP algorithm) and dashed lines-from reference (MAP algorithm).	195
B.2	BER comparison with <i>Robertson</i> [7]. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, AWGN channel, 8 decoding iterations and different frame size. Solid lines-simulation (Log-MAP algorithm) and dashed lines-from reference (MAP algorithm).	195

B.3	BER comparison with <i>Hanzo</i> [8]. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, 1000 bits frame size, AWGN channel, Log-MAP algorithm, 8 decoding iterations and different coding rate. Solid lines-simulation and dashed lines-from reference.	196
B.4	BER comparison with <i>Valenti</i> [9]. Turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 3GPP interleaver, AWGN/uncorrelated Rayleigh fading channel, Log-MAP algorithm and different frame size/number of decoding iterations. Solid lines-simulation (with no CSI in fading) and dashed lines-from reference (with CSI in fading).	196
B.5	BER comparison with <i>Robertson</i> [7]. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 1000 bits frame size, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations. Solid lines-simulation and dashed lines-from reference (with 1024 bits frame size and optimised interleaver).	197
B.6	BER comparison with <i>Hanzo</i> [8]. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, 1000 bits frame size, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations and different coding rate. Solid lines-simulation and dashed lines-from reference.	197
B.7	BER comparison with <i>Valenti</i> [9]. Turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 3GPP interleaver, AWGN/uncorrelated Rayleigh fading channel, Max-Log-MAP algorithm and different frame size/number of decoding iterations. Solid lines-simulation (no CSI in fading) and dashed lines-from reference (with CSI in fading).	198
B.8	BER comparison with <i>Hanzo</i> [8]. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, 1000 bits frame size, AWGN channel, SOVA algorithm, 8 decoding iterations and different coding rate. Solid lines-simulation and dashed lines-from reference.	198
B.9	BER comparison with <i>Hagenauer</i> [10]. Coding rate $R = 1/2$, frame size 400 bits, AWGN/uncorrelated Rayleigh fading channel, SOVA algorithm, 8 decoding iterations and different turbo code generator polynomials. Solid lines-simulation (no CSI in fading) and dashed lines-from reference (with CSI in fading).	199
B.10	BER comparison with <i>Hagenauer</i> [10]. Coding rate $R = 1/2$, frame size 1000 bits, AWGN/uncorrelated Rayleigh fading channel, SOVA algorithm, 8 decoding iterations and different turbo code generator polynomials. Solid lines-simulation (no CSI in fading) and dashed lines-from reference (with CSI in fading).	199
B.11	FER comparison with <i>Berrou</i> [3]. Duo-binary turbo code, such as in the DVB-RCS standard, ATM frame size, i.e. 424 bits, AWGN channel, 8 decoding iterations and different coding rates. Solid lines-simulation (Max-Log-MAP and Log-MAP algorithms) and dashed lines-from reference (improved Max-Log-MAP algorithm).	200

B.12 FER comparison with <i>Berrou</i> [3]. Duo-binary turbo code, such as in the DVB-RCS standard, MPEG frame size, i.e. 1504 bits, AWGN channel, 8 decoding iterations and different coding rates. Solid lines-simulation (Max-Log-MAP and Log-MAP algorithms) and dashed lines-from reference (improved Max-Log-MAP algorithm).	200
B.13 BER/FER comparison with <i>Kabal</i> [11]. Duo-binary turbo code, such as in the DVB-RCS standard, coding rate $R = 1/3$, ATM frame size, i.e. 424 bits, AWGN channel, Max-Log-MAP, Log-MAP algorithms and 8 decoding iterations. Solid lines-simulation and dashed lines-from reference.	201
B.14 BER/FER comparison with <i>Kabal</i> [11]. Duo-binary turbo code, such as in the DVB-RCS standard, coding rate $R = 1/3$, MPEG frame size, i.e. 1504 bits, AWGN channel, Max-Log-MAP, Log-MAP algorithms and 8 decoding iterations. Solid lines-simulation and dashed lines-from reference.	201
B.15 BER/FER comparison with <i>Yu</i> [12]. Duo-binary turbo code, such as in the DVB-RCS standard, MPEG frame size, i.e. 1504 bits, AWGN channel, Max-Log-MAP algorithms, 8 decoding iterations and different coding rates. Solid lines-simulation and dashed lines-from reference. . .	202
B.16 BER/FER comparison with <i>MacKay</i> [13]. Different regular LDPC codes, coding rate $R = 1/2$, AWGN channel, SPA decoding algorithm from <i>Gallager's</i> approach and maximum 200 decoding iterations. Solid lines-simulation and dashed lines-from reference (with variable maximum number of decoding iterations).	202
B.17 BER comparison with <i>Fossorier-1</i> [14], <i>Eleftheriou</i> [15] and <i>Fossorier-2</i> [16]. Different regular LDPC codes and maximum number of decoding iterations, coding rate $R = 1/2$, AWGN channel, SPA decoding algorithm from <i>Gallager's</i> approach. Solid lines-simulation and dashed lines-from reference.	203
C.1 Effect of <i>number of decoding iterations</i> to the BER performance, turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 65536 bits frame size, AWGN channel and Log-MAP algorithm. . .	206
C.2 Effect of <i>number of decoding iterations</i> to the BER performance, turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 65536 bits frame size, AWGN channel and Log-MAP algorithm (<i>zoom</i>).	207
C.3 Effect of <i>number of decoding iterations</i> to the BER performance, turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/3$, 65536 bits frame size, AWGN channel and <i>norm1</i> SOVA algorithm.	207
C.4 Effect of <i>frame (or interleaver) size</i> to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, AWGN channel, Log-MAP algorithm and 8 decoding iterations. BER performance with 256-states convolutional code is also shown from [8] (dashed lines).	208

C.5	Effect of <i>frame (or interleaver) size</i> to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, AWGN channel, Max-Log-MAP algorithm and 8 decoding iterations. BER performance with 128-states convolutional code is also shown from [8] (dashed lines).	208
C.6	Effect of <i>memory order</i> to the BER performance, coding rate $R = 1/2$, 1000 bits frame size, AWGN channel, Log-MAP algorithm and 8 decoding iterations.	209
C.7	Effect of the <i>type of interleaver</i> to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, either 500 or 1440 bits frame size, AWGN channel, Log-MAP algorithm and 8 decoding iterations. Solid lines-random interleaver, dashed lines-3GPP interleaver.	209
C.8	Effect of the <i>type of interleaver</i> to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, either 500 or 1440 bits frame size, AWGN channel, Max-Log-MAP algorithm and 8 decoding iterations. Solid lines-random interleaver, dashed lines-3GPP interleaver.	210
C.9	Effect of <i>puncturing</i> to the BER performance, different turbo code generator polynomials, 1000 bits frame size, AWGN channel, Log-MAP algorithm and 8 decoding iterations. Solid lines-no puncturing, i.e. coding rate $R = 1/3$, dashed lines-with puncturing, i.e. coding rate $R = 1/2$. . .	210
C.10	Effect of <i>puncturing</i> to the BER performance, DVB-RCS turbo encoder, i.e. 8-states, ATM frame size, i.e. 424 bits, AWGN channel, Max-Log-MAP algorithm and 8 decoding iterations. Coding rates $R = 1/3, 1/2$ are with no puncturing.	211
C.11	Effect of <i>channel type</i> to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 1000 bits frame size, <i>norm2</i> SOVA algorithm and 8 decoding iterations.	211
C.12	Effect of <i>decoding algorithm</i> to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 1000 bits frame size, AWGN channel and 8 decoding iterations.	212
C.13	Effect of <i>decoding algorithm</i> to the BER performance, turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, coding rate $R = 1/2$, 1000 bits frame size, AWGN channel and 8 decoding iterations.	212

List of Tables

2.1	Reference work on binary turbo codes for BER performance validation.	51
2.2	Reference work on duo-binary turbo codes for BER/FER performance validation.	52
2.3	Reference work on LDPC codes for BER/FER performance validation. .	52
3.1	Best found values of scaling factor (Z) and modified branch metric (x_0), assuming two coding rates (R) and different turbo code generator polynomials, using <i>norm1</i> SOVA.	76
3.2	Required E_b/N_o value at BER of 10^{-4} using <i>norm1</i> SOVA, Max-Log-MAP and Log-MAP algorithms, 1000 bits frame size, 8 decoding iterations in the AWGN channel. Different turbo code generator polynomials are assumed with coding rate either $R=1/3$ or $R=1/2$	79
3.3	Correlation coefficient between intrinsic and extrinsic information of the second decoder against the E_b/N_o value, using the standard SOVA, <i>norm1/norm2</i> SOVA. Turbo encoder $(1, 15/13)_o$, coding rate $R=1/3$, 1000 bits frame size in the AWGN channel and 8 decoding iterations. . .	79
4.1	Proposed SISO decoding algorithms; operation and notation.	96
4.2	Decoding complexity estimation of SISO decoding algorithms based on <i>max/max*</i> operation replacement. M is the turbo encoder memory order.	96
4.3	Relative decoding complexity comparison of SISO decoding algorithms based on <i>max/max*</i> operation replacement with respect to Max-Log-MAP and Log-MAP turbo decoder.	97
4.4	Relative decoding complexity comparison example of SISO decoding algorithms based on <i>max/max*</i> operation replacement with respect to Max-Log-MAP and Log-MAP turbo decoder.	103
4.5	Example of three levels of the <i>max*</i> operator applied to eight arguments.	106
4.6	Numerical example of all the possible combinations of the <i>max/max*</i> operation applied to four arguments.	107
4.7	Decoding complexity estimation of SISO decoding algorithms based on different levels of <i>max/max*</i> operation. M is the turbo encoder memory order.	109

4.8	Relative decoding complexity comparison of SISO decoding algorithms based on different levels of \max/\max^* operation with respect to Max-Log-MAP and Log-MAP turbo decoder.	110
4.9	Relative decoding complexity comparison example of SISO decoding algorithms based on different levels of \max/\max^* operation with respect to Max-Log-MAP and Log-MAP turbo decoder.	116
5.1	Decoding complexity estimation of SISO decoding algorithms based on \max/\max^* operation replacement. It is assumed a binary turbo encoder with memory order equal to three.	129
5.2	Relative decoding complexity comparison example of SISO decoding algorithms with respect to Max-Log-MAP and Log-MAP turbo decoder. It is assumed a binary turbo encoder with memory order equal to three.	130
5.3	Overall complexity estimation of one constant Log-MAP operation for binary turbo codes.	137
5.4	Overall complexity estimation of one Type-I Constant Log-MAP operation for duo-binary turbo codes.	138
5.5	Overall complexity estimation summary of one Constant Log-MAP operation for duo-binary turbo codes.	139
6.1	Piecewise linear approximation of $\tanh(x)$ function.	159
6.2	Quantization table of $\tanh(x)$ function.	160
6.3	Piecewise linear approximation of $\tanh^{-1}(x)$ function.	171
6.4	Quantization table of $\tanh^{-1}(x)$ function.	171

Chapter 1

Introduction

1.1 Background

Turbo codes can achieve near *Shannon* channel capacity limit performance, after a certain number of decoding iterations and with moderate decoding complexity. In the first publication by *Berrou et al* [6] a rate 1/2 turbo code could achieve a bit error rate (BER) of around 10^{-5} at bit energy to noise power spectral density (E_b/N_o) value of 0.7 dB, assuming binary phase shift keying (BPSK) modulation and an additive white Gaussian noise (AWGN) channel. This was only 0.5 dB away from the channel capacity limit. Coding experts, although quite skeptical at the beginning, started to replicate the results and to realise later on the significance of this work [17].

In a period of more than ten years, the turbo decoding ‘principle’ has been applied to a variety of communication systems, so that nowadays decoding is jointly performed with other similar processes based on soft information exchange. Such examples are turbo synchronisation, turbo equalisation, turbo multi-user detection for code division multiple access (CDMA) systems and turbo channel estimation [18]. Turbo codes have been extended to serial concatenated convolutional codes (SCCC) and hybrid concatenated convolutional codes (HCCC), which achieve near channel capacity limit performance with higher asymptotic gain at the expense of small complexity increase [19]. Another related extension with near channel capacity limit performance are turbo

product codes, which use high rate block codes and operate at higher decoding speeds [18], and also turbo multiple input multiple output (MIMO) systems [20].

A new area of research, which came out after the introduction of turbo codes, is the iterative decoding of low-density parity-check (LDPC) codes. These codes, although being proposed by *Gallager* in 1962 [21], were rediscovered by *MacKay* and *Neal* in the late 1990's [22], [23]. This is mainly because when LDPC codes were published for the first time, the technology was not ready for their practical implementation. In the meantime, a remarkable work can be found by *Tanner* in 1981 [24], in which LDPC codes were generalised and a graphical representation of them was proposed (i.e. the so-called *Tanner* graphs). An advantage of LDPC codes against turbo codes is that the decoding complexity grows linearly with the frame length, so that after a certain number of decoding iterations they perform close to the channel capacity limit. For example, as little as 0.04 dB away from it at BER of 10^{-6} using a rate 1/2 LDPC code with block length of 10^7 bits in the AWGN channel [25].

Nowadays, it is believed that any simple code that uses a large pseudo-random interleaver (i.e. *turbo-like* code) and is decoded by the *sum-product* algorithm (SPA) (i.e. an iterative decoding algorithm based on factor graphs theory) can approach the *Shannon* channel capacity limit [26]. Irregular repeat-accumulate (RA) codes are such an example [27]. The question now is how much faster a code can approach the *Shannon* channel capacity limit and also if it is, for example, 0.1 dB or 0.001 dB away from it [17].

Satellites are an important delivery mechanism for communication services over the radio interface [28, 29]. Other competitor interfaces are cable and fibre. The advantages of deploying satellite communications are wide coverage and high available bandwidth. This explains why satellites have been mostly used for fixed services, e.g. broadcasting. For instance, almost 80% of digital television in Europe is received via satellite. Moreover, new fixed satellite services are promising, like high speed Internet access. Propagation delay and attenuation due to rain are two drawbacks of geostationary earth orbit (GEO) satellites. Delay to speech services can be compensated by modern echo-cancellors. Adaptive modulation and coding techniques are used to overcome the

rain attenuation, delivering services at BER of better than 10^{-10} .

In the mobile satellite area, the International Maritime Satellite organisation (INMARSAT) has been a success for the last twenty years providing speech and low data rate services in the niche areas of sea and aeronautical coverage [28, 29]. Some other successful satellite systems, which are currently deployed in Korea/Japan and in the USA, are the provision of digital television (MBSAT) in the former case and digital radio (XM radio, Sirius) in the later case [30].

The economic failure of big investments in the mobile satellite area in the early 2000's such as Iridium, Globalstar and ICO has shown that future satellite systems should cooperate and not compete with the terrestrial mobile systems. On the other hand, GEO satellites are going to be deployed with higher power, increased number of beams and available bandwidth. That makes satellites to fit in the future visions of mobile communications as part of a hierarchical structure [28, 29]. One crucial aspect is the concept of digital multimedia broadcast (DMB) via satellite, which supports mainly non-real time applications and is integrated with the 3G terrestrial network. Another aspect is broadband non-broadcasting satellite applications with emphasis on passenger vehicles such as planes, ships and trains.

1.2 Research Motivation and Objectives

Advanced channel coding schemes, such as turbo and LDPC codes, have been adopted by many standardisation committees in the satellite area, e.g. the European Telecommunications Standards Institute (ETSI)-S-UMTS, DVB-RCS and DVB-S2 and the National Aeronautics and Space Administration (NASA)-CCSDS or practical satellite systems, e.g. INMARSAT-BGAN and European Telecommunications Satellite organisation (EUTELSAT)-Skyplex [31]. In addition, they are considered as strong candidates for updating existing standards in other areas, such as the IEEE 802.16 standard for Wireless Metropolitan Area Networks (WMANs) and the ETSI/International Telecommunication Union (ITU) standards for Asymmetric Digital Subscriber Lines (ADSLs) [31], due to the near channel capacity performance and the power savings.

Motivated by the variety of capacity-approaching coding schemes on many practical applications, we focus our attention to the decoder side. This is because, although the encoder is specified by the existing standards, decoding algorithms are left open to the receiver designer. This is decided according to different parameter requirements. For instance, the desired BER, the available signal-to-noise ratio (SNR), the bandwidth efficiency and the decoding complexity.

One of the most crucial factors in practical systems is the trade-off between decoding complexity and performance. Algorithms with reduced decoding complexity are expected to degrade the BER performance, but with practical implementation advantages. For example, in Max-Log-MAP turbo decoding the resulting BER performance is 0.5 dB inferior compared to Log-MAP turbo decoding [7]. However, the former algorithm is approximately half as complex as the latter one. Another aspect is the *error floor* that is observed in the performance of turbo codes. This is the flattening of the BER curve at high SNR values and is explained because of the relatively low minimum distance of turbo codes [1, 19]

Inspired by the advanced channel coding schemes that are being deployed in satellite communications, this research work is focused on *novel decoding algorithms*, suitable for turbo and LDPC iterative decoding. The following objectives are thus set up

- Search for decoding algorithms that reduce the code error floor at lower BER values.
- Search for decoding algorithms that improve the BER performance, but in a decoding complexity trade-off.
- Search for decoding algorithms that improve the BER performance with approximately the same decoding complexity.

In the rest of the thesis, parallel concatenated convolutional codes (PCCC) in both binary and duo-binary form are primarily investigated. Also, the rediscovered version of LDPC codes (as from *MacKay's* work) has been studied briefly, as proof of concept of the iterative decoding process. This approach is in good agreement with a recent

European Space Agency (ESA) project, called MHOMS [32], where both PCCC (in duo-binary form), SCCC and LDPC codes are investigated for future high rate satellite modems used in different applications. A more detailed description of this project is given in Section 2.1.2.

The considered type of channel is either the AWGN or an uncorrelated (or fully interleaved) Rayleigh/Rician fading. The modulation type consists of either BPSK or quadrature phase shift keying (QPSK). The resulting decoding algorithms have direct application mostly to fixed satellite communication links, where the ideal AWGN channel is assumed. Also, the uncorrelated Rayleigh/Rician fading channel is an extreme case of a mobile satellite fading channel with ideal propagation scenario conditions.

To report briefly, the first of the above objectives was achieved by using two appropriate modifications to the improved (soft-output *Viterbi* algorithm) SOVA decoding for binary turbo codes and also to the logarithmic domain sum-product algorithm (LLR-SPA) for regular binary LDPC codes. The second objective was achieved by using two appropriate modifications to the Max-Log-MAP and Log-MAP decoding for both binary and duo-binary turbo codes. The third objective, which is more promising, was achieved by using an appropriate modification to the Constant Log-MAP decoding for duo-binary turbo codes.

1.3 Original Achievements and Personal Contributions

In an approximate period of three years of research work, including writing up, eight original achievements were reached that are summarised below.

1. A normalisation scheme and a simple two-step algorithm approach to reduce the error floor of the improved SOVA turbo decoder at lower BER values, with small increase to the decoding complexity.
2. Four decoding algorithms to improve the Max-Log-MAP decoding of binary/duo-binary turbo codes, at reasonable complexity increase. This is based on the \max/\max^* operation replacement method.

-
3. The same four decoding algorithms reduce the Log-MAP decoding complexity of binary/duo-binary turbo codes, at reasonable BER performance degradation.
 4. Another M decoding algorithms, depending on the turbo encoder memory size M , to improve the Max-Log-MAP decoding of binary turbo codes, at reasonable complexity increase. This is based on the application of the \max/\max^* operation in different *levels*.
 5. The same M decoding algorithms reduce the Log-MAP decoding complexity of binary turbo codes, at reasonable BER performance degradation.
 6. An algorithm for Constant Log-MAP decoding of duo-binary turbo codes, which is found to have superior performance and the same decoding complexity as an existing one.
 7. Two decoding algorithms to reduce the error floor of regular binary LDPC codes, based on the LLR-SPA, with small increase to the decoding complexity.
 8. Two approximation methods on top of the previous approach, to reduce the decoding complexity of the LLR-SPA for LDPC codes. One of the two methods has small performance degradation, while the other one has identical or even better performance compared to no approximation.

The first year of research work was devoted to binary turbo codes with emphasis on SOVA turbo decoder, due to previous relative experience on a MRes project at the same research center (CCSR) [33]. The second year was devoted to improvements of Max-Log-MAP and Log-MAP turbo decoding, applicable to both binary and duo-binary turbo codes. Also, the Constant Log-MAP algorithm for duo-binary turbo codes was developed at the end of the same year. Less than the first half of the third year was spent on an internship exchange program at University of Bologna, Italy within the context of SatNEx EU IST Project [34]. This involved the investigation of LDPC codes for the first time. Lastly, the other part of the third year up to the present date, was spent on summarising the research work and writing up.

The overall publications contribution, included the current submissions, has come to twelve. A full publication list is given in *Appendix A*. Among them, two book chapters,

two journal publications and four conferences publications have already been achieved. Also, four more scientific papers have been submitted for publication. Furthermore, one *citation* on the research work on the improved SOVA turbo decoder was received in the Proceedings of *IEEE Globecom* 2004 and the writer acted as a *reviewer* three times. This includes peer reviewing to the *International Journal of Satellite Communications and Networking* (in February 2005), the *IEEE Communications Letters* (in May 2005) and the upcoming *IEEE Turbo Coding Conference 2006* (in November 2005).

1.4 Thesis Outline

The thesis organisation as well as the *novel* research work is shown in Fig. 1.1.

After this Introduction, Chapter 2 provides all the basic concepts in this research work. It starts with a brief history of satellite communications and includes some successful satellite systems and research projects description. Basic concepts from *Information Theory* are then introduced, such as the *Shannon* channel capacity limit and the need of forward error correcting (FEC) coding. A review of coding schemes used so far is also given. Some basics on binary turbo codes, duo-binary turbo codes and LDPC codes are following. Moreover, all the adapted decoding algorithms are described. Next, the computer simulation environment is been set up. That includes a generic block diagram, simulation assumptions, simulation performance validation and comparison between the considered coding schemes.

In Chapter 3, the improved SOVA turbo decoder is introduced. A *novel* normalisation scheme is then described, which is essential to avoid possible overflow of the iterative decoder. After that, a simple *novel* method with two steps of correcting factor to the SOVA turbo decoder output is given. That results in remarkable BER performance for large frame lengths and high number of decoding iterations in the AWGN/uncorrelated Rician fading channel. That is, no error at BER floor down to 10^{-6} .

Two *novel* approaches on improved decoding algorithms for binary turbo codes are described in Chapter 4. This is improved Max-Log-MAP decoding algorithms at reasonable decoding complexity increase, based on either the *max/max** operation re-

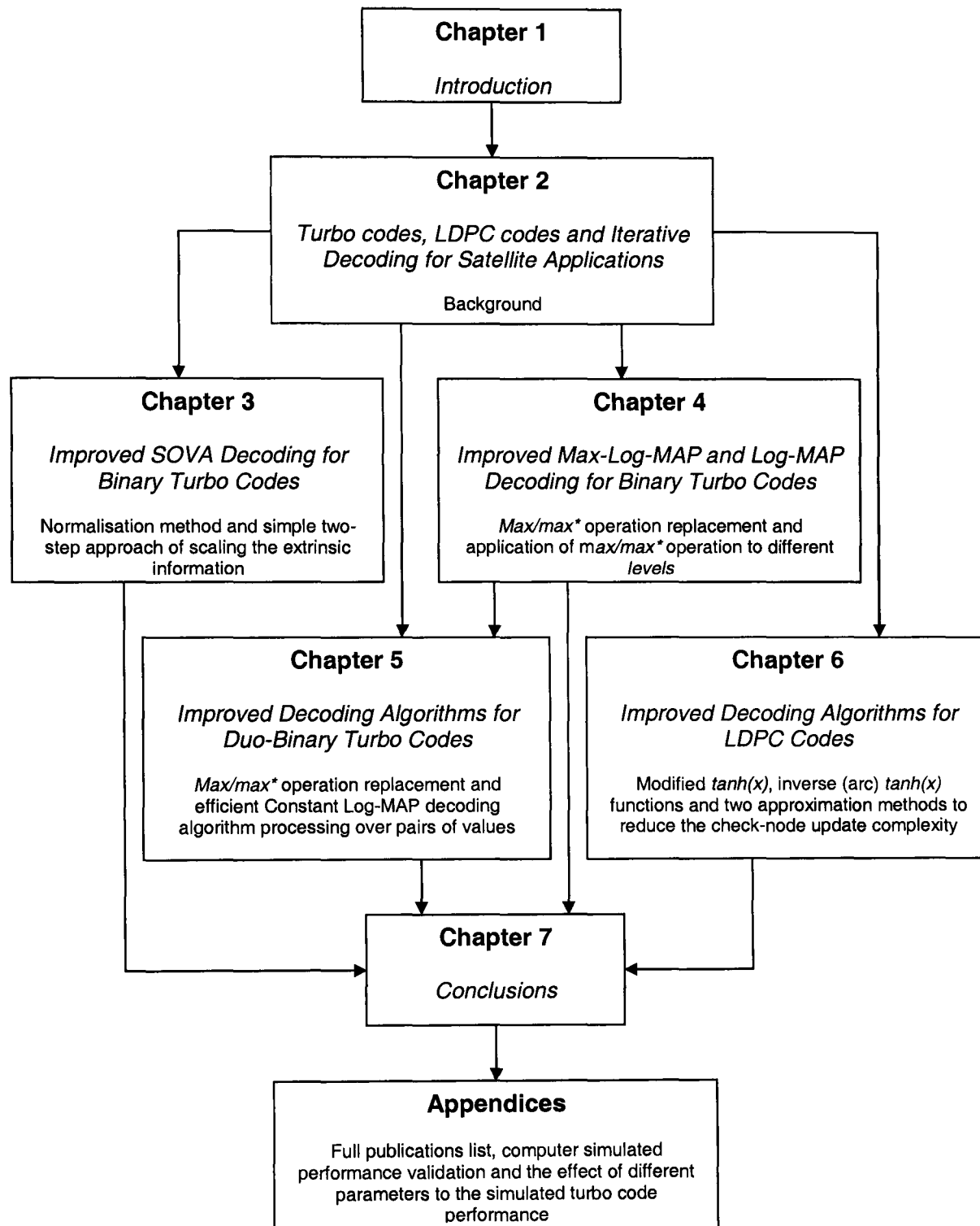


Figure 1.1: Thesis organisation.

placement method or the application of the \max/\max^* operation in different *levels*. On the other hand, the same decoding algorithms reduce the decoding complexity of Log-MAP decoding at reasonable BER performance degradation.

The contribution of Chapter 5 is based on two *novel* improved decoding algorithms for duo-binary turbo codes. As a practical application, the DVB-RCS turbo code standard is considered. The \max/\max^* operation replacement method is adapted from the previous Chapter to either improve the Max-Log-MAP decoding at reasonable decoding complexity increase or reduce the decoding complexity of Log-MAP decoding at reasonable BER performance degradation. Another interesting *novel* approach is then described, based on the Constant Log-MAP algorithm. This results in a BER performance improvement compared to an existing algorithm with no cost to the decoding complexity, approaching the Log-MAP performance, as in the binary case.

In Chapter 6 two *novel* algorithms to reduce the error floor of regular binary LDPC codes are described. Based on the LLR-SPA, the effect of the argument approximation used in $\tanh(x)$ and inverse (arc) $\tanh(x)$ functions is investigated. By an appropriate modification (i.e. clipping), the error floor of the code is reduced to BER below 10^{-7} . Two more *novel* approaches are introduced to reduce the decoding complexity of the two previous methods. These are based on piecewise linear approximation and quantization table methods.

The last Chapter 7 concludes this research work. Summary of the undertaken research work is given. After that, some topics for further research are suggested.

Three appendices are attached at the end. In Appendix A, a full publications list is reported. Appendix B, is related to the turbo/LDPC performance validation by means of computer simulations and Appendix C is based on the effect of different parameters to the simulated turbo code performance. This may be found useful to some of the readers.

1.5 Summary

In this Chapter turbo and LDPC codes were introduced, as being two of the most significant channel coding schemes over the recent years. This kind of codes has motivated our research work, due to the variety of applications in many standards and practical systems used in satellite communications. Research work objectives were set up and

the original achievements as well as the personal contributions were reported. Then, the thesis organisation was given and the next presented Chapters were described in brief.

Chapter 2

Turbo Codes, LDPC Codes and Iterative Decoding for Satellite Applications

This Chapter acts as *background* to the next presented Chapters. Inspired by *satellite communications* where channel coding plays an important role to achieve efficiency on the satellite RF power, a brief overview is given first. Elements from *Information Theory*, principles of *iterative decoding* and soft-input soft-output (SISO) decoding algorithms description follow. Then, a brief on turbo codes (in both binary and duobinary form) as well as on LDPC codes is given. Finally, computer simulation set up, related performance validation and relative comparison of these codes is described.

2.1 Satellite Communications Aspects

In this Section, a brief history of satellite communications is given [35], followed by some successful satellite systems and research projects.

2.1.1 A Brief History of Satellite Communications

Satellite communications were inspired by work by *A. C. Clark* and *J. Pierce* in the 1940's on how to bounce signals off passive satellites [35]. Echo I satellite was launched in 1960 requiring high transmission power to overcome path losses, of the order of 180 dB. Telstar I was launched in 1962, but lasted only for a few weeks because of radiation damage. The former was a *passive* satellite, while the later was an *active* satellite. In 1963, Telstar II could offer one TV channel and a number of telephone channels. The first commercial satellite - INTELSAT I (Early Bird) - was launched by the international telecommunications satellite organisation (INTELSAT) in 1965. It could provide 240 telephone circuits between the United States and Europe in a geosynchronous orbit. After the successful operation of INTELSAT II and INTELSAT III, INTELSAT IV could provide 6,000 telephone circuits in 1970. Digital technology to echo-cancellors was used in 1979 to cope with the long transmission delays.

Satellite broadcasting services started in the 1970's, as soon as undersea optical fibres were used for transatlantic telephony. Satellite broadcasting is considered to be nowadays an affordable economic solution for the distribution of broadcast radio and TV programming. Direct to home satellite broadcasting started in the 1990's using high power satellites in the Ku band and small low cost earth stations. More recently, two-way satellite communications promise to deliver high speed Internet access systems.

The mobile satellite operator INMARSAT came into existence at around the same time as the first cellular operators providing analogue services in the 1980's [28]. In its initial period, i.e. first generation (1G), speech and low data services were provided mainly to the maritime market in the L band using global beam coverage satellites. In 1990/1 aeronautical services were added (INMARSAT II) to passenger aircrafts and to some land vehicles introducing spot beam high power satellites. New services were introduced in the second generation (2G), such as paging and navigation, using higher rate digital technology with spot beam operation and desktop size terminals. That happened in 1997/8 (INMARSAT III).

In the mid 1990's, several regional GEO systems emerged in competition, e.g. OMNITRACS, EUTELTRACS, AMSC and OPTUS, using both L and Ku bands and

targeting services on land vehicles. Nevertheless, these systems were moderately successful, in contrast to INMARSAT's regional version of GEO satellites. In this case, INMARSAT III could provide satellite services to around 200,000 customers [28].

In the 1990's low earth orbit (LEO) and medium earth orbit (MEO) satellite systems were launched to compete with the terrestrial global system for mobile communications (GSM). Global coverage could be provided with handoff of a call between satellites. Such examples were the *Iridium* and the *Globalstar* system using LEO satellites as well as the *ICO* system using MEO satellites. These systems failed in the early 2000's because of the huge constellation cost and the quick development of the terrestrial mobile networks, but some of them continue to exist on a smaller scale, e.g. news reporting to remote areas.

The deployment of high power satellites of around 5 KW with 100-200 spot beams (super GEO's) in the mid 1990's, belongs to the third generation (3G) of mobile satellite communications. Such a system is *Thuraya*, which started to provide services like GSM and general packet radio service (GPRS) in 2000, covering Asia and part of Europe. Besides, INMARSAT IV using super GEO satellites, which is compatible with the 3G cellular systems, is expected to increase the data rates from 64 Kbps to 432 Kbps. The existing global area network (GAN) system is going to be replaced by the so-called broadband GAN (BGAN) system, which was launched in March 2005, and is expected to be in operation by the end of this year.

As a final remark, satellites can provide niche unicast services to areas that are not accessible to cellular systems in an economic way [28]. Deployment of large constellations proved to be too expensive, so GEO satellites seem to be the best solution. Some *recommendations* for successful satellite systems are summarised, as follows

- Satellites should not compete but collaborate with cellular systems
- Make use of the wide coverage broadcast feature of satellites
- Select the appropriate service based on the system delivery mechanisms

2.1.2 Some Successful Satellite Systems and Research Projects

Integrated Satellite/Terrestrial Universal Mobile Telecommunication System (S/T-UMTS)

New multimedia services, such as content delivery (e.g. audio/video streaming through a small buffer) and push-and-store services (e.g. by storing at the terminal and accessing later by the user) are promising in 3G (UMTS) systems, as the technology in cache memory devices has been improved rapidly and the cost has been dramatically reduced [28, 36]. These new kinds of services are referred to as multimedia broadcast and multicast services (MBMS) and are well suited to satellite delivery. The role of an *integrated* system is to divide the services according to the delivery mechanism that best matches to them. For example, in a cellular environment MBMS services are not delivered efficiently because they are subject to the propagation channel conditions. This can be overcome by the use of satellites. Satellite-based MBMS services are usually referred to as satellite digital multimedia broadcast (S-DMB) services. In case of building or urban environments, satellite signals are delivered more efficiently by gap-fillers (i.e. repeaters). These are located at some 3G base stations, broadcasting the MBMS signals in the terrestrial environment.

S-DMB in Europe The proposed system is led by Alcatel Space in Europe through the European Union (EU) projects SATIN in the fifth framework program (FP5) [37], MODIS in FP5 [38] and MAESTRO in FP6 [36]. SATIN (Jan. 2001 - Mar. 2003) has provided the architecture and the feasibility of the integrated satellite/terrestrial system, while MODIS (Apr. 2002 - Oct. 2004) has provided demonstrations using the Monaco 3G system. MAESTRO (Jan. 2004 - Jan. 2006) has defined the concept of S-DMB system and the fully operational system is going to take place in 2007/8.

S-DMB in Asia Mobile television is the service delivery in Korea and Japan [28]. It is also known as MBSAT system and is led by the mobile broadcasting corporation (MBCO) [39]. The satellite was launched in March 2004 and the commercial operation started in October 2004. However, no integration with cellular systems is provided.

Digital Audio Broadcasting (DAB) Via Satellite (S-DAB)

There are two commercial systems in the USA providing S-DAB services from the early 2000's [28]. The use of gap-fillers is crucial for full coverage but no integration with cellular systems is provided. *XM radio* deploys GEO satellites, while *Sirius* deploys highly elliptic orbit (HEO) satellites. By adopting HEO satellites it can be achieved improved coverage in urban areas and reduction in the number of required gap-fillers. XM radio offers more than 100 radio programs and data services and has reached four million subscribers in 2005.

High Altitude Platforms (HAPS)

Apart from terrestrial/satellite communication networks, HAPS are another mechanism for fixed/mobile delivery services [30]. The EU project CAPANINA in FP6 is looking at such communications aspects. HAPS are either solar powered airships or planes with a future location in the stratosphere (i.e. 17-22 Km altitude) and at least 60 Km coverage area. The advantage of HAPS is that they are similar to terrestrial networks in terms of *link budget*, but deliver services similar to satellites with regional type coverage area. For example, small dish/antenna size is feasible, due to favourable link budget and local content delivery can be provided, similar to satellite services.

Broadband Mobile Satellite Systems

In Section 2.1.1 it was mentioned that INMARSAT's BGAN system is an example of providing broadband mobile satellite services to users with data rates up to 432 Kbps. In addition, passenger vehicles such as aircrafts, ships and trains are another promising market for broadband-based services [28]. *Connexions* by Boeing is a system that provides broadband links to airplanes since 2002 and it is now pursuing the maritime operators market. The FIFTH EU project in FP5 has looked at the related aspects over high speed trains, while NATACHA EU FP5 project has looked at the related aspects over aircrafts. More recently, ANASTASIA and MOWGLY EU FP6 projects (both started in 2005) are going to extend the former projects to the general case of

passenger vehicles. Another issue is to introduce mobility to the existing DVB-RCS and DVB-S2 standards.

ESA MHOMS Project

MHOMS is an ESA funded project that was started in 2002 [32]. It stands for Modems for High-Order Modulation Schemes and it is composed of two phases. In phase one, which has already finished, advanced modem design algorithms are devised for satellite downlink data rates of 1 Gbps with near channel capacity limit performance. In phase two, a novel Field-Programmable Gate Array (FPGA) prototype is going to be designed and tested in the overall demonstrator. The aim of this project is to provide a feasible solution for high rate satellite modems used in different applications. For example, high-speed Internet access, backbone connectivity, earth observation and point-to-multipoint communications, such as multicasting and broadcasting could be feasible. At the same time, this project has contributed to the DVB-S2 standardisation group.

From the technical point of view, powerful error-correcting schemes (e.g. duo-binary PCCC, SCCC and LDPC codes) are used together with advanced modulation and demodulation techniques. As a final target was set the best trade-off between complexity and performance. Also, adaptive coding and modulation techniques are used to mitigate the deep fading events cause by the higher frequency bands. Some practical issues, such as non-linearity dynamic pre-compensation and synchronization have also been investigated.

A comparison for coding schemes was based on different aspects. BER performance, complexity, flexibility and maturity were the criteria to rank the three coding schemes. The final solution was SCCC. The choice with respect to PCCC was due to the observed error floor and increased complexity of the latter scheme. With respect to LDPC codes, which exhibit better BER performance, SCCC were less complex. It should be also noted that the chosen scheme was approximately 1 dB away from the channel capacity limit, considering the modulation constraints.

2.2 Elements from Information Theory

In this Section, fundamental concepts like channel capacity, the *Shannon* limit and error control coding are introduced. A brief on coding schemes is following.

2.2.1 Limits to Channel Capacity

In 1948 *Claude Shannon* published a landmark paper, *A Mathematical Theory of Communication*. According to this work, for any transmission rate less than or equal to a parameter called *channel capacity*, there exists a *coding scheme* that achieves an arbitrarily small probability of error. Hence, the transmission over a noisy channel can be perfectly reliable. *Shannon's* channel coding theorem launched the fields of *Information Theory* and *Error Control Coding*. Since then, a lot of research has been carried out towards achieving *Shannon's* capacity limit. The reason for that was the lack of guidance on how to find an appropriate coding scheme that achieves maximum data rate at arbitrarily small error probability and with limited complexity [26].

The channel capacity C for an AWGN channel with a limited bandwidth B , is a function of the average received signal power S and the average noise power N , according to the *Shannon-Hartley* formula [1]

$$C = B \log_2 \left(1 + \frac{S}{N} \right) \quad \text{bits/sec} \quad (2.1)$$

Thus, there exists a limit to the value of E_b/N_o , below which no error-free transmission can be reliable at any information rate. Assume that the data rate R takes its maximum possible value, i.e. equal to the channel capacity ($R = C$) and define Γ as the maximum spectral efficiency ($\Gamma = C/B$). By taking into account that $S = RE_b$ and $N = N_o B$, Eq. (2.1) becomes

$$\frac{E_b}{N_o} = \frac{2^\Gamma - 1}{\Gamma} \quad (2.2)$$

This is the minimum required E_b/N_o value for error-free transmission and it is referred to *Shannon's spectral efficiency limit*. In the limiting case when $B \rightarrow \infty$ or $\Gamma \rightarrow 0$,

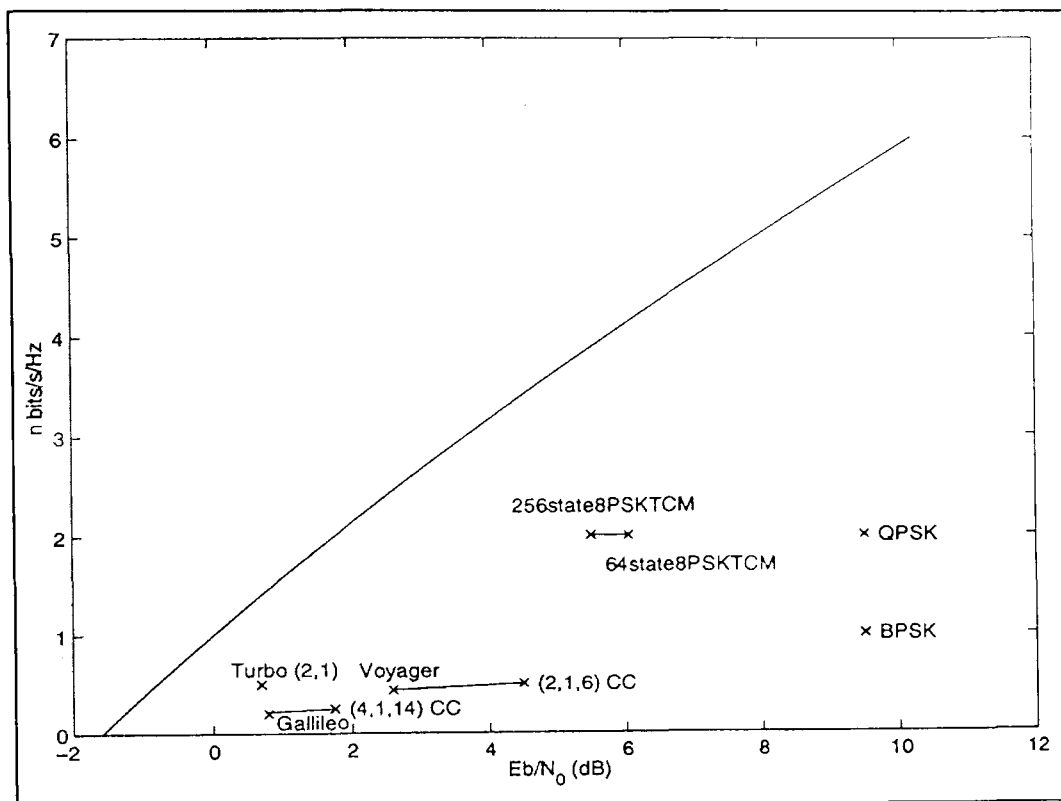


Figure 2.1: *Shannon's* capacity limit for the AWGN channel from *Vucetic* [1]. The achieved spectral efficiency of various modulation and coding schemes is also shown.

i.e. if the bandwidth is not limited, the minimum required E_b/N_o value for error-free transmission is $(E_b/N_o)_{min} = -1.59$ dB.

In Fig. 2.1 it is shown the *Shannon* capacity limit in the case of the AWGN channel with a bit error probability of 10^{-5} [1]. The achieved spectral efficiency of various modulation and coding schemes is also shown. Among the schemes, turbo codes (and also LDPC codes that are not shown here) achieve the best performance against the AWGN channel capacity limit.

2.2.2 Error Control Coding

Shannon had shown in fact how to achieve the channel capacity [18]. The information data are split into blocks of k bits and each possible data block is then mapped to another data block of n symbols, called *codeword*. These code symbols are then transmitted over the channel. The set of codewords and their mapping to data blocks is called *code* or more specifically FEC code. At the receiver a *decoder* must find the

codeword that most closely resembles the word it receives according to the *maximum likelihood decoding* (MLD) criterion. The uncertainty introduced at the decoder occurs because of the noise and interference of the channel. Thus, the decoder is more likely to confuse codewords that resemble each other more closely. The power of the code depends on the ability to correct errors, thus overcoming the channel characteristics.

Remarkably, *Shannon* had shown that channel capacity could be achieved by a *random code*. In a random code the mapping set of codewords is chosen randomly. The drawback is that the channel capacity limit can be approached, only if k and n tend to infinity. As there are 2^k different codewords, the decoder search for the closest codeword becomes impractical, unless the code structure provides a simpler search technique. In the following, an overview of coding schemes is given, up to the recent years.

2.2.3 A Brief History of Coding Schemes

After *Shannon's* remarkable theorem in 1948, work on coding theory in the 1950's and 1960's was mainly devoted to developing efficient encoders and decoders [40]. *Block* and *convolutional codes* were the two basic coding schemes known since that time. A block code maps a group of information bits to another data block, in which encoded bits are calculated according to the mathematical structure of the code.

Convolutional codes (CC), proposed by *Elias* in 1955 [41], were an alternative to block codes. The difference from block codes is that the encoder contains memory, so that the encoded data depend not only on the input data but also on some previous input data. During the 1970's coding research was shifted from theory to practical applications. For example, satellite communications in the early 1970's were using the $(2, 1, 6)$ *Odenwalder* CC (with rate $1/2$, number of states 2^6 and soft decision *Viterbi* decoding). This is also shown in Fig. 2.1.

Cyclic codes belong to the family of *linear codes* and they were first studied by *Prange* in 1957. They are suitable for both random and burst error correction. Another class of cyclic codes that include both binary and non-binary alphabets are *Bose-Chaudhuri-Hocquenghem* (BCH) codes. These codes were proposed simultaneously by *Hocquenghem* on the one side in 1959 [42] and *Bose* and *Ray-Chaudhuri* on the other

side in 1960 [43]. *Reed-Solomon* (RS) codes, which were proposed by *Reed* and *Solomon* in 1960 [44], are non-binary BCH codes.

Concatenated codes were introduced by *Forney* in 1966 [45] achieving low error rates. In this case, two levels of coding are applied in a serial form. The two encoders, namely *inner* and *outer encoder*, are linked together through an *interleaver*. At the decoder side, each component code is decoded separately in order to obtain low computational complexity. The space missions of *Voyager* to Uranus in 1986 and *Gallileo* in 1989 were using an inner CC concatenated with an outer (255, 223) RS code, e.g. see Fig. 2.1.

In 1982 *trellis coded modulation* (TCM) was proposed by *Ungerboeck*. Previous existing systems were examining coding and modulation as separate entities. The breakthrough in TCM was that convolutional codes were combined together with modulation schemes, such as amplitude or phase shift keying (ASK or PSK), without any bandwidth expansion. Coding gains of 3-6 dB could be achieved compared to uncoded systems with the same spectral efficiency, as shown in Fig. 2.1.

Serial concatenated decoding is based on providing either hard or soft decisions from the outer to the inner decoder. In 1993 *iterative decoding* was proposed by *Berrou et al* for decoding a new class of error correcting codes, called *turbo codes* [6]. The basic idea is to use two convolutional codes in parallel linked together by a long interleaver and to decode them in an iterative manner, until a maximum number of decoding iterations is reached. As from Fig. 2.1, a rate 1/2 turbo code can achieve a BER of 10^{-5} at 0.7 dB in the AWGN channel, approaching very close to the channel capacity limit. As a practical application, turbo codes were used in the *Cassini-Huygens* mission to Saturn in 1997.

The near capacity performance of turbo codes made the search for other combinations of component codes. For example, *turbo product codes* were proposed by *Pyndiah et al* in 1994 [46] and can be decoded by the *Chase* algorithm. Recently, a *fast Chase* algorithm was proposed in [47]. *Turbo TCM* was proposed by *Robertson et al* in 1995 [48] and both *SCCC* and *HCCC* were proposed by *Benedetto et al* in 1998 [49], [50]. *Duo-binary turbo codes* were proposed by *Berrou et al* in 1999 [51], as a remedy for punctured turbo codes. Binary turbo codes have also been adopted by many standards,

such as the Consultative Committee for Space Data Systems (CCSDS) from NASA (1998) and the Third Generation Partnership Project (3GPP) for S/T-UMTS (1999) [1] and also in the duo-binary form in the ETSI DVB-RCS/RCT (2000) [4].

Low-density parity-check (LDPC) codes were introduced by *Gallager* in 1962 [21]. At that time, more effort was given to practical applications of concatenated codes, so these codes were forgotten for more than thirty years. The rediscovery of LDPC codes is owned to *MacKay* and *Neal* in 1996 [22], who were inspired by the iterative decoding process of turbo codes. The basic advantage of LDPC codes compared to turbo codes is that they do not show an early error floor at BER values of 10^{-5} . This is because of relatively higher minimum distances.

The construction of LDPC codes, as proposed by *Gallager*, is based on large computer searches, due to the randomness of the parity-check matrix. The encoding time is also proportional to the square of the coded block size. That makes them difficult to be applied to practical systems. Efficient encoding methods with linear complexity in time, such as progressive edge growth (PEG) codes, array codes, circulant PEG codes and accumulate RA codes (i.e. turbo-like codes) are some examples of the latest developments of LDPC encoding [52].

In 2004 the Digital Video Broadcasting over Satellite (DVB-S) standard was updated to its new version (i.e. DVB-S2) by using a LDPC code that is constructed by a structured irregular RA code. In the new CCSDS standard that is under consideration, an accumulate RA code seems to be a strong candidate [52]. Both turbo and LDPC codes are now under consideration for application to WMANs [3], ADSLs [53] and industrial standards for magnetic data storage systems [54].

2.3 Soft-Input Soft-Output (SISO) Decoding Based on Trellises

Assume a typical telecommunication system composed of a pair of encoder, decoder, a pair of modulator, demodulator and a channel. Also, the cases of BPSK modulation

and the AWGN channel are considered. The *Gaussian* process has zero mean and variance σ^2 .

The performance of a conventional decoder is significantly enhanced if, in addition to the *hard decision* made by the demodulator on the transmitted bit, some extra *soft information* on the *reliability* of that decision is passed to the decoder input [18]. Assume that the received signal is close to the decision threshold in the demodulator (i.e. between 0 and 1), then that decision has low reliability. In this case, it would be desirable that the decoder changes the decision when searching for the most probable codeword. Thus, the decoder should be able to make soft decisions, yielding a performance improvement of around 2 dB in most of the cases [18].

Concatenated codes can be decoded if the output of the one decoder is the input to the next decoder. Thus, an appropriate decoder that generates soft information as well as makes use of it is required. This is the *SISO* decoder. The soft-output of a SISO decoder is based on the estimation of the probability that the information bit (denoted by u) is one to the probability that the information bit is zero. When the logarithm of this ratio is obtained, the soft output is usually referred to as *log-likelihood ratio* (LLR). In other words, this is the estimation of the *a posteriori probability* (APP) of the transmitted bit (denoted by x), given the observation of the received sequence of bits (denoted by r). Assuming BPSK modulation, where x takes values $+/-1$, the following formula holds

$$LLR = L(\hat{u}) = L(u|r) = \ln \frac{P(u=1|r)}{P(u=0|r)} = \ln \frac{P(x=+1|r)}{P(x=-1|r)} = L(x|r) \quad (2.3)$$

The *sign* of the LLR value corresponds to the hard decision of the transmitted bit. If it is positive, then bit '1' is assumed to be transmitted, otherwise if it is negative, then bit '0' is assumed to be transmitted. The *magnitude* of the LLR value corresponds to the reliability of this decision, which is a measure of the certainty of the transmitted bit (i.e. soft decision).

The *demodulator output* (i.e. in soft form) is thus based on the APP of the transmitted bit. From Eq. (2.3) and using *Bayes'* rule, we have

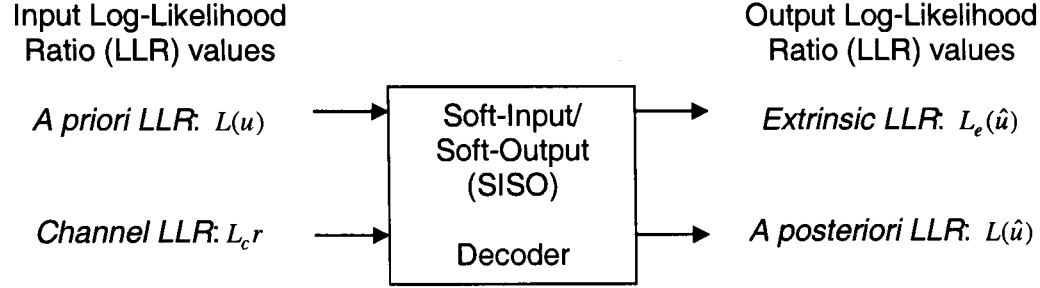


Figure 2.2: LLR values used in a SISO decoder.

$$\begin{aligned}
 L(\hat{u}) &= L(x|r) = \ln \frac{P(x=+1|r)}{P(x=-1|r)} = \ln \frac{P(r|x=+1) P(x=+1)}{P(r|x=-1) P(x=-1)} = \\
 &= \ln \frac{\frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\left(\frac{r-1}{\sigma}\right)^2\right\}}{\frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\left(\frac{r+1}{\sigma}\right)^2\right\}} + \ln \frac{P(r|x=+1)}{P(r|x=-1)} = \\
 &= \frac{2}{\sigma^2} r + \ln \frac{P(r|x=+1)}{P(r|x=-1)} = L_c r + L(u)
 \end{aligned} \tag{2.4}$$

where the term $L_c = 2/\sigma^2$ is called *channel reliability value* and the term $L(u)$ is the *a priori* LLR value of the information bit u .

The introduction of encoder/decoder scheme yields benefits on decision making [55]. For a systematic code, the *soft decoder* output is in the form of

$$L(\hat{u}) = L(x|r) = L_c r + L(u) + L_e(\hat{u}) \tag{2.5}$$

The new term $L_e(\hat{u})$, with respect to Eq. (2.4), is called *extrinsic* LLR. It represents an extra estimation on the LLR of the information bits, which was obtained during the decoding process utilising the code constraints [4]. It is also independent of both the *a priori* information and channel LLR values of the information bits.

A schematic diagram of the LLR values used in a SISO decoder from Eq. (2.5) is shown in Fig. 2.2. In the *iterative decoding process*, the extrinsic LLR is fed back to the input of another component decoder to serve as *a priori* information of the data bits for the next *decoding iteration*. A unified approach on the use of the extrinsic information in the iterative decoding process can be found in [56].

Typical SISO decoding algorithms based on trellises are either the soft-output *Viterbi* algorithm (i.e. SOVA) or the maximum a posteriori probability (MAP), also known as *BCJR*, algorithm and its approximations (i.e. Log-MAP, Max-Log-MAP). In the following, we give an overview to these SISO decoding algorithms, assuming binary convolutional codes and BPSK modulation. This is because improvements to the SOVA turbo decoder are reported in Chapter 3, while improvements to Max-Log-MAP and Log-MAP turbo decoder are reported in Chapter 4. A complexity comparison between the presented SISO decoding algorithms is also given at the end.

2.3.1 SOVA Decoding

The SOVA is regarded as an extension to the well-known *Viterbi* algorithm (VA) that generates reliability values on bits by observing the estimated codeword sequence. Note that the VA finds the path through the trellis with the largest *path metric* by observing the received sequence [40]. The process of the VA is based on a recursive manner by introducing the *add-compare-select* operation. Assuming a (n, k) convolutional encoder, at each time unit the VA adds 2^k branch metrics to each previously stored path metric, it compares the metrics of all 2^k paths entering each state and it selects the path with the largest metric (i.e. *survivor*). The survivor of each state is then stored along with its metric. This process is described mathematically as [4]

$$M_k(S_k) = M_{k-1}(S_{k-1}) + \ln P(u_k) + \ln \{p(r_k|s', s)\} \quad (2.6)$$

where $M_k(S_k)$ and $M_{k-1}(S_{k-1})$ are the path metrics associated with the trellis path S_k and S_{k-1} at time instants k and $k-1$ respectively, $\ln P(u_k)$ is the *a priori* information of bit u_k and $\ln \{p(r_k|s', s)\}$ is the *branch metric* corresponding to state transition $s' \rightarrow s$. For a systematic binary convolutional code with rate $1/n$ and an AWGN channel, the path metric is simplified to [4]

$$M_k(S_k) = M_{k-1}(S_{k-1}) + \frac{1}{2}L(u_k)x_k + \frac{1}{2}L_c r_{k,1}x_k + \frac{1}{2} \sum_{u=2}^n L_c r_{k,u}x_{k,u} \quad (2.7)$$

where x_k and $x_{k,u}$, $u = 2, \dots, n$ are the transmitted systematic and parity bits respectively and $r_{k,1}$ and $r_{k,u}$, $u = 2, \dots, n$ are the received systematic and parity values respectively. L_c is the channel reliability value and $L(u_k)$ is the LLR of the *a priori* information of systematic bits.

It can be shown that the final survivor path is the *maximum-likelihood* path. After the VA process is finished, SOVA needs to store on its memory two paths, the *survivor* and the *concurrent* path. The later one is the path which had diverged at a past time $j = k - \delta m$ and merged to the same state as the survivor path at time $j = k$. The path metric difference $\Delta = M_1 - M_2$ between these two paths is stored and SOVA starts the process from the end of the trellis by tracing back.

The reliability value, i.e. LLR, of a bit is produced by an *updating rule* based on the estimated bits of the survivor path \hat{u}_s and the concurrent path \hat{u}_c . All the LLR values of the survivor sequence are first initialized to $L_j(\hat{u}_k) = +\infty$ and then are computed as

$$L_j(\hat{u}_k) = \hat{u}_k \min \{L_j(\hat{u}_k), \Delta\}, \quad \hat{u}_{s,j} \neq \hat{u}_{c,j} \quad (2.8)$$

only when the estimated bits of survivor and concurrent path are different from each other. This algorithm was described by *Hagenauer* in [57]. A *modified version* of it, which had been actually proposed earlier by *Battail* [58], updates the reliability values in case of $\hat{u}_{s,j} = \hat{u}_{c,j}$ by

$$L_j(\hat{u}_k) = \hat{u}_k \min \{L_j(\hat{u}_k), \Delta + L_c\}, \quad \hat{u}_{s,j} = \hat{u}_{c,j} \quad (2.9)$$

where L_c represents the reliability of the concurrent path. The extended updating rule from the above equation makes this modified version of SOVA superior to the proposed algorithm from [57].

2.3.2 MAP Decoding

The MAP algorithm, usually referred to as *BCJR* algorithm, is a well-known process to estimate the transmitted sequence of bits of a linear code [59]. This is the main

difference to the VA, which estimates the transmitted codeword sequence. For a single convolutional code, these two algorithms have approximately the same BER performance, although the former one is more complex [1]. This has made the MAP algorithm unattractive for practical implementations. However, the MAP algorithm became of interest again, after the introduction of turbo codes.

Assume that u is an information block of N bits, encoded by a systematic binary convolutional code of rate $1/n$, BPSK modulated and transmitted over the AWGN channel. Assume also a trellis transition from a state s' , at time instant $k-1$, to a state s , at time instant k . The objective of the MAP algorithm is to estimate the transmitted block of information bits by observing the received sequence r . After appropriate *initialisation*, the *forward* and *backward recursion* are computed recursively, as

$$\alpha_k(s) = \sum_{s'} \alpha_{k-1}(s') \gamma_k(s', s) \quad (2.10)$$

$$\beta_{k-1}(s') = \sum_s \beta_k(s) \gamma_k(s', s) \quad (2.11)$$

where γ_k is the *branch transition probability* associated with the *a priori* information of bit u_k , denoted by $P(u_k)$, and the *branch metric* that corresponds to the trellis transition, denoted by $P(r_k|u_k)$. That is,

$$\gamma_k(s', s) = P(u_k) P(r_k|u_k) \quad (2.12)$$

Considering the Gaussian channel distribution, it can be shown that the above equation can be expressed as [4]

$$\gamma_k(s', s) = A_k B_k \exp \left\{ \frac{1}{2} L_c r_{k,1} x_k + \frac{1}{2} \sum_{u=2}^n L_c r_{k,u} x_{k,u} + \frac{1}{2} x_k L(u_k) \right\} \quad (2.13)$$

where A_k, B_k are constants, x_k and $x_{k,u}$, $u = 2, \dots, n$ are the transmitted systematic and parity bits of the convolutional code respectively. Similarly, $r_{k,1}$ and $r_{k,u}$, $u = 2, \dots, n$ are the received systematic and parity values. L_c is the channel reliability value and $L(u_k)$ is the LLR of the *a priori* information of systematic bits.

The decoder soft-output value (i.e. LLR) of the transmitted bit u_k can be computed from

$$L(\hat{u}_k) = \ln \frac{P(x_k = +1|r)}{P(x_k = -1|r)} = \ln \frac{\sum_{(s',s):x_k=+1} \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s)}{\sum_{(s',s):x_k=-1} \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s)} \quad (2.14)$$

Using Eq. (2.13) in (2.14), we obtain

$$L(\hat{u}_k) = L_c r_{k,1} + L(u_k) + \ln \frac{\sum_{(s',s):x_k=+1} \alpha_{k-1}(s') \gamma_k^{(e)}(s', s) \beta_k(s)}{\sum_{(s',s):x_k=-1} \alpha_{k-1}(s') \gamma_k^{(e)}(s', s) \beta_k(s)} = L_c r_{k,1} + L(u_k) + L_e(\hat{u}_k) \quad (2.15)$$

where

$$\gamma_k^{(e)}(s', s) = \exp \left(\frac{1}{2} \sum_{u=2}^n L_c r_{k,u} x_{k,u} \right) \quad (2.16)$$

that represents the *extrinsic* term in the branch transition probability computation. Note also that the constants A_k, B_k are canceled out in the soft-output computation from Eq. (2.14). Therefore, they can be set to one in the branch transition probability computation from Eq. (2.13).

2.3.3 Log-MAP and Max-Log-MAP Decoding

If the MAP algorithm operates in the *logarithmic domain*, then the Log-MAP algorithm is obtained [7]. That makes easier a hardware decoder implementation by using additions instead of multiplications and a look-up table (LUT) of values for non-linear functions. Furthermore, the error performance does not degrade, when these modifications are taken into account. The basic operation is the *Jacobian logarithm* (or \max^* operation), which is defined as [60]

$$\max^*(a, b) = \max(a, b) + \ln \{1 + \exp(-|a - b|)\} = \max(a, b) + LUT \quad (2.17)$$

In practical implementations the LUT consists of eight values [7]. The forward, backward recursion and branch transition probabilities from the previous Section are now computed from [4]

$$\tilde{\alpha}_k(s) = \ln \sum_{s'} \exp \{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) \} = \max_{s'}^* \{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) \} \quad (2.18)$$

$$\tilde{\beta}_{k-1}(s') = \ln \sum_s \exp \{ \tilde{\beta}_k(s) + \tilde{\gamma}_k(s', s) \} = \max_s^* \{ \tilde{\beta}_k(s) + \tilde{\gamma}_k(s', s) \} \quad (2.19)$$

$$\tilde{\gamma}_k(s', s) = \frac{1}{2} L_c r_{k,1} x_k + \frac{1}{2} \sum_{u=2}^n L_c r_{k,u} x_{k,u} + \frac{1}{2} x_k L(u_k) \quad (2.20)$$

where *tilde* denotes values in the logarithmic domain. Following this approach, the LLR value from Eq. (2.14) becomes

$$\begin{aligned} L(\hat{u}_k) &= \ln \frac{P(x_k = +1|r)}{P(x_k = -1|r)} = \ln \frac{\sum_{(s',s):x_k=+1} \exp \{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) + \tilde{\beta}_k(s) \}}{\sum_{(s',s):x_k=-1} \exp \{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) + \tilde{\beta}_k(s) \}} = \\ &= \max_{(s',s):x_k=+1}^* \{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) + \tilde{\beta}_k(s) \} \\ &- \max_{(s',s):x_k=-1}^* \{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) + \tilde{\beta}_k(s) \} \end{aligned} \quad (2.21)$$

Omitting the LUT for $\ln\{\cdot\}$ as from Eq. (2.17), the Log-MAP algorithm is simplified to the Max-Log-MAP algorithm. In this case, the \max^* operator in Eqs. (2.18), (2.19) and (2.21) is replaced by the \max operator.

Note that the Max-Log-MAP algorithm can be regarded as a *dual-VA* by updating the LLR output after having processed the trellis both in the forward and the backward direction [60]. It is thus equivalent to the modified SOVA proposed by *Battail* [61].

2.3.4 Decoding Complexity Comparison

Complexity issues between different decoding algorithms can be found in [1]. The comparison is based on the number of \max operations, LUT values, additions and multiplications. It can be shown that the Log-MAP algorithm is approximately three times

more complex than the SOVA, while the Max-Log-MAP algorithm is approximately twice more complex than the SOVA. This can also be explained because of the soft-output computation of the considered SISO decoding algorithms [7]. The updating process of the Log-MAP algorithm is based on all trellis paths, while the Max-Log-MAP algorithm considers two best paths. The SOVA also takes into account two paths, but not necessarily the best paths. Note that the SOVA requires approximately the double complexity compared to the standard VA.

When decoding a single convolutional code, the MAP and Log-MAP algorithms have identical BER performance. The same does for SOVA and Max-Log-MAP algorithms [1]. At very high BER values, e.g. 10^{-2} , there is very small performance degradation of SOVA compared to the MAP algorithm [1].

In iterative decoding, SOVA is *sub-optimum* in terms of BER performance. In particular, the performance degradation against the MAP turbo decoder is approximately 0.7 dB at BER of 10^{-4} , assuming a rate 1/2 turbo code for BPSK signals over the AWGN channel [7]. In case of the iterative Max-Log-MAP algorithm, the performance degradation against the MAP turbo decoder is approximately 0.4 dB at the same BER value [7].

2.4 Binary Turbo Codes

Turbo codes have been one of the most remarkable scientific inventions in the coding theory field for more than ten years [6]. The *encoder* consists of two recursive systematic convolutional (RSC) encoders, which are connected in *parallel* by an interleaver. The *interleaver* permutes the input block sequence of bits. The *decoder* is based on an *iterative process* by decoding the two constituent RSC codes separately (i.e. local decoding) and exchanging information between them. This method is good approximation of the optimum MLD and is done because the overall decoding complexity of MLD is growing exponentially with the turbo encoder memory size and frame length.

We refer to *binary* or *classical* turbo codes, such as in PCCC scheme (e.g. see Fig. 2.3), which were originally proposed in [6]. SCCC or HCCC schemes make use of

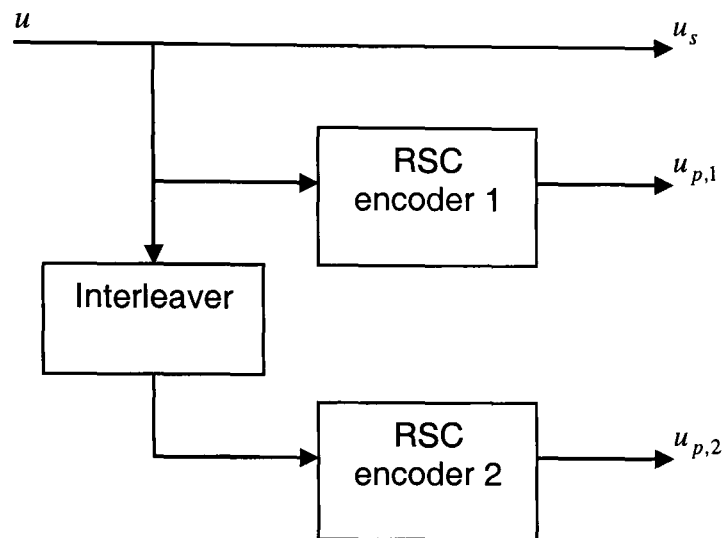


Figure 2.3: PCCC scheme.

RSC constituent encoders, but in different type of concatenation. Other turbo coding schemes include *non-binary turbo codes* (i.e. extension to more than one input bit sequence) and *block turbo codes* (i.e. serial concatenation of two linear block codes separated by a row-column interleaver).

2.4.1 Binary Turbo Encoder

Referring to Fig. 2.3, the information bit sequence u_s is called the *systematic* bit sequence (i.e. the uncoded output of the first RSC encoder). In addition, the coded bit sequence $u_{p,1}$ (or $u_{p,2}$) of the first (or second) RSC encoder is called the *parity* bit sequence. The systematic bit sequence of the second RSC encoder is not transmitted.

The basic process of the *interleaver* is to pass to the second RSC encoder a permuted version of the information bit sequence. Also, it has to generate a long block code from small memory constituent encoders [1]. The *design* of the interleaver is crucial, as its role is twofold. First, it breaks the low-weight input bit sequences and hence increases the code free distance and second, it decorrelates the inputs at the decoder side (i.e. extrinsic information and channel reliability values) by spreading out the burst errors. It can be proved that the turbo code performance is improved when the interleaver size is increased [1]. That is, assuming RSC encoders and a *good* interleaver obtained

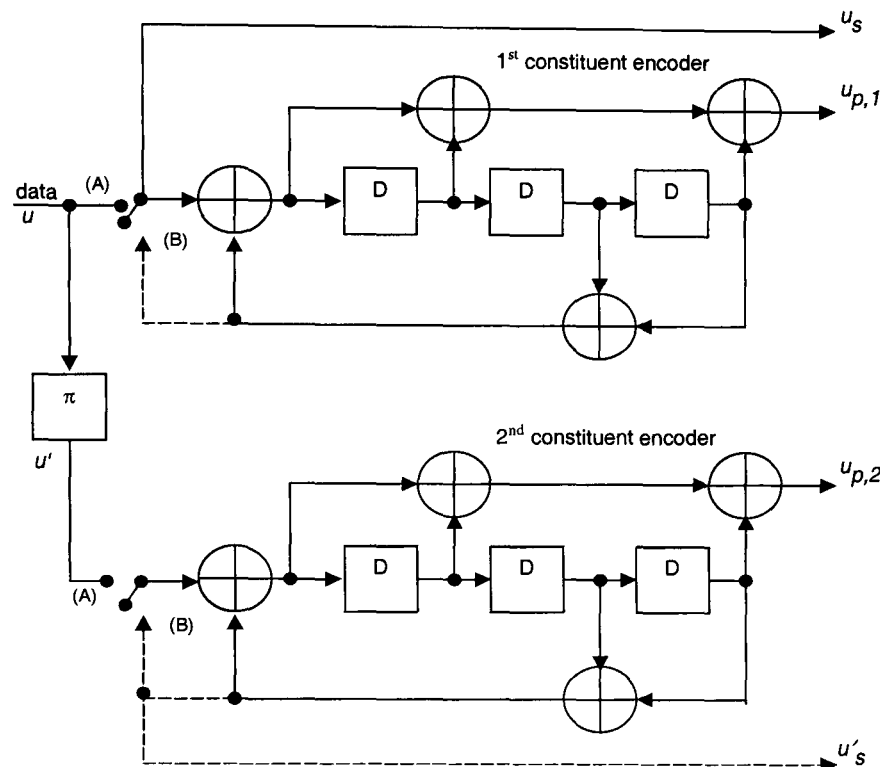


Figure 2.4: Typical turbo encoder.

by random permutations, the bit error probability is reduced by a factor of N , when the interleaver size is increased by N times [19]. This is also known as *interleaving performance gain*, reducing the asymptotic bit error probability performance of turbo codes [1].

The *trellis termination* of a RSC encoder is more complicated than in a non-recursive convolutional encoder, due to the presence of feedback. This is overcome by a *tail bit sequence* that forces the encoder to the all-zero state. The *circular trellis* or *tail-biting* technique is another solution to cope with the trellis termination problem (e.g. see Section 2.5.1).

A *typical* turbo encoder is shown in Fig. 2.4. It is composed of two RSC constituent encoders with 8-states and the overall coding rate is equal to $1/3$. A rate $1/2$ turbo code can be obtained by *puncturing* the parity bits. In fact, this turbo encoder has been selected by the *3GPP standard* (e.g. in S/T-UMTS) [62] for transmission of data packets, ranging from 40 to 5114 bits. Some other turbo encoders for practical applications, such as in the CCSDS standard, are reported in [1, 31].

The *transfer function* of the turbo encoder in Fig. 2.4 is represented by

$$G(D) = \left\{ 1, \frac{g_1(D)}{g_0(D)} \right\} = \left\{ 1, \frac{1 + D + D^3}{1 + D^2 + D^3} \right\} \quad (2.22)$$

where $g_1(D)$ is the feed-forward polynomial and $g_0(D)$ is the feedback polynomial, in octal form, of the constituent RSC encoders respectively. The trellis termination is done by turning the switch from position *A* to position *B*.

2.4.2 Binary Turbo Decoder

The *turbo decoding principle* is based on an *iterative* decoding process between two SISO decoders. As shown in Fig. 2.5, the input of a SISO decoder is fed by the *a priori* information of systematic bits (L_{in}) and the received channel values corresponding to both systematic and parity bits ($L_c r$). It then produces a soft-output value ($L_{\hat{u}}$). The term L_c is the *channel reliability* value, as described in Section 2.3. The *extrinsic* information of systematic bits (L_e), in an analogous way to Eq. (2.5), is

$$L_e = L_{\hat{u}} - (L_{in} + L_c r) \quad (2.23)$$

with the same sign as the transmitted information bit sequence. In Fig. 2.5, $L_{e,1}$ is then used as *a priori* information for the subsequent SISO decoder. The extrinsic information of the latter is de-interleaved and fed back to the first SISO decoder. In

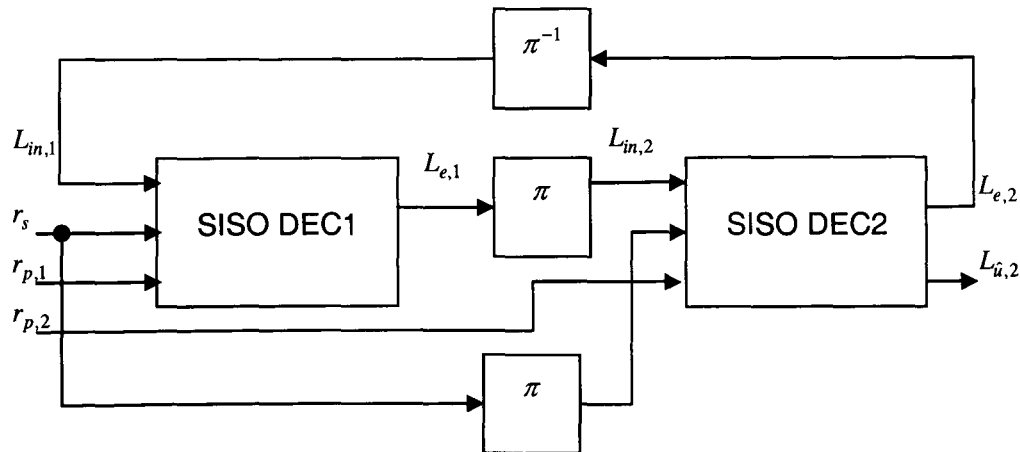


Figure 2.5: Turbo decoder.

this way, one round of iterative decoding process is completed. After a certain number of iterations, hard decisions are taken from the second decoder output.

Applying a *stopping criterion* [63], the number of decoding iterations can be reduced without significant performance degradation. By using semi-analytical methods, such as the extrinsic information transfer (*EXIT*) charts [64], the convergence behaviour of the turbo decoder can be predicted. Nowadays, the *turbo principle* based on the iterative decoding process, rather than the parallel concatenation of two convolutional encoders, is a *de facto* algorithm that can be applied to many areas of digital communications systems [31, 19].

2.4.3 Binary Turbo Performance Example

Turbo codes exhibit near *Shannon* capacity limit performance in the AWGN channel assuming large frame sizes. However, they are subject to error floors at BER lower than 10^{-5} . The reason for that is because at high SNR values, there exists a certain number of multiplicities (defined as the total number of codewords with Hamming weight d), which results in relatively small minimum distances [1]. This phenomenon can be overcome by better interleaver design. The observed error floor to the PCCC performance has made researchers to look for alternative coding solutions, e.g. SCCC, HCCC and turbo product codes. For example, turbo product codes are better than PCCC for coding rates greater than $1/2$ [65] and SCCC/HCCC are better than PCCC in the high SNR region [19].

Computer-based simulation results of binary turbo codes and comparison to related work can be found in Section 2.7. The effect of different parameters to the turbo code performance is also reported. Comparison to duo-binary turbo codes and also LDPC codes is attempted in the same Section.

As a remarkable performance example, we refer to NASA's Jet Propulsion Laboratory (JPL), which is considered to be a worldwide leading research site to design practical codecs for deep-space communications. The turbo code performance, as appeared in the web site [2], is shown in Fig. 2.6. It is assumed a frame size of 16384 bits and different generator polynomials, coding rates and number of decoding iterations. Note that the

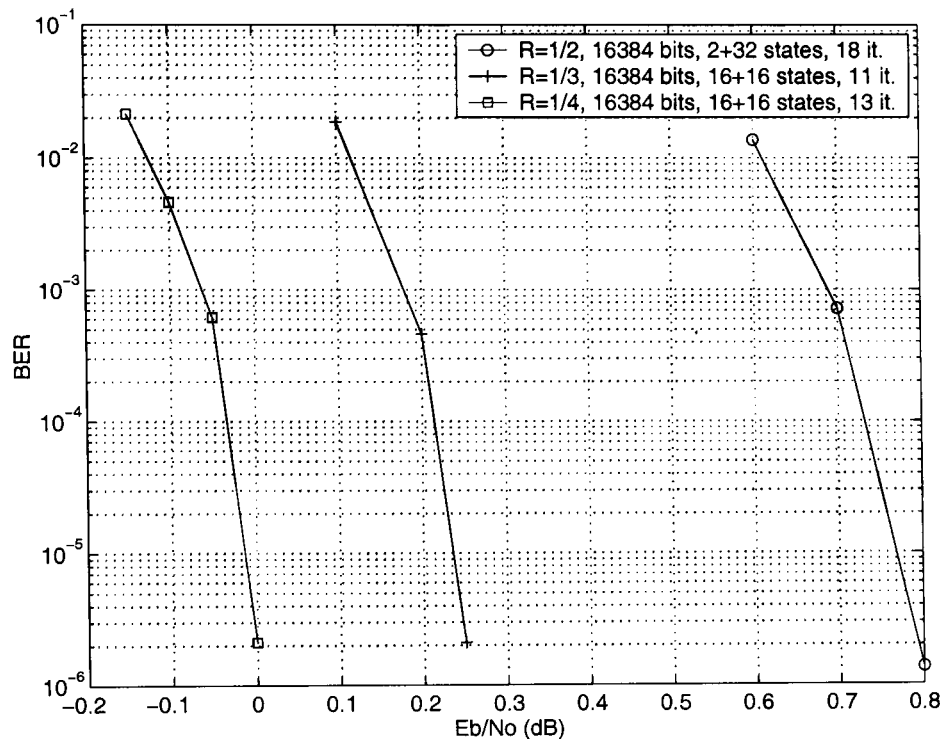


Figure 2.6: Turbo code performance from the JPL web site [2]. Frame size 16384 bits, different coding rates and number of decoding iterations. For coding rate $R = 1/2$, an asymmetrical turbo code is considered with lower complexity.

Shannon limit for the binary *Gaussian* channel at BER of 10^{-5} is approximately 0.2 dB for a code with rate $R = 1/2$, -0.5 dB for rate $R = 1/3$ and -0.8 dB for rate $R = 1/4$ respectively. From Fig. 2.6 it is justified the astonishing performance of turbo codes over the AWGN channel.

2.5 Duo-Binary Turbo Codes

Duo-binary turbo codes are built from RSC constituent encoders with two inputs and are shown to perform better than the classical (i.e. binary) turbo codes at very low BER values and high coding rates, due to increased minimum distances and lower density of erroneous paths [51, 3]. That makes them attractive in practical systems. For example, an 8-states duo-binary turbo code is currently been adopted by the ETSI DVB-RCS and DVB-RCT standards [3]. A more detailed description of the duo-binary turbo code, such as in DVB-RCS standard can be found in Section 5.2.

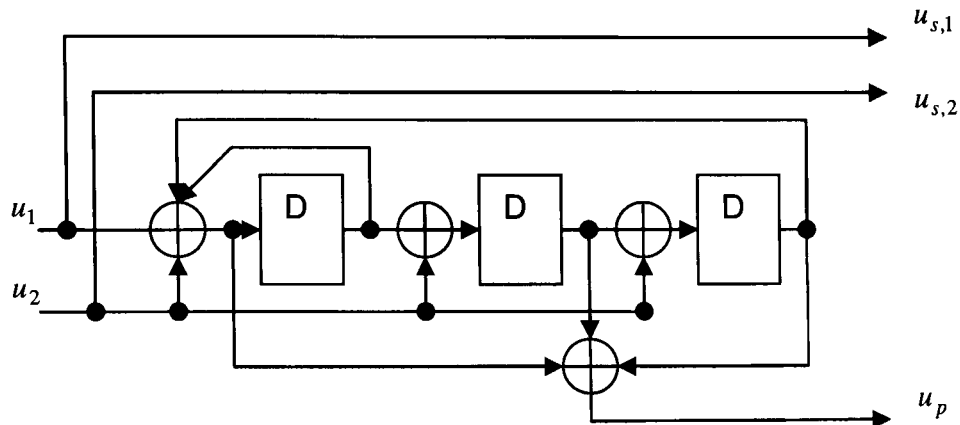


Figure 2.7: Duo-binary constituent RSC encoder.

As discussed in the previous Section, one major drawback of binary turbo codes is the performance degradation at higher coding rates, due to the puncturing technique. In case of duo-binary turbo codes, the need of puncturing is less crucial. This is because they make use of RSC constituent encoders with coding rate $R = 2/3$, thus the correcting ability of the constituent encoders is less degraded [3]. In contrast, binary turbo codes make use of RSC constituent encoders with coding rate $R = 1/2$, so that more redundant symbols need to be discarded for equivalent coding rate.

2.5.1 Duo-Binary Turbo Encoder

A typical RSC *constituent encoder* for duo-binary turbo codes is shown in Fig. 2.7. It is an 8-states convolutional encoder with coding rate $R = 2/3$. This encoder is then formed into parallel concatenation through an interleaver, e.g. see Section 5.2. If the constituent RSC encoder has two output bits, such as in the DVB-RCS standard of Section 5.2, then the corresponding trellis diagram is shown in Fig. 2.8. In this case, there are eight states and four bit transitions per trellis node with two input bits (i.e. u_2, u_1) and also two output bits (i.e. $u_{p,1}$ and $u_{p,2}$) respectively. The transition labels are shown in order to the right of the diagram of Fig. 2.8.

There are two key advantages of using duo-binary turbo codes, instead of binary turbo codes. These are, the use of *circular coding* or *tail-biting* technique and support of *two levels* of interleaving [4].

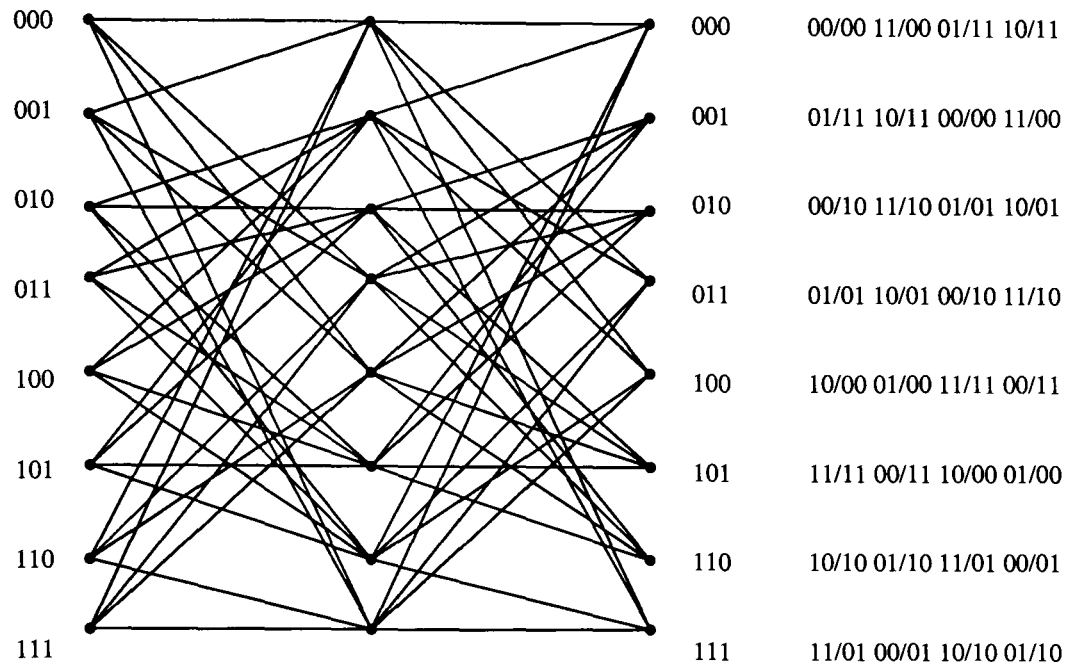


Figure 2.8: Trellis diagram of duo-binary constituent RSC encoder.

In circular trellis or tail-biting, the encoder retrieves the initial state at the end of the encoding process and the decoder can be initialised to any state and finish to this state in a circular manner. This makes a circular trellis RSC encoder to be considered exactly as a block encoder. An advantage of this technique is that no trellis termination is required to the constituent RSC encoders. Therefore, there is no need of tail bits, so that no extra bandwidth is wasted on flushing bits. This is very important in satellite broadcasting applications, e.g. in DVB-RCS, where a relatively small number of packets is transmitted, e.g. ranging from 12 to 216 bytes.

In duo-binary turbo codes two levels of interleaving can be applied. The first one performs intra-symbol permutations (i.e. inside bit couples) that increases the minimum distance of the code. The second interleaving is as in binary turbo codes and performs inter-symbol permutations (i.e. between bit couples), so as to reduce the correlation during the iterative decoding process.

By taking into account the required computations of an 8-state duo-binary turbo decoder, it is shown that this decoder is around 30% more complex than a turbo decoder in the binary form [51]. If the comparison is per bit, then the duo-binary turbo decoder

is simpler by 35%. This is because the number of trellis transitions is reduced to half, as the block size of bits is divided by two. It is thus concluded that duo-binary turbo codes are more attractive than binary turbo codes, allowing parallel architectures for high data rate decoding.

2.5.2 Duo-Binary Turbo Decoder

Duo-binary turbo codes can be decoded by following the same principle as binary turbo codes. The *symbol-based* iterative decoding is thus introduced [4]. Decoding with the *dual code* is another example, which is considered to be attractive especially for high coding rates [65, 66]. However, a logarithmic domain implementation of it requires a greater number of terms for transition metrics and also the computation of the \max^* operation requires greater precision, as negative quantities are involved [3]. In the following, a brief description of symbol-based Log-MAP and Max-Log-MAP iterative decoding is given. This is because improvements to the Log-MAP and Max-Log-MAP algorithms, suitable for duo-binary turbo codes, are addressed in Chapter 5.

Log-MAP and Max-Log-MAP Iterative Decoding

Assume an information block of N bit pairs, denoted by u , with possible values $u = 0, 1, 2$ and 3 (in decimal form) or $u = 00, 01, 10$ and 11 (in binary form) respectively. The block is encoded by a duo-binary turbo code, it is QPSK modulated and transmitted over the AWGN channel. Let x be the transmitted sequence and r the received sequence of symbols.

The decoder soft-output (i.e. LLR) provides an estimation of the transmitted symbol, given the observation of the received sequence, as [4]

$$\begin{aligned}
L\{\hat{u}_k(i)\} &= \log \frac{P(u_k = i|r)}{P(u_k = 0|r)} = \log \frac{\sum_{(s',s):u_k=i} \exp \left\{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s) + \tilde{\beta}_k(s) \right\}}{\sum_{(s',s):u_k=0} \exp \left\{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s) + \tilde{\beta}_k(s) \right\}} = \\
&= \max_{(s',s):u_k=i}^* \left\{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s) + \tilde{\beta}_k(s) \right\} \\
&- \max_{(s',s):u_k=0}^* \left\{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s) + \tilde{\beta}_k(s) \right\}, \text{ for } i = 1, 2 \text{ and } 3 \quad (2.24)
\end{aligned}$$

Compute now $L(\hat{u}_k) = \max [L\{\hat{u}_k(1)\}, L\{\hat{u}_k(2)\}, L\{\hat{u}_k(3)\}]$. The final decision is taken according to

$$\hat{u}_k = \begin{cases} 01, & \text{if } L(\hat{u}_k) = L\{\hat{u}_k(1)\} \text{ and } L\{\hat{u}_k(1)\} > 0 \\ 10, & \text{if } L(\hat{u}_k) = L\{\hat{u}_k(2)\} \text{ and } L\{\hat{u}_k(2)\} > 0 \\ 11, & \text{if } L(\hat{u}_k) = L\{\hat{u}_k(3)\} \text{ and } L\{\hat{u}_k(3)\} > 0 \\ 00, & \text{otherwise} \end{cases} \quad (2.25)$$

Assume that a symbol transition occurs from a trellis state s' , at time instant $k-1$, to a trellis state s , at time instant k . The computation of forward and backward recursion is done in a similar way to Eqs. (2.18), (2.19) from Section 2.3.3, but based on symbol values rather on bit values, i.e.

$$\tilde{\alpha}_k(s) = \log \sum_{s'} \exp \left\{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s) \right\} = \max_{s'}^* \left\{ \tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s) \right\} \quad (2.26)$$

$$\tilde{\beta}_{k-1}(s') = \log \sum_s \exp \left\{ \tilde{\beta}_k(s) + \tilde{\gamma}_k(s',s) \right\} = \max_s^* \left\{ \tilde{\beta}_k(s) + \tilde{\gamma}_k(s',s) \right\} \quad (2.27)$$

If the encoder starts from the zero state s_0 , at time instant $k=0$, and ends at the final state s_N , at time instant $k=N$, for a circular trellis we have $\alpha_0(s_0) = \alpha_N(s_N)$ and $\beta_N(s_N) = \beta_0(s_0)$. It can be shown that the branch transition probabilities are computed from [4]

$$\begin{aligned}
\tilde{\gamma}_k(s',s) &= \frac{1}{2} L_c [r_{k,s,I} x_{k,s,I}(i) + r_{k,s,Q} x_{k,s,Q}(i) + r_{k,p,I} x_{k,p,I}(i, s',s) + \\
&+ r_{k,p,Q} x_{k,p,Q}(i, s',s)] + \ln \frac{P(u_k = i)}{P(u_k = 0)}, \text{ for } i = 0, 1, 2 \text{ and } 3 \quad (2.28)
\end{aligned}$$

where I and Q represent the QPSK modulation mapping components and k, s and k, p denote systematic and parity symbols at time instant k respectively. The right-hand side of Eq. (2.28) represents the *a priori* information of the transmitted symbols.

At time instant $k = 0$, the *a priori* information of the first decoder is initialised to $P\{u_k(i)\} = 1/4$ for $i = 0, 1, 2$ and 3 , while the *a priori* information of the second decoder is the extrinsic information of the first decoder. During the iterative decoding process, the *a priori* information of the first decoder is the extrinsic information of the second decoder in the previous step and the *a priori* information of the second decoder is the extrinsic information of the first decoder. The extrinsic information is computed from [4]

$$\begin{aligned} L_e\{\hat{u}_k(i)\} &= L\{\hat{u}_k(i)\} - \frac{1}{2} L_c [r_{k,s,I} x_{k,s,I}(i) + r_{k,s,Q} x_{k,s,Q}(i) - r_{k,s,I} x_{k,s,I}(0) - \\ &\quad - r_{k,s,Q} x_{k,s,Q}(0)] - \ln \frac{P(u_k = i)}{P(u_k = 0)}, \text{ for } i = 1, 2 \text{ and } 3 \end{aligned} \quad (2.29)$$

The Max-Log-MAP algorithm omits the LUT of values that is used in the Log-MAP algorithm. Thus, the \max^* operator in Eqs. (2.24), (2.26) and (2.27) is replaced by the \max operator.

2.5.3 Duo-Binary Turbo Performance Example

Computer-based simulation results of duo-binary turbo codes and comparison to related work can be found in Section 2.7. Comparison to binary turbo codes and also to LDPC codes is shown in the same Section.

As a performance example, we refer to *Berrou's* recent work [3] where a 16-states duo-binary turbo code is proposed to improve the already adopted 8-states duo-binary turbo code in the DVB-RCS standard. In this way, minimum distances are increased from 30% to 50%, depending on the coding rate, at the expense of double decoding complexity. Frame error rate (FER) results are shown in Fig. 2.9.

The simulation parameters are either ATM or MPEG frame size, i.e. 424 or 1504 bits, coding rates $R = 1/2, 2/3$ and $3/4$, QPSK modulation, AWGN channel, improved

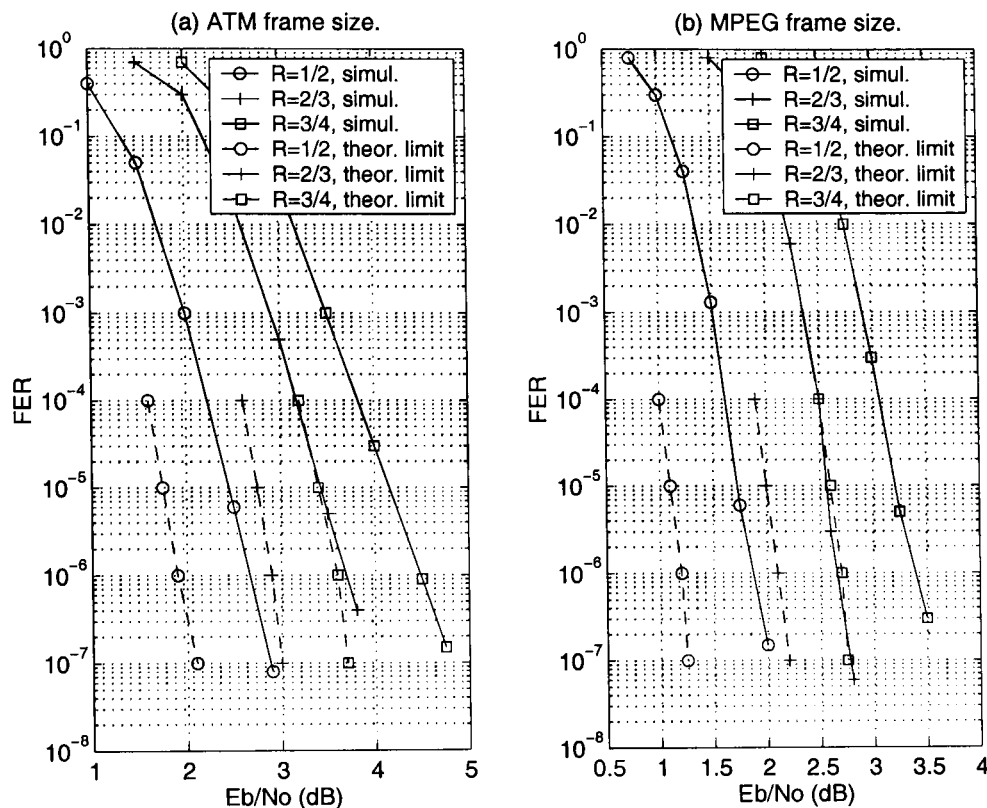


Figure 2.9: Duo-binary turbo code FER performance from *Berrou* [3]. 16-states encoder, different coding rates, QPSK modulation, AWGN channel, improved Max-Log-MAP algorithm, 4 bits quantization and 8 decoding iterations. Solid lines-simulation and dashed lines-theoretical limits. (a) ATM frame size, i.e. 424 bits, (b) MPEG frame size, i.e. 1504 bits.

Max-Log-MAP algorithm, 4 bits quantization and 8 decoding iterations. It is noticed that the simulation curves are very close to the *theoretical* limits on FER performance. That is, they are within 1 dB at FER of 10^{-6} with absence of any error floor. This also justifies the excellent performance of duo-binary turbo codes at very low BER/FER values.

2.6 LDPC Codes

Iterative decoding of *LDPC codes* is one of the most recent developments in the coding theory field [67]. LDPC codes belong to the family of *linear block codes* and can achieve near *Shannon* limit performance for large block sizes of data and at reasonable

decoding complexity. *Gallager* first introduced this type of codes in his *doctoral thesis* in 1962 [21] but after that, they had been forgotten for many years, mainly due to the appearance of *concatenated codes* proposed by *Forney*. A remarkable work by *Tanner* in 1981 gave rise to the so-called *Tanner graphs*, which are generalised graphical representations of LDPC codes [67]. During the mid 1990's, LDPC codes were rediscovered by *MacKay* and *Neal* [22] first and *Luby* and others later, by investigating the advantages of linear block codes using a very sparse (i.e. low-density) parity-check matrix [67].

Nowadays, LDPC codes can perform similarly or even better than turbo codes, allowing flexible high-speed parallel decoding implementations. As a practical application, the DVB-S2 standard has adopted LDPC codes as FEC scheme [68]. In addition, the under development new CCSDS standard for deep-space communications has considered the application of LDPC codes in [52]. Some other applications of the use of LDPC codes is in higher protocol layers, such as in packet and transport layer coding, for the new Digital Video Broadcasting for Handheld terminals (DVB-H) [69] and 3GPP MBMS [70] standards development.

The study of LDPC codes is currently focused on two main areas. The first area is related to the encoding problem, dealing with the construction of very sparse parity-check matrices at linear time. The second area is related to the iterative decoding by appropriate message-passing algorithms, which may also simplify the decoder complexity. These two areas are described in more detail below, after the introduction of *factor graphs*.

2.6.1 Factor Graphs

Let us define an LDPC code first. A (N, K) linear block code is a *LDPC code*, if the parity-check matrix H has a low density of ones, independently of the block size N .

The number of parity-checks on the received codeword is equal to $M = N - K$. An LDPC code is *regular*, if there are exactly d_s ones in each column and $d_c = d_s(N/M)$ ones in each row where $d_s \ll M$ (or $d_c \ll M$). For an *irregular* LDPC code, the matrix H has still low-density but the number of ones in each column or row is not

constant. The coding rate R is related to the column weight d_s and the row weight d_c , as $R = K/N = 1 - d_s/d_c$.

A *Tanner graph* can represent a parity-check matrix H of an LDPC code by the use of *nodes* connected by *edges*. Note that this is analogous to the trellis representation of a convolutional code, which applies to the decoding process. A *Tanner graph* can be considered as a *bipartite* graph whose nodes may be separated into two types and edges may only connect two nodes of different types. The two nodes in a *Tanner graph* are called *variable (or symbol) nodes*, denoted by v -nodes, and *check-nodes*, denoted by c -nodes, respectively.

A *construction* of such a graph can be obtained, if a check-node j is connected to a variable node i , whenever the element $h_{i,j}$ in H is one. The number of M rows of H , specify the c -node connections and the number of N columns of H , specify the v -node connections.

Assume a $(10, 5)$ linear block code with $d_s = 2$ and $d_c = 4$ with the following H matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Then, the corresponding *Tanner graph* to the H matrix is shown in Fig. 2.10 [67].

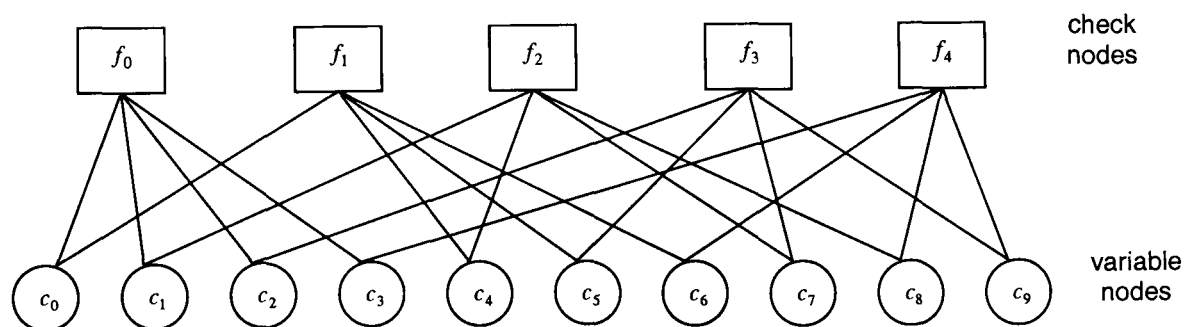


Figure 2.10: *Tanner graph* example.

When a path consisting of v edges closes back on the beginning of it, then the associated path is called *cycle of length v* . The *shortest cycles* in a bipartite graph are of length four and they are mainly responsible for the performance degradation of LDPC codes, producing an error floor in the high SNR region. The minimum cycle length of a *Tanner* graph is called *girth* (γ) of the graph.

2.6.2 LDPC Encoding

Assume a linear block code (N, K) , a generator matrix G with dimension $K \times N$ and an information block u of data with dimension $K \times 1$ [4]. Then, the codewords x are obtained as $x = G^T u$. Usually, the generator matrix G is in systematic form, i.e. $G = [I_K | P]$, where I_K is a $K \times K$ identity matrix concatenated with a $K \times (N - K)$ matrix P , which represents the parity-checks. The linear block code can be also described by a parity-check matrix H with dimension $M \times N$ where $M = N - K$. If the generator matrix is in systematic form, the parity-check matrix can be written as $H = [P^T | I_M]$, with the property $HG^T = 0$. The encoding process is thus defined.

LDPC codes are described by a *random* very sparse parity-check matrix H that can be constructed for any block size and coding rate. *Gallager* had proposed *regular* LDPC codes where the parity-check matrix H consists of other sub-matrices with certain properties. The ensemble of these codes has excellent properties, if $d_s \geq 3$ and $d_c > d_s$. *Gallager* has also shown that the error probability of LDPC codes with fixed d_s decreases exponentially at low noise value and for large block size. Also, the minimum distance can increase linearly with the block length [4].

The work of *MacKay* basically consists of finding semi-random generated sparse H matrices avoiding cycles of length four [4, 67]. He was the first to show the near capacity performance of LDPC codes by means of computer simulation results. The lack of sufficient structure of this kind of codes makes difficult a low-complexity encoding process. This is because the generator matrix G is not generally sparse, so that the encoding complexity is proportional to N^2 . Also, the parity-check matrix H is not usually in systematic form. This can be overcome by column reordering and *Gaussian* elimination.

Irregular LDPC codes can be described by variable degree distribution polynomials of the variable nodes and check-nodes [67]. They were proposed first by *Richardson* and *Luby*. The obtained codes were shown (by the use of *density evolution* method) to achieve performance within a decoding threshold, which is very close to the *Shannon* limit. The encoding process can be achieved in linear time.

Regular LDPC codes based on *finite geometries* have similar properties to the cyclic or quasi-cyclic block codes [67]. The encoder can be implemented by shift-registers and these codes perform very well for short block sizes. The iterative decoding complexity may increase, due to the large values of d_s and d_c . The choice of the block size and coding rate is also not flexible enough.

Repeat-accumulated (RA) codes were proposed by *Divsalar* and combine both the properties of serial turbo codes and LDPC codes [67]. The encoder is rather simple. It consists of a bit repeater, an interleaver and a differential encoder (i.e. accumulator). The drawback of this method is that it results in low rate codes. If one part of the bits is repeated more than the other part, then *irregular* RA codes can be obtained. They perform close to the theoretical capacity limits, even at higher coding rates. Usually, they are non-systematic codes. *Extended* irregular RA codes are in systematic form and allow both low and high coding rates.

The parity-check matrix of *array codes* is very simple and consists of identity matrices, null matrices and a basic matrix that uses permutations and cyclic shifts [67]. Array codes are very efficient to be generated in linear time but they are not quite flexible.

In *combinatorial codes* the random generation of the parity-check matrix is based on combinatorial mathematics [67]. This is because of the constraints introduced when designing such a code. No cycles of length four are feasible.

2.6.3 LDPC Decoding

The parity-check matrix H can be used to detect errors at the receiver, as $Hr = H(x + e) = HG^T u + He = He = z$ where e is the error vector and z is the syndrome vector. The *decoding problem* is based on finding the most likely error vector \hat{e} that

corresponds to the syndrome vector z , given the received sequence r . If the syndrome vector is *null*, then no decoding error occurs.

Two *practical* algorithms to decode LDPC codes were originally proposed by *Gallager*, based on either a hard or a soft decision iterative algorithm. In the former algorithm, also known as *bit-flipping*, digits may be changed, if they are contained in more than some fixed number of unsatisfied parity-check equations. The latter algorithm is usually referred to as sum-product algorithm (SPA), message-passing algorithm or belief propagation algorithm. It computes the APP of each noise symbol, given the received signal, in which messages (i.e. probabilities) are sent from noise symbols to check-nodes and vice versa, based on the bipartite graph defined by the parity-check matrix H . This algorithm is valid for statistically independent messages or when the graph contains no cycles. For a graph with girth γ , this assumption is valid up to the $\gamma/2$ -th iteration.

In the following, the SPA in the logarithmic domain (LLR-SPA) is described in brief using *Gallager's* approach [15]. More decoding algorithms, including reduced complexity ones, are addressed in Chapter 6.

Logarithmic Domain SPA (LLR-SPA)

Let $M(n)$ denote the set of check-nodes connected to the symbol-node n and $N(m)$ denote the set of symbol-nodes participating in the m -th parity-check equation. $N(m) \setminus n$ is the set of symbol nodes that participate in the m th parity-check equation, i.e. the position of ones in the m th row of the parity-check matrix H , excluding n . Similarly, $M(n) \setminus m$ represents the set $M(n)$, excluding the m -th check-node.

Define (λ) as the LLR of the message that symbol node n sends to check-node m , indicating the probability of symbol u_n being zero or one, based on all checks involving n except m , i.e. $\lambda_{n \rightarrow m}(u_n) = \ln \{q_{n \rightarrow m}(0)/q_{n \rightarrow m}(1)\}$. Similarly, define (Λ) as the LLR of the message that the m th check-node sends to the n th symbol node, indicating the probability of symbol u_n being zero or one, based on all symbols checked by m except n , i.e. $\Lambda_{m \rightarrow n}(u_n) = \ln \{r_{m \rightarrow n}(0)/r_{m \rightarrow n}(1)\}$.

The LLR-SPA is summarised in three steps [15].

1. **Initialisation.** After transmission through the channel, compute the APP of each symbol node n , as $L(u_n) = L_c r_n$ where L_c is the channel reliability value. The *initialisation* is done in every position of the parity-check matrix H , such that $h_{m,n} = 1$, as

$$\lambda_{n \rightarrow m}(u_n) = L(u_n) \quad (2.30)$$

$$\Lambda_{m \rightarrow n}(u_n) = 0 \quad (2.31)$$

2. **Iterative process.**

(a) *Check-node update.* For each m and for each $n \in N(m)$, compute

$$\Lambda_{m \rightarrow n}(u_n) = \left\{ \prod_{n' \in N(m) \setminus n} \text{sign}[\lambda_{n' \rightarrow m}(u_{n'})] \right\} \times \phi \left\{ \sum_{n' \in N(m) \setminus n} \phi[|\lambda_{n' \rightarrow m}(u_{n'})|] \right\} \quad (2.32)$$

where

$$\phi(x) = \ln \left(\frac{e^x + 1}{e^x - 1} \right), \quad x > 0 \quad (2.33)$$

(b) *Symbol-node update.* For each n and for each $m \in M(n)$, compute

$$\lambda_{n \rightarrow m}(u_n) = L(u_n) + \sum_{m' \in M(n) \setminus m} \Lambda_{m' \rightarrow n}(u_n) \quad (2.34)$$

For each n , compute

$$\lambda_n(u_n) = L(u_n) + \sum_{m \in M(n)} \Lambda_{m \rightarrow n}(u_n) \quad (2.35)$$

3. **Decision.** Decide if $\lambda_n(u_n) \geq 0$, then $\hat{u}_n = 0$ and if $\lambda_n(u_n) < 0$, then $\hat{u}_n = 1$. Compute the syndrome $\hat{u}H^T$ and if $\hat{u}H^T = 0$, then halt the algorithm and report \hat{u} as the decoder output. Otherwise, go to step 1. If a certain number of decoding iterations is reached and the algorithm does not halt, then a decoding failure is reported.

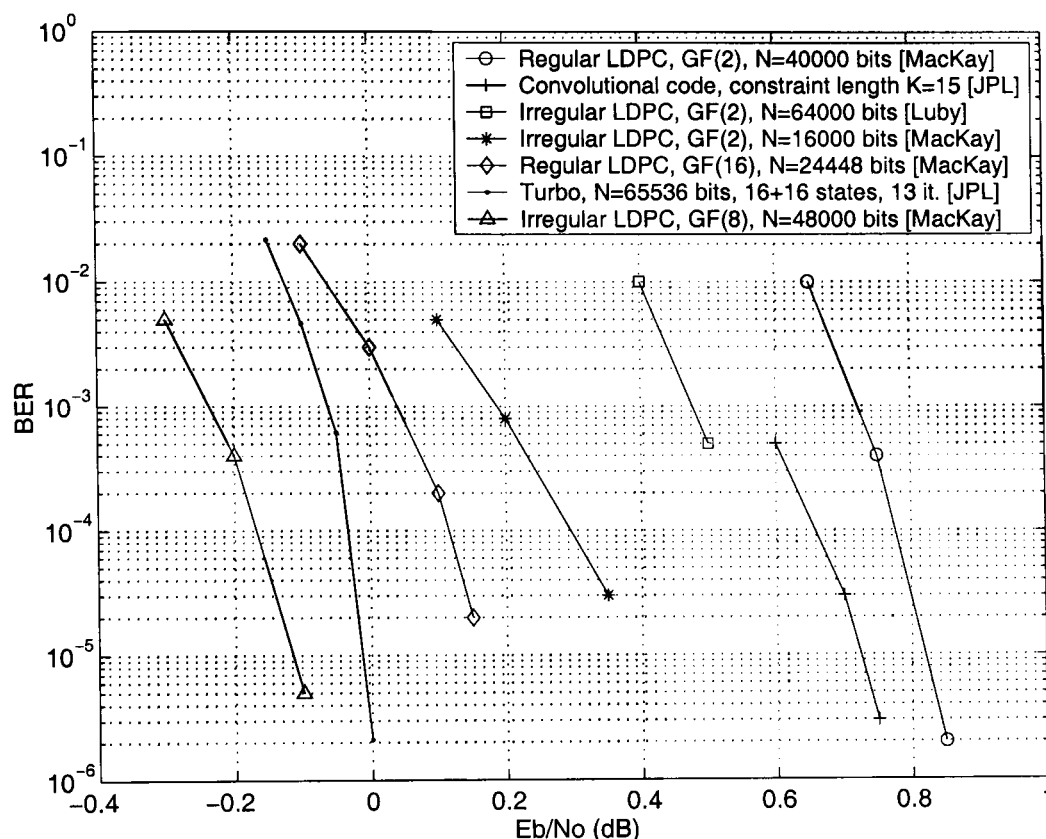


Figure 2.11: BER performance of different LDPC codes (i.e. regular, irregular, binary and non-binary) and comparison to turbo and convolutional codes, as from *Soleymani* [4]. Coding rate $R = 1/4$ over the AWGN channel.

Note that the messages in Eqs. (2.32), (2.34) represent *extrinsic* values, while the messages in Eq. (2.34) represent the soft-output values. The implementation of Eq. (2.32) requires $2d_c$ additions and $2d_c$ operations of the ϕ function.

2.6.4 LDPC Performance Example

Computer-based simulation results of LDPC codes and comparison to related work can be found in Section 2.7. In the same Section comparison to binary and duo-binary turbo codes is also reported.

As a performance example, we refer to [4] because different LDPC codes are compared each other. Fig. 2.11 depicts the related BER computer simulated results. It is assumed that rate 1/4 codes are used over the AWGN channel. Note that the *Shannon* limit in this case, is approximately -0.8 dB.

From Fig. 2.11 can be verified the superior performance of irregular LDPC codes compared to regular LDPC codes. Also, using non-binary alphabets, the BER performance of a LDPC code is improved with respect to the binary case. This is in agreement with the situation that occurs in binary/duo-binary turbo codes. It is interesting that, except for the regular binary LDPC code, the rest of LDPC codes outperform JPL's convolutional code with constraint length equal to 15. What is more impressive is that the irregular non-binary LDPC code outperforms JPL's turbo code with 16-states, which was also shown in Fig. 2.6.

Performance comparison between irregular LDPC codes and binary turbo codes is also reported in [4]. Up to frame sizes of 1000 bits, binary turbo codes can perform better. For larger frame sizes, irregular LDPC codes are now better and perform even closer to the *Shannon* limit. Moreover, related performance and complexity comparison between duo-binary turbo codes, SCCC and LDPC codes can be found in [32]. For a short frame size, duo-binary turbo codes are the best performed codes, while LDPC codes are the best ones for a large block size. On the other hand, SCCC are the least complex of the three codes, so that a small performance degradation is acceptable.

2.7 Computer Simulation Environment

The corresponding BER/FER performance of turbo and LDPC codes is obtained by means of computer simulations. Mainly, C programming language has been used. Sometimes, computer code was written from the scratch, e.g. in case of LDPC codes, and in some other times it was build up from existing ones, e.g. in case of turbo codes. Also, MATLAB program was deployed in the background, in order to analyse, verify and plot the results. Extensive computer simulations were run under either personal desktops or simulation servers of the mobile communications group, at CCSR. The C programs were developed using a simple editor (`nedit`) available in Linux. The `gcc` compiler (version 2.96) was used over Red Hat Linux 7.3, in all cases of computer simulations. The simulation time for a specific E_b/N_o value was varying from several hours, e.g. two to four, to several days, e.g. one to five, according to the desired BER value and number of decoding iterations and of course depending on the complexity of the decoding

algorithm, e.g. SOVA or Log-MAP.

A general computer simulation chain that was considered is shown in Fig. 2.12. In following, some more details are given for each of the individual blocks.

- **Data generator.** This is based on pseudo-random number generators that create random bits [71].
- **Encoder.** Three individual cases are considered. That is, binary turbo codes, duo-binary turbo codes (such as in the DVB-RCS standard) and LDPC codes.

In case of *binary* turbo codes, different generator polynomials (i.e. memory order) and frame size are supported. The standard interleaver used is a pseudo-random one, while some of the 3GPP interleaver patterns [62] were also deployed. The coding rate can be either $R = 1/3$ or $R = 1/2$. The latter is obtained by puncturing.

The *DVB-RCS* turbo code, as described in Section 5.2, has specific generator polynomials with memory order equal to three. It can support twelve frame sizes and seven coding rates with optimised interleaver patterns. In our simulations, it was considered either Asynchronous Transfer Mode (ATM) or Moving Picture Experts Group (MPEG) frame sizes, i.e. either 424 or 1504 bits frame size respectively, with different coding rates obtained by puncturing.

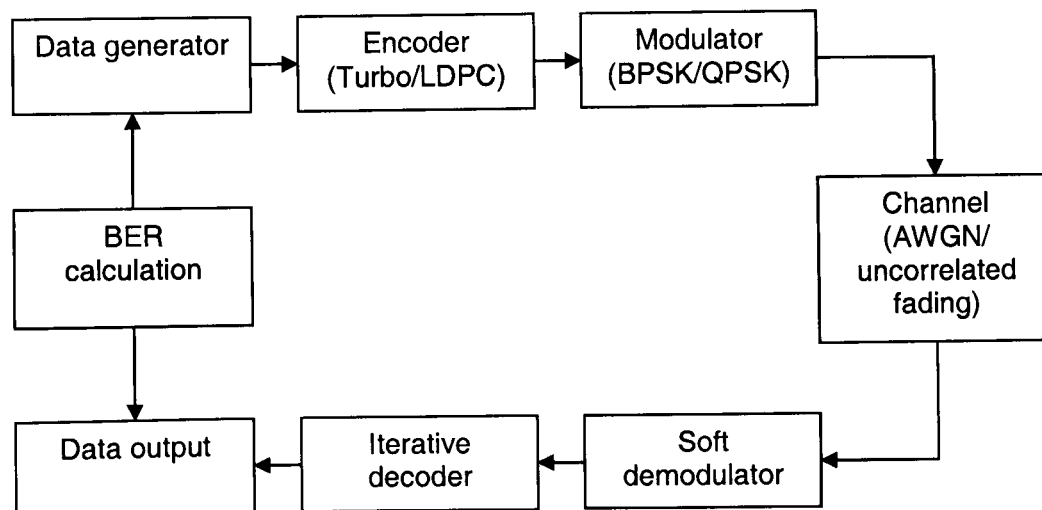


Figure 2.12: Computer simulation chain.

Lastly, the *LDPC* codes were based on *MacKay's* database [13]. In this case, regular codes are assumed with fixed column and row weight, i.e. $(d_s, d_c) = (3, 6)$, and also fixed coding rate, i.e. $R = 1/2$. The block size includes six cases. That is, $(N, K) = (96, 48), (504, 252), (816, 408), (1008, 504), (4000, 2000)$ and $(8000, 4000)$ respectively.

- **Modulator.** For binary turbo and LPDC codes, BPSK modulation is considered. The DVB-RCS turbo code assumes QPSK modulation. Note that the BER performance of the uncoded BSPK/QPSK is identical, although the latter scheme has greater spectral efficiency.
- **Channel.** In all cases of channel coding, the AWGN channel is deployed. In addition, binary turbo codes have been simulated over an uncorrelated (or fully interleaved) Rayleigh/Rician fading channel. Channel models can be also found in [71].
- **Soft demodulator.** In all cases of soft demodulation, the demodulated bit values are multiplied by the channel reliability value (L_c). The channel coefficient (α) is set to one in the uncorrelated fading channel. That is, no channel state information (CSI) is assumed at the receiver.
- **Iterative Decoder.** The SOVA, Max-Log-MAP and Log-MAP algorithms are supported in binary turbo codes. This is the basis for our work in Chapters 3, 4. The DVB-RCS turbo code can be decoded by the Max-Log-MAP, Log-MAP and Constant Log-MAP algorithms and has motivated the work in Chapter 5. In case of LDPC codes, the LLR-SPA based on both the *tanh* rule and *Gallager's* approach are considered. This has inspired our work in Chapter 6.

A fixed number of decoding iterations is employed in all cases of channel coding. In binary turbo codes, simulation results were obtained with up to 18 iterations. For the DVB-RCS turbo code, this number was reduced to 8, due to the medium block size of the code. In LDPC codes, up to maximum 200 iterations were considered in order to have a fair comparison in simulation time. It is noted that in the high SNR region, where the channel is considered to be in a good condi-

Table 2.1: Reference work on binary turbo codes for BER performance validation.

Reference work	Generator polynom.	Coding rate (R)	Frame size (bits)	Channel type	Decoding algorithm	Decoding iterations
<i>Berrou</i> [6]	$(1, 21/37)_o$ 16-states	1/2	65536	AWGN	MAP	1, 2, 3, 6, 18
<i>Robertson</i> [7]	$(1, 21/37)_o$ 16-states	1/2	100, 400, 1024	AWGN	Log-MAP, Max-Log-MAP	8
<i>Hanzo</i> [8]	$(1, 5/7)_o$ 4-states	1/3, 1/2	1000	AWGN	Log-MAP, Max-Log-MAP, SOVA	8
<i>Valenti</i> [9]	$(1, 15/13)_o$ 8-states	1/3	640, 5114	AWGN, Rayleigh	Log-MAP, Max-Log-MAP	10, 14
<i>Hagenauer</i> [10]	$(1, 5/7)_o$ 4-states, $(1, 21/37)_o$ 16-states	1/2	400, 1024	AWGN, Rayleigh	SOVA	8

tion, the number of decoding iterations can be reduced without any significant performance degradation.

- **BER calculation.** This is based on comparing the estimated bits to the generated ones. A total number of 50 million bits were generated or in other case, at least 100 bit errors were calculated in the high SNR region. In a similar way, the FER can also be reported.

2.7.1 Computer Simulated Performance Validation

The BER performance of *binary* turbo codes has been validated assuming the Log-MAP, Max-Log-MAP and SOVA algorithms. In Table 2.1 it is shown the reference work for performance validation with a certain number of different parameters. This is because binary turbo codes have been quite popular, since their announcement in 1993. Obtained computer simulated performance results and related comparison can be found in Appendix B.

Table 2.2: Reference work on duo-binary turbo codes for BER/FER performance validation.

Reference work	Encoder type	Coding rate (R)	Frame size (bits)	Channel type	Decoding algorithm	Decoding iterations
<i>Berrou</i> [3]	DVB-RCS	1/2, 2/3, 3/4	ATM, 424 MPEG, 1504	AWGN	improved Max-Log-MAP	8
<i>Kabal</i> [11]	DVB-RCS	1/3	ATM, 424 MPEG, 1504	AWGN	Log-MAP, Max-Log-MAP	8
<i>Yu</i> [12]	DVB-RCS	1/2, 2/3, 4/5	MPEG, 1504	AWGN	Max-Log-MAP	8

Table 2.3: Reference work on LDPC codes for BER/FER performance validation.

Reference work	Block size (N,K)	column/row weight (d_s, d_c)	Coding rate (R)	Channel type	Decoding algorithm	Decoding iterations
<i>MacKay</i> [13]	(96, 48) (816, 408) (4000, 2000)	(3, 6)	1/2	AWGN	SPA	variable
<i>Fossorier-1</i> [14]	(504, 252)	(3, 6)	1/2	AWGN	SPA	max. 1000
<i>Eleftheriou</i> [15]	(1008, 504)	(3, 6)	1/2	AWGN	LLR-SPA	max. 80
<i>Fossorier-2</i> [16]	(8000, 4000)	(3, 6)	1/2	AWGN	LLR-SPA	max. 100

The BER/FER performance validation of *duo-binary* turbo codes, such as in the DVB-RCS standard, is based on the Log-MAP and Max-Log-MAP algorithms, e.g. see Appendix B. The reference work for performance validation with different parameter values is shown in Table 2.2.

LDPC codes have been validated using the LLR-SPA. Reference work for performance validation includes different parameter values and is shown in Table 2.3. Obtained computer simulation results and related comparison is reported in Appendix B.

The effect of *different parameters* to the simulated turbo code performance, as in the *binary* case, is shown in Appendix C. Our motivation for this is based on a related

work from [8] and can be found useful to some of the readers.

2.7.2 Computer Simulated Performance Comparison

In this Section, the computer simulated performance obtained with the three assumed codes is compared to each other. The three following cases are shown

- Binary Turbo Codes - Duo-Binary Turbo Codes
- Binary Turbo Codes - LDPC Codes
- Binary Turbo Codes - Duo-Binary Turbo Codes - LDPC Codes

A comparison between binary and duo-binary turbo codes is shown in Figs. 2.13, 2.14. BER/FER results are reported for an 8-states *binary* turbo code with pseudo-random interleaver and BPSK modulation. The *duo-binary* turbo code is such as in the DVB-RCS standard with optimised interleaver and QPSK modulation. In both cases, the rest of the parameters are ATM (or MPEG) frame size, i.e. 424 (or 1504) bits, coding rates $R = 1/3$ or $1/2$, AWGN channel, Max-Log-MAP algorithm and 8 decoding iterations. It is noticed that for both the assumed coding rates, there is no need of puncturing to the duo-binary turbo code.

From Figs. 2.13 and 2.14 it is clear that duo-binary turbo codes outperform binary turbo codes in the high SNR region and exhibit no error floor. The performance improvement is approximately 1 dB at FER of 10^{-4} or equivalently at BER of 10^{-7} . Note that the comparison includes the same frame size, number of states, coding rate, decoding algorithm and number of decoding iterations.

Binary turbo codes are compared to LDPC codes in Figs. 2.15- 2.18. Different generator polynomials that have from 4 to 16-states and frame sizes (from 48 to 4000 bits) are shown in the BER/FER performance of binary turbo codes, which are decoded by the Log-MAP algorithm, after 10 decoding iterations. The BER/FER performance of LDPC codes with column and row weight $(d_s, d_c) = (3, 6)$ is shown for different block sizes (from (96, 48) to (8000, 4000)) using the SPA decoding algorithm from *Gallager's*

approach and either maximum 10 or a higher number of decoding iterations. In both cases, the rest of the parameters are coding rate $R = 1/2$ and the AWGN channel.

It can be seen that binary turbo codes with small or medium frame size perform up to 1 dB better than LDPC codes (in regular form) at BER of 10^{-4} or FER of 10^{-3} . As the frame/block size is increased, binary turbo codes exhibit an error floor and the performance difference becomes smaller. This has been already verified by the simulation comparison between binary and duo-binary turbo codes, as above. On the other hand, LDPC codes show the absence of error floor at any considered block size and can outperform binary turbo codes at BER lower than 10^{-6} or FER lower than 10^{-4} when a large block size is concerned. Similarly, the comparison includes the same frame/block size, coding rate, and an optimum decoding algorithm.

Assume now the case where both binary, duo-binary and LDPC codes are compared each other, such as in Figs. 2.19 and 2.20. The simulation parameters are identical to the case of binary turbo codes to LDPC codes comparison, but the frame/block size is now different. That is, binary turbo codes with frame size of either 408 or 2000 bits and LDPC codes with block size of either (816, 408) or (4000, 2000) respectively. On top of it, the DVB-RCS turbo code with either ATM (424 bits) or MPEG (1504 bits) frame size and Log-MAP decoding is concerned. This is done to have a fair comparison between all the coding schemes.

What is interesting from these Figures, it is that duo-binary turbo codes with medium frame size can overcome the error floor that is exhibited by binary turbo codes, so that they can perform better than LDPC codes (in regular form), e.g. within 1 dB at BER of 10^{-5} or FER of 10^{-4} . On the other hand, LDPC codes can outperform duo-binary turbo codes at BER lower than 10^{-7} or FER lower than 10^{-5} when a large block size is concerned. In a similar way, the comparison includes almost the same frame/block size, the same coding rate and an optimum decoding algorithm. Finally, our remarks are in agreement with [32]. In this Reference, it is also mentioned that duo-binary turbo codes have lower computational complexity than LDPC codes. However, the decoding of LDPC codes can be based on parallel architectures, so that the overall decoding complexity can be balanced.

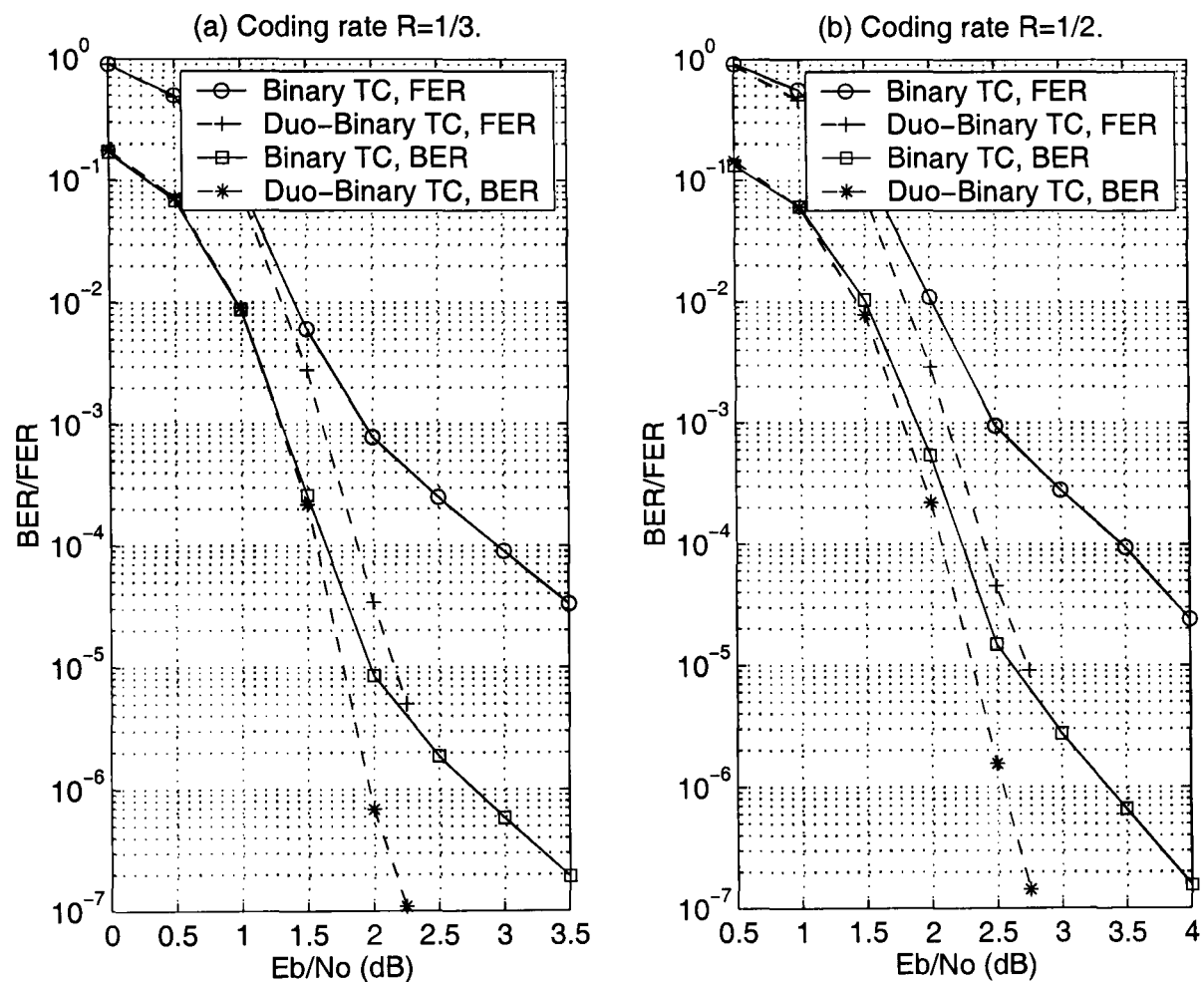


Figure 2.13: BER/FER comparison between binary (solid lines) turbo code with generator polynomials $(1, 13/15)_o$, i.e. 8-states, and duo-binary turbo code (dashed lines), such as in the DVB-RCS standard. ATM frame size, i.e. 424 bits, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations and different coding rates.

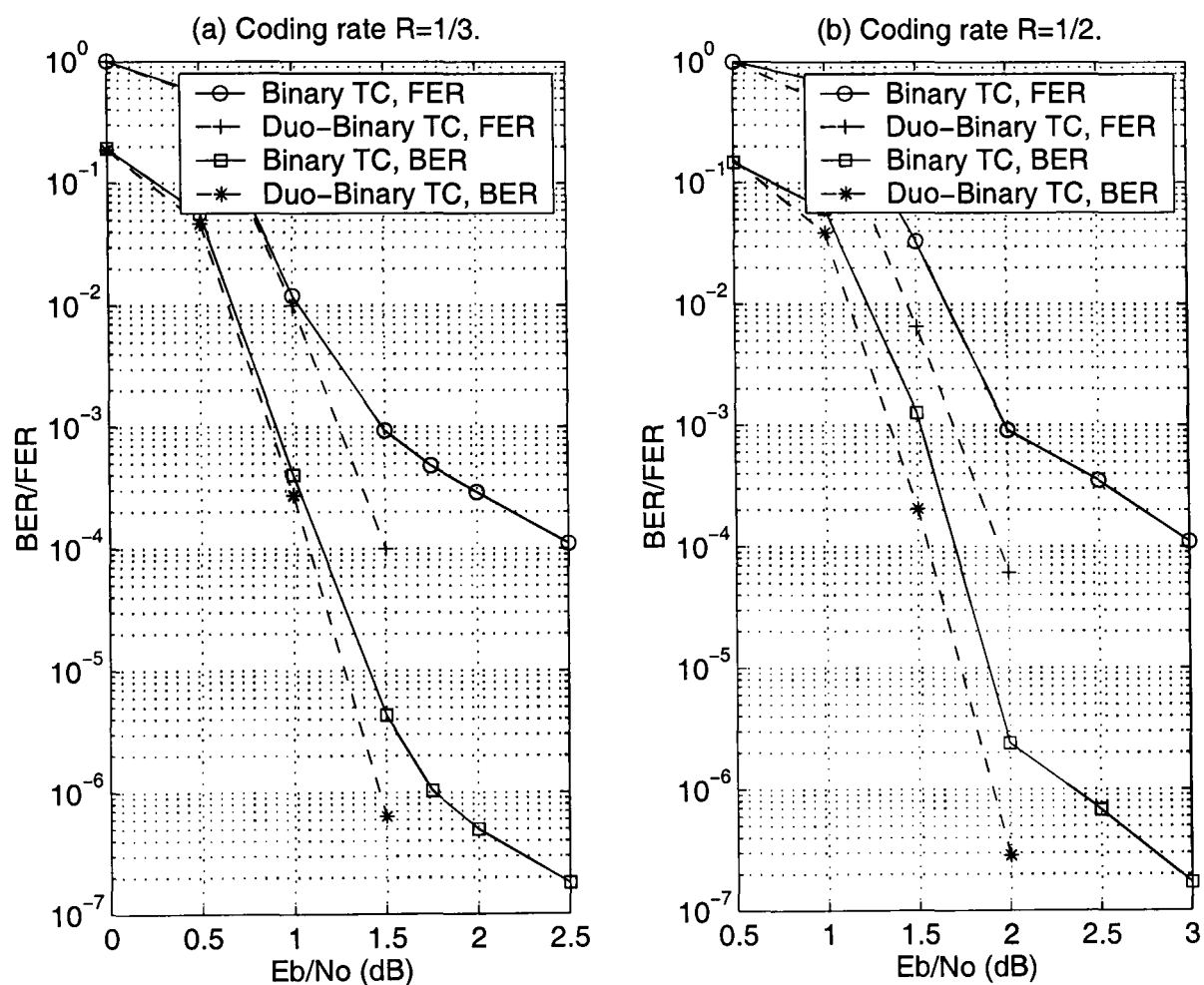


Figure 2.14: BER/FER comparison between binary (solid lines) turbo code with generator polynomials $(1, 13/15)_o$, i.e. 8-states, and duo-binary turbo code (dashed lines), such as in the DVB-RCS standard. MPEG frame size, i.e. 1504 bits, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations and different coding rates.

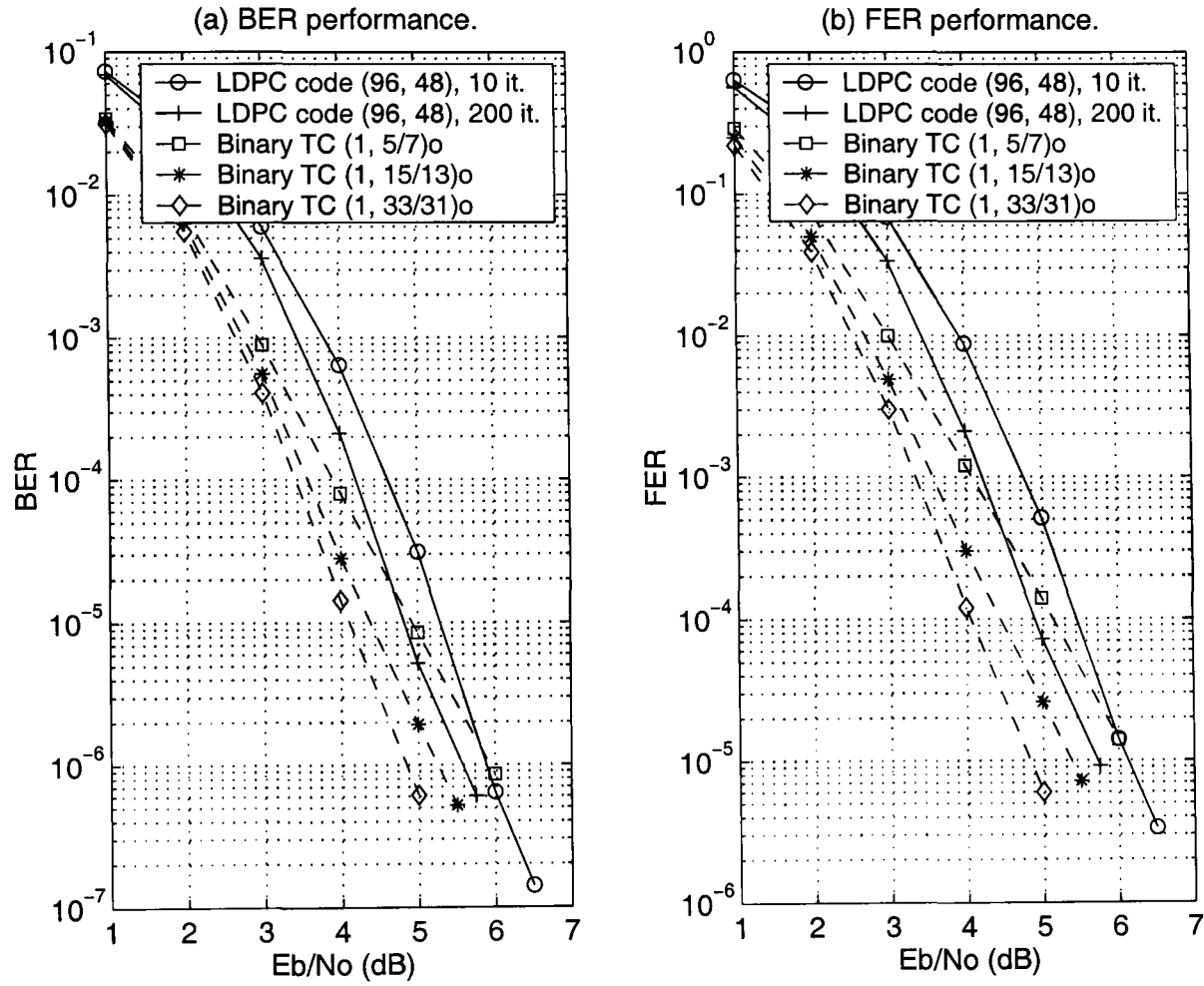


Figure 2.15: BER/FER comparison between (96, 48) LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from *Gallager's* approach and either maximum 10 or 200 decoding iterations. Turbo code, 48 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.

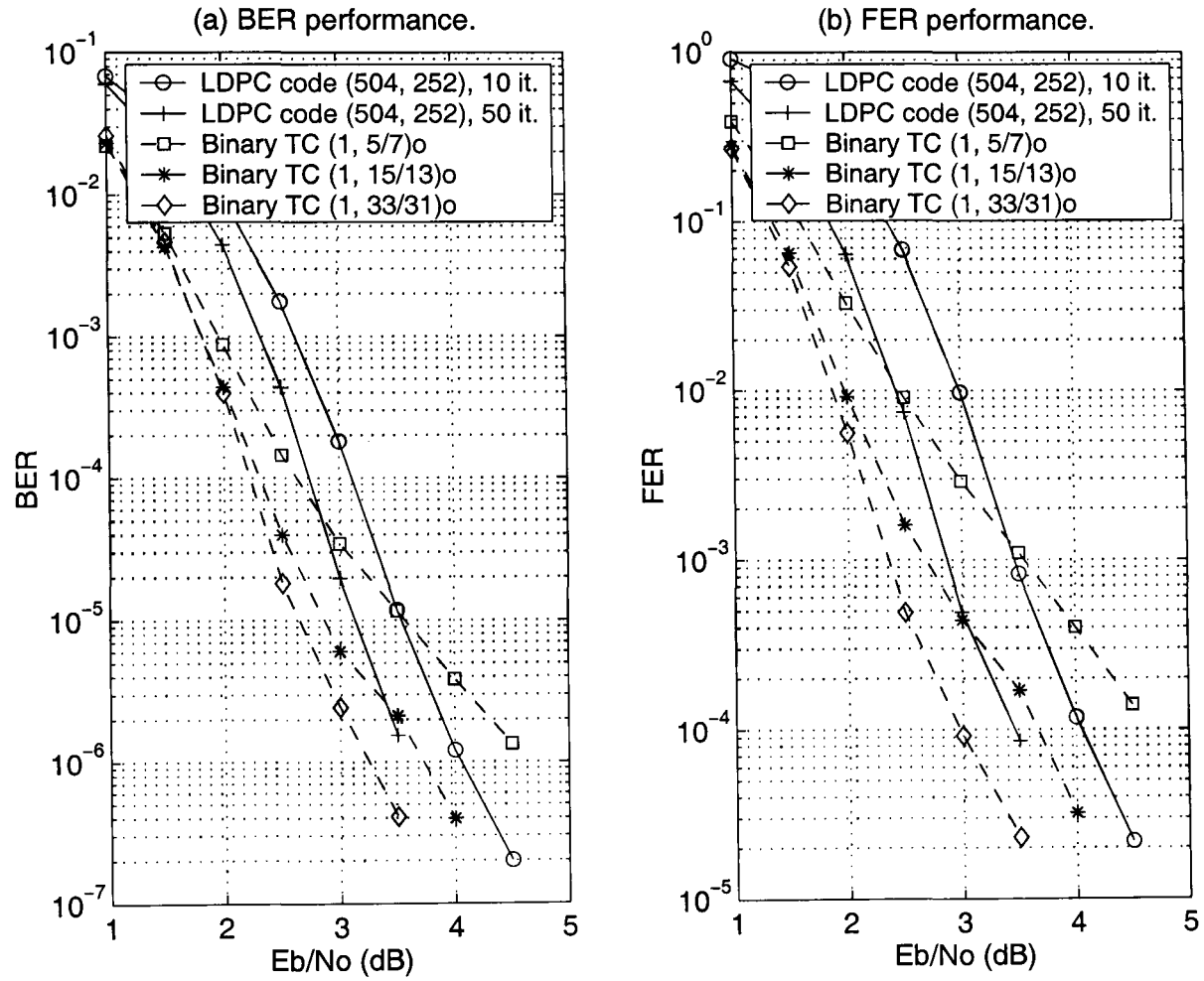


Figure 2.16: BER/FER comparison between (504, 252) LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from *Gallager's* approach and either maximum 10 or 50 decoding iterations. Turbo code, 252 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.

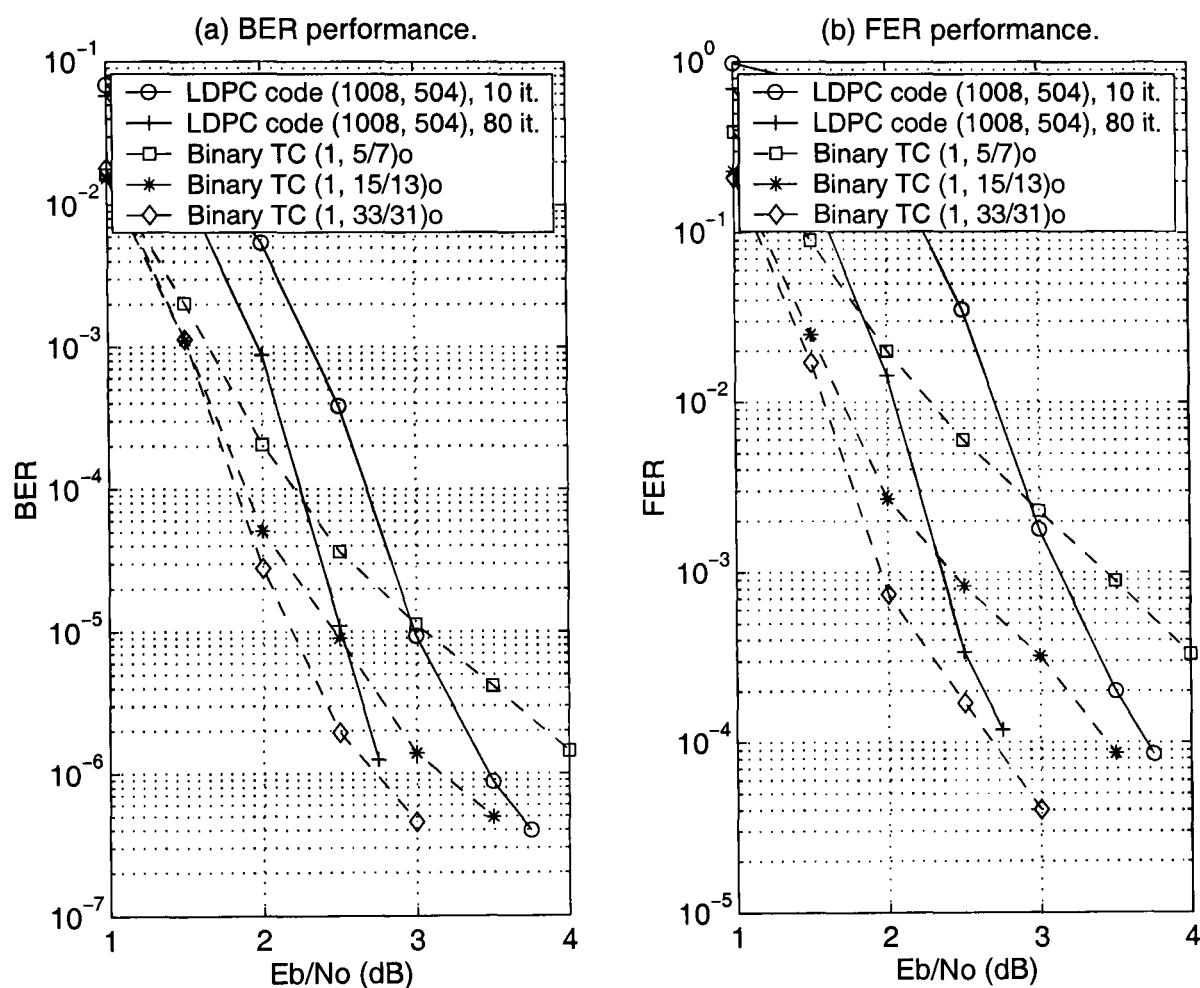


Figure 2.17: BER/FER comparison between (1008, 504) LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from *Gallager's* approach and either maximum 10 or 80 decoding iterations. Turbo code, 504 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.

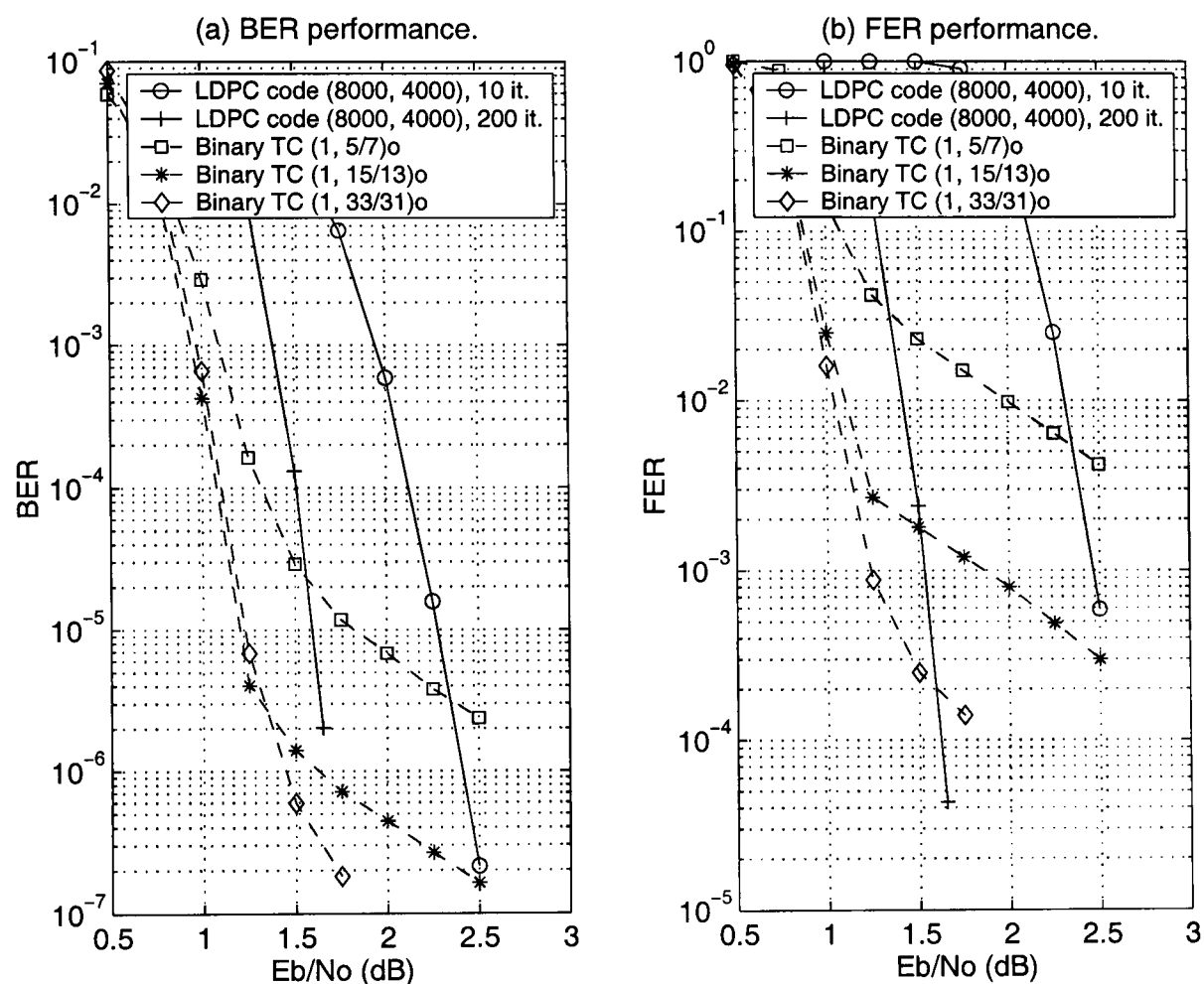


Figure 2.18: BER/FER comparison between (8000, 4000) LDPC code (solid lines) and binary turbo code (dashed lines) with different generator polynomials. LDPC code, SPA decoding algorithm from *Gallager's* approach and either maximum 10 or 200 decoding iterations. Turbo code, 4000 bits frame size, Log-MAP algorithm and 10 decoding iterations. In both cases, coding rate $R = 1/2$ and the AWGN channel.

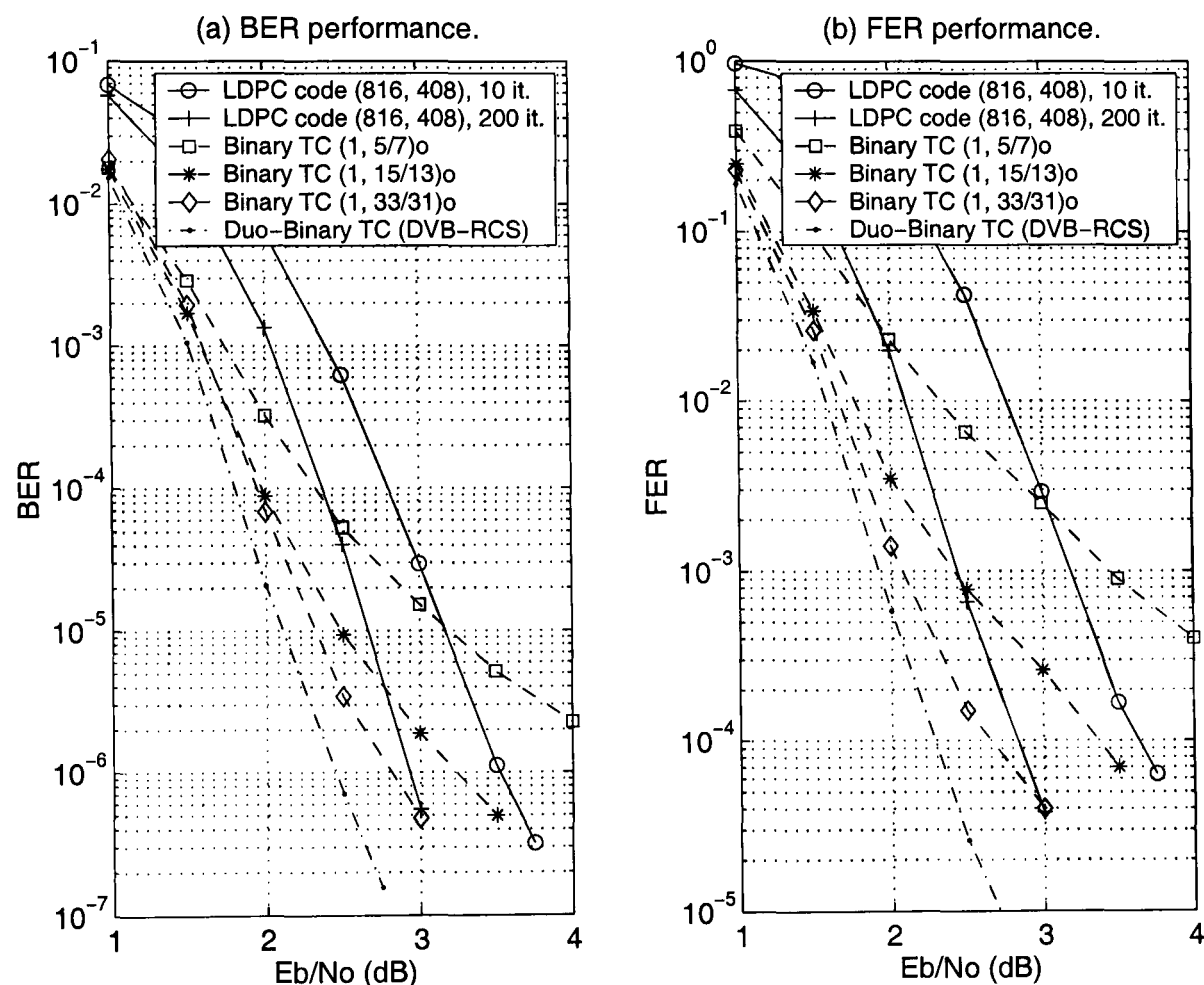


Figure 2.19: BER/FER comparison between (816, 408) LDPC code (solid lines), binary turbo code (dashed lines) with different generator polynomials and duo-binary turbo code (dashed-dotted line), such as in the DVB-RCS standard. LDPC code, SPA decoding algorithm from *Gallager's* approach and either maximum 10 or 200 decoding iterations. Turbo code, 408 bits frame size, Log-MAP algorithm and 10 decoding iterations. Duo-binary turbo code, ATM frame size, i.e. 424 bits, Log-MAP algorithm and 8 decoding iterations. In all cases, coding rate $R = 1/2$ and the AWGN channel.

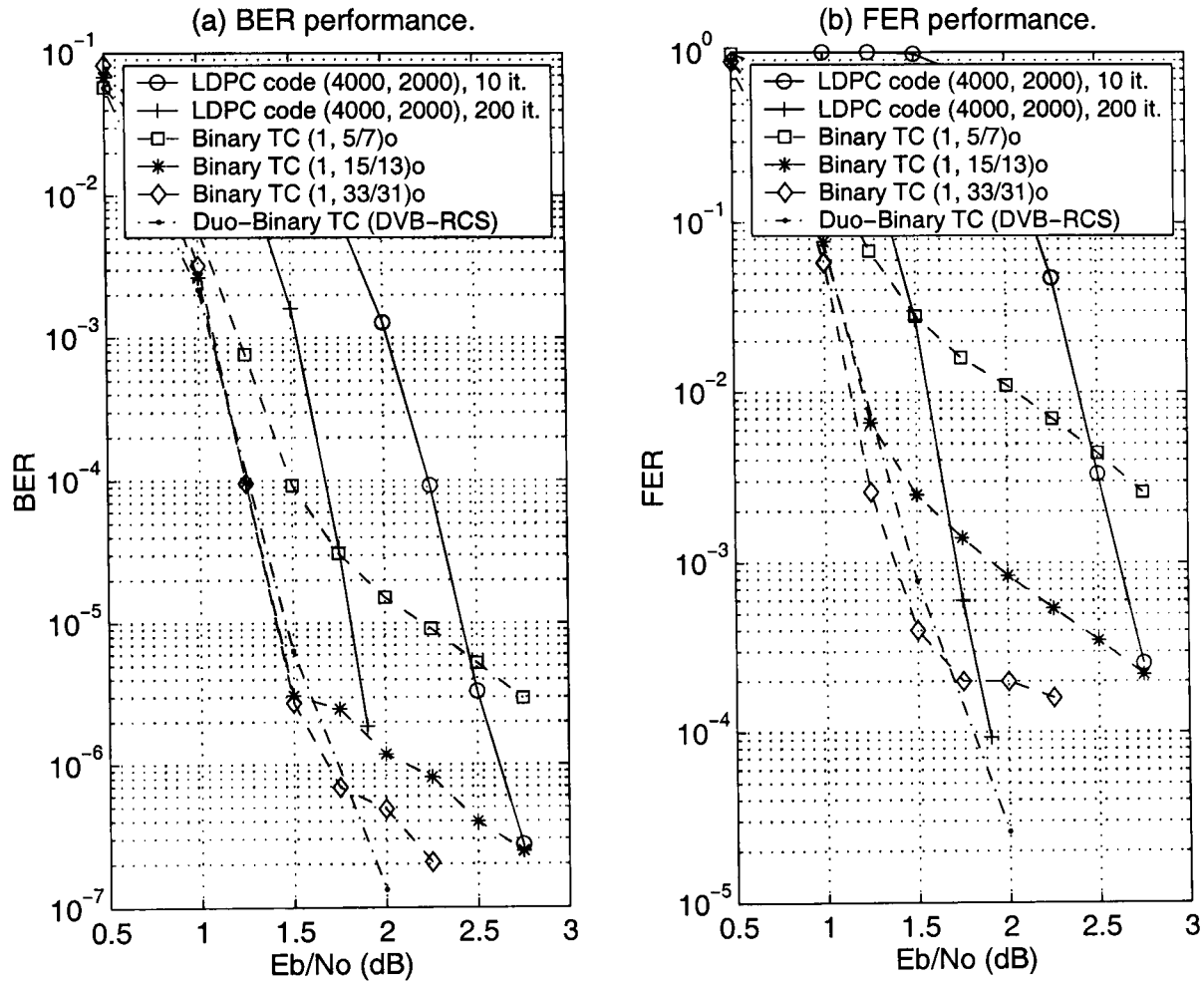


Figure 2.20: BER/FER comparison between (4000,2000) LDPC code (solid lines), binary turbo code (dashed lines) with different generator polynomials and duo-binary turbo code (dashed-dotted line), such as in the DVB-RCS standard. LDPC code, SPA decoding algorithm from *Gallager's* approach and either maximum 10 or 200 decoding iterations. Turbo code, 2000 bits frame size, Log-MAP algorithm and 10 decoding iterations. Duo-binary turbo code, MPEG frame size, i.e. 1504 bits, Log-MAP algorithm and 8 decoding iterations. In all cases, coding rate $R = 1/2$ and the AWGN channel.

2.8 Summary

The most important issues of this Chapter, which are related to the next presented Chapters, are highlighted.

- Classical (i.e. binary) turbo codes, although been known for more than ten years, have inspired a lot of research work, up to the existing days. This is due to the astonishing performance over the AWGN channel for large frame sizes and high number of decoding iterations.
- Duo-binary turbo codes have been proposed in order to reduce the error floor observed in binary turbo codes at lower BER values. Combined with two-levels of interleaving and circular trellis (or tail-biting) technique, it results in very powerful codes that show the absence of error floor at FER of 10^{-7} [3].
- LDPC codes can be seen as the strongest competitors to turbo codes, as they can also approach the *Shannon* limit over the AWGN channel. Naturally, they exhibit no error floor (in regular form) at low BER values and can be decoded in parallel, allowing high throughputs.
- SISO trellis-based decoding, such as the SOVA, MAP, Log-MAP and Max-Log-MAP algorithms have been reviewed. Also, the SPA algorithm using LLR values, suitable for decoding LDPC codes, was described.
- The performance of the above codes was evaluated and compared from each other. Excellent match between simulation results and relevant work is found, for different parameter values. It was verified that duo-binary turbo codes perform better than the binary ones. The latter codes are superior to LDPC codes at medium frame sizes and also medium BER values. In contrast, at lower BER values and large block sizes, LDPC codes are better than binary turbo codes and it seems that they are even better than duo-binary turbo codes.

Chapter 3

Improved SOVA Decoding for Binary Turbo Codes

This is the *first* of four Chapters where original work is introduced. Starting with classical (i.e. binary) turbo codes, a simple two-step approach of improving SOVA turbo decoder is proposed. The idea behind this is the scaling of the extrinsic information that is produced by the decoder output. Computer simulation results, run with various parameters, indicate that the error floor of the code can be reduced to lower BER values.

3.1 Introduction

Iterative SOVA decoding has attracted a lot of interest due to low complexity and relatively easy extension to the conventional VA. Low decoding complexity also allows high data throughputs to be achieved, resulting in SOVA-based iterative decoding of turbo codes being a strong candidate for future communication systems [72]. On the other hand, the main drawback of iterative SOVA is the sub-optimum BER performance against the MAP algorithm. This is the objective of *improved* iterative SOVA techniques; based on the conventional SOVA algorithm, it is possible to improve the BER performance of the code and, at the same time, to keep the decoding complexity low, with respect to Max-Log-MAP and Log-MAP iterative decoding.

In this Chapter, the SOVA decoder output based on *Hagenauer's* approach [57] is improved by scaling the extrinsic information with a constant factor that has two steps. In addition, a normalisation scheme is proposed that modifies the branch metrics computation, so as to avoid possible overflow of the decoder. Both the methods add very small computational complexity over the conventional iterative SOVA decoder.

3.2 SOVA Turbo Decoder Implementations

As mentioned in Section 2.3.1, SOVA has already been known before the invention of turbo codes, as an extension to the well-known VA [57]. The main goal is to produce, except for the maximum likelihood path sequence, a reliability value of each estimated bit. This is done by considering two trellis paths (i.e. best path and its strongest competitor path) to update the estimated reliability values, in contrast to the MAP algorithm where all trellis paths are considered. When SOVA was applied to iterative decoding, the performance degradation against the MAP turbo decoder was 0.7 dB at BER of 10^{-4} [7, 65], assuming BPSK signals over the AWGN channel. However, the advantage is that it is approximately three times less complex compared to the MAP turbo decoder [7].

Different approaches to SOVA decoder implementations can be found in [57, 58]. *Hagenauer* first proposed the SOVA algorithm in [57]. Coding gains of approximately 1 to 4 dB against the classical hard-decision VA were feasible when it was applied to either decoder, demodulator or equalizer. Moreover, relevant work on soft-decision VA-based decoding of convolutional codes had been done earlier by *Battail* [58]. In the rest of the Chapter, we refer to the SOVA decoder implementation from [57] as *HR-SOVA* and to the SOVA decoder implementation from [58] as *BR-SOVA*. The two algorithms differ in the way that the reliability bit values are estimated, as BR-SOVA stores in addition, the reliability values of the strongest competitor path.

In [61] it was shown that the BR-SOVA is equivalent to the Max-Log-MAP algorithm, but with decoding complexity savings when iterative decoding is applied. Furthermore, *Bi (Bidirectional)*-SOVA decoding for turbo codes was reported in [73], which makes use of updating the estimated reliability bit values twice, once in a forward and once in

a backward mode. The Bi-SOVA implementation based on the HR-SOVA, was shown to have performance close to the Max-Log-MAP iterative decoder, but with reduced decoding complexity [73]. Finally, a *List* Bi-SOVA algorithm suitable for turbo codes was presented in [74]. In this case, the decoder soft-output was computed by using more than one pair of path metrics. As shown in [74], this algorithm can approach the BER performance of MAP iterative decoding, despite being less complex.

In [75, 76, 5] it was shown that the iterative BR-SOVA is 0.5 dB superior to the HR-SOVA at BER of 10^{-4} , assuming BPSK signals over the AWGN channel. An overall performance/complexity comparison between BR-SOVA, HR-SOVA and List-SOVA for turbo decoding can be found in [77]. Performance evaluation of the Bi-SOVA for serial concatenated convolutional codes was reported in [78]. Finally, the Bi-SOVA with a scaling factor of the extrinsic information was shown to perform 0.2 dB better than Max-Log-MAP turbo decoding at BER of 10^{-4} in [79], assuming BPSK signals over the AWGN channel.

3.3 Relevant Work on Improved SOVA Turbo Decoder

The conventional SOVA (i.e. HR-SOVA) is considered in this Section as well as in the next Sections. This is because other SOVA decoder implementations require extra memory storage and, as a consequence, the decoding complexity is increased. For instance, BR-SOVA needs to store the reliability of the strongest competitor path, while Bi-SOVA processes over two modes, i.e. forward and backward. In contrast, it is possible to improve the HR-SOVA turbo decoder by simple techniques, so as to achieve BER performance close to Max-Log-MAP or even to Log-MAP iterative decoding. The general concept of improving the HR-SOVA turbo decoder is shown in Fig. 3.1, where a kind of normalisation of the extrinsic information is performed.

3.3.1 Fundamental Approaches

In the literature there exist two fundamental approaches on the improved SOVA turbo decoder by *Papke et al* [80] and *Lin et al* [5] respectively. In the first approach [80],

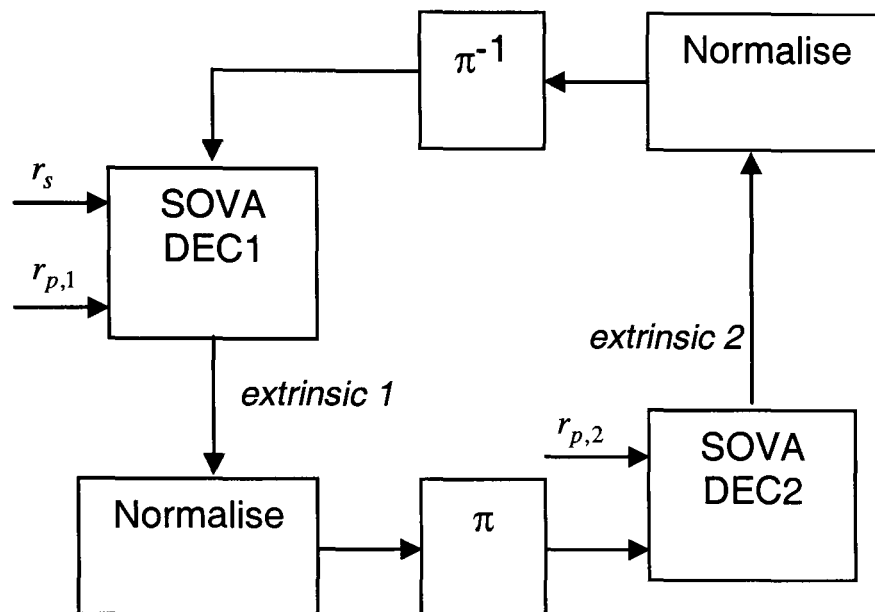


Figure 3.1: Improved (normalised) SOVA turbo decoder.

the decoder soft-output is corrected based on the *Gaussian* assumption distribution. This is done in two ways. First, in every decoding iteration the extrinsic information is multiplied by a constant factor (i.e. scaling) that depends on the variance of the decoder output. In a second adjustment, the correlation of the decoder input is eliminated by adding two more correcting coefficients, however this gives less performance improvement. In the second approach [5], which has inspired our research work contribution, the reliability values of the decoder output are limited into a smaller range. This is based on observing the absolute reliability values of both the HR/BR-SOVA against the number of decoding iterations. In the following, the two fundamental approaches are described in more detail.

Scaling Based on the Decoder Output Statistics

Assume an information sequence of bits, denoted by u , a BPSK modulated sequence, denoted by x , an AWGN channel with noise variance σ_n^2 , followed by a SOVA decoder. By using LLR values, the encoder input is multiplied by the channel reliability value $L_c = 2/\sigma_n^2$. It can be proved that the decoder output, denoted by v , is also *Gaussian* distributed [80]. The conditional LLR at the decoder output, given the observation of the SOVA output is

$$LLR = L(\hat{x}) = L(x|v) = \ln \frac{P(x = +1|v)}{P(x = -1|v)} \quad (3.1)$$

Using Bayes' rule and assuming $P(x = +1) = P(x = -1)$ we have

$$L(\hat{x}) = \ln \frac{P(v|x = +1)}{P(v|x = -1)} = \ln \left\{ \exp \left(-\frac{1}{2\sigma_v^2} \left[(v - m_v)^2 - (v + m_v)^2 \right] \right) \right\} = m_v \frac{2}{\sigma_v^2} v \quad (3.2)$$

or

$$L(\hat{x}) = cv, \quad \text{where } c = 2m_v/\sigma_v^2 \quad (3.3)$$

where m_v and σ_v^2 are the mean and variance of the decoder output respectively. Ideally, $m_v = \sigma_v^2/2$ but this does not occur, especially in low SNR values (i.e. bad channels). In this situation, the factor c is found to be less than one, in contrast to the MAP algorithm, where c is always equal to one [80]. From Eq. (3.3) it is concluded that the SOVA output v should be multiplied by the factor c . As this is less than one, SOVA is regarded to be too *optimistic* in the reliability estimation. Following this approach, coding gain improvement of approximately 0.3 dB at BER of 10^{-4} can be observed, assuming a memory four turbo encoder and BPSK signals over the AWGN channel [80].

Reducing the Correlation Effects

Another feature of SOVA decoder is the *correlation* that is observed between the intrinsic (i.e. LLR channel values plus *a priori* information of systematic bits) and the extrinsic information during the iterative process. For example, a typical value of correlation is 0.3, depending also on the available SNR value [80]. Generally, in the high SNR region the correlation decreases. Little correlation is observed between intrinsic and extrinsic information in case of MAP iterative decoding.

By modelling the two LLRs of intrinsic (i.e. $L_1 = L(\hat{x})_I$) and extrinsic (i.e. $L_2 = L(\hat{x})_E$) information as *Gaussian* random variables with mean m_{L_1} and m_{L_2} , variance $\sigma_{L_1}^2$ and $\sigma_{L_2}^2$ and correlation ρ respectively, it can be proved that the conditional LLR is [80]

$$L(\hat{x}) = L(x|L_1, L_2) = \ln \frac{P(x = +1|L_1, L_2)}{P(x = -1|L_1, L_2)} = L_1 + L_2' = L_1 + (\alpha L_2 + \beta L_1) \quad (3.4)$$

where

$$\alpha = \frac{1 - \rho \frac{\sigma_{L_1}}{\sigma_{L_2}}}{1 - \rho^2} \quad \text{and} \quad \beta = \frac{\rho^2 - \rho \frac{\sigma_{L_2}}{\sigma_{L_1}}}{1 - \rho^2} \quad (3.5)$$

This approach adds two more coefficients (i.e. α and β) over the standard SOVA iterative decoder, in order to correct the correlation between intrinsic and extrinsic information. That is, multiplication of the extrinsic information by α and multiplication of the intrinsic information by β . Following this approach, the coding gain improvement is approximately 0.1 dB at BER of 10^{-4} , assuming a memory four turbo encoder and BPSK signals over the AWGN channel [80].

In overall, although the required computations in order to find the correcting factor c and also the correcting coefficients α and β add extra decoding complexity, this approach is straightforward. Simulation results have shown that the improved SOVA turbo decoder can approach the Log-MAP decoding with a degradation up to 0.3 dB at BER of 10^{-4} , assuming a memory four turbo encoder and BPSK signals over the AWGN channel [80]. In other words, 0.4 dB of coding gain improvement is observed over the standard SOVA turbo decoder.

Limiting the Reliability Values

In [5] it was observed that the iterative HR-SOVA produces larger reliability values at the decoder output, compared to the corresponding values of the iterative BR-SOVA. That is, due to its updating process, the iterative HR-SOVA overestimates the decoder reliability values. This concept is shown in Fig. 3.2 of Section 3.4.1. Computer simulation results in [5] have also shown that the BR-SOVA is 0.5 dB superior to the HR-SOVA at BER of 10^{-4} for a rate 1/3 turbo code with block interleaver.

The improved iterative HR-SOVA from [5] is based on limiting the range of the path reliability (Δ), i.e. the path metric difference between best path and survivor path. This is done by defining an optimum threshold value (Δ_{TH}), as

$$\text{if } \Delta > \Delta_{TH}, \text{ then } \Delta = \Delta_{TH} \quad (3.6)$$

For example, it was found by trial and error that the best threshold value is $\Delta_{TH} = 4$. The idea behind this is to keep both the decoder soft-output and extrinsic information values small, during the first few iterations. The improved iterative SOVA scheme can provide 0.5 dB of extra coding gain with respect to the conventional HR-SOVA, assuming a memory four turbo encoder over the AWGN channel. This scheme can also approach very close or even perform better than the iterative BR-SOVA [5].

3.3.2 Latest Research Work

Based on the fundamental approaches on improved SOVA turbo decoder a lot of research has been done mainly to reduce the implementation complexity of the normalisation schemes, whilst improving the BER performance significantly.

In [81] a digital signal processing (DSP)-based SOVA implementation was described with data rate of 10 Kbps that outperforms the existing *Viterbi* decoder of the NASA standard. In this case, a scaling factor c was used that increases linearly with the number of decoding iterations, e.g. $c = 0.5 + 0.05 \cdot i$, where i is the current number of decoding iteration.

The two methods from [80, 81] were compared with each other in [82]. As shown in [82], the performance degradation of the reduced complexity method proposed in [81] against the method proposed in [80] was approximately 0.1 to 0.2 dB at BER of 10^{-5} for a memory three turbo encoder, assuming BPSK signals over the AWGN channel. Two first order polynomials in the general form of $c = a + b \cdot i$ were used, where the selection of a and b depends on the E_b/N_o value [82]. Moreover, it was found that it is not necessary to normalise both the decoder outputs, but only one of them. Following the approach from [80], the performance degradation when only one decoder output is normalised, was approximately 0.15 dB at BER of 10^{-5} for a memory three turbo encoder, assuming BPSK signals over the AWGN channel [82]. In addition, there was no significant difference to the BER performance when the second decoder was

normalised by a constant factor Z . That is, normalise the first decoder as $c_1 = 2m_v/\sigma_v^2$ and for the second decoder choose either $c_2 = 1$ or $c_2 = Z$ respectively, depending on the available computational complexity at the decoder [82].

The concept of normalising both the decoder outputs by a constant factor, i.e. $c_1 = c_2 = Z$, was introduced in [83, 84]. In addition, a hardware implementation of the SOVA turbo decoder with modified architecture was described, which is capable of both area and power consumption savings. Following this approach, two important issues can be highlighted. First, the improved SOVA performs approximately 2 dB better than the conventional SOVA at BER of 10^{-4} for a rate 1/3 turbo code, assuming BPSK signals over the AWGN channel. Second, the BER performance is slightly better than the approach from [80], where the decoder output statistics are used. Typical values of scaling factor were reported to be $Z = 0.25$ and $Z = 0.33$.

Two new normalisation methods as well as finite precision simulation results and practical implementation issues for very large scale integration (VLSI)-based decoders were described in [85, 72]. The first method was based on three-point pseudo-median filtering techniques [85]. The basic idea was to relate the reliability values at time index k to the reliability values at neighbouring time indices $(k - 1)$ and $(k + 1)$ respectively and then define upper and lower bounds to the reliability values. In this case, the modified SOVA can improve the BER performance against the conventional SOVA to 0.2 dB for a wide range of SNR values, assuming a 4-states turbo code with coding rate $R = 1/2$ and BPSK signals over the AWGN channel. Furthermore, in the second method a scaling factor was used based on the number of matching bits within a block (i.e. using a mapping function) between the signs of the reliability estimation and the extrinsic information. The scaling factor was composed of at least two steps, one being constant and the other being linearly increased. That is, $c = Z$ or $c = a + b \cdot m$, depending on the mapping function (m). The modified SOVA was shown to reduce the BER performance gap from MAP iterative decoder to 0.2 dB for a wide range of SNR values, assuming the same simulation parameters as previously. In [72] the linear function used as scaling factor was quantized to five levels and also adaptive thresholding was applied, similar to [5]. The performance of the resulting improved SOVA turbo decoder was only 0.1 dB inferior to MAP turbo decoder at medium BER val-

ues, assuming a 4-states turbo code with coding rate $R = 1/2$ and BPSK signals over the AWGN channel. However, two normalisation approaches were combined together, adding extra computational complexity at the decoder.

Another method of improving the SOVA turbo decoder was proposed for PCCC in [86] and also for SCCC in [87]. Based on the approach from [80], two attenuators were employed to reduce the correlation effects between intrinsic and extrinsic information. The two attenuators were calculated either analytically or they were fixed. One of them was applied directly to the decoder output and the other one was applied after the calculation of the extrinsic information. By computer simulation results of a 16-states PCCC with coding rate $R = 4/5$ [86], it was shown to be a performance improvement of approximately 0.8 to 1 dB in case of the AWGN channel and also performance improvement of 1.4 to 2 dB over an uncorrelated Rayleigh fading channel, both at BER of 10^{-5} . In both cases, the proposed improved SOVA turbo decoder performs very close to the MAP turbo decoder, however two attenuators instead of one are required for this method.

Finally, new SISO decoding algorithms that improve the BER performance of the SOVA turbo decoder were reported in [77]. The basic concept was proper path collection *tuning* between the *conventional* SOVA (i.e. HR-SOVA), *modified* (M)-SOVA (i.e. BR-SOVA) and *path-augmented* (PA)-SOVA (i.e. List-SOVA) and then *max/max** operation replacement to the reliability estimation for further trade-off between BER performance and complexity. This may result in a unified approach to trellis-based SISO decoding algorithms, suitable for a programmable turbo decoder implementation. Computational complexity comparison among the proposed SISO algorithms was also shown [77]. The best algorithm, denoted by *Max*-M-SOVA*, was up to 0.1 dB inferior to the MAP iterative decoding at BER of 10^{-5} , assuming a rate 1/3 turbo code and BPSK signals over the AWGN channel.

3.4 Proposed Method

Here, an original method is explained to improve iterative SOVA decoding of turbo codes. This is based on simple two-step approach on scaling factor of the extrinsic

information and is described in more detail below.

3.4.1 Motivation

Referring to Fig. 3.2, a simple graphical representation of both the average absolute reliability value and the average absolute extrinsic information of the HR/BR-SOVA respectively is given against the number of decoding iterations. This was reported in [5] where a normalisation method for the HR-SOVA turbo decoder was proposed. The over-estimated reliability values of the HR-SOVA in the first few decoding iterations can also be observed with respect to the BR-SOVA turbo decoder. For example, the two curves in [5] and thus in Fig. 3.2, cross over in the fourth decoding iteration assuming a rate 1/3 turbo code with 16-states over the AWGN channel.

By carefully observing Fig. 3.2, another normalisation method can be deduced, which can be regarded as the inverse of the method described above. It seems that *after* a certain number of decoding iterations, both the average absolute reliability value and the average absolute extrinsic information of the HR-SOVA become smaller than those of the BR-SOVA. Thus, after a certain number of decoding iterations, the reliability values of the HR-SOVA can be increased, so as to approach the corresponding values of the BR-SOVA.

3.4.2 Simple Two-Step Approach

A simple normalisation method of the iterative HR-SOVA can be obtained where the extrinsic information is increased during the *last* decoding iteration only [88, 89]. It can be regarded as a simple two-step approach with fixed scaling factor.

Assume that i and N are the current and total number of decoding iterations respectively. The extrinsic information that is passed from one component decoder to the other one can be normalised, if it is multiplied by a constant number, such that

$$\text{keep } c[i] = Z, \text{ until } i \leq N \quad (3.7)$$

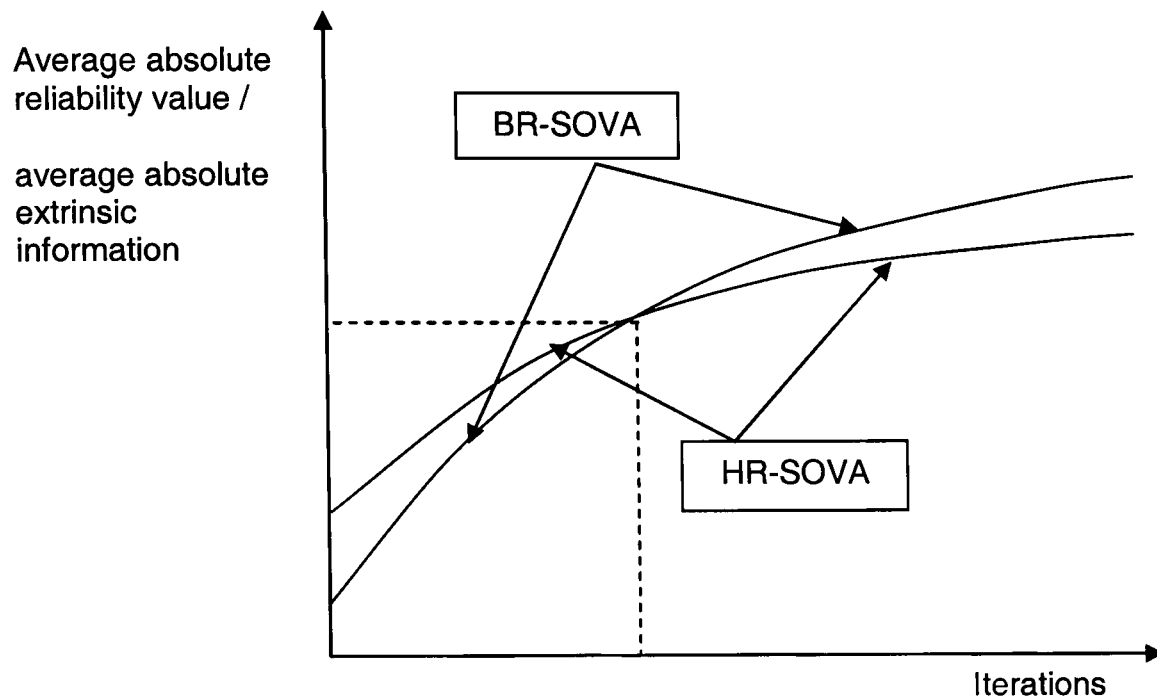


Figure 3.2: BR/HR-SOVA graphical comparison from *Lin et al* [5].

This approach was described in [83, 84] and it is referred to as *norm1* method. If the scaling factor is further increased in the last decoding iteration, then the proposed *norm2* method is obtained, as

$$\begin{aligned} &\text{keep } c[i] = Z, \text{ until } i \leq N - 1 \\ &\text{and } c[i] > Z, \text{ when } i = N \end{aligned} \quad (3.8)$$

The process of finding the best values of Z for different turbo encoders and coding rates is based on trial and error. However, the values do not depend on the E_b/N_o value nor the channel type. Optimised values of Z using the *norm1* method are shown in Table 3.1 of Section 3.5.1. As reported in the same Section, an empirical rule is followed to obtain the best values of Z using the *norm2* method.

3.4.3 Modified Branch Metrics

At time instant k , assume a bit transition between two states of a trellis path. Then, the corresponding *branch metric* of the standard VA is defined as

$$\lambda_k \triangleq \sum_{i=0}^{n-1} (r_{k,i} - x_{k,i})^2 \quad (3.9)$$

where $1/n$ is the code rate, n is the codeword length, $x_{k,i}$ is the i -th transmitted bit, assuming BPSK signals and $r_{k,i}$ is the corresponding value at the receiver. This approach was reported in [74].

Early computer simulation experiments have shown that an extra block had to be added after the demodulator output and also after the computation of the extrinsic information in the iterative decoding process. This extra block divides all the input values by their maximum. In other words, it performs a kind of normalisation for numerical stability reasons. As the processing channel values are less than one, in absolute form, the corresponding value of $x_{k,i}$ at the receiver should be modified to $x_{0k,i}$, where $0 < x_{0k,i} \leq 1$ [90]. Thus, Eq. (3.9) becomes

$$\lambda_k = \sum_{i=0}^{n-1} (r_{k,i} - x_{0k,i})^2 \quad (3.10)$$

By trial and error, we have found the best parameter values of x_0 for different turbo encoders and coding rates, which are reported in Table 3.1 of Section 3.5.1.

3.5 Computer Simulation Results

The general block diagram of the normalised SOVA iterative decoder used in computer simulation results has already been shown in Fig. 3.1 of Section 3.3. Best parameter values selection and BER performance evaluation/comparison is reported first and then some more performance evaluation results are shown.

3.5.1 Best Parameter Values

The best found values of scaling factor (Z) and modified branch metrics (x_0) are summarised in Table 3.1. It is assumed four different generator polynomials and two coding rates (R). This applies for the the *norm1* method. In case of the *norm2* method, it

Table 3.1: Best found values of scaling factor (Z) and modified branch metric (x_0), assuming two coding rates (R) and different turbo code generator polynomials, using *norm1* SOVA.

(Z, x_0)	$R = 1/3$	$R = 1/2$
$(1, 5/7)_o$ 4-states	(1.95, 0.23)	(1.85, 0.24)
$(1, 15/13)_o$ 8-states	(1.85, 0.23)	(1.71, 0.25)
$(1, 21/37)_o$ 16-states	(1.55, 0.23)	(1.80, 0.24)
$(1, 33/31)_o$ 16-states	(1.53, 0.23)	(1.41, 0.24)

was found that the best value of Z during the last decoding iteration had to be around twice the value of *norm1*, while the value of x_0 is kept constant in both *norm1/norm2* cases. It is noted that the parameter values (Z, x_0) were evaluated in case of the AWGN channel. In addition, they were found to perform well in case of an uncorrelated fading channel.

As an example, Fig. 3.3 depicts the impact of the parameter value x_0 to the BER performance of the $(1, 15/13)_o$ turbo encoder with coding rate $R = 1/3$. It is assumed different E_b/N_o values, 1000 bits frame size, AWGN channel, *norm1* SOVA algorithm (with $Z = 1.85$) and 8 decoding iterations. It can be shown that the value of $x_0 = 0.23$ gives the best BER performance. This is related to the best value that matches the modified branch metrics computation from Eq. (3.10). After having set the best value of x_0 , small variations to the BER performance are observed by changing the parameter Z .

Simulation Performance Evaluation

In the following Fig. 3.4 the BER performance of the four assumed turbo encoders with the *norm1* SOVA (shown in solid lines) is compared to the Log-MAP performance (shown in dashed lines). The rest of the parameters are; coding rate $R = 1/3$, frame size 1000 bits, AWGN channel and 8 decoding iterations.

It is noticed that the *norm1* SOVA performs very close to the Log-MAP algorithm, although being much less complex. For example, it is found that the best performed

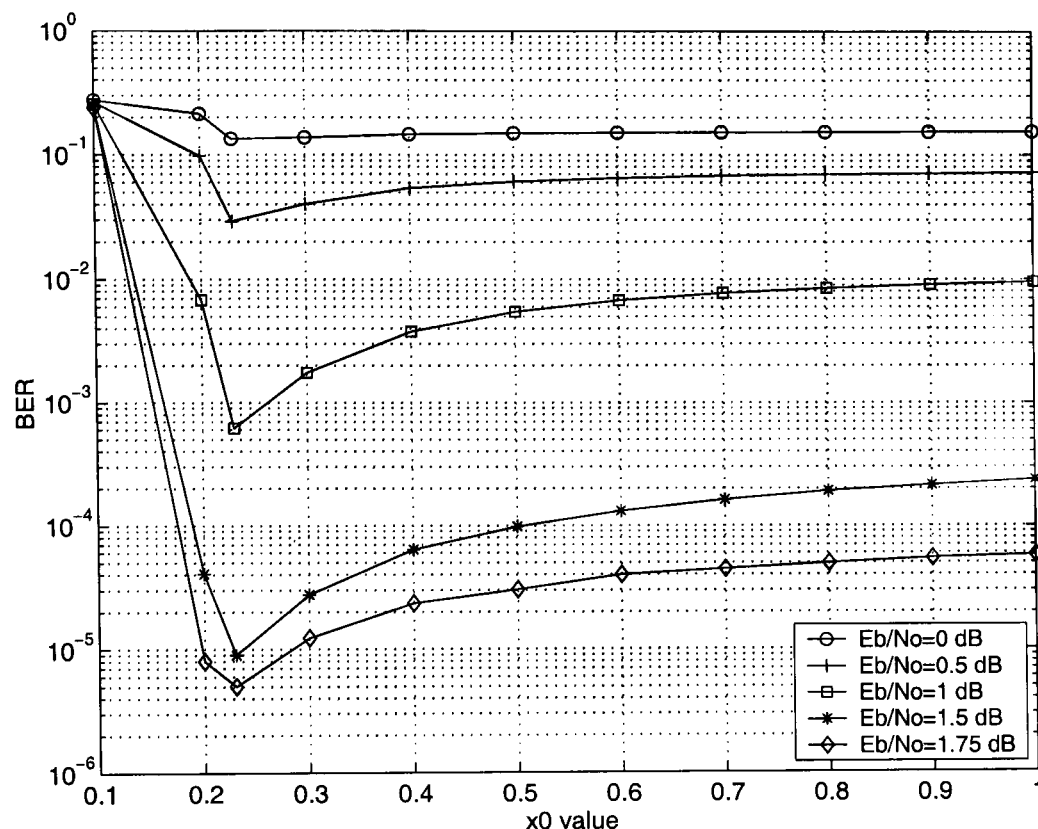


Figure 3.3: Impact of the parameter x_0 to the turbo code BER performance for different E_b/N_o values. $(1, 15/13)_o$ turbo encoder, coding rate $R=1/3$, 1000 bits frame size, *norm1* SOVA algorithm and 8 decoding iterations in the AWGN channel.

turbo encoder with the *norm1* SOVA is the $(1, 33/31)_o$, which performs as close as 0.2 dB compared to Log-MAP turbo decoding at BER of 10^{-4} .

In Table 3.2 we report the required E_b/N_o value at BER of 10^{-4} for different turbo code generator polynomials with two coding rates and three iterative decoding algorithms; *norm1* SOVA, Max-Log-MAP and Log-MAP, using linear *interpolation* method. The rest of the parameters are 1000 bits frame size, AWGN channel and 8 decoding iterations.

It is noticed that except for the 4-state turbo encoder, the *norm1* SOVA performs either *identically* to (e.g. for most of the rate half codes) or *better* than the Max-Log-MAP algorithm (e.g. for rate third codes). In addition, the decoding complexity savings of the *norm1* SOVA are crucial when it is compared to the Max-Log-MAP algorithm.

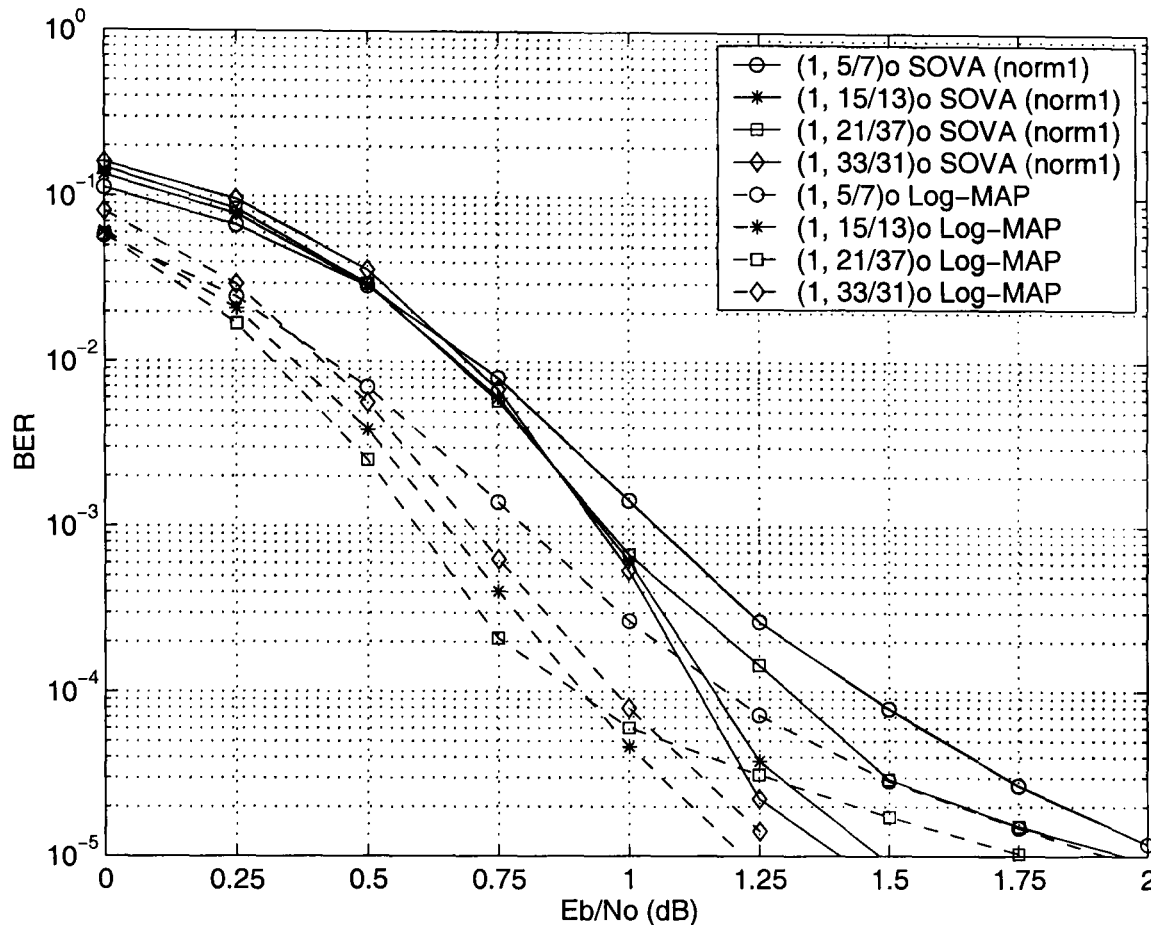


Figure 3.4: Normalised SOVA (solid lines) and Log-MAP (dashed lines) iterative decoding performance comparison for different generator polynomials, coding rate $R=1/3$, 1000 bits frame size and 8 decoding iterations in the AWGN channel.

Computing the Correlation Coefficient

As reported in [80], in iterative SOVA decoding the soft-output is affected by a correlation between the extrinsic and intrinsic information (i.e. LLR channel values plus *a priori* information of systematic bits). Since the extrinsic information is fed forward as *a priori* information to the next decoding iteration, it would degrade the resulting BER performance. In general, the correlation coefficient between intrinsic and extrinsic information decreases, as E_b/N_o improves. This is because the iterative SOVA decoder makes more accurate estimates of the transmitted sequence of bits. In case of the iterative MAP algorithm, an approximate value of zero can be observed, independently of the E_b/N_o value. Inspired by this phenomenon, we try to compute the correlation coefficient of the second SOVA decoder using the `corrcoef` command in *MATLAB*®.

Table 3.2: Required E_b/N_o value at BER of 10^{-4} using *norm1* SOVA, Max-Log-MAP and Log-MAP algorithms, 1000 bits frame size, 8 decoding iterations in the AWGN channel. Different turbo code generator polynomials are assumed with coding rate either $R=1/3$ or $R=1/2$.

	SOVA (<i>norm1</i>)	Max-Log-MAP	Log-MAP
R=1/3			
(1, 5/7) _o 4-states	1.45 dB	1.40 dB	1.20 dB
(1, 15/13) _o 8-states	1.15 dB	1.25 dB	0.90 dB
(1, 21/37) _o 16-states	1.30 dB	1.40 dB	0.90 dB
(1, 33/31) _o 16-states	1.15 dB	1.35 dB	0.95 dB
R=1/2			
(1, 5/7) _o 4-states	2.10 dB	1.95 dB	1.85 dB
(1, 15/13) _o 8-states	1.85 dB	1.85 dB	1.55 dB
(1, 21/37) _o 16-states	1.85 dB	1.85 dB	1.50 dB
(1, 33/31) _o 16-states	1.80 dB	1.95 dB	1.55 dB

Table 3.3: Correlation coefficient between intrinsic and extrinsic information of the second decoder against the E_b/N_o value, using the standard SOVA, *norm1/norm2* SOVA. Turbo encoder (1,15/13)_o, coding rate $R=1/3$, 1000 bits frame size in the AWGN channel and 8 decoding iterations.

	$E_b/N_o=1.5$ dB	$E_b/N_o=1.75$ dB
SOVA (<i>no norm</i>)	0.1168	0.1049
SOVA (<i>norm1</i>)	0.0660	0.0617
SOVA (<i>norm2</i>)	0.0544	0.0510

In Fig. 3.5 the correlation coefficient is shown using the standard SOVA (i.e. no normalisation) and the *norm1* SOVA against different E_b/N_o values. The (1,15/13)_o turbo encoder is assumed with coding rate $R = 1/3$ and 1000 bits frame size in the AWGN channel, after either 2 or 8 decoding iterations. In addition, similarly to Fig. 3.5, the correlation coefficient using the *norm2* SOVA is reported in Table 3.3, but after 8 decoding iterations.

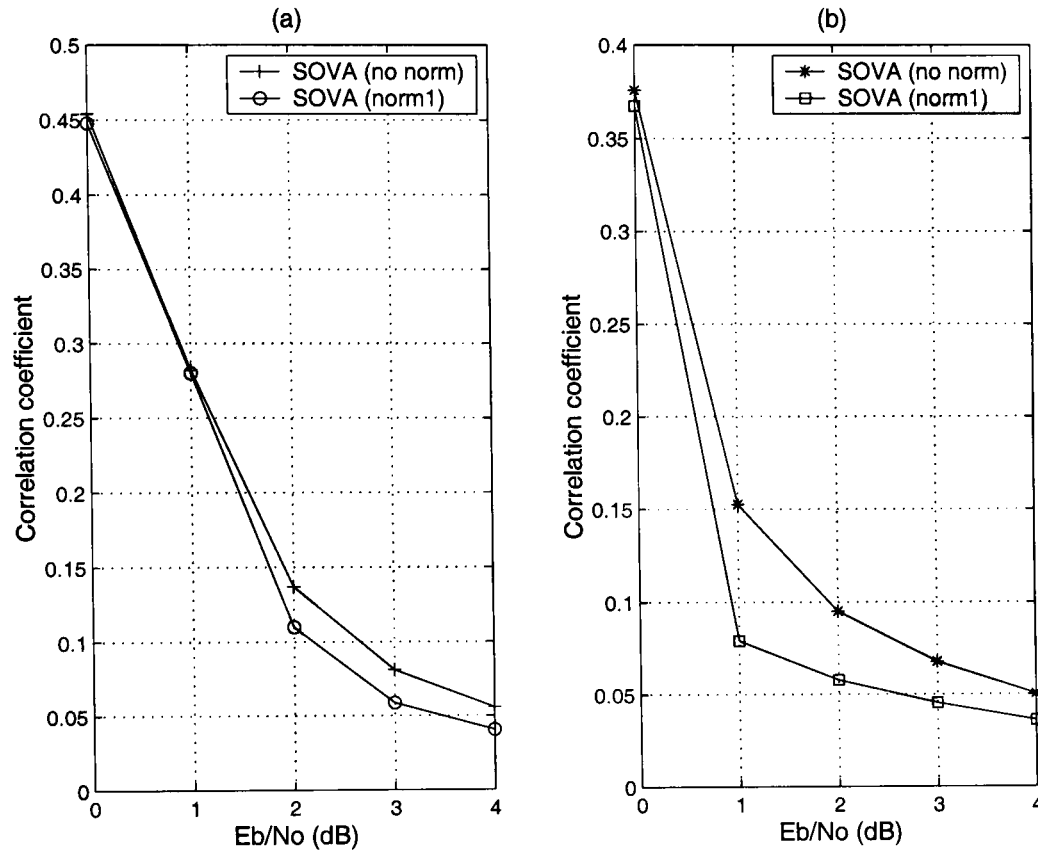


Figure 3.5: Correlation coefficient between intrinsic and extrinsic information of the second decoder against the E_b/N_o value, using standard SOVA and *norm1* SOVA. Turbo encoder $(1, 15/13)_o$, coding rate $R=1/3$, 1000 bits frame size in the AWGN channel. (a) 2 decoding iterations, (b) 8 decoding iterations respectively.

From both Fig. 3.5 and Table 3.3 it is observed that the better the BER performance of the code the smaller the correlation coefficient. It is thus concluded that the correlation coefficient can be regarded as another tool, in order to predict the BER performance behaviour of a code for given E_b/N_o value. In this way, the superior performance of the *norm2* SOVA compared to the *norm1* SOVA is verified. Finally, it is noted that similar results on correlation coefficient value of the iterative SOVA decoder have been reported in [80].

3.5.2 Simulation Performance Comparison

In the following Fig. 3.6, the *norm1*/*norm2* SOVA are compared to the normalised SOVA from [80, 82] respectively. In [80] a scaling factor was used based on the decoder

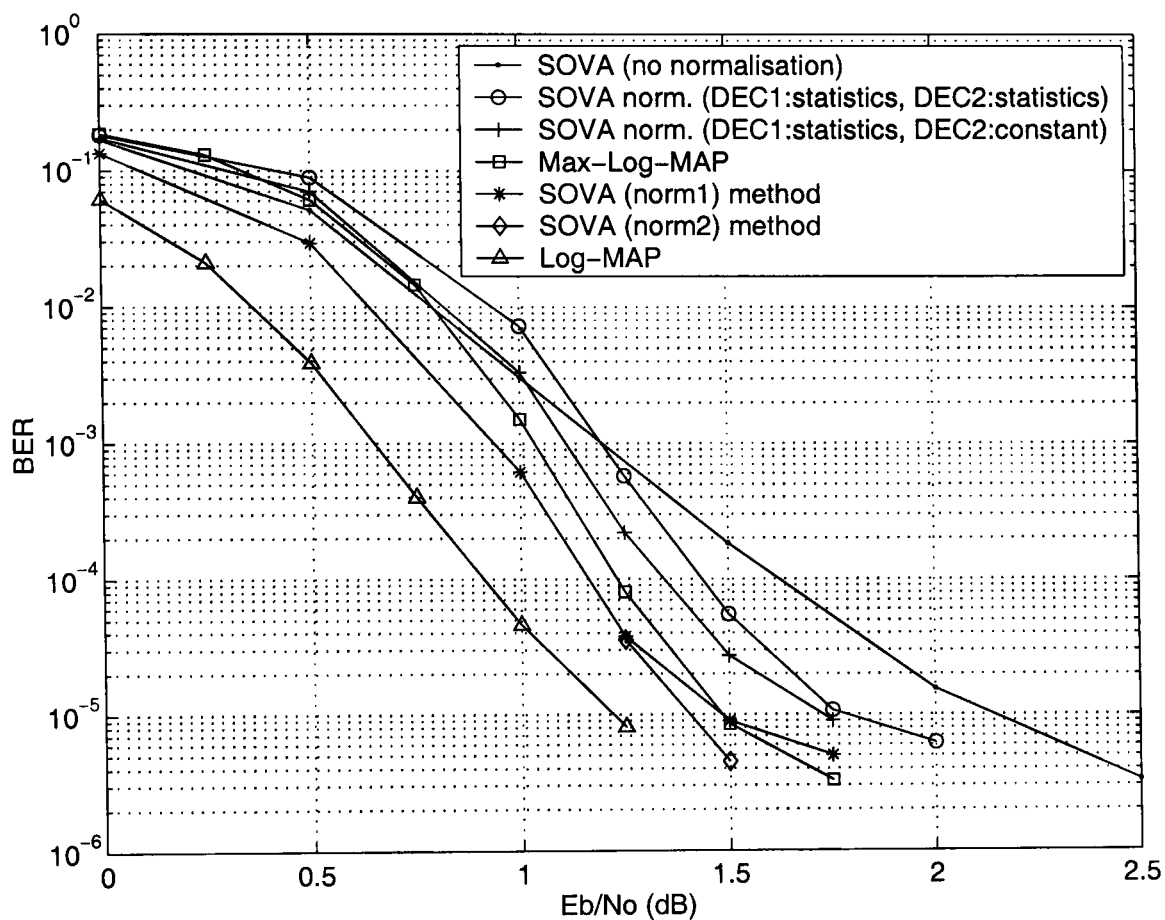


Figure 3.6: BER performance comparison of different normalised iterative SOVA algorithms. Coding rate $R=1/3$, 1000 bits frame size and 8 decoding iterations in the AWGN channel.

soft-output statistics, while in [82] one decoder was normalised as in [80] and the other one used a constant value for scaling. For comparison, the BER performance of standard SOVA (i.e. no normalisation), Max-Log-MAP and Log-MAP iterative decoding is also shown. The simulation parameters are turbo encoder $(1, 15/13)_o$, coding rate $R=1/3$, 1000 bits frame size, AWGN channel and 8 decoding iterations.

Referring to Fig. 3.6, the normalised SOVA using soft-output statistics performs 0.3 dB better than the standard SOVA at BER of 10^{-5} . This is in agreement with [80]. It is also verified that the standard SOVA is 0.7 dB inferior to Log-MAP iterative decoding at BER of 10^{-4} , similar to [7, 80]. In addition, there is 0.1 dB difference between the normalised SOVA using soft-output statistics in both decoder outputs and the normalised SOVA using soft-output statistics in one decoder output and a constant value in the other decoder output. Meanwhile, in [82] was reported that

approximately the same BER performance occurs. This can be explained because of a different interleaver, i.e. prime interleaver, used in [82].

Norm1 SOVA performs 0.4 to 0.5 dB better than the standard SOVA at BER between 10^{-4} and 10^{-5} . This is in agreement with most of the improved SOVA decoding algorithms, e.g. see [5, 77, 80, 82], [85]- [87]. *Norm2* SOVA provides an extra coding gain of 0.25 dB at BER less than 10^{-5} with respect to the *norm1* SOVA. Furthermore, the *norm1* SOVA is 0.25 dB inferior to the Log-MAP iterative decoding and 0.1 dB better than the Max-Log-MAP iterative decoding at BER of 10^{-4} . Similar results of improved SOVA iterative decoding compared to Max-Log-MAP and Log-MAP algorithms were reported in [80, 82].

In order to show results with different coding rates, it is assumed two rate half turbo

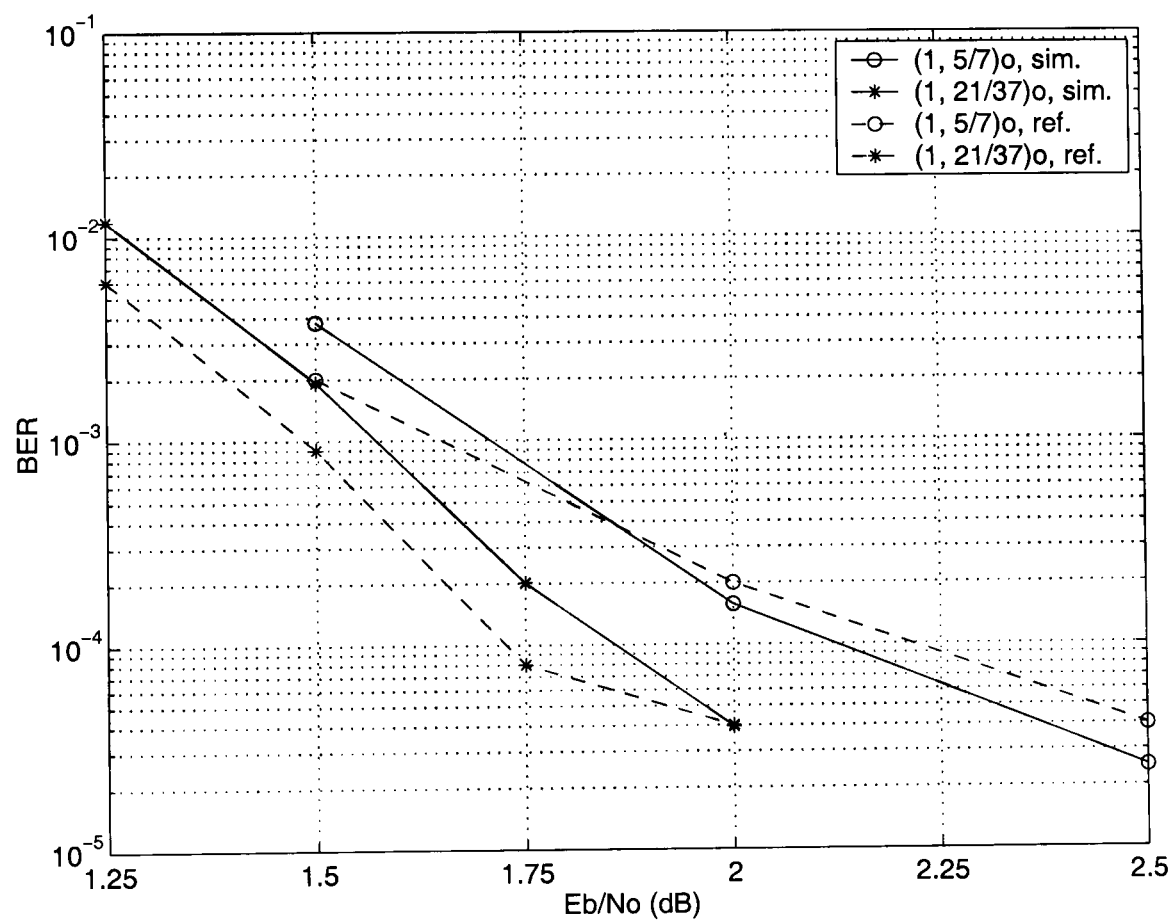


Figure 3.7: BER performance of different turbo encoders using *norm1* SOVA (solid lines) and reference performance comparison (dashed lines). Coding rate $R=1/2$, 1000 bits frame size and 8 decoding iterations in the AWGN channel.

encoders with 4 and 16 states, i.e. $(1, 5/7)_o$ and $(1, 21/37)_o$, respectively. BER performance comparison using the *norm1* SOVA and improved SOVA from [80] are shown in Fig. 3.7. The other important parameters are 1000 bits frame size, AWGN channel and 8 decoding iterations.

From Fig. 3.7 it is noticed that the *norm1* SOVA performs slightly better than the SOVA approach from [80] at BER lower than 10^{-4} when considering the 4-state turbo encoder. In addition, it has identical performance at BER values of around 10^{-5} when considering the 16-state turbo encoder.

3.5.3 More Computer Simulation Results

In this Section some more computer simulation results of the improved iterative SOVA decoder are given for different frame sizes and channel types. The $(1, 15/13)_o$ turbo encoder is assumed with coding rate $R = 1/3$ and 18 decoding iterations.

AWGN Channel

The effect of the *norm1/norm2* SOVA for large frame sizes (i.e. 5114, 10000 and 65536 bits) in the AWGN channel is shown in Figs. 3.8-3.10. The interleaver used in Fig. 3.8 is the one from the 3GPP standard [62], while it is a pseudo-random one in the rest of the cases. For comparison, the BER performance of the standard SOVA (i.e. no normalisation), Max-Log-MAP and Log-MAP iterative decoding is also shown.

It is noticed that the *norm2* SOVA provides a maximum coding gain of 0.7 dB at BER of 10^{-6} compared to the *norm1* SOVA. In addition, no error floor is observed at BER of 10^{-6} with the *norm2* SOVA, similar to Max-Log-MAP and Log-MAP algorithms. The performance degradation against the Log-MAP algorithm is 0.3 dB at the same BER value. The coding gain improvement of the *norm1/norm2* SOVA with respect to the Max-Log-MAP algorithm is 0.2 dB at BER of 10^{-4} . In addition, the coding gain improvement of the *norm1/norm2* SOVA with respect to the standard SOVA is 0.7 dB at BER of 10^{-4} .

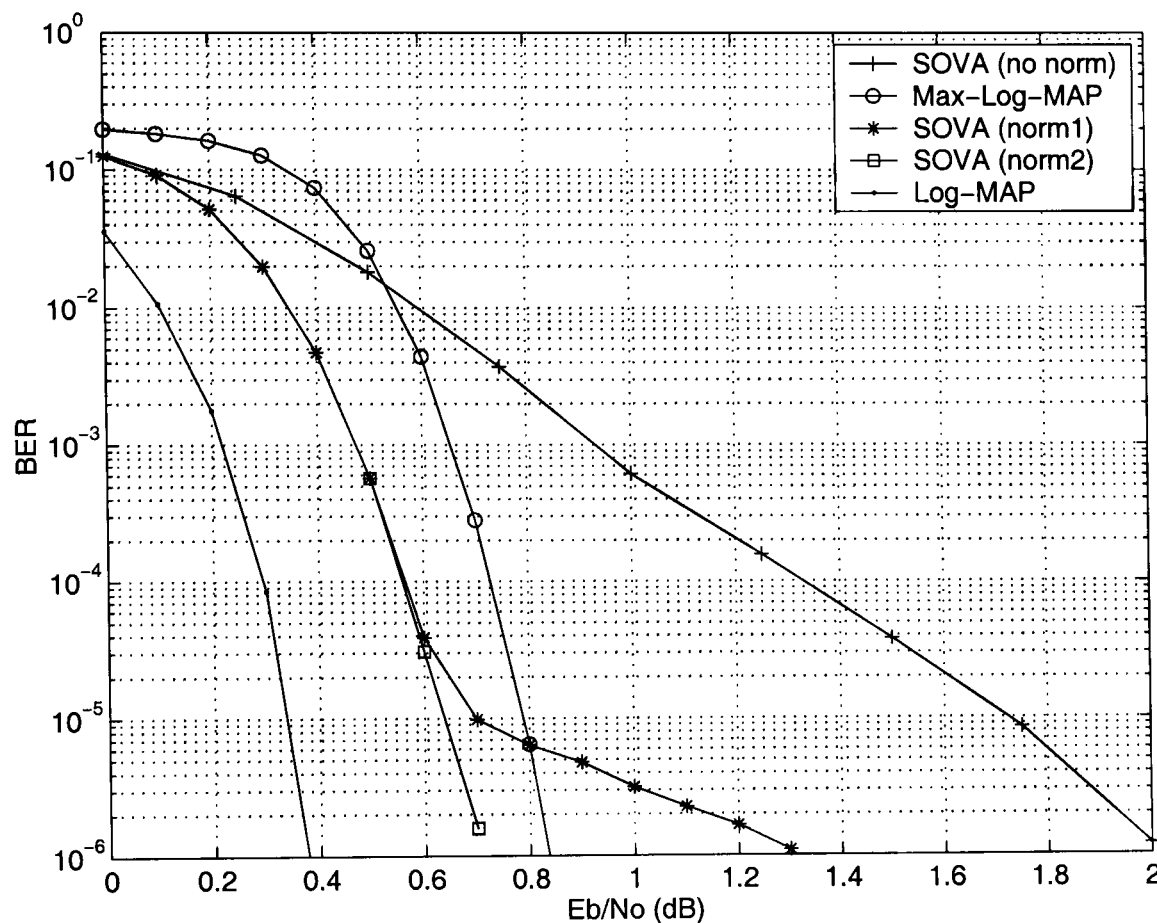


Figure 3.8: BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 5114 bits frame size, 3GPP interleaver and 18 decoding iterations in the AWGN channel.

In order to clarify the BER results from Figs. 3.8-3.10, we try to compare with other available results. In case of Fig. 3.8, similar BER performance between Log-MAP and Max-Log-MAP decoding was reported in [91], but after 8 decoding iterations. A 16-state turbo encoder was used in [77] with 16384 bits frame and after 10 decoding iterations. BER results with Log-MAP, Max-Log-MAP and PA-SOVA are in agreement with the results from Fig. 3.9. Finally, a 16-state turbo encoder was used in [92], but after 20 decoding iterations. The Log-MAP performance is close to the performance from Fig. 3.10.

The same Z value is used in all cases of frame size in Figs. 3.8-3.10. This may not be the optimum one in the case of 65536 bits frame (i.e. Fig. 3.10), explaining the small increase in the slope of the normalised SOVA BER curves.

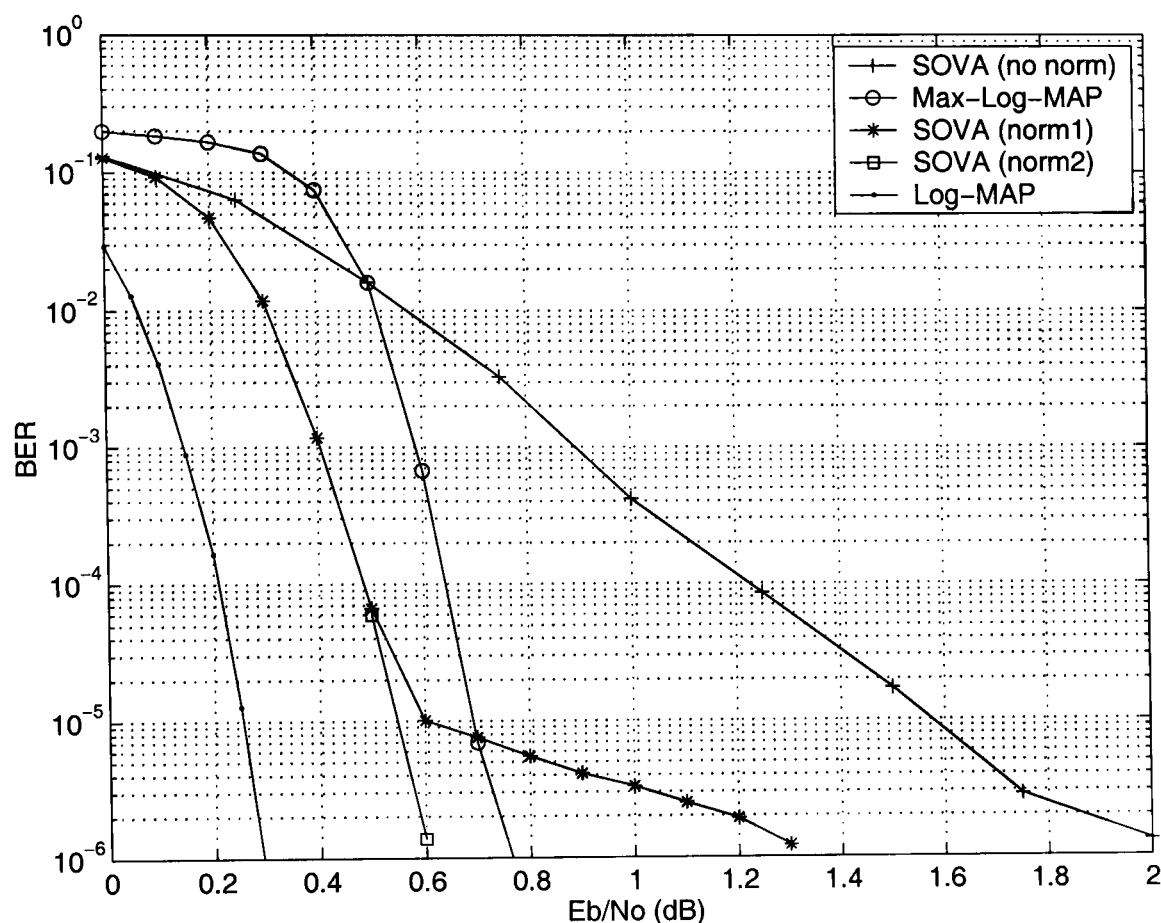


Figure 3.9: BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 10000 bits frame size and 18 decoding iterations in the AWGN channel.

Uncorrelated Rayleigh/Rician Fading Channel

In the following Fig. 3.11, it is shown the BER performance of the *norm1*/*norm2* SOVA over either an uncorrelated Rayleigh or Rician fading channel with different Rice factor values K . It is assumed the case of 10000 bits frame size. The decoding process is performed without knowledge of the channel (i.e. no CSI is available). For comparison, the BER performance of the *norm1*/*norm2* SOVA over the AWGN channel is also plotted from Fig. 3.9.

From Fig. 3.11 it is noticed that there is the same BER performance behaviour of the *norm2* against the *norm1* SOVA with respect to either an uncorrelated fading channel or the AWGN channel. That is, the performance improvement is independent of the channel type. Furthermore, the same parameter values of (Z, x_0) are used, as in the AWGN channel case. It is also verified that as the Rice factor K increases, the BER

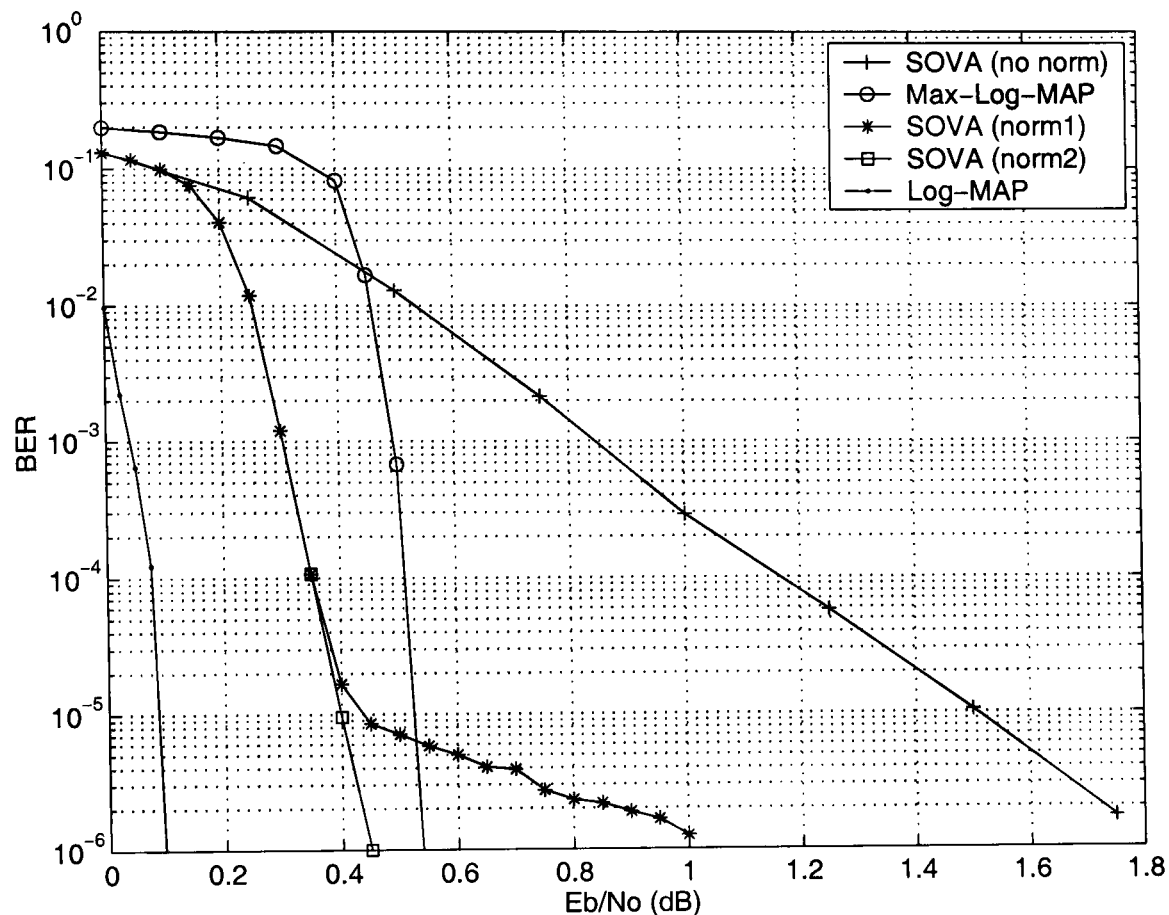


Figure 3.10: BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 65536 bits frame size and 18 decoding iterations in the AWGN channel.

performance of the code improves. By comparing the BER performance of the *norm2* SOVA in the AWGN channel and an uncorrelated Rayleigh fading channel, it is noticed that there is 2 dB degradation at BER of 10^{-6} . This performance degradation with no CSI available is acceptable [1].

3.5.4 Discussion

Usually, the correcting factor is less than one (e.g. see [72], [80]-[84] and [86]-[87]), in contrast to the approach presented here. This can be explained by the normalisation process that was described in Section 3.4.3. That makes the resulting values at the decoder input to be less than one, in absolute form. Therefore, a correcting factor that is greater than one is needed, so that the reliability values are being increased during the iterative process. When there is not such normalisation, LLR values should be

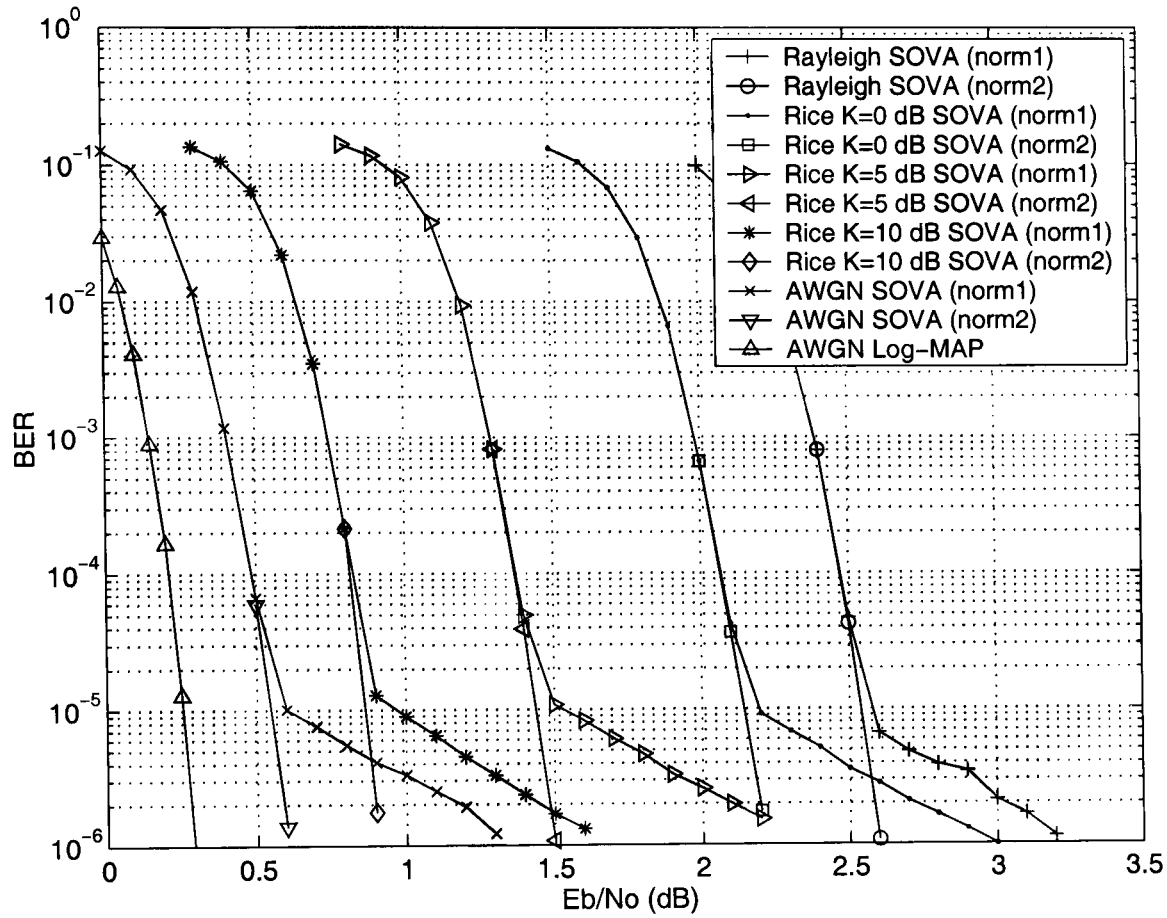


Figure 3.11: BER performance of normalised SOVA and comparison. Coding rate $R=1/3$, 10000 bits frame size and 18 decoding iterations in an uncorrelated Rayleigh/Rician fading channel.

limited to a smaller range, as [5] indicates, by introducing a correcting factor that is less than one.

The concept of increasing the correcting factor was described in different ways in [72], [80]- [82] and [85]. In [80] it was noticed that the correcting factor increases, as the BER decreases. Moreover, the correcting factor was linearly increased against the number of decoding iterations in [81] and at high E_b/N_o values in [82]. A simpler method was proposed in [72, 85], where five steps were applied to quantize the linearly increasing correcting factor. However, the proposed method (i.e. *norm2* SOVA) simplifies the normalisation approach, as two steps were shown to be enough to improve the performance at low BER values.

The approach of the *norm1/norm2* SOVA is based on the least complex HR-SOVA

decoding algorithm and is independent on the noise variance of the channel (i.e. $L_c = 1$). The decoding complexity that is added is relatively small. That is, one more multiplication of the extrinsic information with a fixed value per component decoder and per decoding iteration. One drawback is that the scaling factor should be calculated in advance based on trial and error. As shown from Figs. 3.8-3.11 the *norm1* SOVA improves the performance at medium BER values with respect to the conventional SOVA iterative decoder, while the *norm2* SOVA improves further the performance at low BER values. In particular, assuming large frame lengths and high number of decoding iterations, the *norm2* SOVA removes the error floor at BER below 10^{-6} , independently of the channel type. In addition, the *norm2* SOVA performance is 0.3 dB inferior to the Log-MAP iterative decoder at BER of 10^{-6} in the AWGN channel.

3.6 Summary

In this Section the most important issues on improved SOVA turbo decoder are summarised.

- The iterative SOVA decoder is *sub-optimum* in terms of BER performance compared to Log-MAP iterative decoding, e.g. 0.7 dB performance degradation at BER of 10^{-4} . This is because the soft-output is based on updating the reliability of two only trellis paths.
- The *advantage* of SOVA is that it is three times less complex than the Log-MAP algorithm, when it is applied to turbo decoding. That makes easier a hardware decoder implementation.
- There have been several approaches on *improving* the HR-SOVA turbo decoder, e.g. BR-SOVA, Bi-SOVA and List-SOVA. They add extra computational complexity, as the reliability updating process is based on either more than two trellis paths, extra updating mode backwards or more pairs of path metrics respectively. However, they are still less complex than the Log-MAP iterative decoder.
- Improving the *simplest* approach of SOVA, i.e. HR-SOVA, is more challenging,

as it is the least complex algorithm compared to other SOVA decoder implementations.

- There exist *two fundamental attempts* on improving the HR-SOVA. One is based on the *Gaussian* assumption distribution (scaling of extrinsic information and reduction of correlation between intrinsic and extrinsic information) and the other one is based on limiting the reliability values.
- Several recent methods have been proposed to improve the HR-SOVA iterative decoder. Among them, the best found SOVA turbo decoder can approach the MAP performance to 0.1 dB at BER of 10^{-4} [72].
- Two improved SOVA iterative decoding algorithms were considered in computer simulations. *Norm1* SOVA is based on scaling the extrinsic information with constant value. Moreover, if the value of scaling is increased during the last decoding iteration, a novel method *norm2* SOVA is obtained.
- It was shown that the *norm1* SOVA improves the performance at medium BER values, while the *norm2* SOVA improves further the performance at low BER values. The reason for that is the reduction of the correlation coefficient between intrinsic and extrinsic information.
- Assuming large frame lengths, no error floor was observed at BER of 10^{-6} , using the *norm2* SOVA. This is 0.3 dB inferior to the BER performance of the Log-MAP algorithm in the AWGN channel.

Chapter 4

Improved Max-Log-MAP and Log-MAP Decoding for Binary Turbo Codes

This is the *second* Chapter where original work is proposed based on classical (i.e. binary) turbo codes. Two different approaches to efficient iterative decoding algorithms are shown to be good alternatives to Max-Log-MAP and Log-MAP decoding. In both cases, this is achieved by a trade-off between BER performance and decoding complexity. This trade-off is demonstrated through various computer simulation results run in the AWGN channel as well as in an uncorrelated Rayleigh fading channel. Decoding complexity estimation of the proposed SISO algorithms is also given.

4.1 Introduction

In practical turbo decoder applications the MAP algorithm, as described in [6], is generally too complex to be implemented, due to the large number of multiplications and non-linear functions, e.g. exponentials. A simple solution was shortly proposed in [7] to use the Log-MAP and Max-Log-MAP algorithms. Both the algorithms operate in the logarithmic domain and as a consequence, the multiplications become additions and the exponentials are simplified, e.g. see Section 2.3.3.

As mentioned in Section 2.3.4, BER results in [7] have shown that the Log-MAP turbo decoder performs almost the same as the MAP turbo decoder, while the Max-Log-MAP turbo decoder has a small performance degradation of approximately 0.4 dB at high to medium BER values when assuming a rate 1/2 turbo code for BPSK signals over the AWGN channel. However, the Max-Log-MAP turbo decoder is half as complex as the Log-MAP turbo decoder, providing good compromise between BER performance and decoding complexity. This is mainly because the Max-Log-MAP algorithm makes use of the *max* operation, in contrast to the *max** operation that is used in the Log-MAP algorithm [60].

Efficient iterative decoding algorithms are targeting either BER performance improvement with respect to Max-Log-MAP decoding or decoding complexity reduction with respect to Log-MAP decoding. This is achieved by small decoding complexity increase in the first case and small BER performance degradation in the second case. In some other cases, BER performance improvements to the Max-Log-MAP turbo decoding are possible with *very small* increase to the decoding complexity. In a similar way, decoding complexity reduction of the Log-MAP turbo decoding is possible with *negligible* BER performance degradation. Obviously, these algorithms are more promising.

In this Chapter, novel SISO decoding algorithms suitable for binary turbo codes are presented. Two different approaches are taken into account that compromise in general the BER performance and decoding complexity with respect to Max-Log-MAP and Log-MAP decodings. In the following Section, the most important algorithm improvements with Max-Log-MAP and Log-MAP turbo decoding are summarised.

4.2 Relevant Work on Improved Max-Log-MAP and Log-MAP Turbo Decoder

The *improved* Max-Log-MAP turbo decoder was first introduced in [91] by scaling the extrinsic information of the two component decoders with a constant factor. This concept had already been known for the improved SOVA turbo decoder and it adds negligible extra decoding complexity, e.g. see Section 3.3.2. A typical value of scaling

factor was reported to be $c_1 = c_2 = Z = 0.7$ and was based on trial and error. Furthermore, simulation results in [91] have shown that the improved Max-Log-MAP algorithm can approach the Log-MAP turbo decoder up to 0.1 dB for a wide range of SNR values, assuming the 3GPP turbo code with coding rate $R = 1/3$ for BPSK signals over the AWGN channel. In addition, the BER performance of the same code was only 0.2 dB inferior to the Log-MAP turbo decoder over an uncorrelated Rayleigh fading channel.

The *Constant* Log-MAP turbo decoder was first proposed in [93] and an extensive performance evaluation of it was given in [94]. The goal of this algorithm is to use a LUT of *two* values, instead of the more usually assumed eight values. The *simplified* \max^* operator is computed from

$$\max^*(x, y) = \max(x, y) + c \quad (4.1)$$

where the correcting factor c takes two possible values, as

$$c = \begin{cases} 3/8, & \text{if } |x - y| < 2 \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

That results in simplified logic circuits with area savings of around 40% in 0.5 μm CMOS technology [93]. Moreover, the reduced implementation complexity has negligible impact on the BER performance degradation compared to the Log-MAP turbo decoder. For example, this is 0.03 dB at high to medium BER values, assuming a rate 1/2 turbo code for BPSK signals over the AWGN channel [93].

In [9], apart from the *Constant* Log-MAP, the *Linear* Log-MAP turbo decoder was introduced. The goal of this algorithm is to approximate the correcting factor with a *linear* function. The new simplified \max^* operator is computed from

$$\max^*(x, y) = \max(x, y) + f_c \quad (4.3)$$

Similarly, the correcting factor f_c depends on the absolute difference $|x - y|$ and is computed from the general expression $f_c = -a(|x - y| - b)$, where a, b are positive

constants. Optimised values of f_c were given for the 3GPP turbo code with coding rate $R = 1/3$, assuming BPSK signals over either the AWGN or an uncorrelated Rayleigh fading channel. The BER performance degradation against the Log-MAP turbo decoder was reported to be 0.01 dB at high to medium BER values.

Two recent methods on the *improved* Max-Log-MAP turbo decoder were presented in [95, 96]. They both use scaling of the extrinsic information, as in [91]. In addition, the optimum correcting values (i.e. c_1 and c_2) in [95] are pre-calculated off-line by maximising the *mutual information* exchanged between the component decoders (i.e. *a priori* information and reliability channel values). Simulation results of a rate 1/2 turbo code with the resulting Max-Log-MAP algorithm from [95] indicate similar performance improvements to [91] at high to medium BER values and over the AWGN channel.

On the other hand, the authors in [96] have used, apart from scaling, an approximated correcting factor (denoted by f_c) in the soft-output calculation of the Max-Log-MAP algorithm (denoted by $L(\hat{u}_k)_{MLM}$) based on either a logarithm or a linear function. That means the modified soft-output $L(\hat{u}_k)_{mod}$ should be computed from $L(\hat{u}_k)_{mod} = L(\hat{u}_k)_{MLM} + f_c$ and the correcting factor from $f_c = \log(1 + e^{-x})$ in the first case, or $f_c = -ax + b$ in the second case where a, b are positive constants, similar to [9]. The modified Max-Log-MAP algorithm in [96] was reported to have performance degradation of up to 0.1 dB with respect to the Log-MAP turbo decoder for a wide range of SNR values. This is in agreement with [91]. The simulation parameters were the 3GPP2 8-states turbo code with rate 1/5 for BPSK signals over the AWGN channel [97].

The *enhanced* Max-Log-MAP turbo decoder for both binary and duo-binary turbo codes was reported in [98, 11]. A *stopping criterion* was also used in these references. Although there is a different terminology, it is identical to the improved Max-Log-MAP algorithm from [91]. In both cases, the enhanced Max-Log-MAP algorithm can approach the Log-MAP turbo decoder performance to 0.1 dB at high to medium BER values, assuming an 8-states turbo code for BPSK signals over the AWGN channel.

Combinations between Log-MAP and Max-Log-MAP turbo decoding are also possible,

as [99] indicates. At each decoding iteration, the reliability values are measured on every frame and it is checked if they are greater or smaller than those in the previous iteration. The check is performed by a *stopping criterion*, which then determines what decoding algorithm is to be used according to whether the reliability values are being increased or not. The resulting scheme was found to be 0.1 dB inferior in terms of BER performance compared to Log-MAP turbo decoder with nearly Max-Log-MAP decoding complexity. The simulation parameters were the 3GPP turbo encoder with coding rate $R = 1/3$ for BPSK signals over the AWGN channel.

Finally, new SISO decoding algorithms suitable for turbo codes were reported in [100]. One variation of them is a VA-based algorithm, which performs close to Log-MAP turbo decoding with lower decoding complexity. When this algorithm is further simplified, i.e. by omitting the correcting factor and using the *max* operation only, it has identical BER performance to Max-Log-MAP turbo decoding and even lower decoding complexity. The main idea was to split the joint probability used in the *a posteriori* probability computation of the MAP algorithm into two terms and then define new calculations for the forward/backward recursion. A rate 1/3 turbo code assuming BPSK signals over the AWGN channel was considered in computer simulation results.

4.3 SISO Algorithms Based on Max/Max* Operation Replacement for Turbo Decoding

Here, novel SISO decoding algorithms are presented where both the *max* operation of the Max-Log-MAP algorithm and the *max** operation of the Log-MAP algorithm are combined in an appropriate way [101]. The decoding complexity is estimated and computer simulation results are shown.

4.3.1 Motivation

As discussed in Section 3.3.2, a unified approach to iterative SOVA decoding is feasible by proper “path collection tuning” and *max/max** operation replacement to the reliability (i.e. LLR) estimation [77]. That gives overall performance improvement against

the conventional SOVA, at the expense of decoding complexity increase. In this case, the maximum coding gain improvement when using the \max/\max^* operation replacement was reported to be 0.4 dB at BER of up to 10^{-5} , assuming a rate 1/3 turbo code for BPSK signals over the AWGN channel.

Motivated by this approach on the improved SOVA turbo decoder, the \max/\max^* operation replacement is *extended* to include the case of Max-Log-MAP and Log-MAP algorithms. It is expected that the BER performance of the iterative Max-Log-MAP algorithm is improved, at the expense of decoding complexity increase. Alternatively, the decoding complexity of iterative Log-MAP algorithm is reduced, at the expense of small BER performance degradation. The resulting sub-optimum SISO decoding algorithms provide good trade-off between BER performance and complexity and are described in more detail in the next Section.

4.3.2 Proposed SISO Decoding Algorithms and Complexity Estimation

Novel SISO decoding algorithms are obtained, if the \max/\max^* operation replacement is applied not only to the soft-output (i.e. LLR), as previously done in [77] for the improved SOVA turbo decoder, but is also expanded to include the forward and backward recursion (i.e. $\tilde{\alpha}$ and $\tilde{\beta}$) of the corresponding Max-Log-MAP and Log-MAP algorithms, e.g. see Section 2.3.3.

By taking into account the different combinations of the \max/\max^* operation needed for both $\tilde{\alpha}$, $\tilde{\beta}$ and LLR computation, it is concluded that there exist eight different SISO decoding algorithms, which are summarised in Table 4.1. Apart from the existing Max-Log-MAP and Log-MAP algorithms, the six novel SISO decoding algorithms are denoted as $SISO - A1/2$, $SISO - B$, $SISO - C$ and $SISO - D1/2$ respectively.

It is noted that the already known Max-Log-MAP and Log-MAP algorithms belong to this family of SISO decoding algorithms. For example, in the extreme case when the \max operation is applied to both $\tilde{\alpha}$, $\tilde{\beta}$ and LLR, then the Max-Log-MAP algorithm is obtained. Similarly, in the opposite extreme case when the \max^* operation is applied to all stages, then the Log-MAP algorithm is obtained.

Table 4.1: Proposed SISO decoding algorithms; operation and notation.

Basic operation for computing $\tilde{\alpha}$, $\tilde{\beta}$ and LLR respectively	Type
(max, max, max)	Max-Log-MAP
(max, max^*, max)	SISO-A1 (SISO-A)
(max^*, max, max)	SISO-A2 (SISO-A)
(max, max, max^*)	SISO-B
(max^*, max^*, max)	SISO-C
(max, max^*, max^*)	SISO-D1 (SISO-D)
(max^*, max, max^*)	SISO-D2 (SISO-D)
(max^*, max^*, max^*)	Log-MAP

Table 4.2: Decoding complexity estimation of SISO decoding algorithms based on max/max^* operation replacement. M is the turbo encoder memory order.

	max operations	LUT operations	additions (total)
Max-Log-MAP	$5 \times 2^M - 2$	//	$10 \times 2^M + 11$
SISO-A	$5 \times 2^M - 2$	1.5×2^M	$11.5 \times 2^M + 11$
SISO-B	$5 \times 2^M - 2$	$2 \times 2^M - 2$	$12 \times 2^M + 9$
SISO-C	$5 \times 2^M - 2$	3×2^M	$13 \times 2^M + 11$
SISO-D	$5 \times 2^M - 2$	$3.5 \times 2^M - 2$	$13.5 \times 2^M + 9$
Log-MAP	$5 \times 2^M - 2$	$5 \times 2^M - 2$	$15 \times 2^M + 9$

As indicated by computer simulation experiments, the BER performance between SISO-A1 and SISO-A2 (or SISO-D1 and SISO-D2) is identical. This is explained because there is no difference in the order of the max/max^* operation that may happen in either the forward or the backward recursion. Therefore, both algorithms are denoted as SISO-A (or SISO-D).

In Table 4.2 the decoding complexity of the proposed SISO decoding algorithms is estimated per information bit and decoding iteration. Calculations are shown for turbo encoders with memory order M and are based on [7].

Table 4.3: Relative decoding complexity comparison of SISO decoding algorithms based on \max/\max^* operation replacement with respect to Max-Log-MAP and Log-MAP turbo decoder.

	\max + LUT ops. (increase) w.r.t. Max-Log-MAP	\max + LUT ops. (decrease) w.r.t. Log-MAP	additions (increase) w.r.t. Max-Log-MAP	additions (decrease) w.r.t. Log-MAP
SISO-A	$1.5 \times 2^M / (5 \times 2^M - 2)$	$(3.5 \times 2^M - 2) / (5 \times 2^M - 2)$	$1.5 \times 2^M / (10 \times 2^M + 11)$	$(3.5 \times 2^M - 2) / (15 \times 2^M + 9)$
SISO-B	$(2 \times 2^M - 2) / (5 \times 2^M - 2)$	$3 \times 2^M / (5 \times 2^M - 2)$	$(2 \times 2^M - 2) / (10 \times 2^M + 11)$	$3 \times 2^M / (15 \times 2^M + 9)$
SISO-C	$3 \times 2^M / (5 \times 2^M - 2)$	$(2 \times 2^M - 2) / (5 \times 2^M - 2)$	$3 \times 2^M / (10 \times 2^M + 11)$	$(2 \times 2^M - 2) / (15 \times 2^M + 9)$
SISO-D	$(3.5 \times 2^M - 2) / (5 \times 2^M - 2)$	$1.5 \times 2^M / (5 \times 2^M - 2)$	$(3.5 \times 2^M - 2) / (10 \times 2^M + 11)$	$1.5 \times 2^M / (15 \times 2^M + 9)$

In order to break down the complexity estimation from Table 4.2, two important issues are highlighted. First, it is known that one \max^* operation is composed of one \max operation plus one value of LUT. As a consequence, the number of LUT operations shown in Table 4.2 corresponds to the number of extra additions with respect to Max-Log-MAP turbo decoding. Second, assuming the Max-Log-MAP turbo decoder, the number of \max operations when computing either the forward or backward recursion is equal to 1.5×2^M and is equal to $2 \times 2^M - 2$ when computing the soft-output value. Thus, the complexity estimation of the proposed SISO decoding algorithms is obtained from the corresponding complexity of Max-Log-MAP turbo decoder by taking account the appropriate \max/\max^* operation replacement to the forward/backward recursion or soft-output computation.

The *relative* decoding complexity increase (or decrease) of the proposed SISO decoding algorithms with respect to Max-Log-MAP (or Log-MAP) turbo decoder is reported in Table 4.3. The comparison consists of \max , LUT operations and number of additions and is done in a similar way to [7].

4.3.3 Computer Simulation Results

BER simulation performance results of the proposed sub-optimum SISO decoding algorithms are shown in the case of two channel types, i.e. AWGN (Figs. 4.1 and 4.2) and uncorrelated Rayleigh fading with no CSI available at the receiver (Figs. 4.3 and 4.4). As a comparison, the BER performance of the Max-Log-MAP and Log-MAP algorithms is also shown. A rate 1/2 turbo encoder is assumed with either 4-states or 16-states and generator polynomials either $(1, 5/7)_o$ or $(1, 21/37)_o$ respectively. The frame size is 1000 bits and either 2 or 8 decoding iterations are performed.

Discussion

From Figs. 4.1- 4.4 is noticed similar BER performance behaviour of the proposed SISO decoding algorithms, independently of the memory order, channel type and number of decoding iterations. It is also verified that the more (or less) complex the SISO decoding algorithm with respect to Max-Log-MAP (or Log-MAP) algorithm, the better (or worse) it performs.

All the SISO decoding algorithms perform the same at BER lower than 10^{-5} . This is because the reliability channel value is increased, which results in an increased *a priori* information that is fed to the two component decoders input. Also, the performance gap difference between the Max-Log-MAP and Log-MAP algorithms becomes greater with increasing memory order of the turbo encoder. This was explained in [100] as resulting from the log-sum approximation error in the corresponding soft-output calculation, which increases with the number of possible states.

The relative computational complexity comparison of the proposed SISO algorithms when using the above turbo encoders, i.e. $(1, 5/7)_o$ and $(1, 21/37)_o$, is shown in Table 4.4. This is done by replacing $M = 2$ and $M = 4$ in Table 4.3 respectively.

From Table 4.4 it is observed that SISO-A and SISO-B require approximately half the decoding complexity compared to SISO-C and SISO-D. On the other hand, SISO-A and SISO-B have greater performance degradation compared to SISO-C and SISO-D and with respect to the Log-MAP iterative decoding. In order to find a SISO

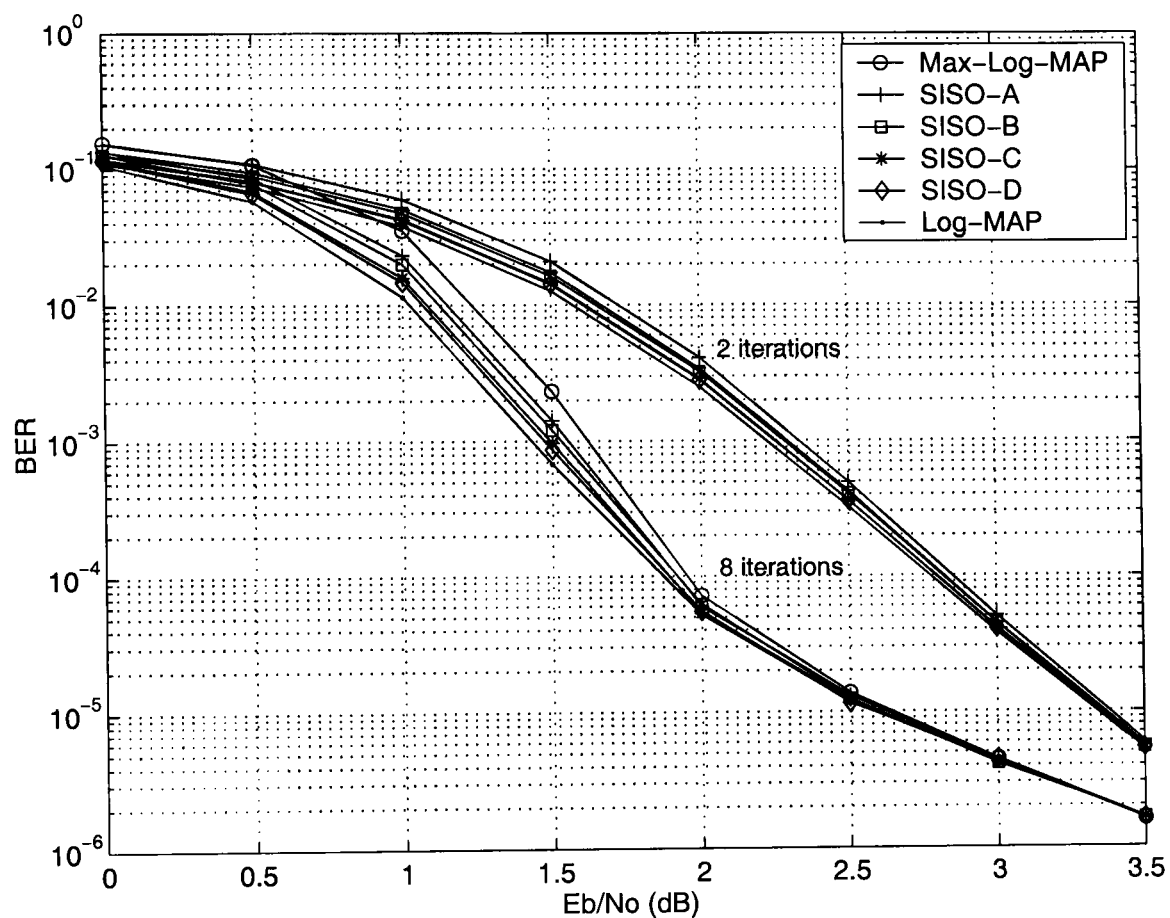


Figure 4.1: BER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.

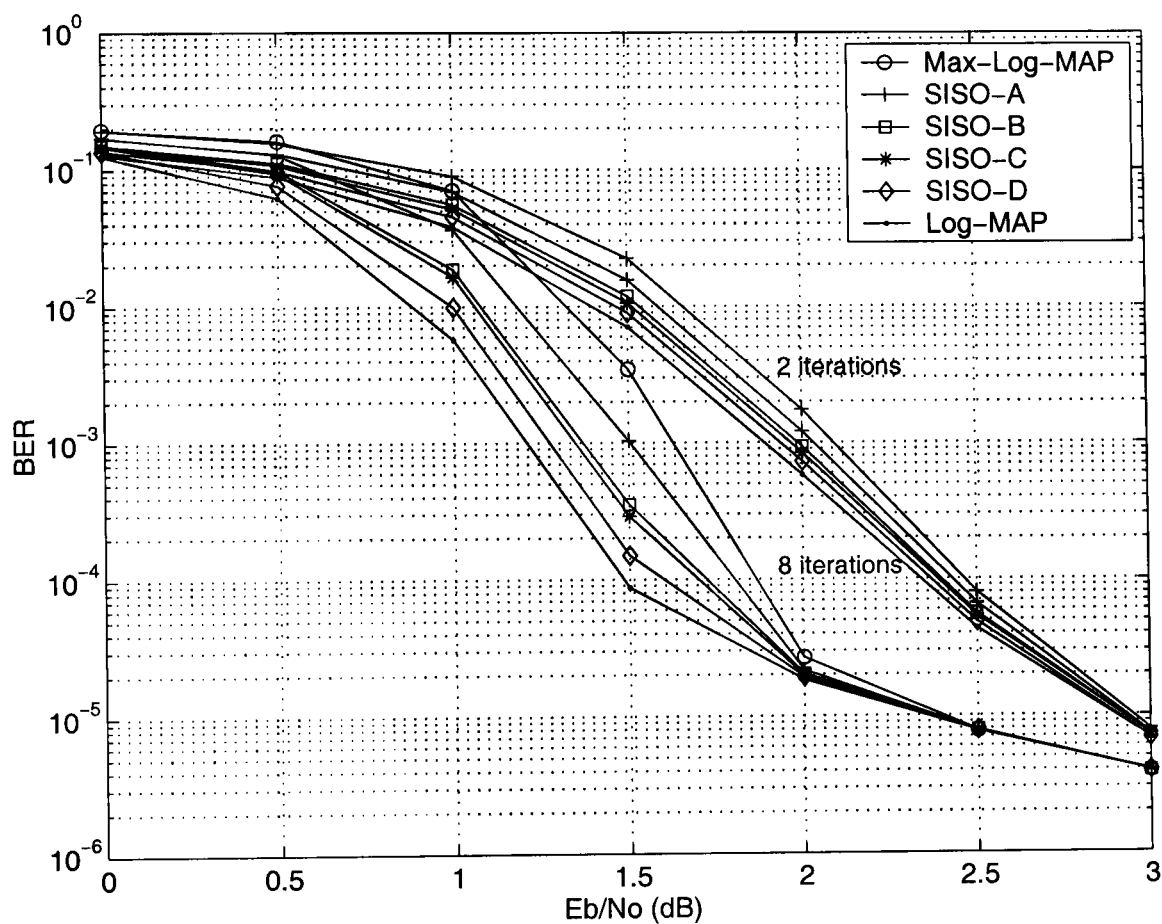


Figure 4.2: BER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.

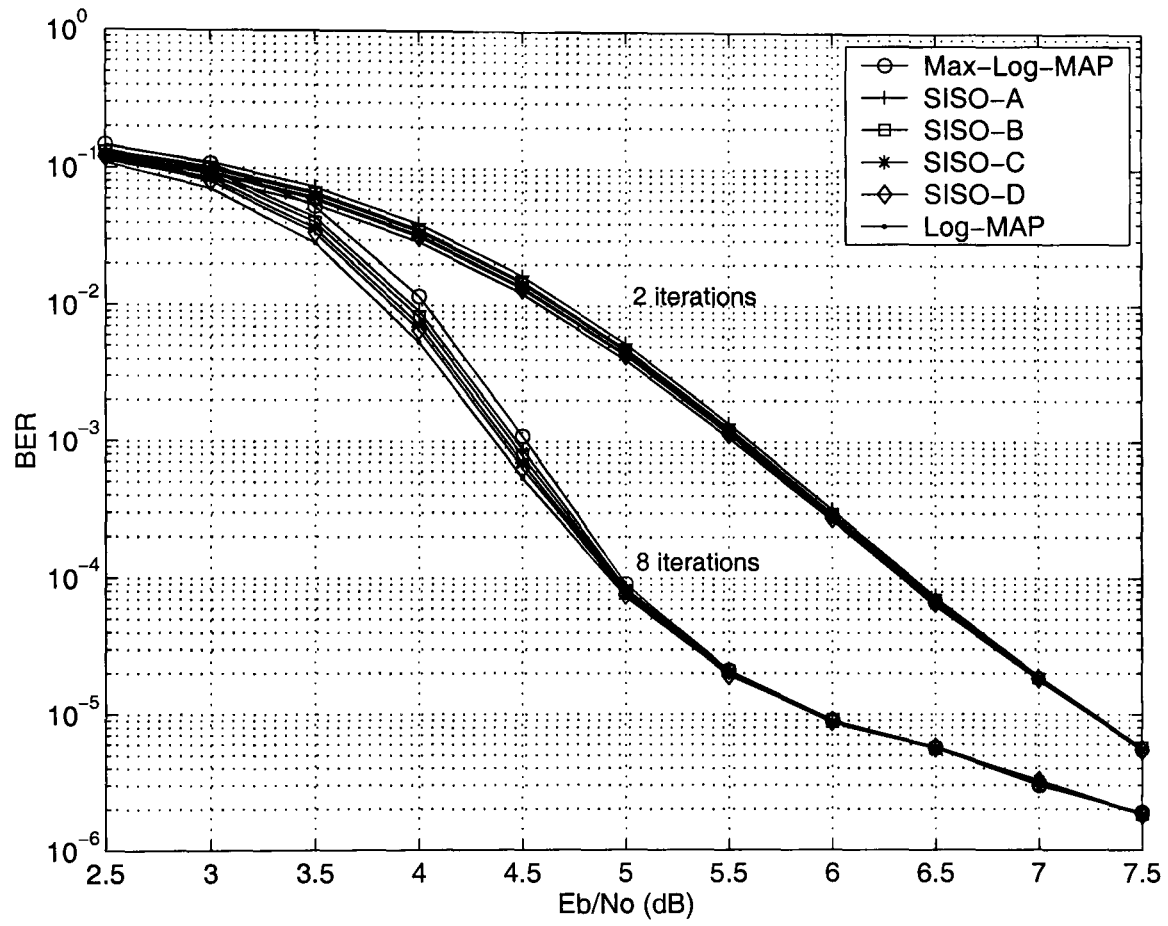


Figure 4.3: BER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. Turbo code generator polynomials $(1,5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.

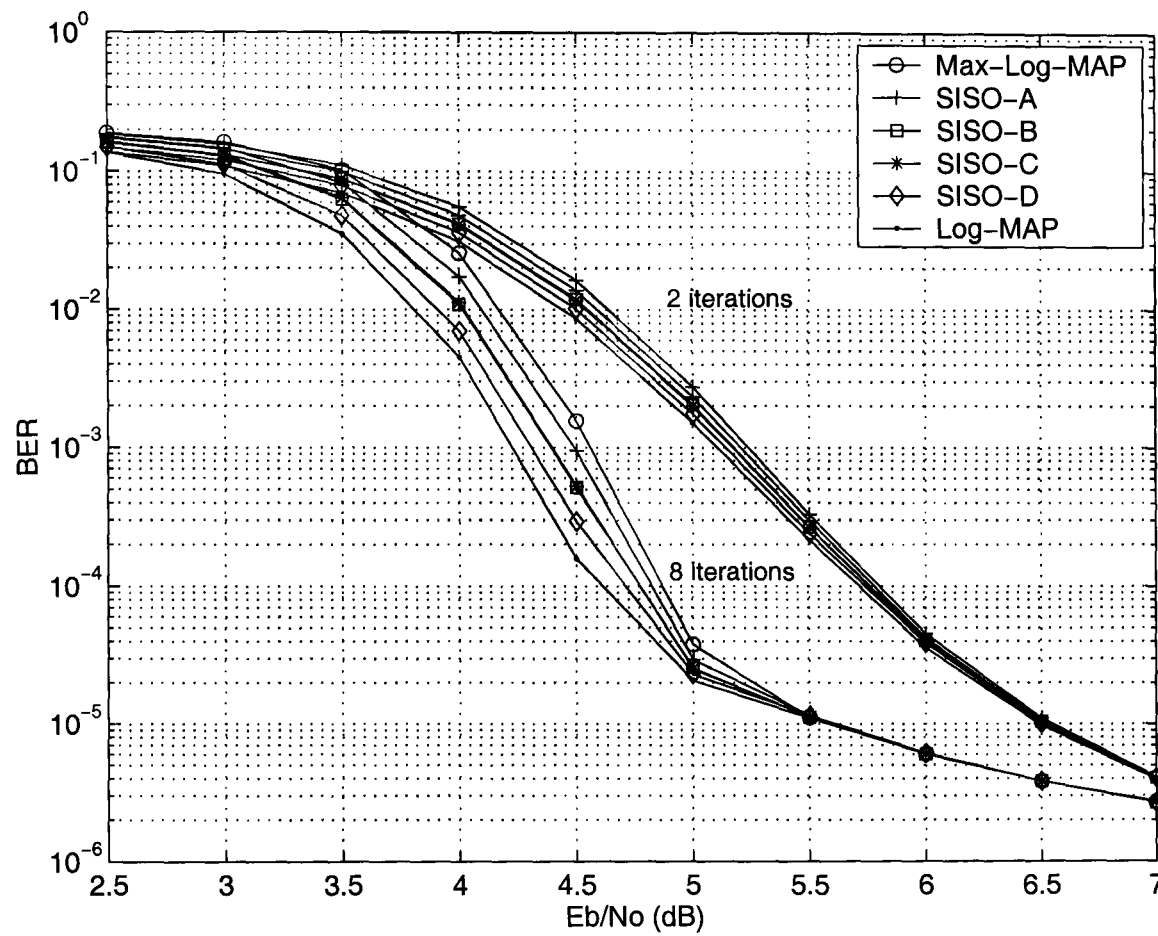


Figure 4.4: BER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.

Table 4.4: Relative decoding complexity comparison example of SISO decoding algorithms based on \max/\max^* operation replacement with respect to Max-Log-MAP and Log-MAP turbo decoder.

	\max + LUT ops. (increase) w.r.t. Max-Log-MAP	\max + LUT ops. (decrease) w.r.t. Log-MAP	additions (increase) w.r.t. Max-Log-MAP	additions (decrease) w.r.t. Log-MAP
M=2				
SISO-A	33.33%	66.66%	11.76%	17.39%
SISO-B	33.33%	66.66%	11.76%	17.39%
SISO-C	66.66%	33.33%	23.53%	8.70%
SISO-D	66.66%	33.33%	23.53%	8.70%
M=4				
SISO-A	30.77%	69.23%	14.04%	21.69%
SISO-B	38.46%	61.54%	17.54%	19.28%
SISO-C	61.54%	38.46%	28.07%	12.05%
SISO-D	69.23%	30.77%	31.58%	9.64%

decoding algorithm with good trade-off, we restrict the search among them so that the relative complexity increase (or decrease) of the \max plus LUT operations does not exceed the 50% with respect to the decoding complexity of the original algorithm, i.e. Max-Log-MAP (or Log-MAP). It is thus concluded that SISO-B provides a *good* trade-off between BER performance and complexity with respect to the Max-Log-MAP iterative decoding. On the other hand, SISO-C provides a *good* trade-off between BER performance and complexity with respect to the Log-MAP iterative decoding.

As an example, let us assume the 16-state turbo encoder over the AWGN channel, as Fig. 4.2 indicates. At medium BER values, SISO-B improves the iterative Max-Log-MAP performance up to 0.28 dB, while SISO-C degrades the iterative Log-MAP performance up to 0.13 dB. In the first case, the relative complexity increase is 38.46% more LUT operations and 17.54% extra additions. In the second case, the relative complexity decrease is 38.46% fewer LUT operations and 12.05% fewer additions.

The presented computer simulation results indicate that the BER performance of the proposed SISO decoding algorithms is a trade-off against the decoding complexity, thus creating a range of performance/complexity options.

4.4 SISO Algorithms Based on the Application of Max/ Max* Operation on Levels for Turbo Decoding

Another novel approach to SISO decoding algorithms is presented either to reduce the computational complexity of the Log-MAP algorithm or to improve the BER performance of the Max-Log-MAP algorithm when they are both applied to turbo decoding [102]. An appropriate *design rule* is given, followed by decoding complexity estimation and computer simulation results.

4.4.1 Motivation

Inspired by the SISO decoding algorithms presented in Section 4.3.2, a search for further trade-off between the turbo code BER performance and complexity was considered. As a reference, the algorithms SISO-B and SISO-C were taken into account. This is because they were shown to provide a good trade-off between BER performance and complexity. In more detail, it was shown that the BER performance of the Max-Log-MAP algorithm can be improved, if the \max^* operator is applied to the soft-output computation (i.e. SISO-B algorithm). Similarly, the decoding complexity of the Log-MAP algorithm can be reduced, if the \max operator is applied to the soft-output computation (i.e. SISO-C algorithm).

Motivated by this approach, a *certain number* of \max^* (or \max) operations is introduced in the soft-output computation (i.e. LLR) of the Max-Log-MAP (or Log-MAP) algorithm, e.g. see Section 2.3.3. This is done by applying the \max^* (or \max) operation into different *levels*, producing a complexity/BER performance trade-off. The resulting sub-optimum SISO decoding algorithms are described in more detail in the next Section.

4.4.2 Proposed SISO Decoding Algorithms and Complexity Estimation

Assume a binary turbo code with memory order M that has 2^M possible states and 2×2^M branch metrics. Assume also that it is iteratively decoded by the Max-Log-MAP (or Log-MAP) algorithm, as described in Section 2.3.3. For the computation of the forward/backward recursion there exist always two arguments of the \max (or \max^*) operator. However, for the soft-output computation, each of the two \max (or \max^*) operators, i.e. one for all branch metrics with transmitted value $x_k = +1$ and the other with $x_k = -1$, has 2^M arguments. That makes the total number of the \max (or \max^*) operations in case of the soft-output computation to be equal to $2 \times (2^M - 1)$. When searching for novel SISO decoding algorithms, we have to take into account all the possible combinations between the \max and \max^* operation in the soft-output computation, which are equal to $2^{2 \times (2^M - 1)}$. Typical values of memory order are $M = 2, 3$ and 4 . Thus, this way of mixing the \max/\max^* operations is *prohibited*.

It can be proved that the \max^* operator is *linear* [103]. In the case of the \max operator, this is straightforward. The \max^* operator can thus be applied in a *tree approach*, i.e. in *pairs* of arguments, as [7]

$$\begin{aligned} \max^*(x_1, x_2, x_3, x_4) &= \ln(e^{x_1} + e^{x_2} + e^{x_3} + e^{x_4}) \\ &= \max^*\{\max^*(x_1, x_2), \max^*(x_3, x_4)\} \end{aligned} \quad (4.4)$$

Note that from Eq. (4.4) there exist more than one *level* (denoted by L) that the \max^* operator can be applied. Let us define as *level-1* (denoted by $L1$) the first \max^* operation, as seen from the left to the right direction of Eq. (4.4), and as *level-2* (denoted by $L2$) the next two \max^* operations, after the $L1$ \max^* operation. It is obvious that the levels L increase *logarithmically* with the total arguments of the \max^* operator. For example, let us consider eight values x_1, x_2, \dots, x_8 . Then, there exist $\log_2 8 = 3$ levels of the \max^* operator, i.e. $L1, L2$ and $L3$, which are shown in Table 4.5.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
level-3								
(L3)	$a_1 = max^*(x_1, x_2)$		$a_2 = max^*(x_3, x_4)$		$a_3 = max^*(x_5, x_6)$		$a_4 = max^*(x_7, x_8)$	
level-2								
(L2)	$b_1 = max^*(a_1, a_2)$				$b_2 = max^*(a_3, a_4)$			
level-1								
(L1)	$c_1 = max^*(b_1, b_2)$							

It is now proposed to apply a *certain number* of \max^* (or \max) operations into *levels* when computing the LLR (i.e. soft-output) value of the Max-Log-MAP (or Log-MAP) turbo decoder, which increases (or reduces) the decoding complexity at reasonable BER performance. Recall that when searching for novel SISO decoding algorithms, we have to take into account all the possible combinations between the \max and \max^* operations in the soft-output computation. This search is now based on *levels* rather than on all possible values. Each of the two \max^* (or \max) operators in the soft-output computation of the Log-MAP (or Max-Log-MAP) turbo decoder has $\log_2 2^M = M$ levels. Therefore, all the possible combinations between the \max and \max^* operations in the soft-output computation are now reduced to 2^M .

After complexity calculations and preliminary BER experimental results, it was found that it is better to apply the \max operation at an early stage (i.e. high level) and then continue with the \max^* operation at the latter stages (i.e. lower levels). For example, consider the Log-MAP algorithm and a 16-states turbo encoder. At low to medium BER values, it was found that applying the \max^* operation to $L4$ and then applying the \max operation to $L3, L2$ and $L1$, the resulting SISO decoding algorithm was performing 0.2 dB worse than applying the \max operation to $L4$ and then applying the \max^* operation to $L3, L2$ and $L1$. This case had to be rejected because the former algorithm was relatively more complex than the latter one. The same rejection, due to BER performance/complexity mismatch, had also to occur in the case where the \max^* operation was applied to $L1$ and $L3$ and the \max operation to $L2$ and $L4$, in contrast to the case where the \max^* operation was applied to $L1$ and $L2$ and the \max operation to $L3$ and $L4$.

Table 4.6: Numerical example of all the possible combinations of the \max/\max^* operation applied to four arguments.

Levels	$(x_1, x_2, x_3, x_4) = (1, 3.5, 4.2, 2.8)$	\max/\max^* oper. output
\max -L1, \max/\max -L2	$\max\{\max(x_1, x_2), \max(x_3, x_4)\}$	$\max(3.5, 4.2) = 4.20$
\max -L1, \max^*/\max -L2	$\max\{\max^*(x_1, x_2), \max(x_3, x_4)\}$	$\max(3.579, 4.2) = 4.20$
\max -L1, \max/\max^* -L2	$\max\{\max(x_1, x_2), \max^*(x_3, x_4)\}$	$\max(3.5, 4.42) = 4.42$
\max -L1, \max^*/\max^* -L2	$\max\{\max^*(x_1, x_2), \max^*(x_3, x_4)\}$	$\max(3.579, 4.42) = 4.42$
\max^* -L1, \max/\max -L2	$\max^*\{\max(x_1, x_2), \max(x_3, x_4)\}$	$\max^*(3.5, 4.2) = 4.60$
\max^* -L1, \max^*/\max -L2	$\max^*\{\max^*(x_1, x_2), \max(x_3, x_4)\}$	$\max^*(3.579, 4.2) = 4.63$
\max^* -L1, \max/\max^* -L2	$\max^*\{\max(x_1, x_2), \max^*(x_3, x_4)\}$	$\max^*(3.5, 4.42) = 4.76$
\max^* -L1, \max^*/\max^* -L2	$\max^*\{\max^*(x_1, x_2), \max^*(x_3, x_4)\}$	$\max^*(3.579, 4.42) = 4.78$

This idea is also demonstrated through a numerical example. Assume four random values (x_1, x_2, x_3, x_4) being equal to $(1, 3.5, 4.2, 2.8)$ respectively. All the possible combinations of the \max/\max^* operation are shown in Table 4.6. From this Table it is observed that the application of the \max^* operator is beneficial to the resulting output only when it is applied to the lower levels. The maximum value that can be obtained is when the \max^* operator is applied both to $L1$ and $L2$, i.e. 4.78.

Hence, due to the BER performance/complexity mismatch described above, all the possible combinations between the \max and \max^* operations in the soft-output computation had to be reduced from the initial value of 2^M . In more detail, assuming a turbo encoder with memory order M , we propose exactly M novel sub-optimum SISO decoding algorithms, which the number of them varies accordingly. It is noticed that the \max/\max^* operations are not mixed inside *levels*. The *design rule* is described as following.

- **Reduced complexity Log-MAP algorithms.** For the forward/backward recursion apply the \max^* operator in the conventional way. For the soft-output, apply the \max operation at an early stage and then continue with the \max^* operation at the latter stages, according to the desired decoding complexity/BER performance trade-off.
- **Improved Max-Log-MAP algorithms.** For the forward/backward recursion ap-

ply the *max* operator in the conventional way. For the soft-output, follow the above rule.

Decoding complexity estimation of the proposed SISO decoding algorithms and appropriate notation is shown in Table 4.7. The complexity calculations are made per information bit and per decoding iteration, assuming a turbo encoder with memory order $M \in \{2, 3 \text{ and } 4\}$ [7].

In order to break down the complexity estimation from Table 4.7, the two most important issues from Section 4.3.2 are highlighted. First, the number of LUT operations corresponds to the number of extra additions with respect to Max-Log-MAP turbo decoding. Second, the complexity estimation of the proposed SISO decoding algorithms is obtained from the corresponding complexity of Max-Log-MAP turbo decoder by taking account the appropriate *max*/*max** operations in the forward/backward recursion as well as in the soft-output computation. The required number of *max*/*max** operations in the forward/backward recursion is always the same and equal to 1.5×2^M . Only the LLR calculation is affected, which makes the total number of *max*/*max** operations vary from zero to the maximum value, i.e. $2 \times 2^M - 2$.

The *relative* decoding complexity increase (or decrease) of the proposed SISO decoding algorithms with respect to Max-Log-MAP (or Log-MAP) turbo decoder is reported in Table 4.8. The comparison consists of *max*, LUT operations and number of additions and is done in a similar way to [7].

In the two Tables, both reduced complexity Log-MAP algorithms, denoted by *LM* – *max** – *L0, 1, ..., 1234*, and improved Max-Log-MAP algorithms, denoted by *MLM* – *max** – *L0, 1, ..., 1234*, are shown. This is the case of a memory $M = 4$ turbo encoder. For a memory $M = 3$ turbo encoder, there exist all the algorithms up to *LM/MLM* – *max** – *L0, 1, ..., 123* and for a memory $M = 2$ turbo encoder, there exist all the algorithms up to *LM/MLM* – *max** – *L0, 1, 12* respectively.

As an example, the notation *LM/MLM* – *max** – *L123* depicts that the *max** operator is applied to *L1, L2* and *L3*, assuming either the Max-Log-MAP or the Log-MAP algorithm. If the turbo encoder has memory four, this implies that the *max* operator

Table 4.7: Decoding complexity estimation of SISO decoding algorithms based on different levels of \max/\max^* operation. M is the turbo encoder memory order.

Reduced complexity Log-MAP algorithms	\max operations	LUT operations	additions (total)
LM- \max^* -L1234 (valid for $M = 2, 3, 4$ i.e. Log-MAP)	$5 \times 2^M - 2$	$5 \times 2^M - 2$	$15 \times 2^M + 9$
LM- \max^* -L123 (valid for $M = 3, 4$ if $M = 3$, Log-MAP)	$5 \times 2^M - 2$	$3 \times 2^M + 14$	$13 \times 2^M + 25$
LM- \max^* -L12 (valid for $M = 2, 3, 4$ if $M = 2$, Log-MAP)	$5 \times 2^M - 2$	$3 \times 2^M + 6$	$13 \times 2^M + 17$
LM- \max^* -L1 (valid for $M = 2, 3, 4$)	$5 \times 2^M - 2$	$3 \times 2^M + 2$	$13 \times 2^M + 13$
LM- \max^* -L0 (valid for $M = 2, 3, 4$)	$5 \times 2^M - 2$	3×2^M	$13 \times 2^M + 11$
Improved Max-Log-MAP algorithms	\max operations	LUT operations	additions (total)
MLM- \max^* -L0 (valid for $M = 2, 3, 4$ i.e. Max-Log-MAP)	$5 \times 2^M - 2$	//	$10 \times 2^M + 11$
MLM- \max^* -L1 (valid for $M = 2, 3, 4$)	$5 \times 2^M - 2$	2	$10 \times 2^M + 13$
MLM- \max^* -L12 (valid for $M = 2, 3, 4$)	$5 \times 2^M - 2$	6	$10 \times 2^M + 17$
MLM- \max^* -L123 (valid for $M = 3, 4$)	$5 \times 2^M - 2$	14	$10 \times 2^M + 25$
MLM- \max^* -L1234 (valid for $M = 4$)	$5 \times 2^M - 2$	$2 \times 2^M - 2$	$12 \times 2^M + 9$

Table 4.8: Relative decoding complexity comparison of SISO decoding algorithms based on different levels of max/max^* operation with respect to Max-Log-MAP and Log-MAP turbo decoder.

Reduced complexity Log-MAP algorithms	max + LUT ops (decrease) w.r.t. Log-MAP	additions (decrease) w.r.t. Log-MAP
LM- max^* -L123 (valid for $M = 3, 4$ if $M = 3$, Log-MAP)	$(2 \times 2^M - 16)/$ $(5 \times 2^M - 2)$	$(2 \times 2^M - 16)/$ $(15 \times 2^M + 9)$
LM- max^* -L12 (valid for $M = 2, 3, 4$ if $M = 2$, Log-MAP)	$(2 \times 2^M - 8)/$ $(5 \times 2^M - 2)$	$(2 \times 2^M - 8)/$ $(15 \times 2^M + 9)$
LM- max^* -L1 (valid for $M = 2, 3, 4$)	$(2 \times 2^M - 4)/$ $(5 \times 2^M - 2)$	$(2 \times 2^M - 4)/$ $(15 \times 2^M + 9)$
LM- max^* -L0 (valid for $M = 2, 3, 4$)	$(2 \times 2^M - 2)/$ $(5 \times 2^M - 2)$	$(2 \times 2^M - 2)/$ $(15 \times 2^M + 9)$
Improved Max-Log-MAP algorithms	max + LUT ops (increase) w.r.t. Max-Log-MAP	additions (increase) w.r.t. Max-Log-MAP
MLM- max^* -L1 (valid for $M = 2, 3, 4$)	$2/(5 \times 2^M - 2)$	$2/(10 \times 2^M + 11)$
MLM- max^* -L12 (valid for $M = 2, 3, 4$)	$6/(5 \times 2^M - 2)$	$6/(10 \times 2^M + 11)$
MLM- max^* -L123 (valid for $M = 3, 4$)	$14/(5 \times 2^M - 2)$	$14/(10 \times 2^M + 11)$
MLM- max^* -L1234 (valid for $M = 4$)	$(2 \times 2^M - 2)/$ $(5 \times 2^M - 2)$	$(2 \times 2^M - 2)/$ $(10 \times 2^M + 11)$

is applied to $L4$. For a memory three turbo encoder, $LM - max^* - L123$ would imply exactly the same as the Log-MAP algorithm, whereas $MLM - max^* - L123$ would be the most complex and best performing of the improved Max-Log-MAP algorithms. For a memory two turbo encoder, this notation is not valid, as there exist two levels only. The notation $LM/MLM - max^* - L0$, implies that the max operator instead is applied to all levels of the LLR value.

We also note that the SISO-B and SISO-C decoding algorithms described in Section 4.3.2 are particular cases of these decoding algorithms. For example, assuming a memory two turbo encoder, $MLM - max^* - L12$ is identical to SISO-B and $LM - max^* - L0$ is identical to SISO-C respectively.

4.4.3 Computer Simulation Results

In a similar way to Section 4.3.3, BER simulation performance results of the proposed sub-optimum SISO decoding algorithms are shown in the case of two channel types, i.e. AWGN (Figs. 4.5 and 4.6) and uncorrelated Rayleigh fading with no CSI available at the receiver (Figs. 4.7 and 4.8). As a comparison, the BER performance of the Max-Log-MAP and Log-MAP algorithms is also shown. A rate 1/2 turbo encoder is assumed with either 4-states or 16-states and generator polynomials either $(1, 5/7)_o$ or $(1, 21/37)_o$ respectively. The frame size is 1000 bits and either 2 or 8 decoding iterations are performed.

Discussion

As in Section 4.3.3, from Figs. 4.5-4.8 similar BER performance behaviour of the proposed SISO decoding algorithms is noticed that is independent of the memory order, channel type and number of decoding iterations. It is also verified that the more (or less) complex the SISO decoding algorithm with respect to Max-Log-MAP (or Log-MAP) algorithm, the better (or worse) it performs. This depends on the number of levels that the max/max^* operator is applied when computing the LLR value.

The same observation as in Section 4.3.3 is also valid in the presented computer simulation results. That is, all the SISO decoding algorithms perform the same at BER

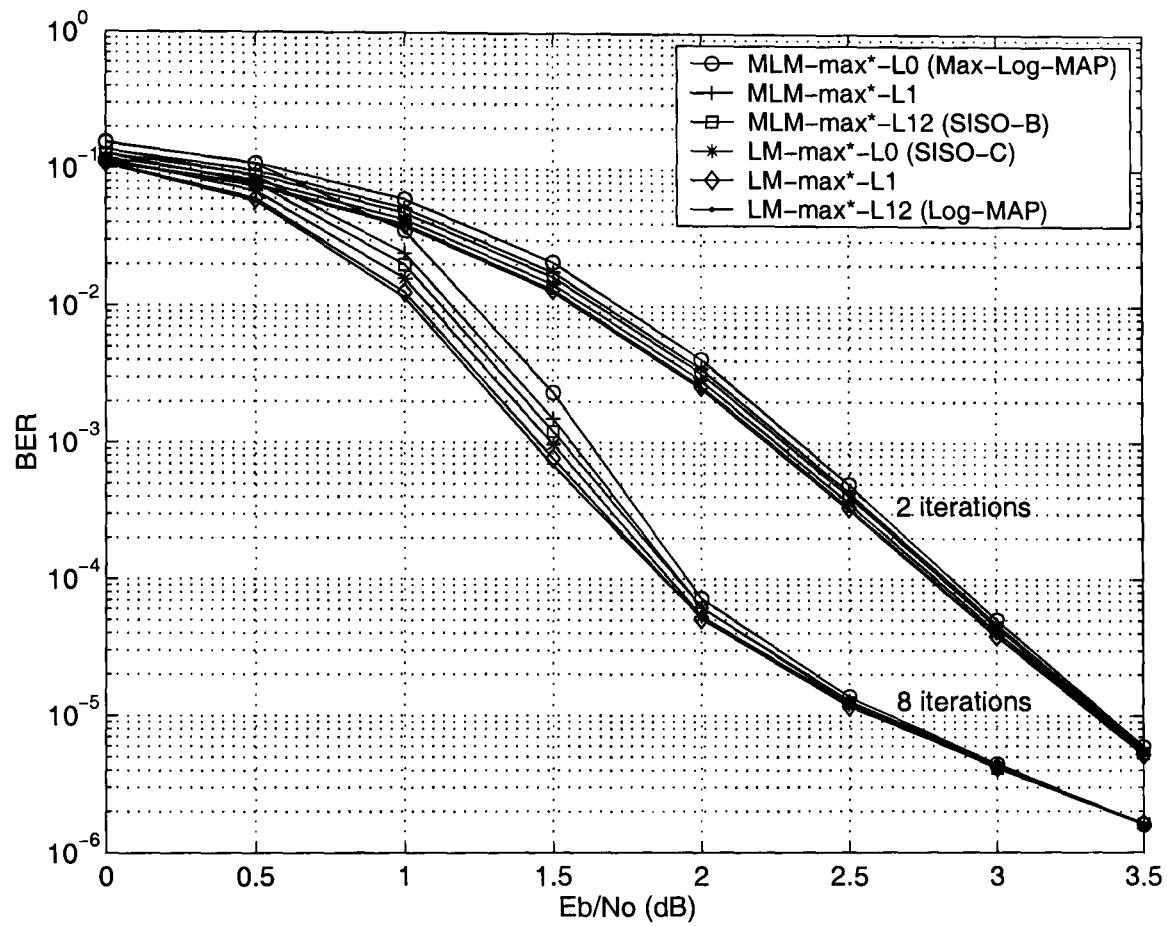


Figure 4.5: BER performance comparison of SISO decoding algorithms based on different levels of \max/\max^* operation. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.

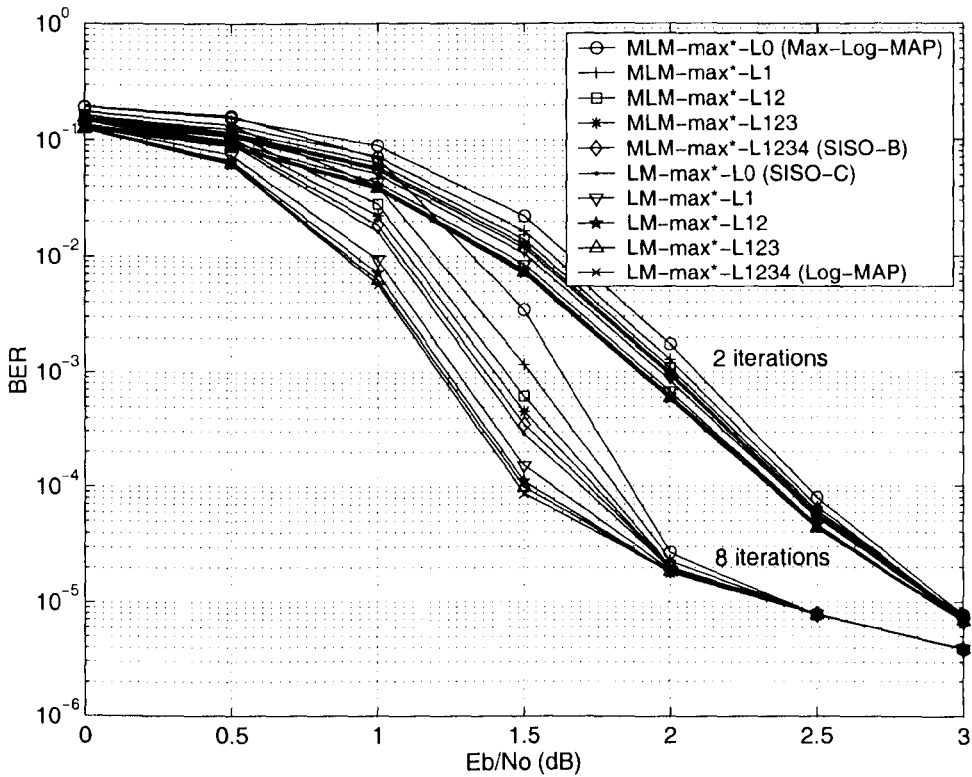


Figure 4.6: BER performance comparison of SISO decoding algorithms based on different levels of \max/\max^* operation. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in the AWGN channel.

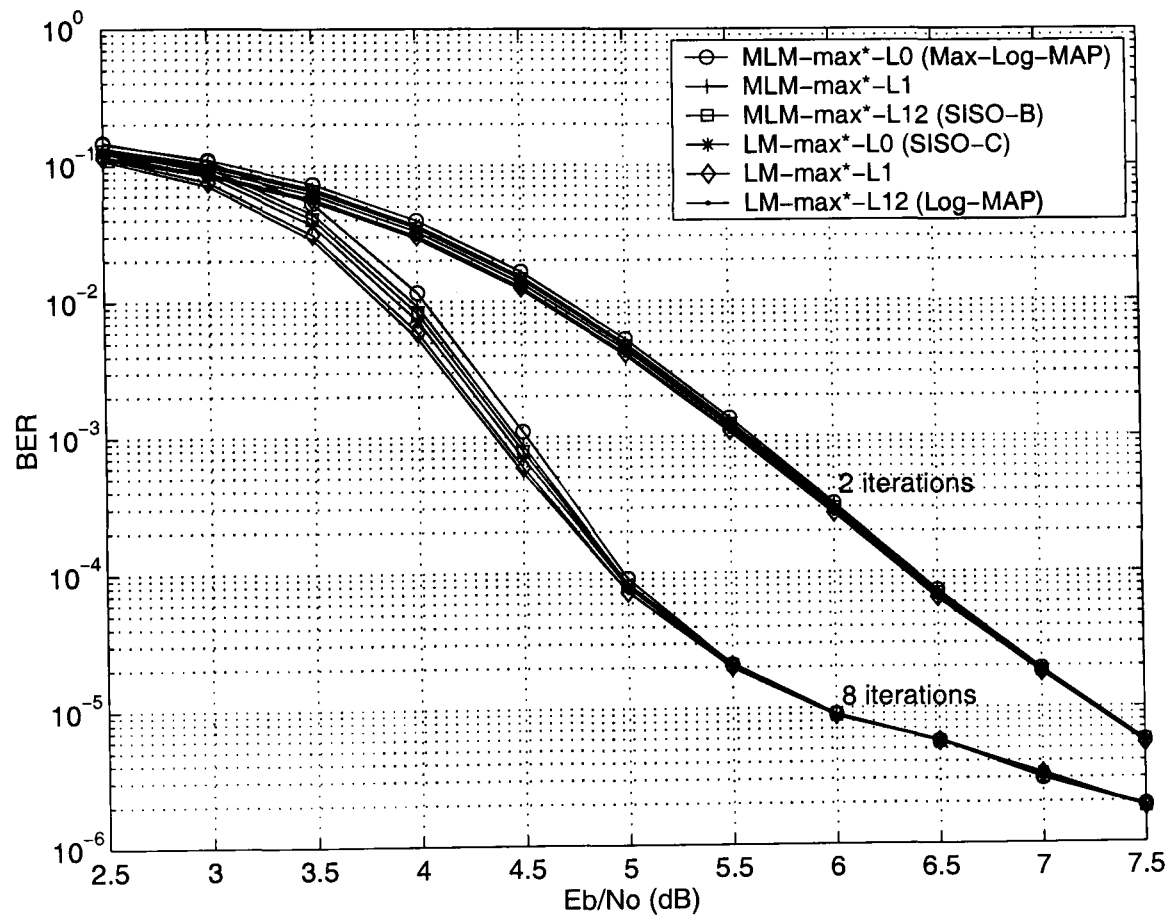


Figure 4.7: BER performance comparison of SISO decoding algorithms based on different levels of \max/\max^* operation. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.

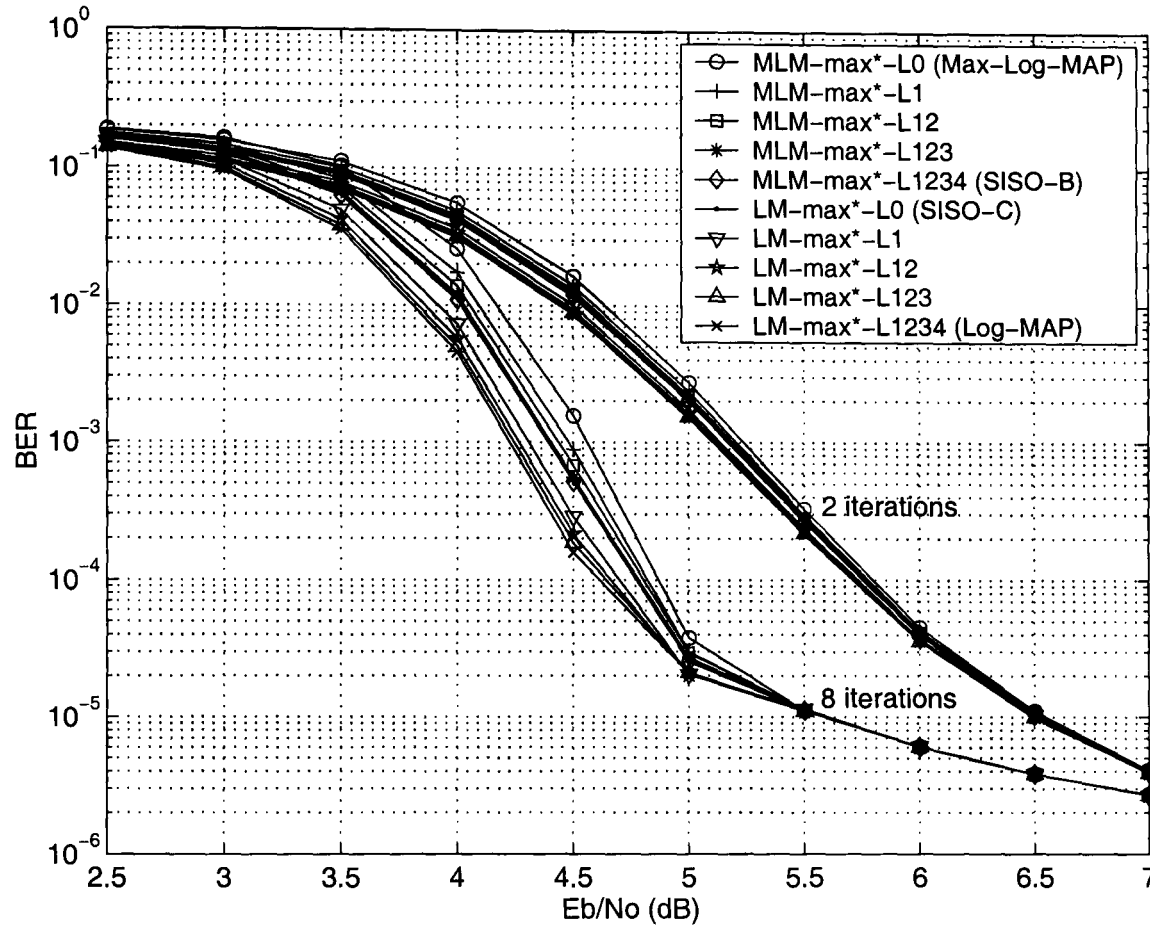


Figure 4.8: BER performance comparison of SISO decoding algorithms based on different levels of max/max^* operation. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R=1/2$, 1000 bits frame size and either 2 or 8 decoding iterations in an uncorrelated Rayleigh fading channel.

Table 4.9: Relative decoding complexity comparison example of SISO decoding algorithms based on different levels of \max/\max^* operation with respect to Max-Log-MAP and Log-MAP turbo decoder.

Reduced complexity Log-MAP algorithms	\max + LUT ops (decrease) w.r.t. Log-MAP	additions (decrease) w.r.t. Log-MAP
M=2		
LM- \max^* -L1	22.22%	5.80%
LM- \max^* -L0	33.33%	8.70%
M=4		
LM- \max^* -L123	20.51%	6.43%
LM- \max^* -L12	30.77%	9.64%
LM- \max^* -L1	35.90%	11.25%
LM- \max^* -L0	38.46%	12.05%
Improved Max-Log-MAP algorithms	\max + LUT ops (increase) w.r.t. Max-Log-MAP	additions (increase) w.r.t. Max-Log-MAP
M=2		
MLM- \max^* -L1	11.11%	3.92%
MLM- \max^* -L12	33.33%	11.76%
M=4		
MLM- \max^* -L1	2.56%	1.17%
MLM- \max^* -L12	7.70%	3.51%
MLM- \max^* -L123	17.95%	8.19%
MLM- \max^* -L1234	38.46%	17.54%

lower than 10^{-5} . A physical explanation was also given in the same Section.

The relative computational complexity comparison of the proposed SISO algorithms when using the above turbo encoders, i.e. $(1, 5/7)_o$ and $(1, 21/37)_o$, is shown in Table 4.9. This is done by replacing $M = 2$ and $M = 4$ in Table 4.8 respectively.

From Table 4.9 it is observed that the application of the \max/\max^* operation in *levels* when computing the LLR value creates a range of SISO decoding algorithms. A

good trade-off between BER performance and complexity with respect to the Max-Log-MAP iterative decoding is achieved when the \max^* operator is applied in all *levels* (i.e. identical to SISO-B). On the other hand, a good trade-off between BER performance and complexity with respect to the Log-MAP iterative decoding is achieved when the \max operator is applied in all *levels* (i.e. identical to SISO-C).

As an example, let us assume the 16-state turbo encoder over the AWGN channel, as Fig. 4.6 indicates. At medium BER values, $MLM - \max^* - L1234$ improves the iterative Max-Log-MAP performance up to 0.28 dB, while $LM - \max^* - L0$ degrades the iterative Log-MAP performance up to 0.13 dB. In the first case, the relative complexity increase is 38.46% more LUT operations and 17.54% extra additions. In the second case, the relative complexity decrease is 38.46% fewer LUT operations and 12.05% fewer additions.

From Fig. 4.6 we also note that $LM - \max^* - L123$, which introduces the \max operator in $L4$ only, has 20.51% fewer LUT operations and 6.43% fewer additions than the Log-MAP turbo decoder. The performance degradation is 0.01 dB at medium BER values. It is thus considered to be a *good option* for reduced complexity algorithm compared to Log-MAP iterative decoding with negligible BER performance loss. Furthermore, $MLM - \max^* - L1$, which introduces the \max^* operator in $L1$ only, has 2.56% more LUT operations and 1.17% more additions than the Max-Log-MAP turbo decoder. The performance improvement is 0.1 dB at medium BER values. It is thus considered to be a *good option* for improved performance algorithm compared to Max-Log-MAP iterative decoding with negligible decoding complexity increase.

The presented computer simulation results indicate that the BER performance of the proposed SISO decoding algorithms is a trade-off against the decoding complexity. A range of performance/complexity options is thus feasible. As a consequence, it is believed that the gap between Max-Log-MAP and Log-MAP iterative decoding has now been closed.

4.5 Summary

In this Section the most important issues on improved Max-Log-MAP and Log-MAP turbo decoder are summarised.

- The iterative Max-Log-MAP decoder is *sub-optimum* in terms of BER performance compared to Log-MAP iterative decoding, e.g. 0.4 dB performance degradation at BER of 10^{-4} . This is because it makes use of the *max* operator in contrast to the more complex *max** operator.
- The *advantage* of Max-Log-MAP algorithm is that it is half as complex as the Log-MAP algorithm when it is applied to turbo decoding. That makes easier a hardware decoder implementation.
- There have been several attempts in the past on either improving the BER performance of the Max-Log-MAP turbo decoder or reducing the decoding complexity of the Log-MAP turbo decoder.
- Among them, the best found algorithm, which is using scaling of the extrinsic information of the Max-Log-MAP turbo decoder with a constant factor, can approach the Log-MAP turbo decoder up to 0.1 dB for a wide range of SNR values.
- On the other hand, an efficient algorithm for reducing the complexity of the Log-MAP turbo decoder is the Constant Log-MAP. The performance degradation is 0.03 dB at high to medium BER values but with decoding complexity savings.
- Two novel approaches to SISO decoding algorithms were presented that demonstrate the trade-off between BER performance and complexity. That is, either improvement to the BER performance with respect to the Max-Log-MAP iterative decoding at the expense of complexity increase or decoding complexity reduction of the Log-MAP iterative decoding at the expense of BER performance degradation. Analytical complexity estimation and relative complexity comparison was given in both cases.
- The first approach was based on *max*/*max** operation replacement to either the forward/backward recursion or soft-output computation of the Max-Log-MAP or

Log-MAP algorithm. Four novel SISO decoding algorithms were proposed with trade-off between performance/complexity.

- For example, assuming a 16-states turbo code and medium BER values, SISO-B improves the iterative Max-Log-MAP performance up to 0.28 dB, while SISO-C degrades the iterative Log-MAP performance up to 0.13 dB. In the first case, the relative complexity increase is 38.46% more LUT operations and 17.54% extra additions. In the second case, the relative complexity decrease is 38.46% fewer LUT operations and 12.05% fewer additions.
- The second approach was based on the application of the \max/\max^* operation in *levels* when computing the soft-output of the Max-Log-MAP or Log-MAP algorithm. For a turbo encoder with memory order M , there exist M novel SISO decoding algorithms with trade-off between performance/complexity.
- For example, assuming a 16-states turbo code and medium BER values, $LM - \max^* - L123$ has 20.51% fewer LUT operations and 6.43% fewer additions than the Log-MAP turbo decoder. The performance degradation is 0.01 dB at medium BER values. Furthermore, $MLM - \max^* - L1$ has 2.56% more LUT operations and 1.17% more additions than the Max-Log-MAP turbo decoder. The performance improvement is 0.1 dB at medium BER values.
- The variety of the proposed SISO decoding algorithms, lead us to believe that the gap between Max-Log-MAP and Log-MAP turbo decoding has now been closed.

Chapter 5

Improved Decoding Algorithms for Duo-Binary Turbo Codes

In this *third* Chapter of original work based on turbo codes, the so-called duo-(or double) binary turbo codes are investigated. As a practical application, the Digital Video Broadcasting Return Channel over Satellite (DVB-RCS) standard is adopted. Two different approaches to improved iterative decoding algorithms are described, one based on combination between Max-Log-MAP and Log-MAP decoding and the other on Constant Log-MAP decoding. Various computer simulations are run in the AWGN channel, mainly because of the fixed satellite communication link assumption in DVB-RCS. Decoding complexity estimation of the proposed algorithms is also given.

5.1 Introduction

As described in Section 2.5, duo-binary turbo codes have many advantages compared to the binary (i.e. classical) turbo codes, for equivalent implementation complexity [51, 104, 3]. The path error density is lowered and the decoder latency is divided by two, the influence of puncturing is less crucial, due to constituent encoders with higher coding rates, and interleaving between bit pairs and also inside bit pairs is supported. For that reason, they have been adopted by the Digital Video Broadcasting (DVB) Project to

provide full asymmetric two-way communications over the return channel for satellite (DVB-RCS) [105] as well as terrestrial (DVB-RCT) [106] networks respectively.

When decoding duo-binary turbo codes the same principles occur as in binary turbo codes. Either the conventional MAP or the Log-MAP algorithm can be used. Alternatively, to reduce further the decoding complexity, either SOVA, Max-Log-MAP or Constant Log-MAP algorithms can be applied. The only difference is that these algorithms should be modified to operate on symbols rather on bits. The concept of *symbol-based* iterative decoding is thus adopted [4], e.g. see Section 2.5.2. As mentioned also in Section 2.5.2, an approach to reduce the computational complexity is the iterative decoding of high rate convolutional codes based on the *dual code* [65, 66], but with some implementation disadvantages in the logarithmic domain.

As it was shown in most of the cases in Section 4.1, BER performance improvements to Max-Log-MAP turbo decoding are possible with small increase to the decoding complexity. In other way, decoding complexity reduction of Log-MAP turbo decoding is possible but with small BER performance degradation. This is the case of binary turbo codes. It is believed that the same decoding algorithm behaviour occurs in duo-binary turbo codes with symbol-based iterative decoding.

From the available literature on duo-binary turbo codes, the Max-Log-MAP algorithm has been widely used, due to the fact that is less complex and can approach the Log-MAP algorithm with up to 0.05 dB in performance degradation, as reported in [51, 104]. This is also mentioned in Section 5.3. However, it was not until October 2005 that the leading authors claimed in [3] that computer simulation results were not based on the *conventional* Max-Log-MAP but on the *improved* algorithm¹. As a consequence, our search for alternative decodings to Max-Log-MAP and Log-MAP algorithms suitable for duo-binary turbo codes has been proved to be quite reasonable, although having started earlier, before [3] came to our attention.

In this Chapter, two novel approaches to improved iterative decoding algorithms are described. The first approach is regarded as an *extension* of the SISO decoding algo-

¹A description of the *improved* Max-Log-MAP algorithm for binary turbo codes [91] can be found in Section 4.2.

rithms presented in Section 4.3 to duo-binary turbo codes. The second approach is based on an *efficient* Constant-Log-MAP algorithm for duo-binary turbo codes. In the first case, the trade-off between BER/FER performance and complexity is observed. In the second case, it is noticed BER/FER performance improvement when comparing to an existing algorithm with approximately the same computational complexity.

5.2 DVB-RCS Standard

The DVB-RCS standard specifies terminal-to-hub satellite communications over the return link [105] with transmission speeds ranging from 144 Kbps to 2 Mbps. A variety of frame sizes and coding rates are supported for different user applications, owing to a very flexible transmission scheme. This scheme is mainly composed of an 8-states duo-binary turbo code, which is considered to be a good alternative solution to the conventional serial concatenation of an outer RS code and an inner convolutional code. Some more details of the DVB-RCS standard, in terms of physical layer aspects, are given below.

- **Encoder.** A typical DVB-RCS turbo encoder is shown in Fig. 5.1. Twelve frame sizes (i.e. $N = 48, 64, 212, 220, 228, 424, 432, 440, 752, 848, 856$ and 864 bit pairs) and seven coding rates (i.e. $R = 1/3, 2/5, 1/2, 2/3, 3/4, 4/5$ and $6/7$) are supported. Thanks to the application of non-uniform interleaving, circular coding (i.e. tail-biting) and a simple puncturing device, a very powerful code is obtained. That is, it performs from 1 to 1.5 dB away from the AWGN channel capacity limit [4, 104], even by using frame sizes of relative small or medium length.

The generator polynomials of the constituent RSC codes are described in octal form as $(15, 13, 11)_o$ or $(1 + D + D^3, 1 + D^2 + D^3, 1 + D^3)$ representing the recursive polynomial, first parity bits and second parity bits respectively. In Fig. 5.1 ($u_{s,1}$, $u_{s,2}$) are the systematic bits, ($u_{p,1}$, $u_{p,2}$) are the first and second parity bits of the *first* constituent encoder and ($u'_{p,1}$, $u'_{p,2}$) are the first and second parity bits of the *second* constituent encoder.

For coding rates $R < 1/2$ either puncturing (e.g. when $R = 2/5$) or not (e.g. when

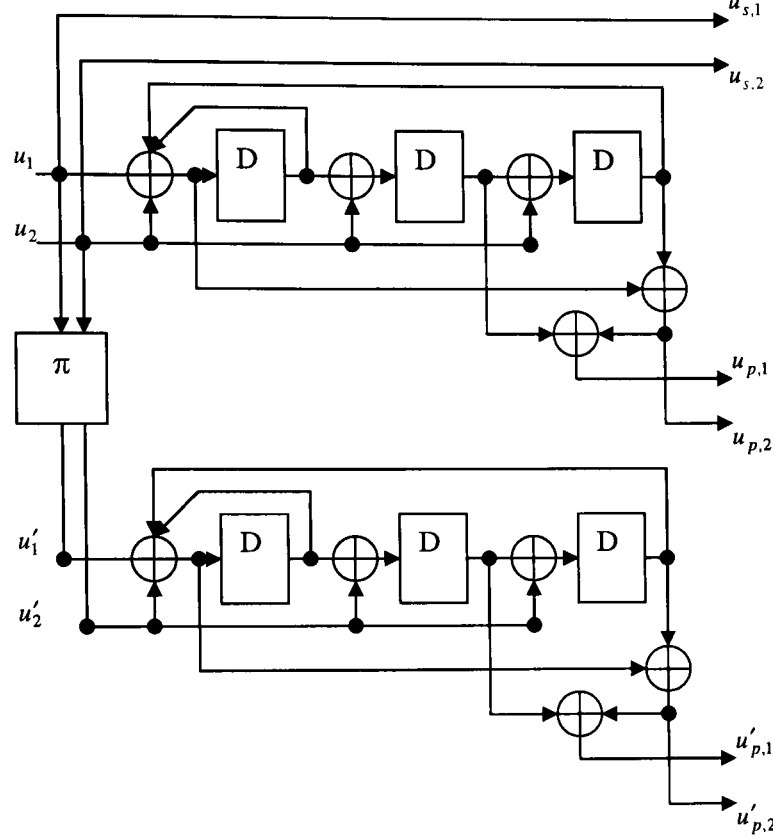


Figure 5.1: DVB-RCS turbo encoder.

$R = 1/3$) is applied only to the second parity bits of the constituent encoders $(u_{p,2}, u'_{p,2})$. For coding rates $R \geq 1/2$, the second parity bits $(u_{p,2}, u'_{p,2})$ are not transmitted, while appropriate puncturing is applied to the first parity bits of the constituent encoders $(u_{p,1}, u'_{p,1})$. It is noted that the second parity bit sequence provides an extra protection for coding rates $R < 1/2$, making the code more powerful.

- **Interleaving.** There are two kinds of interleaving. Let j be the input data pair sequence and i the output data pair sequence after interleaving. Also, assume that N is the total number of data pairs (i.e. frame size) with $j = i = 0, 1, \dots, N - 1$. The *first* level of interleaving (i.e. Level 1) is done inside bit pairs, as

$$\text{if } j \bmod 2 = 0, \text{ then } (u_{s,1}, u_{s,2}) = (u_{s,2}, u_{s,1}) \quad (5.1)$$

That is, the bit pair input sequence is inverted. The *second* interleaving (i.e.

Level 2) is done between bit pairs, as

$$i = (P_0 \cdot j + P + 1) \bmod N \quad (5.2)$$

The parameters P_0 and P depend on the frame size N . The latter one is computed from

$$\begin{aligned} \text{if } j \bmod 4 &= 0, \text{ then } P = 0 \\ \text{if } j \bmod 4 &= 1, \text{ then } P = N/2 + P_1 \\ \text{if } j \bmod 4 &= 2, \text{ then } P = P_2 \\ \text{if } j \bmod 4 &= 3, \text{ then } P = N/2 + P_3 \end{aligned} \quad (5.3)$$

The new parameters P_1 , P_2 and P_3 also depend on the frame size N . In our computer simulations, two frame sizes are considered, 53 and 752 bytes. That is, assumption of either ATM or MPEG frames respectively. Thus, if $N = 212$ bit pairs (i.e. 53 bytes), then $(P_0, P_1, P_2, P_3) = (13, 106, 108, 2)$. Similarly, if $N = 752$ bit pairs (i.e. 188 bytes), then $(P_0, P_1, P_2, P_3) = (19, 376, 224, 600)$.

- **Circular Coding.** In circular coding, which is also known as tail-biting, the encoder retrieves the initial state at the end of the encoding operation, so that data encoding may be represented by a circular trellis. A correspondence table, which is usually stored in memory, is used to determine the circulation state s_c from the final state s_N , as $s_c = (I + G^N)^{-1} \cdot s_N$, where I is the unity matrix and G is the generator matrix of the code respectively. It is noted that the encoded data block N should not be a multiple of the period L of the encoder's recursive generator polynomial, as it would result in $G^L = I$ [4].

Assuming the two frame sizes as above, this correspondence is as following. If $N = 212$ bit pairs and $s_N = (0, 1, 2, 3, 4, 5, 6, 7)$, then $s_c = (0, 3, 7, 4, 5, 6, 2, 1)$. Similarly, if $N = 752$ bit pairs and s_N as previously, then $s_c = (0, 5, 3, 6, 2, 7, 1, 4)$.

- **Transmission Order.** The transmission order is composed of two types. In the *natural* order, the systematic bit pairs $(u_{s,1}, u_{s,2})$ are transmitted first, followed

by the *first* parity bits $(u_{p,1}, u'_{p,1})$ and then by the *second* parity bits $(u_{p,2}, u'_{p,2})$ of the two constituent encoders. In the *reverse* order, the *first* parity bits $(u_{p,1}, u'_{p,1})$ are transmitted first, followed by the *second* parity bits $(u_{p,2}, u'_{p,2})$ and then by the systematic bit pairs $(u_{s,1}, u_{s,2})$ of the two constituent encoders.

- **Modulation.** After the turbo encoding operation and appropriate transmission order, QPSK modulation is used with *Gray* coding and *I/Q* symbol mapping. The *I* channel corresponds to the encoded bits concerning the first component encoder and the *Q* channel corresponds to the encoded bits concerning the second component encoder respectively.

5.3 Relevant Work on Duo-Binary Turbo Codes and Related DVB-RCS Standard Improvements

The advantages of duo-binary turbo codes have been addressed for the first time in [51, 104] and a revised version with up-to-date research work can be found in [3]. As shown in [3], non-binary constituent RSC codes are 0.5 dB superior to the related binary ones, at BER below 10^{-4} and high coding rates, e.g. $R = 2/3, 3/4$ and $6/7$. Moreover, the *improved* Max-Log-MAP algorithm for duo-binary turbo codes has only 0.05 dB performance degradation compared to the Log-MAP algorithm (1504 bits frame, i.e. MPEG frames, coding rate $R = 4/5$, AWGN channel, QPSK modulation and 8 decoding iterations). Obviously, the Max-Log-MAP algorithm implementation gives decoding complexity savings. When extending the 8-states duo-binary turbo encoder, in the form of DVB-RCS, to 16-states the minimum distance is increased from 30% to 50%, depending on the coding rate. The corresponding performance improvement is from 0.5 to 1 dB at FER below 10^{-6} , approaching the theoretical limits within 0.7 to 1 dB (424 or 1504 bits frame, i.e. ATM or MPEG frames, coding rates $R = 1/2, 2/3$ and $3/4$, AWGN channel, QPSK modulation and 8 decoding iterations). The main drawback is that the decoding implementation complexity is increased by 50%. This approach was also described in [107].

Performance evaluation of the DVB-RCS turbo code can be found in [104]. As a prac-

tical application, this FEC scheme is to be used in the new on-board satellite processing system *Skyplex* from EUTELSAT. Both software and hardware implementation results were reported in terms of FER performance (424 or 1504 bits frame, i.e. ATM or MPEG frames, coding rates from $R = 1/2$ to $6/7$, AWGN channel, QPSK modulation, 4 input quantization bits, Max-Log-MAP algorithm and 8 decoding iterations). Finally, user bit rates of up to 4 Mbps with 6 decoding iterations were reported in single-chip FPGA.

VLSI implementation issues for the DVB-RCS turbo code were described in [108], based on architecture design in pipelined structure, quantization and new normalisation approach, i.e. rescaling. The resulting FPGA-based decoder was able to work up to 7 Mbps in terms of data rate with 6 decoding iterations. In addition, this work was extended in [109] by applying early stopping criteria to increase the throughput of the DVB-RCS turbo decoder in a multi-channel processing scheme, such as in the base station of a mobile communication system. Two approaches were proposed, one in serial and the other in parallel processing, both suitable for hardware implementation.

The same primary author presented for the first time the Constant Log-MAP decoding algorithm for duo-binary turbo codes, e.g. in the form of DVB-RCS, to reduce the implementation complexity of the iterative Log-MAP decoding [4]. It was claimed that the resulting algorithm has negligible performance loss compared to the Log-MAP algorithm, similar to the binary case (424 bits frame, i.e. ATM frames, coding rates from $R = 1/3$ to $6/7$, AWGN channel, QPSK modulation and 8 decoding iterations). This work has motivated the second proposed algorithm for duo-binary turbo codes, which is presented in Section 5.5.

In a later work, the performance of the standardised DVB-RCS turbo code was further improved. The authors in [12] proposed a novel interleaver design method based on the message-passing principle to increase the loop length distribution formed by a given interleaver. Computer simulation results indicated performance improvements of up to 0.2 dB at BER below 10^{-6} (1504 bits frame, i.e. MPEG frames, coding rates $R = 1/2, 2/3$ and $4/5$, AWGN channel, QPSK modulation and 8 decoding iterations).

The above method was extended in [110] for variable block sizes, including satellite

links with internet protocol (IP) frames, i.e. 40 bytes. Computer simulation results with new size optimised interleavers compared to the DVB-RCS standard have shown the absence of error floor at BER of 10^{-6} (frame size of 40, 159, 216, 265 and 512 bytes, coding rates $R = 1/2, 2/3$ and $3/4$, AWGN channel, QPSK modulation, Max-Log-MAP algorithm and 6 decoding iterations). The importance of this method is to support satellite multimedia communications with adaptive coding rate using the existing DVB-RCS standard.

Performance improvements adapting dithered relative prime (DRP) interleavers for the DVB-RCS turbo code were reported in [111]. That guarantees increased minimum distance of the new interleavers. In the same reference, a minimum distance measuring method was proposed. Coding gains from 0.15 dB to 0.25 dB at FER below 10^{-5} were possible to the resulting turbo code performance (424 or 1504 bits frame, i.e. ATM or MPEG frames, coding rate $R = 1/3$, AWGN channel, QPSK modulation and 8 decoding iterations). In this case, the *enhanced* Max-Log-MAP algorithm was used from [11].

We recall from Section 4.2 that a simple solution to improve the Max-Log-MAP iterative decoding is the *enhanced* Max-Log-MAP algorithm [11]. This is based on scaling the extrinsic information with a constant factor. Assuming the DVB-RCS turbo code and high to medium BER/FER values, the resulting algorithm has performance loss of 0.1 dB against the Log-MAP iterative decoder (424 or 1504 bits frame, i.e. ATM or MPEG frames, coding rate $R = 1/3$, AWGN channel, QPSK modulation and 8 decoding iterations). This performance behaviour also verifies the results obtained in [3], as the two algorithms, although having different terminology, are identical to each other.

In another approach [107], which was later described in [3] from the same primary author, a 16-states duo-binary turbo encoder was presented, as an alternative to the standardised DVB-RCS turbo encoder with 8-states. The turbo decoder was implemented on a single FPGA chip, showing the absence of error floor at FER of 10^{-7} , at the expense of double decoding complexity (1504 bits frame, i.e. MPEG frames, coding rates $R = 1/2$ and $2/3$, AWGN channel, QPSK modulation, 4 input quantization bits,

improved Max-Log-MAP algorithm and 8 decoding iterations). Finally, a throughput of 2 Mbps was achieved with 8 decoding iterations.

5.4 SISO Algorithms Based on Max/Max* Operation Replacement for DVB-RCS Turbo Code

Here, novel SISO decoding algorithms for duo-binary turbo codes are presented by combining the *max* operation of the Max-Log-MAP algorithm and the *max** operation of the Log-MAP algorithm in an appropriate way [112]. The decoding complexity is estimated and computer simulation results are shown for the DVB-RCS turbo code.

5.4.1 Motivation

In Section 4.3.2 novel sub-optimum SISO decoding algorithms were obtained for *binary* turbo codes with good compromise between performance and decoding complexity when comparing to Max-Log-MAP and Log-MAP iterative decoding. This was done by *max/max** operation replacement to either the forward/backward recursion or soft-output computation.

Motivated by this approach, it is proposed to extend the *max/max** operation replacement in a similar way to *duo-binary* turbo codes [112]. The resulting sub-optimum SISO decoding algorithms are expected to compromise the performance and decoding complexity, such as in the binary case.

5.4.2 Proposed SISO Decoding Algorithms and Complexity Estimation

Similar to Section 4.3.2, four SISO decoding algorithms are obtained in different combinations of the *max/max** operation needed for both $\tilde{\alpha}$, $\tilde{\beta}$ and LLR computation. The same algorithm notation is also adapted from the same Section, i.e. *SISO-A*, *SISO-B*, *SISO-C* and *SISO-D*. The difference is that symbol-based iterative decoding is now applied to the considered algorithms, e.g. see Section 2.5.2.

Table 5.1: Decoding complexity estimation of SISO decoding algorithms based on \max/\max^* operation replacement. It is assumed a binary turbo encoder with memory order equal to three.

	\max operations	LUT operations	additions (total)
Max-Log-MAP	38	//	91
SISO-A	38	12	103
SISO-B	38	14	105
SISO-C	38	24	115
SISO-D	38	26	117
Log-MAP	38	38	129

Complexity issues between duo-binary and binary turbo codes have been discussed in [51, 104]. Duo-binary turbo codes offer twice the bit rate at the decoder input but require around twice the computational complexity, due to the double number of trellis branch transitions. As a consequence, the equivalent complexity per decoded bit is approximately the same.

In Table 5.1 the decoding complexity of the proposed SISO decoding algorithms is estimated per information bit and decoding iteration. For comparison, the Max-Log-MAP and Log-MAP algorithms are also considered. It is noted that the DVB-RCS turbo encoder has 8-states. Therefore, calculations are shown for an equivalent binary turbo encoder with memory order equal to three (e.g. see Section 4.3.2).

The *relative* decoding complexity increase (or decrease) of the proposed SISO decoding algorithms with respect to Max-Log-MAP (or Log-MAP) turbo decoder is reported in Table 5.2. The comparison consists of \max , LUT operations and number of additions, similar to [7]. An equivalent binary turbo encoder is assumed with memory order equal to three (e.g. see Section 4.3.2).

Table 5.2: Relative decoding complexity comparison example of SISO decoding algorithms with respect to Max-Log-MAP and Log-MAP turbo decoder. It is assumed a binary turbo encoder with memory order equal to three.

	$max+$ LUT ops. (increase) w.r.t. Max-Log-MAP	$max+$ LUT ops. (decrease) w.r.t. Log-MAP	additions (increase) w.r.t. Max-Log-MAP	additions (decrease) w.r.t. Log-MAP
SISO-A	31.58%	68.42%	13.19%	20.16%
SISO-B	36.84%	63.16%	15.38%	18.60%
SISO-C	63.16%	36.84%	26.37%	10.85%
SISO-D	68.42%	31.58%	28.57%	9.30%

5.4.3 Computer Simulation Results

Both BER and FER computer simulation results are reported in Figs. 5.2-5.5. It is assumed the DVB-RCS turbo encoder, QPSK modulation and the AWGN channel. Eight decoding iterations are considered with Max-Log-MAP, *SISO – A, B, C, D* and Log-MAP algorithms. Four coding rates (i.e. $R = 1/3, 1/2, 2/3$ and $4/5$) and two frame sizes are assumed. BER/FER results obtained with ATM frames (i.e. 424 bits frame) are shown in Figs. 5.2-5.3 and with MPEG frames (i.e. 1504 bits frame) are shown in Figs. 5.4-5.5 respectively.

The simulation parameters are chosen in such way that the observed performance of the four decoding algorithms is independent on the selection of coding rate and frame size. The case of two decoding iterations, such as in Section 4.3.3, is not considered, although the resulting performance behaviour is the same, i.e. independent on the number of decoding iterations.

Discussion

From the four Figures it is verified that the more (or less) complex the SISO decoding algorithm with respect to Max-Log-MAP (or Log-MAP) algorithm, the better (or worse) it performs. The performance gap between Max-Log-MAP and Log-MAP decoding becomes smaller as the coding rate is increased. This is explained by the puncturing

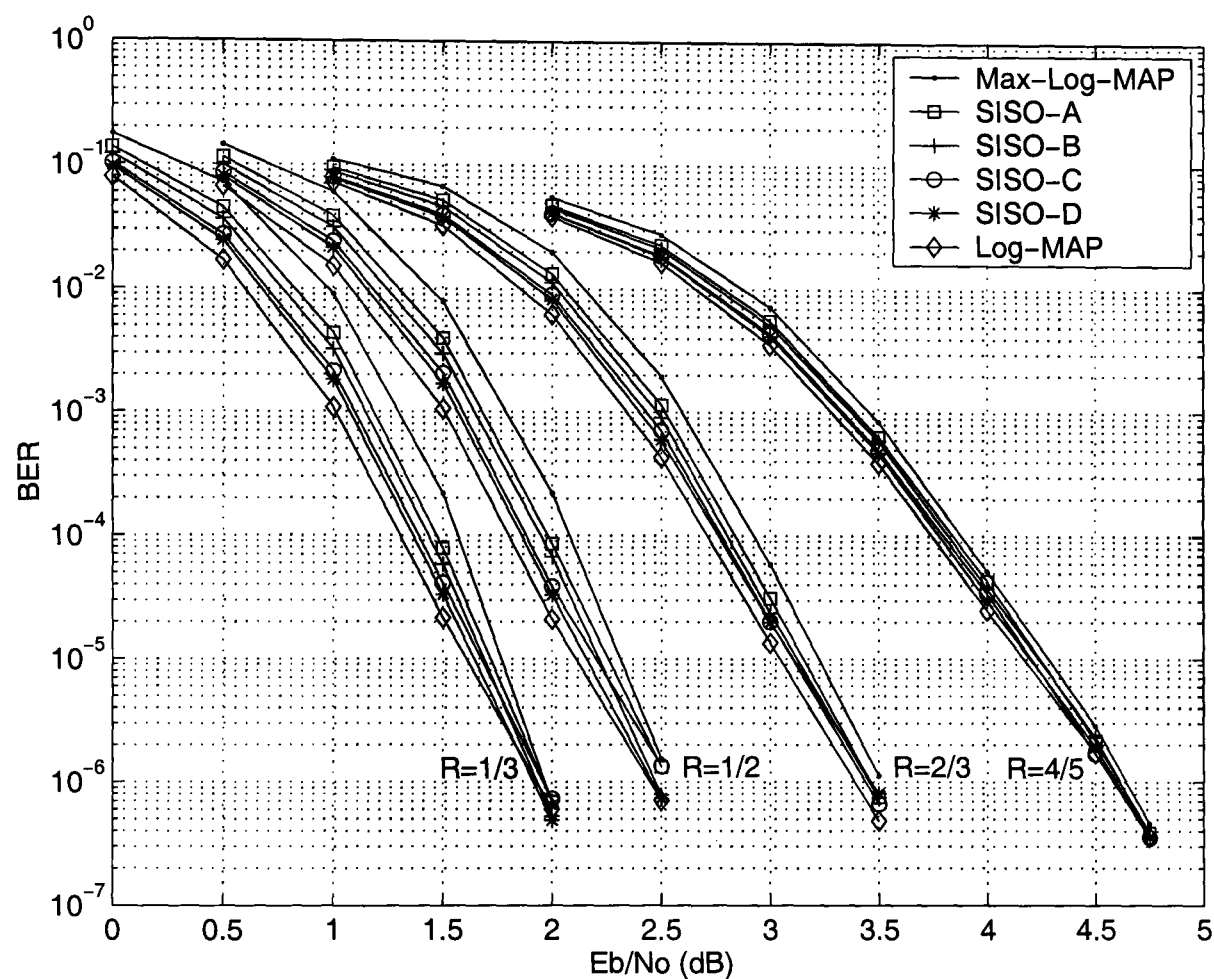


Figure 5.2: BER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.

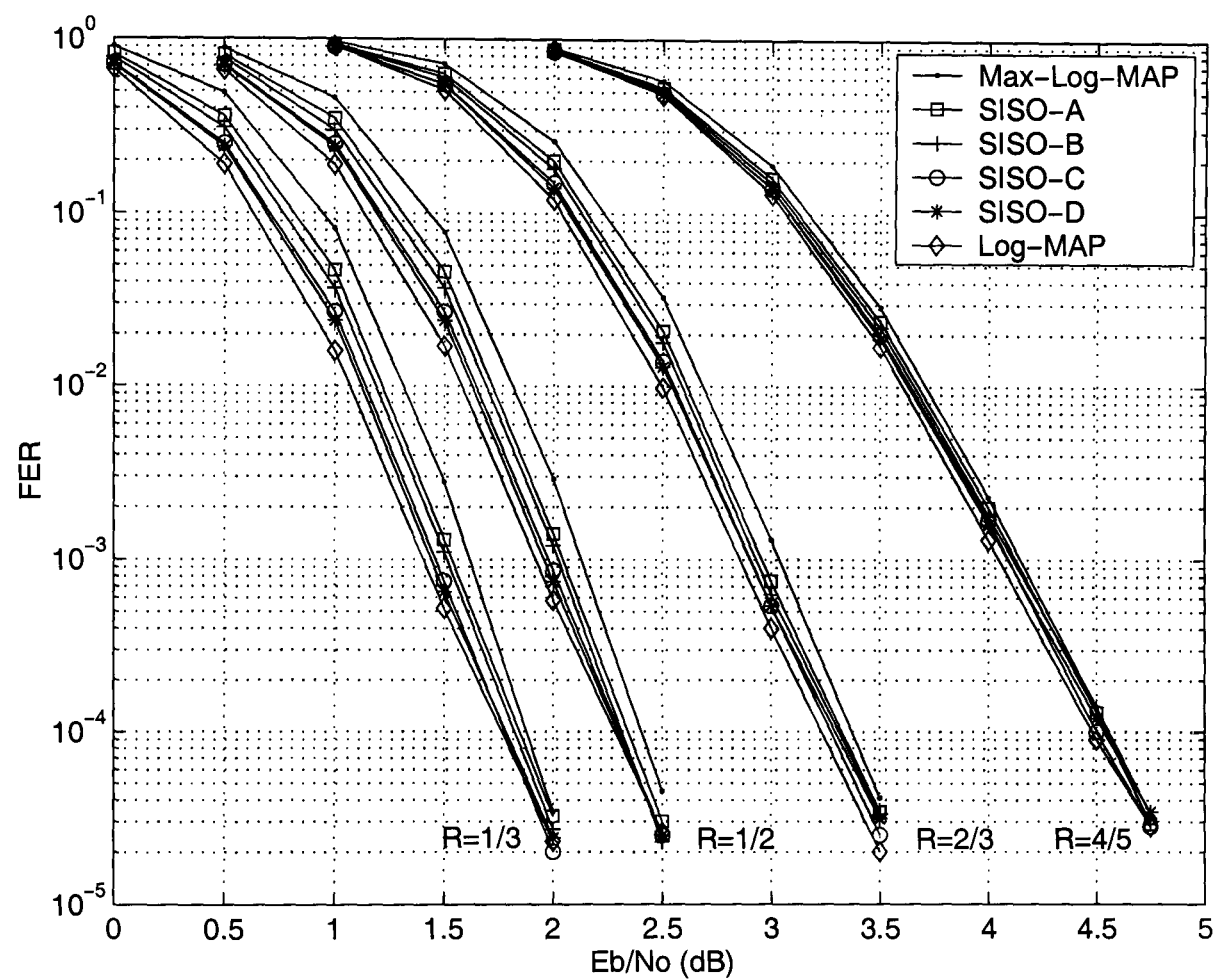


Figure 5.3: FER performance comparison of SISO decoding algorithms based on max/max^* operation replacement. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.

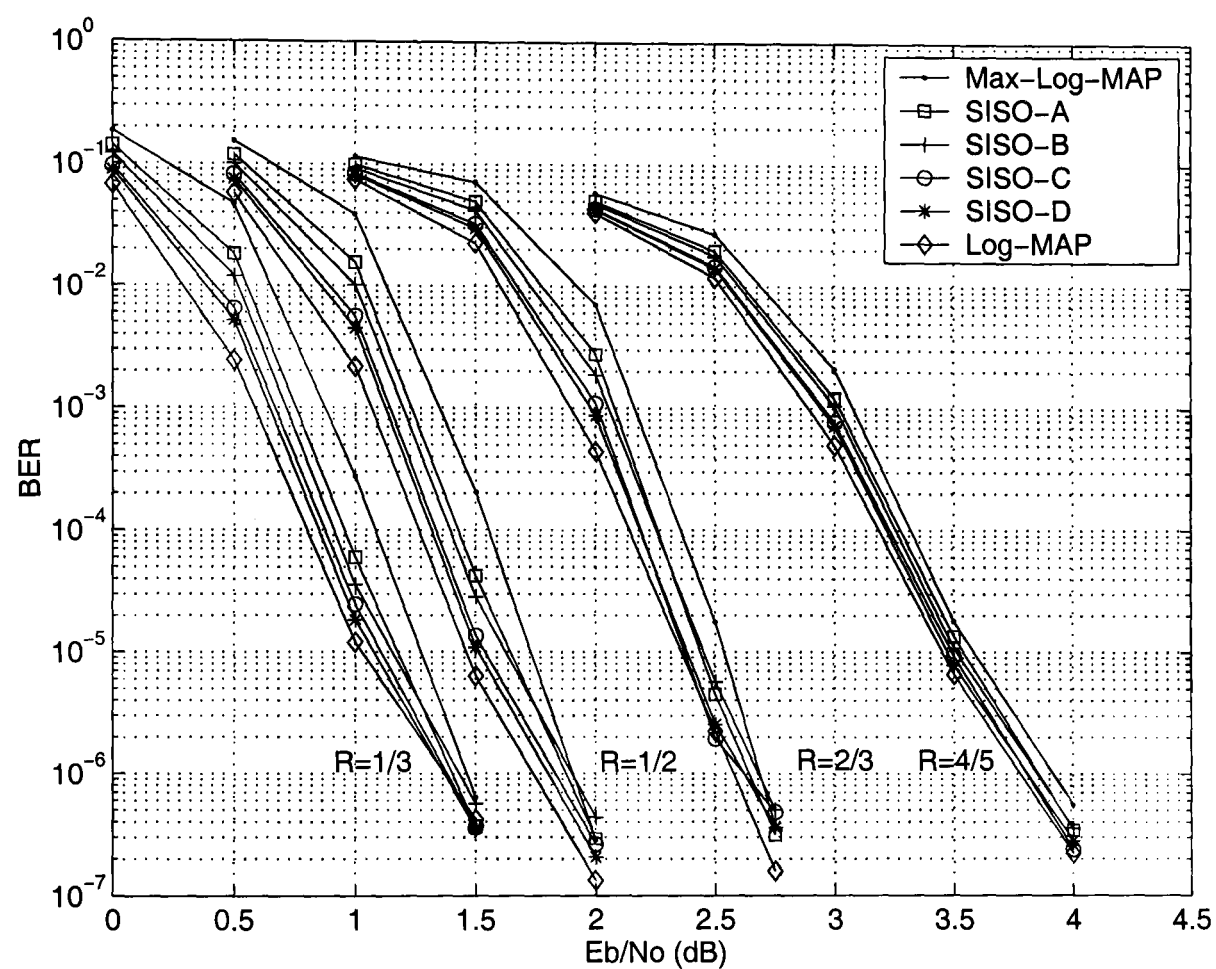


Figure 5.4: BER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.

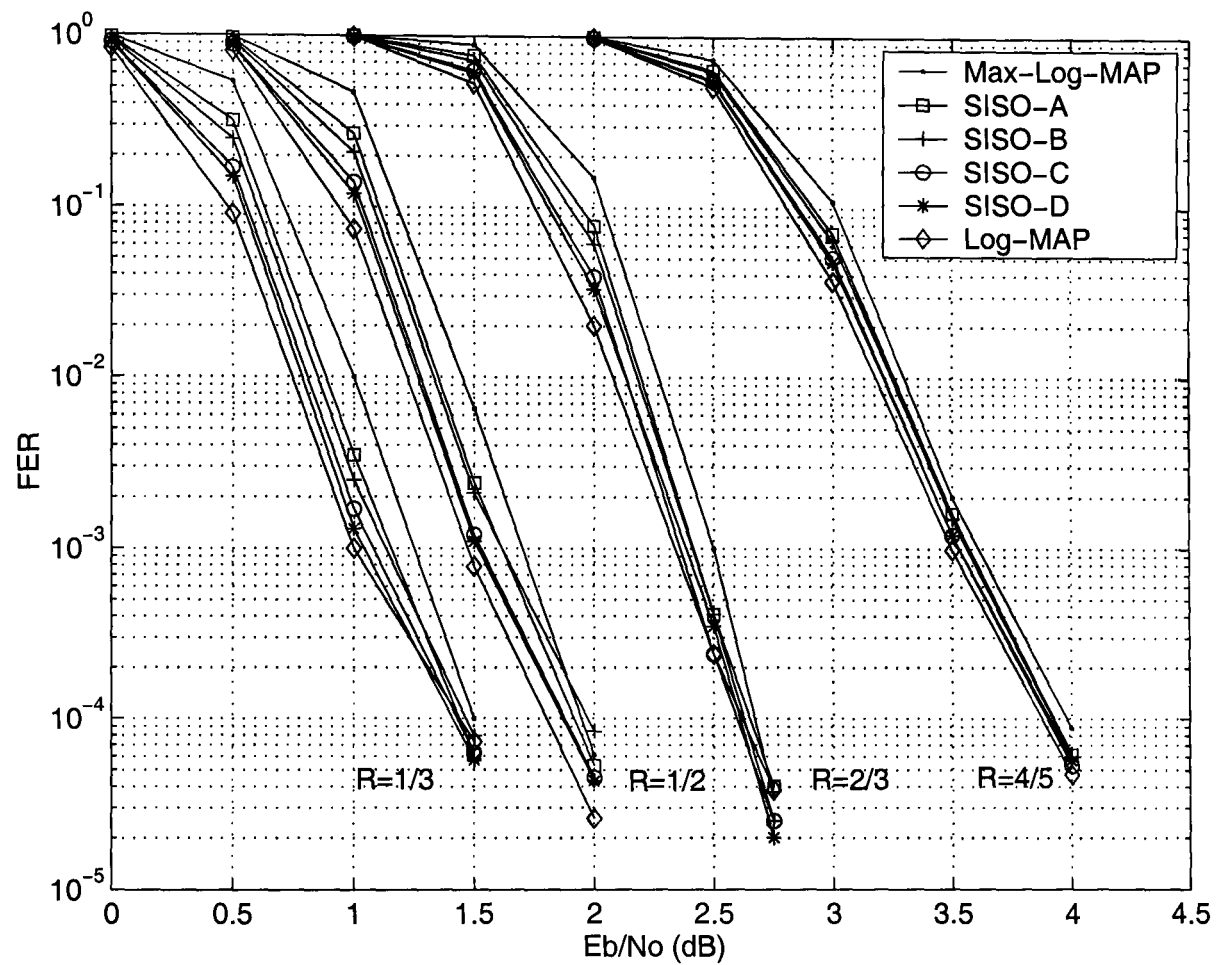


Figure 5.5: FER performance comparison of SISO decoding algorithms based on \max/\max^* operation replacement. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.

technique that makes the code less powerful. As a comparison, the simulation results with Max-Log-MAP iterative decoding from [3, 11, 104, 12] are in agreement with those presented in the four Figures. Similar to Section 4.3.3, SISO-B achieves a good trade-off between BER/FER performance and complexity with respect to the Max-Log-MAP iterative decoding. On the other hand, SISO-C achieves a good trade-off between BER/FER performance and complexity with respect to the Log-MAP iterative decoding.

For example, at low coding rates and medium BER/FER values, SISO-B improves the iterative Max-Log-MAP performance up to 0.13 dB, while SISO-C degrades the iterative Log-MAP performance up to 0.08 dB. In the first case, the relative complexity increase is 36.84% more LUT operations and 15.38% extra additions. In the second case, the relative complexity decrease is 36.84% fewer LUT operations and 10.85% fewer additions.

As a conclusion, the performance behaviour of the proposed SISO decoding algorithms is the same considering duo-binary turbo codes instead of binary ones. Thus, these algorithms provide a reasonable alternative solution to symbol-based Max-Log-MAP and Log-MAP iterative decoding.

5.5 Efficient Constant Log-MAP Decoding Algorithm for DVB-RCS Turbo Code

Here, a novel Constant Log-MAP decoding algorithm for duo-binary turbo codes is proposed [113] by computing the correcting factor in a different way from an existing algorithm presented in [4]. When the proposed algorithm is compared against the Log-MAP decoding, it is observed negligible performance degradation, exactly as in binary turbo codes. This is in contrast to the algorithm from [4] where a non-negligible performance degradation is observed. Decoding complexity estimation is given and computer simulation results are shown for the DVB-RCS turbo code with symbol-based iterative decoding.

5.5.1 Motivation

As reported in Section 5.3, the authors in [4, 108] presented for the first time the Constant Log-MAP decoding algorithm for duo-binary turbo codes to reduce the implementation complexity of the Log-MAP decoding. However, the exact performance against other iterative decoding algorithms, i.e. Max-Log-MAP and Log-MAP, is not clear. For example, the performance of the Constant Log-MAP decoding algorithm seems to be close to Max-Log-MAP decoding in [4], while the same algorithm is reported to perform close to Log-MAP decoding in [108].

Motivated by this performance mismatch, we propose an *efficient* Constant Log-MAP decoding algorithm suitable for duo-binary turbo codes [113], which is found to have the same computational complexity, but better performance than the algorithm from [4]. The difference between the two algorithms is in the way that the correcting factor is computed. In the rest of the Chapter, we refer to the Constant Log-MAP decoding algorithm from [4], as *Type-I Constant Log-MAP* and to the proposed Constant Log-MAP decoding algorithm, as *Type-II Constant Log-MAP*.

5.5.2 Proposed Decoding Algorithm and Complexity Estimation

We recall from Section 4.2 that the Constant Log-MAP decoding algorithm for *binary* turbo codes makes use of a look-up table of two values, instead of the more usually assumed eight values [93, 94]. That reduces the implementation complexity against the Log-MAP decoding algorithm with negligible performance degradation, e.g. 0.03 dB at high to medium BER values. The area savings are around 40% in 0.5 μm CMOS [93], while the memory size savings of the look-up table are 75% in FPGA or DSP implementations, as two instead of eight values are assumed.

The Constant Log-MAP decoding algorithm for *binary* turbo codes simplifies the \max^* operator that is used in the computation of forward/backward recursion and soft-output, according to

$$\max^*(x, y) = \max(x, y) + c \quad (5.4)$$

Table 5.3: Overall complexity estimation of one constant Log-MAP operation for binary turbo codes.

Constant Log-MAP	\max operations	additions	comparisons
$\max(x, y)$	1		
$ x - y < 2$		1	1
$\max(x, y) + c$		1	
<i>Total</i>	1	2	1

where the correcting factor c takes two possible values

$$c = \begin{cases} 3/8, & \text{if } |x - y| < 2 \\ 0 & \text{otherwise} \end{cases} \quad (5.5)$$

In the first instance, one simplified \max^* operator from Eq. (5.4) requires two operations; one \max operation and one addition. However, it is interesting to show how many operations are required for the computation of the correcting factor in Eq. (5.5) as well.

The overall complexity estimation of one Constant Log-MAP operation is summarized in Table 5.3. In this Table, as in the subsequent Tables, both the absolute operation and the processing delay are not counted.

In [4] it was proposed to use the simplified \max^* operator over *four* values in case of *duo-binary* turbo codes with symbol-based iterative decoding. That means the Type-I Constant Log-MAP operates as

$$\max^*(x, y, z, w) = \max(x, y, z, w) + c_0 \quad (5.6)$$

The correcting factor c_0 is computed from

$$c_0 = \begin{cases} 5/8, & \text{if } \max(|a|, |b|, |c|) < 2 \\ 0 & \text{otherwise} \end{cases} \quad (5.7)$$

Table 5.4: Overall complexity estimation of one Type-I Constant Log-MAP operation for duo-binary turbo codes.

Type-I Constant Log-MAP	\max operations	additions	comparisons
$\max(x, y, z, w)$	3		from 1 to 3 (average=2)
$- a = x - \max(x, y, z, w)$		1	
$- b = y - \max(x, y, z, w)$		1	
$- c = z - \max(x, y, z, w)$		1	
$\max(a , b , c) < 2$	2		1
$\max(x, y, z, w) + c_0$		1	
<i>Total</i>	5	4	from 2 to 4 (average=3)

where $-|a|$, $-|b|$ and $-|c|$ are three values among $x - \max(x, y, z, w)$, $y - \max(x, y, z, w)$, $z - \max(x, y, z, w)$ or $w - \max(x, y, z, w)$.

One simplified \max^* operator from Eq. (5.6) requires four operations; three \max operations, assuming that the \max operator is applied over pairs of values, and one addition. Similarly, we take into account the operations that are required for the computation of the correcting factor in Eq. (5.7).

The overall complexity estimation of one Type-I Constant Log-MAP operation is summarised in Table 5.4. We notice that after the $\max(x, y, z, w)$ computation, we need to identify which of the four values among x , y , z or w it corresponds to. For that reason, one to three comparisons may occur in a serial mode. That correspondence is then used to compute the exact values of $-|a|$, $-|b|$ and $-|c|$. For example, in the worst case of $\max(x, y, z, w) = w$, three comparisons with x , y and z are needed, after the $\max(x, y, z, w)$ computation.

We have found that it is better to adapt the Constant Log-MAP decoding algorithm from binary turbo codes to duo-binary turbo codes, rather than implementing as in the previous way. That means the \max^* operator is processed over *pairs* of values, according to

Table 5.5: Overall complexity estimation summary of one Constant Log-MAP operation for duo-binary turbo codes.

Constant Log-MAP	\max operations	additions	comparisons	Total
Type-I	5	4	3 (average)	12 (average)
Type-II	3	6	3	12

$$\max^*(x, y, z, w) = \max^*\{\max^*(x, y), \max^*(z, w)\} \quad (5.8)$$

Using Eq. (5.4) in (5.8), the Type-II Constant Log-MAP operates as

$$\max^*(x, y, z, w) = \max^*\{\max(x, y) + c_1, \max(z, w) + c_2\} \quad (5.9)$$

or

$$\max^*(x, y, z, w) = \max\{\max(x, y) + c_1, \max(z, w) + c_2\} + c_3 \quad (5.10)$$

where the three correcting factors c_1 , c_2 , and c_3 are computed from Eq (5.5) respectively.

One simplified \max^* operator from Eq. (5.10) requires six operations; three \max operations and three additions. This is because it makes use of the binary Constant Log-MAP operator from Eq. (5.4) three times. Therefore, the complexity of one Type-II Constant Log-MAP operation is three times the complexity of one binary Constant Log-MAP operation.

The overall complexity estimation of one Type-II Constant Log-MAP operation is shown in the last row of Table 5.5.

As reported in Section 5.4.2, complexity issues between duo-binary and binary turbo codes have been discussed in [51, 104]. Duo-binary turbo codes offer twice the bit rate at the decoder input but require around twice the computational complexity. As a consequence, the equivalent complexity per decoded bit is approximately the same.

Therefore, the application of the Constant Log-MAP decoding algorithm to duo-binary turbo codes yields the same complexity savings as in the binary case.

It is interesting to compare the complexity of one simplified \max^* operator that is used in the two types of Constant Log-MAP decoding algorithms. One simplified \max^* operator of Type-II Constant Log-MAP requires six operations, while one simplified \max^* operator of Type-I Constant Log-MAP requires four operations. This seems to be a 50% increase in complexity. However, when the comparison includes the operations needed to compute the correcting factor, both the algorithms require twelve operations. This is shown in Table 5.5.

From that Table, it is concluded that both the algorithms have the *same* overall computational complexity. However, from the simulation results presented in the next Section, only the Type-II Constant Log-MAP is found to approach the Log-MAP decoding at negligible performance degradation. It is thus considered to be an *efficient* decoding algorithm.

5.5.3 Computer Simulation Results

In a similar way to Section 5.4.3, computer simulations have been carried out assuming the DVB-RCS turbo code, QPSK modulation and the AWGN channel. Both BER and FER results are reported in Figs. 5.6-5.9. Eight decoding iterations are considered with Max-Log-MAP, Type-I Constant Log-MAP, Type-II Constant Log-MAP and Log-MAP algorithms. Four coding rates (i.e. $R = 1/3, 1/2, 2/3$ and $4/5$) and two frame sizes are assumed. BER/FER results obtained with ATM frames (i.e. 424 bits frame) are shown in Figs. 5.6-5.7 and with MPEG frames (i.e. 1504 bits frame) are shown in Figs. 5.8-5.9 respectively.

Similarly to Section 5.4.3, the simulation parameters are chosen in such way that the observed performance of the four decoding algorithms is independent on the selection of coding rate and frame size. The case of two decoding iterations is not considered, although the resulting performance behaviour is the same, i.e. independent on the number of decoding iterations.

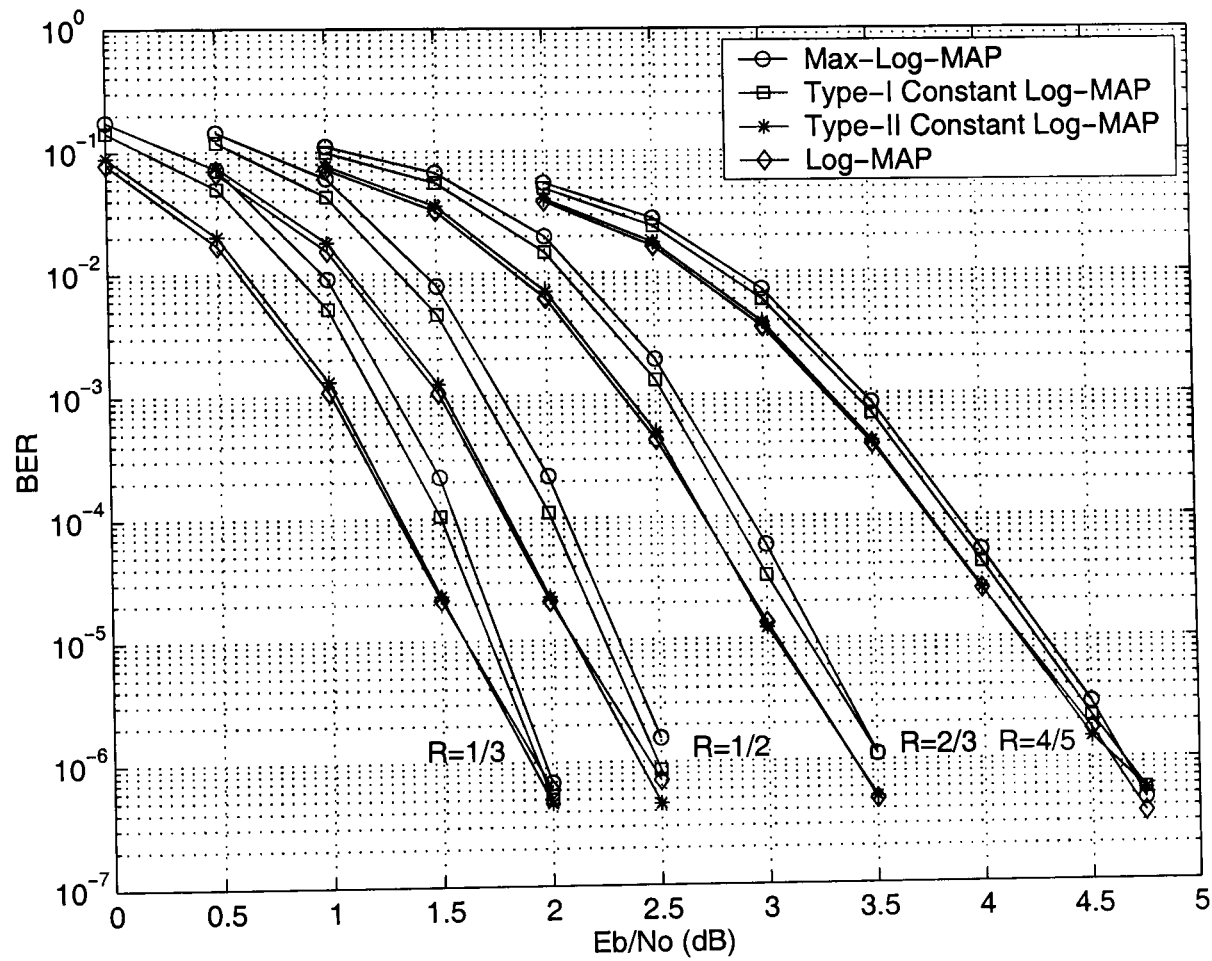


Figure 5.6: BER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.

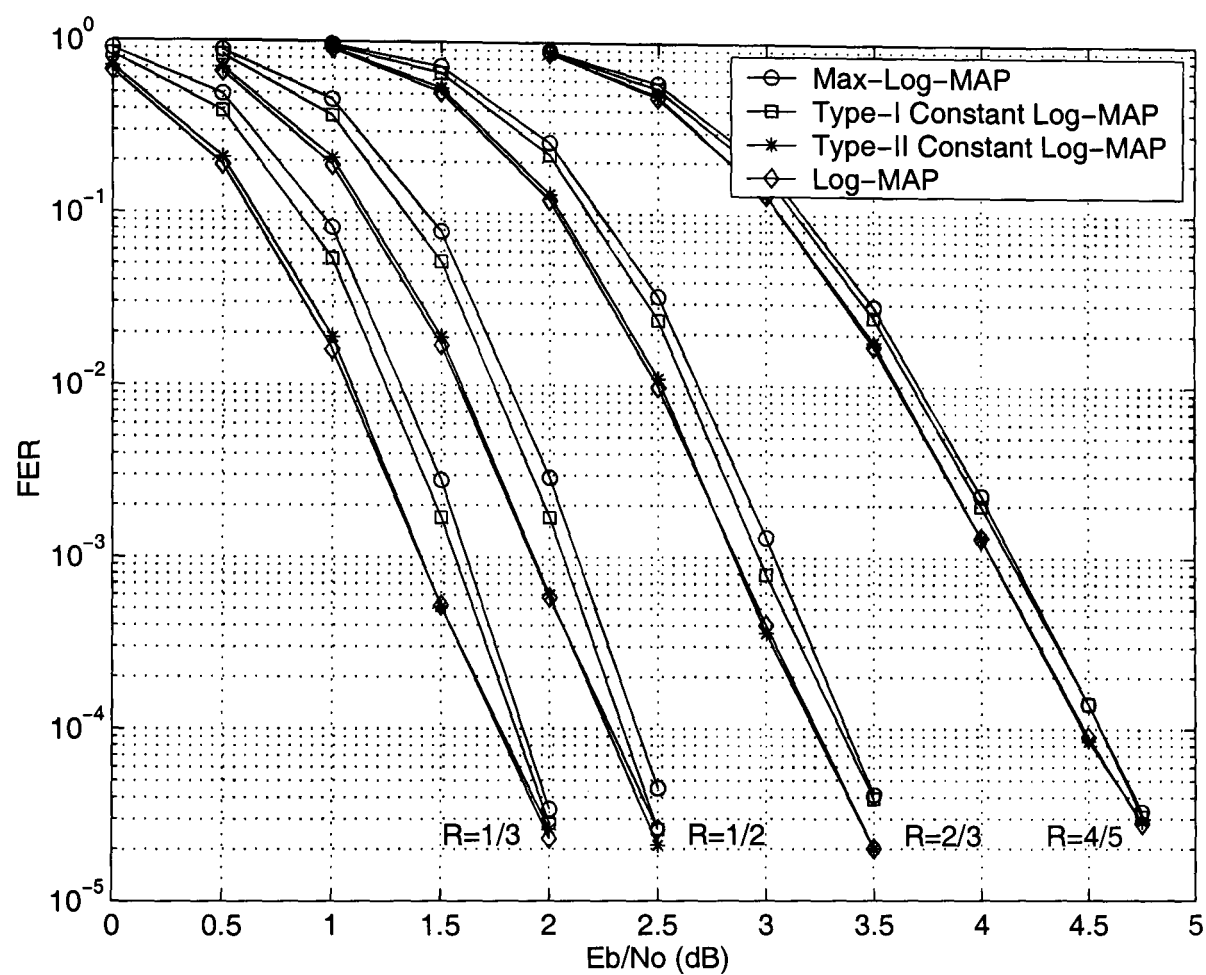


Figure 5.7: FER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, ATM frame size, i.e. 424 bits, and 8 decoding iterations in the AWGN channel.

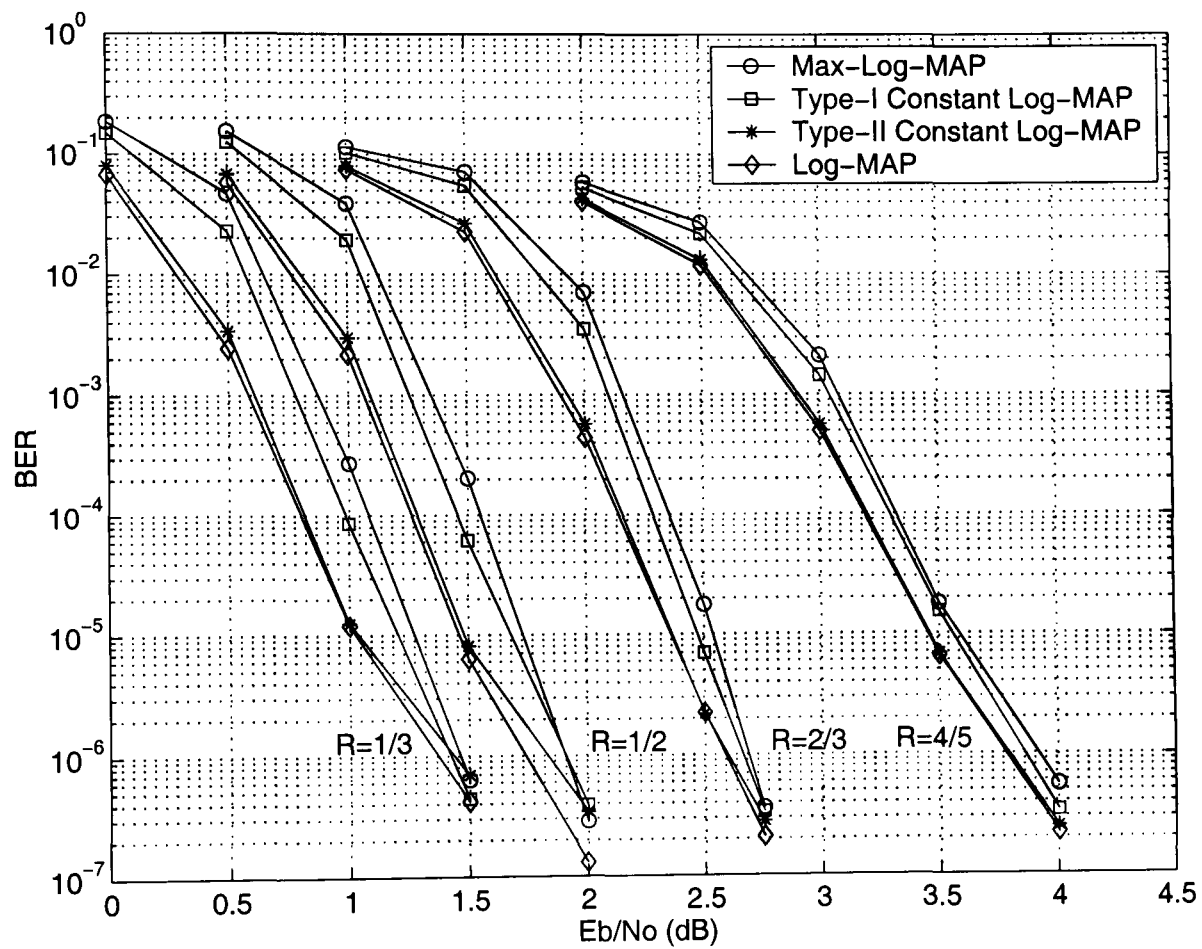


Figure 5.8: BER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.

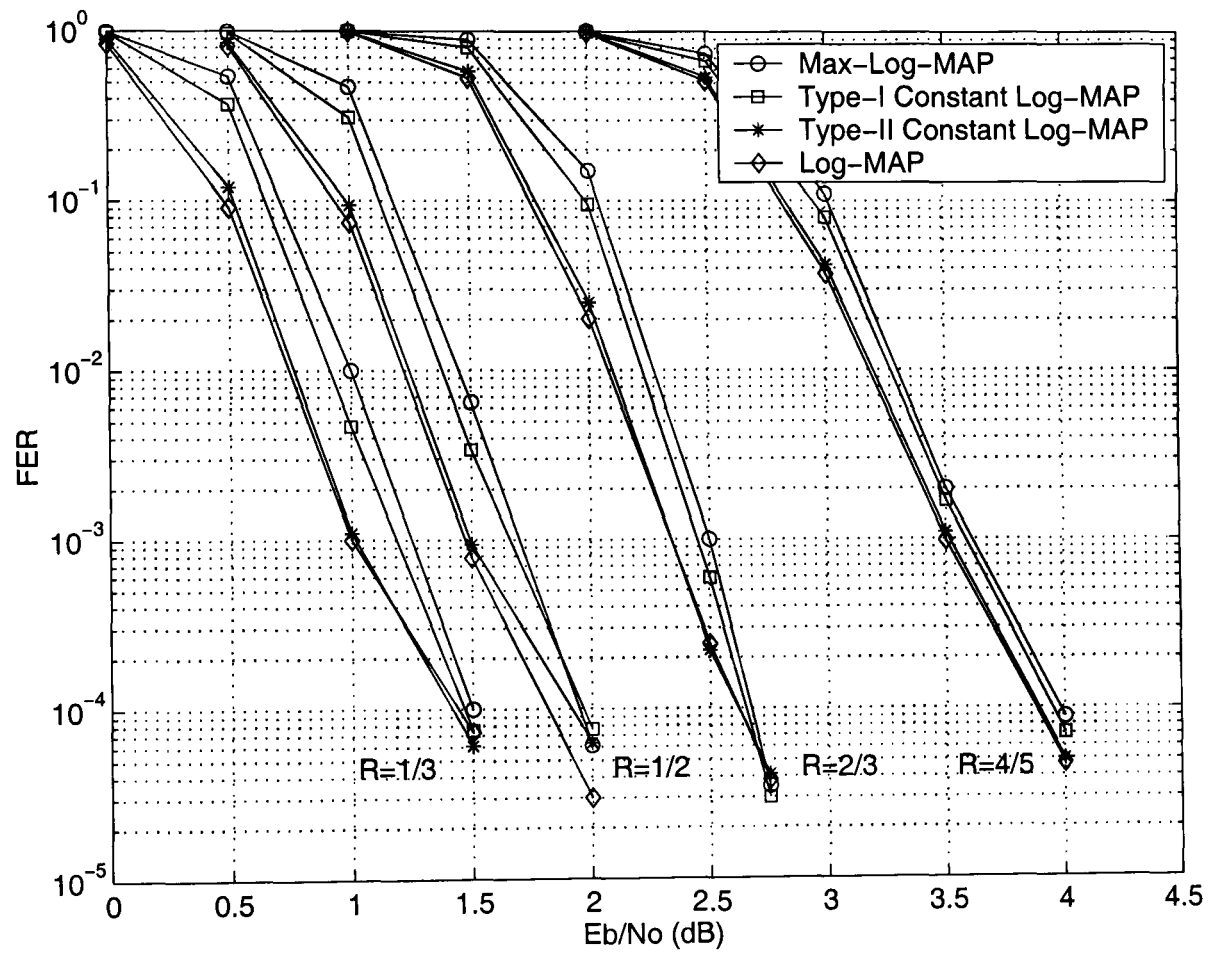


Figure 5.9: FER performance comparison of two Constant Log-MAP iterative decoding algorithms. DVB-RCS turbo encoder, different coding rates, MPEG frame size, i.e. 1504 bits, and 8 decoding iterations in the AWGN channel.

Discussion

From the four Figures it is noticed that the Type-II Constant Log-MAP provides a maximum performance improvement of 0.2 dB at high to medium BER/FER values. This is when the algorithm is compared to the Type-I Constant Log-MAP. However, the performance improvement becomes smaller, as the coding rate is increased. This has been explained by the puncturing technique that makes the code less powerful. On the other hand, it is the Type-II Constant Log-MAP rather than the Type-I Constant Log-MAP, that performs close to Log-MAP decoding. The performance degradation is less than 0.02 dB, similar to the binary case [93]. As a comparison, the simulation results with Max-Log-MAP iterative decoding from [3, 11, 104, 12] are in agreement with those presented in the four Figures.

It is concluded that the Type-I Constant Log-MAP provides small performance improvement against the Max-Log-MAP decoding, while there is non-negligible performance degradation against the Log-MAP decoding, similar to [4]. On the other hand, the Type-II Constant Log-MAP is found to have superior performance compared to the Type-I Constant Log-MAP and is very close to the Log-MAP decoding, exactly as in binary turbo codes. That agrees with the results shown in [108].

5.6 Summary

In this Section the most important issues on improved algorithms for duo-binary turbo codes are summarised, with emphasis on the standardised DVB-RCS turbo code.

- The DVB-RCS standard enables terminal-to-hub communications in a return satellite link with transmission speeds up to 2 Mbps. It has adopted a very powerful and flexible 8-states duo-binary turbo code that supports twelve frame sizes and seven coding rates.
- The resulting DVB-RCS turbo code performance is from 1 to 1.5 dB away from the AWGN channel capacity limit, although relative small or medium frame sizes are used.

-
- The same performance gap exists between Max-Log-MAP and Log-MAP decoding in duo-binary turbo codes, such as in binary turbo codes. This is 0.3 dB performance degradation at BER of 10^{-4} assuming an 8-state duo-binary turbo code, e.g. in the form of DVB-RCS standard, with coding rate $R = 1/2$.
 - Several attempts in the past have shown either improvements to the Max-Log-MAP turbo decoder performance, e.g. scaling of the extrinsic information, or reduction to the decoding complexity of the Log-MAP turbo decoder, e.g. Constant Log-MAP algorithm.
 - There have been reported performance improvements to the standardised DVB-RCS turbo code as well, mainly by using better interleaver design.
 - A very recent work can be found in [3] where the performance of the DVB-RCS was investigated. The *improved* Max-Log-MAP iterative decoding was introduced and also the extension to a 16-states turbo encoder.
 - In the former case, the resulting decoding algorithm can approach the Log-MAP turbo decoder with up to 0.05 dB in performance degradation for a wide range of SNR values. However, in the latter case, coding gains of up to 1 dB are feasible, due to increased minimum distance, but with almost double decoding complexity.
 - Two novel decoding algorithm approaches to duo-binary turbo codes were presented and analytical complexity estimation as well as relative complexity comparison was given. As a practical example, the DVB-RCS turbo code was considered with symbol-based iterative decoding.
 - The first approach is based on \max/\max^* operation replacement to either the forward/backward recursion or soft-output computation of the Max-Log-MAP or Log-MAP algorithm, in a similar way to binary turbo codes. Four novel SISO decoding algorithms were proposed suitable for duo-binary turbo codes with a trade-off between performance/complexity.
 - For example, for low coding rates and medium BER/FER values, SISO-B improves the iterative Max-Log-MAP performance up to 0.13 dB, while SISO-C

degrades the iterative Log-MAP performance up to 0.08 dB. In the first case, the relative complexity increase is 36.84% more LUT operations and 15.38% extra additions. In the second case, the relative complexity decrease is 36.84% fewer LUT operations and 10.85% fewer additions.

- The second approach is based on the Constant Log-MAP algorithm for duo-binary turbo codes. The difference from an existing algorithm is that the simplified \max^* operator is processed over pair of values, instead over four values.
- For example, for low coding rates and high to medium BER/FER values, the proposed algorithm is 0.2 dB superior to the existing algorithm, while it performs close to Log-MAP decoding with a performance degradation less than 0.02 dB, similar to the binary case.
- It is concluded that the two proposed algorithm approaches provide good alternative solutions to Max-Log-MAP and Log-MAP algorithms for the DVB-RCS turbo code.

Chapter 6

Improved Decoding Algorithms for LDPC Codes

In this *fourth and last* Chapter of original work, another class of channel capacity approaching codes is investigated, i.e. LDPC codes. More specifically, two modifications are proposed in check-node update when decoding with the SPA in the logarithmic domain, i.e. LLR-SPA, to cope with the infinite value approximation problem. Various computer simulations are run for randomly constructed regular LDPC codes in the AWGN channel. In some particular cases, the proposed algorithms are found to be beneficial in terms of error floor reduction. Two approaches to reduce the decoding complexity in check-node update are also presented.

6.1 Introduction

As addressed in Section 2.6, LDPC codes were proposed by *Gallager* in the early 1960's [21] but they were forgotten for many years, due to the lack of technology advances at that time for their practical implementation. It was not until the mid 1990's that they were rediscovered by *MacKay* and *Neal* [22]. LDPC codes can achieve near *Shannon* limit performance over the binary erasure channel (BEC) and also over the AWGN channel [23, 25] in an iterative decoding process and at reasonable decoding complexity. That makes them strong competitors to turbo codes. As a practical application, LDPC

codes have been recently adopted by the second generation Digital Video Broadcasting by Satellite (DVB-S2) standard in 2004 [68].

In general, research work on LDPC codes can be based on two categories, either the encoding or the decoding part, e.g. see Sections 2.6.2 and 2.6.3 respectively. The first case includes fast encoding methods and smart parity-check matrix construction for regular/irregular LDPC codes with small number of short cycles [23, 25, 114, 115]. The second case includes reduced complexity iterative decoding algorithms based on the message passing principle, such as the SPA [116, 15, 16].

In this Chapter, we focus our attention to the decoding part, due to the high relevance and our previous experience of turbo codes. Inspired by the research work in [15], of which an updated version can be found quite recently in [16], we propose for the *first time* two modifications to the check-node update of the SPA operating with LLR values. These are necessary for approximating the infinite value of the hyperbolic tangent, i.e. $\tanh(x)$, and inverse (arc) hyperbolic tangent, i.e. $\tanh^{-1}(x)$, functions respectively. The proposed modifications reduce the error floor observed in the performance of LDPC codes with particular block sizes. A suitable explanation is given for this. Furthermore, we present *two novel* approximation methods to reduce the computational complexity of the check-node update. That is, use of piecewise linear function and quantization for the \tanh and inverse (arc) \tanh functions respectively.

6.2 Relevant Work on Optimum and Reduced Complexity Decoding Algorithms for LDPC Codes

Similar to turbo codes, logarithmic domain decoding algorithms have implementation advantages over the corresponding ones in the probability domain, as multiplications become additions and normalisations are eliminated [16]. Different check-node update rules using LLR values are described in [16]. To our knowledge, this reference provides all the relevant work on reduced complexity decoding algorithms for LDPC codes. In addition, [16] is a revision of some conference/journal papers published before by the same authors independently. Here, we adapt the same notation as in Section 2.6 and a

brief description from [16] is following.

Assume a binary LDPC code with block size (N, K) and sparse parity-check matrix H of size $M \times N$ where $M = N - K$. This code can be represented by a bipartite graph, e.g. *Tanner* graph, with M check-nodes in one class and N symbol or variable nodes in the other. Assume also a *regular* LDPC code, denoted by (d_s, d_c) , where every symbol node is connected to d_s check-nodes and every check-node is connected to d_c symbol nodes.

For two statistically independent binary random variables U and V , the *tanh rule* [65] is defined as

$$L(U \oplus V) \triangleq 2 \tanh^{-1} \left\{ \tanh \left(\frac{L(U)}{2} \right) \tanh \left(\frac{L(V)}{2} \right) \right\} \quad (6.1)$$

The *tanh rule* is also referred to as *box-plus* operation. Using LLR values, the check-node update can be computed from [15, 16]

$$\Lambda_{m \rightarrow n}(u_n) = 2 \tanh^{-1} \left\{ \prod_{n' \in N(m) \setminus n} \tanh [\lambda_{n' \rightarrow m}(u_{n'})/2] \right\} \quad (6.2)$$

where (λ) represents the LLR of the message that symbol node n sends to check-node m , indicating the probability of symbol u_n being zero or one, based on all checks involving n except m , i.e. $\lambda_{n \rightarrow m}(u_n) = \ln \{q_{n \rightarrow m}(0)/q_{n \rightarrow m}(1)\}$. Similarly, (Λ) represents the LLR of the message that the m th check-node sends to the n th symbol node, indicating the probability of symbol u_n being zero or one, based on all symbols checked by m except n , i.e. $\Lambda_{m \rightarrow n}(u_n) = \ln \{r_{m \rightarrow n}(0)/r_{m \rightarrow n}(1)\}$. Also, $N(m) \setminus n$ is the set of symbol nodes that participate in the m th parity-check equation, i.e. the position of ones in the m th row of the parity-check matrix H , excluding n . Note that Eq. (6.2) is valid for each m and for each $n \in N(m)$.

It is noted that both \tanh and \tanh^{-1} functions are monotonically increasing and have odd symmetry, i.e. $f(x) = -f(-x)$. Therefore, by taking account the sign and the magnitude of the incoming symbol node messages (λ) , Eq. (6.2) becomes

$$\Lambda_{m \rightarrow n}(u_n) = \left\{ \prod_{n' \in N(m) \setminus n} \text{sign}[\lambda_{n' \rightarrow m}(u_{n'})] \right\} \times 2 \tanh^{-1} \left\{ \prod_{n' \in N(m) \setminus n} \tanh[|\lambda_{n' \rightarrow m}(u_{n'})|/2] \right\} \quad (6.3)$$

The implementation of Eq. (6.3) requires $2d_c$ multiplications, $d_c \tanh$ function operations and d_c inverse (arc) \tanh function operations. Also, the signs can be obtained by getting the overall sign and then using the *XOR* operation with the individual sign in order to get the outgoing, i.e. extrinsic, signs.

In *Gallager's* approach [21], it can be shown that Eq. (6.3) is simplified to

$$\Lambda_{m \rightarrow n}(u_n) = \left\{ \prod_{n' \in N(m) \setminus n} \text{sign}[\lambda_{n' \rightarrow m}(u_{n'})] \right\} \times \phi \left\{ \sum_{n' \in N(m) \setminus n} \phi[|\lambda_{n' \rightarrow m}(u_{n'})|] \right\} \quad (6.4)$$

where

$$\phi(x) = \ln \left(\frac{e^x + 1}{e^x - 1} \right), \quad x > 0 \quad (6.5)$$

with the property $\phi\{\phi(x)\} = x$, thus $\phi(x) = \phi^{-1}(x)$. This has the advantage that only one function is needed to be computed, i.e. $\phi(x)$, and stored in memory, e.g. using a LUT of values. Also, the *sum* instead of the *product* of values is used, in case of the magnitude of the incoming symbol node messages (λ). That makes easier a hardware decoder implementation.

The implementation of Eq. (6.4) requires $2d_c$ additions and $2d_c$ operations of the ϕ function. Therefore, the computational complexity of the check-node update based on both the *tanh rule* and *Gallager's* approach is approximately the same.

Alternatively, the *tanh rule* from Eq. (6.1) can be represented as [65]

$$L(U \oplus V) = \ln \left(\frac{1 + e^{L(U)+L(V)}}{e^{L(U)} + e^{L(V)}} \right) \quad (6.6)$$

Using the *Jacobian* logarithm [60] twice, the above equation becomes

$$L(U \oplus V) = \text{sign}(L(U)) \text{sign}(L(V)) \min(|L(U)|, |L(V)|) + \ln \left(1 + e^{-|L(U)+L(V)|} \right) - \ln \left(1 + e^{-|L(U)-L(V)|} \right) \quad (6.7)$$

Define now two sets of auxiliary binary random variables, as $f_1 = u_{n_1}$, $f_2 = f_1 \oplus u_{n_2}$, $f_3 = f_2 \oplus u_{n_3}, \dots, f_{d_c} = f_{d_c-1} \oplus u_{n_{d_c}}$ and $b_{d_c} = u_{n_{d_c}}$, $b_{d_c-1} = b_{d_c} \oplus u_{n_{d_c-1}}, \dots, b_1 = b_2 \oplus u_{n_1}$. Then, using Eq. (6.7) we can obtain recursively the corresponding LLR values $L(f_1), L(f_2), \dots, L(f_{d_c})$ and $L(b_1), L(b_2), \dots, L(b_{d_c})$ from the incoming messages $\lambda_{n_1 \rightarrow m}(u_{n_1}), \lambda_{n_2 \rightarrow m}(u_{n_2}), \dots, \lambda_{n_{d_c} \rightarrow m}(u_{n_{d_c}})$, which represent already LLR values.

Using the property $u_{n_1} \oplus u_{n_2} \oplus \dots \oplus u_{n_{d_c}} = 0$, we obtain $u_{n_i} = f_{i-1} \oplus b_{i+1}$, for $i \in \{2, 3, \dots, d_c - 1\}$. Thus, the corresponding check-node update for each check-node m becomes [15]

$$\Lambda_{m \rightarrow n_i}(u_{n_i}) = \begin{cases} L(b_2), & i = 1 \\ L(f_{i-1} \oplus b_{i+1}), & i = 2, 3, \dots, d_c - 1 \\ L(f_{d_c-1}), & i = d_c \end{cases} \quad (6.8)$$

The implementation of Eq. (6.8) requires $3(d_c - 2)$ computations of the core operation $L(U \oplus V)$ as from Eq. (6.7). Moreover, the correction function $g(x) = \ln(1 + e^{-|x|})$ in Eq. (6.7) needs to be computed twice and is implemented using a LUT of eight values or a piecewise linear function with six regions. In an other way, the correction function in Eq. (6.7) can be expressed as $g(x, y) = \ln \left\{ (1 + e^{-|x+y|}) / (1 + e^{-|x-y|}) \right\}$ and can be implemented by a single *constant* that takes three possible values, i.e. $\pm c$ and zero. This is analogous to the Constant Log-MAP approximation in the case of turbo decoding.

The procedure described above can be implemented in a *serial* mode, and is exactly the forward-backward algorithm applied to a trellis with a single node [15]. Another approach was described in the same Reference, for applications with high throughput requirements. This is *tree* topology in check-node update, suitable for *parallel* implementation. In this case, the required computations of the core operation $L(U \oplus V)$ are

reduced to $(d_c - 1)$, but d_c extra computations are needed to produce the outgoing, i.e. extrinsic, messages (Λ) simultaneously from check-node m to all the symbol nodes u_{n_i} .

Computer simulation results in [15] indicate that all the reduced complexity variants of the SPA using the *Jacobian* logarithm, e.g. see Eq. (6.7), perform very close to the conventional SPA. In particular, the performance degradation using the piecewise linear approximation for the core operation with either trellis or tree topology is only 0.05 dB at BER of 10^{-5} . The rest of the parameters are two randomly constructed LDPC codes, i.e. (1008, 504) and (6000, 3000) with $(d_s, d_c) = (3, 6)$, and maximum 80 decoding iterations in the AWGN channel. Furthermore, the constant correction term with three values approximation achieves close to SPA performance at high SNR values. Some more reduced complexity algorithms in check-node update from [16] are described below.

It can be proved that $|L(U \oplus V)| \leq \phi \{ \phi(\min(|L(U)|, |L(V)|)) \} = \min(|L(U)|, |L(V)|)$, as the ϕ function from Eq. (6.5) is monotonically decreasing [16]. Hence, the following approximation holds

$$|L(U \oplus V)| \approx \min(|L(U)|, |L(V)|) \quad (6.9)$$

Thus, the corresponding check-node update from Eq. (6.4) can be approximated as

$$\tilde{\Lambda}_{m \rightarrow n}(u_n) = \left\{ \prod_{n' \in N(m) \setminus n} \text{sign}[\lambda_{n' \rightarrow m}(u_{n'})] \right\} \times \min_{n' \in N(m) \setminus n} |\lambda_{n' \rightarrow m}(u_{n'})| \quad (6.10)$$

This approach is also known as *min-sum* algorithm. It is noted that the same algorithm can be obtained from the *Jacobian* logarithm approach as in Eq. (6.7), if the correction function is omitted. This is analogous to the Max-Log-MAP approximation in case of turbo decoding. In practice, the signs of all the incoming messages are needed to be known and only two of all incoming messages that have the smallest magnitude are needed to be stored. That reduces significantly the computational complexity compared to the SPA algorithm, e.g. see [16].

The performance degradation of the min-sum algorithm against the SPA assuming two randomly constructed LDPC codes, i.e. (1008, 504) and (8000, 4000) with $(d_s, d_c) = (3, 6)$, is 0.3 dB and 0.5 dB respectively at BER of 10^{-4} . This is for the case of maximum 100 decoding iterations in the AWGN channel.

It is noted that the magnitude of $(\tilde{\Lambda})$ in Eq. (6.10) is always greater than that of (Λ) in Eq. (6.4). That necessitates a search for *further improvements* to the updating process of the min-sum algorithm, so that more accurate soft values are produced. One straightforward application is to use a *normalisation* factor α that is greater than one [16], so that Eq. (6.10) becomes

$$\tilde{\Lambda}_{m \rightarrow n}(u_n) = \left\{ \prod_{n' \in N(m) \setminus n} \text{sign}[\lambda_{n' \rightarrow m}(u_{n'})] \right\} \times \min_{n' \in N(m) \setminus n} |\lambda_{n' \rightarrow m}(u_{n'})| \times 1/\alpha \quad (6.11)$$

This approach is also known as *normalised* min-sum algorithm. The value of α can be determined by *density evolution* and is kept constant and independent of the SNR value [16]. Simulation results have shown that the normalised min-sum algorithm can approach the SPA performance with degradation of 0.05 dB at BER of 10^{-4} , assuming two randomly constructed LDPC codes, i.e. (1008, 504) and (8000, 4000) with $(d_s, d_c) = (3, 6)$, and maximum 100 decoding iterations in the AWGN channel. Moreover, for the (1008, 504) LDPC code the normalised min-sum algorithm can even have a slightly better performance than the SPA at high SNR values. This can be explained because of the presence of short cycles that make the message-passing algorithm to be sub-optimum, as in this case it operates with *correlated* instead of *uncorrelated* values.

One variation of the min-sum algorithm with correction using single *constant*, which was already described above, is the *offset* min-sum algorithm [16]. The resulting check-node update is described as

$$\tilde{\Lambda}_{m \rightarrow n}(u_n) = \left\{ \prod_{n' \in N(m) \setminus n} \text{sign}[\lambda_{n' \rightarrow m}(u_{n'})] \right\} \times \max \left\{ \min_{n' \in N(m) \setminus n} |\lambda_{n' \rightarrow m}(u_{n'})| - \beta, 0 \right\} \quad (6.12)$$

where β is a positive constant. In this case, the incoming messages (λ) with magnitude less than β are eliminated from the next check-node update step. Similarly, the value of β can be determined by *density evolution* and is kept constant and independent of the SNR value [16]. In terms of BER performance, the offset min-sum algorithm with optimised β values is almost identical to the normalised min-sum algorithm.

In case of a hardware decoder implementation with finite precision, *quantization* effects of the offset min-sum algorithm, which is more convenient than the normalised min-sum algorithm, have been reported in [16]. The resulting BER performance with 4 quantization bits is slightly better than the unquantized SPA at high SNR values, assuming a randomly constructed LDPC code, i.e. (1008, 504) with $(d_s, d_c) = (3, 6)$, and maximum 50 decoding iterations in the AWGN channel. In another case, the resulting BER performance with 6 quantization bits is 0.1 dB inferior to the unquantized SPA, assuming another randomly constructed LDPC code, i.e. (8000, 4000) with $(d_s, d_c) = (3, 6)$, and maximum 50 decoding iterations in the AWGN channel. In contrast, the SPA using *Gallager's* approach with finite precision is subject to error floor at BER below 10^{-6} , even with 7 quantization bits, assuming a randomly constructed LDPC code, i.e. (1008, 504) with $(d_s, d_c) = (3, 6)$, and maximum 50 decoding iterations in the AWGN channel.

The use of both *clipping* and quantization to the LLR values in the *min-sum* algorithm and the two correction methods, namely conditional and unconditional correction, can be found in [117]. In both cases, improvements to the BER performance over the unquantized min-sum algorithm are feasible. Moreover, the two modified min-sum algorithms, which are similar to the offset min-sum algorithm from [16], can approach or even perform better than the SPA in the high SNR region and for medium block sizes, similar to [16].

Reduction in decoding complexity can be also applied in *symbol node* update, e.g. see Section 2.6.3, and reduce the memory storage requirements. In [16] two methods are described for computing the outgoing messages (λ). The first method is based on the summation of all incoming messages (Λ) plus channel LLR values and then subtracting the incoming messages individually to find the extrinsic terms. The second method,

which is less complex, is based on passing *a posteriori* instead of the extrinsic LLR values from symbol nodes to check-nodes.

We have noticed that there exists no related work based on the check-node update from Eqs. (6.2) and (6.3). The reason for that is the increased required amount of computational complexity. However, in Sections 6.3.3 and 6.4.3 two reduced complexity algorithms are presented. That makes our research work contribution more interesting. It is noted that similar reduced complexity algorithms can be applied to the check-node update as from Eq. (6.4), by approximating the ϕ function from Eq. (6.5). This was also suggested in [67]. For reasons of completeness, our computer simulations results are compared to the corresponding ones with check-node update as from Eq. (6.4).

6.3 Modified *tanh* Function in Sum-Product Algorithm for Decoding LDPC Codes

Here, an appropriate modification is proposed to cope with the infinite argument approximation problem of the *tanh* function. Two methods of computational complexity reduction in check-node update are also given [118]. Computer simulation results are presented for regular LDPC codes with various block sizes in the AWGN channel.

6.3.1 Motivation

The hyperbolic tangent function, i.e. $f(x) = \tanh(x)$ is expressed as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (6.13)$$

where $-\infty < x < +\infty$ and $-1 \leq \tanh(x) \leq +1$. A plot of this function can be found in Fig. 6.2 in Section 6.3.3. It is noted that when $x \rightarrow +\infty$, then $\tanh(x) = +1$. This has an impact to the inverse (arc) *tanh* function, as expressed in Eq. (6.16) in Section 6.4.1, because when $x \rightarrow +1$, then $\tanh^{-1}(x) \rightarrow +\infty$. A similar situation occurs for the $\tanh^{-1}(x)$ function when $x \rightarrow -\infty$.

This phenomenon may happen when decoding LDPC codes, because in the high SNR region the channel reliability values are larger than the corresponding extrinsic values that are produced in the first few decoding iterations. Thus, the incoming symbol node messages (λ) having already an increased value, make the check-node update computation from Eq. (6.2) to approach infinity, so an overflow of the decoder may be produced.

6.3.2 Proposed Method

From preliminary computer simulation results, it was found that the SPA with check-node update as from Eq. (6.2), suffers from an error floor at low BER values and for particular block sizes. The reason for that, as explained in the previous Section, is because the argument of the \tanh function is approaching infinity, which makes the inverse (arc) \tanh function output to approach infinity as well.

An appropriate *modification* is thus needed to the \tanh function to perform a kind of decoding normalisation. This is done by

$$\tanh_{\text{modified}}(x) = \begin{cases} \tanh(x), & \text{if } |x| < x_0 \\ \text{sign}(x) \tanh(x_0), & \text{if } |x| \geq x_0 \end{cases} \quad (6.14)$$

and guarantees that the values passed to the inverse (arc) \tanh function are always in the region of $-1 < \tanh(x)_{\text{modified}} < 1$, instead of $-1 \leq \tanh(x) \leq 1$. The value of x_0 is relatively small and positive, e.g. $x_0 \leq 10$. We note that the infinite argument of the \tanh function is approximated by a smaller value (i.e. x_0) using this modification. This technique is also known as *clipping*. The decoding complexity that is added is the use of clipping d_c times to the \tanh function in check-node update, where d_c is the number of symbol nodes that every check-node is connected to.

The best x_0 value can be found simply by computer simulation tests run for different values, i.e. by trial and error. For that reason, a randomly constructed regular (1008, 504) LDPC code with $(d_s, d_c) = (3, 6)$ and coding rate $R = 1/2$ was considered [13]. Also, BPSK modulation, the AWGN channel and maximum 80 decoding iterations were assumed.

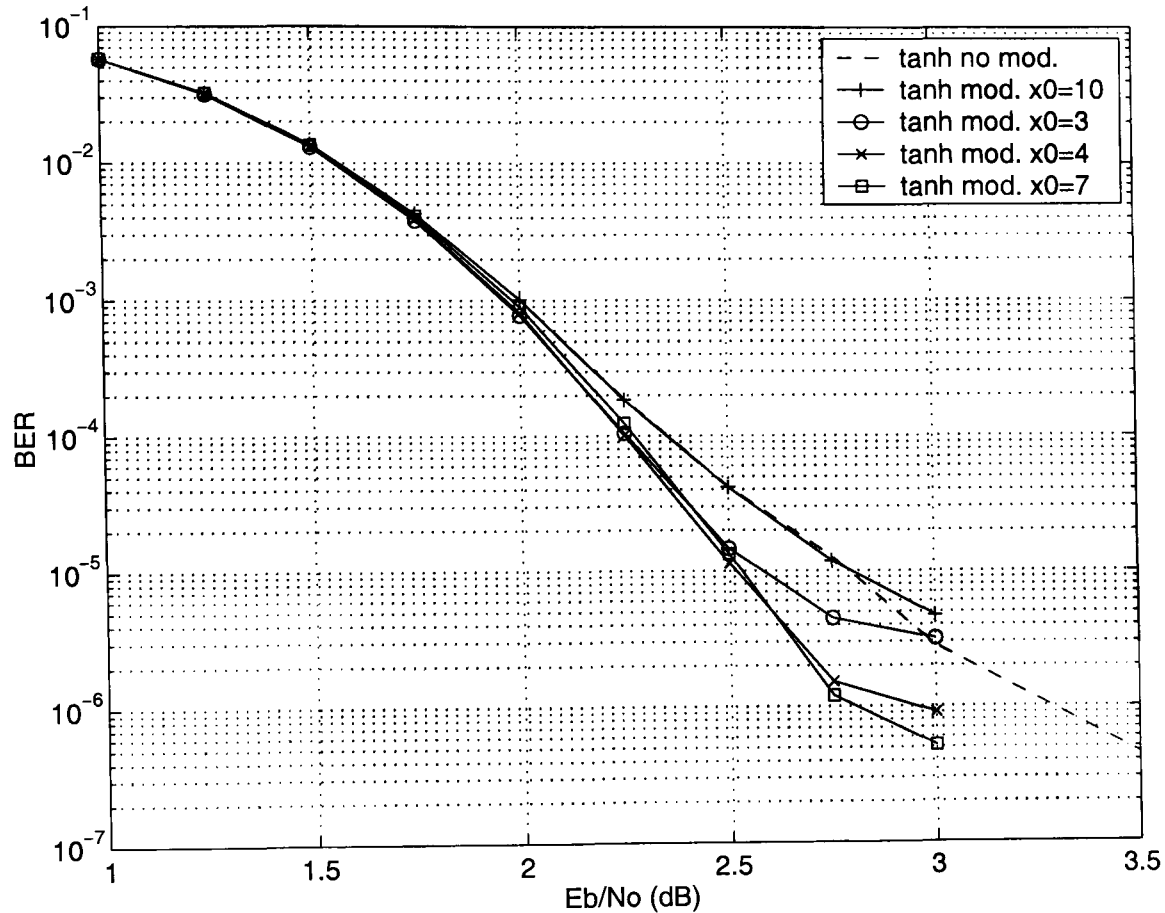


Figure 6.1: The effect to the BER performance when approximating the \tanh function with different values. BER with no approximation is shown in dashed lines. (1008, 504) LDPC code, coding rate $R = 1/2$, AWGN channel and maximum 80 decoding iterations.

In Fig. 6.1 it is shown the impact of the modified \tanh function to the BER performance. More computer simulation results and discussion can be found in Section 6.3.4. It can be observed that in the high SNR region, i.e. at BER lower than 10^{-5} , when $x_0 = 3$ or 10 the BER performance degradation is large, even compared to no modification at all. When $x_0 = 4$ (and also when $x_0 = 5$ and 6, which is not shown here) the BER performance is improved, while when $x_0 = 7$ (and also when $x_0 = 8$ and 9, which is not shown here) the best BER performance is achieved. On the other hand, in the low to medium SNR region, i.e. at BER greater than 10^{-5} , any of the considered x_0 values, except for 10, is acceptable. In the rest of computer simulations, we choose $x_0 = 7$. This results to $\tanh(x_0) = 0.999998$.

6.3.3 Decoding Complexity Reductions

From the implementation point of view, the \tanh function is rather complex, as it involves operations such as additions, exponentials and division. Thus, two approximation methods are proposed to reduce the complexity and ease a hardware decoding implementation. That is, *piecewise linear function* and *quantization*.

In piecewise linear function approximation, seven regions are used as in Table 6.1. This is done in order to have a similar degree of discrimination with the LUT used in Log-MAP turbo decoder implementations, i.e. three-bits or eight values size of LUT. The root mean square error (RMSE) is approximately 0.02. Quantization table with eight values (i.e. three-bits size) is shown in Table 6.2. In this case, the RMSE is approximately 0.07. Both the approximations, as well as the continuous \tanh function, are plotted in Fig. 6.2.

From Fig. 6.1 in the previous Section, it can be pointed out that there is not much difference in the resulting BER performance in the low to medium SNR region, i.e. at BER greater than 10^{-5} , when $x_0 = 3$ or 7. Thus, the regions of the piecewise linear function can be reduced from seven to five. In addition, the size of the quantization table can be reduced from eight to six values. As a consequence, further complexity reduction to the two approximation methods is feasible by observing the operating SNR value.

Table 6.1: Piecewise linear approximation of $\tanh(x)$ function.

x	$\tanh(x)$
$(-7.0, -3.0]$	$0.0012 * x - 0.9914$
$(-3.0, -1.6]$	$0.0524 * x - 0.8378$
$(-1.6, -0.8]$	$0.322 * x - 0.4064$
$(-0.8, 0.8]$	$0.83 * x$
$(0.8, 1.6]$	$0.322 * x + 0.4064$
$(1.6, 3.0]$	$0.0524 * x + 0.8378$
$(3.0, 7.0]$	$0.0012 * x + 0.9914$

Table 6.2: Quantization table of $\tanh(x)$ function.

x	$\tanh(x)$
$(-7.0, -3.0]$	-0.99991
$(-3.0, -1.6]$	-0.9801
$(-1.6, -0.8]$	-0.8337
$(-0.8, 0.0]$	-0.3799
$(0.0, 0.8]$	0.3799
$(0.8, 1.6]$	0.8337
$(1.6, 3.0]$	0.9801
$(3.0, 7.0]$	0.99991

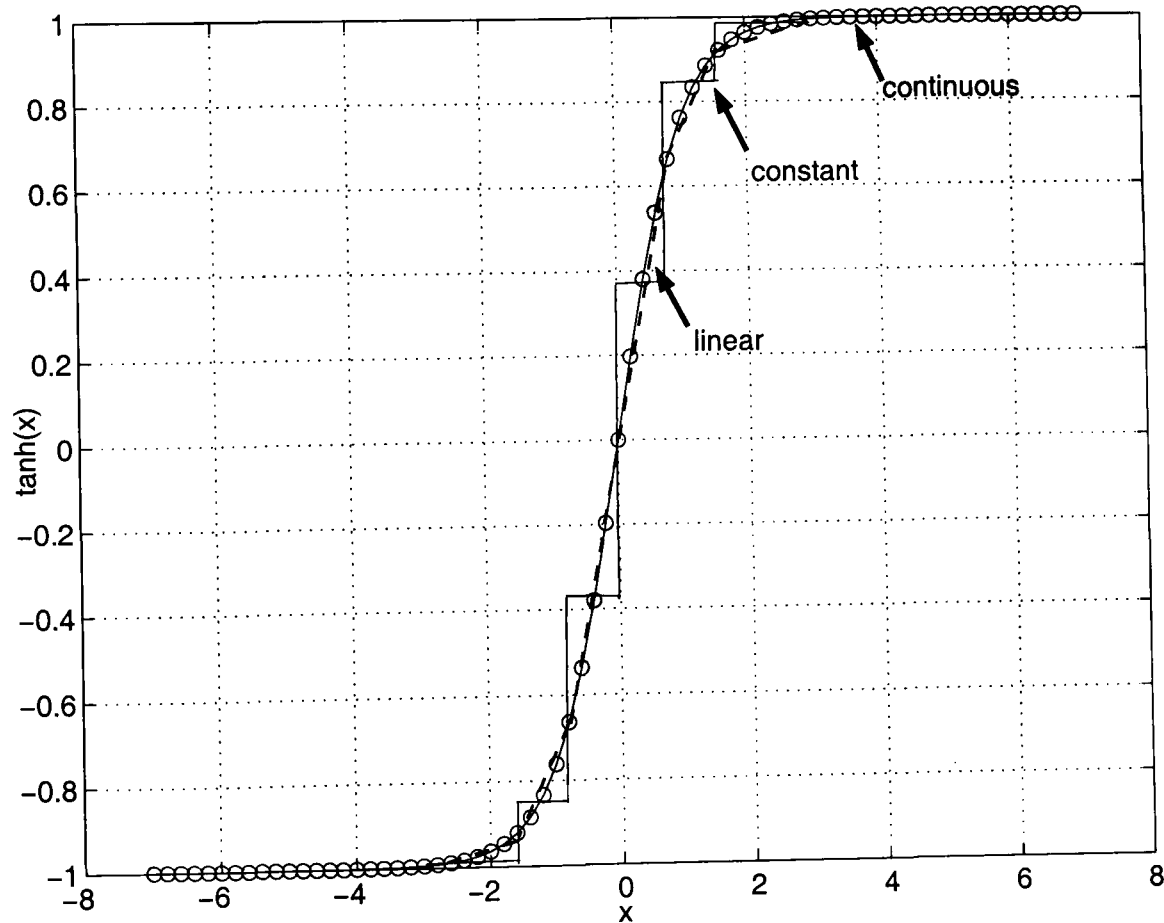


Figure 6.2: Example of \tanh function (continuous-circle line) and approximations with piecewise linear function (dashed line) and quantization (constant function-solid line).

As reported in Section 6.2, the \tanh function is monotonically increasing and has odd

symmetry, i.e. $f(x) = -f(-x)$. Therefore, when using the check-node update from Eq. (6.3), only absolute values are needed. That reduces the required regions of the piecewise linear function from seven to four and also the quantization table from eight to four values. This is assuming $x_0 = 7$. However, as Eq. (6.3) indicates, the *sign* of the incoming symbol node messages needs to be known, which adds some extra computational complexity.

Finally, in computer simulation results to be presented in the next Section, both piecewise linear function and quantization approximations are applied to the inverse (arc) \tanh function as well. The exact approximation values used are shown in Section 6.4.3. The reason for that is because there was no significant impact on the resulting BER performance in either the continuous form or with two approximations to the inverse (arc) \tanh function when already using the two approximations to the \tanh function. Thus, the check-node update from Eq. (6.2) avoids the exact computation of the inverse (arc) \tanh function and the overall computational complexity can be further reduced.

6.3.4 Computer Simulation Results

Computer simulations are run for randomly constructed rate half regular LDPC codes with $(d_s, d_c) = (3, 6)$ and different block sizes [13]. That is, assumption of either *short* block size, e.g. (96, 48), *medium* block size, e.g. (504, 252) and (1008, 504), or *large* block size, e.g. (8000, 4000). BPSK modulation and the AWGN channel are assumed and also two cases of maximum number of decoding iterations. In the *first case* a relatively small number is considered, e.g. 10. In the *second case*, a greater number is considered, e.g. either 50, 80 or 200. This is to show the independent BER performance behaviour of the decoding algorithms and to compare to [116, 15, 16].

The corresponding BER performance of different LDPC codes with or without modification to the \tanh function in the two cases of maximum number of decoding iterations is shown in Figs. 6.3 and 6.4 respectively. The value of $x_0 = 7$ is assumed, e.g. see Section 6.3.2. For comparison, BER results obtained with *Gallager's* approach (shown in dashed lines) from Eq. (6.4) are reported. In this case, upper and low limits, i.e. clipping, had to be applied to the ϕ function from Eq. (6.5), as following

$$\phi_{upper} = \phi(9 \times 10^{-5}) = 10 \text{ and } \phi_{lower} = \phi(10) = 9 \times 10^{-5}.$$

In Figs. 6.5 and 6.6 it is shown the BER performance of the same block size LDPC codes with modified \tanh function (shown in dashed lines) and also using two approximations on top of it to reduce the computational complexity, i.e. seven regions piecewise linear function and eight values of quantization. The first case of maximum number of decoding iterations is shown in Fig. 6.5 and the second case in Fig. 6.6 respectively.

Discussion

From Figs. 6.3 and 6.4 it is noticed that the proposed *modification* to the \tanh function provides a performance improvement from 0.5 to 1 dB in the high SNR region, thus reducing the code error floor. This depends on the maximum number of decoding iterations and is observed for codes with relatively short or medium block size. As in Section 6.2, this improvement can be explained because of the presence of short cycles of the code that makes the message-passing algorithm to operate with *correlated* instead of *uncorrelated* values. In the case of a large block size, there is no significant improvement when considering the proposed modification. Surprisingly, assuming the short code, i.e. (96, 48), and high number of decoding iterations, i.e. 200, there is not much difference in terms of BER performance from either modification to the \tanh function. This can also be explained as above, because of the behaviour of the message-passing algorithm in such a block size.

From the same Figures is noticed that the BER performance with either the modified \tanh function or *Gallager's* approach is approximately the same. In the high SNR region, there is a small performance degradation of *Gallager's* approach, due to the fact that the upper and lower limit values of ϕ function were not optimised in computer simulations.

In addition, computer simulation results reported in [116, 15, 16] are in agreement with the presented ones using *modification* to the \tanh function. This is the case of the (504, 252) LDPC code with maximum 50 decoding iterations [116], the (1008, 504)

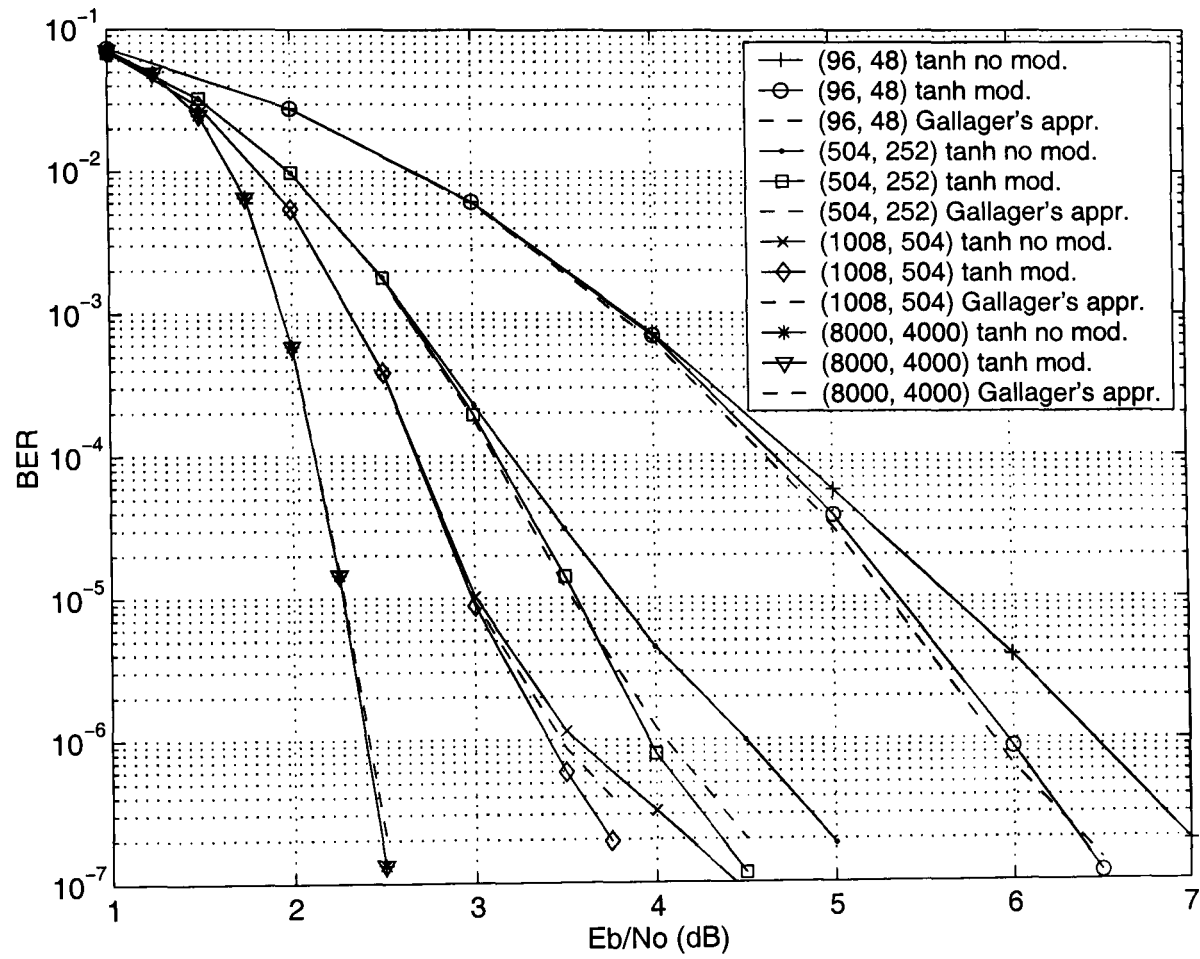


Figure 6.3: BER performance with or without modification to the \tanh function and comparison to *Gallager's* approach (dashed lines). Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.

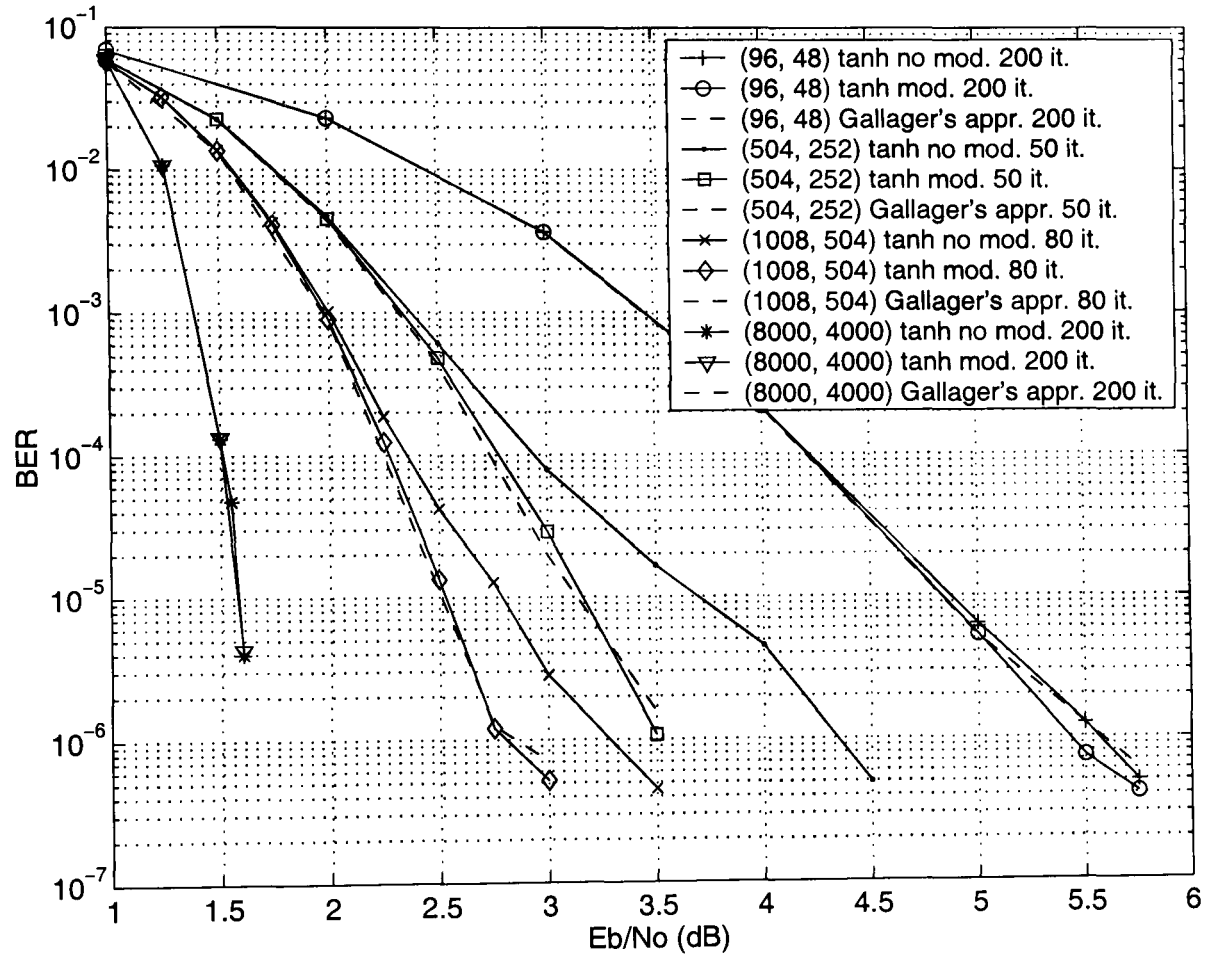


Figure 6.4: BER performance with or without modification to the \tanh function and comparison to *Gallager's* approach (dashed lines). Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel.

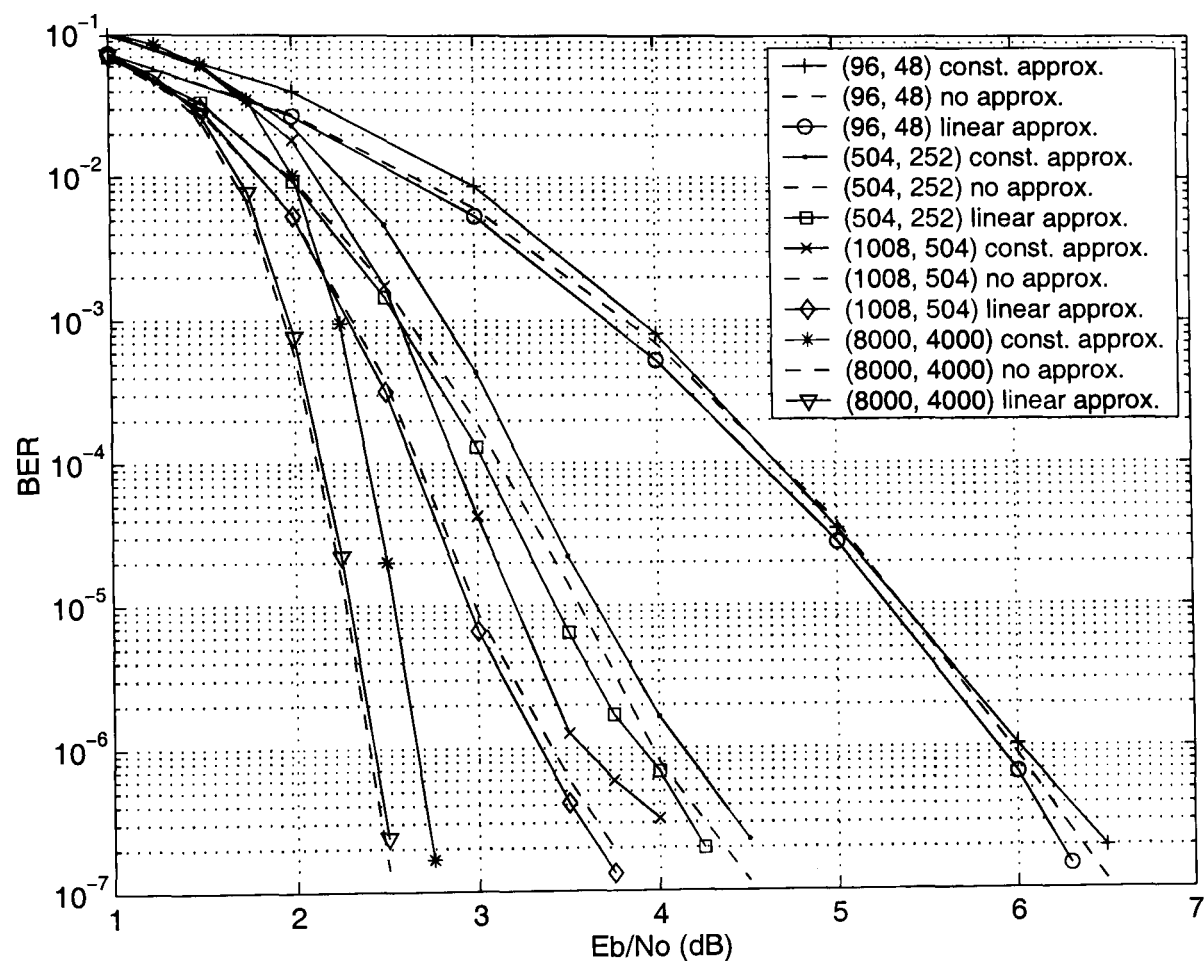


Figure 6.5: BER performance with modified \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.

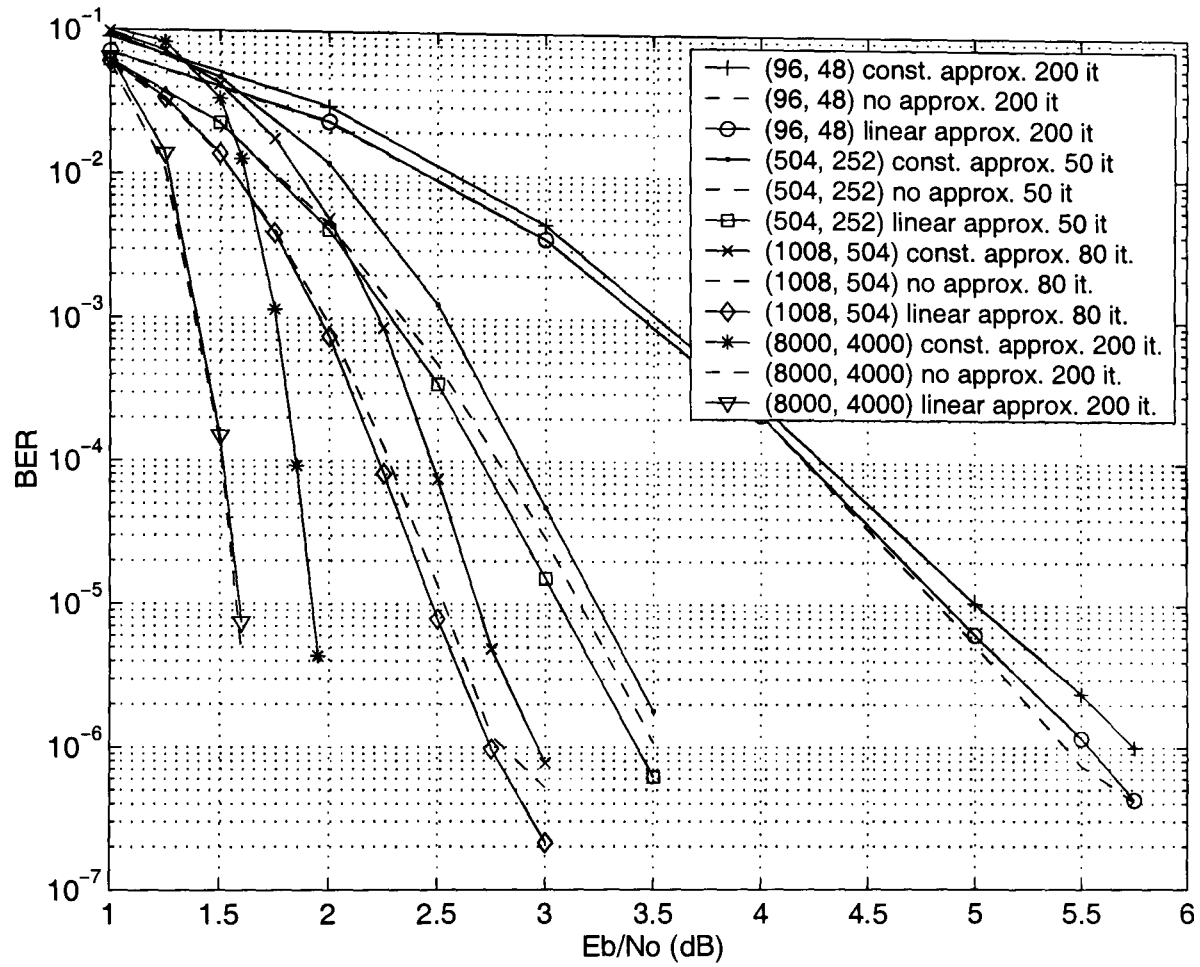


Figure 6.6: BER performance with modified \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel.

LDPC code with maximum 80 decoding iterations [15] and the (8000, 4000) LDPC code with maximum 100 decoding iterations [16] respectively.

From Figs. 6.5 and 6.6 it is noticed that the *piecewise linear approximation* provides a small improvement, e.g. 0.13 dB, to the BER performance compared to the continuous \tanh function using modification in both cases, i.e. $x_0 = 7$. This depends on the maximum number of decoding iterations and does not seem to happen for the large block size code and also for the short code with high number of decoding iterations. The same explanation on the behaviour of the message-passing algorithm for short to medium block sizes can be provided as above. A similar phenomenon was observed when considering the normalised/offset min-sum algorithm [16] and also the min-sum algorithm with both clipping and quantization [117] in the high SNR region. In both cases, these algorithms were performing slightly better than the SPA for LDPC codes with medium block size.

On the other hand, the application of *quantization* to the \tanh function provides approximately 0.25 dB degradation in terms of BER performance compared to the continuous \tanh function using modification in both cases, i.e. $x_0 = 7$. This is the case for codes with medium block sizes. Similarly, this depends on the maximum number of decoding iterations and the performance degradation is increased with increasing the block size. In the high SNR region a further optimisation of the quantization table is needed, especially for codes with medium block size.

It is also noted that in the high SNR region, the BER performance degradation of the *min-sum* algorithm with respect to the sum-product algorithm is 0.3 dB, assuming the (1008, 504) LDPC code with maximum 80 decoding iterations [15] and 0.5 dB, assuming the (8000, 4000) LDPC code with maximum 100 decoding iterations [16]. Therefore, the quantization method applied to the sum-product algorithm provides performance benefits compared to the min-sum algorithm, but with relative increase to the decoding complexity.

6.4 Modified *inverse (arc) tanh* Function in Sum-Product Algorithm for Decoding LDPC Codes

Another modification is proposed here to cope with the infinite output approximation problem of the *inverse (arc) tanh* function. Two methods of computational complexity reduction in check-node update are also given [119]. Computer simulation results are presented for regular LDPC codes with various block sizes in the AWGN channel.

6.4.1 Motivation

In order to get the expression of the inverse (arc) hyperbolic tangent function, i.e. $f(x) = \tanh^{-1}(x)$, it is known that

$$f(x) = y \Rightarrow x = f^{-1}(y) = f^{-1} \{ \tanh(y) \} \quad (6.15)$$

Hence, using Eq. (6.13) we have

$$\tanh^{-1}(x) = \frac{1}{2} \ln \left(\frac{1+x}{1-x} \right) \quad (6.16)$$

where $-1 < x < +1$ and $-\infty < \tanh^{-1}(x) < +\infty$. A plot of this function can be found in Fig. 6.8 in Section 6.4.3. It is noted that when $x \rightarrow +1$, then $\tanh^{-1}(x) \rightarrow +\infty$. As explained in Section 6.3.1, this may produce an overflow when decoding LDPC codes in the high SNR region. The reason for that is the increased value of the incoming symbol node messages (λ) that results to $\tanh(x) \rightarrow +1$ and makes the corresponding check-node update computation from Eq. (6.2) to approach infinity. A similar situation occurs when $x \rightarrow -1$.

6.4.2 Proposed Method

As reported in Section 6.3.2, preliminary computer simulation results have shown that the SPA with check-node update as from Eq. (6.2), suffers from an error floor at low

BER values and for particular block sizes. The reason for that, as explained in the previous Section, is because the argument of the inverse (arc) \tanh function is approaching the ± 1 value, which makes the output to approach infinity.

An appropriate *modification* is thus needed to the inverse (arc) \tanh function to perform a kind of decoding normalisation. This is done by

$$\tanh_{\text{modified}}^{-1}(x) = \begin{cases} \tanh^{-1}(x), & \text{if } |x| < x_0 \\ \text{sign}(x) \tanh^{-1}(x_0), & \text{if } |x| \geq x_0 \end{cases} \quad (6.17)$$

and guarantees that the output value of the inverse (arc) \tanh function does not approach infinity, i.e. $-\infty << \tanh_{\text{modified}}^{-1}(x) << +\infty$. The difference from the method proposed in Section 6.4.2 is that no modification is required to the \tanh function. The value of x_0 is less than one and positive, i.e. $0 < x_0 < 1$. We note that the infinite value of the inverse (arc) \tanh function output is approximated by a smaller value (i.e. $\pm \tanh^{-1}(x_0)$) using this modification. This technique is also known as *clipping*. The decoding complexity that is added is the use of clipping d_c times to the inverse (arc) \tanh function in check-node update, where d_c is the number of symbol nodes that every check-node is connected to.

The best x_0 value can be found simply by computer simulation tests run for different values, i.e. by trial and error. For that reason, the same randomly constructed regular (1008, 504) LDPC code was considered, as in Section 6.3.2.

In Fig. 6.7 it is shown the impact of the modified inverse (arc) \tanh function to the BER performance. More computer simulation results and discussion can be found in Section 6.4.4. It can be observed that in the high SNR region, i.e. at BER lower than 10^{-5} , when $\tanh^{-1}(x_0) = 100$ (and also when $\tanh^{-1}(x_0) > 100$, which is not shown here) the BER performance is identical to the corresponding performance with no modification at all. When $\tanh^{-1}(x_0) = 5$ (and also when $\tanh^{-1}(x_0) = 3$ and 4, which is not shown here) the BER performance is improved, while when $\tanh^{-1}(x_0) = 7$ (and also when $\tanh^{-1}(x_0) = 6, 8, 9$ and 10, which is not shown here) a slightly better BER performance is achieved. On the other hand, in the low to medium SNR region, i.e. at BER greater than 10^{-5} , any value between $3 \leq \tanh^{-1}(x_0) \leq 10$ is acceptable.

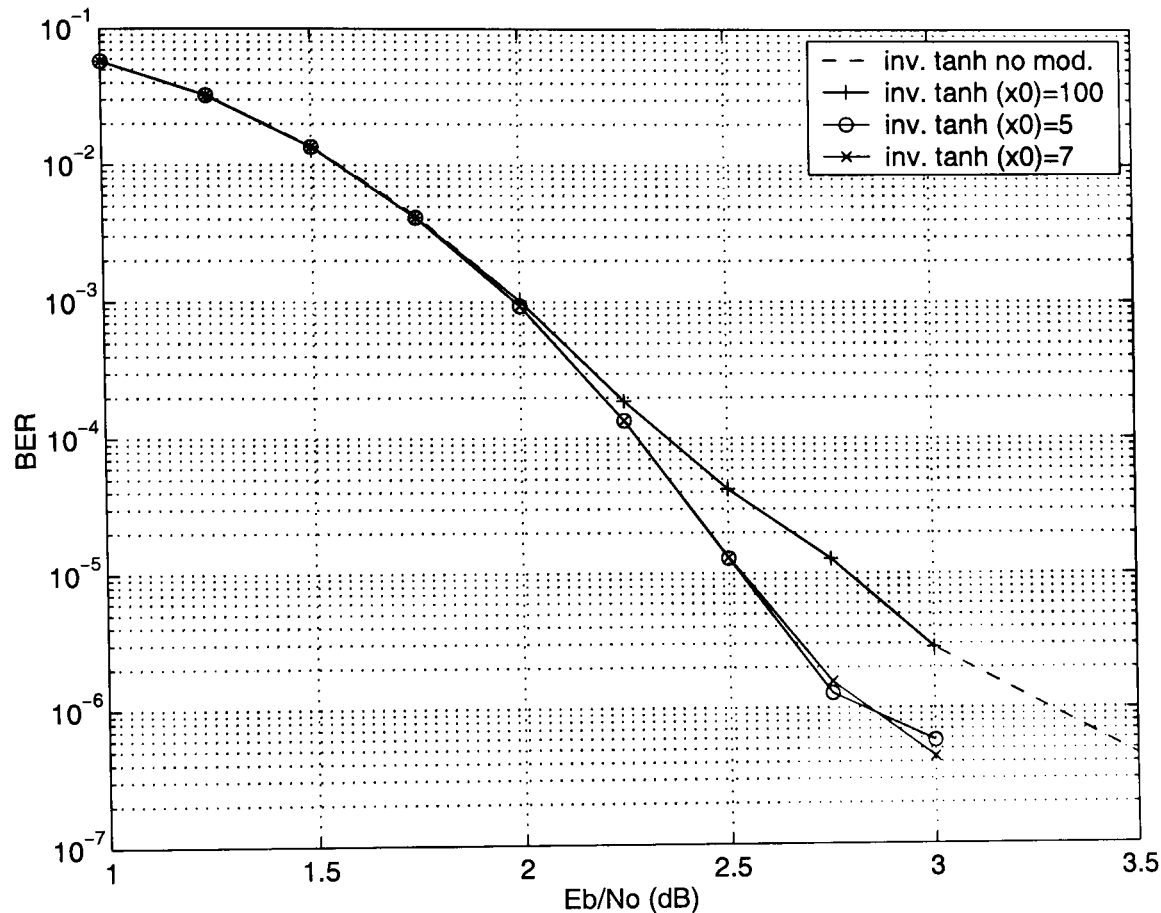


Figure 6.7: The effect to the BER performance when approximating the inverse \tanh function with different values. BER with no approximation is shown in dashed lines. (1008, 504) LDPC code, coding rate $R = 1/2$, AWGN channel and maximum 80 decoding iterations.

In the rest of computer simulations, we choose $\tanh^{-1}(x_0) = 7$. This corresponds to $x_0 = 0.999998$ and is done for reasons of symmetry with Section 6.3.2 where $x_0 = 7$ was chosen for the \tanh function.

6.4.3 Decoding Complexity Reductions

From the implementation point of view, the inverse (arc) \tanh function is rather complex, as it involves operations such as additions, logarithm and division. Thus, two approximation methods are proposed to reduce the complexity and ease a hardware decoding implementation. That is, *piecewise linear function* and *quantization* and is done in a similar way to the approximation methods of the \tanh function in Section

Table 6.3: Piecewise linear approximation of $\tanh^{-1}(x)$ function.

x	$\tanh^{-1}(x)$
$(-0.999998, -0.9951]$	$(x + 0.9914)/0.0012$
$(-0.9951, -0.9217]$	$(x + 0.8378)/0.0524$
$(-0.9217, -0.6640]$	$(x + 0.4064)/0.322$
$(-0.6640, 0.6640]$	$x/0.83$
$(0.6640, 0.9217]$	$(x - 0.4064)/0.322$
$(0.9217, 0.9951]$	$(x - 0.8378)/0.0524$
$(0.9951, 0.999998]$	$(x - 0.9914)/0.0012$

Table 6.4: Quantization table of $\tanh^{-1}(x)$ function.

x	$\tanh^{-1}(x)$
$(-0.999998, -0.9951]$	-3.3516
$(-0.9951, -0.9217]$	-1.9259
$(-0.9217, -0.6640]$	-1.0791
$(-0.6640, 0.0]$	-0.3451
$(0.0, 0.6640]$	0.3451
$(0.6640, 0.9217]$	1.0791
$(0.9217, 0.9951]$	1.9259
$(0.9951, 0.999998]$	3.3516

6.3.3.

We note that there is one-to-one correspondence between the \tanh and the inverse (arc) \tanh functions. Therefore, the two approximation methods of the inverse (arc) \tanh functions can be obtained by the corresponding Tables 6.1 and 6.2 using Eq. (6.15). The piecewise linear function approximation with seven regions is shown in Table 6.3 and the quantization table with eight values (i.e. three-bits size) is shown in Table 6.4 respectively. Both the approximations, as well as the continuous inverse (arc) \tanh function, are plotted in Fig. 6.8.

From Fig. 6.7 in the previous Section, it can be pointed out that there is not much

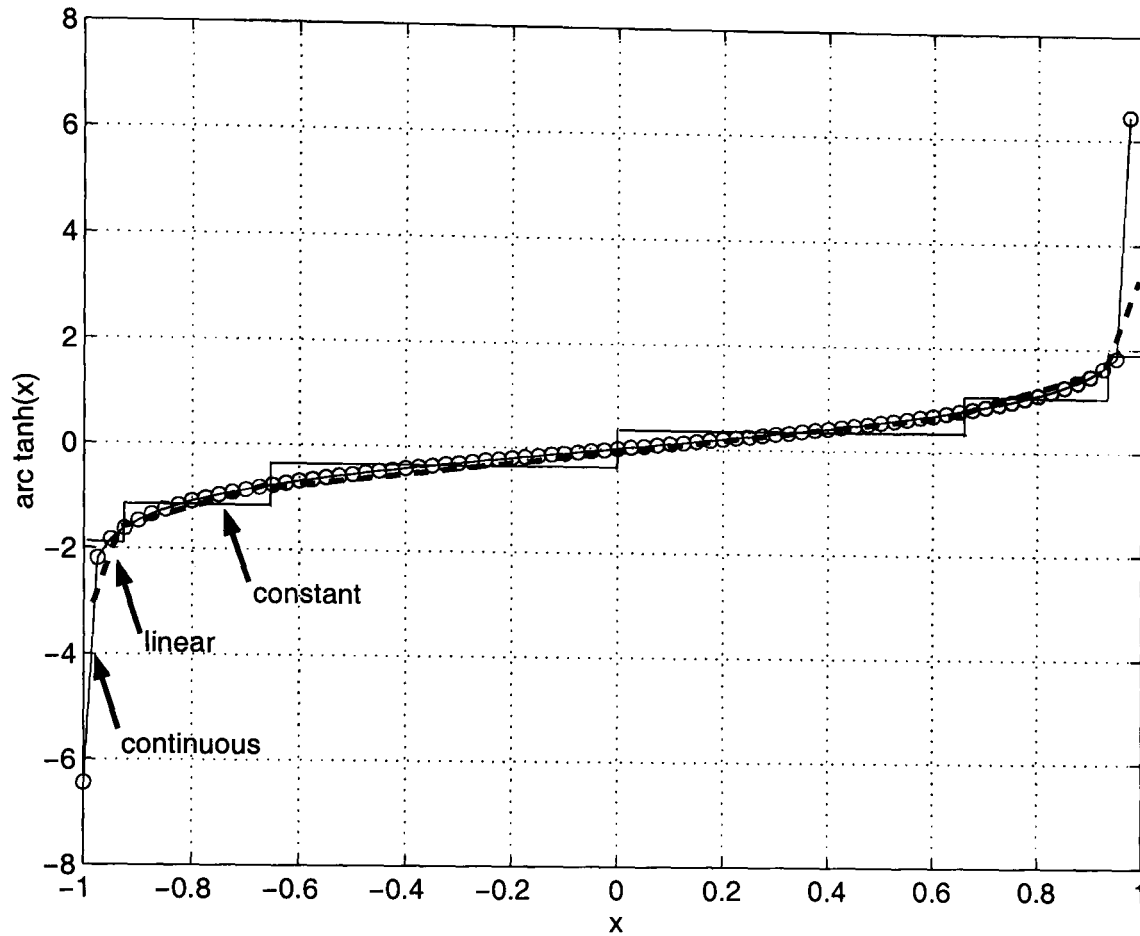


Figure 6.8: Example of inverse (arc) \tanh function (continuous-circle line) and approximations with piecewise linear function (dashed line) and quantization (constant function-solid line).

difference in the resulting BER performance in the low to medium SNR region, i.e. at BER greater than 10^{-5} , when $\tanh^{-1}(x_0) = 3$ or 7. Thus, the regions of the piecewise linear function can be reduced from seven to five. In addition, the size of the quantization table can be reduced from eight to six values. As a consequence, further complexity reduction to the two approximation methods is feasible by observing the operating SNR value. This was also shown in the two approximation methods of the \tanh function in Section 6.3.3.

As reported in Section 6.2, the inverse (arc) \tanh function is monotonically increasing and has odd symmetry, i.e. $f(x) = -f(-x)$. Therefore, when using the check-node update from Eq. (6.2), absolute values may be used. That reduces the required regions of the piecewise linear function from seven to four and also the quantization table

from eight to four values. This is assuming $\tanh^{-1}(x_0) = 7$. However, the *sign* of the arguments to the inverse (arc) \tanh function needs to be known, which adds some extra computational complexity.

Finally, in computer simulation results to be presented in the next Section, both piecewise linear function and quantization approximations are applied to the \tanh function as well. The exact approximation values used have already been reported in Section 6.3.3. The reason for that is because there was no significant impact on the resulting BER performance in either the continuous form or with two approximations to the \tanh function when already using the two approximations to the inverse (arc) \tanh function. Thus, the check-node update from Eq. (6.2) avoids the exact computation of the \tanh function and the overall computational complexity can be further reduced.

6.4.4 Computer Simulation Results

Computer simulations are run for the same randomly constructed rate half regular LDPC codes with $(d_s, d_c) = (3, 6)$ and different block sizes, as in Section 6.3.4. That is, assumption of either *short* block size, e.g. (96, 48), *medium* block size, e.g. (504, 252) and (1008, 504), or *large* block size, e.g. (8000, 4000). Similarly, BPSK modulation and the AWGN channel assumed and also two cases of maximum number of decoding iterations. In the *first case* a relatively small number is considered, e.g. 10. In the *second case*, a greater number is considered, e.g. either 50, 80 or 200. This is to show the independent BER performance behaviour of the decoding algorithms and to compare to [116, 15, 16].

The corresponding BER performance of different LDPC codes with or without modification to the inverse (arc) \tanh function in the two cases of maximum number of decoding iterations is shown in Figs. 6.9 and 6.10 respectively. The value of $\tanh^{-1}(x_0) = 7$ is assumed, e.g. see Section 6.4.2. For comparison, BER results obtained with modification to the \tanh function, as from Figs. 6.3 and 6.4 and *Gallager's* approach (shown in dashed lines) from Eq. (6.4) are reported. In the latter case, the same upper and low limits, i.e clipping, were applied to the ϕ function, as in Section 6.3.4.

In Figs. 6.11 and 6.12 it is shown the BER performance of the same block size LDPC

codes with modified inverse (arc) \tanh function (shown in dashed lines) and also using two approximations on top of it to reduce the computational complexity, i.e. seven regions piecewise linear function and eight values of quantization. The first case of maximum number of decoding iterations is shown in Fig. 6.11 and the second case in Fig. 6.12 respectively.

Discussion

Similar to Section 6.3.4, from Figs. 6.9 and 6.10 it is noticed that the proposed *modification* to the inverse (arc) \tanh function provides a performance improvement from 0.5 to 1 dB in the high SNR region, thus reducing the code error floor. This depends on the maximum number of decoding iterations and is observed for codes with relatively short or medium block size. This improvement was explained in Section 6.3.4. In the case of a large block size, there is no significant improvement when considering the proposed modification. Surprisingly, assuming the short code, i.e. (96, 48), and high number of decoding iterations, i.e. 200, there is not much difference in terms of BER performance from either modification to the inverse (arc) \tanh function. This was also explained in Section 6.3.4.

From the same Figures, it is noticed that the BER performance with either the modified inverse (arc) \tanh function, the modified \tanh function or *Gallager's* approach is approximately the same. In the high SNR region, there is a small performance degradation of *Gallager's* approach, due to the fact that the upper and lower limit values of ϕ function were not optimised in computer simulations.

Similarly, computer simulation results reported in [116, 15, 16] are in agreement with the presented ones using *modification* to the inverse (arc) \tanh function. This is the case of the (504, 252) LDPC code with maximum 50 decoding iterations [116], the (1008, 504) LDPC code with maximum 80 decoding iterations [15] and the (8000, 4000) LDPC code with maximum 100 decoding iterations [16] respectively.

As in Section 6.3.4, from Figs. 6.11 and 6.12 it is noticed that the *piecewise linear approximation* provides a small improvement, e.g. 0.13 dB, to the BER performance compared to the continuous inverse (arc) \tanh function using modification in both cases,

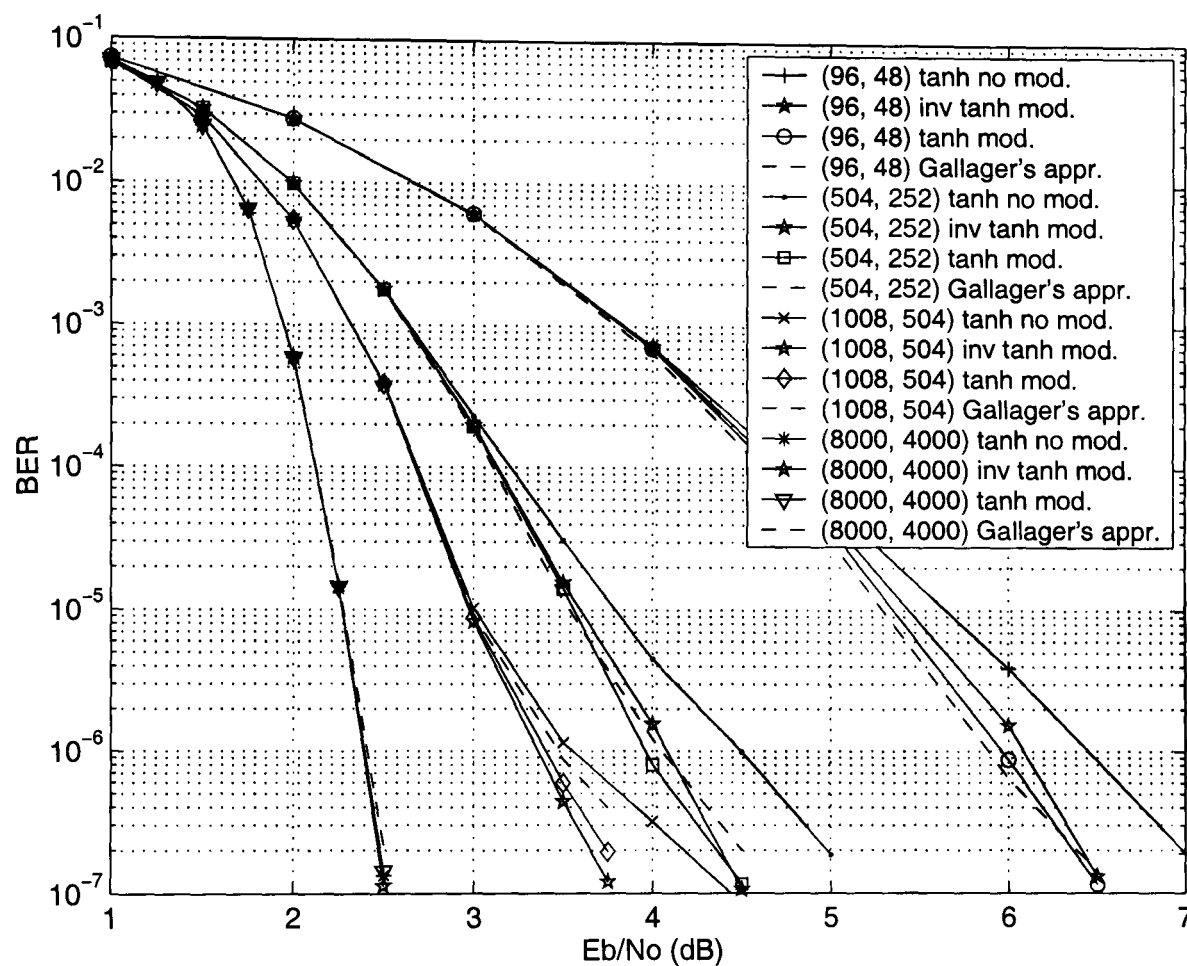


Figure 6.9: BER performance with or without modification to the inverse \tanh function and comparison to *Gallager's* approach (dashed lines) and modified \tanh function. Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.

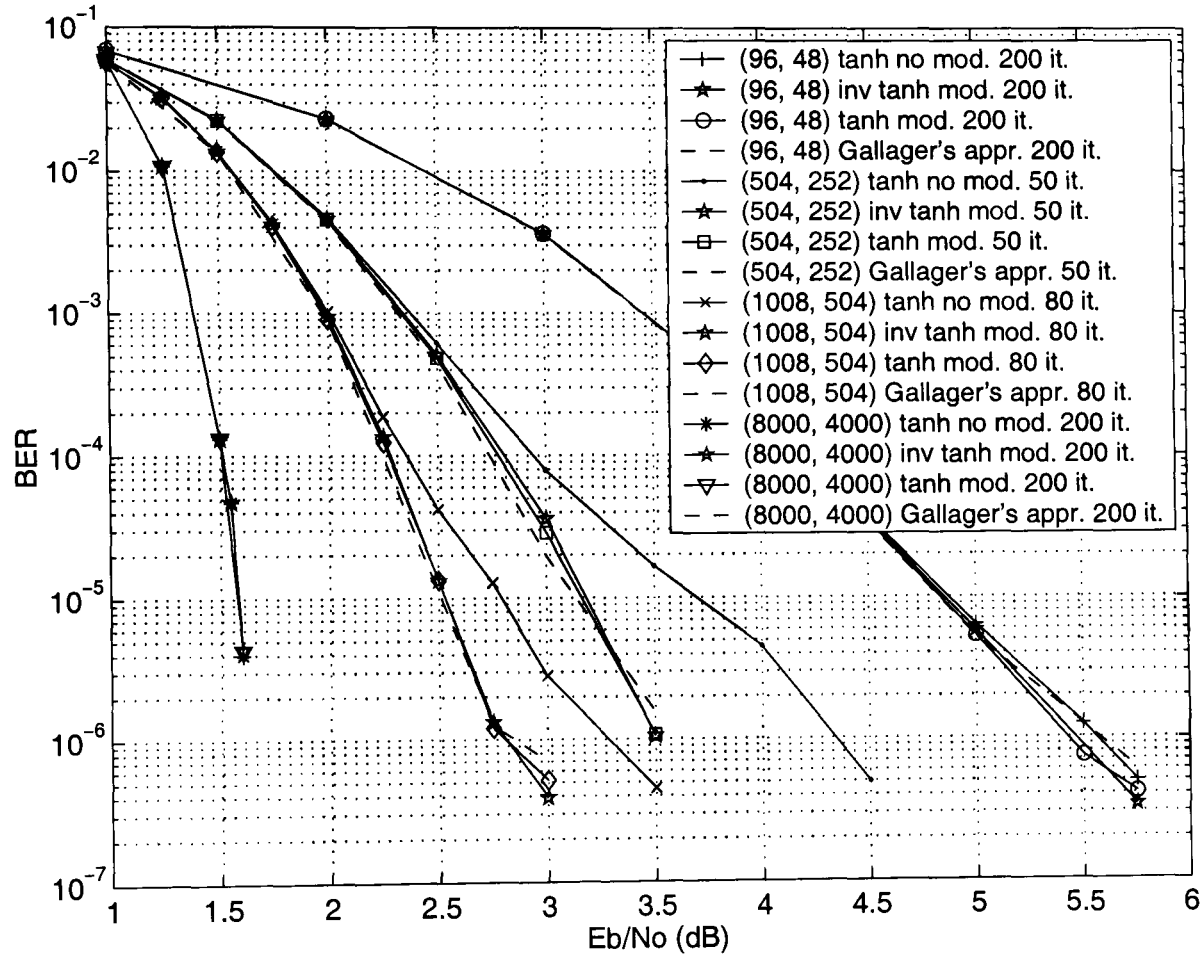


Figure 6.10: BER performance with or without modification to the inverse \tanh function and comparison to *Gallager's* approach (dashed lines) and modified \tanh function. Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel.

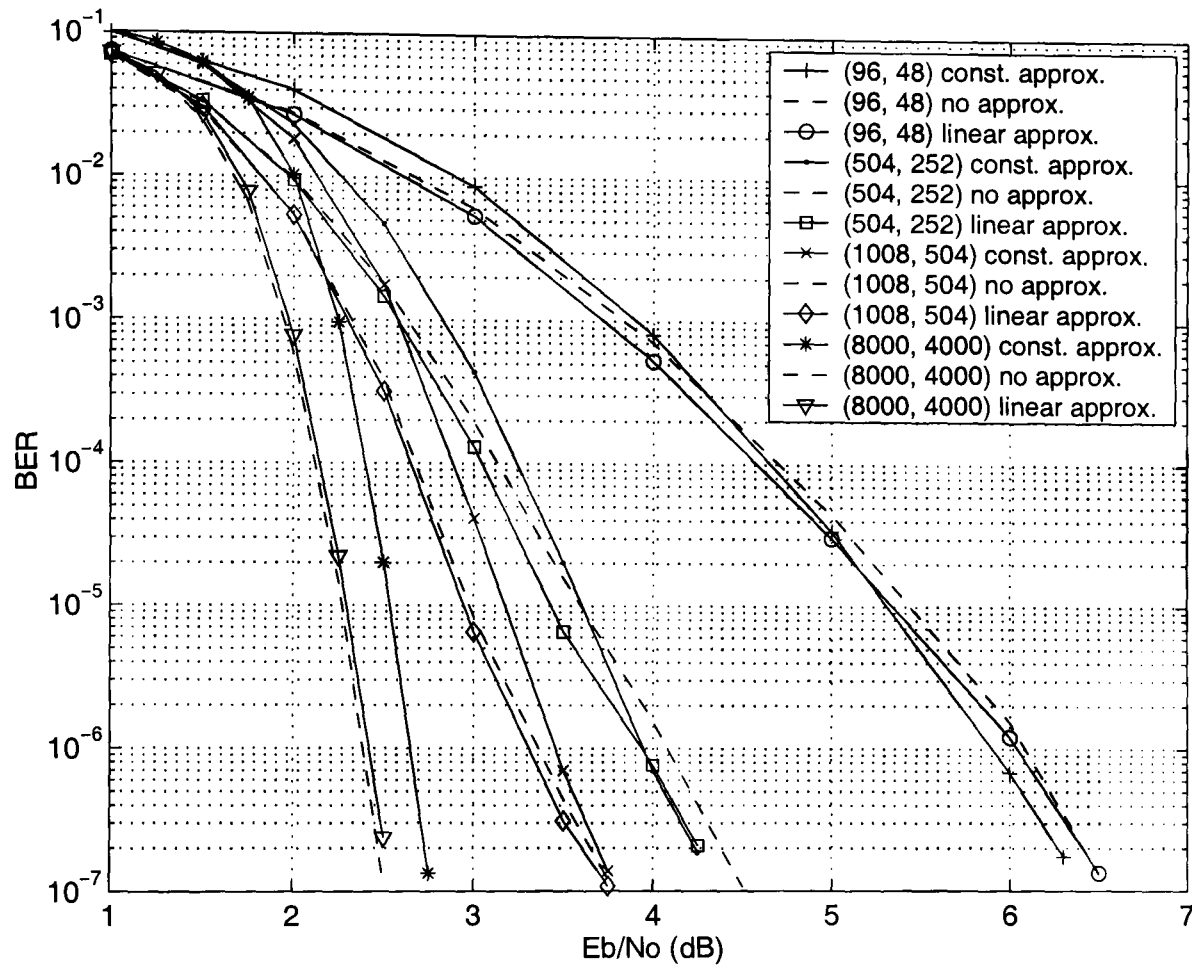


Figure 6.11: BER performance with modified inverse \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and maximum 10 decoding iterations in the AWGN channel.

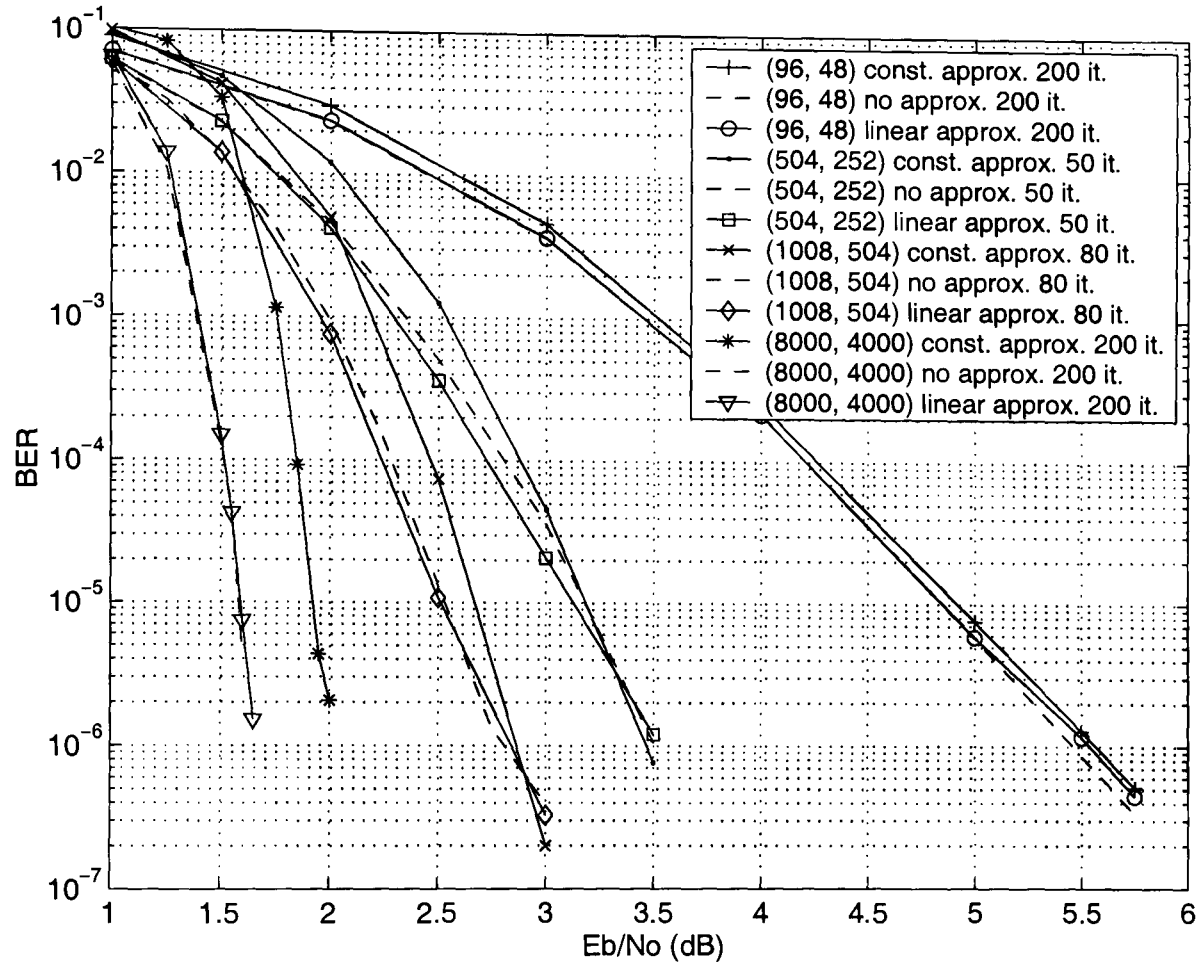


Figure 6.12: BER performance with modified inverse \tanh function (dashed lines) and also using piecewise linear function and quantization approximations. Various block sizes of LDPC codes, coding rate $R = 1/2$ and either maximum 50, 80 or 200 decoding iterations in the AWGN channel.

i.e. $\tanh^{-1}(x_0) = 7$. This depends on the maximum number of decoding iterations and does not seem to happen for the large block size code and also for the short code with high number of decoding iterations. The same explanation on the behaviour of the message-passing algorithm for short to medium block sizes can be given, as in Section 6.3.4. As reported in the same Section, a similar phenomenon was observed when considering the normalised/offset min-sum algorithm [16] and also the min-sum algorithm with clipping [117] in the high SNR region.

On the other hand, the application of *quantization* to the inverse (arc) \tanh function provides approximately 0.25 dB degradation in terms of BER performance compared to the continuous inverse (arc) \tanh function using modification in both cases, i.e. $\tanh^{-1}(x_0) = 7$. This is the case for codes with medium block sizes. Similarly, this depends on the maximum number of decoding iterations and the performance degradation is increased with increasing the block size. In the high SNR region it seems that the quantization table is optimised, especially for codes with medium block size, so that BER results with this approximation can outperform the corresponding ones with continuous inverse (arc) \tanh function.

Similarly, it is noted that in the high SNR region, the BER performance degradation of the *min-sum* algorithm with respect to the sum-product algorithm is 0.3 dB, assuming the (1008, 504) LDPC code with maximum 80 decoding iterations [15] and 0.5 dB, assuming the (8000, 4000) LDPC code with maximum 100 decoding iterations [16]. Therefore, the quantization method applied to the sum-product algorithm provides performance benefits compared to the min-sum algorithm, but with relative increase to the decoding complexity.

6.5 Summary

In this Section the most important issues on reduced complexity decoding algorithms for LDPC codes are summarised.

- Logarithmic domain decoding algorithms for LDPC codes have implementation advantages over the corresponding ones in the probability domain, in a similar

way to turbo decoding.

- Different check-node update rules as well as reduced complexity decoding algorithms using LLR values are described in [16]. Hardware decoder implementation issues with finite precision are also reported.
- There exists no related work on the SPA in the logarithmic domain with check-node update based on the *tanh rule*, mainly because of the increased required computational complexity.
- In this case, clipping was applied to either the *tanh* or inverse (arc) *tanh* function to cope with the infinite value approximation problem. Two methods of computational complexity reduction were also proposed. That is, piecewise linear approximation with seven regions and quantization table with eight values for both the *tanh* and inverse (arc) *tanh* functions. Further reduction in complexity of the two approximation methods is feasible by observing the operating SNR value.
- Computer simulation results presented for regular LDPC codes with short to medium block sizes in the AWGN channel have shown that the check-node update based on the *tanh rule* suffers from error floor at low BER values. Clipping to either the *tanh* or inverse (arc) *tanh* function was shown to be essential to reduce the observed error floor.
- Piecewise linear approximation on top of clipping provides a small improvement, e.g. 0.13 dB, to the BER performance compared to either the continuous *tanh* or inverse (arc) *tanh* function. This is when assuming LDPC codes with short to medium block sizes and is explained because of the presence of short cycles of the code that makes the message-passing algorithm to operate with *correlated* instead of *uncorrelated* values.
- On the other hand, quantization on top of clipping to either the *tanh* or inverse (arc) *tanh* function provides a maximum degradation of 0.25 dB in terms of BER performance compared to the continuous case. This is when assuming LDPC codes with short to medium block sizes. However, this quantized algorithm is

superior to the the min-sum algorithm, but with relative increase to the decoding complexity.

- With the two approximations described above to either the *tanh* or inverse (arc) *tanh* functions on top of clipping, no error floor was observed in most of the cases at BER greater than 10^{-7} . This is in contrast to the error floor reported in [16] using *Gallager's* approach with finite precision BER results.

Chapter 7

Conclusions

7.1 Research Work Summary and Contribution

Nowadays, turbo codes are seen to be quite mature. They have been in existence in the 3G mobile handsets, in the NASA mission to Saturn and in the return link of digital video broadcasting over either terrestrial or satellite signals. After the application to many standards, practical applications have already been introduced. Such an example can be found in Bell Labs Research, *Lucent* technologies. A channel decoder chip compliant with the 3GPP standard has been implemented that supports both data and voice services in a unified turbo/Viterbi decoder architecture [120].

It is also believed that LDPC codes are the next generation of capacity-approaching codes to be applied to standards. This is mainly because they reduce the error floor to lower BER values compared to turbo codes. A start has been made in the new DVB-S2 standard. In this way, it has been shown that hardware-based LDPC decoders can provide the best trade-off between performance and complexity over satellite transmission signals. Among the competitors were both SCCC and turbo codes in duo-binary form [32]. *Flarion* technologies [121] has used LDPC hardware decoders for Wireless Local Area Networks (WLANs) combined with orthogonal frequency division multiplexing (OFDM) signals. Furthermore, the new CCSDS standard is going to be updated to its new version, considering turbo-like codes, i.e. an accumulate RA code.

Case studies for the application of capacity-approaching codes to ASDLs [53] and magnetic data recording [54] have shown that both turbo and LDPC codes are strong candidates. It seems that the increase to the decoding complexity is reasonable, while more effort needs to be done in order to find the way to future practical systems (e.g. coding/decoding parameters optimisation, latency and trade-off between performance and complexity).

There are also some recent developments in higher layers of communication systems. Raptor codes from *Digital Fountain* [122] have been adopted by the 3GPP to provide MBMS services in Release 6 [70]. Packet level FEC is applied in the transport layer of UMTS networks in order to facilitate with lower error rates, suitable for video transmission. In such a case, conventional encoders are used as in the physical layer, but they process packets of bits rather than individual bits.

It seems that the channel coding research field has almost closed [26]. Nowadays, any simple code with certain properties (i.e. turbo-like code) can be iteratively decoded by the sum-product algorithm approaching the channel capacity limit. This is true for large frame (or block) sizes, resulting to codes with no cycles on graphs representation. On the other hand, there is still a gap from the channel capacity limit for small and moderate frame sizes, due to the presence of cycles on graphs representation. Decoding complexity is still an open issue of research, especially for practical codes that are used in particular standards. Another interesting application is fading channels, where iterative decoding codes perform more than 1 dB from the channel capacity limit. Iterative decoding is a sub-optimum algorithm for global decoding of turbo or LDPC codes. It would be desirable to find an appropriate algorithm with MLD performance but with limited decoding complexity.

Another *challenge* of iterative decoding schemes is to find reduced complexity algorithms with limited performance degradation compared to the optimum ones. This thesis has tried to contribute to this research field, although turbo and LDPC codes have been known for more than ten years. As a summary of the research work contribution, the following issues can be highlighted.

- A novel improved SOVA iterative decoding algorithm for binary turbo codes was

proposed in Chapter 3. This is based on scaling the extrinsic information with a constant factor, the value of which needs only to be increased in the last decoding iteration.

- It was shown that this approach, namely *norm2* SOVA, improves the performance compared to the conventional SOVA turbo decoder, not only at medium but also at low BER values. The reason for that is the reduction of the correlation coefficient between intrinsic and extrinsic information.
- No error floor was observed at BER down to 10^{-6} , assuming turbo codes with large frame lengths and high number of decoding iterations. This is 0.3 dB inferior to the BER performance of Log-MAP decoding algorithm in the AWGN channel.
- Four novel SISO decoding algorithms for binary turbo codes were proposed in Chapter 4. Good trade-off between performance/complexity was shown and with respect to Max-Log-MAP and Log-MAP iterative decoding. It is based on the *max/max** operation replacement to either the forward/backward recursion or the soft-output computation of the Max-Log-MAP or Log-MAP algorithms.
- For example, assuming a 16-states turbo code and medium BER values, *SISO – B* improves the iterative Max-Log-MAP performance up to 0.28 dB, but with 38.46% more LUT operations and 17.54% extra additions. *SISO – C* degrades the iterative Log-MAP performance up to 0.13 dB, but with 38.46% fewer LUT operations and 12.05% fewer additions.
- In the same Chapter, another *M* novel SISO decoding algorithms for binary turbo codes were proposed, depending on the turbo encoder memory order *M*. They show the trade-off between performance/complexity and are based on the application of the *max/max** operation in different *levels* when computing the soft-output of the Max-Log-MAP or Log-MAP algorithms.
- For example, assuming a 16-states turbo code and medium BER values, *MLM – max* – L1* has 2.56% more LUT operations and 1.17% more additions than the Max-Log-MAP iterative decoder and improves the performance to 0.1 dB.

$LM - max^* - L123$ has 20.51% fewer LUT operations and 6.43% fewer additions than the Log-MAP iterative decoder and 0.01 dB performance degradation.

- It is believed that the variety of the proposed SISO decoding algorithms for binary turbo codes, as from Chapter 4, has closed the gap between Max-Log-MAP and Log-MAP turbo decoding.
- The max/max^* operation replacement from Chapter 4 was applied to duo-binary turbo codes and more specifically to the DVB-RCS turbo code in Chapter 5. Good trade-off between performance/complexity was shown and with respect to Max-Log-MAP and Log-MAP iterative decoding.
- For example, assuming low coding rates and medium BER/FER values, $SISO - B$ improves the iterative Max-Log-MAP performance up to 0.13 dB, but with 36.84% more LUT operations and 15.38% extra additions. $SISO - C$ degrades the iterative Log-MAP performance up to 0.08 dB, but with 36.84% fewer LUT operations and 10.85% fewer additions.
- In the same Chapter, a novel Constant Log-MAP algorithm for duo-binary turbo codes was proposed. The simplified max^* operator is processed over pair of values, instead over four values. This is the main difference from an existing algorithm, which has the same decoding complexity.
- For example, assuming low coding rates and high to medium BER/FER values, the proposed algorithm is 0.2 dB superior to the existing algorithm. It also performs close to Log-MAP decoding with a performance degradation less than 0.02 dB, similar to the binary case.
- It is believed that the two presented algorithm approaches, as from Chapter 5, provide good alternative solutions to Max-Log-MAP and Log-MAP iterative decoding for the DVB-RCS turbo code.
- Clipping to either the \tanh or inverse (arc) \tanh function, which are used in check-node update computation based on the \tanh rule, was shown to be essential to reduce the observed error floor of regular binary LDPC codes with short to medium block sizes.

-
- Piecewise linear approximation on top of clipping to either the \tanh or inverse (arc) \tanh function provided small improvement, e.g. 0.13 dB, to the BER performance compared to the continuous case. This was explained because of the presence of short cycles of LDPC codes that makes the message-passing algorithm to operate with *correlated* instead of *uncorrelated* values.
 - Quantization on top of clipping to either the \tanh or inverse (arc) \tanh function provided a maximum degradation of 0.25 dB in terms of BER performance compared to the continuous case. This is when assuming LDPC codes with short to medium block sizes.
 - With the two above approximations to either the \tanh or inverse (arc) \tanh functions and on top of clipping, no error floor was observed in most of the considered LDPC codes at BER greater than 10^{-7} .

7.2 Suggested Future Research Work

Suggested directions for future research work can be split into two parts. The first part includes research work that can be applied directly, inspired by the relevant work in Chapters 3 to 6. The second part can be seen as longer term work.

Starting with the first part, improved SOVA decoding for duo-binary turbo codes was not considered in Chapter 3. Therefore, it is interesting to show how the two-step normalisation approach works in this case. Moreover, relevant work on improved SOVA decoding for duo-binary turbo codes has to be considered.

In Chapter 4 all the proposed SISO decoding algorithms (based on either the \max/\max^* operation replacement method or the application of the \max/\max^* operation in different *levels*) have used the conventional Max-Log-MAP and Log-MAP algorithms for binary turbo codes. That is, no scaling was applied to the extrinsic information. Considering the fact that the improved Max-Log-MAP algorithm (i.e. applying scaling) has near Log-MAP performance with obvious decoding complexity savings, the investigation of the proposed SISO decoding algorithms when applying scaling seems to be quite reasonable.

As shown in Chapter 5, the \max/\max^* operation replacement method from Chapter 4 was considered. The \max/\max^* operation in different *levels*, as from Chapter 4, is also applicable. This can be done to show a further trade-off between the performance of SISO decoding algorithms and the corresponding complexity, in case of the DVB-RCS turbo code.

In Chapter 6 the performance of the LLR-SPA based on *Gallager's* approach was not optimised in the high SNR region. In this case, the upper and lower limits of the ϕ function have to be reconsidered. Also, two approximation methods to the ϕ function can be proposed, in a similar way to the proposed approximation methods to the \tanh and inverse (arc) \tanh functions. That is, approximations based on piecewise linear function and quantization table. Also, as reported in Chapter 6, the quantization table of the \tanh function needs to be optimised in the high SNR region. Furthermore, all the presented computer simulation results were based on the AWGN channel. It would be interesting to investigate different channel types (e.g. uncorrelated fading channel) and also the recently proposed DVB-S2 LDPC encoder with different block lengths. A universal method on reduced complexity decoding algorithms for LDPC codes would be targeted.

As a longer term research work, it was seen that the DVB-RSC turbo encoder has been very recently extended to 16-states [3] and the reported FER results are quite promising. It would be interesting to investigate this new encoding scheme and apply the two decoding algorithms proposed in Chapter 5. This would result in alternative decoding solutions to the improved Max-Log-MAP algorithm that was considered in [3]. In addition, extended performance investigation (by means of theoretical analysis) of the DVB-RCS turbo code at very low BER values, e.g. equal to 10^{-11} , is crucial when considering practical applications, such as video services. That would enable us to make some useful remarks on the performance behaviour of the decoding algorithms proposed in Chapter 5 at very low BER values.

Quite recently in [16], all the reduced decoding complexity algorithms for LDPC codes have been almost covered. However, the serial implementation approach of check-node update has similarities to the forward-backward algorithm applied to trellis decoding.

Furthermore, an operation similar to the \max^* operation was defined using a LUT of values to reduce the decoding complexity, e.g. see Eq. (6.7). In this case, the idea based on the two algorithm approaches, as from Chapter 4, could be applied. That is, mixing some of the new operations, which are now defined as from the min-sum and sum-product algorithm, either to the forward/backward direction or the LLR computation and apply them in different *levels*.

In our research work, the use of LUT (such as in the proposed Constant Log-MAP algorithm for duo-binary turbo codes) and also piecewise linear function with seven values and quantization table with eight values (such as to approximate the \tanh /inverse (arc) \tanh functions for LDPC decoding) have been considered. Usually, fixed point implementation with finite precision values and quantization effects are crucial in hardware implementations. In this case, the BER performance and the decoding complexity play an important role. This approach would enable the direct impact of the proposed decoding algorithms to a hardware decoding implementation.

All the proposed improved decoding algorithms have considered BPSK modulation and ideal propagation channel conditions, i.e. AWGN/uncorrelated fading channel. Considering the first fact, turbo code extensions to high order modulation schemes, such as 8-PSK and 16-QAM, are feasible using the *pragmatic* approach [123, 124, 33]. In this case, a variety of spectral efficiencies are supported and the decoder needs not to be redesigned, thanks to the application of a binary turbo encoder, a puncturing technique and appropriate signal mapping. The key idea is that the demodulator output provides soft bit LLR values, before entering the iterative decoder input. The same approach can be applied to LDPC codes. Therefore, the proposed improved decoding algorithms performance could be investigated over high order modulation schemes.

Considering the second fact as above, realistic mobile satellite fading channels using the so-called gap fillers [30, 36], the effect of the satellite non-linear high power amplifier and the use of predistortion or precoding techniques would be of significant importance. Another interesting area would be the investigation of adaptive coding and modulation techniques, which are currently used in the DVB-S2 standard, based on different traffic models. In this way, the BER performance of the proposed decoding algorithms could

be taken under consideration to the higher communication layers, so that a practical satellite communication system can be designed more effectively without wasting extra resources. In overall, this extension would enable some useful remarks on the application of efficient iterative decoding techniques not only to deep-space satellite communications but also to some practical satellite communication systems.

Appendix A

Publications List

Book Chapters (co-authored)

1. S. Papaharalabos, '**Forward error correction**', Chapter 5, *Digital Satellite Communications*, Satnex consortium, Springer, to be published, 2006.
2. S. Papaharalabos, '**Modulation**', Chapter 6, *Digital Satellite Communications*, Satnex consortium, Springer, to be published, 2006.

Journals

3. S. Papaharalabos, P. Sweeney, and B. G. Evans, '**Modification of branch metric calculation to improve iterative SOVA decoding of turbo codes**', *IEE Elect. Letters*, vol. 39, no. 19, pp. 1391-1392, Sep. 2003.
4. S. Papaharalabos, P. Sweeney, and B. G. Evans, '**SISO algorithms based on combined max/max* operations for turbo decoding**', *IEE Elect. Letters*, vol. 41, no. 3, pp. 142-143, Feb. 2005.

Conferences

5. S. Papaharalabos, P. Sweeney, and B. G. Evans, '**Turbo coding performance evaluation using an improved iterative SOVA decoder**', in *Proc. AIAA Inter. Commun. Satel. Syst. Conf. (ICSSC)*, Monterey, USA, May 2004, No. 3108.
6. S. Papaharalabos, P. Sweeney, and B. G. Evans, '**A new method of improving**

SOVA turbo decoding for AWGN, Rayleigh and Rician fading channels', in *Proc. IEEE Vec. Tech. Conf. (VTC) Spring*, Milan, Italy, May 2004, pp. 2862-2866.

7. S. Papaharalabos, G. Albertazzi, P. Sweeney, B. G. Evans, A. Vanelli-Coralli and G. E. Corazza, '**Performance evaluation of a modified sum-product decoding algorithm for LDPC codes**', in *Proc. IEEE Inter. Works. Satel. and Space Commun. (IWSSC)*, Siena, Italy, Sep. 2005.

8. S. Papaharalabos, P. Sweeney, and B. G. Evans, '**Max/max* operation replacement to improve the DVB-RCS turbo decoder**', in *Proc. AIAA Inter. Commun. Satel. Syst. Conf. (ICSSC)*, Rome, Italy, Sep. 2005.

Submitted work

9. S. Papaharalabos, P. Sweeney, B. G. Evans and P. T. Mathiopoulos, '**Improved performance iterative SOVA decoding**', submitted to *IEE Proc. Commun.*, May 2005.

10. S. Papaharalabos, P. Sweeney, and B. G. Evans, '**Efficient Constant Log-MAP decoding for duo-binary turbo codes**', submitted to *IEEE Turbo Coding 2006*, Oct. 2005.

11. S. Papaharalabos, P. Sweeney, and B. G. Evans, '**Filling the gap between Log-MAP and Max-Log-MAP turbo decoding**', submitted for Journal publication, Nov. 2005.

12. S. Papaharalabos, G. Albertazzi, P. Sweeney, B. G. Evans, A. Vanelli-Coralli and G. E. Corazza '**Modified log-domain sum-product algorithm for LDPC codes**', submitted for Journal publication, Nov. 2005.

Appendix B

Turbo and LDPC Codes Computer Simulated Performance Validation

Different decoding algorithms are considered in computer simulations set up for three kinds of codes, e.g. see Section 2.7.1. That is

- Binary turbo codes, Figs. B.1-B.10. This is the case of Log-MAP, Max-Log-MAP and SOVA algorithms.
- Duo-binary turbo codes, Figs. B.11-B.15. This is the DVB-RCS turbo code with Log-MAP and Max-Log-MAP algorithms.
- LDPC codes, Figs. B.16, B.17. This is the LLR-SPA (logarithmic domain SPA) using *Gallager's* approach.

Exact simulation parameters were given in Tables 2.1-2.3. In the following Figures, simulation results are shown in solid lines and reference work in dashed lines.

The *Log-MAP algorithm* for *binary* turbo codes is compared to *Berrou* [6], *Robertson* [7], *Hanzo* [8] and *Valenti* [9]. This is to take into account different parameters, such as the turbo code generator polynomials, the coding rate, the frame size, the

interleaver type, the channel type and the number of decoding iterations. The resulting performance validation is shown in Figs. B.1-B.4. Excellent match is noticed between simulation results and reference work. In Fig. B.4 it is noticed approximately 1 dB performance degradation with no CSI compared to CSI available in an uncorrelated Rayleigh fading channel, which is acceptable, as the channel capacity limit is also increased in this case by approximately 1 dB [1].

The *Max-Log-MAP algorithm* for *binary* turbo codes is compared to *Robertson* [7], *Hanzo* [8] and *Valenti* [9]. There were no results with Max-Log-MAP decoding reported in *Berrou's* work [6]. Different parameters selection, such as in case of Log-MAP algorithm, and related performance comparison is shown in Figs. B.5-B.7. Again, excellent match is noticed between simulation results and reference work. In Fig. B.5 the performance of Max-Log-MAP decoding is with optimised turbo code interleaver. This explains why this performs better than the simulated performance with pseudo-random interleaver at BER less than 10^{-4} . In Fig. B.7 there is very small BER performance degradation at BER of 10^{-7} . This can be explained by either the smaller number of bit errors that are counted or the correlation properties of random number generators that are used to produce a very large number of transmitted bits. In a fading channel with no CSI available, the same BER performance behaviour occurs, as in Log-MAP decoding.

The *SOVA algorithm* for *binary* turbo codes is compared to *Hanzo* [8] and *Hagenauer* [10]. In [7] *Robertson* provided identical results obtained from *Hagenauer*, while there were no results with SOVA decoding reported either in *Berrou's* [6] or *Valenti's* [9] work. Different parameters selection, such as in previous cases, and related performance comparison is shown in Figs. B.8-B.10. Excellent match is noticed between simulation results and reference work. The small BER performance difference in Fig. B.8, although both based on HR-SOVA, can be explained by the different implementations of the SOVA updating rule. Similarly to Log-MAP and Max-Log-MAP algorithms, BER results obtained in the case of a fading channel with no CSI are acceptable.

The *Max-Log-MAP algorithm* for *duo-binary* turbo codes, in the form of DVB-RCS, is compared to *Berrou* [3], *Kabal* [11] and *Yu* [12]. Different turbo code parameters

selection includes variation on the coding rate and frame size. BER/FER results and related comparison is shown in Figs. B.11-B.15. Similarly, excellent match is noticed between simulation results and reference work. In Figs. B.11, B.12 the *improved* Max-Log-MAP algorithm is used instead. This explains the small improvement to the turbo code performance compared to the Max-Log-MAP algorithm.

The *Log-MAP algorithm* for *duo-binary* turbo codes, in the form of DVB-RCS, is compared to *Kabal* [11]. As a comparison, *Berrou's* work [3] with the improved Max-Log-MAP algorithm is also considered. BER/FER results for different coding rates and frame sizes is shown in Figs. B.11-B.14. Similarly, excellent match is noticed between simulation results and reference work from *Kabal*. In addition, it is verified that the Log-MAP algorithm is superior to the improved Max-Log-MAP algorithm, in terms of BER/FER performance, e.g. see Figs. B.11, B.12.

The *SPA algorithm* in the logarithmic domain (*LLR-SPA*) for *LDPC codes* is compared to *MacKay* [13], *Fossorier-1* [14], *Fossorier-2* [16], and *Eleftheriou* [15]. This is to take into account different parameters, such as the block size and the number of decoding iterations. The resulting BER/FER performance validation is shown in Figs. B.16, B.17. Once more, excellent match is noticed between simulation results, using *Gallager's* approach, and reference work. In Fig. B.16 variable maximum number of decoding iterations is used instead. This explains the small BER/FER performance degradation of the simulation results in the high SNR region.

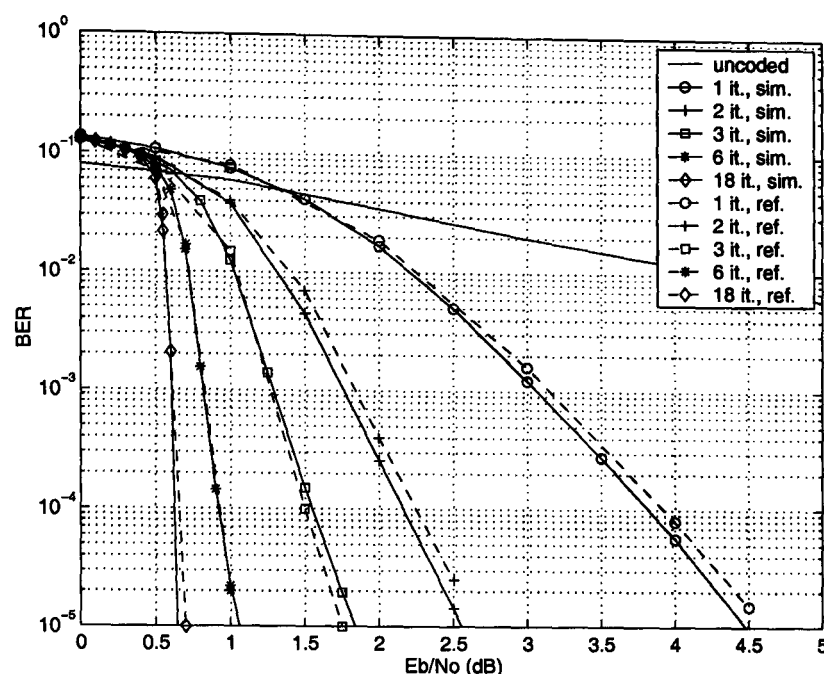


Figure B.1: BER comparison with *Berrou* [6]. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 65536 bits frame size, AWGN channel and different number of decoding iterations. Solid lines-simulation (Log-MAP algorithm) and dashed lines-from reference (MAP algorithm).

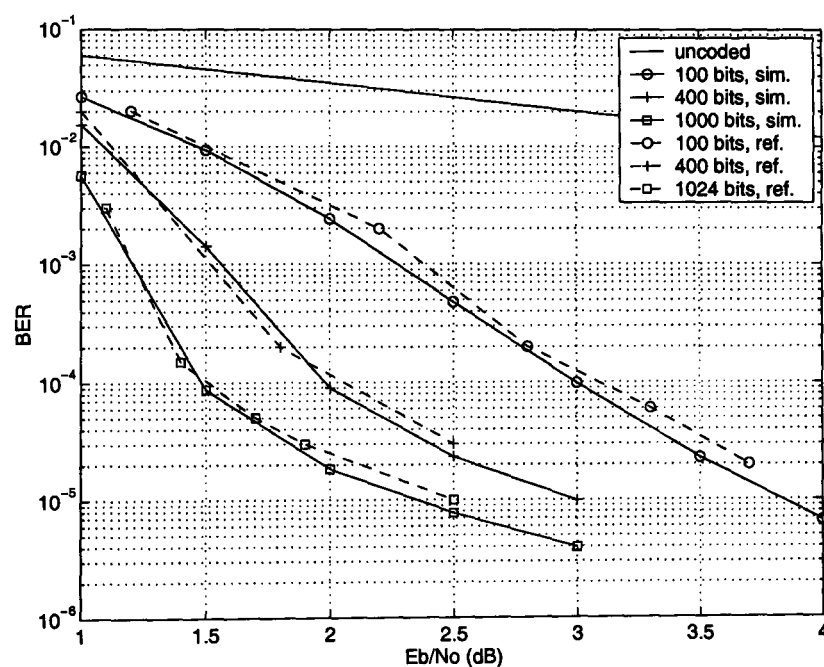


Figure B.2: BER comparison with *Robertson* [7]. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, AWGN channel, 8 decoding iterations and different frame size. Solid lines-simulation (Log-MAP algorithm) and dashed lines-from reference (MAP algorithm).

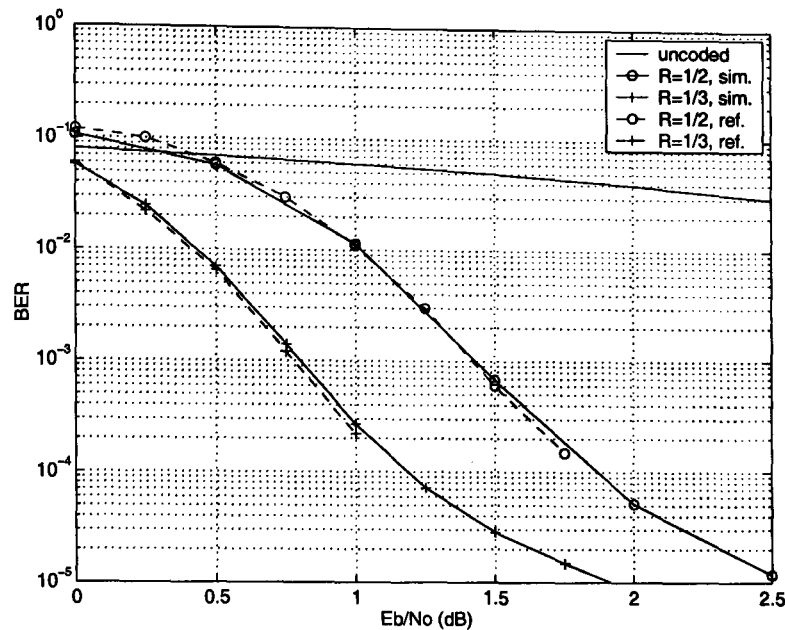


Figure B.3: BER comparison with *Hanzo* [8]. Turbo code generator polynomials $(1,5/7)_o$, i.e. 4-states, 1000 bits frame size, AWGN channel, Log-MAP algorithm, 8 decoding iterations and different coding rate. Solid lines-simulation and dashed lines-from reference.

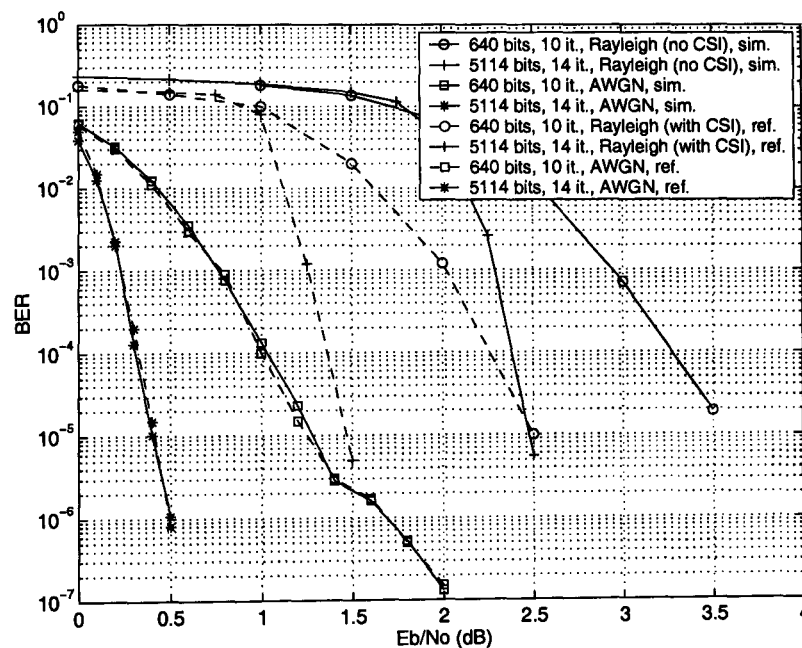


Figure B.4: BER comparison with *Valenti* [9]. Turbo code generator polynomials $(1,15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 3GPP interleaver, AWGN/uncorrelated Rayleigh fading channel, Log-MAP algorithm and different frame size/number of decoding iterations. Solid lines-simulation (with no CSI in fading) and dashed lines-from reference (with CSI in fading).

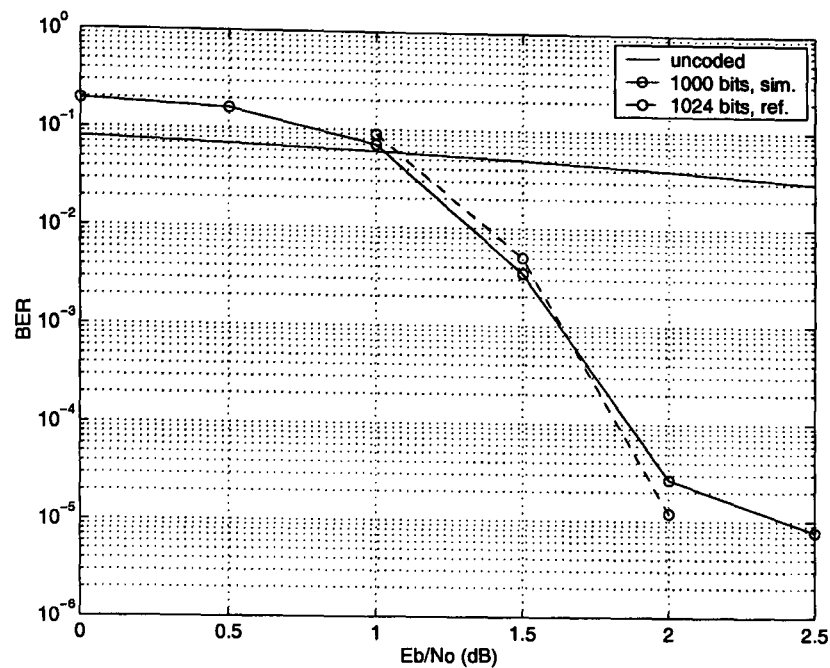


Figure B.5: BER comparison with *Robertson* [7]. Turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 1000 bits frame size, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations. Solid lines-simulation and dashed lines-from reference (with 1024 bits frame size and optimised interleaver).

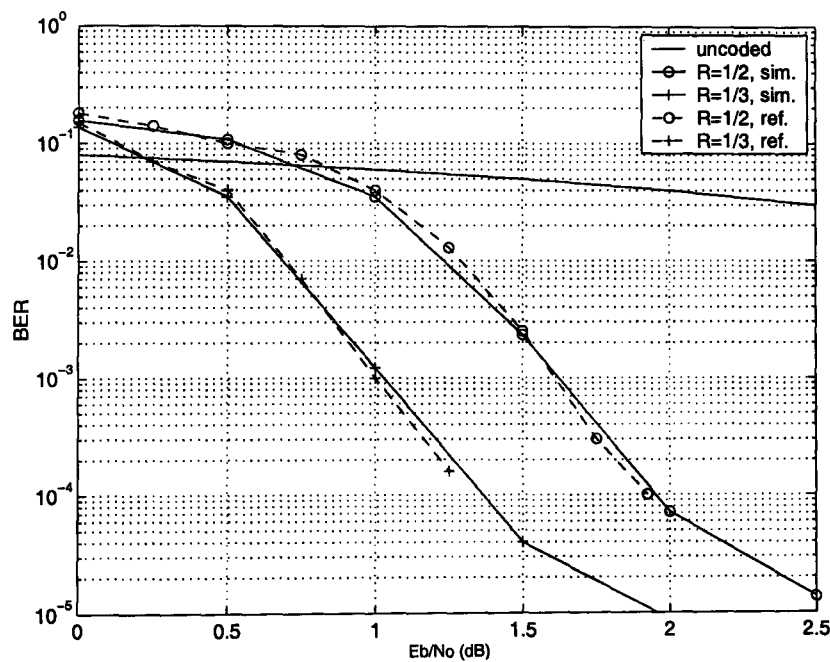


Figure B.6: BER comparison with *Hanzo* [8]. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, 1000 bits frame size, AWGN channel, Max-Log-MAP algorithm, 8 decoding iterations and different coding rate. Solid lines-simulation and dashed lines-from reference.

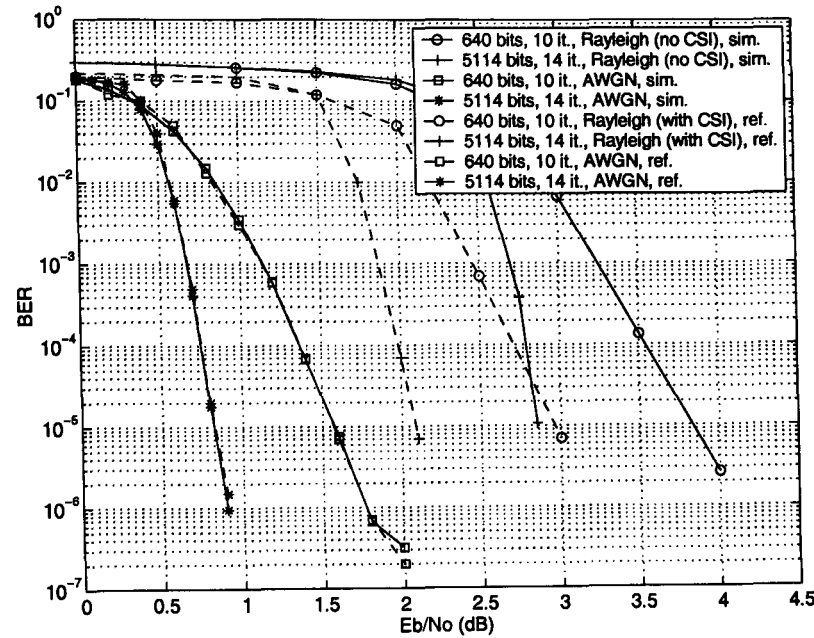


Figure B.7: BER comparison with *Valenti* [9]. Turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 3GPP interleaver, AWGN/uncorrelated Rayleigh fading channel, Max-Log-MAP algorithm and different frame size/number of decoding iterations. Solid lines-simulation (no CSI in fading) and dashed lines-from reference (with CSI in fading).

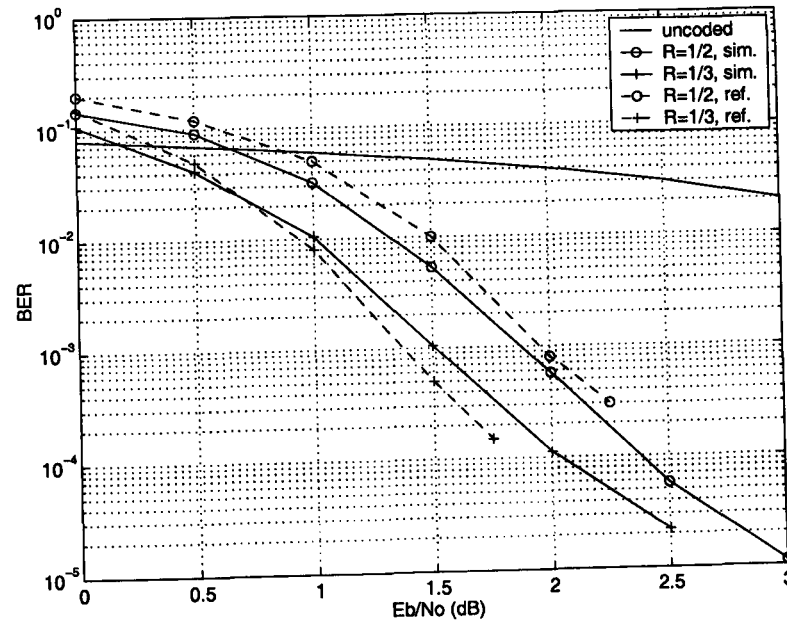


Figure B.8: BER comparison with *Hanzo* [8]. Turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, 1000 bits frame size, AWGN channel, SOVA algorithm, 8 decoding iterations and different coding rate. Solid lines-simulation and dashed lines-from reference.

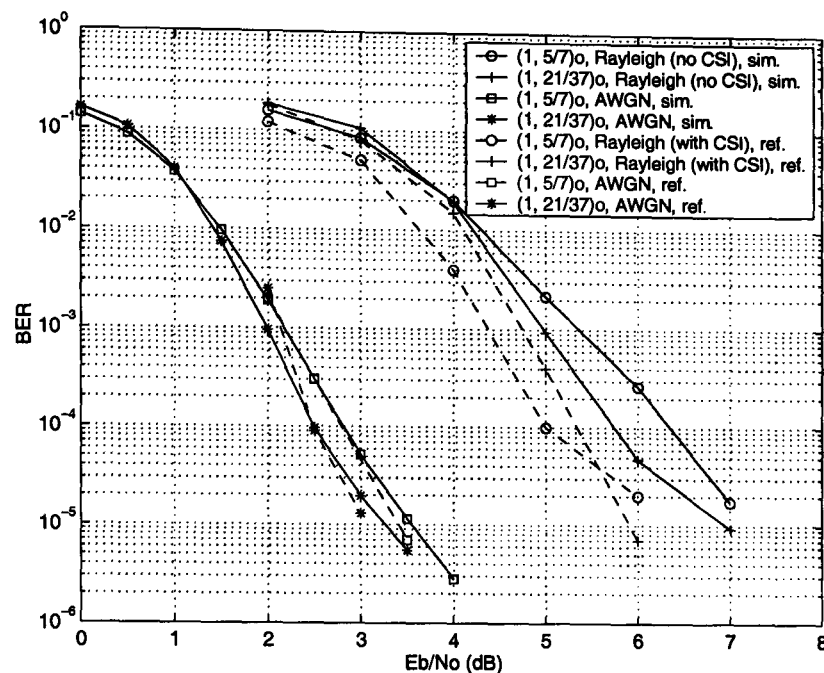


Figure B.9: BER comparison with *Hagenauer* [10]. Coding rate $R = 1/2$, frame size 400 bits, AWGN/uncorrelated Rayleigh fading channel, SOVA algorithm, 8 decoding iterations and different turbo code generator polynomials. Solid lines-simulation (no CSI in fading) and dashed lines-from reference (with CSI in fading).

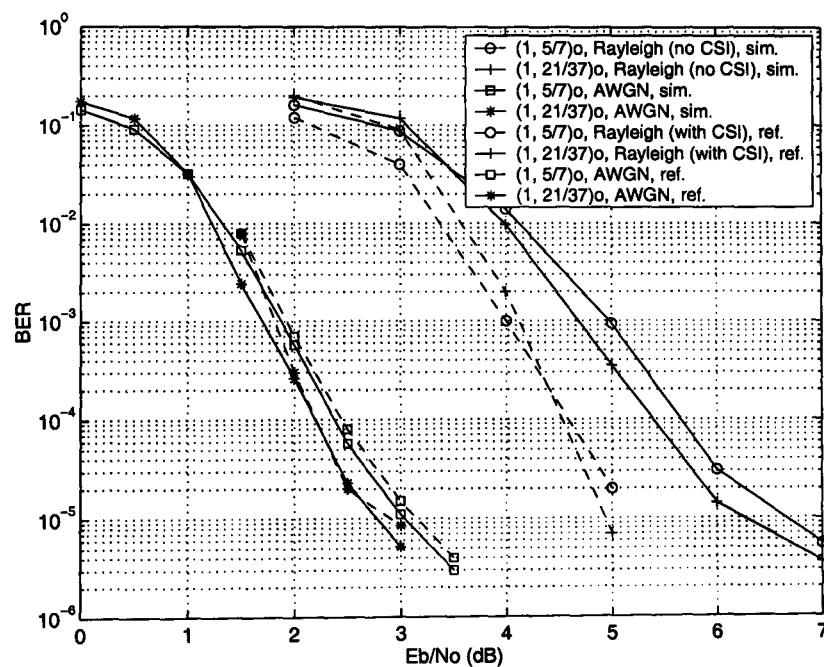


Figure B.10: BER comparison with *Hagenauer* [10]. Coding rate $R = 1/2$, frame size 1000 bits, AWGN/uncorrelated Rayleigh fading channel, SOVA algorithm, 8 decoding iterations and different turbo code generator polynomials. Solid lines-simulation (no CSI in fading) and dashed lines-from reference (with CSI in fading).

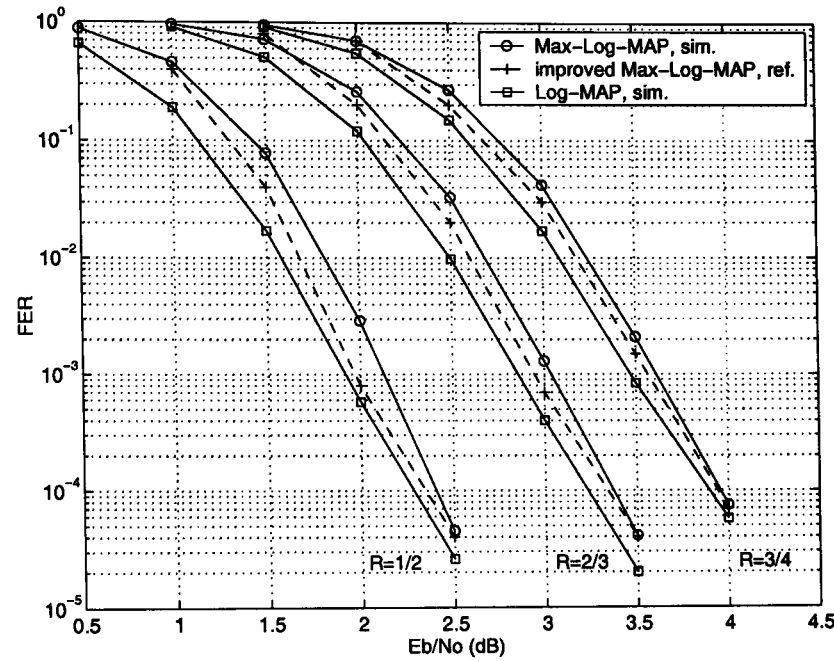


Figure B.11: FER comparison with *Berrou* [3]. Duo-binary turbo code, such as in the DVB-RCS standard, ATM frame size, i.e. 424 bits, AWGN channel, 8 decoding iterations and different coding rates. Solid lines-simulation (Max-Log-MAP and Log-MAP algorithms) and dashed lines-from reference (improved Max-Log-MAP algorithm).

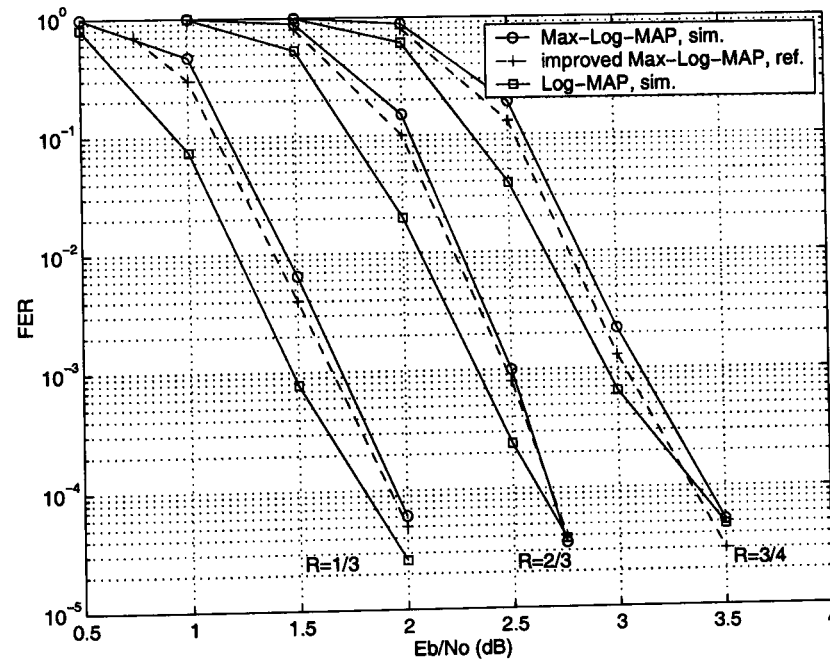


Figure B.12: FER comparison with *Berrou* [3]. Duo-binary turbo code, such as in the DVB-RCS standard, MPEG frame size, i.e. 1504 bits, AWGN channel, 8 decoding iterations and different coding rates. Solid lines-simulation (Max-Log-MAP and Log-MAP algorithms) and dashed lines-from reference (improved Max-Log-MAP algorithm).

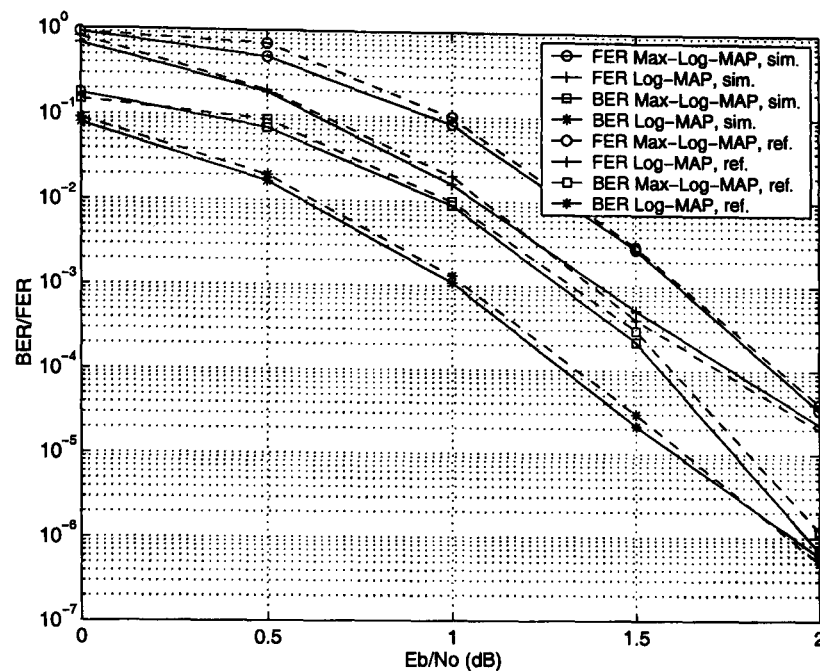


Figure B.13: BER/FER comparison with *Kabal* [11]. Duo-binary turbo code, such as in the DVB-RCS standard, coding rate $R = 1/3$, ATM frame size, i.e. 424 bits, AWGN channel, Max-Log-MAP, Log-MAP algorithms and 8 decoding iterations. Solid lines-simulation and dashed lines-from reference.

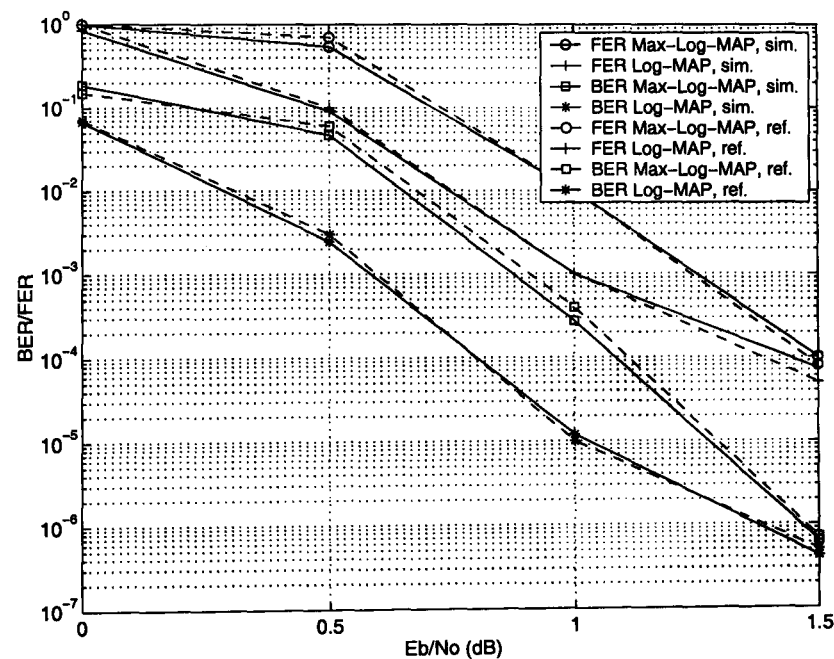


Figure B.14: BER/FER comparison with *Kabal* [11]. Duo-binary turbo code, such as in the DVB-RCS standard, coding rate $R = 1/3$, MPEG frame size, i.e. 1504 bits, AWGN channel, Max-Log-MAP, Log-MAP algorithms and 8 decoding iterations. Solid lines-simulation and dashed lines-from reference.

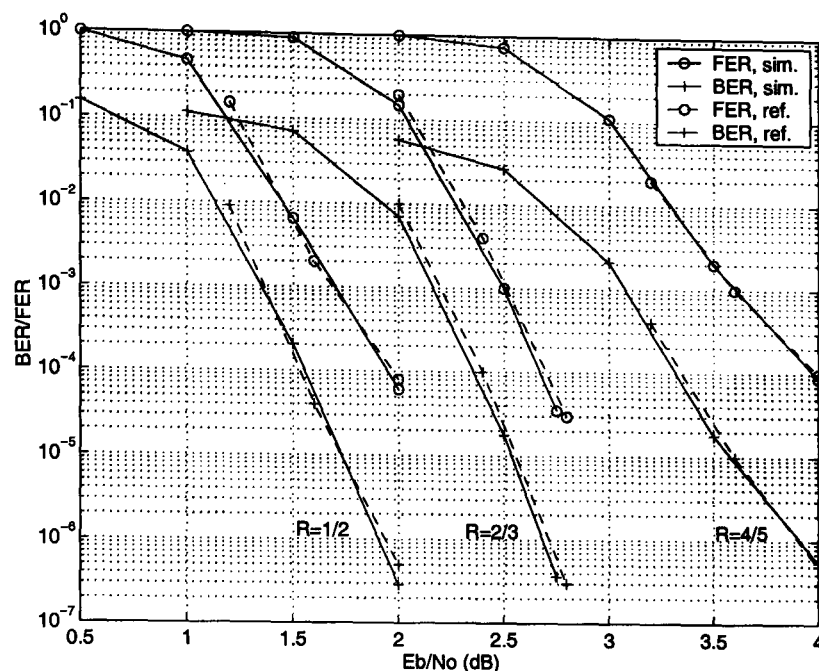


Figure B.15: BER/FER comparison with *Yu* [12]. Duo-binary turbo code, such as in the DVB-RCS standard, MPEG frame size, i.e. 1504 bits, AWGN channel, Max-Log-MAP algorithms, 8 decoding iterations and different coding rates. Solid lines-simulation and dashed lines-from reference.

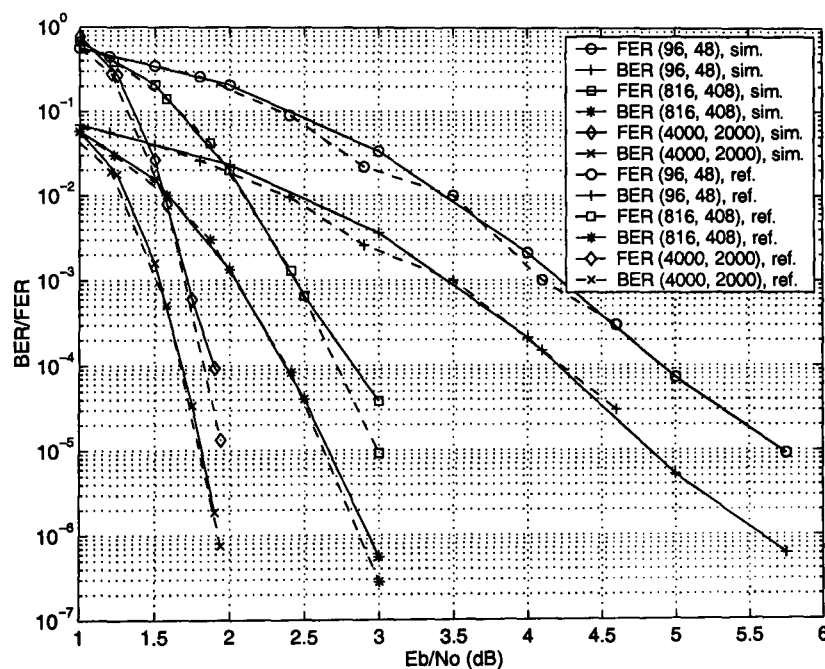


Figure B.16: BER/FER comparison with *MacKay* [13]. Different regular LDPC codes, coding rate $R = 1/2$, AWGN channel, SPA decoding algorithm from *Gallager's* approach and maximum 200 decoding iterations. Solid lines-simulation and dashed lines-from reference (with variable maximum number of decoding iterations).

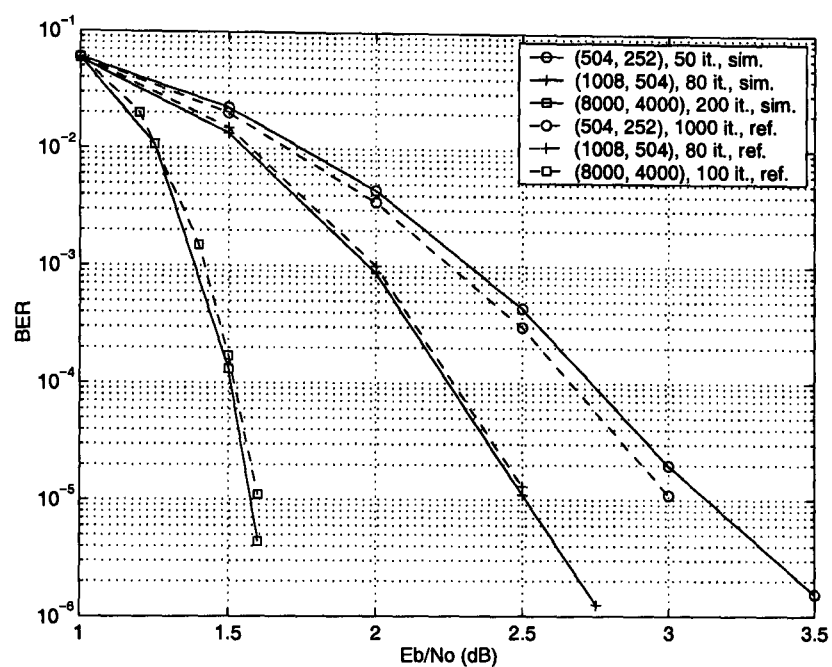


Figure B.17: BER comparison with *Fosserier-1* [14], *Eleftheriou* [15] and *Fosserier-2* [16]. Different regular LDPC codes and maximum number of decoding iterations, coding rate $R = 1/2$, AWGN channel, SPA decoding algorithm from *Gallager's* approach. Solid lines-simulation and dashed lines-from reference.

Appendix C

The Effect of Different Parameters to the Simulated Turbo Code Performance

The turbo code performance is affected by different parameters. In the following, computer simulation results are reported for different turbo code configurations. This is inspired by the work from [8]. In more detail, it is shown

- Effect of number of decoding iterations, Figs. C.1-C.3.
- Effect of frame (or interleaver) size, Figs. C.4, C.5.
- Effect of memory order, Fig. C.6.
- Effect of the type of interleaver, Figs. C.7, C.8.
- Effect of puncturing, Figs. C.9, C.10.
- Effect of channel type, Fig. C.11.
- Effect of decoding algorithm, Figs. C.12, C.13.

From Figs. C.1-C.3 it is noticed that the BER performance is improved by increasing the number of decoding iterations. The improvement is smaller, e.g. less than 0.1

dB, when the number of decoding iterations is already high. Usually, eight decoding iterations are enough to cope with decoding complexity issues.

In Figs. C.4, C.5 it is shown that the BER performance of the turbo code is improved by increasing the frame (or interleaver) size. In particular, for very large frames, e.g. greater than 10000 bits, the channel capacity limit can be approached by a few tenths of dB. For frame sizes of hundreds of bits, e.g. 100, the resulting BER performance is comparable to that of a convolutional code with either 128 or 256 states, so that both codes require the same amount of decoding complexity [8].

Fig. C.6 depicts the fact that increasing the memory order (otherwise, the number of states) of the turbo encoder, better BER performance is obtained. This is true especially in the high SNR region. Due to decoding complexity limitations, turbo encoders with up to 16-states are considered in practice.

In Figs. C.7, C.8 it is illustrated the effect of the 3GPP interleaver [62] to the turbo code performance compared to the corresponding performance with a pseudo-random interleaver. The performance improvement is approximately 0.25 dB at BER of 10^{-6} . This can be explained because of better spread of the information bits before entering to the second component encoder, which increases the minimum free distance of the code.

From Figs. C.9, C.10 it is shown that puncturing a rate $1/3$ turbo code to a rate $1/2$ turbo code, it occurs approximately 0.5 dB degradation in the BER performance. For higher coding rates, the performance degradation is greater, e.g. see Fig. C.10. For instance, puncturing a rate $1/2$ turbo code to a rate $6/7$ turbo code, the BER performance degradation is approximately 3 dB.

In Fig. C.11 it is shown that in an uncorrelated Rayleigh fading channel a rate $1/3$ turbo code requires approximately 2 dB more to achieve the same BER performance, as in the case of the AWGN channel. This is when no CSI is available at the receiver. In the other case when CSI is available at the receiver, the same turbo code requires approximately 1 dB more than in the AWGN channel case [1]. Moreover, the turbo code BER performance in an uncorrelated Rician fading channel, with different Rice factors K , is better than in an uncorrelated Rayleigh fading channel and at the same

time inferior to the performance in the AWGN channel.

From Figs. C.12, C.13 it is shown that the Max-Log-MAP algorithm is approximately 0.4 dB inferior to the Log-MAP algorithm at BER of 10^{-4} , while SOVA is approximately 0.7 dB inferior to the Log-MAP algorithm at the same BER value. This is when assuming a 16-states turbo code with coding rate equal to $R = 1/3$. For a 4-states turbo code with coding rate equal to $R = 1/2$, the gap between the SOVA and Log-MAP algorithms, in terms of BER performance degradation, is reduced to 0.4 dB and the related gap between the Max-Log-MAP and Log-MAP algorithms is reduced to 0.1 dB.

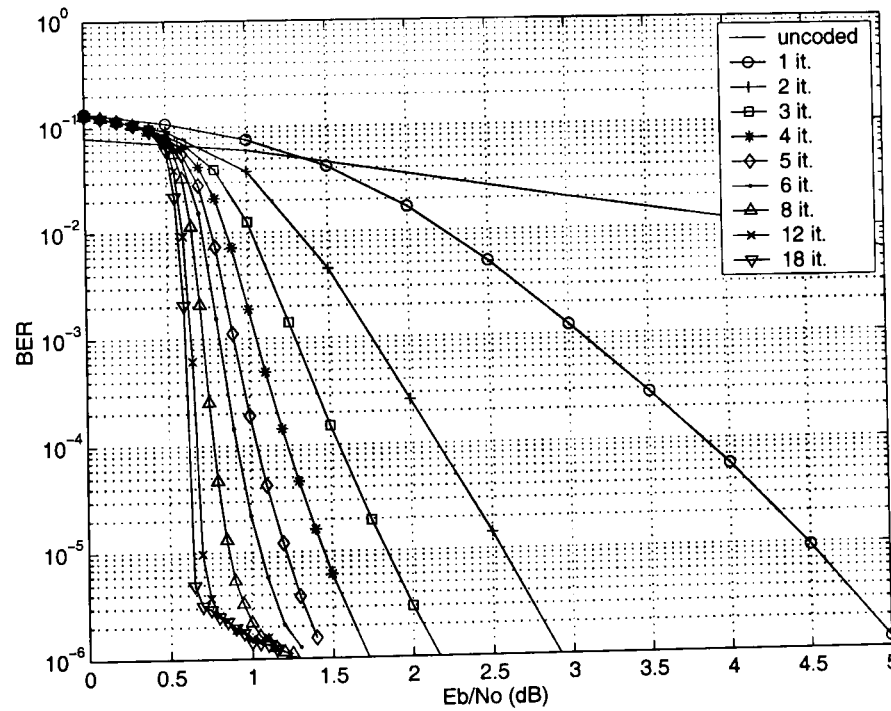


Figure C.1: Effect of *number of decoding iterations* to the BER performance, turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 65536 bits frame size, AWGN channel and Log-MAP algorithm.

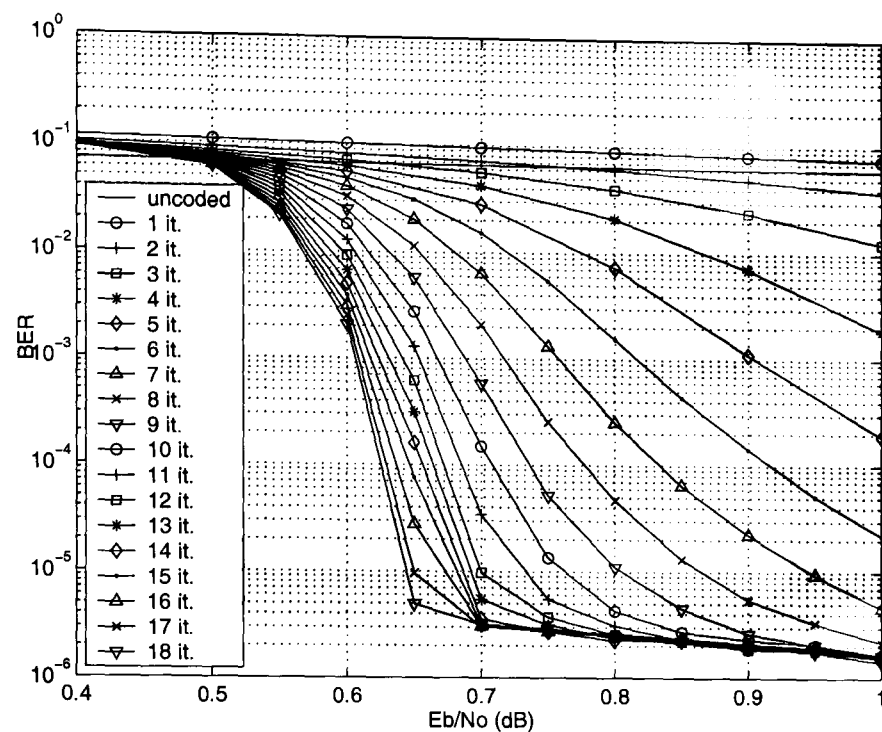


Figure C.2: Effect of *number of decoding iterations* to the BER performance, turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/2$, 65536 bits frame size, AWGN channel and Log-MAP algorithm (*zoom*).

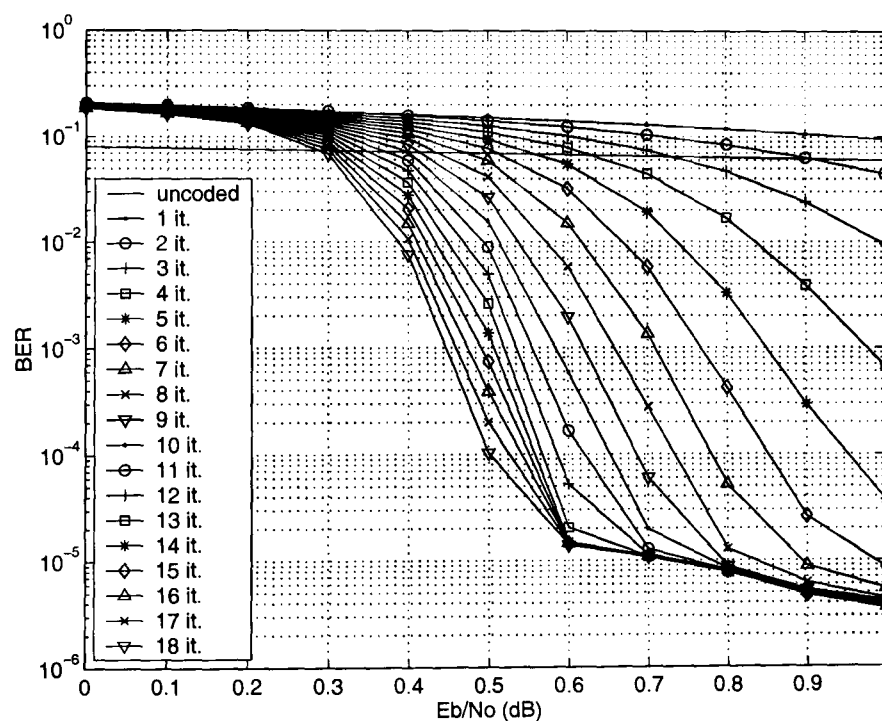


Figure C.3: Effect of *number of decoding iterations* to the BER performance, turbo code generator polynomials $(1, 21/37)_o$, i.e. 16-states, coding rate $R = 1/3$, 65536 bits frame size, AWGN channel and *norm1* SOVA algorithm.

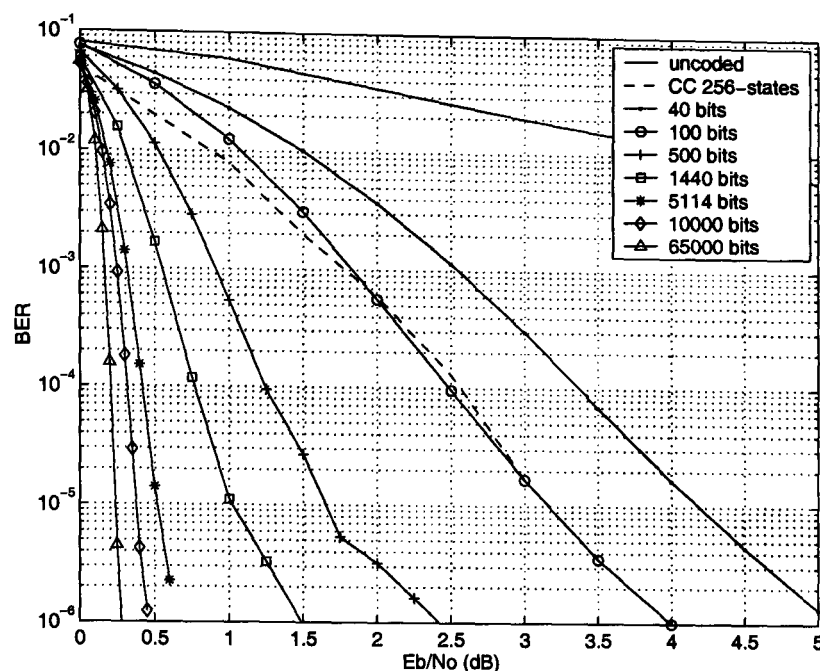


Figure C.4: Effect of *frame (or interleaver) size* to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, AWGN channel, Log-MAP algorithm and 8 decoding iterations. BER performance with 256-states convolutional code is also shown from [8] (dashed lines).

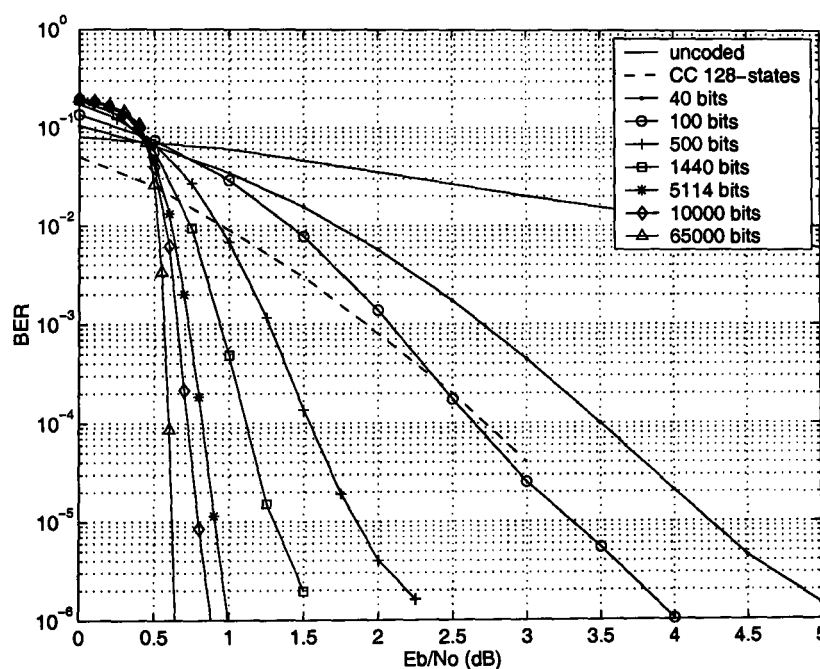


Figure C.5: Effect of *frame (or interleaver) size* to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, AWGN channel, Max-Log-MAP algorithm and 8 decoding iterations. BER performance with 128-states convolutional code is also shown from [8] (dashed lines).

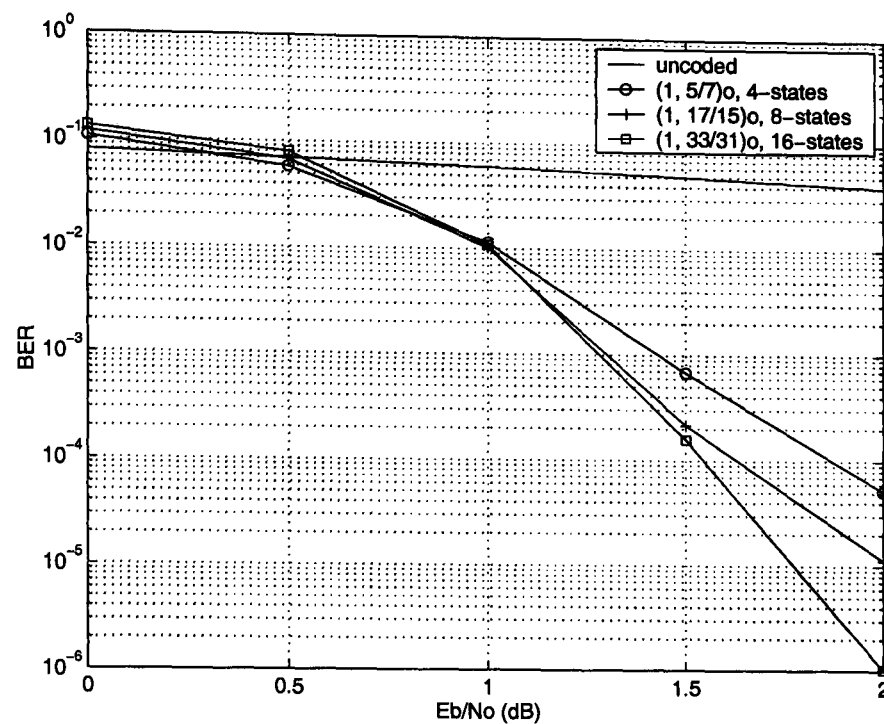


Figure C.6: Effect of *memory order* to the BER performance, coding rate $R = 1/2$, 1000 bits frame size, AWGN channel, Log-MAP algorithm and 8 decoding iterations.

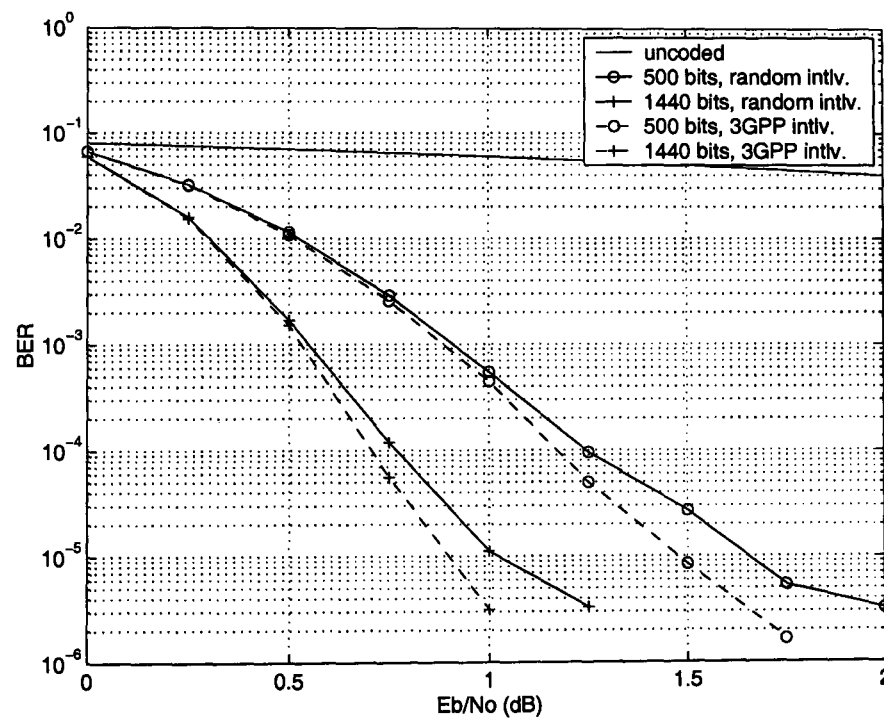


Figure C.7: Effect of the *type of interleaver* to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, either 500 or 1440 bits frame size, AWGN channel, Log-MAP algorithm and 8 decoding iterations. Solid lines-random interleaver, dashed lines-3GPP interleaver.

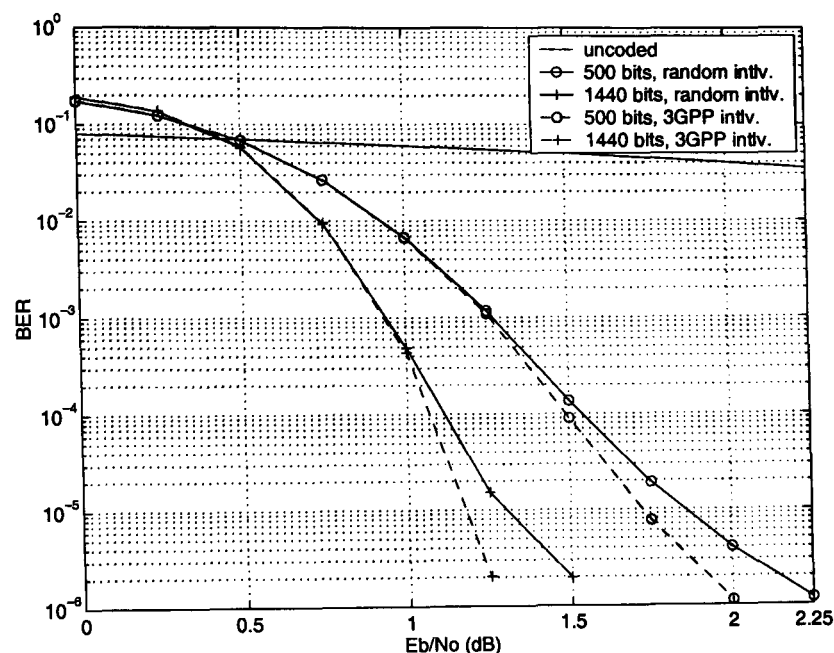


Figure C.8: Effect of the *type of interleaver* to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, either 500 or 1440 bits frame size, AWGN channel, Max-Log-MAP algorithm and 8 decoding iterations. Solid lines-random interleaver, dashed lines-3GPP interleaver.

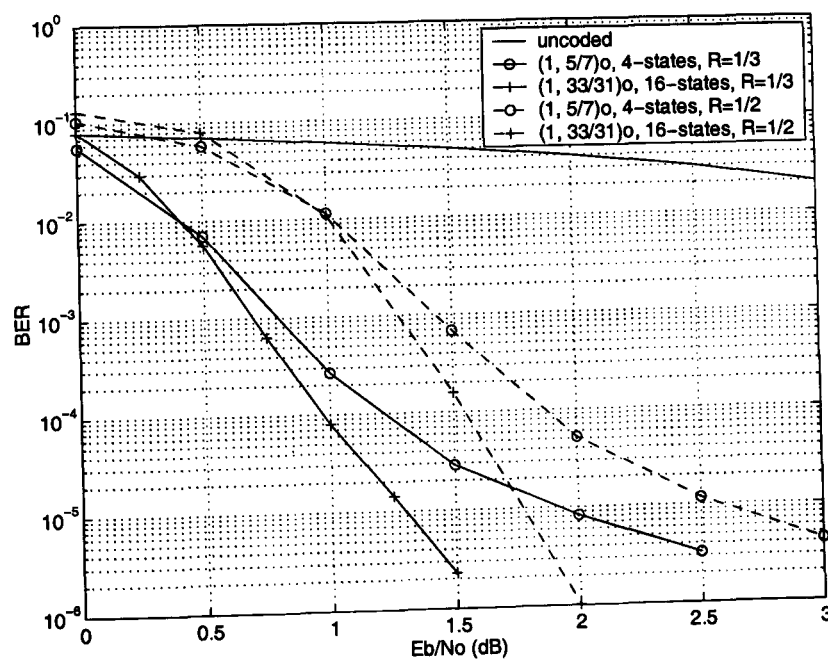


Figure C.9: Effect of *puncturing* to the BER performance, different turbo code generator polynomials, 1000 bits frame size, AWGN channel, Log-MAP algorithm and 8 decoding iterations. Solid lines-no puncturing, i.e. coding rate $R = 1/3$, dashed lines-with puncturing, i.e. coding rate $R = 1/2$.

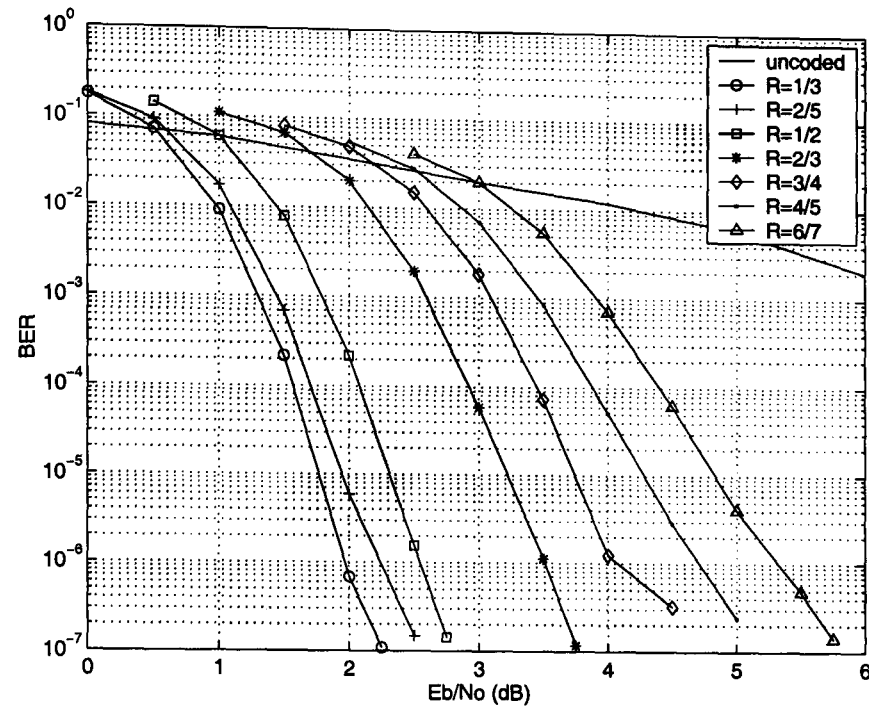


Figure C.10: Effect of *puncturing* to the BER performance, DVB-RCS turbo encoder, i.e. 8-states, ATM frame size, i.e. 424 bits, AWGN channel, Max-Log-MAP algorithm and 8 decoding iterations. Coding rates $R = 1/3, 1/2$ are with no puncturing.

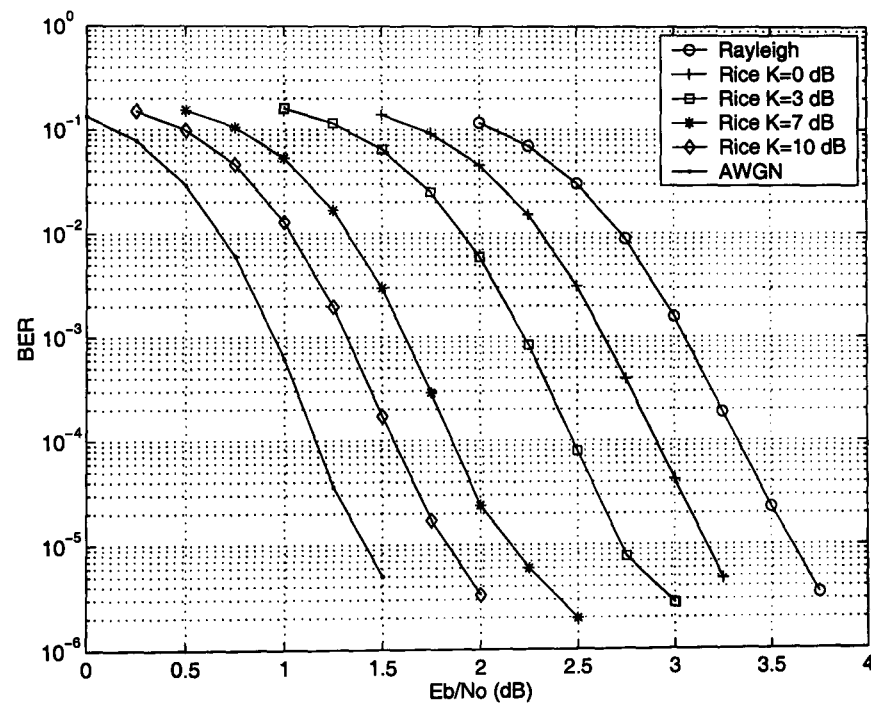


Figure C.11: Effect of *channel type* to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 1000 bits frame size, *norm2* SOVA algorithm and 8 decoding iterations.

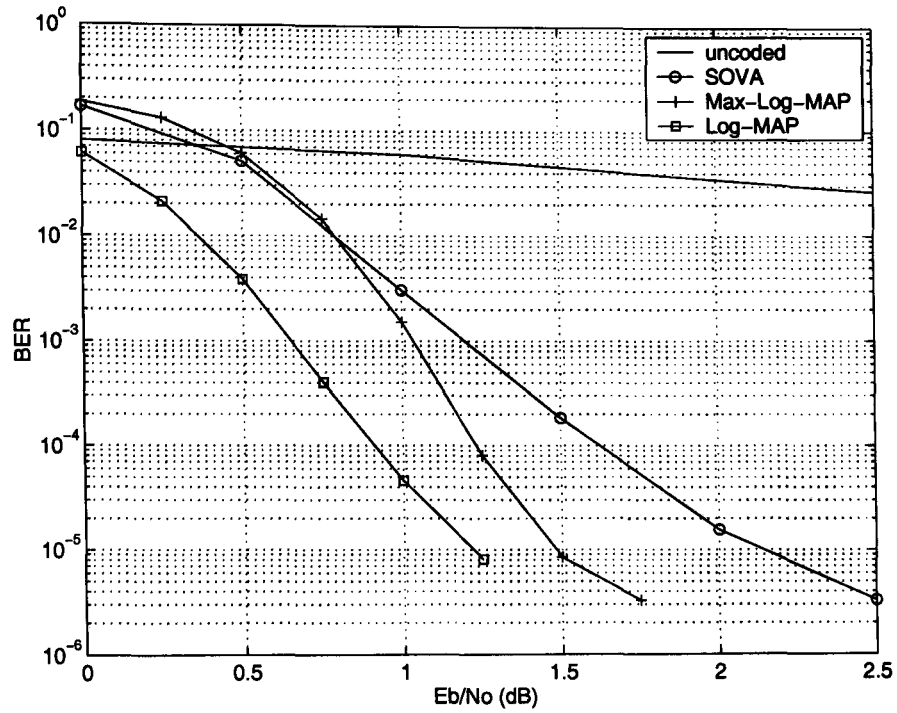


Figure C.12: Effect of *decoding algorithm* to the BER performance, turbo code generator polynomials $(1, 15/13)_o$, i.e. 8-states, coding rate $R = 1/3$, 1000 bits frame size, AWGN channel and 8 decoding iterations.

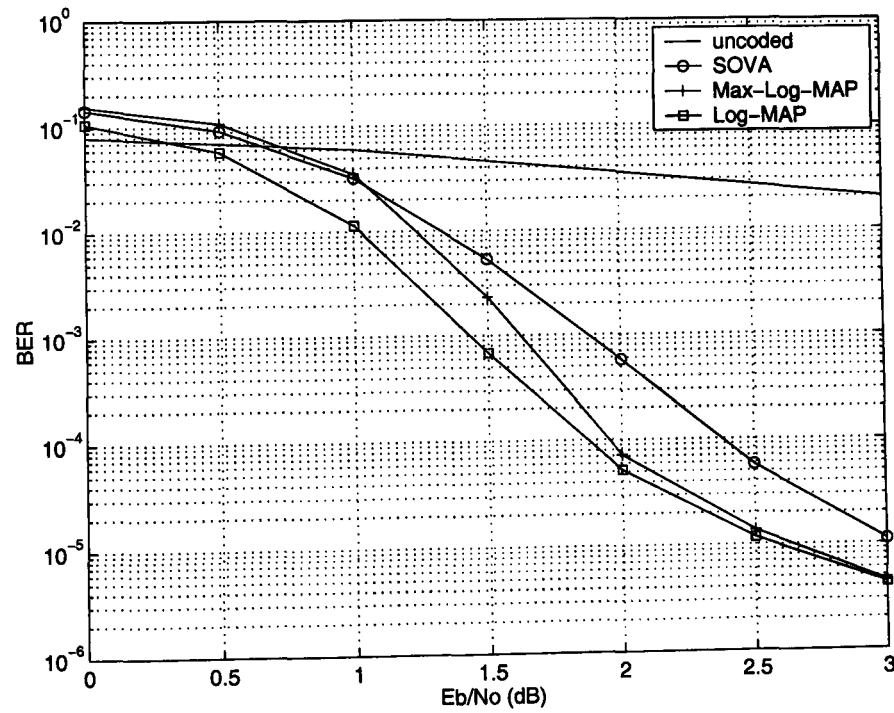


Figure C.13: Effect of *decoding algorithm* to the BER performance, turbo code generator polynomials $(1, 5/7)_o$, i.e. 4-states, coding rate $R = 1/2$, 1000 bits frame size, AWGN channel and 8 decoding iterations.

References

- [1] B. Vucetic and J. Yuan, *Turbo Codes Principles and Applications*. Boston/Dordrecht/ London: Kluwer Academic Publishers, 2000.
- [2] Information processing group, Jet Propulsion Laboratory (JPL), California Institute of Technology (CALTECH), NASA. [Online]. Available: <http://www331.jpl.nasa.gov/public/TurboPerf.html>
- [3] C. Douillard and C. Berrou, "Turbo codes with rate- $m/(m+1)$ constituent convolutional codes," *IEEE Trans. Commun.*, vol. 53, no. 10, pp. 1630–1638, Oct. 2005.
- [4] M. R. Soleymani, Y. Gao, and U. Vilaipornsawai, *Turbo Coding for Satellite and Wireless Communications*. Boston/ Dordrecht/ London: Kluwer Academic Publishers, 2002.
- [5] L. Lin and R. Cheng, "Improvements in SOVA-based decoding for turbo codes," in *Proc. IEEE Inter. Conf. Commun. (ICC)*, Montreal, Canada, June 1997, pp. 1473–1478.
- [6] C. Berrou, A. Glavieux, and P. Thitimajhima, "Near Shannon limit error correcting coding and decoding: Turbo codes," in *Proc. IEEE Inter. Conf. Commun. (ICC)*, Geneva, Switzerland, May 1993, pp. 1064–1070.
- [7] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the Log domain," in *Proc. IEEE Inter. Conf. Commun. (ICC)*, Seattle, WA, June 1995, pp. 1009–1013.

-
- [8] J. P. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: an overview," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2208–2233, Nov. 2000.
 - [9] M. C. Valenti and J. Sun, "The UMTS turbo code and an efficient decoder implementation suitable for software-defined radios," *Inter. Journal of Wireless Inform. Networks*, vol. 8, no. 4, pp. 203–215, Oct. 2001.
 - [10] J. Hagenauer, P. Robertson, and L. Papke, "Iterative ("Turbo") decoding of systematic convolutional codes with the MAP and SOVA algorithms," in *Proc. ITG-Fachtagung 'Codierung'*, Munich, Germany, Oct. 1994, pp. 21–29.
 - [11] Y. O. C. Mouhamedou, P. Guinand, and P. Kabal, "Enhanced Max-Log-APP and enhanced Log-APP decoding for DVB-RCS," in *Proc. 3rd Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 2003, pp. 259–262.
 - [12] J. Yu, M.-L. Boucheret, R. Vallet, and G. Mesnager, "Interleaver parameter-selecting strategy for DVB-RCS turbo codes," *IEE Electron. Lett.*, vol. 38, no. 15, pp. 805–807, July 2002.
 - [13] D. J. C. MacKay. (2005) Online database of low-density parity-check codes. [Online]. Available: <http://wol.ra.phy.cam.uk/mackay/codes/data.html>
 - [14] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity-check codes," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 406–414, Mar. 2002.
 - [15] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Proc. IEEE Globecom*, San Antonio, USA, Nov. 2001, pp. 1036–1036E.
 - [16] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.
 - [17] E. Guizzo, "Closing in on the perfect code," *IEEE Spectrum*, vol. 41, no. 3, pp. 36–42, Mar. 2004.

-
- [18] A. Burr, "Turbo-codes: the ultimate error control codes?" *IEE Electron. and Commun. Eng. Journ.*, vol. 13, no. 4, pp. 155–165, Aug. 2001.
- [19] S. Benedetto, G. Montorsi, and D. Divsalar, "Concatenated convolutional codes with interleavers," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 102–109, Aug. 2003.
- [20] S. Haykin, M. Shellathurai, and T. Willink, "Turbo-MIMO for wireless communications," *IEEE Commun. Mag.*, vol. 42, no. 10, pp. 48–53, Oct. 2004.
- [21] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
- [22] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *IEE Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.
- [23] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
- [24] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, Sept. 1981.
- [25] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
- [26] E. Biglieri, "Digital transmission in the 21st century: conflating modulation and coding," *IEEE Commun. Mag.*, vol. 40, 50th anniversary issue, no. 5, pp. 128–137, May 2002.
- [27] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. 2nd Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 2000, pp. 1–8.
- [28] B. G. Evans, "Role of satellites in mobile/wireless systems," in *Proc. 15th IEEE Inter. Symp. Pers. Ind. and Mob. Rad. Commun. (PIMRC)*, Barcelona, Spain, Sept. 2004, pp. 2055–2060.

-
- [29] B. Evans, M. Werner, E. Lutz, M. Bousquet, G. E. Corazza, G. Maral, R. Rumeau, and E. Ferro, "Integration of satellite and terrestrial systems in future multimedia communications," *IEEE Wireless Commun. Mag.*, vol. 12, no. 4, pp. 72–80, Oct. 2005.
- [30] L. Henden *et al.* (2005, Jan.) Broadcast and multicast - a vision on their role in future broadband access networks. [Online]. Available: <http://ist-maestro.dyndns.org/MAESTRO/index.htm>
- [31] C. Berrou, "The ten-year-old turbo codes are entering into service," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 110–116, Aug. 2003.
- [32] S. Benedetto, R. Garelo, G. Montorsi, C. Berrou, C. Douillard, D. Giancristofaro, A. Ginesi, L. Giugno, and M. Luise, "MHOMS: High-speed ACM modem for satellite applications," *IEEE Wireless Commun. Mag.*, vol. 12, no. 2, pp. 66–77, Apr. 2005.
- [33] S. Papaharalabos, "Turbo coding for high data rate downlink in S-UMTS air interface," Master's thesis, University of Surrey, Guildford, UK, 2002.
- [34] SatNEx FP6 EU Project. [Online]. Available: <http://www.satnex.org>
- [35] *IEEE Commun. Mag.*, 50th anniversary issue, vol. 40, no. 5, May 2002.
- [36] MAESTRO FP6 EU Project. [Online]. Available: <http://www.ist-maestro.dyndns.org>
- [37] SATIN FP5 EU Project. [Online]. Available: <http://www.ist-satin.org>
- [38] MODIS FP5 EU Project. [Online]. Available: <http://www.ist-modis.org>
- [39] Mobile broadcasting corporation (MBCO) in Japan. [Online]. Available: <http://www.mbco.co.jp/english>
- [40] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1983.
- [41] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, p. 4, pp. 37–47, 1955.

-
- [42] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffers*, vol. 2, pp. 147–156, 1959.
- [43] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting group code," *Inf. Control*, vol. 3, pp. 68–79, Mar. 1960.
- [44] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, pp. 300–304, June 1960.
- [45] G. D. Forney, *Concatenated codes*. MA: MIT Press, 1966.
- [46] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *Proc. IEEE Globecom*, San Francisco, USA, Nov. 1994, pp. 339–343.
- [47] S. A. Hirst, B. Honary, and G. Markarian, "Fast Chase algorithm with an application in turbo decoding," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1693–1699, Oct. 2001.
- [48] P. Robertson and T. Worz, "Coded modulation scheme employing turbo codes," *IEE Electron. Lett.*, vol. 31, no. 18, pp. 1546–1547, Aug. 1995.
- [49] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenations of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.
- [50] —, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *Euro. Trans. Telecommun.*, vol. 9, no. 2, pp. 909–926, Mar. 1998.
- [51] C. Berrou and M. Jézéquel, "Non-binary convolutional codes for turbo coding," *IEE Electron. Lett.*, vol. 35, no. 1, pp. 39–40, Jan. 1999.
- [52] K. Andrews, S. Dolinar, D. Divsalar, and J. Thorpe. Design of low-density parity-check (LDPC) codes for deep-space applications. The Interplanetary Network Progress Report 42-159. Jet Propulsion Laboratory, Pasadena, California, USA, pp. 1-14, Nov. 15, 2004. [Online]. Available: http://ipnpr.jpl.nasa.gov/tmo/progress_report/42-159/159K.pdf

-
- [53] E. Eleftheriou, S. Ölcer, and H. Sadjadpour, "Application of capacity approaching coding techniques to digital subscriber lines," *IEEE Commun. Mag.*, vol. 42, no. 4, pp. 88–94, Apr. 2004.
- [54] A. Dholakia, E. Eleftheriou, T. Mittelholzer, and M. P. C. Fossorier, "Capacity-approaching codes: can they be applied to the magnetic recording channel?" *IEEE Commun. Mag.*, vol. 42, no. 2, pp. 122–130, Feb. 2004.
- [55] B. Sklar, *Digital communications fundamentals and applications*, 2nd ed. Prentice Hall PTR, 2001.
- [56] G. Colavolpe, G. Ferrari, and R. Raheli, "Extrinsic information in iterative decoding: a unified view," *IEEE Trans. Commun.*, vol. 49, no. 12, pp. 2088–2094, Dec. 2001.
- [57] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Globecom*, Dallas, USA, Nov. 1989, pp. 1680–1686.
- [58] G. Battail, "Ponderation des symboles decodes par l' algorithme de Viterbi," *Ann. Telecommun.*, vol. 42, no. 1-2, pp. 31–38, Jan. 1987, (in French).
- [59] L. Bahl, J. Cocke, F. Jeinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
- [60] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 260–264, Feb. 1998.
- [61] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and Max-Log-MAP decodings," *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 137–139, May 1998.
- [62] *Technical Specification Group, Radio Access Network; Multiplexing and Channel Coding (FDD), Release 1999*, 3GPP, TS 25.212 Std. V3.11.0, 2002-09.

-
- [63] Y. Wu, B. D. Woerner, and W. J. Ebel, "A simple stopping criterion for turbo decoding," *IEEE Commun. Lett.*, vol. 4, no. 8, pp. 258–260, Aug. 2000.
- [64] S. T. Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.
- [65] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.
- [66] S. Riedel, "MAP decoding of convolutional codes using reciprocal dual codes," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 1176–1187, May 1998.
- [67] W. E. Ryan, *An introduction to LDPC codes*, ser. Handbook for Coding and Signal Processing for Recording Systems. New York: CRC Press, 2004.
- [68] *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications*, ETSI EN 302 307 Std. v1.1.1, 2004.
- [69] *Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)*, ETSI EN 302 304 Std. v1.1.1, 2004.
- [70] *Technical Specification Group, Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and Codecs, (Release 6)*, ETSI TS 26.346 Std. v6.2.0, 2005-09.
- [71] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C*. UK: Cambridge University Press, 1992.
- [72] Z. Wang and K. K. Parhi, "High performance, high throughput turbo/SOVA decoder design," *IEEE Trans. Commun.*, vol. 51, no. 4, pp. 570–579, Apr. 2003.
- [73] J. Chen, M. P. C. Fossorier, S. Lin, and C. Xu, "Bi-directional SOVA decoding for turbo codes," *IEEE Commun. Lett.*, vol. 4, no. 12, pp. 405–407, Dec. 2000.

-
- [74] W. Feng and B. Vucetic, "A list bidirectional soft output decoder of turbo codes," in *Proc. 1st Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 1997, pp. 288–292.
- [75] D. Wang and H. Kobayashi, "High-performance SOVA decoding for turbo codes over cdma 2000 mobile radio," in *Proc. IEEE Milit. Commun. Conf. (MILCOM)*, Los Angeles, USA, Oct. 2000, pp. 189–193.
- [76] Y. Fahmy, H. A. G. A. Kader, and M. M. S. El-Soudani, "On the use of SOVA for iterative decoding," in *Proc. IEEE Medit. Electr. Conf. (MELECON)*, Cairo, Egypt, May 2002, pp. 168–172.
- [77] C. H. Wang, W. T. Wang, and C. C. Chao, "A unified structure of trellis-based soft-output decoding algorithms for turbo codes," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1355–1366, Aug. 2004.
- [78] W. Lei and K. Jingming, "An improved design and SOVA algorithm for serial concatenated convolutional code," in *Proc. IEEE Inter. Conf. Commun. Techn. (ICCT)*, Beijing, China, Aug. 2000, pp. 327–330.
- [79] Y. C. Chang and J. K. Lain, "Improved decoding with the Bi-directional SOVA for turbo codes," in *Proc. IEEE Vec. Tech. Conf. (VTC) Spring*, Stockholm, Sweden, May 2005.
- [80] L. Papke, P. Robertson, and E. Villebrun, "Improved decoding with SOVA in parallel concatenated (turbo-code) scheme," in *Proc. IEEE Inter. Conf. Commun. (ICC)*, Dallas, USA, June 1996, pp. 102–106.
- [81] Z. Blazeck and V. K. Bhargava, "A DSP-based implementation of a turbo decoder," in *Proc. IEEE Globecom*, Sydney, Australia, Nov. 1998, pp. 3201–3205.
- [82] R. A. Stirling-Gallacher, "Performance of sub-optimal normalization schemes for a turbo decoder using the soft output Viterbi algorithm," in *Proc. 11th IEEE Inter. Symp. Pers. Ind. and Mob. Rad. Commun. (PIMRC)*, London, UK, Sept. 2000, pp. 888–892.

-
- [83] T. W. Kwon, D. W. Kim, W. T. Kim, E. K. Joo, J. R. Choi, P. Choi, J. J. Kong, S. H. Choi, W. H. Chung, and K. W. Lee, "A modified two-step SOVA-based turbo decoder for low power and high performance," in *Proc. IEEE Tencon*, Cheju, Korea, Sept. 1999, pp. 297–300.
- [84] D. W., K. T., W. Kwon, J. R. Choi, and J. J. Kong, "A modified two-step SOVA-based turbo decoder with a fixed scaling factor," in *Proc. IEEE Inter. Symp. Circuits and Systems*, Geneva, Switzerland, May 2000, pp. 37–40.
- [85] Z. Wang, H. Suzuki, , and K. K. Parhi, "Efficient approaches to improving performance of VLSI SOVA-based turbo decoders," in *Proc. IEEE Inter. Symp. Circuits and Systems*, Geneva, Switzerland, May 2000, pp. 287–290.
- [86] C. X. Huang and A. Ghrayeb, "An improved SOVA algorithm for turbo codes over AWGN and fading channels," in *Proc. 15th IEEE Inter. Symp. Pers. Ind. and Mob. Rad. Commun. (PIMRC)*, Barcelona, Spain, Sept. 2004, pp. 1121–1125.
- [87] —, "Improved SOVA and APP decoding algorithms for serial concatenated codes," in *Proc. IEEE Globecom*, Dallas, USA, Nov. 2004, pp. 189–193.
- [88] S. Papaharalabos, P. Sweeney, and B. G. Evans, "Turbo coding performance evaluation using an improved iterative SOVA decoder," in *Proc. AIAA Intern. Commun. Satel. Syst. Conf. (ICSSC)*, Monterey, California, May 2004, paper 3108.
- [89] —, "A new method of improving SOVA turbo decoding for AWGN, Rayleigh and Rician fading channels," in *Proc. IEEE Vec. Tech. Conf. (VTC) Spring*, Milan, Italy, May 2004, pp. 2862–2866.
- [90] —, "Modification of branch metric calculation to improve iterative SOVA decoding of turbo codes," *IEE Electron. Lett.*, vol. 39, no. 19, pp. 1391–1392, Sept. 2003.
- [91] J. Vogt and A. Finger, "Improving the Max-Log-MAP turbo decoder," *IEE Electron. Lett.*, vol. 36, no. 23, pp. 1937–1939, Nov. 2000.

-
- [92] J. Yuan, W. Feng, and B. Vucetic, "Performance of parallel and serial concatenated codes on fading channels," *IEEE Trans. Commun.*, vol. 50, no. 10, pp. 1600–1608, Oct. 2002.
- [93] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *IEE Electron. Lett.*, vol. 34, no. 16, pp. 1577–1578, Aug. 1998.
- [94] B. Classon, K. Blankenship, and V. Desai, "Turbo decoding with the Constant-Log-MAP algorithm," in *Proc. 2nd Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 2000, pp. 467–470.
- [95] H. Claussen, H. R. Karimi, and B. Mulgrew, "Improved Max-Log-MAP turbo decoding using maximum mutual information combining," in *Proc. 14th IEEE Inter. Symp. Pers. Ind. and Mob. Rad. Commun. (PIMRC)*, Beijing, China, Sept. 2003, pp. 424–428.
- [96] N. Y. Yu, M. G. Kim, and Y. S. Kim, "Two modified Max-Log-MAP algorithms for turbo decoder enhancement," in *Proc. 3rd Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 2003, pp. 207–210.
- [97] *TSG-C, Physical layer standard for CDMA2000 spread spectrum systems*, 3GPP2 Std. Release C, May 2002.
- [98] K. Gracie, S. Crozier, and P. Guinand, "Performance of an MLSE-based early stopping technique for turbo codes," in *Proc. IEEE Vec. Tech. Conf. (VTC) Fall*, Los Angeles, USA, Sept. 2004, pp. 2287–2291.
- [99] S. Park, "Combined Max-Log-MAP and Log-MAP of turbo codes," *IEE Electron. Lett.*, vol. 40, no. 4, pp. 251–252, Feb. 2004.
- [100] J. Tan and G. L. Stuber, "New SISO decoding algorithms," *IEEE Trans. Commun.*, vol. 51, no. 6, pp. 845–848, June 2003.
- [101] S. Papaharalabos, P. Sweeney, and B. G. Evans, "SISO algorithms based on combined max/max* operations for turbo decoding," *IEE Electron. Lett.*, vol. 41, no. 3, pp. 142–143, Feb. 2005.

-
- [102] —, “Improved SISO decoding algorithms for Max-Log-MAP and Log-MAP iterative decoding,” *submitted for publication*, 2005.
- [103] E. Boutillon, W. J. Gross, and P. Gulak, “VLSI architectures for the MAP algorithm,” *IEEE Trans. Commun.*, vol. 51, no. 2, pp. 175–185, Feb. 2003.
- [104] C. Douillard, M. Jézéquel, C. Berrou, N. Brengarth, J. Tusch, and N. Pham, “The turbo code standard for DVB-RCS,” in *Proc. 2nd Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 2000, pp. 535–538.
- [105] *Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems*, ETSI EN 301 790 Std. v1.3.1, 2003.
- [106] *Digital Video Broadcasting (DVB); Interaction channel for digital terrestrial television incorporating multiple access OFDM*, ETSI EN 301 958 Std. v1.1.1, 2001.
- [107] R. Crespo and C. Berrou, “A flexible computer-based platform for turbo coding and decoding (Turbo 2000),” in *Proc. 3rd Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 2003, pp. 583–586.
- [108] Y. Du and M. R. Soleymani, “VLSI implementation of DVB/RCS turbo code,” in *Proc. IEEE Canad. Conf. on Elect. and Comp. Eng. (CCECE) 2003*, vol. 3, Montreal, Canada, May 2003, pp. 1581–1584.
- [109] P. Sadeghi and M. R. Soleymani, “Multi-channel processing of DVB/RCS turbo codes,” in *Proc. IEEE Canad. Conf. on Elect. and Comp. Eng. (CCECE) 2003*, vol. 3, Montreal, Canada, May 2003, pp. 1601–1604.
- [110] A.-L. Philippot, J. Yu, M.-L. Boucheret, C. Morlet, and C. Bazile, “Turbo coding for DVB-RCS with variable block sizes,” in *Proc. 3rd Inter. Symp. on Turbo Codes and Relat. Topics*, Brest, France, Sept. 2003, pp. 523–526.
- [111] Y. Ould-Cheikh-Mouhamedou, S. Crozier, and P. Kabal, “Distance measurement method for double binary turbo codes and a new interleaver design for DVB-RCS,” in *Proc. IEEE Globecom*, vol. 1, Dallas, USA, Nov./Dec. 2004, pp. 172–178.

-
- [112] S. Papaharalabos, P. Sweeney, and B. G. Evans, "Max/max* operation replacement to improve the DVB-RCS turbo decoder," in *Proc. AIAA Intern. Commun. Satel. Syst. Conf. (ICSSC)*, Rome, Italy, Sept. 2005.
- [113] —, "Efficient Constant Log-MAP decoding for duo-binary turbo codes," *submitted for publication*, 2005.
- [114] *IEEE Trans. Inform. Theory*, Special issue: codes on graphs and iterative algorithms, vol. 47, no. 2, Feb. 2001.
- [115] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
- [116] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.
- [117] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.
- [118] S. Papaharalabos, G. Albertazzi, P. Sweeney, B. G. Evans, A. Vanelli-Coralli, and G. E. Corazza, "Performance evaluation of a modified sum-product decoding algorithm for LDPC codes," in *Proc. IEEE Inter. Works. on Satel. and Space Commun. (IWSSC)*, Siena, Italy, Sept. 2005.
- [119] —, "Modified sum-product decoding algorithm for LDPC codes," *submitted for publication*, 2005.
- [120] M. A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L. M. Davis, G. Woodward, C. Nicol, and R.-H. Yan, "A unified turbo/Viterbi channel decoder for 3GPP mobile wireless in 0.18 μm in CMOS," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, Nov. 2002.
- [121] Flarion Technologies. [Online]. Available: <http://www.flarion.com>

-
- [122] Digital Fountain. [Online]. Available: <http://www.dfountain.com>
- [123] K. Fagervik and T. G. Jeans, "Low complexity bit by bit soft output demodulator," *IEE Electron. Lett.*, vol. 32, no. 11, pp. 985–987, May 1996.
- [124] S. LeGoff, A. Glavieux, and C. Berrou, "Turbo codes and high spectral efficiency modulation," in *Proc. IEEE Inter. Conf. Commun. (ICC)*, New Orleans, USA, May 1994, pp. 645–649.