UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science
Southampton SO17 1BJ

Channel Coding and Space-Time Coding For Wireless Channels

by

LIEW Tong Hooi
B.Eng

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy at the University of Southampton

November 2000

Supervisor: Professor Lajos Hanzo
Dipl Ing, MSc, PhD, SMIEEE
Chair of Telecommunications
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ
United Kingdom

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Channel Coding and Space-Time Coding For Wireless Channels
by LIEW Tong Hooi

This thesis is based on the research of channel coding. First, we give an introduction to the family of conventional Bose-Chaudhuri-Hocquenghem (BCH) codes, characterising the performance of various BCH codes. Then we further our research into turbo codes employing BCH codes as the component codes. Various decoding algorithms are presented for turbo codes. This is followed by our simulation results, studying the effects of various parameters, affecting the performance of turbo codes.

Subsequently, another form of non-binary block codes, referred to as Redundant Residue Number System (RRNS) codes, which exhibit identical distance properties to the well-established Reed-Solomon (RS) codes, are investigated. Different bit-to-symbol mapping schemes are proposed, which result in systematic and non-systematic RRNS codes. An RRNS decoder is proposed, which accepts soft inputs and provide soft outputs. This facilitates the iterative decoding of turbo RRNS codes.

Our investigations into channel coding are also expanded to study space-time codes, which are constituted by jointly designed channel coding, modulation, transmit diversity and optional receiver diversity schemes. Specifically, combined space-time block codes and different channel codecs studied. Various simulation results are presented for space-time block codes using no channel coding. This is followed by the investigations of the bit-to-symbol mapping of the binary channel coded bits to higher modulation constellations. Finally, the performance of various channel codecs is compared by considering their estimated complexity in conjunction with space-time block coding.

In the last chapter, space-time trellis codes are then compared to the class of space-time block codes in conjunction with a range of channel codes over wideband channels. Various factors affecting the performance of space-time block codes are investigated. Finally, space-time coded adaptive Orthogonal Frequency Division Multiplexing (AOFDM) is investigated.

Acknowledgements

Three years ago, Professor Lajos Hanzo have managed to secure a place for me in the Communication Group at the Southampton University. Throughout this period, I am touched by his enthusiasm in research and hard work, and hence inspire my interest in the research. This inspiration continues until the present day and will continue for the rest of my life. Here I would like to express my greatest gratitude to him.

For the past three years, I was fortunate to have a bunch of helpful and charming colleagues in the Communication Group. My first mentor in the group, Jason Woodard who have said, "Of course, you can ask whatever questions you like." How Hee Thong who have inspired me to play more sports and become my personal trainer. Yeap Bee Leong who gave useful advice in my research and as well as my communication skills. Yang Lie-Liang who have introduced an ancient Chinese theory to me. Thanks to the pair in the group, Wong Chong Hin and Kuan Ee Lin who have set a good example. Jonathan Blogh who enlight me of the British's way. Yee Mong Suan who always present different kind of perspective in life. Choi Byoung Jo who answered my questions regarding OFDM. Not forgetting also Thomas Keller and Peter Cherriman for lending their time and computing expertise and Denise Harvey for her help with the administrative paper work. Also, special thanks to Jozef Hàmorskỳ, Ng Soon Xin, Jocehn Pliquett, Terry Mitchell, Yen Kai and Lee Chee Siong.

Finally, my heartful thanks to my family members in Malaysia and Li Na. For the past 27 years, my father has always been planning ahead of me. To him, education is the most important thing for us while we were small and always 'catch' us for a small reading session. With his hard work, he has managed to provide the support for our basic education and even higher education. For my mother who always worry for me. With my arrival in the UK, only then I realised how difficult is it for her to prepare the meals and take care of the whole family. For Li Na who brings out my sweet childhood memories. She also taught me that life is not only about hard work but also to enjoy yourself.

List of Publications

1. **T. H. Liew, C. H. Wong, L. Hanzo** "Block turbo coded burst-by-burst adaptive modems", Proceedings of Microcoll'99, Budapest, Hungary, 21-24 March 1999, pp 59-62.

2. **B. L. Yeap, T. H. Liew, J. Hamorsky and L. Hanzo** "Block-based turbo-coded, turbo-equalised partial-response modulation for dispersive mobile channels", Proceedings of Microcoll'99, Budapest, Hungary, 21-24 March 1999, pp 71-74.

3. **C. H. Wong, T. H. Liew and L. Hanzo** "Turbo Coded Burst by Burst Adaptive Wideband Modulation with Blind Modem Mode Detection", ACTS Mobile Communications Summit, Sorrento, Italy, 8-11 June 1999, pp 303-308.

4. **T. H. Liew, L-L. Yang and L. Hanzo** "Soft-Decision Redundant Residue Number System Based Error Correction Coding", Proceedings of VTC '99 Fall, Amsterdam, Holland, 19-22 September 1999, pp 2546-2550.

5. **B. L. Yeap, T. H. Liew, J. Hamorsky and L. Hanzo** "Comparitive Study of Turbo Equalisers using Convolutional Codes and Block-Based Turbo-Codes for GMSK Modulation", Proceedings of VTC 1999 Fall, Amsterdam, Holland, 19-22 September 1999, pp 2974-2978.

6. **M. S. Yee, T. H. Liew and L. Hanzo** "Radial basis function decision feedback equalisation assisted block turbo burst-by-burst adaptive modems", Proceedings of VTC 1999 Fall, Amsterdam, Holland, 19-22 September 1999, pp 1600-1604.

7. **T.H. Liew, B.L. Yeap, J. Woodard and L. Hanzo** "Modified MAP Algorithm for Extended Turbo BCH Codes and Turbo Equalisers", 3G Mobile Communications Technologies, 27-29 March 1999, pp 185-189.

8. **T. H. Liew, L-L. Yang and L. Hanzo** "Turbo Decoded Redundant Residue Number System Codes", Proceedings of VTC 2000 Spring, Tokyo, Japan, 15-18 May 2000, pp 576-580.

9. **S. X. Ng, T. H. Liew, L-L. Yang and L. Hanzo** "Turbo Coding Performance Using Block Component Codes", Proceedings of VTC 2000 Spring, Tokyo, Japan, 15-18 May 2000, pp 849-853.

10. **T. Keller, T. H. Liew and L. Hanzo** "Adaptive Rate RRNS Coded OFDM Transmission for Mobile Communication Channels", Proceedings of VTC 2000 Spring, Tokyo, Japan, 15-18 May 2000, pp 230-234.

11. **T. H. Liew, J. Pliquett, B. L. Yeap, L-L. Yang and L. Hanzo** "Comparative Study of Space Time Block Codes and Various Concatenated Turbo Coding Schemes", PIMRC, London, UK, 18-21 September 2000, pp 741-745.

12. **T. H. Liew, J. Pliquett, B. L. Yeap, L-L. Yang and L. Hanzo** "Concatenated Space Time Block Codes and TCM, Turbo TCM, Convolutional as well as Turbo Codes", Globecom 2000, San Francisco, USA, 27 Nov - 1 Dec 2000, pp 1829-1833.

13. **T. Keller, T. H. Liew and L. Hanzo** "Adaptive redundant residue number system coded multicarrier modulation", IEEE Journal on Selected Areas in Communications, Nov 2000, Vol. 18, No 11, pp 2292-2301.

14. **T. Keller, T. H. Liew and L. Hanzo** "Block-Coded Adaptive OFDM", pp 613-636, Chapter 23 in Single- and Multi-carrier Quadrature Amplitude Modulation, John Wiley & Sons, Ltd. England 2000, ISBN 0 471 49239 6

15. **M. Yee, T. H. Liew and L. Hanzo** "Burst-by-burst adaptive turbo coded radial basis function assisted feedback equalisation", *accepted for IEEE Transactions on Communications.*

16. **T. H. Liew, L-L. Yang and L. Hanzo** "Systematic Redundant Residue Number System Codes: Analytical Upper Bound and Iterative Decoding Performance over AWGN & Rayleigh Channels", *submitted to IEEE Transactions of Vehicular Technology.*

17. **C. H. Wong, T. H. Liew and L. Hanzo** "Blind-detection Assisted, Block Turbo Coded, Decision-feedback Equalised Burst-by-burst Adaptive Modulation", *submitted to the IEEE Journal on Selected Areas in Communication.*

18. **B. L. Yeap, T. H. Liew, J Hámorský and L. Hanzo** "Comparitive Study of Turbo Equalisers using Convolutional Codes, Convolutional-based Turbo Codes and Block-based Turbo Codes", *submitted to IEEE Journal on Selected Areas in Communication.*

19. **H. T. How, T. H. Liew, E. L. Kuan and L. Hanzo** "An AMR Coding, RRNS Channel Coding and Adaptive JD-CDMA System for Speech Communications", *submitted to IEEE Journal on Selected Areas in Communications*

20. **T. H. Liew, B. J. Choi and L. Hanzo** "Comparative Study of Concatenated Turbo Coded and Space-Time block coded as well as Space-Time Trellis coded OFDM", *submitted to the VTC 2001 Spring, Tel Aviv, Israel.*

21. **H. T. How, T. H. Liew, E. L. Kuan and L. Hanzo** "A Burst-by-Burst Adaptive Joint-Detection Based CDMA Speech Transceiver", *submitted to ICC2001, Helsinki, Finland.*

22. **B. J. Choi, T. H. Liew and L. Hanzo** "Concatenated Space-Time Block Coded and Turbo Coded Symbol-by-Symbol Adaptive OFDM and Multi-Carrier CDMA Systems", *submitted to the VTC 2001 Spring, Tel Aviv, Israel.*

23. **B. L. Yeap, T. H. Liew and L. Hanzo** "Turbo Equalization of Serially Concatenated Systematic Convolutional Codes and Systematic Space Time Trellis Codes", *submitted to the VTC 2001 Spring, Tel Aviv, Israel.*

Contents

Chapter 1

Introduction

1.1 A Historical Perspective on Channel Coding

The history of channel coding or forward error correction (FEC) coding dates back to Shannon's pioneering work [1] in 1948, predicting that arbitrarily reliable communications is achievable with the aid of channel coding, upon adding redundant information to the transmitted messages. However, Shannon refrained from proposing explicit channel coding schemes for practical implementations. Furthermore, although upon increasing the amount of redundancy added the associated information delay increases, he did not specify the maximum delay that may have to be tolerated, in order to be able to communicate near the Shannonian limit. In recent years researchers have been endeavouring to reduce the amount of latency inflicted for example by a turbo codec's interleaver that has to be tolerated for the sake of attaining a given target performance.

Historically, one of the first practical FEC codes was the single error correcting Hamming code [2], which was a block code proposed in 1950. Convolutional FEC codes date back to 1955 [3], which were discovered by Elias, while Wozencraft and Reiffen [4,5], as well as Fano [6] and Massey [7] proposed various algorithms for their decoding. A major milestone in the history of convolutional error correction coding was the invention of a maximum likelihood sequence estimation algorithm by Viterbi [8] in 1967. A classic interpretation of the Viterbi algorithm (VA) can be found, for example, in Forney's often-quoted paper [9]. One of the first practical applications of convolutional codes was proposed by Heller and Jacobs [10] during the seventies.

We note here that the VA does not result in minimum bit error rate (BER), it rather finds the most likely transmitted sequence of transmitted bits. However, it performs close to the minimum possible BER, which can be achieved only with the aid of the extremely complex full-search algorithm evaluating the probability of all possible 2^k binary strings of a $k-$bit message. The minimum BER decoding algorithm was proposed in 1974 by Bahl *et al.* [11], which was termed the Maximum A-Posteriori (MAP) algorithm. Although the MAP algorithm slightly outperforms the VA in BER terms, because of its significantly higher complexity it was rarely used in practice, until turbo codes were contrived by Berrou *et al.* in 1993 [12,13].

Focusing our attention on block codes, the single-error correcting Hamming block code was too weak for practical applications. An important practical milestone was the discovery of the family of multiple error correcting Bose-Chaudhuri-Hocquenghem (BCH) binary block codes [14] in 1959 and in 1960 [15,16]. In 1960, Peterson [17] recognised that these codes exhibit a cyclic structure, implying that all cyclically shifted versions of a legitimate codeword are also legitimate codewords. The first method for constructing trellises for linear block codes was proposed by Wolf [18] in 1978. Due to the associated high complexity, there was only limited research in trellis decoding of linear block codes [19,20]. It was in 1988, when Forney [21] showed that some block codes have relatively simple trellis structures. Motivated by Forney's work, Honary, Markarian and Farrell *et al.* [19,22–25] as well as Lin and Kasami *et al.* [20,26,27] proposed various methods for reducing the associated complexity. The Chase algorithm [28] is one of the most popular techniques proposed for near maximum likelihood decoding of block codes.

Furthermore, in 1961 Gorenstein and Zierler [29] extended the binary coding theory to treat non-binary codes as well, where code symbols were constituted by a number of bits, and this led to the birth of burst-error correcting codes. They also contrived a combination of algorithms, which is referred to as the Peterson-Gorenstein-Zierler (PGZ) algorithm. In 1960 a prominent non-binary subset of BCH codes was discovered by Reed and Solomon [30]; they were named after their inventors Reed-Solomon (RS) codes. These codes exhibit certain optimality properties, since their codewords have the highest possible minimum distance between the legitimate codewords for a given code-rate. This, however, does not necessarily guarantee attaining the lowest possible BER. The PGZ decoder can also be invoked for decoding non-binary RS codes. A range of powerful decoding algorithms for RS codes was found by Berlekamp [31,32] and Massey [33,34]. Various soft decision decoding algorithms were proposed for soft decoding of RS codes by Sweeney [35–37] and Honary [19]. In recent years RS codes have found practical applications, for example, in Compact Disc (CD) players, in deep-space scenarios [38], and in the family of Digital Video Broadcasting (DVB) schemes [39], which were standardised by the European Telecommunications Standardisation Institute (ETSI).

Inspired by the ancient theory of Residue Number Systems (RNS) [40–42], which constitute a promising number system for supporting fast arithmetic operations [40, 41], a novel class of non-binary codes referred to as Redundant Residue Number System (RRNS) codes were introduced in 1967. An RRNS code is a maximum-minimum distance block code, exhibiting similar distance properties to Reed-Solomon (RS) codes. Watson and Hastings [42] as well as Krishna *et al.* [43, 44] exploited the properties of the RRNS for detecting or correcting a single error and also for detecting multiple errors. Recently, the soft decoding of RRNS codes was proposed in [45].

During the early 1970s, FEC codes were incorporated in various deep-space and satellite communications systems, and in the 1980s they also became common in virtually all cellular mobile radio systems. However, for a long time FEC codes and modulation have been treated as distinct subjects in communication systems. By integrating FEC and modulation, in 1987 Ungerboeck [46–48] proposed Trellis Coded Modulation (TCM), which is capable of achieving significant coding gains over power and band-limited transmission media. A further historic breakthrough was the invention of turbo codes by Berrou, Glavieux, and Thitimajshima [12, 13] in 1993, which facilitate the operation of communications systems near the Shannonian limits. Turbo coding is based on a composite codec constituted by two parallel concatenated codecs. Since its recent invention turbo coding has evolved at an unprecedented rate and has reached a state of maturity within just a few years due to the intensive research efforts of the turbo coding community. As a result of this dramatic evolution, turbo coding has also found its way into standardised systems, such as for example the recently ratified third-generation (3G) mobile radio systems [49]. Even more impressive performance gains can be attained with the aid of turbo coding in the context of video broadcast systems, where the associated system delay is less critical, than in delay-sensitive interactive systems.

More specifically, in their proposed scheme Berrou *et al.* [12, 13] used a parallel concatenation of two Recursive Systematic Convolutional (RSC) codes, accommodating the turbo interleaver between the two encoders. At the decoder an iterative structure using a modified version of the classic minimum bit error rate Maximum A-Posteriori Algorithm (MAP) invented by Bahl *et al.* [11] was invoked by Berrou *et al.*, in order to decode these parallel concatenated codes. Again, since 1993 a large body of work has been carried out in the area, aiming for example for reducing the associated decoder complexity. Practical reduced-complexity decoders are for example the Max-Log-MAP algorithm proposed by Koch and Baier [50], as well as by Erfanian *et al.* [51], the Log-MAP algorithm suggested by Robertson, Villebrun and Hoeher [52], and the SOVA algorithm advocated by Hagenauer as well as Hoeher [53, 54]. Le Goff, Glavieux and Berrou [55], Wachsmann and

Huber [56] as well as Robertson and Worz [57] suggested to use these codes in conjunction with bandwidth efficient modulation schemes. Further advances in understanding the excellent performance of the codes are due, for example to Benedetto and Montorsi [58, 59], Perez, Seghers and Costello [60]. During the mid-nineties Hagenauer, Offer and Papke [61], as well as Pyndiah [62] extended the turbo concept to parallel concatenated block codes as well. Nickl *et al.* shows in [63] that Shannon's limit can be approached within 0.27 dB by employing a simple turbo Hamming code. In [64] Acikel and Ryan proposed an efficient procedure for designing the puncturing patterns for high-rate turbo convolutional codes. Jung and Nasshan [65, 66] characterised the achievable turbo coded performance under the constraints of short transmission frame lengths, which is characteristic of interactive speech systems. In collaboration with Blanz they also applied turbo codes to a CDMA system using joint detection and antenna diversity [67]. Barbulescu and Pietrobon addressed the issues of interleaver design [68]. The tutorial paper by Sklar [69] is also highly recommended as background reading.

Driven by the urge to support high data rates for a wide range of bearer services, Tarokh, Seshadri and Calderbank [70] proposed space-time trellis codes in 1998. By jointly designing the FEC, modulation, transmit diversity and optional receive diversity scheme, they increased the throughput of band-limited wireless channels. A few months later, Alamouti [71] invented a low-complexity space-time block code, which offers significantly lower complexity at the cost of a slight performance degradation. Alamouti's invention motivated Tarokh *et al.* [72, 73] to generalise Alamouti's scheme to an arbitrary number of transmitter antennas. Then, Tarokh *et al.*, Bauch *et al.* [74, 75], Agrawal [76], Li *et al.* [77, 78] and Naguib *et al.* [79] extended the research of space-time codes from considering narrow-band channels to dispersive channels [70, 71, 73, 79, 80].

In Figure 1.1, we show the evolution the channel coding research over the past fifty years since Shannon's legendary contribution [1]. These mile-stones have been incorporated also in the range of monographs and text-books summarised in Figure 1.2. At the time of this writing, the Shannon limit has been approached within 0.27 dB [63] over Gaussian channels. Now the challenge is to contrive FEC schemes, which are capable of achieving a performance near the *capacity of wireless channels.* The design of an attractive channel coding and modulation scheme depends on a range of contradictory factors, which are portrayed in Figure 1.3. The message of this illustration is multi-fold. For example, given a certain transmission channel, it is always feasible to design a coding and modulation ('codulation') system, which can further reduce the BER achieved. This typically implies however further investments and/or penalties in terms of the required increased implementational complexity and coding/interleaving delay as well as reduced effective throughput. Different

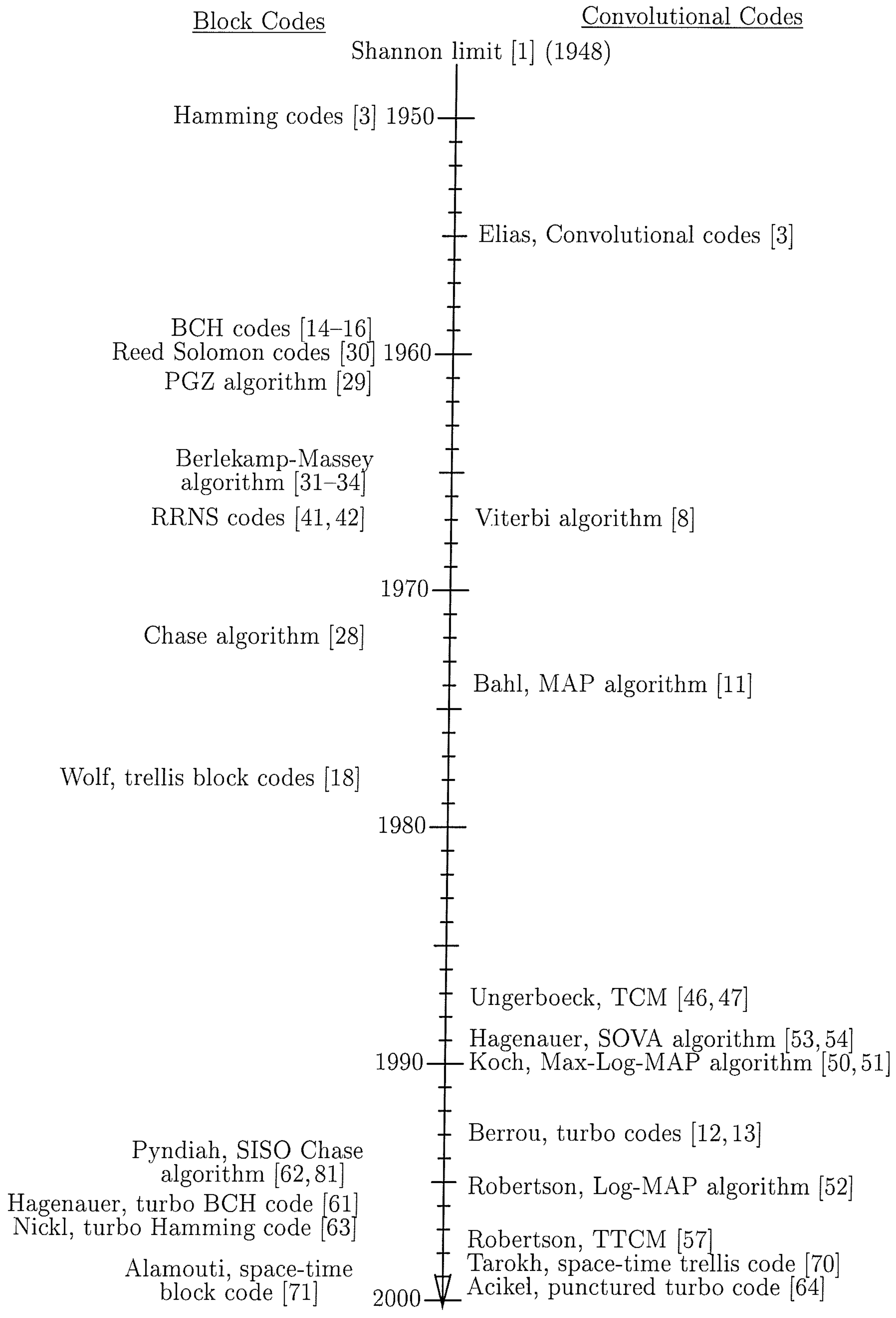

Block Codes
Convolutional Codes
Shannon limit [1] (1948)
Hamming codes [3] 1950
Elias, Convolutional codes [3]
BCH codes [14–16]
Reed Solomon codes [30] 1960
PGZ algorithm [29]
Berlekamp-Massey algorithm [31–34]
RRNS codes [41,42]
Viterbi algorithm [8]
1970
Chase algorithm [28]
Bahl, MAP algorithm [11]
Wolf, trellis block codes [18]
1980
Ungerboeck, TCM [46,47]
Hagenauer, SOVA algorithm [53,54]
1990
Koch, Max-Log-MAP algorithm [50,51]
Berrou, turbo codes [12,13]
Pyndiah, SISO Chase algorithm [62,81]
Robertson, Log-MAP algorithm [52]
Hagenauer, turbo BCH code [61]
Nickl, turbo Hamming code [63]
Robertson, TTCM [57]
Alamouti, space-time block code [71]
Tarokh, space-time trellis code [70]
2000
Acikel, punctured turbo code [64]

Figure 1.1: A brief history of channel coding.

solutions accrue, when optimising different codec features. For example, in many applications the most important codec parameters is the achievable coding gain, which quantifies the amount of bit-energy reduction attained by a codec at a certain target BER. Naturally, transmitted power reduction is extremely important in battery powered devices. This transmitted power reduction is only achievable at the cost of an increased implementational complexity, which itself typically increases the power consumption and hence erodes some of the power gain.

Viewing this system optimisation problem from a different perspective, it is feasible to transmit at a higher bit rate in a given fixed bandwidth by increasing the number of bits per modulated symbol. However, when aiming for a given target BER, the channel coding rate has to be reduced, in order to increase the transmission integrity. Naturally, this reduces the *effective throughput* of the system and results in an overall increased system complexity. When the channel's characteristic and the associated bit error statistics change, different solutions may become more attractive. This is because Gaussian channels, narrowband and wideband Rayleigh fading or various Nakagami fading channels [104, 105] inflict different impairment. These design trade-offs constitute the subject of this thesis.

1.2 Organisation of Thesis

Below, we present the outline of the thesis:

- **Chapter 2:** An overview of the conventional BCH codes is given. The Viterbi Algorithm is detailed using a BCH code as our example. This is followed by simulation results of various BCH codes employing hard decision and soft decision decoding methods. The classic Chase algorithm is introduced and its performance is investigated.
- **Chapter 3:** The concept of turbo codes using BCH codes as component codes, is introduced. A detailed derivation of the MAP algorithm is given. Then, the MAP algorithm was modified in order to highlight the concept of the Max-Log-MAP and Log-MAP algorithms. Furthermore, the SOVA algorithm is introduced. Then a simple turbo decoding example is given, highlighting how iterative decoding assists in correcting multiple errors. We propose a novel MAP algorithm for decoding extended BCH codes. Finally, we show how the various coding parameters affect the performance of turbo BCH codes.
- **Chapter 4:** The concept of Residue Number Systems (RNS) is introduced and extended to Redundant Residue Number Systems (RRNS), introducing the family of

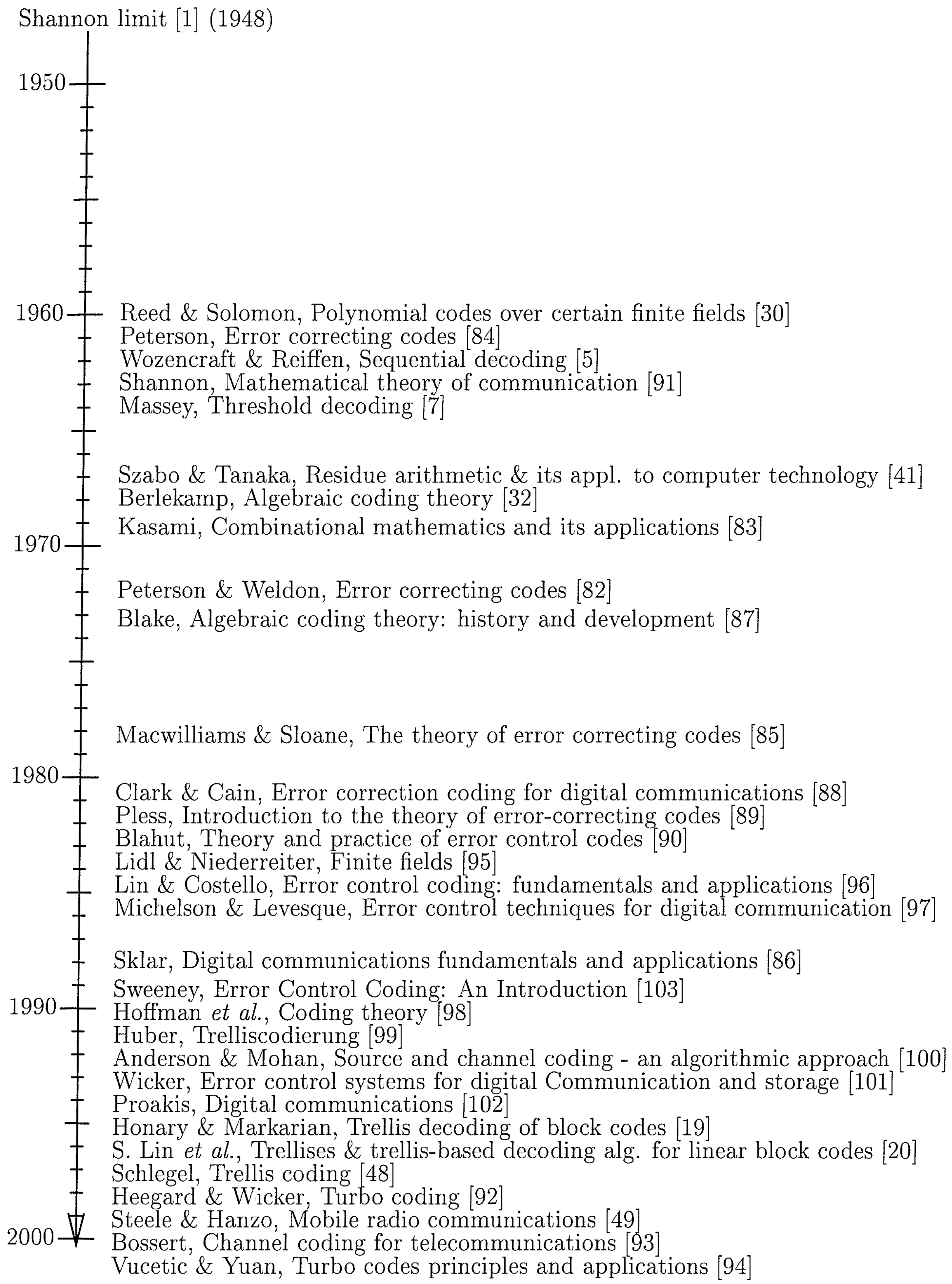

Shannon limit [1] (1948)
1950
1960
Reed & Solomon, Polynomial codes over certain finite fields [30]
Peterson, Error correcting codes [84]
Wozencraft & Reiffen, Sequential decoding [5]
Shannon, Mathematical theory of communication [91]
Massey, Threshold decoding [7]
Szabo & Tanaka, Residue arithmetic & its appl. to computer technology [41]
Berlekamp, Algebraic coding theory [32]
Kasami, Combinational mathematics and its applications [83]
1970
Peterson & Weldon, Error correcting codes [82]
Blake, Algebraic coding theory: history and development [87]
Macwilliams & Sloane, The theory of error correcting codes [85]
1980
Clark & Cain, Error correction coding for digital communications [88]
Pless, Introduction to the theory of error-correcting codes [89]
Blahut, Theory and practice of error control codes [90]
Lidl & Niederreiter, Finite fields [95]
Lin & Costello, Error control coding: fundamentals and applications [96]
Michelson & Levesque, Error control techniques for digital communication [97]
Sklar, Digital communications fundamentals and applications [86]
Sweeney, Error Control Coding: An Introduction [103]
1990
Hoffman *et al.*, Coding theory [98]
Huber, Trelliscodierung [99]
Anderson & Mohan, Source and channel coding - an algorithmic approach [100]
Wicker, Error control systems for digital Communication and storage [101]
Proakis, Digital communications [102]
Honary & Markarian, Trellis decoding of block codes [19]
S. Lin *et al.*, Trellises & trellis-based decoding alg. for linear block codes [20]
Schlegel, Trellis coding [48]
Heegard & Wicker, Turbo coding [92]
Steele & Hanzo, Mobile radio communications [49]
2000
Bossert, Channel coding for telecommunications [93]
Vucetic & Yuan, Turbo codes principles and applications [94]

Figure 1.2: Mile-stones in channel coding.

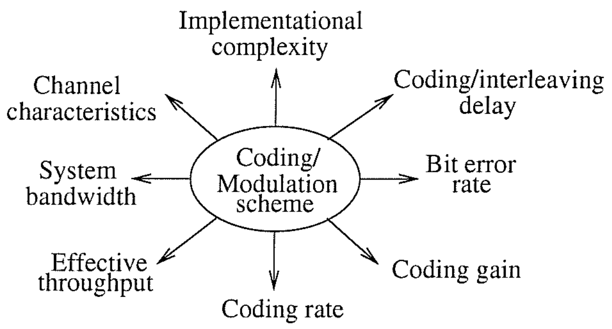


Figure 1.3: Factors affecting the design of channel coding and modulation scheme.

RRNS codes. Some coding theoretic aspects of RRNS codes are investigated, demonstrating that RRNS codes exhibit similar distance properties to RS codes. A procedure for multiple error correction is then given. Different bit-to-symbol mapping methods are highlighted, yielding non-systematic and systematic RRNS codes. A novel bit-to-symbol mapping method is proposed, which results in efficient systematic RRNS codes. The classic Chase algorithm is then modified in order to create a Soft-Input Soft-Output (SISO) RRNS decoder. This enables us to implement the iterative decoding of turbo RRNS codes. Finally, simulations results are given for various RRNS codes, employing hard decision and soft decision decoding methods. The performance of the RRNS codes is compared to that of RS codes and the performance of turbo RRNS codes is studied.

- **Chapter 5:** Space-time block codes are introduced. The derivation of the MAP decoding of space-time block codes is then given. A system is proposed by concatenating space-time block codes and various channel codes. The complexity and memory requirements of various channel decoders are derived, enabling us to compare the performance of the proposed channel codes by considering their decoder complexity. Our simulation results related to space-time block codes using no channel coding are first presented. Then, we investigate the effect of mapping data and parity bits from binary channel codes to non-binary modulation schemes. Finally, we compare our simulation results for various channel codes concatenated with a simple space-time block code. Our performance comparisons are conducted by also considering the complexity of the associated channel decoder.

- **Chapter 6:** The encoding process of space-time trellis codes is highlighted. This is followed by employing an Orthogonal Frequency Division Multiplexing (OFDM) modem in conjunction with space-time codes over wideband channels. Turbo codes and RS codes are concatenated with space-time codes in order to improve their performance. Then, the performance of the advocated space-time block code and space-time trellis codes is compared. Their complexity is also considered in comparing both schemes. The effect of delay-spread and maximum Doppler frequency on the performance of the space-time codes is investigated. A Signal Interference Ratio (SIR) related term is defined in the context of dispersive channels for the advocated space-time block code, and we will show how the SIR affects the performance of the system. In our last section, we propose space-time coded Adaptive OFDM (AOFDM). We then show by employing multiple antennas that with the advent of space-time coding, the wideband fading channels have been converted to AWGN-like channels.

- **Chapter 7:** The main findings are summarised and suggestions for future research are presented.

The novel contributions of the thesis are as follows:

- The extended MAP algorithm was proposed for extended BCH codes. The algorithm was employed in both extended turbo BCH codes and in turbo equalisers [106].
- Different bit-to-symbol mapping methods were proposed for systematic and non-systematic RRNS encoders. The classic Chase algorithm was then invoked for contriving soft decision decoding of RRNS codes [45].
- The classic Chase algorithm was adapted for accepting soft inputs and to provide soft outputs. Using the proposed Soft Input Soft Output (SISO) Chase algorithm, iterative decoding of turbo RRNS codes was contrived [107].
- A system which concatenates space-time block codes with various channel codes was proposed. Different bit-to-symbol mapping methods were studied for the data and parity bits from binary channel codes to non-binary modulation schemes [108]. The performance of various concatenated space-time and channel codes was compared. The issues of performance versus complexity was also addressed [109].
- Low complexity space-time block codes were concatenated with turbo codes. The performance of the concatenated scheme was comparatively studied in conjunction with the more complex family of space-time trellis codes using OFDM [108].

Chapter 2

Conventional BCH Codes

2.1 Introduction

Bose-Chaudhuri-Hocquenghem (BCH) codes were discovered by Hocquenghem [14] and independently by Bose and Chaudhuri [15] in 1959 and 1960, respectively. These codes constitute a prominent class of *cyclic block codes* that have multiple-error detection and correction capabilities.

In this chapter, we will commence with an introduction to BCH codes in Section 2.2. Their state and trellis diagrams are constructed in Section 2.2.2. The trellis decoding of BCH codes using the Viterbi Algorithm (VA) [8] is detailed in Section 2.3.2. Simulation results of various BCH codes employing the Berlekamp-Massey Algorithm (BMA) [31–34] and the Viterbi algorithm are given in Section 2.3.5. Finally, in Section 2.4 we investigate the low-complexity Chase algorithm [28] in the context of the soft decision based decoding of BCH codes.

2.2 BCH codes

A BCH encoder accepts k information data bits and produces n coded bits. The minimum Hamming distance of the codewords is d_{min} and the BCH code concerned is denoted as BCH(n, k, d_{min}). Table 2.1 lists some commonly used code generators, $g(x)$, for the construction of BCH codes [110]. The coefficients of $g(x)$ are presented as octal numbers arranged so that when they are converted to binary digits, the rightmost digit corresponds to the zero-degree coefficient of $g(x)$.

n	k	d_{min}	$g(x)$
7	4	3	13
15	11	3	23
	7	5	721
31	26	3	45
	21	5	3551
	16	7	107657
63	57	3	103
	51	5	12471
	45	7	1701317
	39	9	166623567
	36	11	1033500423
127	120	3	211
	113	5	41567
	106	7	11554743
	99	9	3447023271
	92	11	624730022327
	85	13	130704476322273
	78	15	26230002166130115
	71	19	6255010713253127753
	64	21	1206534025570773100045
255	247	3	435
	239	5	267543
	231	7	156720665
	223	9	75626641375
	215	11	23157564726421
	207	13	16176560567636227
	199	15	7633031270420722341
	191	17	2663470176115333714567
	187	19	52755313540001322236351
	179	21	22624710717340432416300455
	171	23	15416214212342356077061630637
	163	25	7500415510075602551574724514601
	155	27	3757513005407665015722506464677633
	147	29	1642130173537165525304165305441011711
	139	31	461401732060175561570722730247453567445
	131	37	215713331471510151261250277442142024165471

Table 2.1: Table of generators for BCH codes [110] ©IEEE, Stenbit.

2.2.1 BCH Encoder

Since BCH codes are cyclic codes, their encoders can be implemented using shift register circuits [111–113]. The codes can be encoded either non-systematically or systematically. However, systematic BCH codes were found to perform slightly better, than their non-systematic counterparts. Hence, only systematic BCH codes will be discussed in this report.

For systematic codes, the generator polynomial, $g(x)$, is written as follows:

$$g(x) = g_0 + g_1x + g_2x^2 + \ldots + g_{n-k-1}x^{n-k-1} + g_{n-k}x^{n-k} \ . \tag{2.1}$$

The generator polynomial, $g(x)$, formulates n codeword bits by appending $(n-k)$ parity bits to the k information data bits. The encoder employs a shift register having $(n-k)$ stages as depicted in Figure 2.1, where $\otimes$ represents multiplication, whereas $\oplus$ modulo two addition. In simple plausible terms the code exhibits error correction ability, since only certain encoded sequences, obeying the encoding rules are legitimate and hence corrupted or illegitimate codewords can be recognised and corrected. The parity bits are computed from the information data bits according to the rules imposed by the generator polynomial.

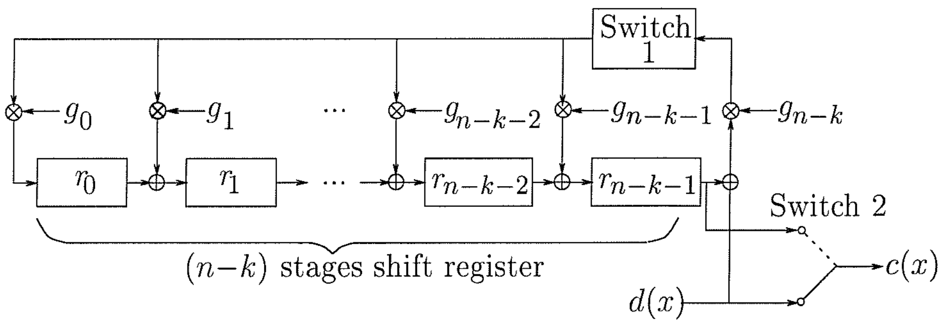


Figure 2.1: Systematic encoder for BCH codes having $(n-k)$ shift register stages.

The following steps describe the encoding procedures:

1. Switch 1 is closed during the first k shifts, in order to allow the information data bits, $d(x)$, shift into the $n-k$ stages of the shift register.

2. At the same time, Switch 2 is in the down position to allow the data bits, $d(x)$, to be copied directly to the codeword, $c(x)$.

3. After kth shifts, Switch 1 is opened and Switch 2 is moved to the upper position.

4. The remaining $n-k$ shifts clear the shift register by appending the parity bits to the codeword, $c(x)$.

Let us consider the BCH(7,4,3) code as an example for illustrating the process of encoding. From Table 2.1, the generator polynomial is

$$\begin{aligned} g &= 13_{octal} \\ &= 1011_{bin} \\ g(x) &= x^3 + x + 1. \end{aligned} \tag{2.2}$$

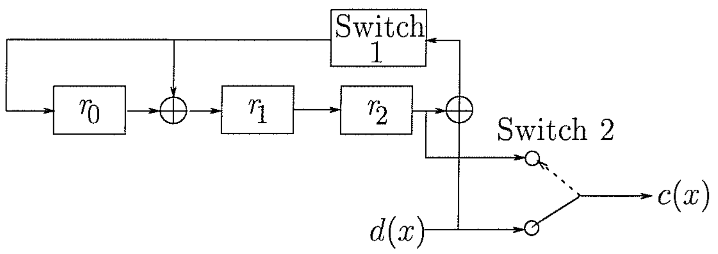


Figure 2.2: Systematic encoder for the BCH(7,4,3) code having $n - k = 3$ register stages.

Figure 2.2 shows the specific encoder, which is a derivative of Figure 2.1. Observe that all the multipliers illustrated in Figure 2.1 are absent in Figure 2.2. Explicitly, if the generator polynomial coefficient is 1, the multiplier is replaced by a direct hard-wire connection as shown in Figure 2.2, whereas if the coefficient is 0, no connection is made.

Let us use the shift register shown in Figure 2.2 for encoding four ($k = 4$) information data bits, $d = 1\ 0\ 1\ 1$ ($d(x) = 1 + x^2 + x^3$). The operational steps are as follows:

Input queue	Shift index	Shift register	Codeword
		$r_0 r_1 r_2$	$c_0 c_1 c_2 c_3 c_4 c_5 c_6$
1 0 1 1	0	0 0 0	- - - - - - -
1 0 1	1	1 1 0	- - - - - - 1
1 0	2	1 0 1	- - - - - 1 1
1	3	1 0 0	- - - - 0 1 1
-	4	1 0 0	- - - 1 0 1 1
-	5	0 1 0	- - 0 1 0 1 1
-	6	0 0 1	- 0 0 1 0 1 1
-	7	0 0 0	1 0 0 1 0 1 1

The shift registers must be reset to zero before the encoding process starts. After the fourth shift, Switch 1 is opened and Switch 2 is moved to the upper position. The parity bits contained in the shift register are appended to the codeword. The codeword is $c =$

1 0 0 1 0 1 1 ($c(x) = 1 + x^3 + x^5 + x^6$). The binary representation of both $d(x)$ and $c(x)$ is shown in Figure 2.3.

	d_0	d_1	d_2	d_3
$d(x)$:	1	0	1	1

	c_0	c_1	c_2	c_3	c_4	c_5	c_6
$c(x)$:	1	0	0	1	0	1	1

Figure 2.3: Binary representation of the uncoded data bits and coded bits.

2.2.2 State and Trellis Diagrams

Let us study Figure 2.2 in the context of the example outlined in Section 2.2.1. As the data bits are shifted into the register by one bit at a time, the parity bits, $\{r_0, r_1, r_2\}$, represent the state of the register. The corresponding operations are shown below:

Input queue	Shift index	Shift register $r_0r_1r_2$	State	output bit
1 0 1 1	0	0 0 0	0	-
1 0 1	1	1 1 0	6	1
1 0	2	1 0 1	5	1
1	3	1 0 0	4	0
-	4	1 0 0	4	1
-	5	0 1 0	2	0
-	6	0 0 1	1	0
-	7	0 0 0	0	1

In the example above, there are a few points worth noting:

- The encoding process always starts at the all zero state and ends at the all zero state.
- The number of the output bits is always one following a clock pulse.
- For the first k (which is four for this example) shifts, the output bit is the same as the input bit.
- After the kth shift, the parity bits of the shift register are shifted to the output.

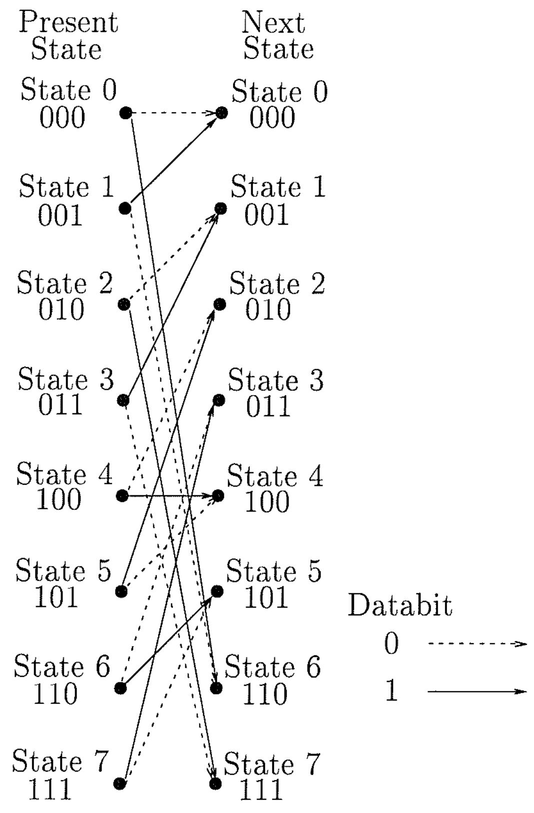


Figure 2.4: State transition diagram for the BCH(7,4,3) code having $2^{n-k} = 8$ states.

- The number of states is equal to 2^{n-k} increasing exponentially, when $n - k$ increases.

For the BCH(7,4,3) code, $n - k$ is 3 and the total number of encoder states is $2^3 = 8$. By using the shift register shown in Figure 2.2, we can find all the subsequent states when the register is in a particular state. Figure 2.4 shows all possible state transitions at any encoder state for the BCH(7,4,3) code. The branch emanating from the present state to the next state indicates the state transition. The broken line branch is the transition initiated by a databit of logical 0, whereas the solid branch is due to the databit being logical 1. The number of branches emanating from the present state is 2, which corresponds to the number of possible input bits, namely 1 and 0. As explained earlier, the output bit is the same as the databit.

The state diagram corresponding to the state transition diagram is shown in Figure 2.5. It consists of a total of $2^{n-k} = 8$ states connected by all the possible transitions shown in the state transition diagram of Figure 2.4. By using the state diagram in Figure 2.5, we can encode the databits, $d = 1\ 0\ 1\ 1$, without using the shift register shown in Figure 2.2. The first databit is a logical 1, hence the state changes from 000 to 110, as illustrated by

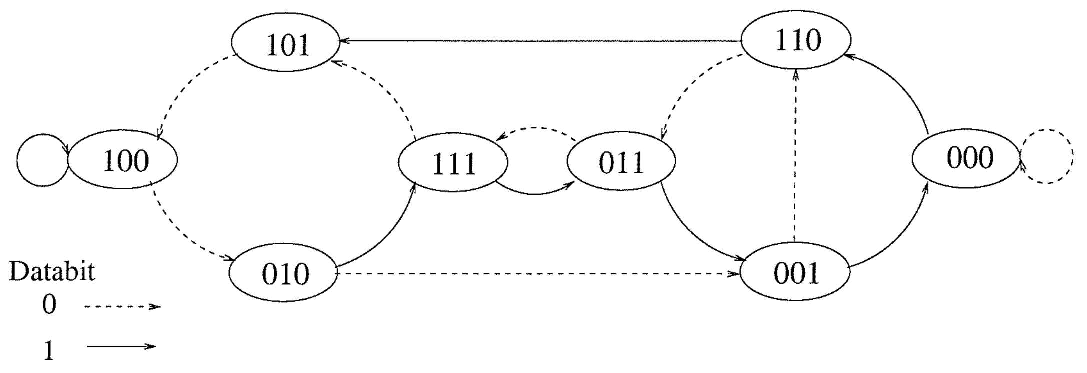


Figure 2.5: State diagram for the BCH(7,4,3) code having $2^{n-k} = 8$ states.

the solid branch emanating from state 000 in Figure 2.5. The encoder output is the same as the input databit, which is a logical 1. At the next instant, the present state becomes 110 and the databit is logical 1. This causes the state transition from 110 to 101. The encoding cycle is repeated for subsequent databits, which change the states. By following the change of states throughout the first k cycles of the encoding process, a particular path associated with states 000 $\rightarrow$ 110 $\rightarrow$ 101 $\rightarrow$ 100 $\rightarrow$ 100, can be observed.

After the kth cycle, the state changes correspond to shifting out the parity bits from the shift register. In our example, the parity bits are 100 at the kth cycle. In the following cycle, the parity bits are shifted to the right. The rightmost bit of the parity bits is shifted out to become the output bit and the leftmost bit is filled with logical 0. As a result, the state changes are 100 $\rightarrow$ 010 $\rightarrow$ 001 $\rightarrow$ 000. The whole encoding process can be associated with state transitions of 000 $\rightarrow$ 110 $\rightarrow$ 101 $\rightarrow$ 100 $\rightarrow$ 100 $\rightarrow$ 010 $\rightarrow$ 001 $\rightarrow$ 000.

Another representation of the encoding process is the trellis diagram shown in Figure 2.6. This is formed by concatenating the consecutive instants of the state transition diagram of Figure 2.4 starting from the all zero state. The diagram illustrates all the possible $2^k = 16$ paths for the BCH(7,4,3) code. The trellis has $2^{n-k} = 8$ rows (8 different states) and $n + 1 = 8$ columns. The nodes in the same row represent the same state, whereas the nodes in the same column illustrate all the possible states 000 (State a), 001 (State b), 010 (State c), ..., 111 (State h). The state transitions between adjacent columns are drawn either by a solid line or broken line, according to whether the encoder output bit is logical 1 or 0, respectively.

Initially, there is only one state, which is the all zero state (State a). The number of trellis states increases, as each new databit is inserted into the encoder. The symbol signalling instants corresponding to the column positions in the trellis, shown in Figure 2.6, are indexed

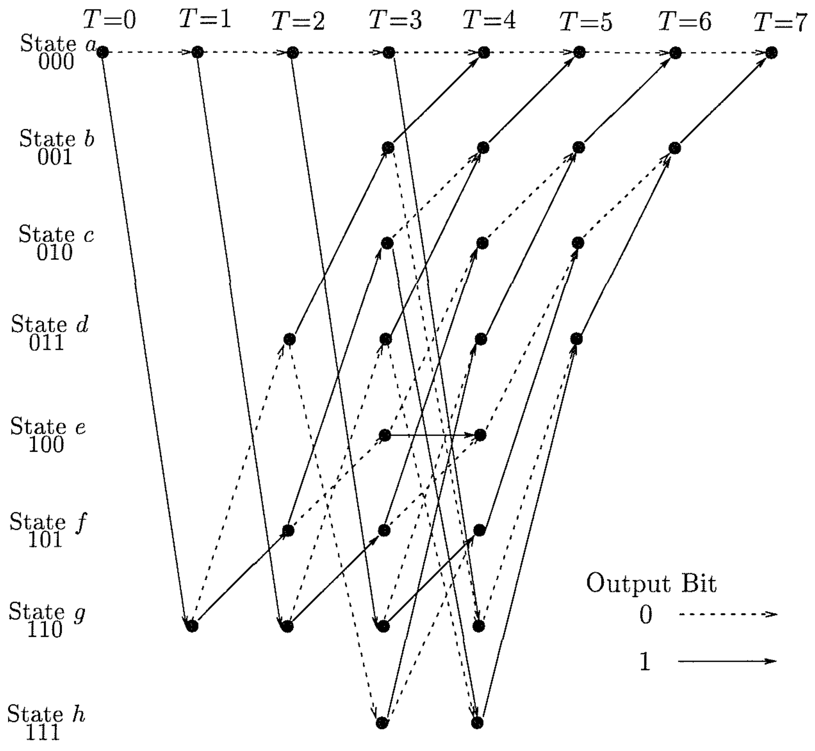


Figure 2.6: Trellis diagram for the BCH(7,4,3) code having $2^{n-k} = 8$ states and $n + 1 = 8$ consecutive stages.

by the integer T. On inserting the first databit into the encoder, $T = 0$, two different nodes are possible at the next instant. The arrival of the second databit, when $T = 1$, causes the number of possible nodes at the next instant to increase to 2^2. The number of possible nodes continues to increase with T, until the maximum number of $2^{n-k} = 8$ is reached. The maximum number of states is reached, when $T = n - k = 3$, and from then on the number of possible states is constant. After $T = k$, the number of possible states is thus divided by two at every instant in the trellis merging towards the zero state, which is reached at $T = n$.

2.3 Trellis Decoding

2.3.1 Introduction

The trellis decoding of *linear block codes* was first introduced by Wolf [18] in 1978. However, this technique is only feasible for certain BCH codes, since the number of states increases exponentially, when $n - k$ increases. The reason was outlined in the earlier sections.

2.3.2 Viterbi Algorithm

The Viterbi Algorithm (VA) was proposed by Viterbi in 1967 [8]. The algorithm searches all the possible paths in the trellis and either their Hamming or their Euclidean distances from the received sequence at the decoder's input are compared depending on whether hard or soft decision decoding is used. The path exhibiting the smallest distance from the received sequence is selected as the most likely transmitted sequence and the associated information data bits are regenerated. This method is known as maximum likelihood sequence estimation, since the most likely path is selected from the set of all the paths in the trellis.

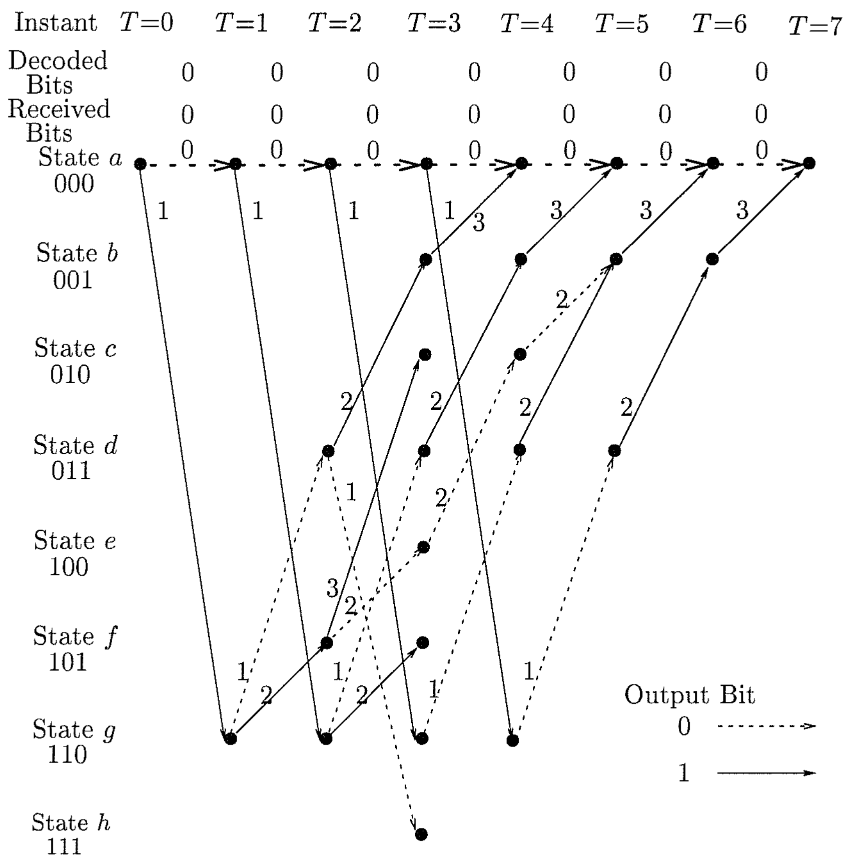


Figure 2.7: Example of Viterbi decoding of the BCH(7,4,3) code.

Figure 2.7 records the 'history' of the paths selected by the BCH(7,4,3) Viterbi decoder. Suppose that there are no channel errors and hence the input sequence of the decoder is the same as the encoded sequence, i.e. 0 0 0 0 0 0 0 . At the first instant, $T = 1$, the received bit is logical 0, which is compared with the possible transmitted bits 0 and 1 of the branches from node a to a and from node a to g, respectively. The metrics of these two branches are their Hamming distances, namely the differences between the possible transmitted bits

0 or 1 and the received bit 0. Their Hamming distances are 0 and 1, respectively.

Now, we define the branch metric as the Hamming distance of an individual branch from the received bits, and the path metric at the T-th instant as the sum of the branch metrics at all of its branches from $T = 0$ to the T-th instant. Hence the path metrics, printed on top of each branch in Figure 2.7, at instant $T = 1$ are 0 and 1 for the paths $a \rightarrow a$ and $a \rightarrow g$, respectively. At the second instant $T = 2$, the received bit is 0 and the branch metrics are 0, 1, 0 and 1 for the branches $a \rightarrow a$, $a \rightarrow g$, $g \rightarrow d$ and $g \rightarrow f$, respectively. The path metrics are 0, 1, 1 and 2 for the corresponding paths $a \rightarrow a \rightarrow a$, $a \rightarrow a \rightarrow g$, $a \rightarrow g \rightarrow d$ and $a \rightarrow g \rightarrow f$. At the third instant, the received bit is 0. There are eight possible branches and their path metrics, which are shown in Figure 2.7, are 0, 1, 2, 1, 3, 2, 1 and 2 for the paths $a \rightarrow a \rightarrow a \rightarrow a$, $a \rightarrow a \rightarrow a \rightarrow g$, $a \rightarrow g \rightarrow d \rightarrow b$, $a \rightarrow g \rightarrow d \rightarrow h$, $a \rightarrow g \rightarrow f \rightarrow c$, $a \rightarrow g \rightarrow f \rightarrow e$, $a \rightarrow a \rightarrow g \rightarrow d$ and $a \rightarrow a \rightarrow g \rightarrow f$, respectively.

Let α_1 and α_2 denote the corresponding paths $a \rightarrow a \rightarrow a \rightarrow a \rightarrow a$ and $a \rightarrow g \rightarrow d \rightarrow b \rightarrow a$ that begin at the initial node a and remerge in node a at $T = 4$. Their respective path metrics are 0 and 3. Any further branches associated with $T > 4$ stemming from node a at $T = 4$ will add identical branch metrics to the path metrics of both paths α_1 and α_2, and this means that the path metric of α_2 is larger at $T = 4$ and will remain larger for $T > 4$. The Viterbi decoder will select the path having the smallest metric, which is the all zero state sequence, and therefore discards the path α_2. The path α_1 is referred to as the *survivor*. This procedure is also applied at the other nodes for $T \geq n - k = 3$. Notice that paths $a \rightarrow g \rightarrow f \rightarrow c$ and $a \rightarrow a \rightarrow g \rightarrow f$ etc cannot survive, since their path metrics are larger than that of their counterparts of the merging pairs and they are therefore eliminated from the decoder's memory. Thus, there are only $2^{n-k} = 8$ paths that survive from instant $T = n - k$ to $T = k$. Following instant $T = k$, the number of surviving paths reduces by a factor of two for each instant.

Sometimes, two paths will merge, which have the same path metrics. At instant $T = 5$, the paths $a \rightarrow a \rightarrow a \rightarrow g \rightarrow d \rightarrow b$ and $a \rightarrow g \rightarrow f \rightarrow e \rightarrow c \rightarrow b$ remerge at node b. Both paths have the same path metric, which is 2. Normally, the Viterbi decoder will choose the survivor randomly and discard the other path. However, this situation never (or rarely) occurs in the *Soft Decision Viterbi Algorithm* or *Soft Output Viterbi Algorithm* (SOVA), which is the preferred algorithm in practical applications, since the quantised soft decision metrics are unlikely to become identical.

2.3.3 Hard Decision Viterbi Decoding

For hard decision decoding, the demodulator provides only hard-decisions (logical 1 or 0) when regenerating the transmitted sequence. In this case, the Hamming distances between the received bits and the estimated transmitted bits in the trellis are used as a metric, i.e., as a confidence measure.

2.3.3.1 Correct Hard Decision Decoding

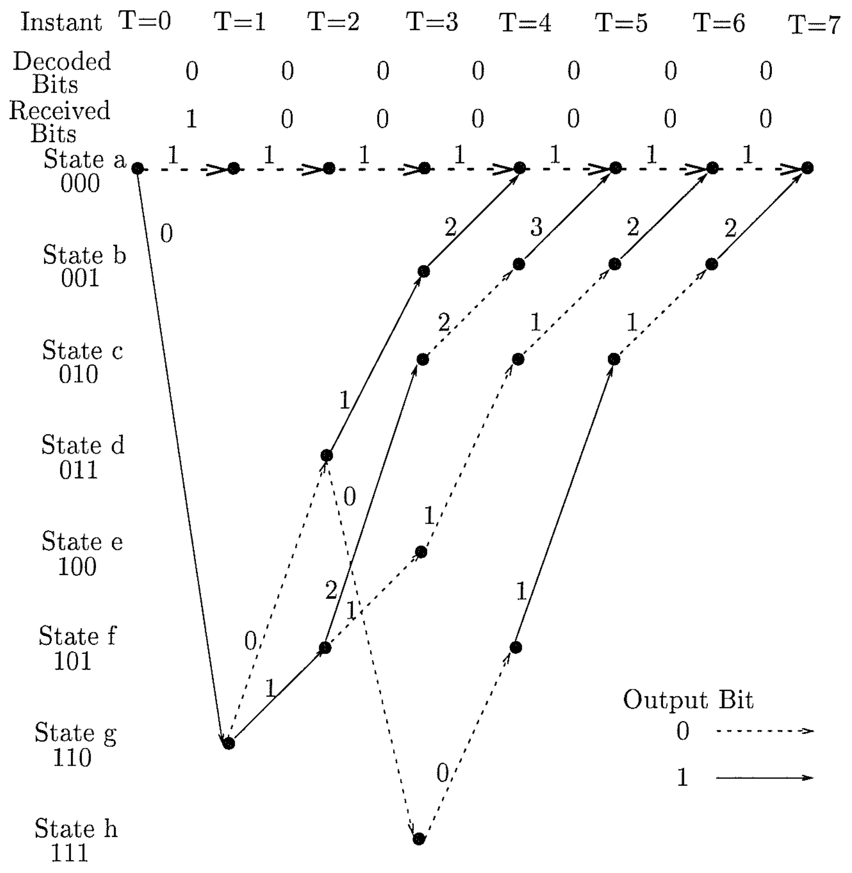


Figure 2.8: Hard decision Viterbi decoding of the BCH(7,4,3) code.

Let us illustrate the philosophy of hard decision decoding using the BCH(7,4,3) code that was previously used as an example. Assume that the transmitted sequence is 0 0 0 0 0 0 0. A channel error is introduced to the first transmitted bit and the received sequence provided by the demodulator is 1 0 0 0 0 0 0. The decoder compares the demodulator's output bit with both of the possible decoded bits indicated by the continuous and broken lines in Figure 2.8, which correspond to a binary one and zero, respectively. When the demodulator's output bit and the decoded bit are identical, their Hamming distance is zero. By contrast, when these two bits are different, a Hamming distance of one is added to the accumulated path metrics, which is written on the corresponding trellis transition. As we traverse through the trellis, the above mentioned branch metrics are summed and at $T = 7$ the path having the lowest Hamming weight is deemed the survivor path. Hence the decoded sequence is the associated string of ones and zeros.

Again, Figure 2.8 demonstrates, how the Viterbi decoder selects the survivor path (marked by the thick broken line) at the top of the figure, which has the smallest path metric, and hence decodes the received sequence correctly. Note that the path metric of the survivor is equivalent to the number of errors in the received sequence, as long as the decoder is capable of correcting the errors. This is not true, when due to channel errors the decoder diverges from the error-free trellis path.

2.3.3.2 Incorrect Hard Decision Decoding

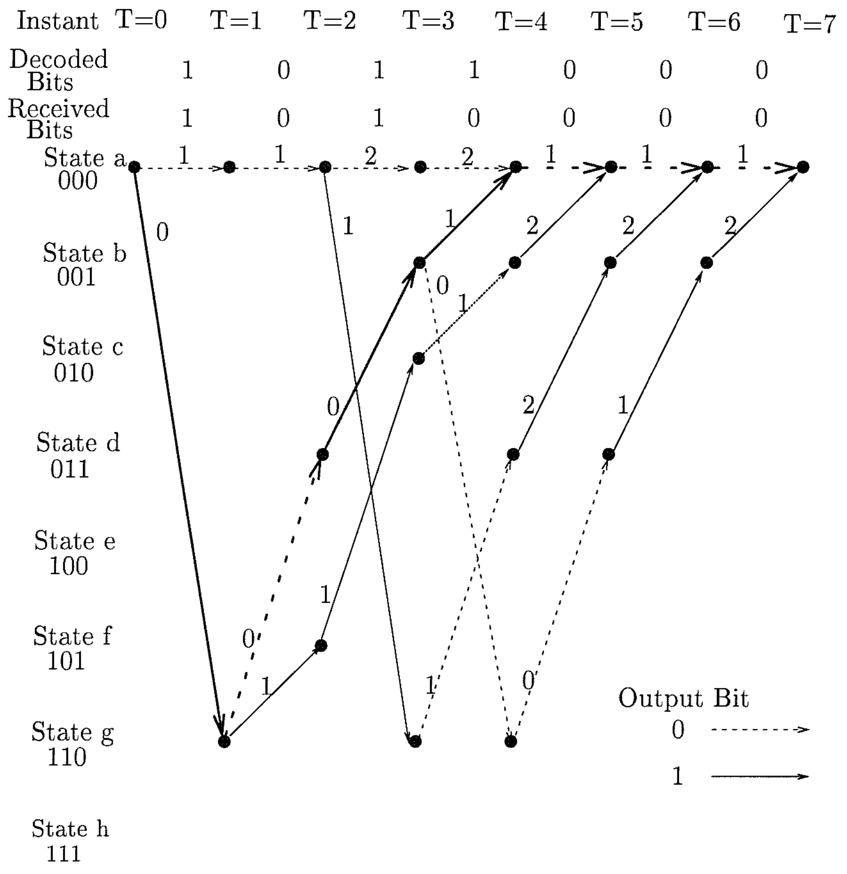


Figure 2.9: Incorrect hard decision Viterbi decoding of the BCH(7,4,3) code.

When the number of channel errors exceeds the correcting capability of the code, incorrect decoding will occur, as illustrated in Figure 2.9. Two channel errors are introduced in the first and third position of the received sequence. The incorrect decoding occurs in the four initial branches (marked by thick line), which results in the decoded sequence of 1 0 1 1 0 0 0.

The last two examples of correct and incorrect decoding, are related to decisions that depend on whether the Hamming distance of the received sequence with respect to the correct path is smaller than the distance of the received sequence to other paths in the trellis. Observe furthermore that, the surviving path's metric is now different from the number of errors encountered.

2.3.4 Soft Decision Viterbi Decoding

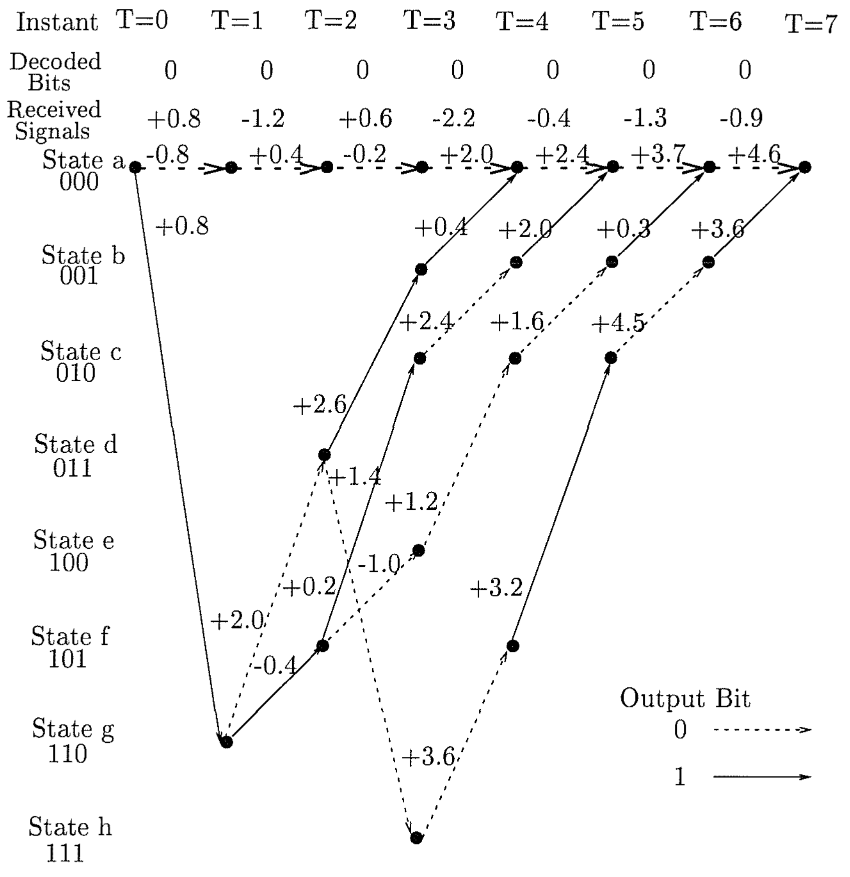


Figure 2.10: Soft decision Viterbi decoding of the BCH(7,4,3) code.

So far, we have discussed hard decision decoding. We now explore the techniques of soft decision decoding. In this approach, the received signal at the output of the demodulator is sampled. The sampled values are then directly input to the Viterbi decoder.

Assuming that we are using *Binary Phase Shift Keying* (BPSK) at the transmitter, a logical 0 will be transmitted as -1.0 and a logical 1 is sent as $+1.0$. The transmitted sequence is $-1\ -1\ -1\ -1\ -1\ -1\ -1$, if we are transmitting a sequence of logical 0s. At the receiver, the soft outputs of the demodulator are $+0.8,\ -1.2,\ +0.6,\ -2.2,\ -0.4,\ -1.3,\ -0.9$, which corresponds to the sequence of 1 0 1 0 0 0 0 , if we use hard decision decoding, as in our last example.

The demodulator's soft outputs are used as a measure of confidence, which are shown in Figure 2.10. The first demodulator soft output signal is $+0.8$, implying that the transmitted signal is likely to have been $+1$ and the confidence measure of the decision is 0.8. Considering the path $a \rightarrow g$, corresponding to a logical 1, the branch metric of the path is $+0.8$. However, path $a \rightarrow a$ does not tally with the received signal, and the branch metric of the path is therefore -0.8, accumulating a negative path metric, or a 'penalty' due to its dissimilarity. At second instant, the received signal is -1.2 which results in path metrics (accumulated confidence) of $+0.4$, -2.0, $+2.0$ and -0.4 for the paths $a \rightarrow a \rightarrow a$, $a \rightarrow a \rightarrow g$, $a \rightarrow g \rightarrow d$ and $a \rightarrow g \rightarrow f$, respectively.

Let us denote α_1 and α_2 the paths $a \rightarrow a \rightarrow a \rightarrow a \rightarrow a$ and $a \rightarrow g \rightarrow d \rightarrow b \rightarrow a$. The total accumulated path metrics for paths α_1 and α_2 are $+2.0$ and $+0.4$, respectively. The Viterbi decoder selects the path associated with the larger path metric because of its stronger accumulated confidence. Hence, path α_1 is selected (instead of path α_2 which was selected in our previous hard decision example). Hence, it was shown that soft decision decoding performs better than hard decision decoding.

2.3.5 Simulation Results

The following simulation results were obtained using simple Binary Phase Shift Keying (BPSK) over an Additive White Gaussian Noise (AWGN) channel.

2.3.5.1 The Berlekamp-Massey Algorithm

In this section, we characterise the performance of different BCH codes using the Berlekamp-Massey algorithm.

Figures 2.11, 2.12, 2.13 and 2.14 show the performance of the BCH codes having the same codeword length n. In figure 2.14, we can see that the BCH(127,120,3) code, which has a coding rate of about 0.95, exhibits the worst performance at a Bit Error Rate (BER) of 10^{-5}. The BCH(127,113,5) code, which has a coding rate of 0.89, gives a slightly better result than the BCH(127,120,3) scheme, with an improvement of 0.75 dB at BER=10^{-5}. As the coding rate decreases, the performance of the family codes from the BCH$(127, k, t)$ becomes better.

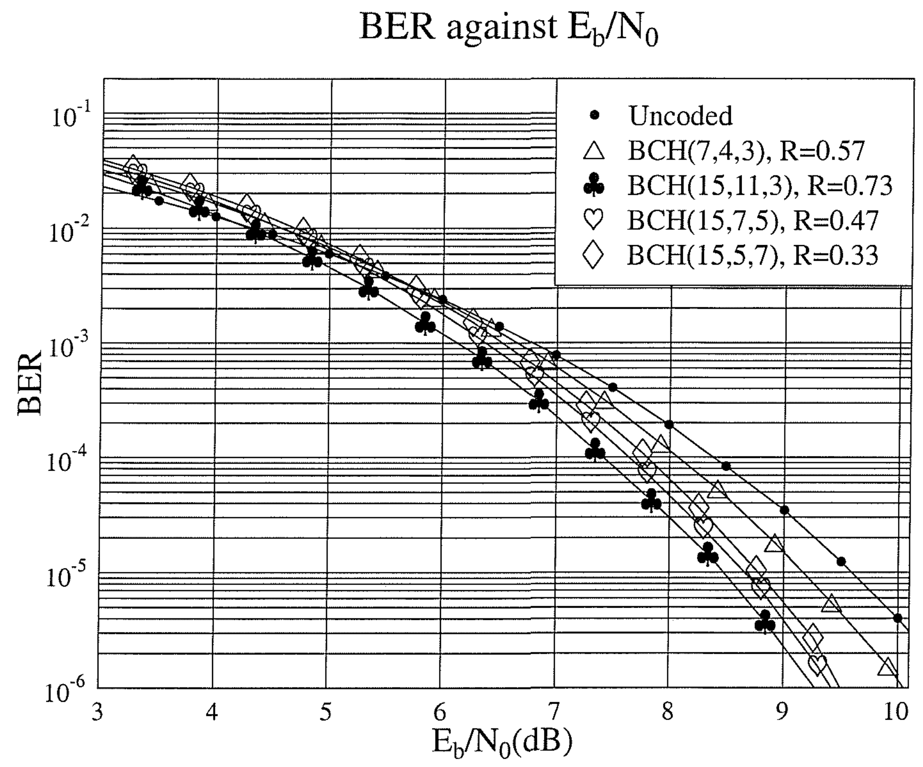


Figure 2.11: BER performance of the BCH codes with n=7 and n=15 over AWGN channels.

However, this is not true when the coding rate decreases to about 0.5 or below since the amount of code redundancy becomes excessive, inevitably reducing the amount of energy per transmitted bit. For example, the BCH(127,71,19) code (code rate=0.56) performs better, than the BCH(127,64,21) code (code rate=0.50). This applies to other families of the BCH code as well. These trends become more explicit in the context of Figure 2.19 and 2.20 which will be explained later.

Figure 2.15 shows our performance comparison for different-rate BCH codes selected from each family. By contrast, Figure 2.16 provides the comparison of a set of near-half-rate codes. From these figures we surmise that the BCH coded performance improves, when n increases.

2.3.5.2 Hard Decision Viterbi Decoding

Figure 2.17 shows the performance of the BCH(31,21,5) and BCH(15,7,5) codes using two different decoding algorithms, namely the Hard-Decision Viterbi Algorithm [HD-VA] and that of the Hard-Decision Berlekamp-Massey algorithm [HD-BM]. The performance of the algorithms appears to be fairly similar.

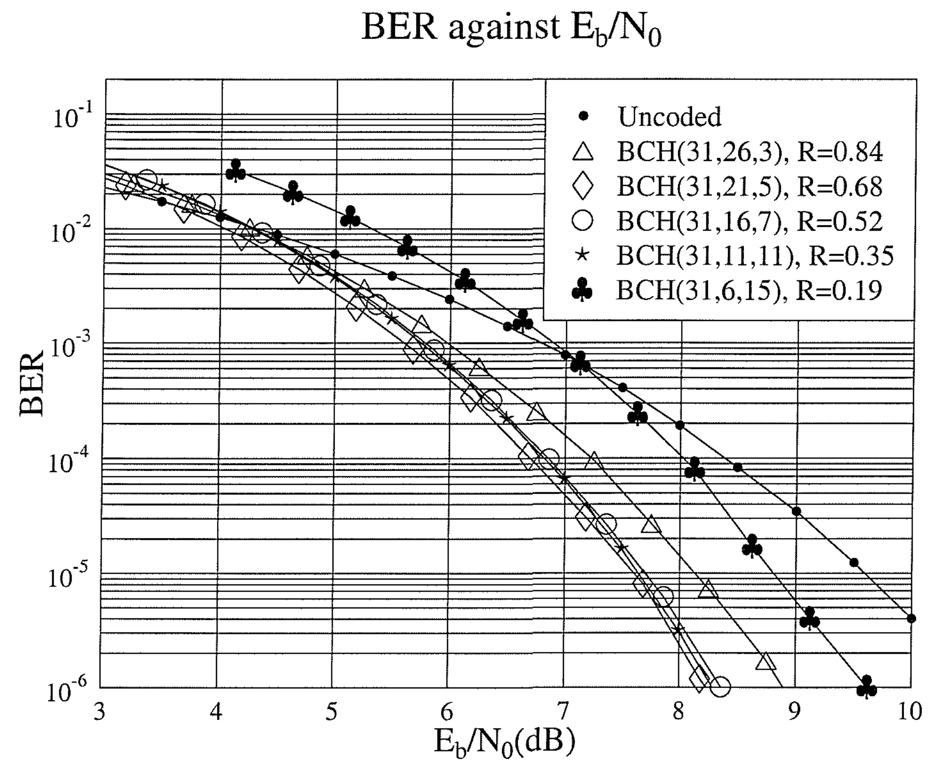


Figure 2.12: BER performance of the BCH codes with n=31 over AWGN channels.

2.3.5.3 Soft Decision Viterbi Decoding

Figure 2.18 shows our performance comparison between the Soft-Decision Viterbi Algorithm [SD-VA] and Hard-Decision Berlekamp-Massey algorithm [HD-BM]. As it is seen in the figure, there is an improvement of about 2 dB at a BER of 10^{-5}.

2.3.6 Conclusion On Block Coding

The performance of a range of BCH codes using the Berlekamp-Massey decoding algorithm has been investigated through simulations. The coding gain of the various BCH codes at BER= 10^{-3} and BER= 10^{-6} over AWGN channels was tabulated in Tables 2.2, 2.3 and 2.4.

Figure 2.19 and 2.20 shows the coding gain against the code rate for different BCH codes at BER= 10^{-3} and BER= 10^{-6}, respectively, suggesting the following conclusions. As k and the code rate $R = \frac{k}{n}$ increases, there is a maximum coding gain for each family (n is constant) of the BCH codes studied. Furthermore, the maximum coding gain is typically found, when the code rate is between 0.5 and 0.6. For example, in Figure 2.20, the maximum

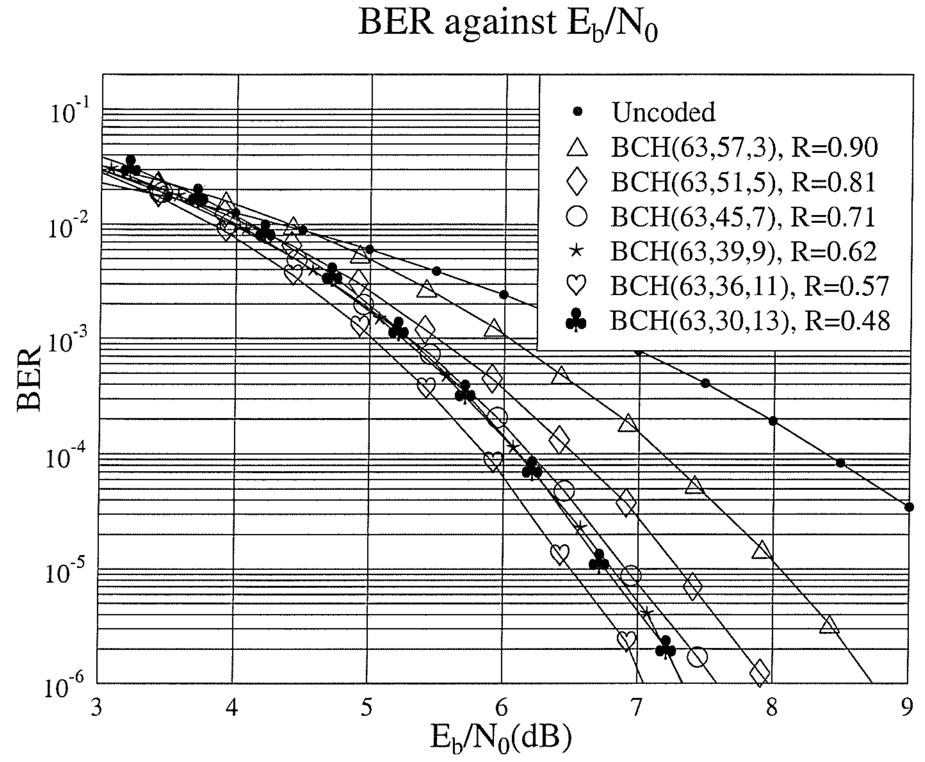


Figure 2.13: BER performance of the BCH codes with n=63 over AWGN channels.

coding gain of the BCH codes having a codeword length of $n = 127$ is 4.1 dB when the code rate is 0.56. For BCH codes with $n = 63$, the maximum coding gain is 3.5 dB when the code rate is 0.57.

Observe the sudden coding gain improvement at BER=10^{-3} in Figure 2.19 at coding rate of $R = 0.23$, as seen in Table 2.4 for the BCH(127,29,43) code. This is a consequence of the relatively modest coding rate reduction from 0.28 to 0.23 in comparison to the BCH(127,36,31) code. Explicitly, while the BCH(127,29,43) code is capable of correcting 21 errors, the BCH(127,36,31) code can only correct 15 errors. The same phenomenon is observed in Figure 2.20 at BER=10^{-6} in the context of these two codes.

2.4 Soft Input Algebraic Decoding

2.4.1 Introduction

In this section we investigate the benefits of using soft inputs in the context of the classic algebraic decoding. The decoding techniques, our simulation results and related conclusions

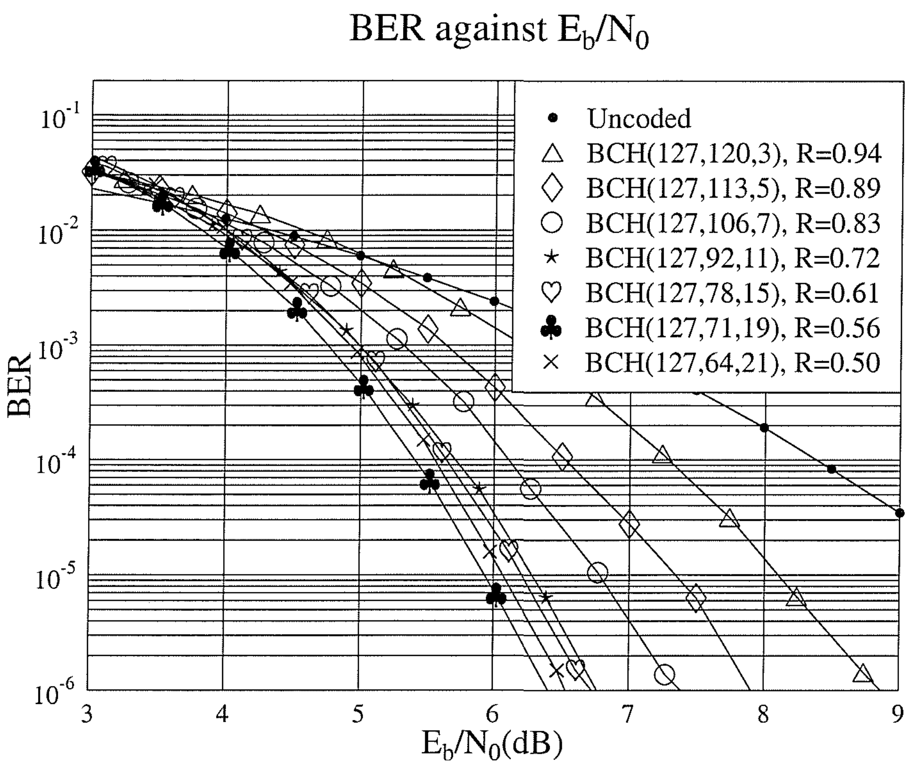


Figure 2.14: BER performance of the BCH codes with n=127 over AWGN channels.

Code	Code Rate R	E_b/N_0 (dB)		Gain (dB)	
		BER			
		10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded	1.00	6.78	10.53	0.00	0.00
BCH(7,4,3)	0.57	6.65	10.05	0.13	0.48
BCH(15,11,3)	0.73	6.12	9.27	0.66	1.26
BCH(15,7,5)	0.47	6.37	9.42	0.41	1.11
BCH(15,5,7)	0.33	6.52	9.50	0.26	1.03
BCH(31,26,3)	0.84	5.98	8.90	0.80	1.63
BCH(31,21,5)	0.68	5.61	8.23	1.17	2.30
BCH(31,16,7)	0.52	5.77	8.37	1.01	2.16
BCH(31,11,5)	0.35	5.75	8.29	1.03	2.24
BCH(31,6,7)	0.19	6.89	9.62	-0.11	0.91

Table 2.2: Coding gain of BCH codes with $n = 7, n = 15$ and $n = 31$ using the Berlekamp-Massey Algorithm over AWGN channels.

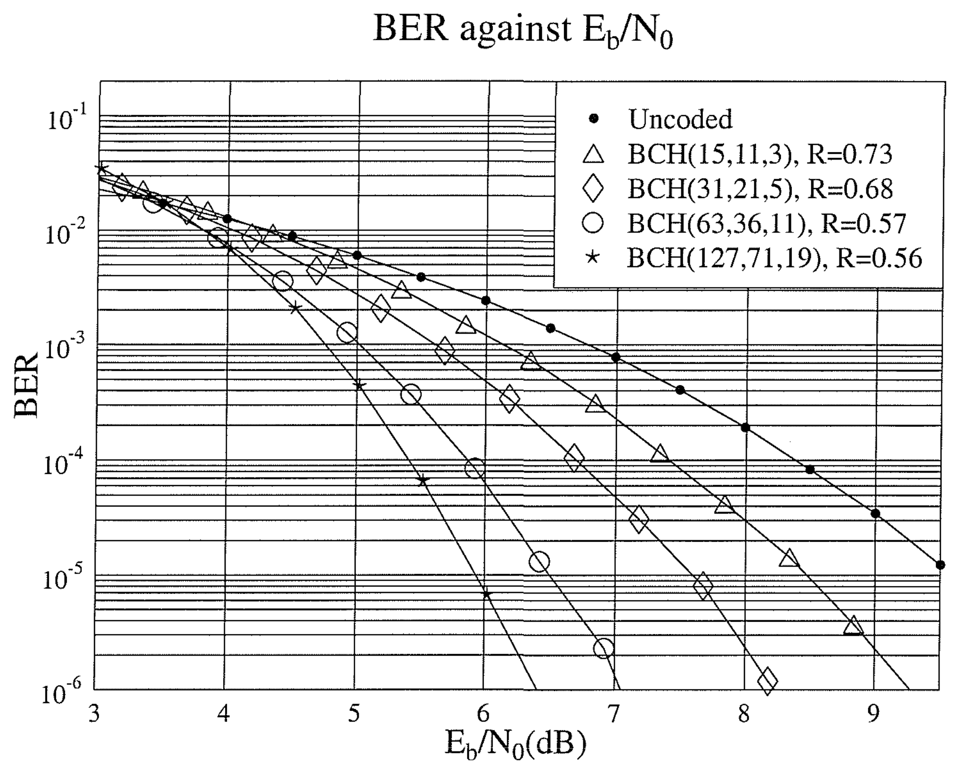


Figure 2.15: BER performance comparison of selected BCH codes over AWGN channels.

Code	Code Rate R	E_b/N_0 (dB)		Gain (dB)	
		BER			
		10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded	1.00	6.78	10.53	0.00	0.00
BCH(63,57,3)	0.90	6.03	8.75	0.75	1.78
BCH(63,51,5)	0.81	5.50	7.97	1.28	2.56
BCH(63,45,7)	0.71	5.29	7.60	1.49	2.93
BCH(63,39,9)	0.62	5.24	7.34	1.54	3.19
BCH(63,36,11)	0.57	5.02	7.05	1.76	3.48
BCH(63,30,13)	0.48	5.28	7.33	1.50	3.20
BCH(63,24,15)	0.38	5.78	7.77	1.00	2.76
BCH(63,18,21)	0.29	5.70	7.80	1.08	2.73
BCH(63,16,23)	0.25	5.80	7.83	0.98	2.70
BCH(63,10,27)	0.16	7.03	9.09	-0.25	1.44
BCH(63,7,31)	0.11	7.76	10.03	-0.98	0.50

Table 2.3: Coding gain of the BCH codes with $n = 63$ using the Berlekamp-Massey algorithm over AWGN channels.

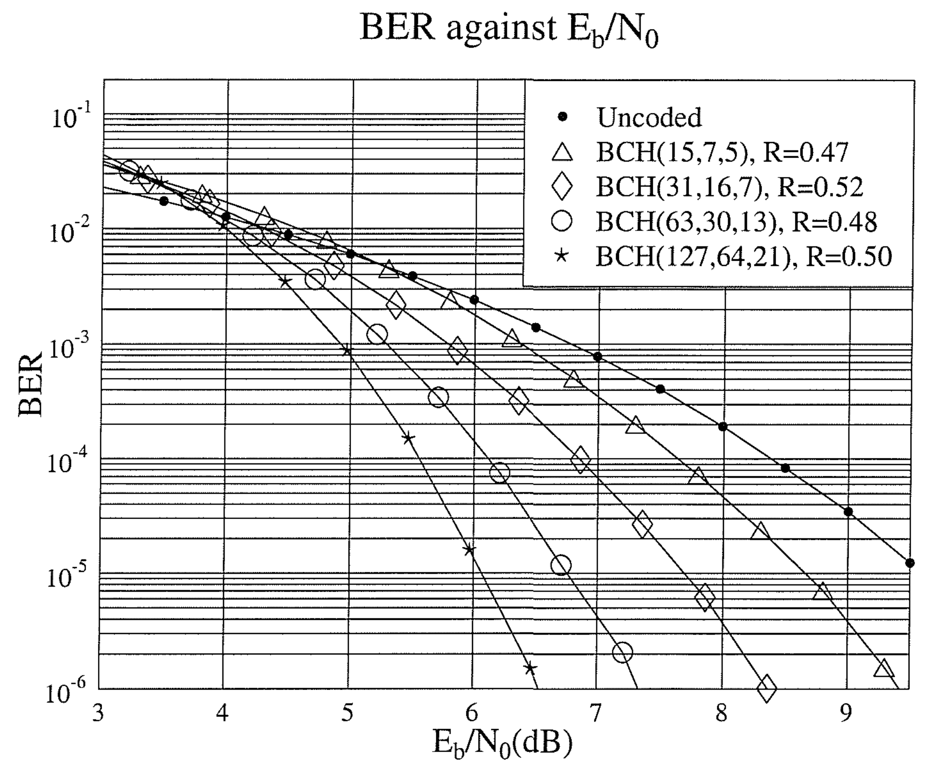


Figure 2.16: BER performance comparison of near-half-rate BCH codes over AWGN channels.

will be outlined in this section. Since the discovery of BCH codes in 1960, numerous algorithms [17,29,31,32,34,114] have been suggested for their decoding. The Berlekamp-Massey algorithm [31–34] is widely recognised as an attractive decoding technique.

However, the Berlekamp-Massey algorithm assumes that the output of the demodulator is binary. This implies that the algorithm is incapable of directly exploiting the soft outputs provided by the demodulator at the receiver. In Section 2.3.5.3, we have shown that for the trellis decoding of BCH codes, there is an improvement of 2 dB, if we use the soft decision Viterbi algorithm rather than the hard decision Viterbi algorithm.

In 1972, Chase [28] invented a class of decoding algorithms that utilise the soft outputs provided by the demodulator. At the receiver the demodulator provides the received signal value y_k, given that the corresponding data bit u_k was either 1 or 0, indicating two different features:

- Its polarity shows, whether u_k is likely to be 1 (positive y_k) or 0 (negative y_k).
- Its magnitude $|y_k|$ indicates the confidence measure provided by the demodulator.

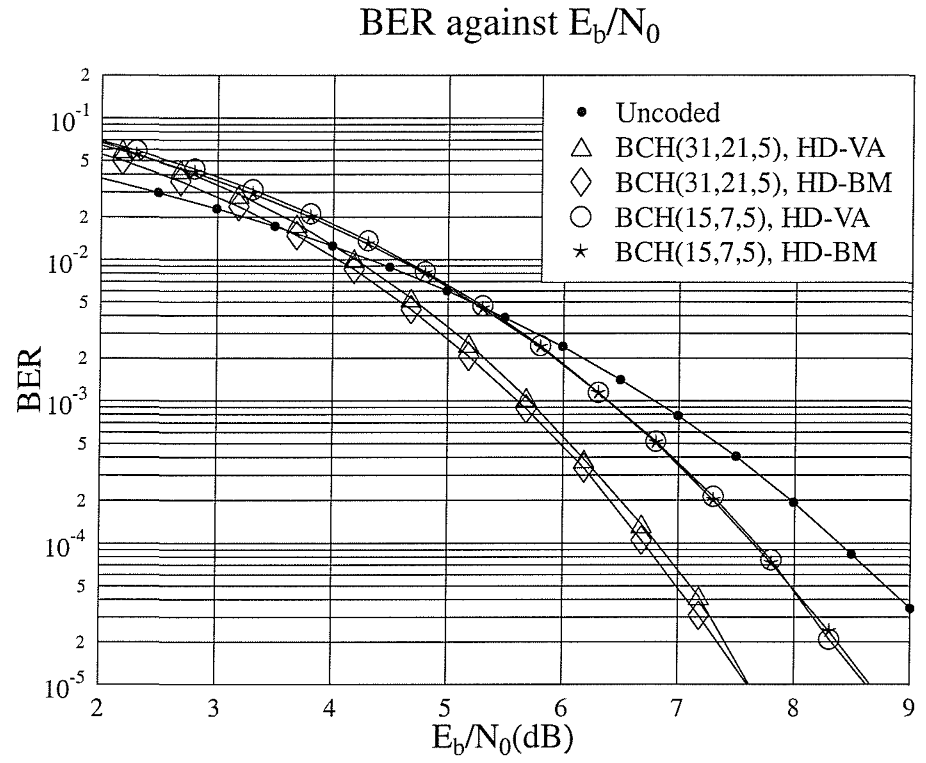


Figure 2.17: BER comparison between hard decision Viterbi decoding and Berlekamp-Massey decoding for various BCH codes and decoding algorithms over AWGN channels.

As mentioned earlier, the hard-decision based Berlekamp-Massey algorithm only utilises the binary bit provided by the demodulator. The error correcting capability t of the BCH(n, k, d_{min}) code is related to the minimum Hamming distance d_{min} between the codewords. In general, the error correcting capability, t, of the BCH code is defined as the maximum number of guaranteed correctable errors per codeword, given by [96]:

$$t = \lfloor \frac{d_{min} - 1}{2} \rfloor , \tag{2.3}$$

where $\lfloor i \rfloor$ means the largest integer not exceeding i.

Figure 2.21 shows a stylised example of conventional algebraic decoding. There are four valid codewords $c_1, ..., c_4$ shown in Figure 2.21 and the minimum separation between them is d_{min}. Each codeword is surrounded by a decoding sphere of radius t. Let us assume that we have transmitted two identical BCH codewords, say $\underline{c}_2$, over the noisy channel. The associated received vector of n binary bits $\underline{z}_1$ and $\underline{z}_2$, is provided by the demodulator. As we can see, $\underline{z}_1$ is not within the decoding sphere of any valid codeword. Hence, the conventional algebraic decoder is incapable of correcting the errors in $\underline{z}_1$. On the other hand, the binary n-tuple $\underline{z}_2$ falls within the decoding sphere of codeword $\underline{c}_4$ and hence it

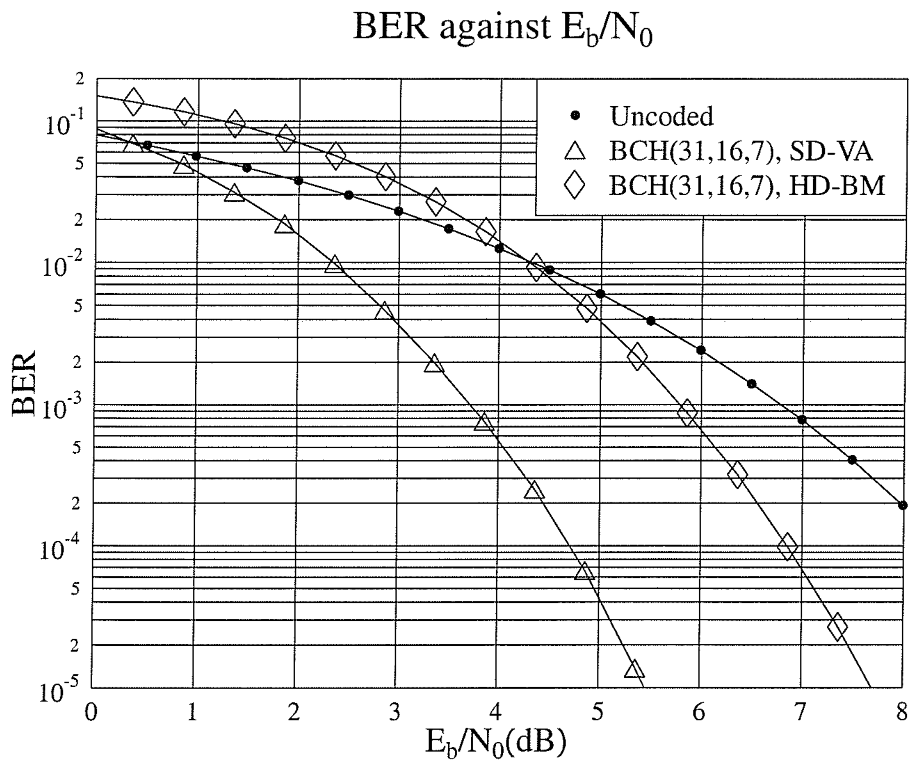


Figure 2.18: BER comparison between soft decision Viterbi decoding and Berlekamp-Massey decoding over AWGN channels.

is decoded to the valid codeword $\underline{c}_4$. However, if we additionally consider the soft-decision based confidence measures $|\underline{y}_k|$ provided by the demodulator, the decoder might be able to correct more than t errors in the n-tuple $\underline{z}_1$. Moreover, the received n-tuple $\underline{z}_2$ might be more likely to be due to codeword $\underline{c}_2$ rather than $\underline{c}_4$. These problems are circumvented by the Chase algorithm of the forthcoming section.

2.4.2 Chase Algorithms

Figure 2.22 shows the geometric sketch of the decoding process aided by channel measurement information, which is elaborated on below. Accordingly, the received binary n-tuple $\underline{z}_1$ is perturbed with the aid of a set of test patterns TP, which is a binary sequence that contains 1s in the location of the bit positions that are to be tentatively inverted. By adding this test pattern, modulo two, to the received binary sequence, a new binary sequence $\underline{z}'_1$ is obtained:

$$\underline{z}'_1 = \underline{z}_1 \oplus TP \; . \tag{2.4}$$

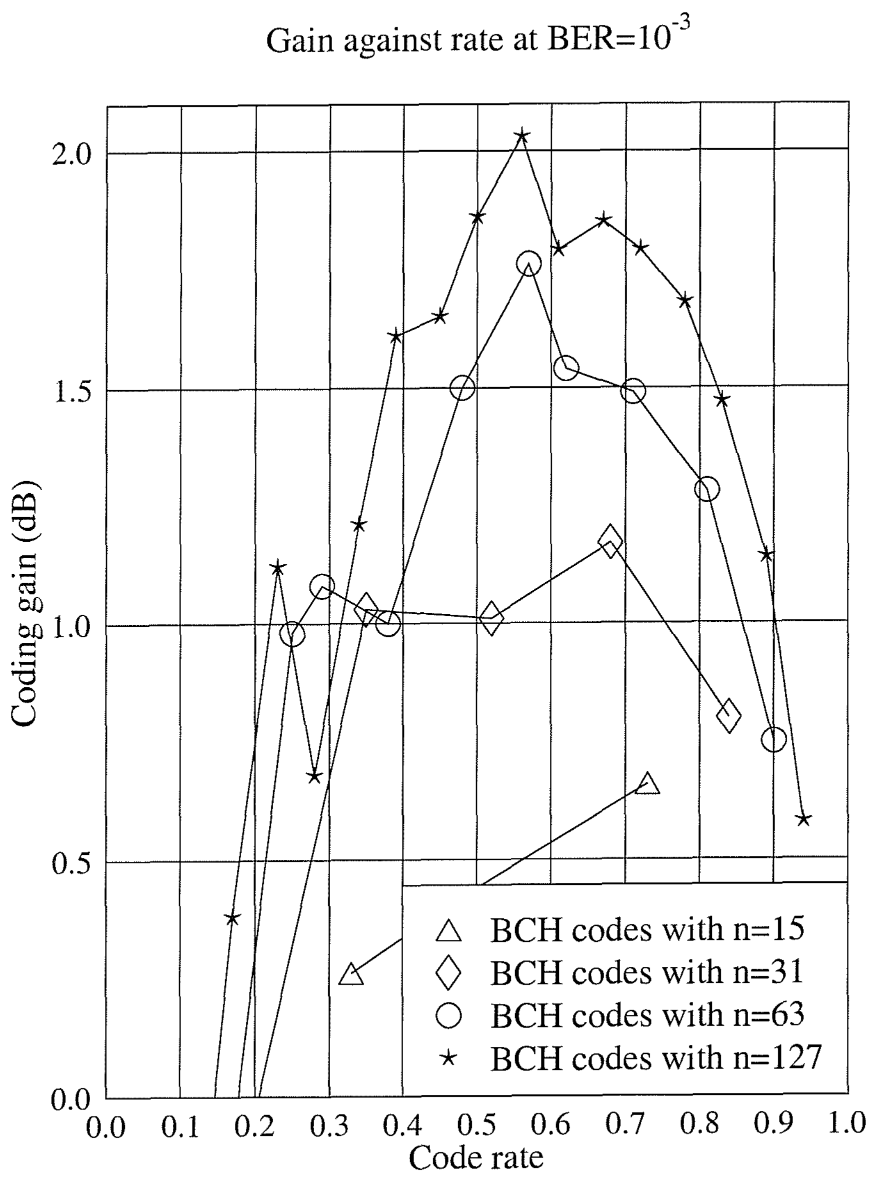


Figure 2.19: Coding gain against code rate for different BCH codes at BER= 10^{-3} employing Berlekamp-Massey algorithm over AWGN channels using the codes summarised in Table 2.1.

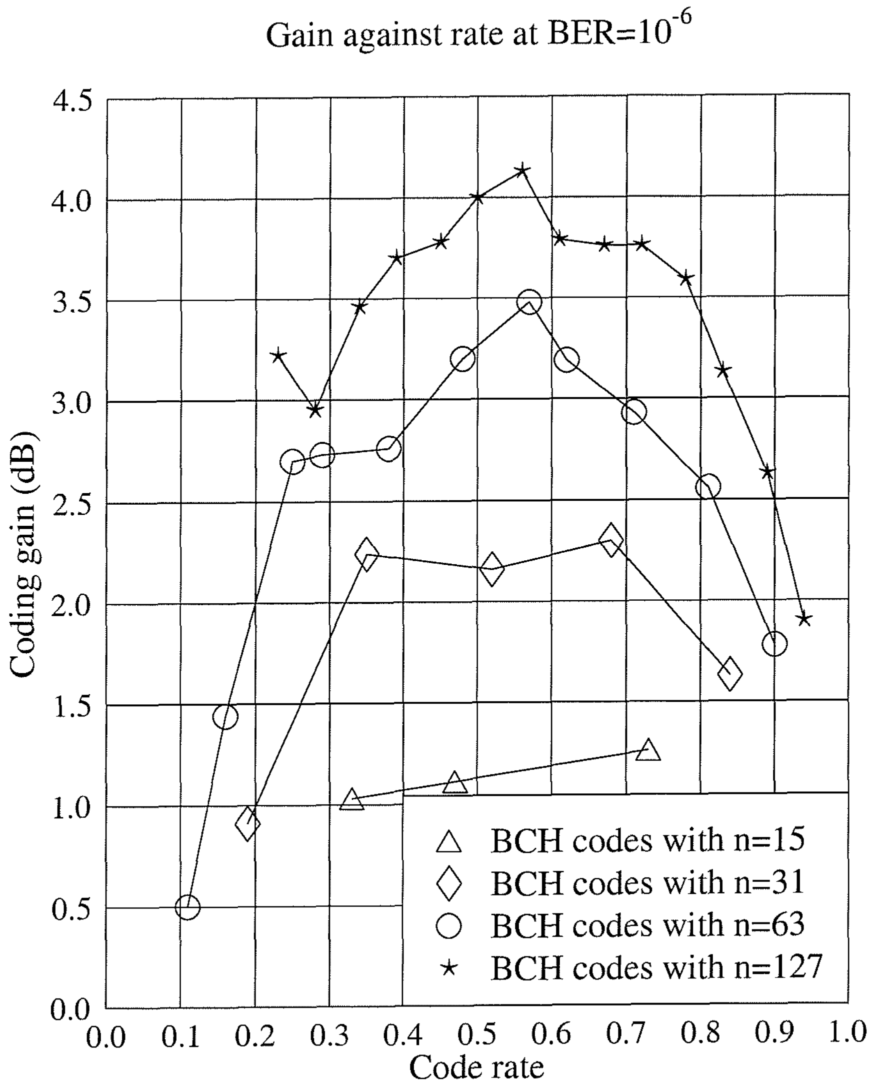


Figure 2.20: Coding gain against code rate for different BCH codes at BER= 10^{-6} employing Berlekamp-Massey algorithm over AWGN channels using the codes summarised in Table 2.1.

Code	Code Rate R	E_b/N_0 (dB)		Gain (dB)	
		BER			
		10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded	1.00	6.78	10.53	0.00	0.00
BCH(127,120,3)	0.94	6.20	8.63	0.58	1.90
BCH(127,113,5)	0.89	5.64	7.90	1.14	2.63
BCH(127,106,7)	0.83	5.31	7.40	1.47	3.13
BCH(127,99,9)	0.78	5.10	6.94	1.68	3.59
BCH(127,92.11)	0.72	4.99	6.77	1.79	3.76
BCH(127,85,13)	0.67	4.93	6.77	1.85	3.76
BCH(127,78,15)	0.61	4.99	6.74	1.79	3.79
BCH(127,71,19)	0.56	4.75	6.40	2.03	4.13
BCH(127,64,21)	0.50	4.92	6.53	1.86	4.00
BCH(127,57,23)	0.45	5.13	6.75	1.65	3.78
BCH(127,50,27)	0.39	5.17	6.83	1.61	3.70
BCH(127,43,29)	0.34	5.57	7.07	1.21	3.46
BCH(127,36,31)	0.28	6.10	7.58	0.68	2.95
BCH(127,29,43)	0.23	5.66	7.31	1.12	3.22
BCH(127,22,47)	0.17	6.40	8.08	0.38	2.45
BCH(127,15,55)	0.12	7.20	8.88	-0.42	3.25
BCH(127,8,63)	0.06	9.05	10.86	-2.27	-0.33

Table 2.4: Coding gain of the BCH codes with $n = 127$ using the Berlekamp-Massey algorithm over AWGN channels.

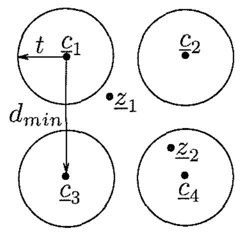


Figure 2.21: Stylised example of conventional algebraic decoding.

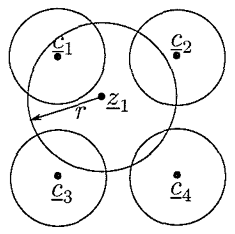


Figure 2.22: Stylised illustration of the Chase algorithm.

As shown in Figure 2.22, r represents the maximum Hamming distance of the perturbed binary received sequence $\underline{z}_1'$ from the original binary received sequence $\underline{z}_1$. By using a number of test patterns, the perturbed received binary sequence $\underline{z}_1'$ may fall within the decoding sphere of a number of valid BCH codewords. If we increase r, the perturbed received sequence $\underline{z}_1'$ will fall within the decoding sphere of more valid BCH codewords.

If the perturbed received binary sequence $\underline{z}_1'$ falls within the decoding sphere of a valid BCH codeword c_1, by invoking algebraic decoding a new error pattern $\underline{e}'$ is obtained, which may be an all-zero or a non-zero tuple. The actual error pattern $\underline{e}$ associated with the binary received sequence $\underline{z}_1$ is given by

$$\underline{e} = \underline{e}' \oplus TP, \tag{2.5}$$

which may or may not be different from the original test pattern TP, depending on whether the perturbed received binary sequence $\underline{z}_1'$ falls into the decoding sphere of a valid codeword. However, only those perturbed received binary sequences $\underline{z}_1'$ that fall into the decoding sphere of a valid codeword are considered.

A maximum-likelihood decoder is capable of finding the codeword that satisfies

$$\min_{m} \text{weight}(\underline{z} \oplus c_m) \ , \tag{2.6}$$

where the range of m is over all possible codewords. Based on similar principles, Chase [28] defined a new channel decoder. However, for the sake of low complexity only a certain limited set of valid codewords are considered by Chase's technique, namely those surrounded by the decoding spheres that the perturbed received binary sequence $\underline{z}_1'$ may fall into. In this case, we are concerned with finding the error pattern $\underline{e}$ of minimum analogue weight, where the analogue weight of an error sequence $\underline{e}$ is defined as

$$W(\underline{e}) = \sum_{i=1}^{n} e_i |y_i| \ . \tag{2.7}$$

The Chase algorithm can be summarised in the flow-chart shown in Figure 2.23. Each time, the algorithm considers an n-tuple codeword of the BCH code, which is constituted by n number of the received bits $\underline{z}$ and their soft metrics $\underline{y}$. The received bits $\underline{z}$ and their confidence values $\underline{y}$ are assembled, which is the first step shown in Figure 2.23. Then, a set of test patterns TP is generated. For each test pattern, a new sequence $\underline{z}'$ is obtained by modulo two addition of the particular test pattern TP and the received sequence $\underline{z}$. The conventional algebraic decoder is invoked to decode the new sequence $\underline{z}'$, as seen in Figure 2.21. If the conventional algebraic decoder found a non-zero error pattern $\underline{e}'$, we are able to find the actual error pattern $\underline{e}$, using Equation 2.5, associated with the received

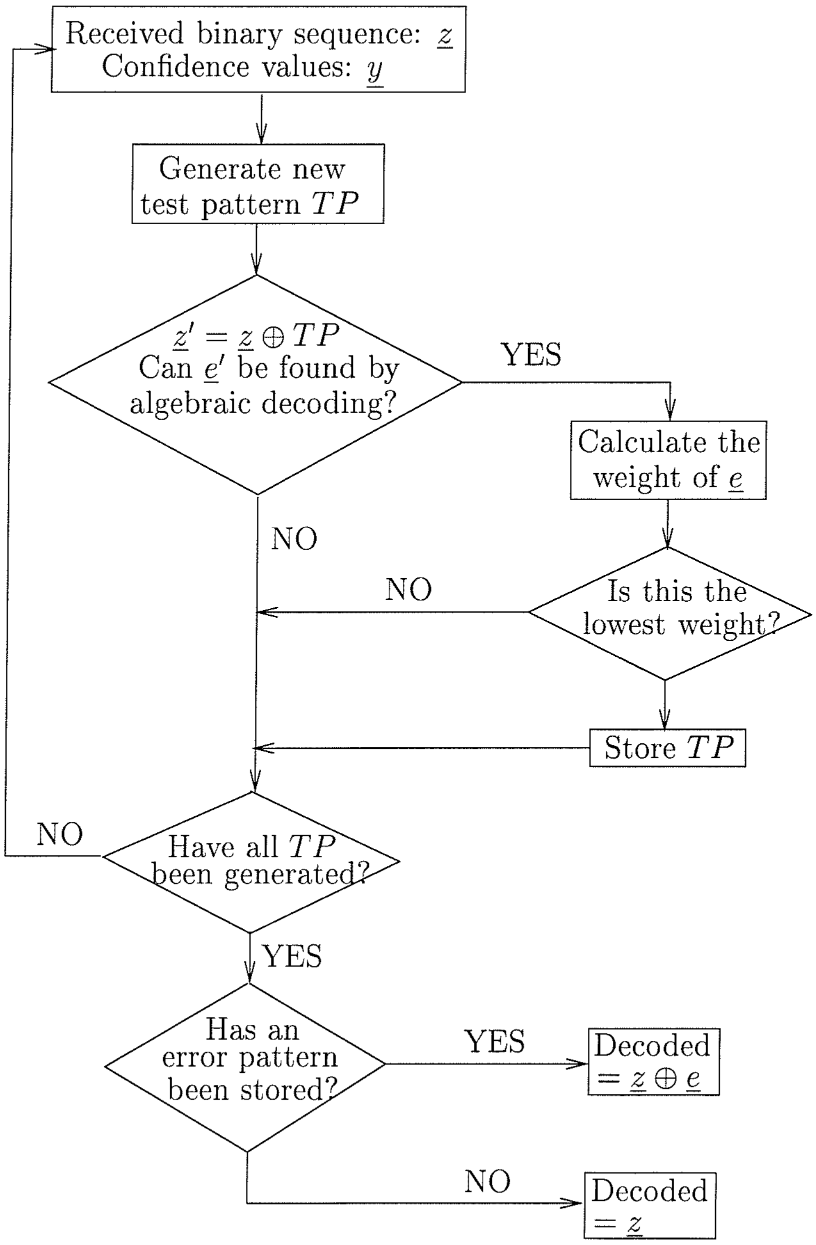

Received binary sequence: $\underline{z}$
Confidence values: $\underline{y}$
Generate new test pattern TP
$\underline{z}' = \underline{z} \oplus TP$
Can $\underline{e}'$ be found by algebraic decoding?
YES
Calculate the weight of $\underline{e}$
NO
NO
Is this the lowest weight?
Store TP
NO
Have all TP been generated?
YES
Has an error pattern been stored?
YES
Decoded $= \underline{z} \oplus \underline{e}$
NO
Decoded $= \underline{z}$

Figure 2.23: Flow Chart of Chase algorithm.

binary sequence $\underline{z}$. Using Equation 2.7, the analogue weight W of the actual error pattern $\underline{e}$ can be calculated. The generated test pattern TP will be stored in the memory, if the associated analogue weight W is found to be the lowest. The above procedure will be repeated for every test pattern generated. Upon completing the loop in Figure 2.23, the memory is checked. If there is an error pattern stored, the binary decoded sequence will be $\underline{z} \oplus \underline{e}$. Otherwise, the binary decoded sequence is the same as the received sequence $\underline{z}$.

The number of test patterns used can be varied according to the tolerable complexity, which also has an effect on the achievable performance. In the following sections, we present two variants of this algorithm, namely the Chase Algorithm 1 and Chase Algorithm 2. The nature of both algorithms depends essentially on the number of test patterns used.

2.4.2.1 Chase Algorithm 1

For this particular algorithm, typically a large set of test patterns TP is considered. In fact, the algorithm considers the entire set of possible test patterns within a sphere of radius $r = d_{min} - 1$ surrounding the received binary sequence $\underline{z}$. Thus, all possible test patterns of binary weight less than or equal to $d_{min} - 1$ are considered.

Let us illustrate the operation of the algorithm with the aid of an example, where the BCH(7,4,3) code is used, which has $d_{min} = 3$. Hence, all the test patterns having a binary weight less than or equal to $d_{min} - 1 = 2$ are generated, which is the second step in Figure 2.23. A fraction of the test patterns TP, which have a binary weight less than or equal to 1 are shown below:

$$\begin{aligned}
TP_0 &= 0\,0\,0\,0\,0\,0\,0 \\
TP_1 &= 0\,0\,0\,0\,0\,0\,1 \\
TP_2 &= 0\,0\,0\,0\,0\,1\,0 \\
TP_3 &= 0\,0\,0\,0\,1\,0\,0 \\
TP_4 &= 0\,0\,0\,1\,0\,0\,0 \\
TP_5 &= 0\,0\,1\,0\,0\,0\,0 \\
TP_6 &= 0\,1\,0\,0\,0\,0\,0 \\
TP_7 &= 1\,0\,0\,0\,0\,0\,0\,.
\end{aligned} \tag{2.8}$$

Using the test patterns TP, which have binary weights equal to 1, we are also able to generate the test patterns that have binary weights larger than 1.

Let us assume that BPSK is used and the transmitted sequence is $-1\ -1\ -1\ -1\ -1\ -1\ -1$. The soft demodulator outputs $\underline{y}$, the hard decision decoded binary sequence $\underline{z}$ and the confidence measures $|\underline{y}|$ are shown in Table 2.5.

i	0	1	2	3	4	5	6
y_i	-0.9	-1.3	-0.4	-2.2	+0.6	-1.2	+0.8
z_i	0	0	0	0	1	0	1
δ_i	0.9	1.3	0.4	2.2	0.6	1.2	0.8

Table 2.5: Example of soft demodulator outputs, hard-decision decoded sequence and confidence measures.

Using for example the second test pattern TP_1 in Equation 2.8, the perturbed received sequence $\underline{z}'$ is

$$\begin{aligned} \underline{z}' &= \underline{z} \oplus TP_1 \\ &= 0\,0\,0\,0\,1\,0\,1 \oplus 0\,0\,0\,0\,0\,0\,1 \\ &= 0\,0\,0\,0\,1\,0\,0\,. \end{aligned} \tag{2.9}$$

Due to this perturbation, we now have a sequence within a Hamming distance of one from a legitimate codeword, namely the 0 0 0 0 0 0 0 sequence, and hence the perturbed received binary sequence $\underline{z}'$ is decoded by the algebraic decoder and the decoded sequence is 0 0 0 0 0 0 0. Therefore, the associated error sequence $\underline{e}'$ is

$$\underline{e}' = 0\,0\,0\,0\,1\,0\,0\,, \tag{2.10}$$

and the actual error sequence $\underline{e}$ is

$$\begin{aligned} \underline{e} &= \underline{e}' \oplus TP_1 \\ &= 0\,0\,0\,0\,1\,0\,0 \oplus 0\,0\,0\,0\,0\,0\,1 \\ &= 0\,0\,0\,0\,1\,0\,1\,. \end{aligned} \tag{2.11}$$

As we will show below, this is the most likely error sequence $\underline{e}$, allowing us to correct two, rather than just one error, which was a limitation of the hard-decision Berlekamp-Massey algorithm.

In order to quantify the probability of the possible error sequences, their analogue weight is determined next. The analogue weight of the actual error sequence $\underline{e}$ is

$$\begin{aligned} W_{TP_1}(\underline{e}) &= \sum_{i=1}^{n} \delta_i e_i \\ &= 0.9 \times 0 + 1.3 \times 0 + 0.4 \times 0 + 2.2 \times 0 + 0.6 \times 1 + 1.2 \times 0 + 0.8 \times 1 \\ &= 1.4\,. \end{aligned} \tag{2.12}$$

i	0	1	2	3	4	5	6	7
TP_i	TP_0	TP_1	TP_2	TP_3	TP_4	TP_5	TP_6	TP_7
$W_{TP_i}(\underline{e})$	2.2	1.4	1.6	1.4	2.2	1.6	2.2	2.2

Table 2.6: Analogue weights associated with the test patterns in Equation 2.8, for the BCH(7,4,3) Chase decoding example.

This process is repeated for other test patterns of binary weight less than or equal to $d_{min} - 1$ as shown in Equation 2.8. The analogue weights associated with each test pattern in Equation 2.8 are shown in Table 2.6. The lowest possible analogue weight is 1.4, taking into consideration all test patterns of binary weight less than or equal to $d_{min} - 1$. As we can see from Table 2.6, test patterns TP_1 and TP_3 have produced the lowest analogue weight, which is 1.4. It can be readily shown that the associated actual error pattern $\underline{e}$ of both test patterns TP_1 and TP_3 is the same. Therefore, the Chase-decoded sequence $\hat{\underline{z}}$ is

$$\begin{aligned}\hat{\underline{z}} &= \underline{e}_1 \oplus \underline{z} \\ &= 1\,0\,1\,0\,0\,0\,0 \oplus 1\,0\,1\,0\,0\,0\,0 \\ &= 0\,0\,0\,0\,0\,0\,0\,. \end{aligned} \tag{2.13}$$

Computer simulations have shown that the performance of this algorithm is similar to that of the soft decision Viterbi algorithm.

The BCH(7,4,3) code has a minimum free distance d_{min} of 3 and the number of possible test patterns is 29. If we employ the algorithm to decode the BCH(31,21,5) code, the minimum free distance d_{min} is now 5. Hence, we need to consider all the test patterns having a binary weight less than or equal to 4. In this case, the number of test patterns is over 36000. Generally, the number of test patterns increases exponentially with n and d_{min}.

2.4.2.2 Chase Algorithm 2

For this variant of the Chase algorithm, a considerably smaller set of possible error patterns is used. Only the positions of the $\lfloor \frac{d_{min}}{2} \rfloor$ lowest confidence measures are considered. The test patterns TP have any combination of 1s, which are located in the $\lfloor \frac{d_{min}}{2} \rfloor$ positions of the lowest confidence values. Hence, there are only $2^{\lfloor \frac{d_{min}}{2} \rfloor}$ possible test patterns, including the all-zero pattern.

For the BCH(7,4,3) code, the number of legitimate test positions is equal to one and the number of test patterns is two. Computer simulations have shown that the BER performance curve of this simplified algorithm is about a quarter of a dB worse, than that of the soft decision Viterbi algorithm. If the number of test positions were equal to three, i.e. there were eight test patterns, then the performance would be the same as that of the soft decision Viterbi algorithm.

Using the same example as in Section 2.4.2.1, we have to search for the three test positions associated the lowest confidence measures. The eight test patterns are

$$\begin{aligned}
TP_0 &= 0\,0\,0\,0\,0\,0\,0 \\
TP_1 &= 0\,0\,0\,0\,0\,0\,1 \\
TP_2 &= 0\,0\,0\,0\,1\,0\,0 \\
TP_3 &= 0\,0\,0\,0\,1\,0\,1 \\
TP_4 &= 0\,0\,1\,0\,0\,0\,0 \\
TP_5 &= 0\,0\,1\,0\,0\,0\,1 \\
TP_6 &= 0\,0\,1\,0\,1\,0\,0 \\
TP_7 &= 0\,0\,1\,0\,1\,0\,1\,,
\end{aligned} \tag{2.14}$$

while their associated analogue weights are summarised in Table 2.7. Notice that in the Chase Algorithm 2 the associated error sequence $\underline{e}'$ of test pattern TP_3 is not considered.

i	0	1	2	3	4	5	6	7
TP_i	TP_0	TP_1	TP_2	TP_3	TP_4	TP_5	TP_6	TP_7
$W_{TP_i}(\underline{e})$	2.2	1.4	1.4	-	1.6	2.5	1.9	1.4

Table 2.7: Analogue weights, associated with the test patterns in Equation 2.14, for the BCH(7,4,3) example.

Computer simulations have shown that a better performance is achieved, when the number of test positions increases, approaching that of the soft decision Viterbi algorithm, as the number of test positions approaches d_{min}.

In closing, we note that unlike in the Chase Algorithm 1, the number of test patterns does not increase, when n increases. However, the associated algorithmic complexity increases exponentially, if the number of test positions increases.

2.4.3 Simulation Results

Simulation results were obtained using simple BPSK over an AWGN channel. Figure 2.24

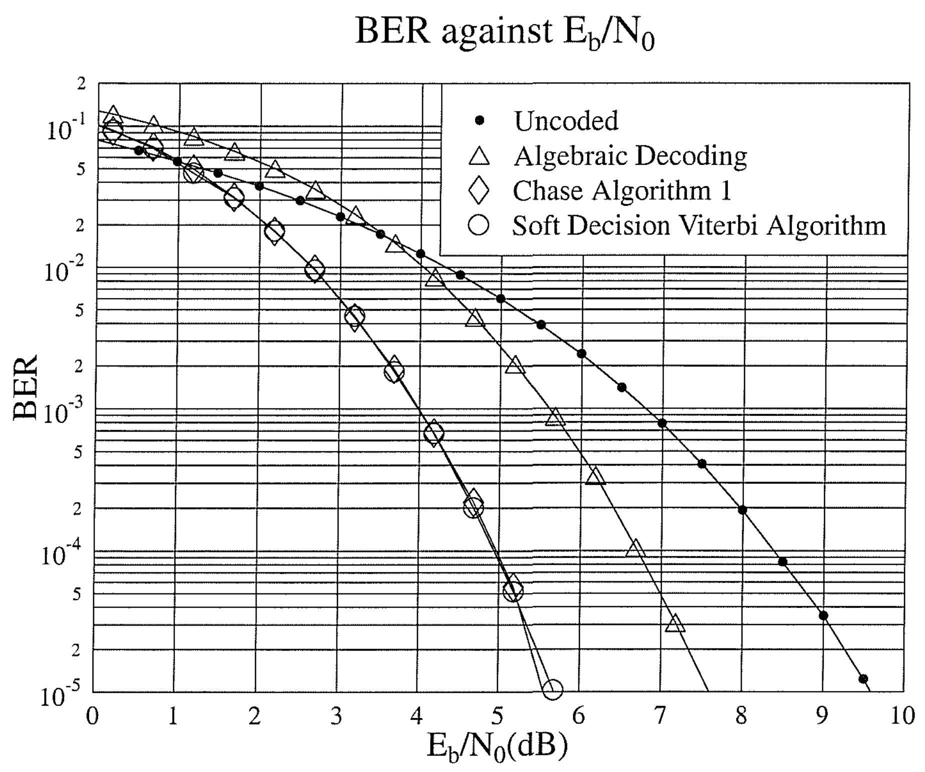


Figure 2.24: Performance comparison between algebraic decoding, the Chase Algorithm 1 and the soft decision Viterbi algorithm using the BCH(31,21,5) code over AWGN channels.

shows our performance comparison between the algorithms considered namely algebraic decoding, the Chase Algorithm 1 and the soft decision Viterbi algorithm using the BCH(31,21,5) code. It is shown in Figure 2.24 that the performance of the Chase Algorithm 1 and that of the soft decision Viterbi algorithm is identical.

Figure 2.25 portrays our performance comparison between algebraic decoding, the Chase Algorithm 2 and the soft decision Viterbi algorithm using the more powerful BCH(31,21,5) code. For the Chase Algorithm 2, the performance of using $\lfloor \frac{d_{min}}{2} \rfloor = 2$ test positions is 1.25 dB better, than that of the algebraic decoding. As the number of test positions increases, the performance of the Chase Algorithm 2 improves. However, the relative improvement becomes smaller, as the number of test positions increases. It is shown in Figure 2.25 that the performance lower bound, i.e. the best possible performance of the Chase Algorithm 2, is the same as that of the soft decision Viterbi algorithm. The lower bound is achieved, when the maximum number of test positions is equal to the minimum free distance d_{min} of the BCH code.

Often it is impractical to implement the Chase Algorithm 1, because the number of test

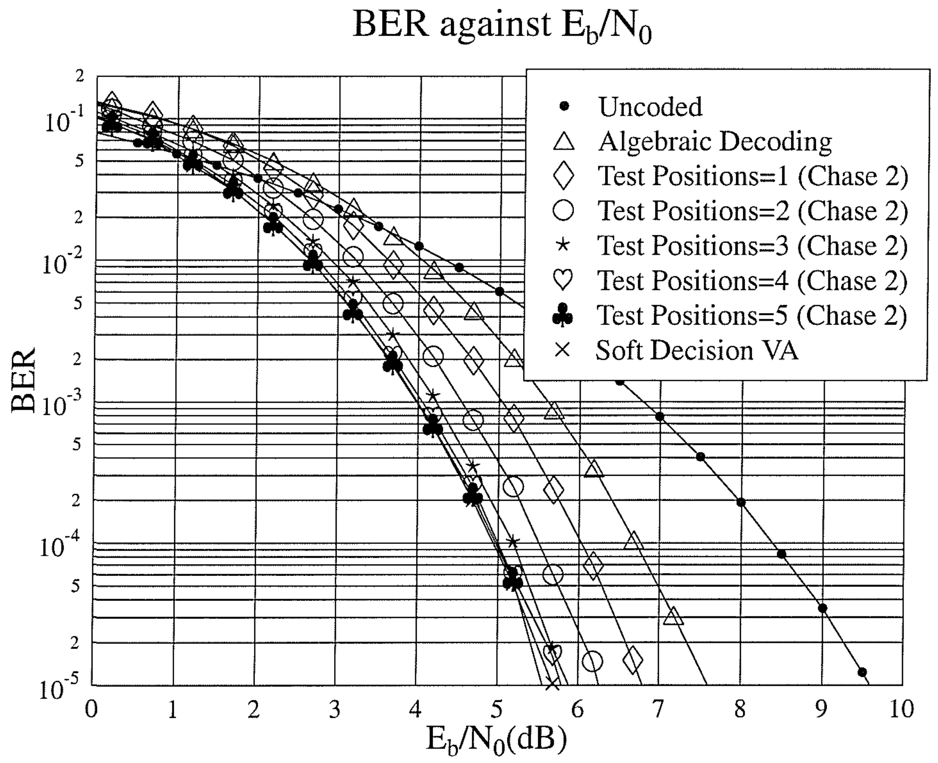


Figure 2.25: Performance comparison between algebraic decoding, the Chase Algorithm 2 (different number of test positions) and the soft decision Viterbi algorithm using the BCH(31,21,5) code over AWGN channels.

patterns increases exponentially, when n and d_{min} increase. However, a similar performance can be achieved using the Chase Algorithm 2 and the maximum number of test positions is equal to the minimum free distance d_{min} of the code. The Chase Algorithm 2 is less complex to implement than trellis decoding, if d_{min} is small. However, the number of test patterns increases exponentially, when d_{min} increases.

2.5 Summary and Conclusion

This chapter served as a brief introduction to the trellis and Chase decoding of BCH codes, before we consider the more complex and novel turbo BCH codes in the next chapter. We commenced with the introduction of BCH codes and the associated definitions in Section 2.2. The detailed encoding process of systematic BCH codes was described in Section 2.2.1. We also showed in Section 2.2.2 that the BCH encoder could be implemented using its state diagram. The conventional Viterbi decoding algorithm was employed in Section 2.3.2 for decoding BCH codes. Several examples of hard decision and soft decision Viterbi decoding were given in the section as well. In Section 2.3.5, we presented simulation results for both the Viterbi and Berlekamp-Massey decoding algorithms. It was found that the performance of employing the soft decision Viterbi decoding algorithm is about 2 dB better, than that of hard decision decoding. The coding gains of various BCH codes having the same codeword length were evaluated and tabulated in Tables 2.2, 2.3 and 2.4. Then, the coding gain versus code rate was plotted in Figure 2.19 and 2.20 for different BCH codes having the same codeword length n. From both figures, we inferred that the maximum coding gain was achieved, when the code rate was between 0.5 and 0.6.

Viterbi decoding of the BCH codes becomes prohibitively complex if $n - k$ increases, since the number of decoding states increases exponentially. The Chase algorithm, which offers a lower complexity associated with a slight performance degradation was detailed in Section 2.4.2. Two variants of the Chase algorithm were described and the Chase Algorithm 2 was found more favourable due to its lower complexity. A detailed example of the Chase decoding process was given in Section 2.4.2. Our simulation results were presented in Section 2.4.3. It was found that there is a lower and upper performance bound for the Chase algorithms. The upper bound performance is the same as that of the conventional algebraic decoding, whereas, the lower bound performance is the same as that of the soft decision Viterbi algorithm. For the Chase Algorithm 2, the lower performance bound can be achieved, if the maximum number of test positions is equal to the minimum free distance d_{min} of the code. However, the complexity of the Chase Algorithm 2 increases exponentially with the minimum free distance d_{min}.

Chapter 3

Turbo BCH Coding

3.1 Introduction

Turbo coding [12] is a novel form of channel coding capable of achieving a performance near the Shannon limit [1]. Generally, so-called Recursive Systematic Convolutional (RSC) codes are used as their component codes. However, block codes can also be employed as their component codes and they have been shown for example by Hagenauer [61, 63] to perform impressively even at near-unity coding rates. Block codes are typically more appropriate for near-unity coding rates, since lower-rate block-based turbo codes exhibit a high decoding complexity. Hence usually convolutional constituent code based turbo codes are used for coding rates lower than 2/3 [61]. In this chapter, we will concentrate entirely on the standard turbo code structure using binary BCH codes as the component codes.

Block-coding based turbo codes can be decoded either using algebraic decoding principles [115–117] or employing trellis-based decoding [61, 63]. However, only trellis-based turbo decoding will be discussed in this chapter.

Following a rudimentary introduction to the turbo encoder and decoder structures as well as to the employment of the log likelihood ratio and soft channel outputs, we embark on the detailed derivation of the Maximum A-Posteriori (MAP) algorithm [11] in Section 3.3.3. In order to reduce its complexity, the MAP algorithm is then modified, in order to derive the Max-Log-MAP and Log-MAP algorithms [50–52]. Beside this, the Soft Output Viterbi Algorithm (SOVA) algorithm is discussed in detail in Section 3.3.5. A simple example of turbo decoding is then given in Section 3.4. In Section 3.5, we propose a novel MAP algorithm for the family of extended BCH codes. The extended MAP algorithm is then also simplified for deriving the extended Max-Log-MAP and Log-MAP algorithms. Finally, various simulations are presented in Section 3.6.

3.2 Turbo Encoder

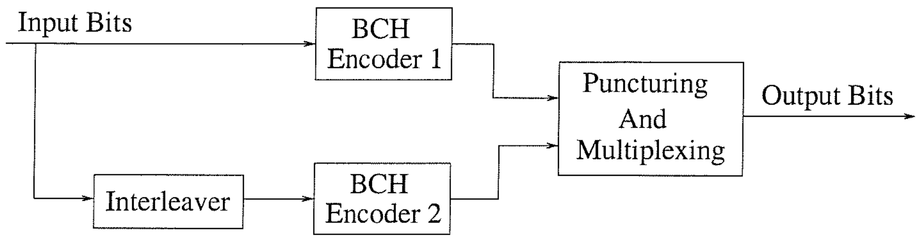


Figure 3.1: Turbo encoder schematic.

The basic structure of the turbo BCH encoder is shown in Figure 3.1. Two BCH encoders, which we have been introduced in Section 2.2.1, are used and an interleaver is placed before the second BCH encoder in Figure 3.1. A number of interleaving techniques, such as block interleaving and random interleaving [68, 112] could potentially be employed for ensuring that the two BCH encoders are fed with near-uncorrelated bits. The significance of this will become clear during our further discourse.

Due to its structure, the turbo encoder shown in Figure 3.1 is also often referred to as a parallel concatenated code [12, 61, 117]. Parallel concatenated codes constitute specific product codes [19, 62, 85]. In general, product codes consist of two linear block codes C_1 and C_2 where C_1 and C_2 have parameters (n_1, k_1, d_{min1}) and (n_2, k_2, d_{min2}), respectively. Typically, $C_1 \equiv C_2$. As shown in Figure 3.2, product codes are obtained by placing the $k_1 \times k_2$ information data bits in an array of k_1 columns and k_2 rows. The k_1 columns and k_2 rows of the information data bits are encoded using C_1 and C_2, respectively. It is shown in [118] that the $(n_1 - k_1)$ last columns of Figure 3.2 are codewords of C_2, exactly as the $(n_2 - k_2)$ last rows are codewords of C_1 by construction. Furthermore, the parameters of the resulting product codes are given by $n = n_1 \times n_2$, $k = k_1 \times k_2$ and $d_{min} = d_{min1} \times d_{min2}$, while the code rate is given by $\frac{k_1}{n_1} \times \frac{k_2}{n_2}$. The structure of parallel concatenated codes is the same as that of product codes, except that the redundancy part arising from checking the parity of the parity part of both codes C_1 and C_2 is omitted. The major disadvantage of parallel concatenated codes is the loss in minimum free distance, which is only $d_{min1} + d_{min2} - 1$, compared to $d_{min1} \times d_{min2}$ in product codes.

The output bits from the two BCH encoders of Figure 3.1 are then punctured and multiplexed. Table 2.1 shows a wide range of the BCH codes exhibiting a variety of coding rates. By appropriately designing the puncturer and the multiplexer, we are capable of achieving an overall coding rate, which is identical to that of the original BCH codes. However, it is not a common practice to apply puncturing of the parity bits in turbo block codes [61, 117],

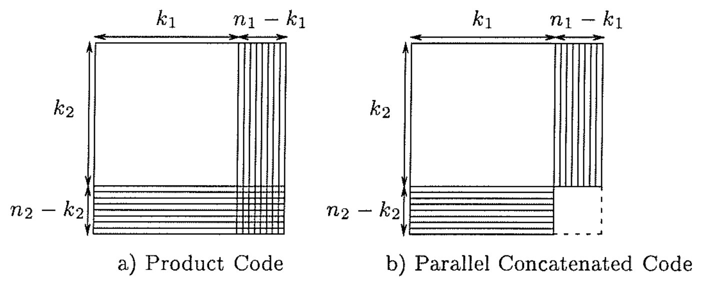


Figure 3.2: Construction of product codes and parallel concatenated codes.

since their puncturing significantly degrades the achievable performance. Having considered the encoder structure, let us now concentrate on the decoder schematic in the next section.

3.3 Turbo Decoder

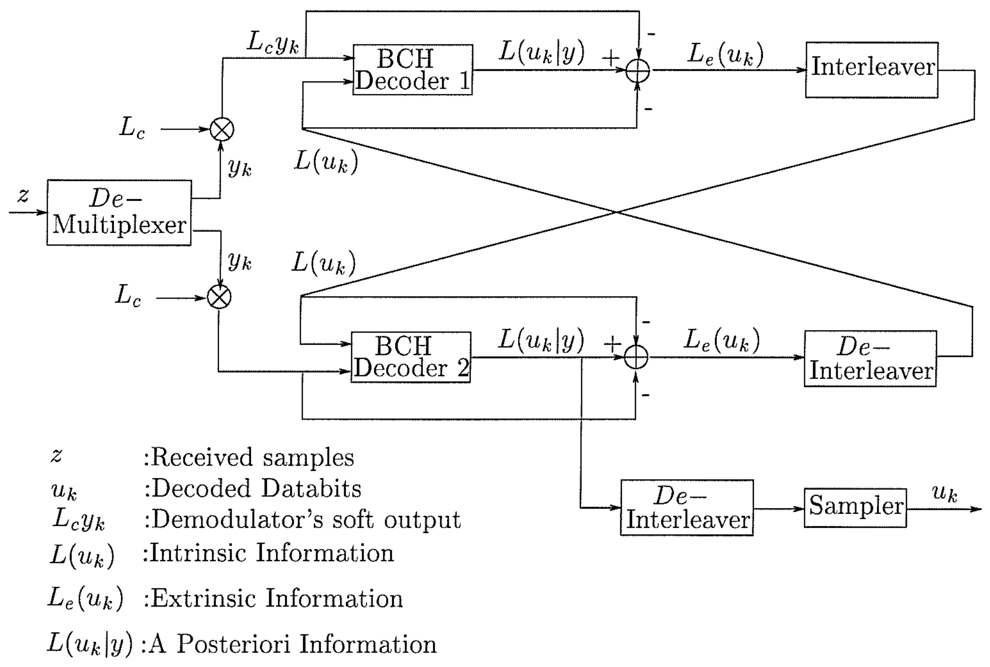


Figure 3.3: Turbo decoder schematic.

Figure 3.3 shows the general structure of a turbo BCH decoder. We summarise below the definition of the various quantities used in the figure.

Soft Channel Output, $L_c y$: The soft channel output is simply the matched filter based demodulator's output, y_k, multiplied by the so-called channel reliability value, L_c. Both of these quantities are described in more depth in Section 3.3.2.

Intrinsic Information, $L(u_k)$: The intrinsic information concerning a data bit u_k is the information known before decoding starts. It can be achieved from any source other than the received channel output sequence or the code constraints and it is also known as *a-priori information* .

Extrinsic Information, $L_e(u_k)$: The extrinsic information related to a data bit u_k is the information carried by the bits surrounding u_k, which was imposed by the code constraints. In other words, as indicated by the terminology 'extrinsic', no information directly concerning the data bit itself is part of the extrinsic information.

A-Posteriori Information, $L(u_k|y)$: The a-posteriori information of a data bit u_k is given by the decoder after taking into account *all* the available sources of information about u_k, i.e. both the intrinsic and extrinsic sources.

Again in this chapter, the turbo decoder of Figure 3.3 is constituted by two BCH decoders. Since we opted for employing the trellis decoding method, two algorithms, namely the *Maximum A-Posteriori* (MAP) [11] and the *Soft Output Viterbi Algorithm* (SOVA) [53, 54] can be used.

The decoder uses the soft channel output $L_c y$ and the intrinsic information $L(u_k)$ to provide the a-posteriori information $L(u_k|y)$ at its output, as shown in Figure 3.3. The extrinsic information, $L_e(u_k)$ is given by subtracting the soft channel output $L_c y$ and the intrinsic information $L(u_k)$ from the a-posteriori information $L(u_k|y)$, which will be justified during our further discourse in Section 3.3.3. After being interleaved or de-interleaved, as seen in the figure, the extrinsic information $L_e(u_k)$ becomes the intrinsic information $L(u_k)$ of the second decoder. Similarly, the extrinsic information gained by the second decoder is passed back to the first decoder as its intrinsic information. Basically, both decoders assist each other by exchanging their information related to the data bits and this results in the *iterative decoding* process, which constitutes the subject of this chapter. We note, however that there is no intrinsic information for the first decoder in the first iteration, since the extrinsic information of the other decoder is unavailable at this stage.

3.3.1 Log Likelihood Ratio

The *Log Likelihood Ratio* (LLR) is a quantity, which was introduced in the context of turbo coding by Robertson [52], in order to simplify the exchange of extrinsic information between the decoders. Let u_k be the data bit at time instant k and the probability of u_k being $+1$

be $P(u_k = +1)$ while that of $u_k = -1$ be $P(u_k = -1)$. We can define the LLR of u_k as [52]:

$$\begin{aligned} L(u_k) &:= \ln \frac{P(u_k = +1)}{P(u_k = -1)} \\ &= \ln \frac{P(u_k = +1)}{1 - P(u_k = +1)} \\ &= \ln \frac{1 - P(u_k = -1)}{P(u_k = -1)} \ . \end{aligned} \tag{3.1}$$

Notice that the two possible values for the data bit u_k are taken to be +1 and -1, rather than 1 and 0. This definition of the two values of the binary variable makes no conceptual difference, but simplifies the mathematics in the derivations which follow.

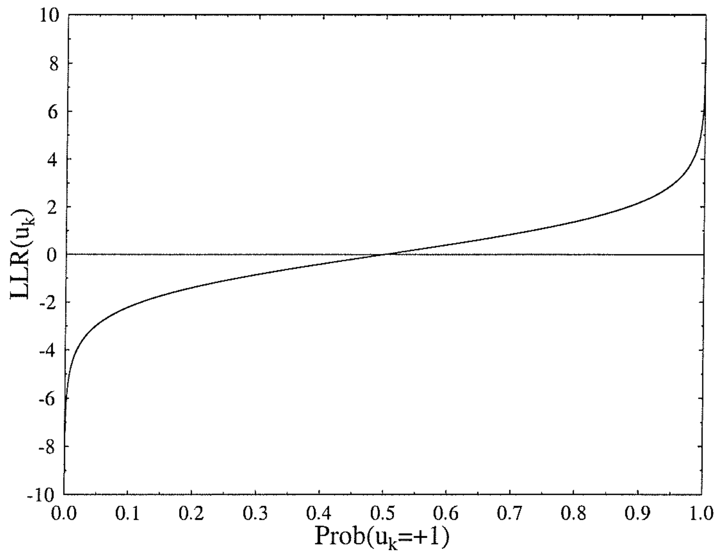


Figure 3.4: Log Likelihood Ratio $L(u_k)$ versus the probability of $u_k = +1$.

Figure 3.4 shows how $L(u_k)$ varies as the probability of $u_k = +1$ varies. From the figure, we can see that the LLR indicates two different things:

- Its polarity shows, whether u_k is more likely to be +1 (positive LLR) or -1 (negative LLR).
- Its magnitude ($|L(u_k)|$) indicates, how likely is the symbol u_k to be +1 or -1.

From Equation 3.1, we derive

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)} = \frac{1 - P(u_k = -1)}{P(u_k = -1)} , \tag{3.2}$$

and accordingly,

$$e^{\pm L(u_k)} = \frac{P(u_k = \pm 1)}{1 - P(u_k = \pm 1)} . \tag{3.3}$$

After solving Equation 3.3 for P($u_k = \pm 1$), we arrive at

$$\begin{aligned} P(u_k = \pm 1) &= \frac{e^{\pm L(u_k)}}{1 + e^{\pm L(u_k)}} \\ &= \left\{ \frac{e^{\frac{-L(u_k)}{2}}}{1 + e^{-L(u_k)}} \right\} . e^{\pm \frac{L(u_k)}{2}} \\ &= C \cdot e^{\left\{ u_k \cdot \frac{L(u_k)}{2} \right\}} , \end{aligned} \tag{3.4}$$

where C is a constant, which is independent of the polarity of u_k.

3.3.2 Soft Channel Output

Apart from the LLR $L(u_k)$, which is based on the unconditional probabilities $P(u_k = \pm 1)$, we are also interested in the LLRs based on conditional probabilities. We will use the conditional LLRs based on the probability that the matched filter output would be y_k, given that the corresponding transmitted bit x_k was either +1 or -1. This conditional LLR is defined as

$$L(y_k|x_k) := \ln \frac{P(y_k|x_k = +1)}{P(y_k|x_k = -1)} . \tag{3.5}$$

We assume that the transmitted channel coded bit x_k has been sent over an AWGN channel using BPSK modulation. Then, the probability density function of the received symbol y_k i.e. that of the soft output of the demodulator, conditioned on the transmitted channel coded bit x_k can be expressed by simply stating that the received signal amplitudes are Gaussian distributed around the transmitted bit values according to the noise variance which is formulated as:

$$P(y_k|x_k = \pm 1) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{E_B}{2\sigma^2}(y_k - a x_k)^2} \tag{3.6}$$

where,

σ^2	is the noise variance
E_B	is the energy per bit
a	is the fading amplitude (=1 for a non-fading AWGN channel).

We can rewrite Equation 3.5 as

$$\begin{aligned} L(y_k|x_k) &= \ln\left\{\frac{e^{-\frac{E_B}{2\sigma^2}(y_k-a)^2}}{e^{-\frac{E_B}{2\sigma^2}(y_k+a)^2}}\right\} \\ &= \frac{E_B}{2\sigma^2}\cdot 4a\cdot y_k \\ &= L_c\cdot y_k\ , \end{aligned} \tag{3.7}$$

where

$$L_c := \frac{E_B}{2\sigma^2}\cdot 4a \tag{3.8}$$

is the so-called *channel reliability value*, which depends only on the Signal to Noise Ratio (SNR) and on the fading amplitude a of the channel. Explicitly, if the SNR or the fading amplitude exhibits a low value, the reliability of the associated bit is low. Hence, for BPSK over an AWGN channel, the conditional LLR $L(y_k|x_k)$ of Equation 3.7 is referred to as the soft channel output, which is simply the matched filter demodulator's soft output y_k multiplied by the channel reliability value L_c. Following the definition of the LLR, let us now review the maximum a-posteriori decoding algorithm.

3.3.3 The Maximum A-Posteriori Algorithm

The MAP algorithm was proposed by Bahl *et al.* [11] in 1974, with the aim of contriving an algorithm for the decoding of convolutional codes. Although it is the minimum BER algorithm, this optimality is achieved at a high complexity and hence in most applications the lower complexity maximum likelihood sequence estimation Viterbi algorithm (VA) was preferred. However, in turbo coding the MAP algorithm is preferred, since it gives a better estimate of the probability of each data bit in terms of the a-posteriori information $L(u_k|y)$.

Throughout this section Bayes' rule will be used repeatedly, which defines the relationship between the joint probability $P(i\wedge j)$, and the conditional probability $P(i|j)$ of i given j, as follows:

$$P(i\wedge j) = P(i|j)\cdot P(j)\ . \tag{3.9}$$

Using Equation 3.9, we can show that:

$$P(\{i\wedge j\}|k) = P(i|\{j\wedge k\})\cdot P(j|k)\ . \tag{3.10}$$

Motivated by this aim, let us define $A\equiv i\wedge j$ and $B\equiv j\wedge k$. Then using Equation 3.9 we arrive at:

$$\begin{aligned}
P(\{i \wedge j\}|k) &\equiv P(A|k) = \frac{P(A \wedge k)}{P(k)} \\
&= \frac{P(i \wedge j \wedge k)}{P(k)} \equiv \frac{P(i \wedge B)}{P(k)} \\
&= \frac{P(i|B) \cdot P(B)}{P(k)} \equiv P(i|\{j \wedge k\}) \cdot \frac{P(j \wedge k)}{P(k)} \\
&= P(i|\{j \wedge k\}) \cdot P(j|k) \ . \qquad (3.11)
\end{aligned}$$

The goal of the MAP algorithm is to find the LLR of each channel decoded bit i.e. original information bit, given that the demodulator's corresponding soft output values are the elements of the vector $\underline{y}$. Note that we do not necessarily have to find the LLRs for the parity bits. This is equivalent to finding the LLR of

$$L(u_k|\underline{y}) = \ln \frac{P(u_k = +1|\underline{y})}{P(u_k = -1|\underline{y})} \ , \qquad (3.12)$$

where k is the index of the data bits. Upon applying Bayes' law, we can rewrite Equation 3.12 as

$$\begin{aligned}
L(u_k|\underline{y}) &= \ln \frac{P(u_k = +1 \wedge \underline{y}) \cdot P(\underline{y})}{P(u_k = -1 \wedge \underline{y}) \cdot P(\underline{y})} \\
&= \ln \frac{P(u_k = +1 \wedge \underline{y})}{P(u_k = -1 \wedge \underline{y})} \ . \qquad (3.13)
\end{aligned}$$

Figure 3.5 shows a simple state transition diagram. If the previous decoder state S_{k-1} and the present state S_k are known, then the data bit u_k, which caused the transition between these two states will be also known. Hence, the probability of $u_k = +1$ is equal to the summation of all the probabilities of state transitions caused by $u_k = +1$.

From Figure 3.5, we can write the probability of $P(u_k = +1)$ as

$$\begin{aligned}
P(u_k = +1) &= P(S_{k-1} = a \wedge S_k = d) + P(S_{k-1} = b \wedge S_k = a) + \\
&\quad P(S_{k-1} = c \wedge S_k = c) + P(S_{k-1} = d \wedge S_k = b) \\
&= \sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} P(S_{k-1} = \grave{s} \wedge S_k = s) \ , \qquad (3.14)
\end{aligned}$$

where $(\grave{s}, s) \Rightarrow u_k = +1$ indicates all the state transitions from the previous state S_{k-1} to the present state S_k that were caused by the data bit $u_k = +1$. Using Equation 3.14, we can rewrite Equation 3.13 quantifying the LLR affecting the probability that the information bit u_k was transmitted, given that the demodulator's corresponding soft output values were

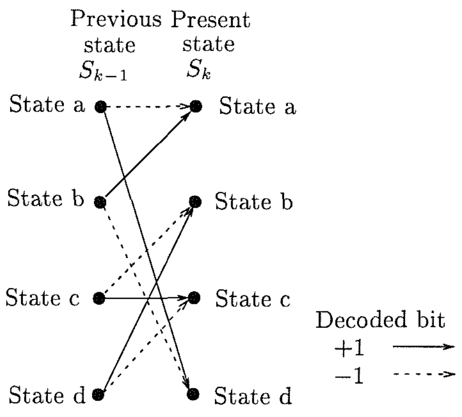


Figure 3.5: Simple state transition diagram.

stored in $\underline{y}$, as follows:

$$L(u_k|\underline{y}) = \ln \left\{ \frac{\sum_{\substack{(\acute{s},s)\Rightarrow \\ u_k=+1}} P(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})}{\sum_{\substack{(\acute{s},s)\Rightarrow \\ u_k=-1}} P(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})} \right\} . \tag{3.15}$$

For simplicity, we shall write $P(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})$ as $P(\acute{s} \wedge s \wedge \underline{y})$.

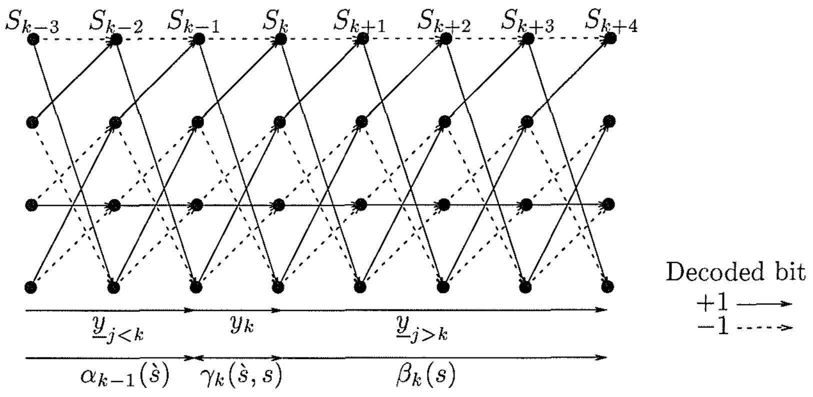


Figure 3.6: Simple trellis diagram.

We now consider the individual probabilities $P(\acute{s} \wedge s \wedge \underline{y})$ in the numerator and denominator of Equation 3.15. As shown in the simple trellis diagram of Figure 3.6, we can split the vector of soft demodulator outputs $\underline{y}$ into three sections:

- y_k, the soft demodulator output sample associated with the present transition;
- $\underline{y}_{j<k}$, the vector of soft demodulator outputs received prior to the present transition;

- $\underline{y}_{j>k}$, the vector of soft demodulator outputs received after the present transition.

We can then rewrite the individual probabilities $P(\grave{s} \wedge s \wedge \underline{y})$ as:

$$P(\grave{s} \wedge s \wedge \underline{y}) = P(\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge y_k \wedge \underline{y}_{j>k}) \ . \tag{3.16}$$

Let us assume that the channel is memoryless. Then the future vector of soft demodulator outputs $\underline{y}_{j>k}$ will depend only on the present state s and not on the previous state $\grave{s}$ nor will it depend on the present and previous vector of soft demodulator outputs y_k and $\underline{y}_{j<k}$. Applying Bayes' rule, we derive from Equation 3.16:

$$\begin{aligned}
P(\grave{s} \wedge s \wedge \underline{y}) &= P(\underline{y}_{j>k} \wedge \grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge y_k) \\
&= P(\underline{y}_{j>k} | \{\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge y_k\}).P(\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge y_k) \\
&= P(\underline{y}_{j>k} | s).P(y_k \wedge s \wedge \grave{s} \wedge \underline{y}_{j<k}) \\
&= P(\underline{y}_{j>k} | s).P(\{y_k \wedge s\} | \{\grave{s} \wedge \underline{y}_{j<k}\}).P(\grave{s} \wedge \underline{y}_{j<k}) \\
&= P(\grave{s} \wedge \underline{y}_{j<k}).P(\{y_k \wedge s\} | \grave{s}).P(\underline{y}_{j>k} | s) \\
&= \alpha_{k-1}(\grave{s}).\gamma_k(\grave{s}, s).\beta_k(s) \ ,
\end{aligned} \tag{3.17}$$

where,

$\alpha_{k-1}(\grave{s}) := P(\grave{s} \wedge \underline{y}_{j<k})$,	is the probability that the decoder is in trellis state $\grave{s}$ at time $k-1$ and the vector of soft demodulator outputs up to this point is $\underline{y}_{j<k}$,
$\gamma_k(\grave{s}, s) := P(\{y_k \wedge s\} \vert \grave{s})$,	is the probability that given the decoder was in trellis state $\grave{s}$ at time $k-1$, it traverses to state s and the soft demodulator output sample for this transition is y_k,
$\beta_k(s) := P(\underline{y}_{j>k} \vert s)$,	is the probability that given the decoder is in trellis state s at time k, and the future vector of soft demodulator outputs will be $\underline{y}_{j>k}$.

Equation 3.17 has shown that the individual probabilities $P(\grave{s} \wedge s \wedge \underline{y})$ in Equation 3.15 can be represented by the product of three independent trellis transition probabilities, namely $\alpha_{k-1}(\grave{s})$, $\gamma_k(\grave{s}, s)$ and $\beta_k(s)$, which are also shown in Figure 3.6. The MAP algorithm first finds the probabilities $\gamma_k(\grave{s}, s)$ for all possible transitions throughout the trellis from state $S_{k-1} = \grave{s}$ to state $S_k = s$. Using the calculated $\gamma_k(\grave{s}, s)$, the MAP algorithm then finds $\alpha_{k-1}(\grave{s})$ and $\beta_k(s)$ using the so-called *forward recursion* and *backward recursion*, respectively. Let us now describe, how the values $\alpha_{k-1}(\grave{s})$, $\gamma_k(\grave{s}, s)$ and $\beta_k(s)$ can be calculated.

3.3.3.1 Calculation of the $\gamma_k(\grave{s}, s)$ Values

First we consider how the $\gamma_k(\grave{s}, s)$ values can be calculated from the demodulator's soft output sequence $\underline{y}$. Using Bayes' rule given in Equation 3.10, we expand the definition of $\gamma_k(\grave{s}, s)$ in Equation 3.17 to:

$$\begin{aligned}\gamma_k(\grave{s}, s) &= P(\{y_k \wedge s\}|\grave{s}) \\ &= P(y_k|\{s \wedge \grave{s}\}).P(s|\grave{s}) \\ &= P(y_k|\{s \wedge \grave{s}\}).P(u_k) \ , \end{aligned} \tag{3.18}$$

where u_k is the original uncoded information bit necessary to cause the transition from state $S_{k-1} = \grave{s}$ to state $S_k = s$. From Equation 3.4 we have:

$$\begin{aligned} P(u_k) &= \left\{ \frac{e^{-\frac{L(u_k)}{2}}}{1 + e^{-L(u_k)}} \right\} .e^{\left\{u_k.\frac{L(u_k)}{2}\right\}} \\ &= C_1 \cdot e^{\left\{u_k.\frac{L(u_k)}{2}\right\}} \ , \end{aligned} \tag{3.19}$$

where C_1 is a constant, which depends only on the LLR $L(u_k)$ and not on whether the estimated data bit u_k is -1 or $+1$. Again, as we have shown in Figure 3.3, $L(u_k)$ is the intrinsic or a-priori information.

In the trellis, a transmitted channel coded bit x_k is associated with a single transition from state $S_{k-1} = \grave{s}$ to state $S_k = s$. Hence, we can rewrite the first term of Equation 3.18 as:

$$P(y_k|\{\grave{s} \wedge s\}) \equiv P(y_k|x_k) \ . \tag{3.20}$$

We substitute Equation 3.6 in Equation 3.20, yielding:

$$\begin{aligned} P(y_k|\{\grave{s} \wedge s\}) &= \frac{1}{\sqrt{2\pi}\sigma} e^{\left\{-\frac{E_b}{2\sigma^2}(y_k - a x_k)^2\right\}} \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{\left\{-\frac{E_b}{2\sigma^2}(y_k^2 - 2a x_k y_k + a^2 x_k^2)\right\}} \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{\left\{-\frac{E_b}{2\sigma^2}(y_k^2 + a^2 x_k^2)\right\}} \cdot e^{\left\{\frac{E_b a}{\sigma^2}(y_k x_k)\right\}} \\ &= C_2 \cdot e^{\left\{\frac{E_b a}{\sigma^2}(y_k x_k)\right\}} \ , \end{aligned} \tag{3.21}$$

where

$$C_2 = \frac{1}{\sqrt{2\pi}\sigma} e^{\left\{-\frac{E_b}{2\sigma^2}(y_k^2 + a^2 x_k^2)\right\}} \ . \tag{3.22}$$

Let us substitute Equation 3.19 and Equation 3.21 into Equation 3.18, which results in:

$$\begin{aligned}\gamma_k(\grave{s},s) &= P(y_k|\{\grave{s}\wedge s\})\cdot P(u_k)\\ &= C_1\cdot C_2\cdot e^{\left\{u_k\frac{L(u_k)}{2}\right\}}\cdot e^{\left\{\frac{E_b a}{\sigma^2}(y_k x_k)\right\}}\\ &= C\cdot e^{\left\{u_k\frac{L(u_k)}{2}\right\}}\cdot e^{\left\{\frac{L_c}{2}(y_k x_k)\right\}}\ , \end{aligned} \qquad (3.23)$$

where $C = C_1 \cdot C_2$ which does not depend on the sign of the uncoded data bit u_k or on that of the transmitted channel coded bit x_k. Hence C is a constant for the summations in the numerator and denominator of Equation 3.15 and hence it cancels out.

3.3.3.2 Forward Recursion

From the definition of $\alpha_{k-1}(\grave{s})$ in Equation 3.17 upon proceeding from stage $k-1$ to stage k, we can write

$$\begin{aligned}\alpha_k(s) &= P(S_k = s\wedge \underline{y}_{j<k+1})\\ &= P(s\wedge \underline{y}_{j<k}\wedge y_k)\\ &= \sum_{\text{all } \grave{s}} P(s\wedge \grave{s}\wedge \underline{y}_{j<k}\wedge y_k)\ , \end{aligned} \qquad (3.24)$$

where the probability $P(s\wedge \underline{y}_{j<k+1})$ is equal to the sum of the joint probabilities $P(s\wedge \grave{s}\wedge \underline{y}_{j<k+1})$ over all possible previous states $\grave{s}$. Again, we assume that the channel is memoryless and upon applying Bayes' rule, we can rewrite Equation 3.24 as follows:

$$\begin{aligned}\alpha_k(s) &= \sum_{\text{all } \grave{s}} P(s\wedge y_k\wedge \grave{s}\wedge \underline{y}_{j<k})\\ &= \sum_{\text{all } \grave{s}} P(\{s\wedge y_k\}|\{\grave{s}\wedge \underline{y}_{j<k}\})\cdot P(\grave{s}\wedge \underline{y}_{j<k})\\ &= \sum_{\text{all } \grave{s}} P(\{s\wedge y_k\}|\grave{s})\cdot P(\grave{s}\wedge \underline{y}_{j<k})\\ &= \sum_{\text{all } \grave{s}} \alpha_{k-1}(\grave{s})\cdot \gamma_k(\grave{s},s)\ . \end{aligned} \qquad (3.25)$$

In Section 3.3.3.1 we have seen that $\gamma_k(\grave{s},s)$ can be calculated from the demodulator's soft output sample y_k and from the intrinsic information $L(u_k)$. Using the previously computed $\gamma_k(\grave{s},s)$ values, we can calculate the $\alpha_k(s)$ values, recursively using Equation 3.25. Figure 3.7 shows a simple example of the forward recursion calculation of $\alpha_k(s)$. For a binary trellis, for example in binary BCH codes, there are only two legitimate transitions from the previous state $S_{k-1} = \grave{s}$ to any present state $S_k = s$, as shown in Figure 3.7. Furthermore, one of these will be associated with an information input bit of $u_k = +1$, and the other with an

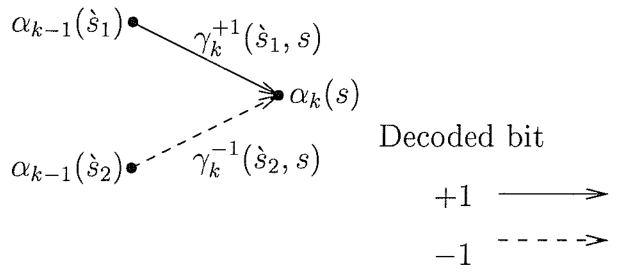

$\alpha_{k-1}(\grave{s}_1)$
$\gamma_k^{+1}(\grave{s}_1, s)$
$\alpha_k(s)$
$\alpha_{k-1}(\grave{s}_2)$
$\gamma_k^{-1}(\grave{s}_2, s)$
Decoded bit
+1
−1

Figure 3.7: Forward recursion for the computation of $\alpha_k(s)$.

input bit of $u_k = -1$. Hence, we can expand Equation 3.25 as:

$$\alpha_k(s) = \alpha_{k-1}(\grave{s}_1) \cdot \gamma_k^{+1}(\grave{s}_1, s) + \alpha_{k-1}(\grave{s}_2) \cdot \gamma_k^{-1}(\grave{s}_2, s) \ , \tag{3.26}$$

where $\gamma_k^{x_k}(\grave{s}, s)$ is the value of $\gamma_k(\grave{s}, s)$ for the transitions between states, when the transmitted bit is $x_k = -1$ or $+1$, as indicated in Figure 3.7.

In Section 2.2.2, we have shown that the trellis of the BCH code always commences from the all zero state $S_0 = 0$. Hence, we have $P(S_0 = 0) = 1$ and $P(S_0 = s) = 0)$ for all $s \neq 0$. The initial conditions for the forward recursion are:

$$\begin{aligned} \alpha_0(S_0 = 0) &= 1 \\ \alpha_0(S_0 = s) &= 0 \qquad \text{for all } s \neq 0 \ . \end{aligned} \tag{3.27}$$

3.3.3.3 Backward Recursion

Similarly, the values of $\beta_k(s)$ can be calculated recursively. From the definition of $\beta_k(s)$ in Equation 3.17, we can write $\beta_{k-1}(\grave{s})$ as:

$$\begin{aligned} \beta_{k-1}(\grave{s}) &= P(\underline{y}_{j>k-1}|\grave{s}) \\ &= P(\{\underline{y}_{j>k} \wedge y_k\}|\grave{s}) \\ &= \sum_{\text{all } s} P(\{\underline{y}_{j>k} \wedge y_k \wedge s\}|\grave{s}) \ , \end{aligned} \tag{3.28}$$

where the probability $P(\underline{y}_{j>k-1}|\grave{s})$ is equal to the sum of the joint probabilities $P(\{\underline{y}_{j>k-1} \wedge s\}|\grave{s})$ over all possible future states s. Again, we assume that the channel is memoryless and upon applying Bayes' rule we can rewrite Equation 3.28 as follows:

$$\begin{aligned}
\beta_{k-1}(\grave{s}) &= \sum_{\text{all } s} P(\{\underline{y}_{j>k} \wedge y_k \wedge s\}|\grave{s}) \\
&= \sum_{\text{all } s} P(\underline{y}_{j>k}|\{\grave{s} \wedge s \wedge y_k\}).P(\{y_k \wedge s\}|\grave{s}) \\
&= \sum_{\text{all } s} P(\underline{y}_{j>k}|s).P(\{y_k \wedge s\}|\grave{s}) \\
&= \sum_{\text{all } s} \beta_k(s).\gamma_k(\grave{s}, s) \ . \qquad (3.29)
\end{aligned}$$

Similarly to the forward recursion, once the $\gamma_k(\grave{s}, s)$ values are known, backward recursion can be used for calculating the values of $\beta_{k-1}(\grave{s})$ from the values of $\beta_k(s)$ using Equation 3.29. Figure 3.8 shows a simple example of the backward recursive calculation of $\beta_k(s)$. As for

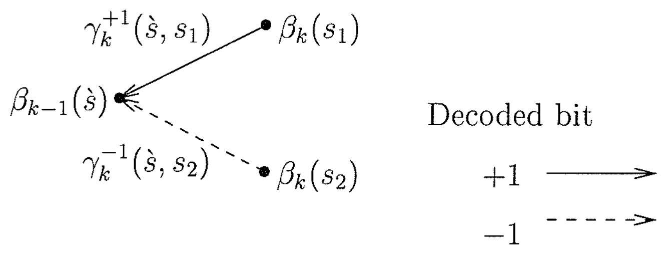


Figure 3.8: Backward recursion for the computation of $\beta_{k-1}(\grave{s})$.

the forward recursion, there are only two transitions from the present states $S_k = s$ to the previous state $S_{k-1} = \grave{s}$, as shown in Figure 3.8. Hence similarly to Equation 3.26, we expand Equation 3.29 to:

$$\beta_{k-1}(\grave{s}) = \beta_k(s_1) \cdot \gamma_k^{+1}(\grave{s}, s_1) + \beta_k(s_2) \cdot \gamma_k^{-1}(\grave{s}, s_2) \ . \qquad (3.30)$$

Again in Section 2.2.2, we have shown that the trellis of the BCH code always ends at the all-zero state $S_n = 0$. Hence, we have $P(S_n = 0) = 1$ and $P(S_n = s) = 0)$ for all $s \neq 0$. The initial conditions for the backward recursion are:

$$\begin{aligned}
\beta_n(S_n = 0) &= 1 \\
\beta_n(S_n = s) &= 0 \qquad \text{for all } s \neq 0 \qquad (3.31)
\end{aligned}$$

3.3.3.4 Summary of the MAP Algorithm

Figure 3.9 shows a summary of the key operations in the MAP algorithm, which constitutes the BCH decoder in Figure 3.3. Initially, the algorithm relies on the demodulator's soft

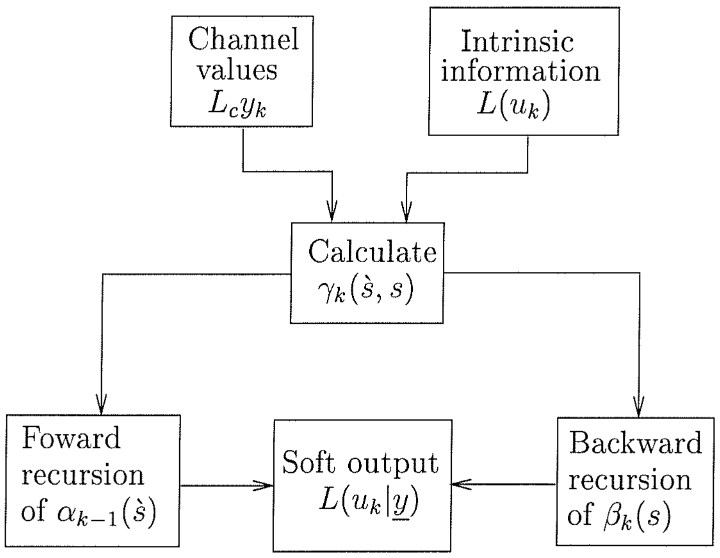

Channel values $L_c y_k$
Intrinsic information $L(u_k)$
Calculate $\gamma_k(\grave{s}, s)$
Foward recursion of $\alpha_{k-1}(\grave{s})$
Soft output $L(u_k|\underline{y})$
Backward recursion of $\beta_k(s)$

Figure 3.9: Summary of the key operations in the MAP algorithm, which constitutes the BCH decoder in Figure 3.3.

output sequence $\underline{y}$ and on the intrinsic information $L(u_k)$, provided by the second component decoder shown in Figure 3.3. Both $\underline{y}$ and $L(u_k)$ are then used to calculate the $\gamma_k(\grave{s}, s)$ values according to Equation 3.23. After that, we use the $\gamma_k(\grave{s}, s)$ values in order to calculate the $\alpha_k(s)$ and $\beta_k(s)$ values by employing forward recursion of Equation 3.25 and backward recursion of Equation 3.29, respectively.

In Equation 3.17, we have shown that $P(\grave{s} \wedge s \wedge \underline{y})$ depends on the values $\alpha_k(s)$, $\gamma_k(\grave{s}, s)$ and $\beta_k(s)$, which we have already described. Using Equation 3.17, we can rewrite Equation 3.15 as follows:

$$\begin{aligned} L(u_k|\underline{y}) &= \ln \left\{ \frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} P(S_{k-1} = \grave{s} \wedge S_k = s \wedge \underline{y})}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} P(S_{k-1} = \grave{s} \wedge S_k = s \wedge \underline{y})} \right\} \\ &= \ln \left\{ \frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s}, s) \cdot \beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s}, s) \cdot \beta_k(s)} \right\} . \end{aligned} \tag{3.32}$$

Equation 3.32 is then used in the final stage of the decoding process to derive the conditional LLR of u_k, given the demodulator's soft output sequence $\underline{y}$ and the intrinsic information $L(u_k)$.

Let us consider the expression of $\gamma_k(\grave{s}, s)$ in Equation 3.23, which is re-stated here for convenience,

$$\gamma_k(\grave{s},s) = C \cdot e^{\left\{u_k \frac{L(u_k)}{2}\right\}} \cdot e^{\left\{\frac{L_c}{2} x_k y_k\right\}} . \tag{3.33}$$

As we have shown in Section 2.2.2, the systematic BCH coded transmitted sequence consists of the original data bit sequence, followed by the parity bit sequence. However, in Equation 3.32 we are interested in deriving the conditional LLRs of the original uncoded data bits u_k. Hence, the channel coded transmitted bits x_k are replaced by u_k in Equation 3.33, yielding:

$$\gamma_k(\grave{s},s) = C \cdot e^{\left\{u_k \cdot \frac{L(u_k)}{2}\right\}} \cdot e^{\left\{\frac{L_c}{2} u_k y_k\right\}} \tag{3.34}$$

Using Equation 3.34 and remembering that in the numerator we have $u_k = +1$ for all terms in the summation, whereas in the denominator we have $u_k = -1$, we can rewrite Equation 3.32 yielding the a-posteriori information for our decision concerning the bit u_k as:

$$\begin{aligned}
L(u_k|\underline{y}) &= \ln \left[\frac{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s},s) \cdot \beta_k(s)}{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s},s) \cdot \beta_k(s)} \right] \\
&= \ln \left[\frac{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\grave{s}) \cdot e^{\{+\frac{L(u_k)}{2}\}} \cdot e^{\{+\frac{Lcy_k}{2}\}} \beta_k(s)}{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\grave{s}) \cdot e^{\{-\frac{L(u_k)}{2}\}} \cdot e^{\{-\frac{Lcy_k}{2}\}} \beta_k(s)} \right] \\
&= L(u_k) + Lcy_k + \ln \left[\frac{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\grave{s}) \cdot \beta_k(s)}{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\grave{s}) \cdot \beta_k(s)} \right] \\
&= L(u_k) + Lcy_k + L_e(u_k) \, ,
\end{aligned} \tag{3.35}$$

where

$$L_e(u_k) = \ln \left[\frac{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\grave{s}).\beta_k(s)}{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\grave{s}).\beta_k(s)} \right] , \tag{3.36}$$

and the following notations apply:

y_k	the demodulator's soft output for the transmitted data bit $x_k = u_k$
$L_c y_k$	soft channel output of y_k
$L(u_k)$	intrinsic or a-priori information
$L_e(u_k)$	extrinsic information
$L(u_k\|\underline{y})$	a-posteriori information.

Now, we are ready to relate Equation 3.35 to Figure 3.3. In the figure, we can see that the decoder accepts two inputs, the intrinsic information $L(u_k)$ of the original uncoded information bits and the soft channel output $L_c y$ constituted by the product of the demodulator's soft output $\underline{y}$ and the channel reliability value L_c of Equation 3.8. At its output, it produces the a-posteriori information $L(u_k|\underline{y})$. The decoder then calculate the extrinsic information $L_e(u_k)$ imposed by the code constraints from the demodulator's soft output sequence $\underline{y}$, but excluding the demodulator's soft output sample y_k due to the transmitted data bit, x_k.

3.3.4 Modifications of the MAP algorithm

3.3.4.1 Introduction

The MAP algorithm, in the form described in Section 3.3.3, is extremely complex due to the floating point multiplication needed in Equations 3.25 and 3.29 for the recursive calculation of $\alpha_k(s)$ and $\beta_k(s)$, as well as because of the multiplications and exponential operations required to calculate $\gamma_k(\grave{s}, s)$ using Equation 3.23. Further factor increasing the complexity are the multiplication and natural logarithm operations required to calculate $L(u_k|y)$ using Equation 3.35.

The Max-Log-MAP algorithm was initially proposed by Koch and Baier [50] and Erfanian *et al.* [51] for reducing the complexity of the MAP algorithm. This technique transfers the computation of the recursions into the logarithmic domain and invokes an approximation for dramatically reducing the complexity. Due to the approximation, its performance is sub-optimal. However, Robertson *et al.* [52] later proposed the Log-MAP algorithm, which partially corrected the approximation made in the Max-Log-MAP algorithm. Hence the performance of the Log-MAP algorithm is similar to that of the MAP algorithm, but at a fraction of its complexity. Let us now consider the Max-Log-MAP algorithm and the Log-MAP algorithm in more detail.

3.3.4.2 Max-Log-MAP Algorithm

The MAP algorithm calculates the a-posteriori LLRs $L(u_k|\underline{y})$ using Equation 3.32 and hence it requires the following values:

1. The $\alpha_{k-1}(\grave{s})$ values, which are calculated in a forward recursive manner using Equation 3.25;
2. The $\beta_k(s)$ values, which are calculated in a backward recursive manner using Equation 3.29;
3. The transition probabilities $\gamma_k(\grave{s}, s)$, which are calculated using Equation 3.23.

The Max-Log-MAP algorithm simplifies these preceding equations by transferring them into the logarithmic domain and then using the approximation [50, 51]:

$$\ln\left(\sum_i e^{x_i}\right) \approx \max_i(x_i) \ , \tag{3.37}$$

where $\max_i(x_i)$ means the maximum value of x_i.

Let us define $A_k(s)$, $B_k(s)$ and $\Gamma_k(\grave{s}, s)$ as follows:

$$A_k(s) \triangleq \ln[\alpha_k(s)] \ , \tag{3.38}$$

$$B_k(s) \triangleq \ln[\beta_k(s)] \ , \tag{3.39}$$

and

$$\Gamma_k(\grave{s}, s) \triangleq \ln[\gamma_k(\grave{s}, s)] \ . \tag{3.40}$$

Upon substituting Equation 3.26 into Equation 3.38 and using Equation 3.37, we arrive at:

$$\begin{aligned} A_k(s) &= \ln\left[\alpha_{k-1}(\grave{s}_1)\cdot\gamma_k^{+1}(\grave{s}_1, s) + \alpha_{k-1}(\grave{s}_2)\cdot\gamma_k^{-1}(\grave{s}_2, s)\right] \\ &= \ln\left[e^{\left\{A_{k-1}(\grave{s}_1)+\Gamma_k^{+1}(\grave{s}_1,s)\right\}} + e^{\left\{A_{k-1}(\grave{s}_2)+\Gamma_k^{-1}(\grave{s}_2,s)\right\}}\right] \\ &\approx \max\left[\left\{A_{k-1}(\grave{s}_1)+\Gamma_k^{+1}(\grave{s}_1, s)\right\}, \left\{A_{k-1}(\grave{s}_2)+\Gamma_k^{-1}(\grave{s}_2, s)\right\}\right] \ . \end{aligned} \tag{3.41}$$

Similarly to Equation 3.41, we can rewrite Equation 3.39 as:

$$\begin{aligned} B_{k-1}(\grave{s}) &= \ln\left[\beta_k(s_1)\cdot\gamma_k^{+1}(\grave{s}, s_1) + \beta_k(s_2)\cdot\gamma_k^{-1}(\grave{s}, s_2)\right] \\ &= \ln\left[e^{\left\{B_k(s_1)+\Gamma_k^{+1}(\grave{s},s_1)\right\}} + e^{\left\{B_k(s_2)+\Gamma_k^{-1}(\grave{s},s_2)\right\}}\right] \\ &\approx \max\left[\left\{B_k(s_1)+\Gamma_k^{+1}(\grave{s}, s_1)\right\}, \left\{B_k(s_2)+\Gamma_k^{-1}(\grave{s}, s_2)\right\}\right] \ . \end{aligned} \tag{3.42}$$

We have shown in Equation 3.41 and 3.42 that the complex floating point multiplications in Equation 3.25 and 3.29 are simplified to simple floating point additions and comparisons.

In Figure 3.7, we have shown that there are two converging paths at the present state $S_k = s$. Equation 3.41 implies that the path with the maximum $\{A_{k-1}(\grave{s}) + \Gamma_k(\grave{s}, s)\}$ will be chosen by the decoder. This path can be viewed as the survivor path, i.e. the maximum likelihood path while we discard any other paths reaching the state. Due to the approximation in Equation 3.37, the value of A_k in the Max-Log-MAP algorithm actually gives the probability of the most likely path through the trellis to state $S_k = s$, rather than that of any path. This approximation is one of the reasons for the sub-optimal performance of the Max-Log-MAP algorithm compared to the MAP algorithm.

Using Equation 3.34, we can rewrite the branch metrics $\Gamma_k(\grave{s}, s)$ in Equation 3.40 as follows:

$$\begin{aligned}\Gamma_k(\grave{s}, s) &= \ln\left[C \cdot e^{\left\{u_k \frac{L(u_k)}{2}\right\}} . e^{\left\{\frac{L_c}{2} u_k y_k\right\}}\right] \\ &= \ln C + \frac{1}{2} u_k L(u_k) + \frac{1}{2} u_k L_c y_k \\ &= C' + \frac{u_k}{2}\left\{L(u_k) + L_c y_k\right\} \ , \qquad (3.43)\end{aligned}$$

where $C' = \ln C$ does not depend on u_k and hence it can be considered a constant for all paths and omitted from Equation 3.43. Explicitly, in Equation 3.43, we have converted the complex multiplications and exponential operations required to calculate $\gamma_k(\grave{s}, s)$ in Equation 3.23, into simple additions.

Finally, the a-posteriori LLRs $L(u_k|\underline{y})$ can be derived from Equation 3.32 according to the Max-Log-MAP algorithm as follows:

$$\begin{aligned}L(u_k|\underline{y}) &= \ln\left[\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s}, s) \cdot \beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s}, s) \cdot \beta_k(s)}\right] \\ &= \ln\left[\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} e^{\{A_{k-1}(\grave{s})+\Gamma_k(\grave{s},s)+B_k(s)\}}}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} e^{\{A_{k-1}(\grave{s})+\Gamma_k(\grave{s},s)+B_k(s)\}}}\right] \\ &\approx \max_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \left[A_{k-1}(\grave{s}) + \Gamma_k(\grave{s}, s) + B_k(s)\right] \\ &- \max_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \left[A_{k-1}(\grave{s}) + \Gamma_k(\grave{s}, s) + B_k(s)\right] \ . \qquad (3.44)\end{aligned}$$

For each original uncoded data bit u_k, the a-posteriori LLR $L(u_k|\underline{y})$ is calculated by considering every transition from the previous state $S_{k-1} = s$ to the present state $S_k = s$.

These transitions are grouped into those that might have occurred if we had $u_k = +1$, and those that might have occurred if $u_k = -1$. From each group, the maximum of $[A_{k-1}(\grave{s}) + \Gamma_k(\grave{s}, s) + B_k(s)]$ is found and the difference is taken. The difference is the a-posteriori LLR $L(u_k|\underline{y})$.

3.3.4.3 Log-MAP Algorithm

The Max-Log-MAP algorithm suffers from a slight performance degradation compared to the MAP algorithm due to the approximation of Equation 3.37. However, the approximation can be rendered exact by using the Jacobian logarithm [95]:

$$\begin{aligned} \ln(e^i + e^j) &= \max(i,j) + \ln(1 + e^{-|i-j|}) \\ &= \max(i,j) + f_c(|i-j|) \ , \end{aligned} \qquad (3.45)$$

where $f_c(\delta) = f_c(|i-j|)$ can be viewed of as a correction term and $\delta = |i-j|$ is the magnitude of the difference. The correction term $f_c(\delta)$ need not be computed for every value of δ, but instead it can be stored in a look-up table. Robertson *et al.* [52] found that such a look-up table needs to contain only eight values for δ, ranging between 0 and 5 for achieving a good accuracy.

3.3.5 The Soft Output Viterbi Algorithm

The *Soft Output Viterbi Algorithm* (SOVA) was proposed by Hagenauer [53] in 1989. The SOVA can be implemented in the so-called register exchange mode [53] or in the trace back mode [54]. The trace back mode is used in this treatise.

The SOVA has two modifications with respect to the soft decision Viterbi Algorithm (VA) that we have discussed in Section 2.3.4. Firstly, the path metrics used are modified, in order to take account of the intrinsic information $L(u_k)$. Secondly, the algorithm is modified so that it provides the a-posteriori information $L(u_k|\underline{y})$ for each decoded data bit.

First of all, we revisit the fundamental idea of the Viterbi algorithm, where the state sequence $\underline{s}_k$ of the maximum likelihood (ML) path is decided on the basis of the demodulator's soft output sequence $\underline{y}_{j\le k}$. The state sequence $\underline{s}_k$ gives the states along the surviving path at state $S_k = s$ at instant k in the trellis. Notice that the state sequence $\underline{s}_k$ is chosen based purely on the demodulator's soft output sequence $\underline{y}$ before and at instant k, $j \le k$. Based on these arguments, we need to maximise the following equation:

$$P(\underline{s}_k|\underline{y}_{j\le k}) = \frac{P(\underline{s}_k \wedge \underline{y}_{j\le k})}{P(\underline{y}_{j\le k})} \ , \qquad (3.46)$$

where Bayes' rule was invoked. Since the probability of encountering the received sequence $P(\underline{y}_{j\leq k})$ is constant for all the paths, we can equally maximise $P(\underline{s}_k \wedge \underline{y}_{j\leq k})$. Let us define the path metrics as

$$M(\underline{s}_k) := \ln P(\underline{s}_k \wedge \underline{y}_{j\leq k}) \; . \tag{3.47}$$

Assuming a memoryless channel, we will have

$$P(\underline{s}_k \wedge \underline{y}_{j\leq k}) = P(\underline{s}_{k-1} \wedge \underline{y}_{j\leq k-1}) \cdot P(S_k = s \wedge y_k | S_{k-1} = \grave{s}) \; . \tag{3.48}$$

Upon substituting Equation 3.48 into Equation 3.47, we arrive at:

$$\begin{aligned} M(\underline{s}_k) &= M(\underline{s}_{k-1}) + \ln\{P(S_k = s \wedge y_k | S_{k-1} = \grave{s})\} \\ &= M(\underline{s}_{k-1}) + \ln\{\gamma(\grave{s}, s)\} \; , \end{aligned} \tag{3.49}$$

and $\gamma(\grave{s}, s)$, is the branch transition probability for the path from state $S_{k-1} = \grave{s}$ to $S_k = s$ which has been defined in Section 3.3.3. Using Equation 3.34, we can rewrite Equation 3.49 as:

$$\begin{aligned} \ln\{\gamma_k(\grave{s}, s)\} &= \ln C + \frac{1}{2}u_k L(u_k) + \frac{1}{2}u_k L_c y_k \\ &= C' + \frac{u_k}{2}\{L(u_k) + L_c y_k\} \; , \end{aligned} \tag{3.50}$$

where $C' = \ln C$ is constant and it will cancel out, when we consider the path metric differences in the trellis. Hence, it can be omitted. Upon using Equation 3.50, we can rewrite Equation 3.49 as:

$$M(\underline{s}_k) = M(\underline{s}_{k-1}) + \frac{u_k}{2}\{L(u_k) + L_c y_k\} \; . \tag{3.51}$$

In Equation 3.51, we have shown how the intrinsic information $L(u_k)$ is included in the path metric calculation. Hence, we have accomplished the first required modification of the conventional VA.

In Section 2.2.2 we have seen that a BCH codeword consists of data and parity bits. In turbo decoding, only the soft outputs of the original data bits are passed from one decoder to another. There is no intrinsic information $L(u_k)$ for the parity bits. Therefore, for the parity bits Equation 3.51 is further reduced to

$$M(\underline{s}_k) = M(\underline{s}_{k-1}) + \frac{1}{2}u_k L_c y_k \; . \tag{3.52}$$

Let us now discuss the above mentioned second modification of the VA, namely the generation of the a-posteriori information $L(u_k|\underline{y})$, which allow us to compute the extrinsic information $L_e(u_k)$ from Equation 3.36. In the binary trellis there are two paths reaching state $S_k = s$ and one of the paths, namely the one having the smaller path metric, will be discarded. Let $\underline{s}_k$ and $\underline{\hat{s}}_k$ denote the ML path, exhibiting the highest path metric, and the discarded path at instant k, having the lower path metric, respectively. Then we can define the path metric difference as:

$$\Delta_k = M(\underline{s}_k) - M(\underline{\hat{s}}_k) \geq 0 \ . \tag{3.53}$$

The probability of the ML path $\underline{s}_k$ is

$$\begin{aligned} P(\underline{s}_k) &= \frac{P(\underline{s}_k)}{P(\underline{s}_k) + P(\underline{\hat{s}}_k)} \\ &= \frac{e^{M(\underline{s}_k)}}{e^{M(\underline{s}_k)} + e^{M(\underline{\hat{s}}_k)}} \\ &= \frac{e^{\Delta_k}}{1 + e^{\Delta_k}} \ . \end{aligned} \tag{3.54}$$

Therefore, the LLR of the decision concerning the ML path $\underline{s}_k$ is defined as follows:

$$L\{P(\underline{s}_k)\} = \ln \frac{P(\underline{s}_k)}{1 - P(\underline{s}_k)} = \Delta_k \ . \tag{3.55}$$

We have shown previously how to derive the LLR of the ML path $\underline{s}_k$ at instant k. Now, we

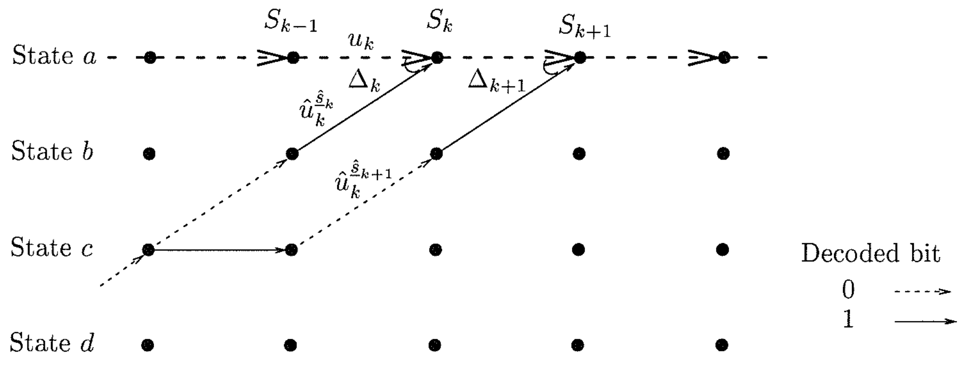


Figure 3.10: Simple SOVA decoding example.

need to find the LLR of the decoded bits u_k associated with the survivor path $\underline{s}_k$. Figure 3.10 shows a simplified section of a trellis, where the ML path is the all-zero state sequence. We can see from the figure that path $c \rightarrow b \rightarrow a$, namely $\underline{\hat{s}}_k$, and path $c \rightarrow c \rightarrow b \rightarrow a$, namely $\underline{\hat{s}}_{k+1}$, are the discarded paths at instants k and $k+1$, respectively. Let $\hat{u}_k^{\underline{\hat{s}}_k}$ and $\hat{u}_k^{\underline{\hat{s}}_{k+1}}$ be the decoded bit at instant k, which would have been the output associated with the discarded paths $\underline{\hat{s}}_k$ and $\underline{\hat{s}}_{k+1}$, had they not been discarded, respectively. As shown in Figure 3.10, u_k and $\hat{u}_k^{\underline{\hat{s}}_k}$ differ. Hence, the LLR of the actually decoded bit u_k is proportional to the

LLR Δ_k of the survivor path. On the other hand, u_k and $\hat{u}_k^{\hat{\underline{s}}_{k+1}}$ are the same, namely '0'. Therefore, we would have made no mistake concerning the decoded bit u_k, irrespective of whether path $\underline{s}_k$ or $\hat{\underline{s}}_{k+1}$ was chosen as the most likely state sequence at trellis stage S_{k+1}. The LLR of the decoded bit u_k is then ∞. We define the LLR of u_k taking into account a discarded path $\hat{\underline{s}}_{k+i}$ as:

$$L_{\hat{\underline{s}}_{k+i}}(u_k) := u_k.\begin{cases} \infty & \text{if } u_k = \hat{u}_k^{\hat{\underline{s}}_{k+i}} \\ \Delta_k & \text{if } u_k \neq \hat{u}_k^{\hat{\underline{s}}_{k+i}} \end{cases}, \tag{3.56}$$

where $i \geq 0$.

However, Equation 3.56 considers only one discarded path, while along the ML path, we would have a number of discarded paths. In Figure 3.10, at any node of the trellis there is a survivor and a discarded path. Upon tracing the trellis backwards across i trellis stages, we have to consider a total of $i + 1$ discarded paths in deriving the a-posteriori information $L(u_k|\underline{y})$. It was shown by Hagenauer [54] that the LLR of the decoded bit u_k can be approximated by

$$L(u_k|\underline{y}) = u_k \cdot \min_{\substack{j=k\dots n \\ u_k \neq \hat{u}_k^{\hat{\underline{s}}_j}}} \Delta_j \ . \tag{3.57}$$

In order to interpret Equation 3.57, let us consider the following arguments. The algorithm aims to explore, whether any unreliable decisions have been encountered, while traversing through the trellis, which may have resulted in a decision error. These potentially unreliable decisions are associated with discarding paths in the trellis, which had a metric value similar to that of the survivor. These decisions are associated with a low difference Δ between the metrics of the survivor and that of discarded path. This is, why the algorithm attempts to find the trellis node, where the path metric difference was the lowest and, additionally, favouring the similar confidence discarded oath would have resulted in a different bit decision.

3.3.5.1 SOVA Decoding Example

In this section we will augment the concept of the SOVA decoding process using the same example as in Section 2.3.4. We employ BPSK modulation and hence a logical 0 will be transmitted as -1.0 and a logical 1 is sent as $+1.0$. Instead of producing the hard output of the decoded bits, we are going to calculate their soft outputs. In order to simplify the calculations, we assume that $L_c = 1$ in Equation 3.8. Furthermore, initially there is no intrinsic information for the data bits u_k. Hence, the LLR values $L(u_k)$ are all reset to 0, implying a probability of 0.5.

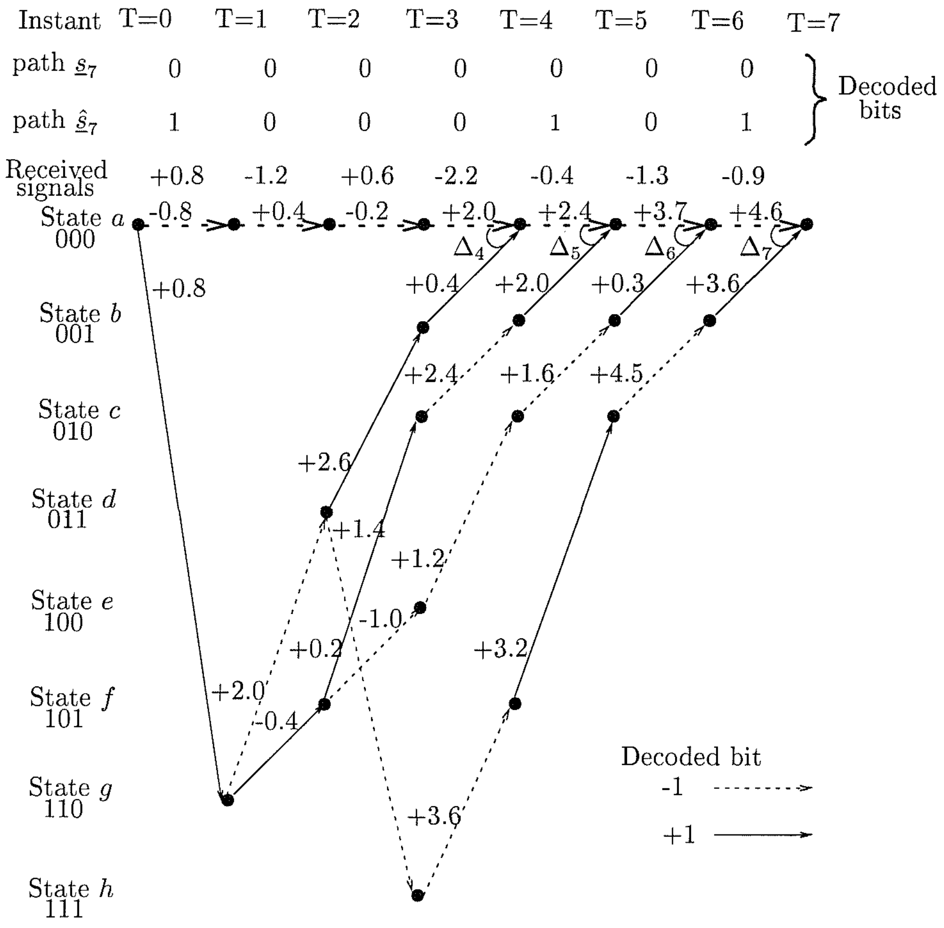


Figure 3.11: An example of SOVA decoding using the BCH(7,4,3) code.

Figure 3.11 shows our example of the SOVA decoding procedure using the BCH(7,4,3) code. The trellis diagram of the figure is the same as that in Figure 2.10 and so are the demodulator soft output values as well as the accumulated path metrics. The Viterbi decoder proceeds in the usual way, finding the ML path $\underline{s}_k$, which is the all zero sequence in this example, by calculating the path metrics. At the end of the trellis, i.e. at stage $T = 7$, the ML path is found and the trace-back procedure [54] begins, in order to find all discarded paths, which would have resulted in different bit decisions.

As seen in Figure 3.11, at instant $T = 7$, the path metric difference between the survivor path $\underline{s}_k$ and the discarded path $\underline{\hat{s}}_k$, according to Equation 3.53 is given by:

$$\begin{aligned}\Delta_7 &= M(\underline{s}_7) - M(\underline{\hat{s}}_7) \\ &= \frac{1}{2}(4.6 - 3.6) \\ &= 0.5 \ .\end{aligned}$$

In this SOVA decoder scenario the decoder starts tracing back the survivor path $\underline{s}_7$ and the discarded path $\hat{\underline{s}}_7$ which merge at $T = 7$. Hence their associated decoded bits are found, which are summarised in Table 3.1 for both paths. For each decoding instant, the exclusive-or (XOR) of the decoded data bits is taken, which will give '1', if and only if u_k and $\hat{u}_k^{\hat{\underline{s}}_7}$ are different. If the decoded bits are different at an instant, then according to Equation 3.56 the associated LLR becomes $L_{\hat{\underline{s}}_7}(u_k) = u_k.\Delta_7$. On the other hand, if the decoded data bits are the same for an instant, then there is no ambiguity about the data bit given by the ML path. Therefore, the corresponding value of $L_{\hat{\underline{s}}_7}(u_k)$ is $u_k.\infty$.

	Instant T						
	0	1	2	3	4	5	6
u_T	0	0	0	0	0	0	0
$u_T^{\hat{\underline{s}}_7}$	1	0	0	0	1	0	1
u_T XOR $u_T^{\hat{\underline{s}}_7}$	1	0	0	0	1	0	1
$\lvert L_{\hat{\underline{s}}_7}(u_k)\rvert$	0.5	∞	∞	∞	0.5	∞	0.5

Table 3.1: Decoded bits for path $\underline{s}_k$ and $\hat{\underline{s}}_k$.

We can see from Figure 3.11 that there are four discarded paths along the all-zero ML path, yielding the path metric of 4.6. Table 3.2 shows the calculated $|L_{\hat{\underline{s}}_k}(u_k)|$ of the data bits for each discarded path in Figure 3.11. Specifically, the survivor path $\underline{s}_7$ and the discarded path $\hat{\underline{s}}_7$ are associated with the decoded bits 0, 0, 0, 0, 0, 0, 0 and 1, 0, 0, 0, 1, 0, 1, respectively. When considering the associated decoded bits u_k, they are different for $k = 0$, but identical for $k = 1, 2$ and 3. Hence in Table 3.2 we have $|L_{\hat{\underline{s}}_7}(u_k)| = 0.5, \infty, \infty$ and ∞ for $k = 0, 1, 2$ and 3, respectively.

Similarly, at $T = 6$ the survivor path results in the decoded bits 0, 0, 0, 0, 0, 0, 0, while the discarded path in 1, 1, 0, 0, 0, 0, 1. Hence for $k = 0$ and 1 they are different, while for $k = 2$ and 3 the decoded bits are identical, yielding $|L_{\hat{\underline{s}}_6}(u_0)| = |L_{\hat{\underline{s}}_6}(u_1)| = 1.7$ and $|L_{\hat{\underline{s}}_6}(u_2)| = |L_{\hat{\underline{s}}_6}(u_3)| = 0$, as seen in Table 3.2. The remaining values in Table 3.2 can be derived similarly. Referring to Equation 3.57, we have to find the smallest Δ_k for which $u_k \neq \hat{u}_k$, i.e. the smallest $|L_{\hat{\underline{s}}_k}(u_k)|$ since this approximates the LLR of the survivor path. Hence, the minimum is taken along each column in Table 3.2 . According to Equation 3.57, the a-posteriori information $L(u_k|\underline{y})$ of bit u_k is then the polarity of the decoded data bit u_k times the minimum path metric difference Δ in Table 3.2.

In Section 3.3.3.4, we have derived the relationship between the a-posteriori information $L(u_k|\underline{y})$ and the extrinsic information $L_e(u_k)$. In order to calculate the extrinsic information,

$L_{\underline{s}_k}(u_k)$	Data bits u_k			
	u_0	u_1	u_2	u_3
$\lvert L_{\underline{s}_7}(u_k)\rvert$	0.5	∞	∞	∞
$\lvert L_{\underline{s}_6}(u_k)\rvert$	1.7	1.7	∞	∞
$\lvert L_{\underline{s}_5}(u_k)\rvert$	0.2	0.2	0.2	∞
$\lvert L_{\underline{s}_4}(u_k)\rvert$	0.8	∞	0.8	0.8
Minimum Δ	0.2	0.2	0.2	0.8
u_k	-1	-1	-1	-1
$L(u_k\lvert\underline{y})$	-0.2	-0.2	-0.2	-0.8

Table 3.2: LLR for the decoded data bits.

we rewrite Equation 3.35 as:

$$L_e(u_k) = L(u_k|\underline{y}) - L_c y_k - L(u_k) \tag{3.58}$$

Applying Equation 3.58, we can find the extrinsic information $L_e(u_k)$ of the data bits by removing the soft channel output $L_c y_k$ and the intrinsic information $L(u_k)$ from the a-posteriori information $L(u_k|\underline{y})$. Specifically, $L(u_k|\underline{y})$ is given at the bottom of Table 3.2, $L_c y_k$ is demodulator's soft output seen as 'received signal' in Figure 3.11, while $L(u_k) = 0$ in the first iteration, since all bit probabilities are 0.5. In summary, Table 3.3 shows the extrinsic information of the data bits in our example. The extrinsic information will then be passed to the second decoder in Figure 3.3.

	Data bits u_k			
	u_0	u_1	u_2	u_3
$L(u_k\lvert\underline{y})$	-0.2	-0.2	-0.2	-0.8
$L_c y_k$	$+0.8$	-1.2	$+0.6$	-2.2
$L(u_k)$	0.0	0.0	0.0	0.0
$L_e(u_k)$	-1.0	-1.4	-0.8	$+1.4$

Table 3.3: The a-posteriori information, soft channel outputs, intrinsic and extrinsic information of the data bits.

3.4 Turbo Decoding Example

In this section, we discuss an example of turbo decoding using the SOVA algorithm detailed in Section 3.3.5. This example illustrates how iterative decoding assists in correcting multiple errors. Our elaborations are based on Sections 2.2, 2.3 and 3.3.5.

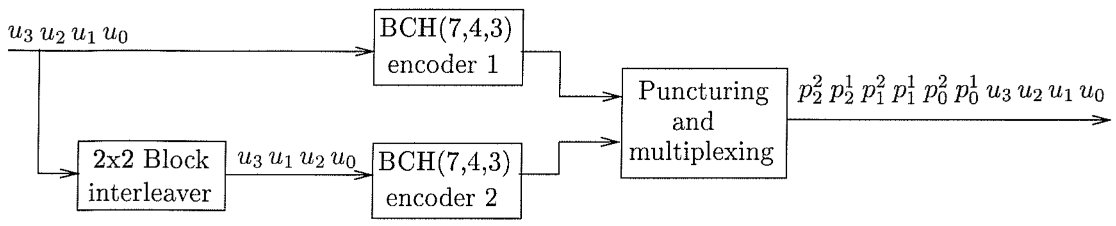
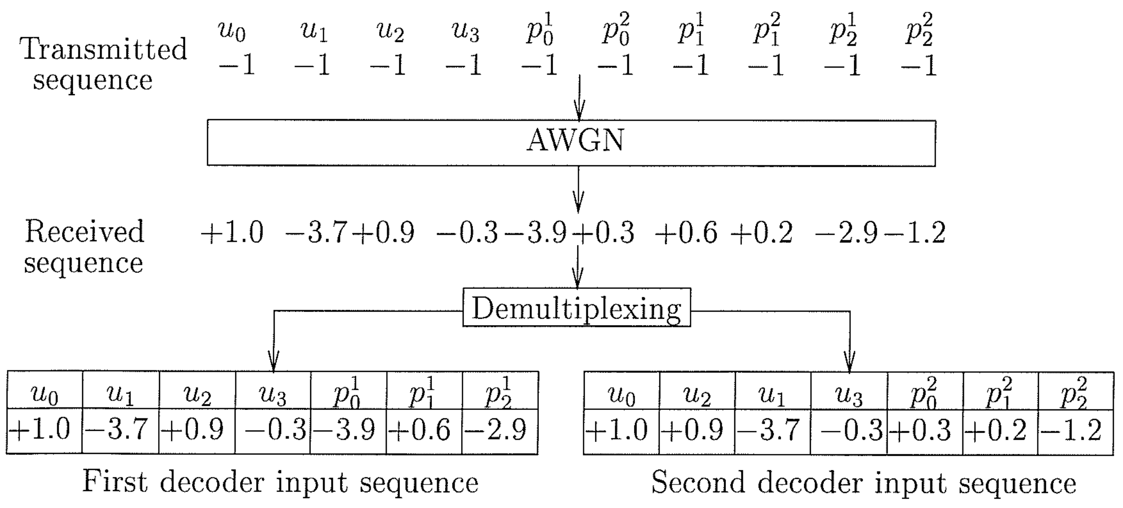


Figure 3.12: BCH(7,4,3) turbo encoder.

The component codes used in this example are two BCH(7,4,3) codes, which are combined, as shown in Figure 3.12, with a 2×2 block interleaver [112] for creating a simple turbo code. As we can see from Figure 3.12, the parity bits p_k^1, generated by BCH encoder 1, and the parity bits p_k^2, of BCH encoder 2, are not punctured. Since the data bits of both component codes are the same, all the data bits of the second component code are punctured. The transmitted sequence therefore contains four data bits and six parity bits, resulting in a code rate of 0.4.

For the sake of simplicity, we assume that all data bits are binary zeros. Hence, both BCH(7,4,3) encoders generate an all-zero output sequence. Assuming that BPSK modulation is used, a logical 0 is transmitted as -1, whereas a logical 1 is sent as $+1$. The transmitted sequence of the turbo BCH encoder is hence a series of ten consecutive -1s.



u_0	u_1	u_2	u_3	p_0^1	p_1^1	p_2^1
+1.0	−3.7	+0.9	−0.3	−3.9	+0.6	−2.9

u_0	u_2	u_1	u_3	p_0^2	p_1^2	p_2^2
+1.0	+0.9	−3.7	−0.3	+0.3	+0.2	−1.2

Figure 3.13: Demultiplexing process of the received sequence.

As we can see in Figure 3.13, the transmitted sequence, which has an energy per bit of $E_b = 1$, is conveyed through an AWGN channel. Since there is no fading in an AWGN channel, the fading amplitude is $a = 1$ and the variance of the AWGN is $\sigma = \sqrt{2}$. Hence the channel reliability value Lc of Equation 3.8 is given by:

$$\begin{aligned} L_c &= \frac{E_B}{2\sigma^2}.4a \\ &= 1\ . \end{aligned} \tag{3.59}$$

The received sequence $\underline{y}$ has been corrupted by noise, which is passed to the demultiplexer that separates the received sequence to give the input sequence of the first and second decoder, as shown in Figure 3.13. In both decoders, the received samples due to the original data bits are the same, except that the sequence of the data bits of the second decoder was interleaved. This assists in increasing the error correction capability. This is because if the first decoder fails to correct the errors, the second decoder will use the rearranged received data bit sequence to correct the errors, and vice versa. In our following discourse we demonstrate how both decoders assist each other in correcting the errors.

The path metric of the SOVA algorithm is given by Equation 3.51, which is repeated here for convenience:

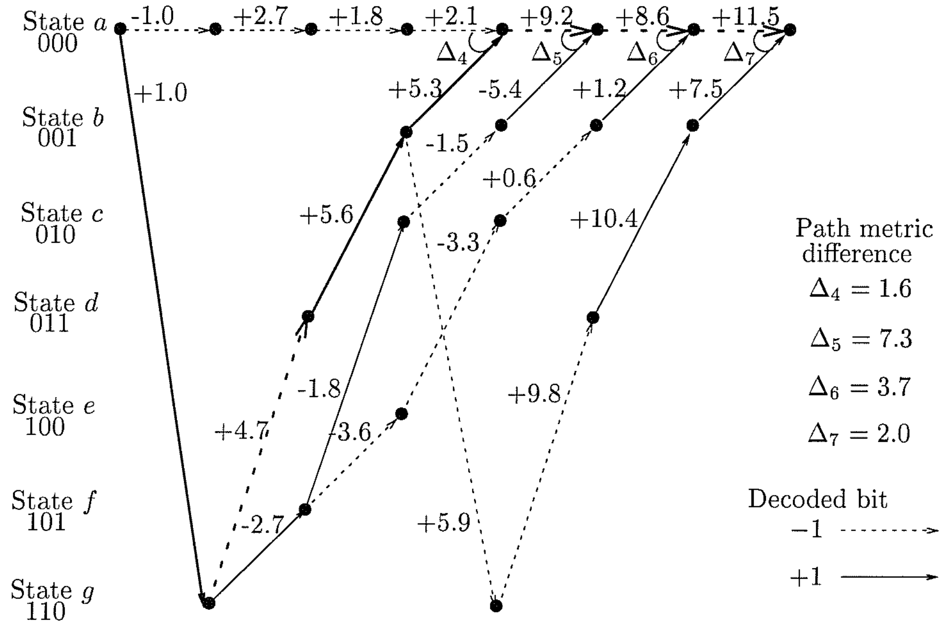
$$M(\underline{s}_k) = M(\underline{s}_{k-1}) + \frac{u_k}{2}\left\{L(u_k) + L_c y_k\right\}\ . \tag{3.60}$$

Here $M(\underline{s}_{k-1})$ is the path metric for the surviving path traversing through the state $S_{k-1} = \grave{s}$ at stage $k-1$ in the trellis, u_k is the estimated transmitted bit associated with a given transition, y_k is the demodulator's soft output sample for that transition and L_c is the associated channel reliability value, which was found to be unity in Equation 3.59. Hence, $L_c y_k = y_k$.

Initially, we consider the operation of the first decoder during its first iteration, where the associated trellis is shown in Figure 3.14. There is no a-priori information and hence we have $L(u_k) = 0$ for all k, which corresponds to the a-priori probability of 0.5, as shown in Figure 3.4. The received signal $L_c y_k$ constituted by the demodulator's soft output is the first decoder's input sequence in Figure 3.13. According to Equation 3.60, the transition metrics are derived by the summation of the intrinsic information $L(u_k)$ and the received signal $L_c y_k$ constituted by the demodulator's soft output, at each trellis transition. The values of the intrinsic information, the received signals and the resultant transition metrics are shown at the top of Figure 3.14 for every instant.

The SOVA proceeds in the same way as the Viterbi algorithm in order to find the ML path (or survivor path) which is shown by the bold line in Figure 3.14. As we can see in the figure, the ML path of the first decoder in the first iteration is the state sequence of $a \rightarrow g \rightarrow d \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow a$, which is not the all-zero state sequence. Similarly to our example in Section 3.3.5.1, there are four discarded paths along the ML path. The decoded bits, of the ML path u_k and those that would have been produced by the four

Instant	T=0	T=1	T=2	T=3	T=4	T=5	T=6	T=7
Intrinsic information	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$L(u_k)$
Received signals	+1.0	-3.7	+0.9	-0.3	-3.9	+0.6	-2.9	$L_c y_k$
Transition metrics	+1.0	-3.7	+0.9	-0.3	-3.9	+0.6	-2.9	$L(u_k) + L_c y_k$



Decoded bits	**+1**	**−1**	**+1**	**+1**	**−1**	**−1**	**−1**	u_k
Discarded path $\underline{\hat{s}}_4$	−1	−1	−1	−1	−1	−1	−1	$u_k^{\hat{s}_4}$
Dsicarded path $\underline{\hat{s}}_5$	+1	+1	+1	−1	+1	−1	−1	$u_k^{\hat{s}_5}$
Discarded path $\underline{\hat{s}}_6$	+1	+1	−1	−1	−1	+1	−1	$u_k^{\hat{s}_6}$
Discarded path $\underline{\hat{s}}_7$	+1	−1	+1	−1	−1	+1	+1	$u_k^{\hat{s}_7}$

Figure 3.14: Trellis diagram for the SOVA decoding during the first iteration of the first decoder.

discarded paths $u_k^{\hat{s}_k}$, are shown at the bottom of Figure 3.14 for the convenience of the reader. In Section 3.3.5, we defined the path metric difference and here we restate this equation as follows:

$$\Delta_k = M(\underline{s}_k^s) - M(\hat{\underline{s}}_k^s) \ , \tag{3.61}$$

where $M(\underline{s}_k^s)$ and $M(\hat{\underline{s}}_k^s)$ are the path metrics of ML path $\underline{s}$ and the discarded path $\hat{\underline{s}}$, respectively. Since there are four discarded paths along the ML path, there will be four path metric differences Δ_k. The path metric difference Δ_k for instants 4 to 7 is shown at the right of the trellis in Figure 3.14. Specifically, at $T = 4$ in state a the discarded path is $a \to g \to d \to b \to a$ and the path metric difference is $\Delta_4 = (5.3 - 2.1)/2 = 1.6$. Similarly, at $T = 5$ in state a the discarded path is $a \to g \to f \to c \to b \to a$ and we have $\Delta_5 = \{9.2 - (-5.4)\}/2 = 7.3$. The remaining path metric difference can be derived similarly.

k	$L_{\hat{\underline{s}}_k}(u_k)$				Min	u_k	$L(u_k\|\underline{y})$	$L_c y_k$	$L(u_k)$	$L_e(u_k)$
	$\|L_{\hat{\underline{s}}_4}(u_k)\|$	$\|L_{\hat{\underline{s}}_5}(u_k)\|$	$\|L_{\hat{\underline{s}}_6}(u_k)\|$	$\|L_{\hat{\underline{s}}_7}(u_k)\|$						
0	1.6	∞	∞	∞	1.6	+1	+1.6	+1.0	0.0	+0.6
1	∞	7.3	3.7	∞	3.7	−1	−3.7	−3.7	0.0	0.0
2	1.6	∞	3.7	∞	1.6	+1	+1.6	+0.9	0.0	+0.7
3	1.6	7.3	3.7	2.0	1.6	+1	+1.6	−0.3	0.0	+1.9

Table 3.4: SOVA output for the first iteration of the first decoder in terms of the a-posteriori information $L(u_k|\underline{y})$, soft channel outputs $L_c y_k$, intrinsic information $L(u_k)$ and extrinsic information $L_e(u_k)$.

The a-posteriori information $|L_{\hat{\underline{s}}_k}(u_k)|$ of the decoded bit u_k taking into account the discarded path $\hat{\underline{s}}_k$ is shown in Table 3.4, for all four discarded paths. We have shown in Equation 3.56 that $|L_{\hat{\underline{s}}_k}(u_k)| = \Delta_k$ if u_k and $u_k^{\hat{s}_k}$ differ, while $|L_{\hat{\underline{s}}_k}(u_k)| = \infty$, if the decoded bits associated with both the survivor path and the discarded path are the same. Again in Figure 3.14 the ML path $\underline{s}_4$ converges with the all-zero state sequence at instant $T = 4$ and the path metric difference is $\Delta_4 = 1.6$. Since the decoded bits of the ML path and the discarded path differ at instant $k = 0, 2$ and 3, the values of the a-posteriori information $|L_{\hat{\underline{s}}_4}(u_k)|$ for the decoded bits u_k, $k = 0, 2$ and 3, are equal to the path metric difference $\Delta_4 = 1.6$. By contrast, at $k = 1$ we have $|L_{\hat{\underline{s}}_4}(u_1)| = \infty$. This is shown in the second column of Table 3.4. Similarly, at $T = 5$ the survivor path is associated with the data bits 1, 0, 1, 1 while the discarded path with bits 1, 1, 1, 0. Hence at $k = 0$ and 2, we have $|L_{\hat{\underline{s}}_5}(u_k)| = \infty$, while at $k = 1$ and 3, $|L_{\hat{\underline{s}}_5}(u_k)| = 7.3$, as seen in the third column of Table 3.4. The associated a-posteriori information for other discarded paths are shown in the consecutive columns as well.

In order to derive the a-posteriori information $L(u_k|\underline{y})$ of the decoded bit u_k, we can approximate the LLR as follows:

$$L(u_k|\underline{y}) = u_k \cdot \min_{\substack{T=k\ldots n \\ u_k \neq \hat{u}_k^{\underline{\hat{s}}_T}}} \Delta_T \ . \tag{3.62}$$

Equation 3.62 is the same as Equation 3.57 and restated here for convenience. In Table 3.4, at trellis stage $k = 3$, the values of $|L_{\underline{\hat{s}}_4}(u_3)|$, $|L_{\underline{\hat{s}}_5}(u_3)|$, $|L_{\underline{\hat{s}}_6}(u_3)|$ and $|L_{\underline{\hat{s}}_7}(u_3)|$ are 1.6, 7.3, 3.7 and 2.0 corresponding to the four discarded paths respectively and the minimum path metric difference in this set is 1.6. Using Equation 3.62, we can derive the a-posteriori information of u_4, yielding $L(u_4|\underline{y}) = +1.6$, since the decoded bit is $u_4 = +1$. Figure 3.3 shows that now we have to derive the extrinsic information $L_e(u_k)$, which is given by Equation 3.58 in Section 3.3.5.1 as:

$$L_e(u_k) = L(u_k|\underline{y}) - L(u_k) - L_c y_k \ . \tag{3.63}$$

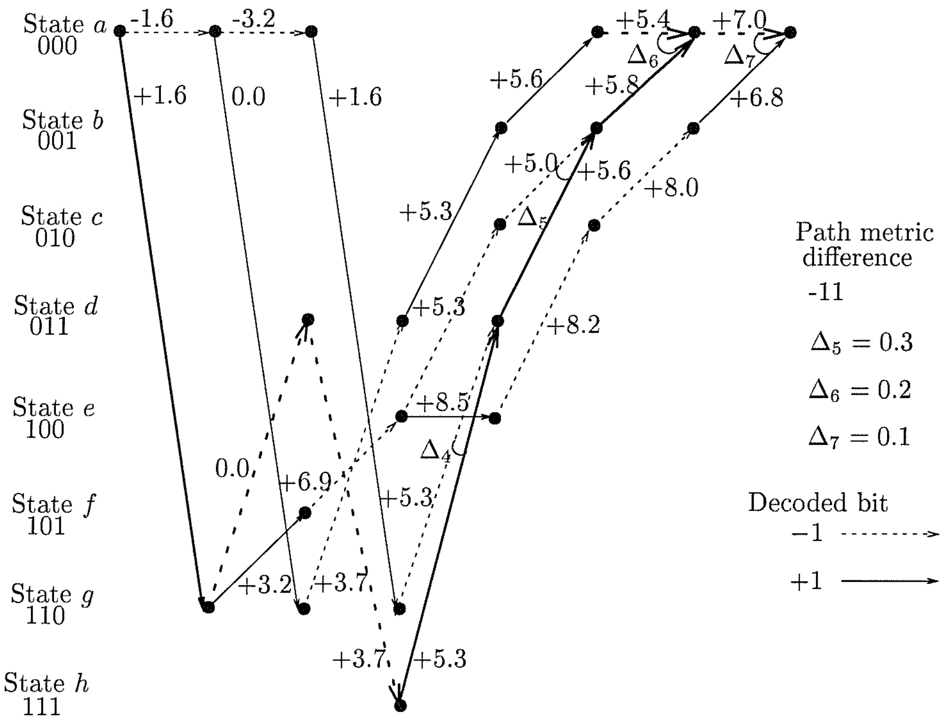
This equation states that the extrinsic information $L_e(u_k)$ is given by subtracting the intrinsic information $L(u_k)$ and the received signal $L_c y_k$ from the a-posteriori information $L(u_k|\underline{y})$ of the decoder. The last column in Table 3.4 shows the extrinsic information calculated from Equation 3.63.

We now proceed to describing the operation of the second component decoder during its first iteration. The a-priori information $L(u_k)$ of the second decoder is then the extrinsic information produced by the first decoder, after interleaving by a 2×2 block interleaver which was shown in Figure 3.3. It also uses the interleaved demodulator soft outputs $L_c y_k$ and the received parity bits produced by the second BCH encoder, i.e. the second decoder input sequence in Figure 3.13.

Figure 3.15 shows the SOVA decoding trellis of the second decoder during its first iteration. The extrinsic information values extracted from Table 3.4 – after interleaving – are shown as the a-priori information $L(u_k)$ at top of the trellis. Also shown is the received signal $L_c y_k$ constituted by the demodulator's soft output and the transition metrics $L(u_k) + L_c y_k$. The decoded bits of the ML path and those of the four other discarded paths are shown at the bottom of the trellis.

The ML path chosen by the second component decoder is shown by the bold line in Figure 3.15. The associated state sequence is $a \to g \to d \to h \to d \to b \to a \to a$ and again, it is not the all-zero state sequence. Using Equation 3.61, we can calculate the path metric difference Δ_k of the ML path and the discarded paths, which are shown at the right of the trellis in Figure 3.15. Note that in Figure 3.14 the intrinsic information $L(u_k)$ was zero, while in Figure 3.15 we have valuable intrinsic information $L(u_k)$ provided by the

Instant	T=0	T=1	T=2	T=3	T=4	T=5	T=6	T=7
Intrinsic information	+0.6	+0.7	0.0	+1.9	0.0	0.0	0.0	$L(u_k)$
Received signals	+1.0	+0.9	-3.7	-0.3	+0.3	+0.2	-1.2	$L_c y_k$
Transition metrics	+1.6	+1.6	-3.7	+1.6	+0.3	+0.2	-1.2	$L(u_k) + L_c y_k$



Decoded bits	**+1**	**−1**	**−1**	**+1**	**+1**	**+1**	**−1**	u_k
Discarded path $\hat{s}_4$	−1	−1	+1	−1	+1	+1	−1	$u_k^{\hat{s}_4}$
Dsicarded path $\hat{s}_5$	+1	+1	−1	−1	−1	+1	−1	$u_k^{\hat{s}_5}$
Discarded path $\hat{s}_6$	−1	+1	−1	+1	+1	−1	−1	$u_k^{\hat{s}_6}$
Discarded path $\hat{s}_7$	+1	+1	−1	+1	−1	−1	+1	$u_k^{\hat{s}_7}$

Figure 3.15: Trellis diagram for the SOVA decoding during the first iteration of the second decoder.

first decoder. This substantially changes the associated transition metrics and path metric differences.

k	$L_{\underline{\hat{s}}_k}(u_k)$				Min	u_k	$L(u_k\|\underline{y})$	$L_c y_k$	$L(u_k)$	$L_e(u_k)$
	$\|L_{\underline{\hat{s}}_4}(u_k)\|$	$\|L_{\underline{\hat{s}}_5}(u_k)\|$	$\|L_{\underline{\hat{s}}_6}(u_k)\|$	$\|L_{\underline{\hat{s}}_7}(u_k)\|$						
0	6.9	∞	0.2	∞	0.2	+1	+0.2	+1.0	+0.6	−1.4
1	∞	0.3	0.2	0.1	0.1	−1	−0.1	+0.9	+0.7	−1.7
2	6.9	∞	∞	∞	6.9	−1	−6.9	−3.7	0.0	−3.2
3	6.9	0.3	∞	∞	0.3	+1	+0.3	−0.3	+1.9	−1.3

Table 3.5: SOVA output for the first iteration of the second decoder in terms of the a-posteriori information $L(u_k|\underline{y})$, soft channel outputs $L_c y_k$, intrinsic information $L(u_k)$ and extrinsic information $L_e(u_k)$.

Using the same steps as described in the context of the first decoder seen in Figure 3.14, we can calculate the a-posteriori information $L(u_k|\underline{y})$ and the extrinsic information $L_e(u_k)$, which are shown in Table 3.5. Table 3.5 has the same structure as Table 3.4. Considering Figure 3.5 in a little more depth, at $T = 4$ we observe in state d that the path metric difference is $\Delta_4 = 6.9$ between the survivor and the discarded paths. Their decoded bit sequences are 1, 0, 0, 1 and 0, 0, 1, 0, respectively. The only coincident bit decision is at $k = 1$, where we therefore have $|L_{\underline{\hat{s}}_4}(u_1)| = \infty$ in Table 3.5. By contrast, for $k = 0, 2$ and 3 we have $|L_{\underline{\hat{s}}_4}(u_0)| = |L_{\underline{\hat{s}}_4}(u_2)| = |L_{\underline{\hat{s}}_4}(u_3)| = 6.9$.

Similarly, at $T = 5$ and in state b the survivor and discarded paths have decoded patterns of 1, 0, 0, 1 and 1, 1, 0, 0, respectively, which are identical for $k = 0$ and 2. Hence the associated a-posteriori information in Table 3.5 at $k = 0$ and 2 is ∞, while at $k = 1$ and 3 it is $\Delta_5 = (5.6 - 5.0)/2 = 0.3$. The remaining a-posteriori information values seen in Table 3.5 accrue similarly. Their minimal are found for each value of k, which are also listed in column six of Table 3.5. According to Equation 3.62 the a-posteriori values $L(u_k|\underline{y})$ are given by the product of the decoded bits u_k, $k = 0, .., 3$, and the above-mentioned minimum path metric difference. They are summarised in column eight of Table 3.5.

Column nine of Table 3.5 repeats the soft demodulator outputs due to the received signal samples +1.0, +0.9, −3.7 and −0.3 extracted from Figure 3.15, while column ten summarises the intrinsic values of +0.6, +0.7, 0.0 and +1.9 provided by the other decoder. These values can be seen in both Figure 3.15 and in Table 3.5. Finally, the extrinsic values of the last column are derived from Equation 3.63.

Let us now compare the a-posteriori information $L(u_k|\underline{y})$ of the second decoder and the first decoder. In the first decoder characterised in Figure 3.14, we have three decoding errors in the data bits sequence and the number of errors is reduced to two in the second decoder of

Figure 3.15. The soft outputs of the data bits u_0, u_1 and u_3, produced by the second decoder, which are the interleaved data bits u_0, u_2 and u_3 of the first decoder, are relatively low and are close to zero which indicates a low confidence. Indeed, by comparing columns eight of Tables 3.4 and 3.5 we find that the confidence values on the whole have been reduced by invoking the second decoder. At first sight this confidence measure reduction might appear undesirable. However, we show that after the first decoding three of the bits were erroneous and hence the polarity of the associated LLRs was wrong. We expect the decoder to eventually change the polarity of these bits during the forthcoming iterations. This can only be achieved by first reducing the LLRs' magnitude and then changing their polarity during the successive iterations. Once the erroneous LLR polarity has been changed, it may become possible to increase the magnitude of the corresponding LLRs, which reflects their increased confidence. By contrast, data bit u_2, which is the interleaved data bit u_1 of the first decoder, shows a significant increase in terms of its reliability or confidence. In summary, we observe that the first decoder provides an estimation of the data bits and that the second decoder improves the reliability of the soft outputs.

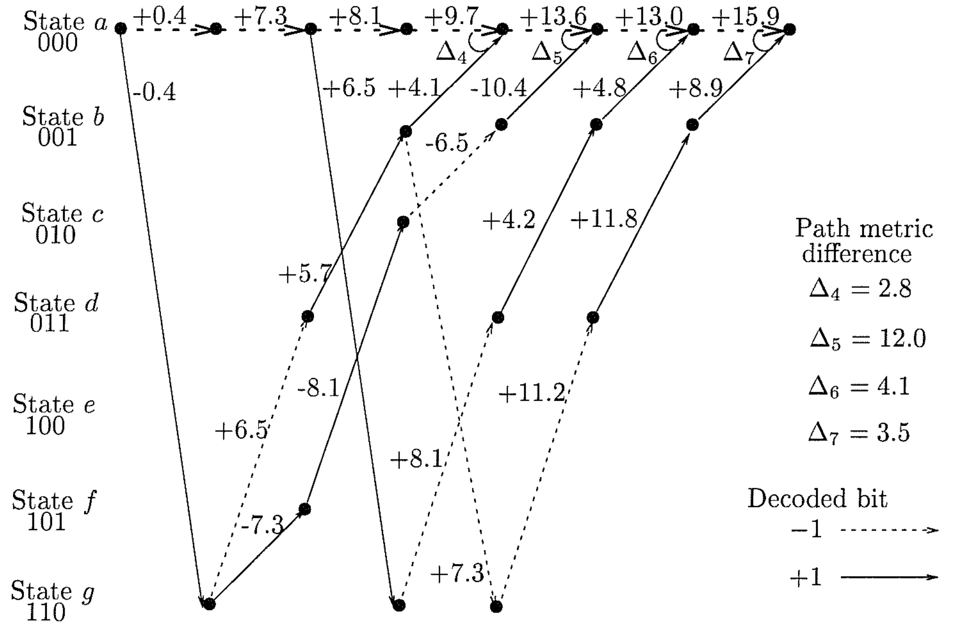
In the second, and all subsequent, iterations the first component decoder is capable of using the extrinsic information provided by the second decoder in the previous iteration as intrinsic information. As a further step, Figure 3.16 shows the trellis diagram of SOVA decoding in the first decoder during the second iteration. It can be seen in the figure that this decoder uses the same channel information $L_c y_k$, as it did in the first iteration. In contrast to Figure 3.14, it has however intrinsic information provided by the second decoder during the first iteration, in order to assist in finding the correct path through the trellis. The selected ML path is again shown by the bold line in Figure 3.14 and it can be seen that now the correct all zero path is chosen. Again, the a-posteriori information $L(u_k|\underline{y})$ and the extrinsic information $L_e(u_k)$ are calculated and summarised in Table 3.6.

	$L_{\underline{s}_k}(u_k)$									
k	$\lvert L_{\underline{s}_4}(u_k)\rvert$	$\lvert L_{\underline{s}_5}(u_k)\rvert$	$\lvert L_{\underline{s}_6}(u_k)\rvert$	$\lvert L_{\underline{s}_7}(u_k)\rvert$	Min	u_k	$L(u_k\lvert\underline{y})$	$L_c y_k$	$L(u_k)$	$L_e(u_k)$
0	2.8	12.0	∞	3.5	2.8	−1	−2.8	+1.0	−1.4	−2.4
1	∞	12.0	∞	∞	12.0	−1	−12.0	−3.7	−3.2	−5.1
2	2.8	12.0	4.1	3.5	2.8	−1	−2.8	+0.9	−1.7	−2.0
3	2.8	∞	∞	∞	2.8	−1	−2.8	−0.3	−1.3	−1.2

Table 3.6: SOVA output for the second iteration of the first decoder in terms of the a-posteriori information $L(u_k|\underline{y})$, soft channel outputs $L_c y_k$, intrinsic information $L(u_k)$ and extrinsic information $L_e(u_k)$.

The second iteration is then completed by finding the extrinsic information generated by

Instant	T=0	T=1	T=2	T=3	T=4	T=5	T=6	T=7
Intrinsic information	-1.4	-3.2	-1.7	-1.3	0.0	0.0	0.0	$L(u_k)$
Received signals	+1.0	-3.7	+0.9	-0.3	-3.9	+0.6	-2.9	$L_c y_k$
Transition metrics	-0.4	-6.9	-0.8	-1.6	-3.9	+0.6	-2.9	$L(u_k) + L_c y_k$



Decoded bits	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	u_k
Discarded path $\hat{\underline{s}}_4$	+1	−1	+1	+1	−1	−1	−1	$u_k^{\hat{s}_4}$
Dsicarded path $\hat{\underline{s}}_5$	+1	+1	+1	−1	+1	−1	−1	$u_k^{\hat{s}_5}$
Discarded path $\hat{\underline{s}}_6$	−1	−1	+1	−1	+1	+1	−1	$u_k^{\hat{s}_6}$
Discarded path $\hat{\underline{s}}_7$	+1	−1	+1	−1	−1	+1	+1	$u_k^{\hat{s}_7}$

Figure 3.16: Trellis diagram for the SOVA decoding in the second iteration of the first decoder.

the first decoder, interleaving it and using it as intrinsic information for the second decoder. It can be shown that the second decoder will also select the all-zero path as the ML path, and hence the output of the turbo decoder after the second iteration will be the correct all -1 sequence. This concludes our example concerning the operation of iterative turbo decoding using the SOVA algorithm.

3.5 MAP Algorithm For Extended BCH codes

3.5.1 Introduction

This section presents a block turbo decoder using extended BCH codes as the component codes. The MAP algorithm is modified in order to incorporate the parity check bit into calculating the LLR of the decoded bits. Identical results are expected, when using the alternative approach of creating an extended trellis for the corresponding extended codeword length.

Conventional BCH codes are denoted by BCH(n, k, d_{min}), where n, k, d_{min} denote the codeword length, the number of information data bits and the minimum free distance, respectively. These codes are referred to as extended BCH codes [96], if a parity check bit is appended to the BCH codeword. This extends the primary BCH codeword by one bit, to $n+1$, and it expands its minimum free distance from d_{min} to $d_{min}+1$. Hence the extended BCH code is denoted by BCH$(n+1, k, d_{min}+1)$. The process of code extension [96] is also referred to as code expansion in [90]. Figure 3.17 shows the weight distribution [19] of both the BCH(31,26,3) and that of the extended BCH(32,26,4) codes. Notice that the extended BCH code has only even weight terms because of the effect of the parity check bit.

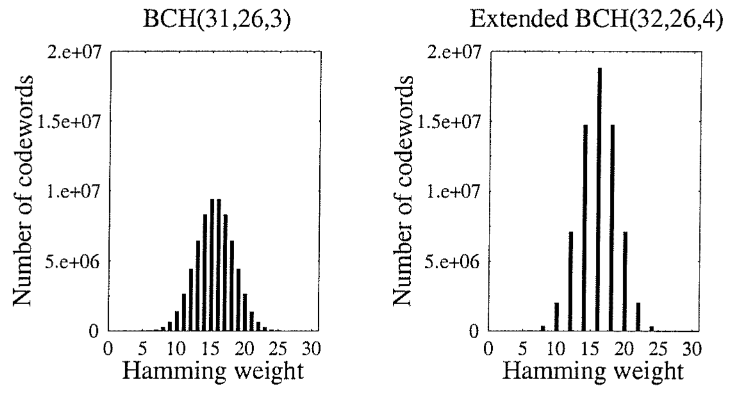


Figure 3.17: Weight distribution of the BCH(31,26,3) and BCH(32,26,4) codes.

3.5.2 Modified MAP Algorithm

3.5.2.1 The Forward and Backward Recursion

In order to incorporate the parity check bit of extended BCH codes in the MAP algorithm, we have to make modifications to Equations 3.26 as well as 3.30 and then to Equation 3.32. Let us define $\alpha_k^e(s)$ as the probability that the current trellis state is $S_k = s$ at time k, where the superscript indicates that the path arriving at this state gave an even number of transmitted bits, which were +1, given that the demodulator's soft output up to this point was $\underline{y}_{j<k}$. Similarly, $\alpha_k^o(s)$ is defined as the probability that the current trellis state is $S_k = s$ at time k, where the superscript indicates that the path leading to this state gave an odd number of transmitted bits, which were +1, given that the demodulator's soft output up to this point was $\underline{y}_{j<k}$. For each state in the trellis, we determine the probabilities $\alpha_k^e(s)$ and $\alpha_k^o(s)$, which are shown in Figure 3.18.

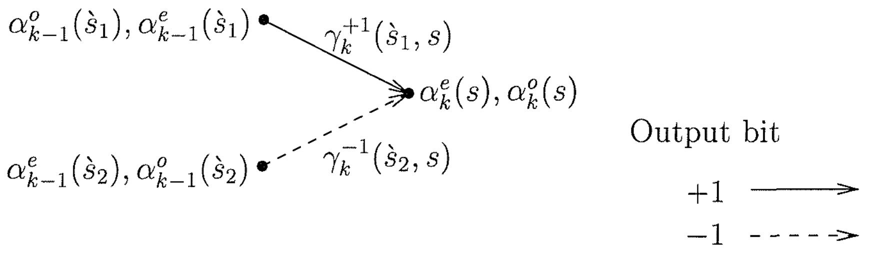

$\alpha_{k-1}^o(\grave{s}_1), \alpha_{k-1}^e(\grave{s}_1)$
$\gamma_k^{+1}(\grave{s}_1, s)$
$\alpha_k^e(s), \alpha_k^o(s)$
$\alpha_{k-1}^e(\grave{s}_2), \alpha_{k-1}^o(\grave{s}_2)$
$\gamma_k^{-1}(\grave{s}_2, s)$
Output bit
+1
−1

Figure 3.18: Forward recursion of $\alpha_k^e(s)$ and $\alpha_k^o(s)$. The corresponding conventional MAP forward recursion was shown in Figure 3.7.

Then, using Equation 3.26 and Figure 3.18, for a binary trellis we can derive the total probability of an even number of transmitted binary +1 bits as the sum of the probabilities of having an even number of +1s at stage $k-1$ and encountering a −1 during the current transition, plus the probability of an odd number of +1s at the previous stage and encounter a +1 during the current transition, yielding:

$$\alpha_k^e(s) = \alpha_{k-1}^o(\grave{s}_1).\gamma_k^{+1}(\grave{s}_1, s) + \alpha_{k-1}^e(\grave{s}_2).\gamma_k^{-1}(\grave{s}_2, s) \ . \tag{3.64}$$

Explicitly, since transition $\gamma_k^{-1}(\grave{s}_2, s)$ from the previous state $\grave{s}_2$ to the present state s results in the transmitted bit being −1, $\alpha_{k-1}^e(\grave{s}_2).\gamma_k^{-1}(\grave{s}_2, s)$ is the probability of the path to state s, which gave an even number of transmitted bits that were +1. Conversely, $\alpha_{k-1}^o(\grave{s}_1)$ is the probability of the path to state $\grave{s}_1$, which gave an odd number of transmitted +1 bits. Since transition $\gamma_k^{+1}(\grave{s}_1, s)$ from the previous state $\grave{s}_1$ to the present state s results in the transmitted bit being +1, $\alpha_{k-1}^o(\grave{s}_1).\gamma_k^{+1}(\grave{s}_1, s)$ is the probability of the path to state s, which also gave an even number of transmitted bits that were +1.

Again, using Equation 3.26 and Figure 3.18, $\alpha_k^o(s)$ is derived for a binary trellis,

$$\alpha_k^o(s) = \alpha_{k-1}^e(\grave{s}_1).\gamma_k^{+1}(\grave{s}_1, s) + \alpha_{k-1}^o(\grave{s}_2).\gamma_k^{-1}(\grave{s}_2, s) \ . \tag{3.65}$$

The same modification is made to the backward recursion and hence we derive from Equation 3.30:

$$\beta_{k-1}^e(\grave{s}) = \beta_k^o(s_1).\gamma_k^{+1}(\grave{s}, s_1) + \beta_k^e(s_2).\gamma_k^{-1}(\grave{s}, s_2) \tag{3.66}$$

and

$$\beta_{k-1}^o(\grave{s}) = \beta_k^e(s_1).\gamma_k^{+1}(\grave{s}, s_1) + \beta_k^o(s_2).\gamma_k^{-1}(\grave{s}, s_2) \ . \tag{3.67}$$

3.5.2.2 Transition Probability

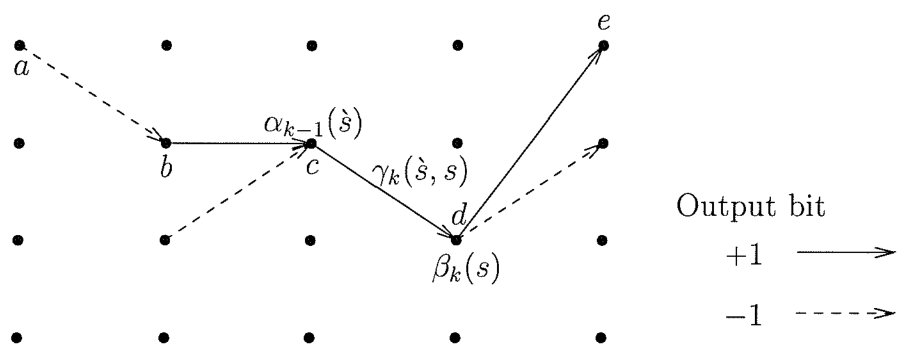


Figure 3.19: Probability of a transition in the trellis.

Figure 3.19 shows a simple trellis diagram, which commences at state a and ends at state e. The probability of the transition from state c to state d is formulated as:

$$P(c \to d) = \alpha_{k-1}(\grave{s}).\gamma_k(\grave{s}, s).\beta_k(s) \ . \tag{3.68}$$

For extended BCH codes, however, the probability of a transition in the trellis no longer depends purely on $\alpha_{k-1}(\grave{s})$, $\gamma_k(\grave{s}, s)$ and $\beta_k(s)$. It also depends on whether the whole path gave an even or odd number of transmitted bits, which were +1. In Figure 3.19, path $a \to b \to c \to d \to e$ gives an odd number of transmitted bits, which were +1. Hence, the probability of the transition $c \to d$ is:

$$P(c \to d) = \alpha_{k-1}(\grave{s}).\gamma_k(\grave{s}, s).\beta_k(s).P(y_n | x_n = +1) \ , \tag{3.69}$$

where x_n is the transmitted parity bit, while y_n is the corresponding demodulator soft output.

Similarly, if the path $a \to b \to c \to d \to e$ had resulted in an even number of transmitted bits, which were +1, the probability of the transition would be

$$P(c \to d) = \alpha_{k-1}(\grave{s}).\gamma_k(\grave{s}, s).\beta_k(s).P(y_n|x_n = -1) \ . \tag{3.70}$$

3.5.2.3 A-Posteriori Information

For each trellis state, we now have $\alpha_k^e(s)$, $\alpha_k^o(s)$, $\beta_k^e(s)$ and $\beta_k^o(s)$ in contrast to $\alpha_k(s)$ and $\beta_k(s)$ in the generic MAP algorithm. In other words, we have separated the probabilities $\alpha_k(s)$ and $\beta_k(s)$ into two groups, respectively, depending on the nature of the trellis paths that reach state s. Additionally, we also have the probability $P(y_n|x_n)$ for the transmitted parity bit. Hence, we can derive the a-posteriori LLR $L(u_k|\underline{y})$ from Equation 3.32, as follows:

$$L(u_k|\underline{y}) = \ln \frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \gamma_k^{+1}(\grave{s}, s).[(\alpha^e.\beta^e + \alpha^o.\beta^o).P(y_n|x_n = +1) + (\alpha^e.\beta^o + \alpha^o.\beta^e).P(y_n|x_n = -1)]}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \gamma_k^{-1}(\grave{s}, s).[(\alpha^e.\beta^o + \alpha^o.\beta^e).P(y_n|x_n = +1) + (\alpha^e.\beta^e + \alpha^o.\beta^o).P(y_n|x_n = -1)]} \ , \tag{3.71}$$

where α^e, α^o, β^e and β^o are shorthands for $\alpha_{k-1}^e(\grave{s})$, $\alpha_{k-1}^o(\grave{s})$, $\beta_k^e(s)$ and $\beta_k^o(s)$, respectively.

By definition, in the numerator of Equation 3.71, the transition from the previous state $S_{k-1} = \grave{s}$ to the present state $S_k = s$ always results in a transmitted bit of +1. The path $\alpha_{k-1}^e(\grave{s}).\gamma_k^{+1}(\grave{s}, s).\beta_k^e(s)$ results in an odd number of transmitted bits that are +1, since both the paths to state $S_{k-1} = \grave{s}$, and from state $S_k = s$, gave an even number of transmitted bits which were +1, and the transition from state $S_{k-1} = \grave{s}$ to state $S_k = s$ results in a transmitted bit of +1. Similarly, the path $\alpha_{k-1}^o(\grave{s}).\gamma_k^{+1}(\grave{s}, s).\beta_k^o(s)$ also results in an odd number of transmitted bits that are +1. Therefore, the probabilities of both transitions are $\alpha^e.\gamma_k^{+1}(\grave{s}, s).\beta^e.P(y_n|x_n = +1)$ and $\alpha^o.\gamma_k^{+1}(\grave{s}, s).\beta^o.P(y_n|x_n = +1)$, respectively. Conversely, paths $\alpha_{k-1}^e(\grave{s}).\gamma_k^{+1}(\grave{s}, s).\beta_k^o(s)$ and $\alpha_{k-1}^o(\grave{s}).\gamma_k^{+1}(\grave{s}, s).\beta_k^e(s)$ would result in an even number of +1 transmitted bits. Hence, the probabilities of both transitions are $\alpha^e.\gamma_k^{+1}(\grave{s}, s).\beta^o.P(y_n|x_n = -1)$ and $\alpha^o.\gamma_k^{+1}(\grave{s}, s).\beta^e.P(y_n|x_n = -1)$, respectively. The same argument is applied to the denominator in Equation 3.71 in order to derive the probability of each transition.

3.5.3 Max-Log-MAP and Log-MAP Algorithm for Extended BCH codes

As mentioned in Section 3.3.4.3, conceptually the Max-Log-MAP and Log-MAP algorithms are the same, except that the approximation made in Equation 3.37 in the context of the Max-Log-MAP algorithm can be made exact by using the Jacobian logarithm [95]. Hence, in this section, we will only describe the Max-Log-MAP algorithm.

The proposed modified MAP algorithm calculates the a-posteriori LLRs $L(u_k|\underline{y})$ using Equation 3.71. Hence, it requires the following values:

1. The $\alpha^e_{k-1}(\grave{s})$ and $\alpha^o_{k-1}(\grave{s})$ values, which are calculated in a forward recursive manner using Equations 3.64 and 3.65, respectively;
2. The $\beta^e_k(s)$ and $\beta^o_k(s)$ values, which are calculated in a backward recursive manner using Equations 3.66 and 3.67, respectively;
3. The transition probabilities $\gamma_k(\grave{s}, s)$;
4. The probability that the number of transmitted bits, which are +1 is even, $P(y_n|x_n = +1)$, or odd, $P(y_n|x_n = -1)$.

The Max-Log-MAP algorithm simplifies the preceding equations by transferring them into the logarithmic domain and then using the approximation [50, 51]:

$$\ln\left(\sum_i e^{x_i}\right) \approx \max_i(x_i) \ , \tag{3.72}$$

where $\max_i(x_i)$ means the maximum value of x_i.

Let us define $A^e_k(s)$, $A^o_k(s)$, $B^e_k(s)$, $B^o_k(s)$ and $\Gamma_k(\grave{s}, s)$ as follows:

$$A^e_k(s) \triangleq \ln[\alpha^e_k(s)] \ , \tag{3.73}$$

$$A^o_k(s) \triangleq \ln[\alpha^o_k(s)] \ , \tag{3.74}$$

$$B^e_k(s) \triangleq \ln[\beta^e_k(s)] \ , \tag{3.75}$$

$$B^o_k(s) \triangleq \ln[\beta^o_k(s)] \ , \tag{3.76}$$

$$\Gamma^{x_k}_k(\grave{s}, s) \triangleq \ln[\gamma^{x_k}_k(\grave{s}, s)] \ , \tag{3.77}$$

and

$$\epsilon^{x_n} \triangleq \ln[P(y_n|x_n)] \ . \tag{3.78}$$

Upon substituting Equation 3.64 into Equation 3.73 and using Equation 3.72, we arrive at:

$$\begin{aligned} A_k^e(s) &= \ln\left[\alpha_{k-1}^e(\grave{s}).\gamma_k^{-1}(\grave{s},s) + \alpha_{k-1}^o(\grave{s}).\gamma_k^{+1}(\grave{s},s)\right] \\ &= \ln\left[e^{\left\{A_{k-1}^e(\grave{s})+\Gamma_k^{-1}(\grave{s},s)\right\}} + e^{\left\{A_{k-1}^o(\grave{s})+\Gamma_k^{+1}(\grave{s},s)\right\}}\right] \\ &\approx \max\left[\left\{A_{k-1}^e(\grave{s}) + \Gamma_k^{-1}(\grave{s},s)\right\}, \left\{A_{k-1}^o(\grave{s}) + \Gamma_k^{+1}(\grave{s},s)\right\}\right] \ . \end{aligned} \tag{3.79}$$

Following the same approach and substituting Equation 3.65 into Equation 3.74, we get:

$$\begin{aligned} A_k^o(s) &= \ln\left[\alpha_{k-1}^o(\grave{s}).\gamma_k^{-1}(\grave{s},s) + \alpha_{k-1}^e(\grave{s}).\gamma_k^{+1}(\grave{s},s)\right] \\ &= \ln\left[e^{\left\{A_{k-1}^o(\grave{s})+\Gamma_k^{-1}(\grave{s},s)\right\}} + e^{\left\{A_{k-1}^e(\grave{s})+\Gamma_k^{+1}(\grave{s},s)\right\}}\right] \\ &\approx \max\left[\left\{A_{k-1}^o(\grave{s}) + \Gamma_k^{-1}(\grave{s},s)\right\}, \left\{A_{k-1}^e(\grave{s}) + \Gamma_k^{+1}(\grave{s},s)\right\}\right] \ , \end{aligned} \tag{3.80}$$

Similarly to Equation 3.79 and 3.80, we can rewrite Equation 3.75 as:

$$\begin{aligned} B_k^e(s) &= \ln\left[\beta_k^e(s).\gamma_k^{-1}(\grave{s},s) + \beta_k^o(s).\gamma_k^{+1}(\grave{s},s)\right] \\ &= \ln\left[e^{\left\{B_k^e(s)+\Gamma_k^{-1}(\grave{s},s)\right\}} + e^{\left\{B_k^o(s)+\Gamma_k^{+1}(\grave{s},s)\right\}}\right] \\ &\approx \max\left[\left\{B_k^e(s) + \Gamma_k^{-1}(\grave{s},s)\right\}, \left\{B_k^o(s) + \Gamma_k^{+1}(\grave{s},s)\right\}\right] \ , \end{aligned} \tag{3.81}$$

and Equation 3.76 as:

$$\begin{aligned} B_k^o(s) &= \ln\left[\beta_k^o(s).\gamma_k^{-1}(\grave{s},s) + \beta_k^e(s).\gamma_k^{+1}(\grave{s},s)\right] \\ &= \ln\left[e^{\left\{B_k^o(s)+\Gamma_k^{-1}(\grave{s},s)\right\}} + e^{\left\{B_k^e(s)+\Gamma_k^{+1}(\grave{s},s)\right\}}\right] \\ &\approx \max\left[\left\{B_k^o(s) + \Gamma_k^{-1}(\grave{s},s)\right\}, \left\{B_k^e(s) + \Gamma_k^{+1}(\grave{s},s)\right\}\right] \ . \end{aligned} \tag{3.82}$$

We quote and simplify the following equation from Equation 3.43:

$$\Gamma_k^{x_k}(\grave{s},s) = \frac{u_k}{2}L(u_k) + \frac{L_c}{2}y_k x_k \ , \tag{3.83}$$

where $L(u_k)$ is the intrinsic information of the data bit u_k and L_c is the channel reliability value.

If we assume that the bit $x_n = \pm 1$ has been sent over a Gaussian or fading channel using BPSK modulation, then we can write for the conditional probability of the matched filter output y_n that:

$$P(y_n|x_n = \pm 1) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{E_b}{2\sigma^2}(y_n \mp a)^2} \ , \tag{3.84}$$

where E_b is the transmitted energy per bit, σ^2 is the noise variance and a is the fading amplitude. For non-fading AWGN channels, the value of a is unity, i.e. $a = 1$.

Using Equation 3.84, we can rewrite Equation 3.78 as:

$$\begin{aligned}
\epsilon^{x_n=\pm 1} &= \ln\left[\frac{1}{\sigma\sqrt{2\pi}}\right] - \frac{E_b}{2\sigma^2}(y_n \mp a)^2 \\
&= C - \frac{E_b}{2\sigma^2}(y_n^2 \mp 2ay_n + a^2) \\
&= C - \frac{E_b}{2\sigma^2}(y_n^2 + a^2) \pm \frac{E_b}{2\sigma^2}2ay_n \\
&= C - C' \pm \frac{L_c}{2}y_n \ , \qquad (3.85)
\end{aligned}$$

where $C = \ln\left[\frac{1}{\sigma\sqrt{2\pi}}\right]$, $C' = \frac{E_b}{2\sigma^2}(y_n^2 + a^2)$ and $L_c = \frac{2a}{\sigma^2}$. Both C and C' are independent of the transmitted parity check bit x_n and hence can be considered as constants, which are omitted, when comparing the various paths through the trellis.

Finally, using Equations 3.72 to 3.85, we can rewrite the a-posteriori LLRs $L(u_k|\underline{y})$ of the MAP algorithm in Equation 3.71 for the Max-Log-MAP algorithm, yielding:

$$L(u_k|y) \approx \max_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} [P(\grave{s},s)] - \max_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} [P(\grave{s},s)] \ , \qquad (3.86)$$

where

$$\begin{aligned}
P(\grave{s},s) = \max[&\{\Gamma_k^{u_k}(\grave{s},s) + A_{k-1}^e(\grave{s}) + B_k^e(s) + \epsilon^{x_n=u_k}\}, \\
&\{\Gamma_k^{u_k}(\grave{s},s) + A_{k-1}^o(\grave{s}) + B_k^o(s) + \epsilon^{x_n=u_k}\}, \\
&\{\Gamma_k^{u_k}(\grave{s},s) + A_{k-1}^e(\grave{s}) + B_k^o(s) + \epsilon^{x_n=-u_k}\}, \\
&\{\Gamma_k^{u_k}(\grave{s},s) + A_{k-1}^o(\grave{s}) + B_k^e(s) + \epsilon^{x_n=-u_k}\}] \ . \qquad (3.87)
\end{aligned}$$

3.6 Simulation Results

In this section, we are going to present our simulation results for turbo codes using simple BPSK over AWGN channels. We will show that some of the parameters are interlinked, which jointly affect the performance of turbo codes, depending on:

- The number of decoding iterations used.
- The decoding algorithm used.
- The component BCH code employed.

- The frame length of the input data.
- The design of the interleaver used.

All our investigations were conducted using the turbo encoder structure shown in Figure 3.1. The data bits of the upper encoder are transmitted through the channel, whereas the interleaved data bits of the lower encoder are punctured. However, the parity bits from both encoders are multiplexed alternately and none of them are punctured. The multiplexing and puncturing patterns are the same for all simulations, unless stated otherwise.

3.6.1 Number of Iterations Used

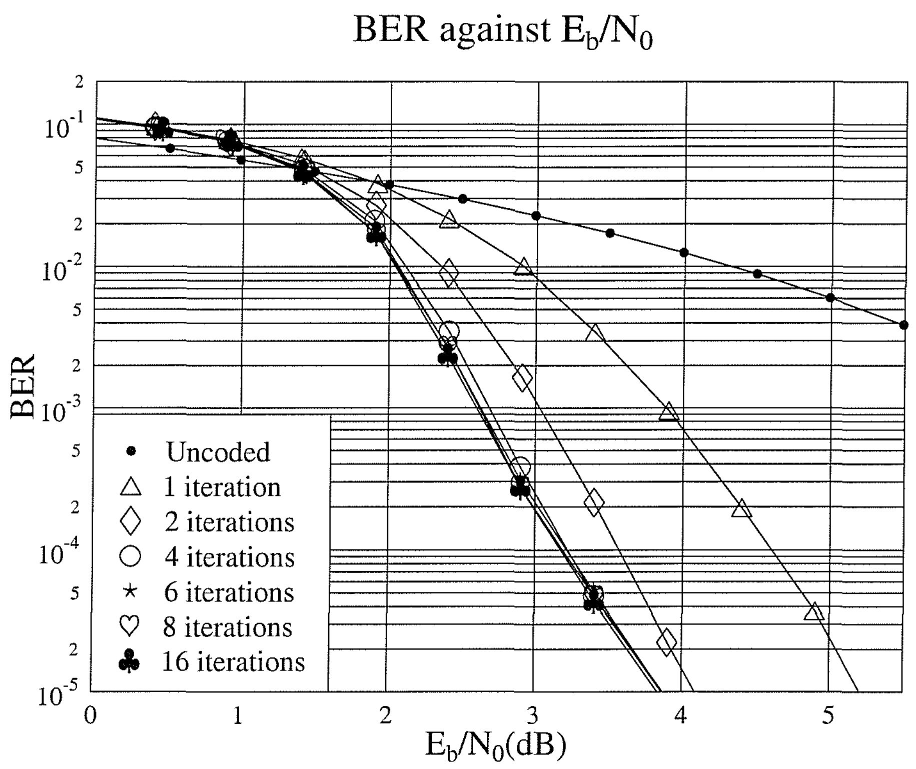


Figure 3.20: Performance comparison of different number of iterations using the rate $R = 0.72$ turbo BCH(31,26,3) code, in conjunction with a 26×26 bit block interleaver and the Log-MAP Algorithm over AWGN channels.

Figure 3.20 shows the performance of the rate $R = \frac{26}{36} = 0.72$ turbo BCH(31,26,3) code, using a 26×26 bit block interleaver and the Log-MAP algorithm, versus the number of decoding iterations used. Again, since all the parity bits of both the upper and lower encoder of Figure 3.1 are transmitted through the channel, this results in a coding rate of

$R = 0.72$. As the number of iterations used by the decoder increases, the decoder performs significantly better. However, after 4 iterations there is only little improvement upon using further iterations.

In [116], Goalic and Pyndiah used the extended BCH(32,26,4) product code for real-time block turbo decoding. The authors invoked the soft input and soft decision output of algebraic decoding and achieved a coding gain of about 6 dB at a BER of 10^{-5} using 4 iterations. By contrast, the coding gain of our turbo BCH(31,26,3) code used in Figure 3.20 is about 5.5 dB at a BER of 10^{-5}. However, in [81,116] the authors use 'factors optimised by simulation' for weighting the soft information. Furthermore, the associated code rate of $\frac{26}{31} \times \frac{26}{31} = 0.66$ used in their scheme was also slightly lower. Moreover, this decoding method is only applicable to product code.

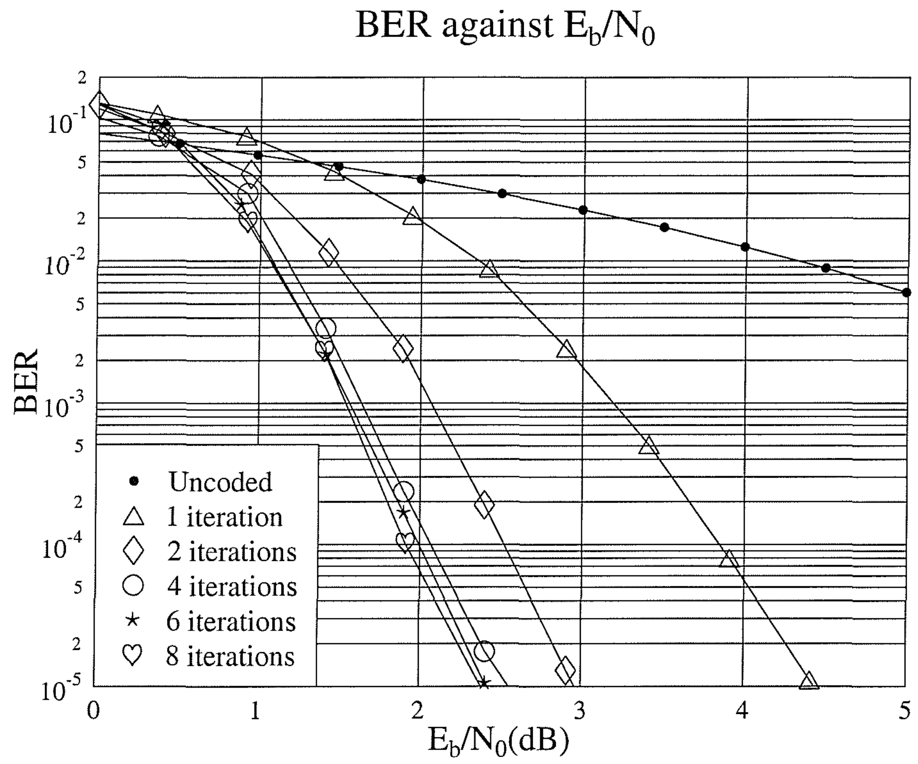


Figure 3.21: Performance comparison of different number of iterations using the rate $R = 0.51$ turbo BCH(31,21,5) code in conjunction with a 21×21 bit block interleaver and the Log-MAP algorithm over AWGN channels.

Figure 3.21 shows the performance of the rate $R = \frac{21}{41} = 0.51$ turbo BCH(31,21,5) code, using a 21×21 bit block interleaver and the Log-MAP algorithm, versus the number of decoding iterations. It can be seen that the coding gain between each iteration is higher than that of the BCH(31,26,3) component code. Specifically, the gain between the first and second iteration is about 1.5 dB, whereas for the previous turbo BCH(31,26,3) code we had an improvement of about 1 dB only. When using two iterations, the coding gain is about 6.75 dB at a BER of 10^{-5}.

If we compare the complexity of both component codes, the rate $R = 0.72$ turbo BCH(31,26,3) code is significantly less complex, than the rate $R = 0.51$ turbo BCH(31,21,3) scheme. Since the trellis-decoding complexity of BCH codes is directly proportional to 2^{n-k}, the turbo BCH(31,21,5) code is about $\frac{2^{31-21}}{2^{31-26}} = 32$ times more complex, than the turbo BCH(31,26,3) scheme. Furthermore, the associated memory requirement is also 32 times higher. As an example, four iterations of the turbo BCH(31,26,3) code yield a BER of about 2×10^{-1} at an E_b/N_0 of 3 dB, which is about an order of magnitude better, than the BER of the turbo BCH(31,21,5) scheme at one iteration, while its complexity is about eight times lower. At two iterations however, the approximately 16 times more complex turbo BCH(31,21,5) code outperforms the BCH(31,26,3) turbo arrangement.

3.6.2 The Decoding Algorithm

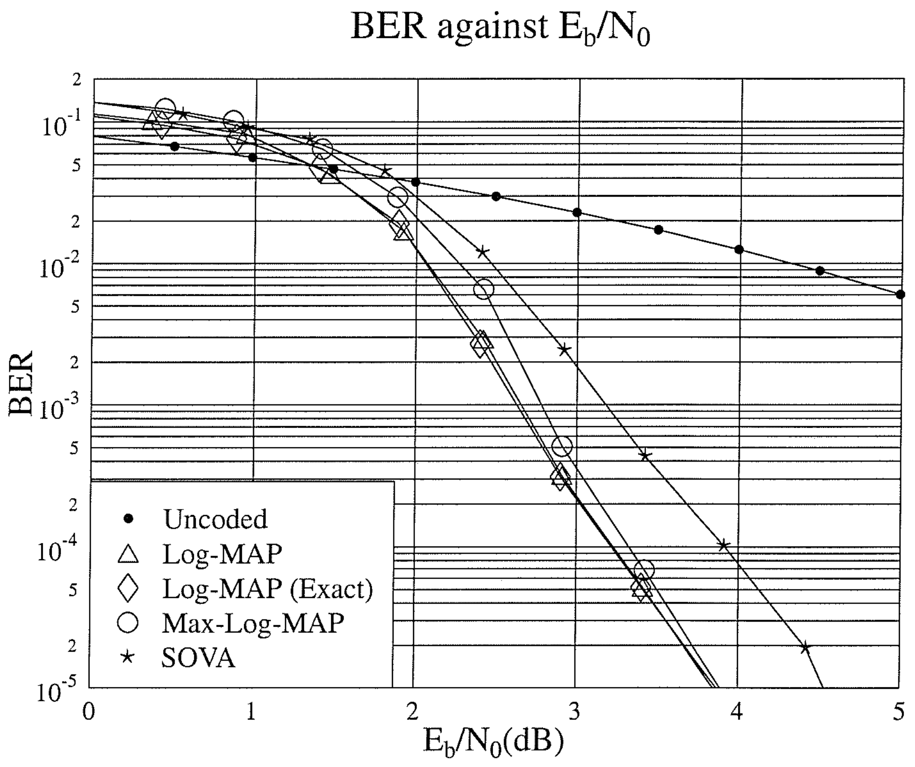


Figure 3.22: Performance comparison between different decoding algorithms for six iterations using the rate $R = 0.72$ turbo BCH(31,26,3) code in conjunction with a 26×26 bit block interleaver over AWGN channels.

Figure 3.22 shows a comparison between the different turbo BCH(31,26,3) component decoders described in Section 3.3, using a 26×26 bit block interleaver. In the figure, the performance of the MAP algorithm [11] is not shown, because it is identical to that of the Log-MAP algorithm [52]. This is justified, since the Log-MAP algorithm is a specific version of the MAP algorithm transformed into the logarithmic domain in order to simplify its operation and to reduce the numerical problems associated with the MAP algorithm, as described in Section 3.3.4.

The "Log-MAP (Exact)" curve refers to a decoder, which calculates the correction term $f_c(\delta)$ in Equation 3.45 of Section 3.3.4.3 exactly, i.e. using

$$f_c(\delta) = \ln(1 + e^{-\delta}) \ , \tag{3.88}$$

rather than a look-up table, as described in [52]. The Log-MAP curve refers to a decoder, which does use a look-up table with eight values of $f_c(\delta)$ stored, and hence introduces an approximation to the calculation of the LLRs. It can be seen in Figure 3.22 that the "Log-MAP (Exact)" and "Log-MAP" algorithms give identical performances. In [52], Robertson found that the look-up table based values of the $f_c(\delta)$ correction term in Equation 3.45 introduces no degradation to the performance of the decoder.

It can be seen from Figure 3.22 that the Max-Log-MAP and the SOVA algorithm both suffer some degradation in performance compared to the Log-MAP algorithm. At a BER of 10^{-4}, this degradation is about 0.1 dB for the Max-Log-MAP algorithm, and about 0.7 dB for the SOVA algorithm. However, at a BER of 10^{-5}, the Max-Log-MAP algorithm suffers only insignificant degradation. The complexity of these algorithms was compared in [52].

Again, Figure 3.23 shows our performance comparison between different decoding algorithms. However, the component code used is the BCH(31,21,5) scheme employing a 21×21 bit block interleaver. The figure shows that the degradation incurred by using the Max-Log-MAP algorithm is approximately 0.2 dB at a BER of 10^{-4}, which is the about the same as that for the BCH(31,26,3) component code in the previous case. However, the SOVA algorithm shows a significant E_b/N_0 performance degradation of 1.4 dB at a BER of 10^{-4} in comparison to the Log-MAP algorithm.

3.6.3 The Effect of Estimating the Channel Reliability Value L_c

In Section 3.3 we have highlighted how the component decoders use the soft inputs, the soft channel outputs $L_c y_k$ and the intrinsic information $L(u_k)$, in order to provide the a-posteriori information $L(u_k|\underline{y})$ as its soft outputs. In this section, we investigate how the imperfect estimation of the channel reliability value L_c affects the performance of the

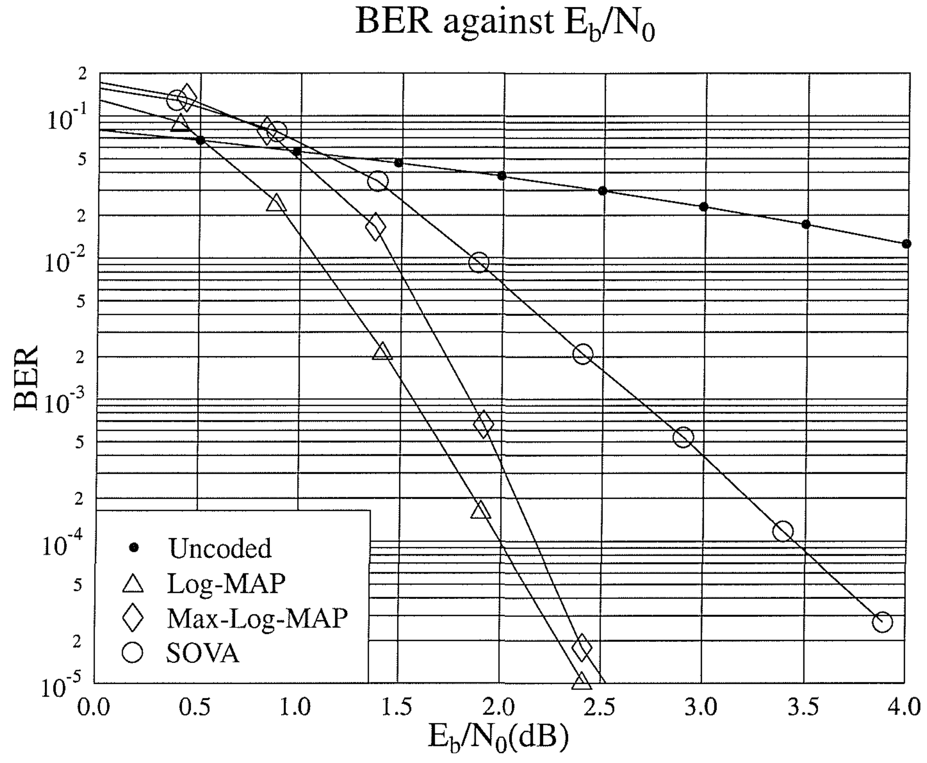


Figure 3.23: Performance comparison between different decoding algorithms for six iterations using the rate $R = 0.51$ turbo BCH(31,21,5) code in conjunction with a 21×21 bit block interleaver over AWGN channels.

algorithms.

Figure 3.24 shows the effect of using imperfect channel reliability values L_c for three different decoding algorithms - namely for the Log-MAP, Max-Log-MAP algorithms and SOVA. For each decoding algorithm, the solid line shows the performance of the algorithm when the channel reliability value L_c is calculated exactly using the known channel SNR. The dashed curves in Figure 3.24 shows how the three algorithms perform, when the channel reliability L_c is not known. For these curves, the value $L_c = 1$, which according to Equation 3.8 corresponds to an E_b/N_0 value of -3 dB, was used at all channel SNRs. It can be seen from Figure 3.24 that the Max-Log-MAP algorithm and the SOVA perform equally well, whether or not the correct value of L_c is known. However, the performance of the Log-MAP algorithm is drastically affected by the incorrect L_c value used. We can see that the performance of the Log-MAP algorithm is even worse than that of the uncoded case, if $L_c = 1$.

The reason for these effects can be understood by considering the different operations described in Section 3.3. For the SOVA the soft channel output $L_c y_k$ is used recursively in

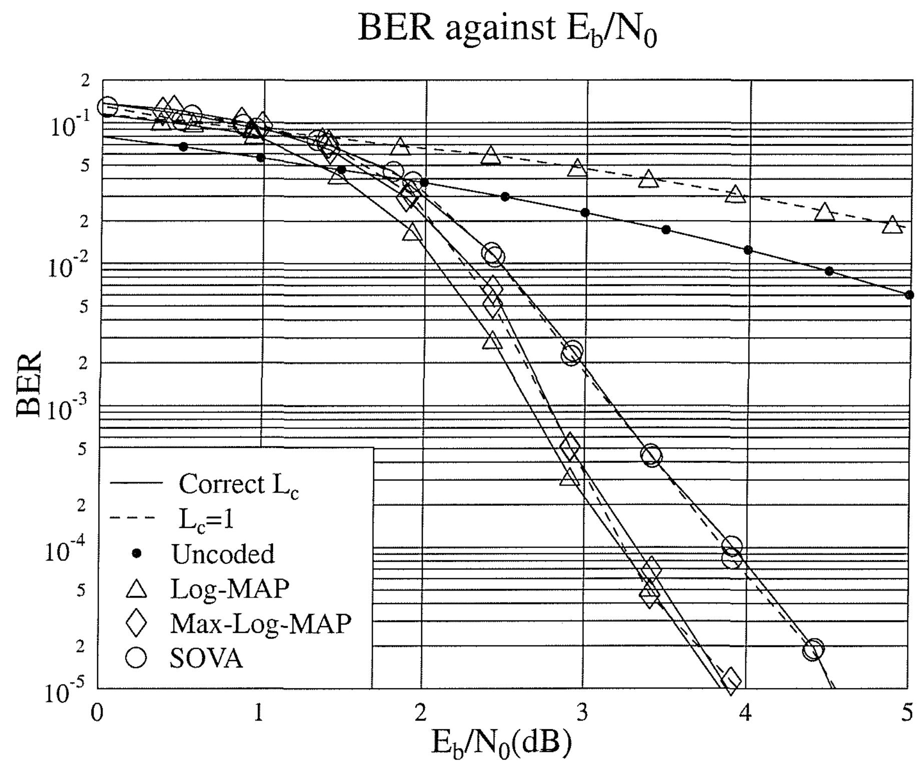


Figure 3.24: Effect of using incorrect channel reliability values L_c on the turbo BCH(31,26,3) code employing six iterations, the log-MAP algorithm and a 26 × 26 bit block interleaver over AWGN channels.

order to calculate the path metric using Equation 3.51. In Equation 3.51, we can see that L_c is used to scale the demodulator's soft output y_k and this has an effect of scaling all the path metrics by the same factor. Since the soft output LLRs generated by the algorithm are given by the path metric differences between the ML path and the discarded paths, the soft output LLRs are also scaled by the same factor. The same phenomenon was also observed for the Max-Log-MAP algorithm.

Let us now consider the Log-MAP algorithm. This is identical to the Max-Log-MAP algorithm, except for a correction factor of $f_c(\delta) = \ln(1 + e^{-\delta})$ used in the calculation of the forward and backward recursion functions in Equations 3.25 and 3.29, respectively. The function $f_c(\delta)$ is a non linear function, which decreases asymptotically towards zero, as δ increases. Since δ depends directly on L_c value, the performance of the Log-MAP algorithm degrades if imperfect estimation of L_c is given.

3.6.4 The Effect of Puncturing

In their original turbo codec, Berrou *et al.* [13] applied puncturing, in order to obtain a half rate code. An impressive performance was attained, even though half of the parity bits from both convolutional encoders were punctured. However, if we use block codes as the component codes, the parity bits are often unpunctured, since the associated performance loss would be excessive.

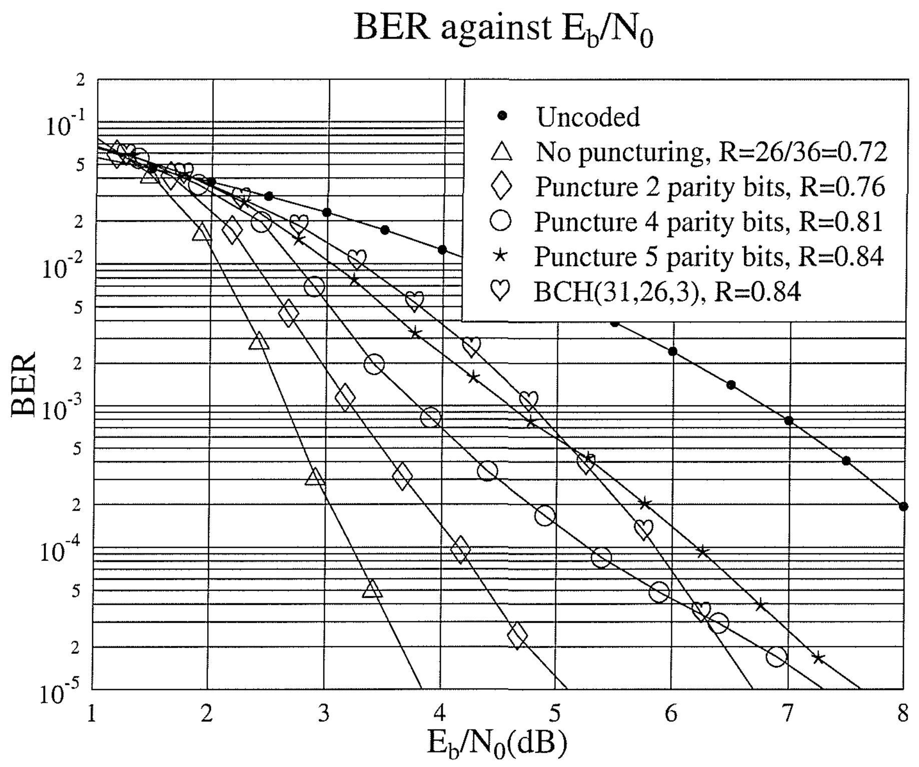


Figure 3.25: Performance comparison between different puncturing patterns of the rate $R = 0.72$ turbo BCH(31,26,3) code employing six iterations, the log-MAP algorithm and a 26×26 bit block interleaver over AWGN channels.

Figure 3.25 shows our performance comparison between different puncturing patterns applied to the turbo BCH(31,26,3) code using a 26×26 bit block interleaver. We have seen in Section 3.2 that the turbo encoder consists of two BCH encoders. In Figure 3.25 two BCH(31,26,3) encoders are used for the turbo encoder and each of them generates five parity bits. Therefore, for every block of 26 data bits, the turbo encoder will produce ten parity bits. In our first example in Figure 3.25 no puncturing is applied, which results in ten transmitted parity bits and a code rate of $R = 0.72$. In our next example we show the performance of the code, when two parity bits are randomly selected and punctured for every block of 26 data bits. In the following examples, we punctured more parity bits, in order to attain a higher coding rate.

We can see from Figure 3.25 that the performance of the turbo BCH(31,26,3) code decreases, as the coding rates R increases. In the figure we also show the performance of the conventional BCH(31,26,3) using the soft decision Viterbi algorithm. It can be seen that, if 5 parity bits are punctured in the turbo code, which results in the same coding rate as that of the conventional BCH(31,26,3) code, the performance of the turbo BCH(31,26,3) code is about 1 dB worse, than that of the conventional BCH(31,26,3) code at a BER of 10^{-5}.

Generally, puncturing does not assist in improving the coding rate without sacrificing the performance of the turbo BCH code. Moreover, there are certain puncturing patterns, which outperform others at a certain coding rate [119]. However, it is impossible to find the optimum puncturing patterns for long codes by exhaustive search. One possible solution is to study the distance profile of the code, since the performance of the code depends on the distance properties and we can investigate the effects of puncturing on the distance profile.

3.6.5 The Effect of the Interleaver Length of the Turbo Code

Many contributions [12, 13, 69], have shown impressive performances for large interleaver lengths. Although interleaver length is affordable in data transmission system (non-real time systems) since a delay of say 10^4 bits is generally acceptable. However, for many other real time applications, such as for example interactive speech and video transmission, the system often can only tolerate a delay of approximately 100 bits.

We show in Figure 3.26, how the interleaver length affects the performance of the rate $R = 0.72$ turbo BCH(31,26,3) code. The interleaver length of 26 and 104 bits, which uses a random and a 13×8 bit block interleaver, respectively, is suitable for the above-mentioned real-time systems. It can seen from the figure that the performance of the turbo BCH(31,26,3) code using a random interleaver having an interleaver depth of 26 bits, is slightly better, than the conventional BCH(31,26,3) code using the soft decision Viterbi algorithm at a BER of 10^{-3}. Notice also that the rate R of the turbo code is lower, than that of the conventional BCH code. In terms of decoding complexity, the turbo code is more demanding, than the conventional VA. As it was shown in Section 3.3.4, both component decoders of the turbo code have to calculate the values of α, β and γ. This results in about three times higher complexity, than decoding the same code using a standard Viterbi decoder. The curves shown in Figure 3.26 were generated using two component decoders and six iterations. The overall complexity of the corresponding turbo decoder is approximately $2 \times 6 \times 3 = 36$ times higher than that of a Viterbi decoder. Therefore, the conventional BCH encoding and decoding method constitute a better choice, if the affordable delay of

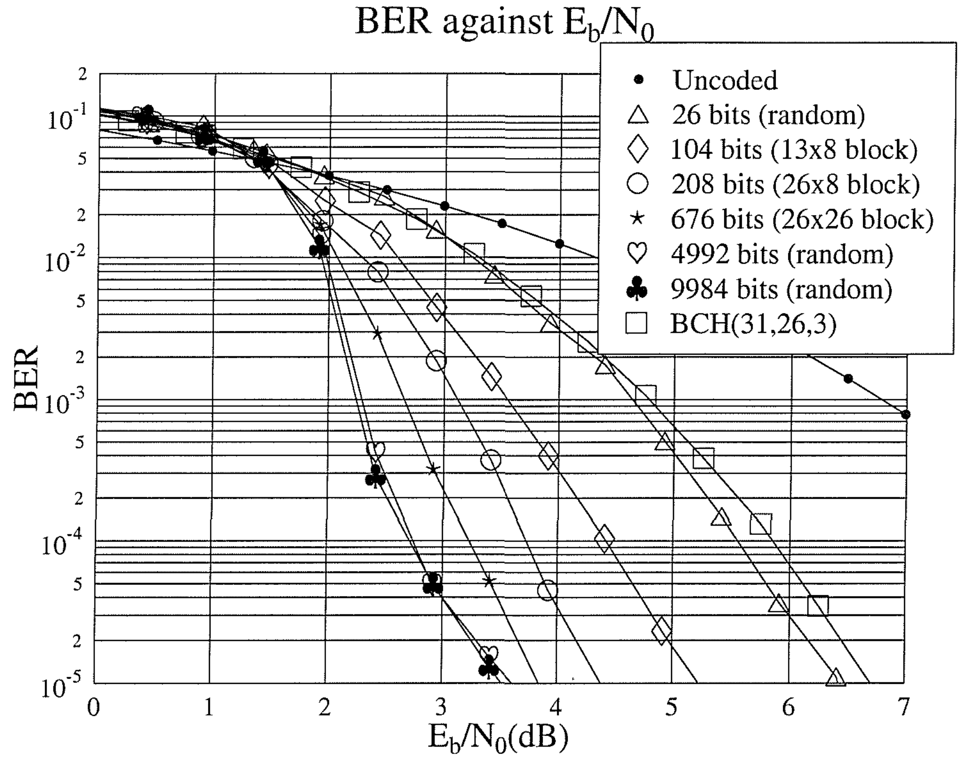


Figure 3.26: Performance comparison between different interleaver length for the rate $R = 0.72$ turbo BCH(31,26,3) code using six iterations, the log-MAP algorithm and the conventional rate $R = 0.83$ BCH(31,26,3) code using the soft decision Viterbi algorithm over AWGN channels.

the system is low.

The performance of turbo codes increases, as the interleaver length increases. However, as shown in Figure 3.26, the incremental coding gain becomes smaller, as the interleaver length increases. It reaches its near-optimum performance when the interleaver length exceeds 5,000 bits. Such a high interleaver length is only suitable for non-real time systems.

3.6.6 The Effect of the Interleaver Design

It is well known that the interleaver design has a vital effect on the performance of turbo codes. The interleaver design together with the component codes used and the puncturing pattern play an important role in determining the minimum free distance d_{min} of turbo codes and in turn predetermines the performance of the code.

In the context of turbo BCH codes we face various problems in designing the interleaver. Let the BCH(n_1, k_1, d_{min1}) and BCH(n_2, k_2, d_{min2}) schemes be the two component codes of a turbo code. Since BCH codes are block codes, which encode k data bits each time, the length of the interleaver has to be multiple of k_1 and k_2. Therefore, the design of the interleaver is not as flexible as that of turbo convolutional codes, which can have more flexibility in terms of the interleaver length. In this section we consider how the interleaver design affects the performance of the code, while keeping the interleaver length, the component codes and the puncturing patterns the same.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

(a) Block interleaver

$$\xrightarrow[\text{randomly}]{\text{Interleave column}}$$

0	9	6	15
8	13	14	11
4	5	2	3
12	1	10	7

(b) Random-in-column block interleaver

Figure 3.27: Bit positions in the (a) block interleaver and (b) Random-in-column block interleaver.

First, we define a new type of interleaver, which is referred to as the random-in-column block interleaver. The bit positions of the block interleaver and random-in-column block interleaver are shown in Figure 3.27. In the block interleaver the data bits are entered into the matrix on a row-by-row basis. After the matrix is full, the data bits are read out on a column-by-column basis. In order to design the random-in-column block interleaver, we rearrange randomly the data bits within each column of the block interleaver. In Figure 3.27, we can see that for each column the bit positions of the block and random-in-column block interleaver are the same, except for the arrangement of the bit positions.

The benefit of using the random-in-column block interleaver is shown in Figure 3.28 in terms of its improved BER performance. The figure shows a comparison of different interleaver designs for an interleaver length of 676 bits using the rate $R = 0.72$ turbo BCH(31,26,3) code. It can be seen that at a BER of 10^{-5} the performance of the random-in-column block interleaver is about 0.2 dB better, than that of the block interleaver. Figure 3.28 also shows that the random interleaver has the worst performance of all. However, this is not always the case, when we use other interleaver lengths. In the following two figures, we show the effect of the interleaver design for both lower and higher interleaver lengths compared to the above mentioned 676-bit interleaver length.

Figure 3.29 shows our performance comparison between different interleaver designs at an interleaver length of 104 bits using the rate $R = 0.72$ turbo BCH(31,26,3) code. Such short interleavers can be used for interactive speech transmission, for example. We can see

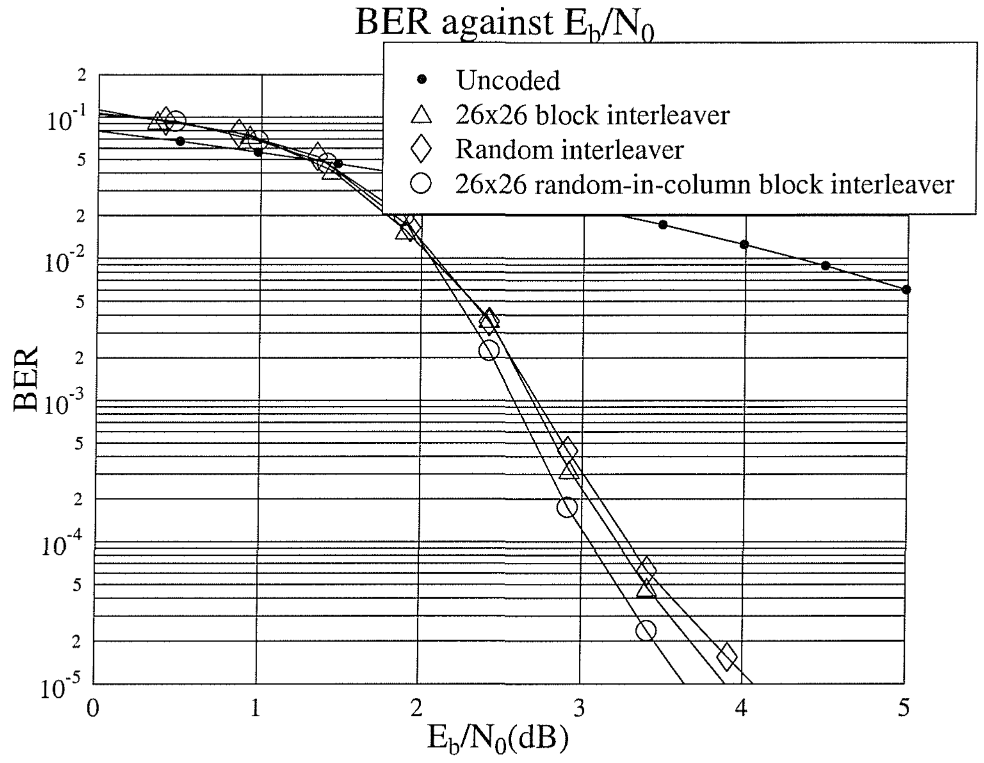


Figure 3.28: Performance comparison between different interleaver designs at an interleaver length of $26 \times 26 = 676$ bits using the rate $R = 0.72$ turbo BCH(31,26,3) code, six iterations and the log-MAP decoding algorithm over AWGN channels.

from Figure 3.29 at a BER of 10^{-3}, which is often targeted by speech systems, that all the interleavers have essentially the same performance. The same trend is observed at low BERs for the 13×8 bit block interleaver, for the 104-bit random interleaver and for the 13×8 bit random-in-column block interleaver. However, at a BER of 10^{-5}, the performance of the 26×4 bit block interleaver is about 1 dB worse, than that of the others.

Previously, we have seen how the interleaver design affects the performance of the code using small (104 bits) and medium (676 bits) interleaver sizes. Let us now characterise the performances of different interleaver designs at an interleaver length of 9984 bits, which is suitable for data transmission. Figure 3.30 shows our performance comparison between different interleaver designs using the turbo BCH(31,26,3) code. At a BER of 10^{-5}, the performance of all the interleavers is about the same. However, for higher BERs the random interleaver outperforms the others.

From the simulation results given above and from our simulation results using other component codes, we can draw some conclusions on the design of interleavers. Specifically, for a turbo BCH(n, k, d_{min}) code, the $k \times k$ random-in-column interleaver is preferred if the

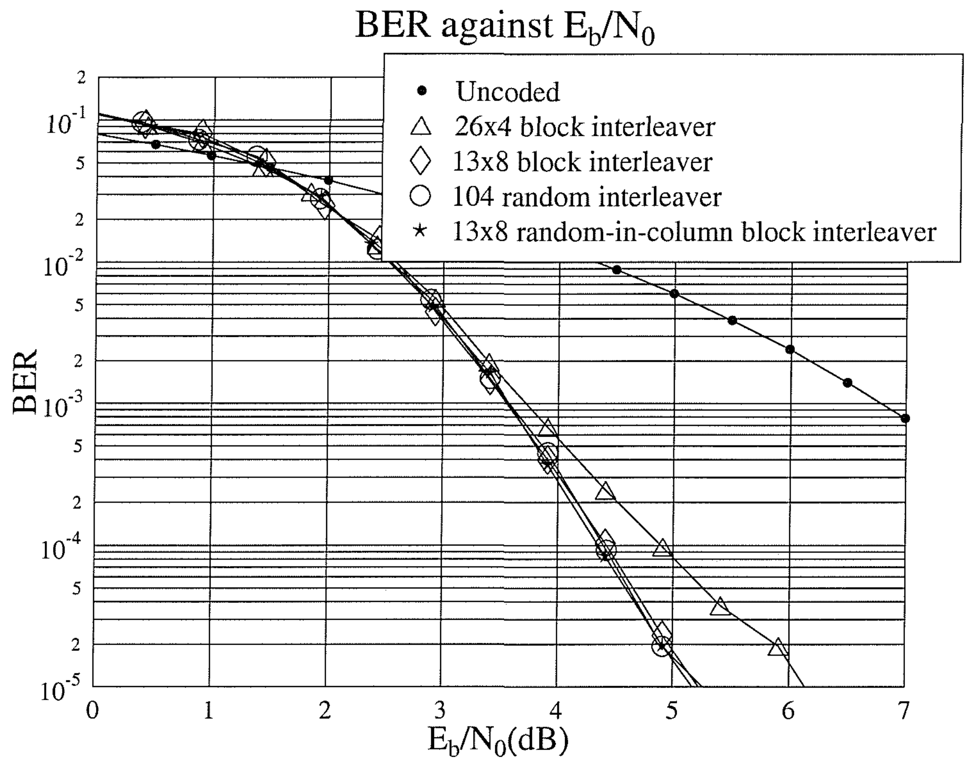


Figure 3.29: Performance comparison between different interleaver designs at an interleaver length of 104 bits using the rate $R = 0.72$ turbo BCH(31,26,3) code, six iterations and the log-MAP decoding algorithm over AWGN channels.

interleaver length is $k \times k$. For a small interleaver length, below $k \times k$, the simple square block interleaver performs better. This implies that the number of columns and rows should not differ significantly. In a data transmission system we can have a high interleaver depth and it was found that the random interleaver is the best choice.

3.6.7 The Component Codes

Figure 3.31 shows the performance of different turbo BCH(n, k, d_{min}) codes using six iterations of the Log-MAP algorithm. The interleaver length of each code is about 10,000 bits and the coding rate R varies from 0.3 to 0.83.

In [61], Hagenauer, Offer and Papke showed simulation results for various Hamming codes, which have the same minimum free distance as BCH codes. The MAP decoding algorithm was employed in the authors' simulations. After comparing the results of [61] and the simulation results of Figure 3.31, we conclude that the BER performance of turbo BCH and Hamming codes is similar.

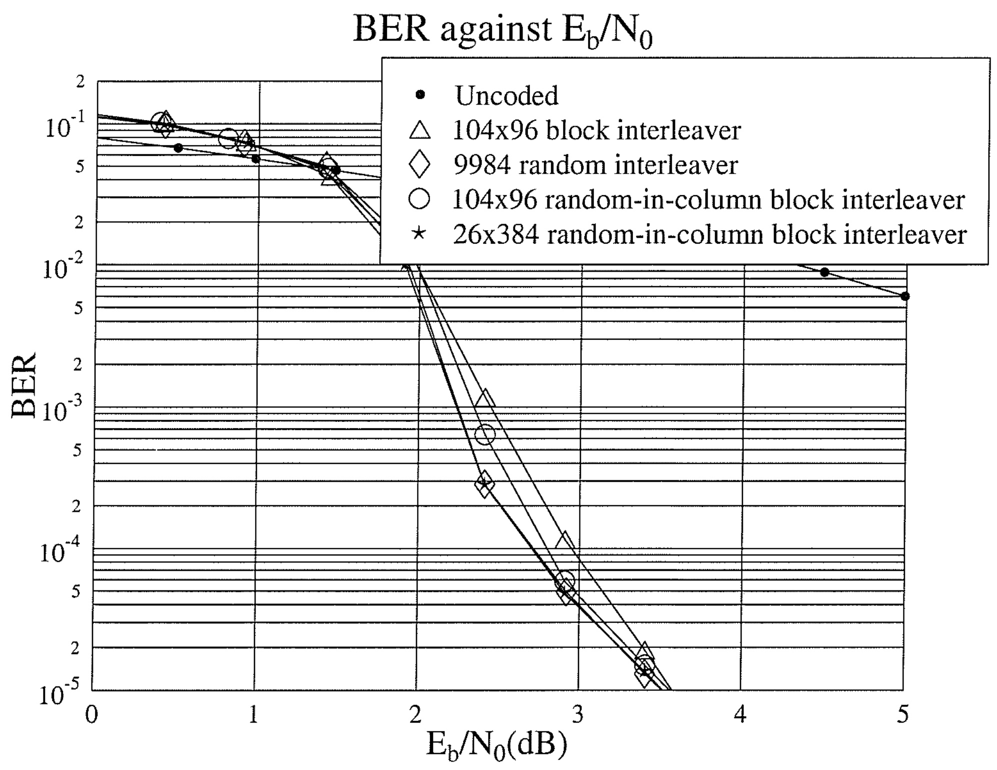


Figure 3.30: Performance comparison between different interleaver designs at an interleaver length of 9984 bits using the rate $R = 0.72$ turbo BCH(31,26,3) code, six iterations and the log-MAP decoding algorithm over AWGN channels.

In Figure 3.31, we have also included the Shannon capacity limit [1] of each BCH code, except for the turbo BCH(7,4,3) code. For the turbo BCH(15,11,3) code, which has a coding rate of $R = 0.58$, the associated performance is about 4 dB away in terms of E_b/N_0 from the Shannon capacity limit at a BER of 10^{-5}. As we increase the coding rate to 0.83 by using the BCH(63,57,3) scheme as component codes, the performance curve is within about 1.1 dB from the Shannon limit, again, when viewed at a BER of 10^{-5}. In Table 3.7 we tabulated the E_b/N_0 distances with respect to the Shannon capacity limit for the various turbo BCH codes studied.

A simple conclusion can be drawn from Table 3.7 at this point. As we increase the codeword length n, while keeping the minimum free distance d_{min} constant, the coding rate R increases. As the coding rate R increases, the discrepancy between the associated performance curve and the Shannon limit is getting smaller. For example, we fix the minimum free distance to $d_{min} = 3$. Table 3.7 shows that the distance from the capacity limit is 4.0 dB when $n = 15$, which decreases to 1.1 dB, as n increases to 63. However, in [63] Nickl, Hagenauer and Burkert have performed several simulations over AWGN channels

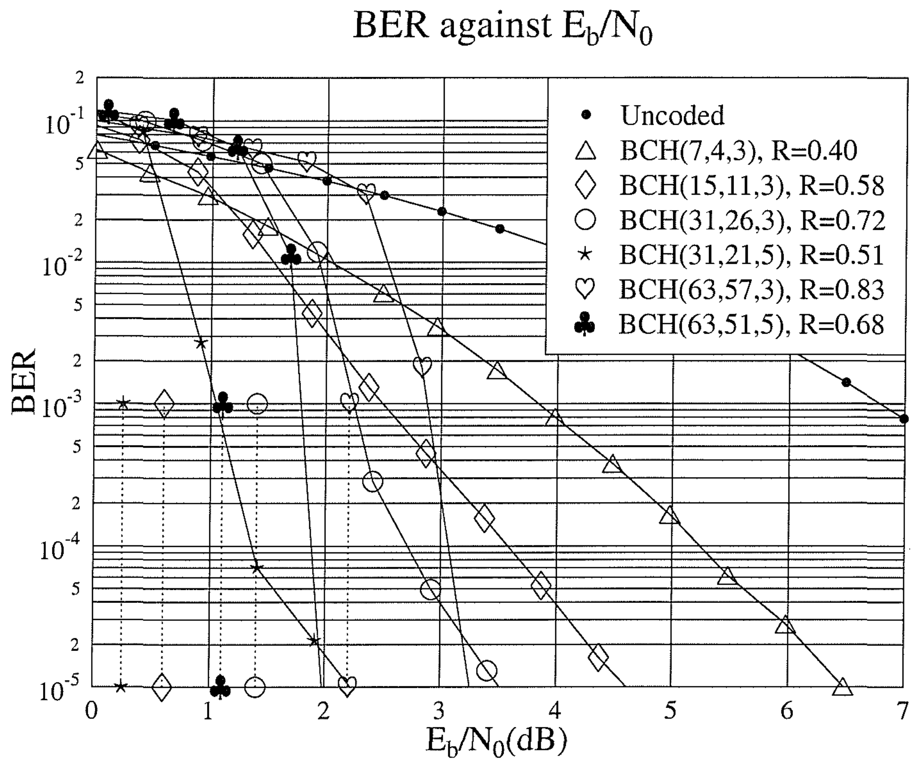


Figure 3.31: Performance comparison of different turbo BCH(n, k, d_{min}) codes employing six iterations, the log-MAP decoding algorithm and an interleaver length of about 10,000 bits over AWGN channels. The associated Shannon limits for different coding rates are indicated by the dotted vertical lines.

Component code	Rate	Distance to Shannon limit
BCH(15,11,3)	0.58	4.0 dB
BCH(31,26,3)	0.72	2.1 dB
BCH(31,21,5)	0.51	2.0 dB
BCH(63,57,3)	0.83	1.1 dB
BCH(63,51,5)	0.68	0.8 dB

Table 3.7: Distance to the Shannon capacity limit for turbo BCH codes using various code rates R.

using different $(n = 2^N - 1, k = 2^N - 1 - N, 3)$ Hamming codes as component codes and a $k \times k$ bit block interleaver. They have shown that the performance improvement for very long Hamming codes $(N > 10)$ is marginal. Furthermore, they have also shown that the performance of the (1023,1013) Hamming code is only 0.27 dB away from the Shannon limit, which is the closest approximation to Shannon's limit reported so far for block codes.

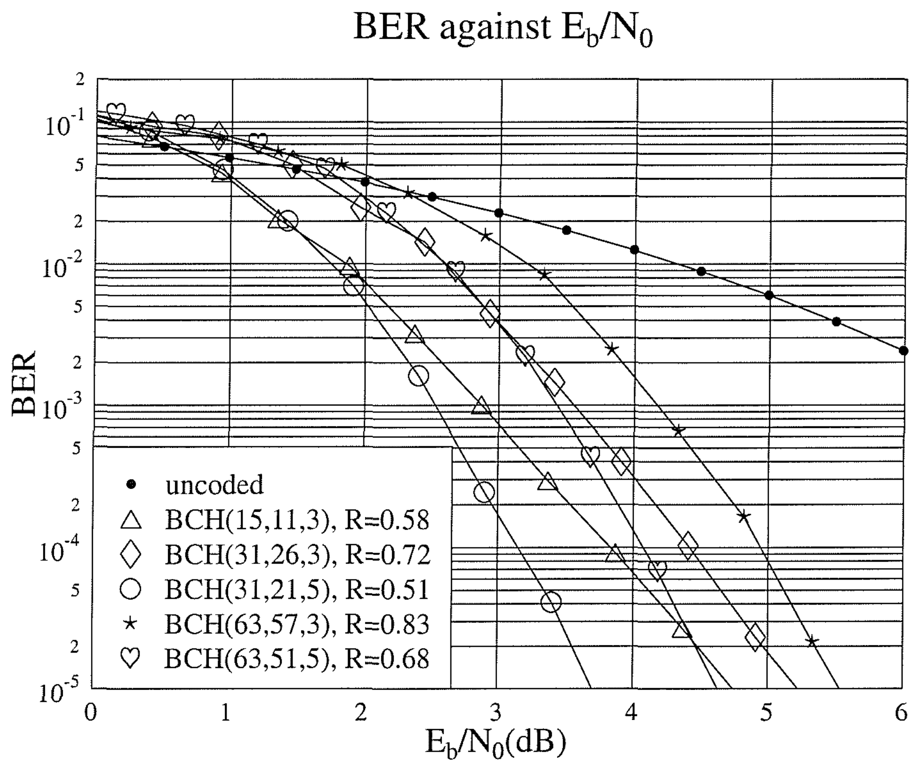


Figure 3.32: Performance comparison of different BCH(n, k, d_{min}) turbo code employing six iterations, log-MAP algorithm and interleaver length of $\approx$ 100 over the AWGN channels.

Previously, we have studied the performance of different component codes using a high interleaver size of 10,000 bits, which is suitable for non-real-time transmission systems. Let us now consider real-time transmission system, which requires a short delay. Hence the interleaver size was reduced to approximately 100 bits. Figure 3.32 shows our performance comparison for different turbo BCH(n, k, d_{min}) codes at an interleaver length of approximately 100 bits. In Figure 3.31, as we increase n while maintaining a minimum distance of $d_{min} = 3$, we observe that there is an increase in coding gain at a BER of 10^{-3}. However, if we limit the interleaver size to approximately 100 bits, the situation is reversed. As shown in Figure 3.32, the turbo code using the BCH(15,11,3) scheme as its component code achieves the highest coding gain at a BER of 10^{-3} compared to the BCH(31,26,3) and the BCH(63,57,3) codes. As n increases, while keeping d_{min} constant, k increases as well.

Therefore, the number of BCH codewords that fit into an interleaver size of about 100 bits becomes smaller. Hence the number of BCH codewords per turbo coded block reduces, and hence the correlation between the upper and lower codewords also increases. As the data bits become more dependent on each other, the turbo decoder is less likely to correct the errors that occurred in the turbo block.

In Figure 3.31, we observe that the BCH(63,51,5) component codes outperform the BCH(31,26,3) component codes in the context of turbo coding, even though their coding rates are nearly the same, namely 0.68 and 0.72, respectively. This is because the BCH(63,51,5) code has a higher minimum free distance d_{min} and n is twice higher. However, for a small interleaver size of about 100, the BCH(31,26,3) code performs as well as BCH(63,51,5) scheme at a BER of 10^{-3} and 0.5 dB worse at a BER of 10^{-5}. This simply implies that at a BER of 10^{-3} we can achieve a high coding gain by using the BCH(31,26,3) code, which has a higher coding rate and a lower complexity. Generally, if the interleaver size is small, it is better to choose BCH(n, k, d_{min}) turbo component codes, which have small k.

3.6.8 BCH$(31, k, d_{min})$ Family Members

Both Figure 2.19 and 2.20 in Section 2.3.6 demonstrate that for a certain family BCH code defined by a constant codeword length of n, a specific BCH code achieve the highest coding gain, which is typically has a code rate in the range of $0.5 - 0.7$.

Here, we present similar results for different turbo BCH$(31, k, d_{min})$ code family members at an interleaver length of about $10,000$ bits in Figure 3.33. It is seen in the figure that the turbo code using the BCH(31,21,5) code as its component code, has the highest coding gain. The coding rate R of the turbo code is about $\frac{1}{2}$.

3.6.9 Mixed Component Codes

In the previous section, we have presented our simulation results using the same component code for both the upper and lower encoder shown in Figure 3.1. Let us now study the effect of different component codes on the performance of the turbo code.

Figure 3.34 shows our performance comparison between turbo codes using both different and identical component codes, at an interleaver length of about $10,000$ bits. The turbo coding scheme denoted by the diamond shapes in Figure 3.34 consisted of the BCH(63,67,3) scheme as the upper component code and the BCH(31,21,5) arrangement as the lower component code. This codec results in a coding rate of $R = 0.63$, which is between the

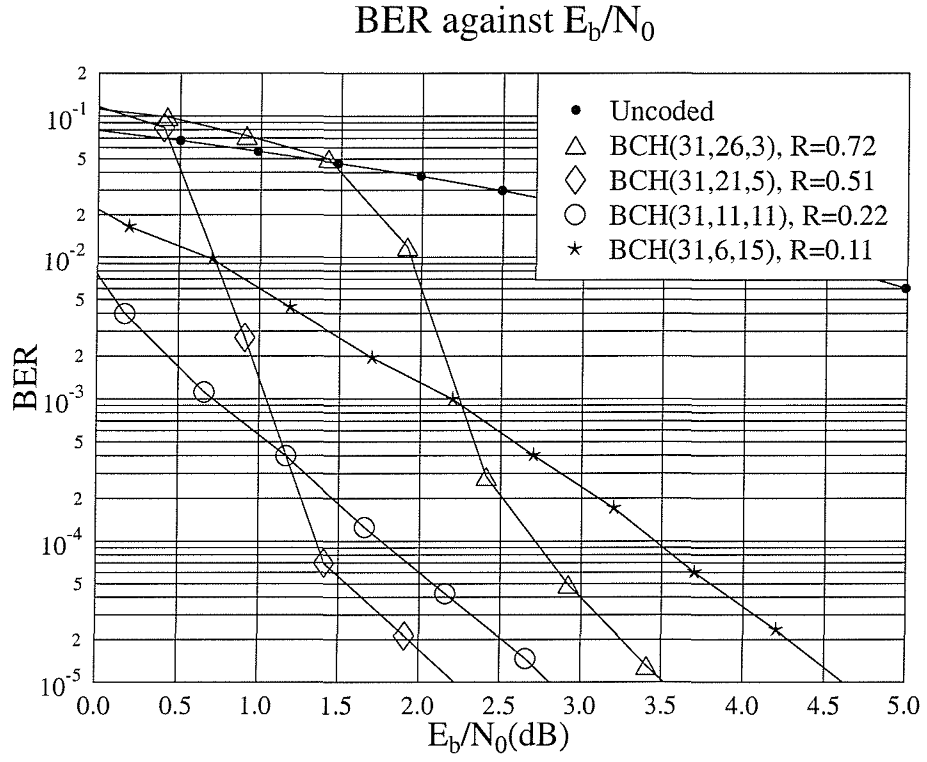


Figure 3.33: Performance comparison of turbo codes using BCH$(31, k, d_{min})$ family members as component codes, employing six iterations, the log-MAP decoding algorithm and an interleaver length of about 10,000 bits over AWGN channels.

coding rate of the first and the third turbo code shown in Figure 3.34.

In Figure 3.34, we have demonstrated that the performance of the rate $R = 0.63$ turbo code using mixed component codes is about 0.25 dB worse than that of the rate $R = 0.51$ turbo BCH(31,21,5) code. Hence, at the cost of slight degradation of the coding gain, we have increased the coding rate from 0.51 to 0.63. Furthermore, the complexity of the turbo code using mixed component codes is lower than that of the turbo BCH(31,21,5) code, but higher than that of the BCH(63,57,3) scheme.

3.6.10 Extended BCH codes

In Section 3.5, we proposed the modified MAP and Log-MAP algorithms in order to incorporate the parity check bit of the extended BCH codes into calculating the LLR of the decoded bits. Figure 3.35 shows the performance of both the BCH(31,26,3) and that of the turbo BCH(32,26,4) code. The Log-MAP algorithm was used and the number of iterations was six for both cases. The interleaver was a random interleaver using an interleaving depth

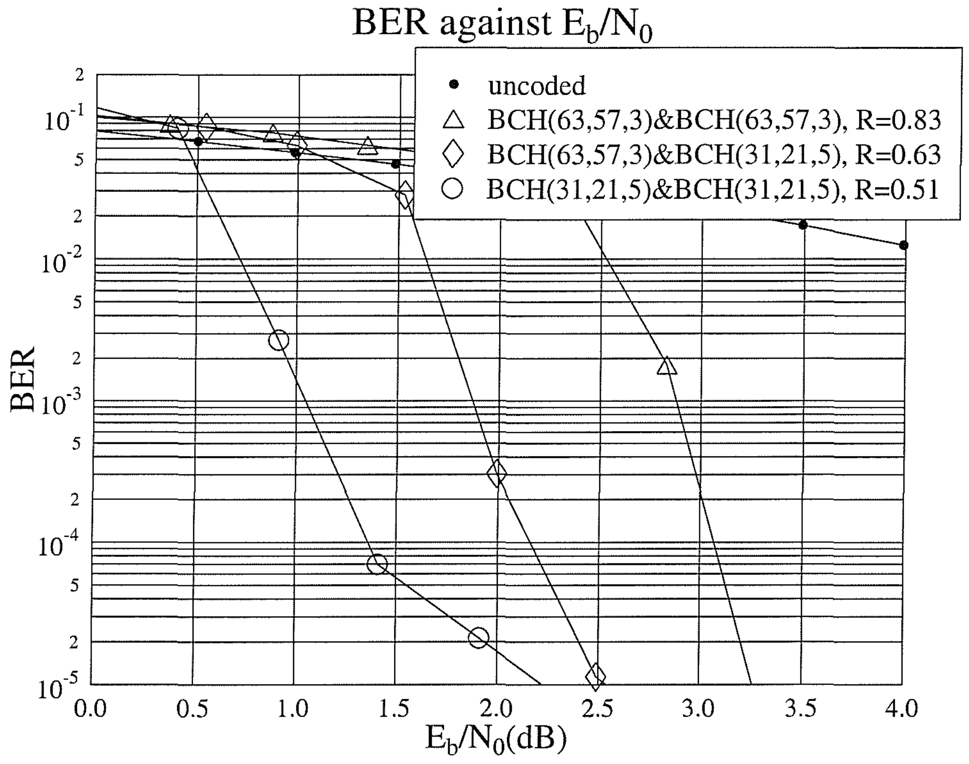


Figure 3.34: Performance comparison between turbo codes using both different and identical BCH component codes, employing six iterations, the log-MAP decoding algorithm and an interleaver length of about 10,000 bit over AWGN channels.

of 9984 bits.

As explained in Section 3.5, a parity check bit was appended to the BCH(31,26,3) component code, extends it to the BCH(32,26,4) code. Explicitly, the minimum free distance d_{min} was increased from three to four. In a turbo code the extra parity bit causes only a small degradation of the coding rate from 0.72 to 0.68. However, as shown in Figure 3.35, the performance of the turbo BCH(32,26,4) code is about 0.7 dB better than that of the turbo BCH(31,26,3) code.

Further simulation results were obtained for the BCH(32,21,6) and the BCH(31,21,5) coded based turbo codes. Unlike for the BCH(31,26,3) and BCH(32,26,4) codes, the performance of both codes remained about the same. This is probably because if we increase the minimum free distance d_{min} from three to four, this results in a 33 percent increase of d_{min}. By contrast, if we increase $d_{min} = 5$ to 6, the increase is only about 20 percent.

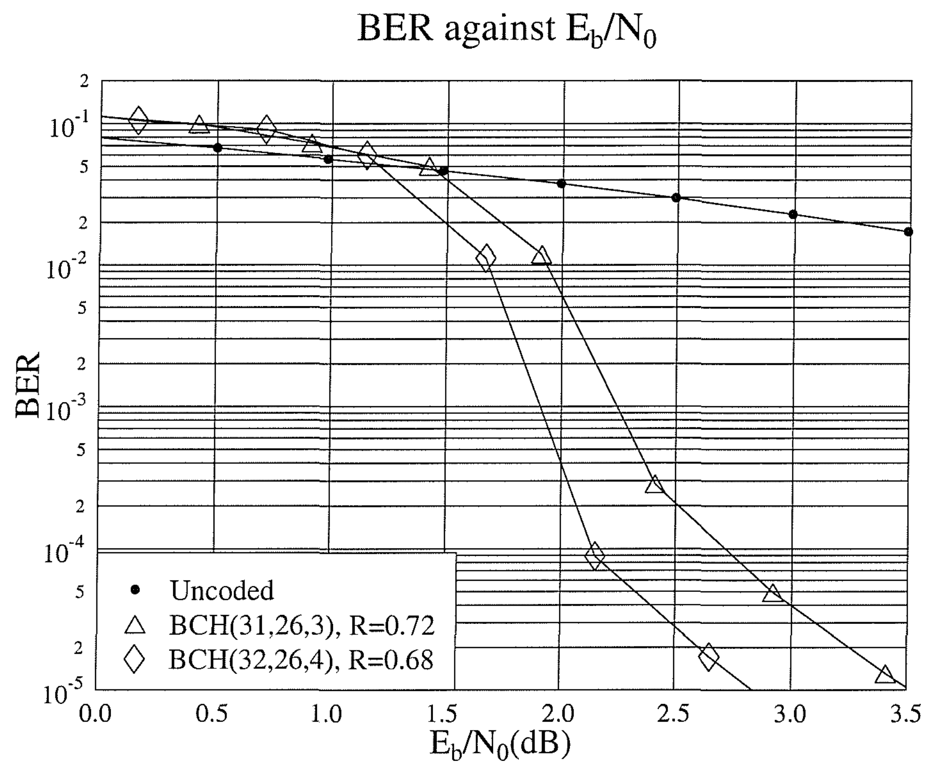


Figure 3.35: Performance comparison between turbo codes using the BCH(31,26,3) or the BCH(32,26,4) code as the component codes, employing six iterations, the log-MAP decoding algorithm and a random interleaver with a depth of 9984 bits over AWGN channels.

3.6.11 BCH Product codes

In Section 3.2 we highlighted the differences between turbo codes and product codes. As shown in Figure 3.2, the structure of turbo codes is the same as that of product codes, except that in turbo codes the redundancy part arising from checking the parity of the parity part of both codes is neglected. Furthermore, the turbo code has a smaller minimum free distance than the product code, as argued in Section 3.2.

In order to exploit the parity of the parity bits of the product code, the Log-MAP algorithm was modified so that it provided the soft output LLRs for the parity bits as well. Upon receiving the soft channel outputs of the product code, the decoder decodes the columns of the product code. The decoder provides the soft output LLRs of both the data and the parity bits. Then, the decoder uses the soft channel outputs plus the intrinsic information of the data and parity bits, in order to decode the rows of the product code. The above process continues for a certain number of iterations. The same decoding techniques have also been proposed for product codes when using soft in soft out algebraic decoding [81,115–117].

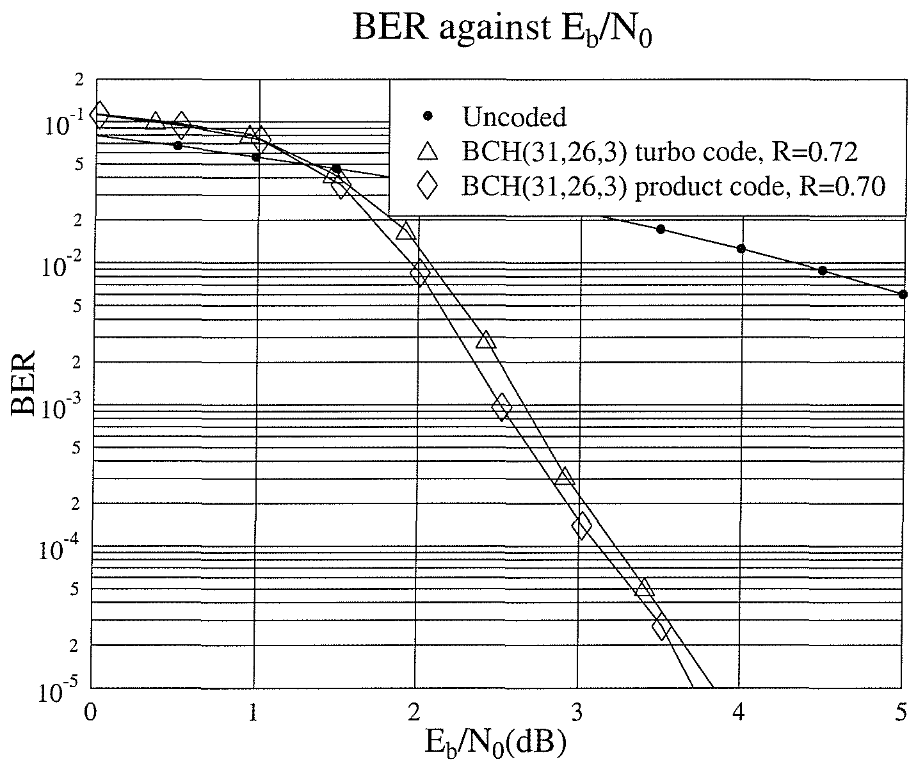


Figure 3.36: Performance comparison between the BCH(31,26,3) product code and the turbo BCH(31,26,3) code employing six iterations, the log-MAP decoding algorithm and a 26×26 bits block interleaver over AWGN channels.

In Figure 3.36, we portray our performance comparison between the BCH(31,26,3) product code and the turbo BCH(31,26,3) code using a 26×26 bit block interleaver. It can be seen that there is no significant performance improvement when using the BCH(31,26,3) product code.

3.7 Summary and Conclusion

In this chapter our discussions revolved around turbo BCH codes, rather than the conventional turbo convolutional codes. The structure of the turbo encoder, which consists of two component codes, was discussed in Section 3.2. The difference between product codes and turbo codes was also highlighted in the section. Then the more complex structure of turbo decoders was presented in Section 3.3. The philosophy of iterative decoding was also detailed. This was followed by the definition of the log likelihood ratio and the derivation of the soft channel outputs in Sections 3.3.1 and 3.3.2, respectively.

The MAP algorithm is the core of generating the soft information exchanged between the decoders in the turbo decoder. The detailed derivation of the algorithm was given in Section 3.3.3. Briefly, the algorithm can be divided into three parts. It first calculates the transition probabilities of each trellis transition. Using the calculated transition probabilities, the algorithm performs forward and backward recursion, which are the remaining two parts of the algorithm. A summary of the MAP algorithm was given in Section 3.3.3.4. Due to the high implementational complexity of the MAP algorithm, the less complex Max-Log-MAP and Log-MAP algorithms were derived in Sections 3.3.4.2 and 3.3.4.3, respectively. The even less complex SOVA algorithm, which is a derivative of the Viterbi algorithm, was presented in Section 3.3.5 and a decoding example was detailed in Section 3.3.5.1. In Section 3.4, a decoding example was given in the context of the turbo BCH(7,4,3) code. The SOVA decoding algorithm was employed in the example and we showed, how iterative decoding improves the reliability of the soft outputs and hence corrects the channel errors that could not be removed by non-iterative decoding.

We modified the MAP algorithm in order to incorporate an additional parity check bit in extended BCH codes. The algorithm was modified such that it kept track of the probability of the paths, which gave odd or even number of transmitted bits that were $+1$. This was vital in calculating the soft outputs, since the probability of the survivor path, which gives odd or even number of $+1$ transmitted bits, is known, as it was detailed in Section 3.5. The reduced-complexity modified Max-Log-MAP and modified Log-MAP algorithms were also derived and presented in Section 3.5.3 for the class of extended BCH codes.

Finally, we presented our simulation results using BPSK over AWGN channels in Section 3.6. We first investigated the effect of iterative decoding on the performance of the turbo BCH codes. It was found that the performance of the BCH turbo code did not improve significantly after four iterations. Various decoding algorithms were compared in Section 3.6.2 and it was found that the Log-MAP and the Max-Log-MAP algorithms gave a similar performance. The SOVA decoding algorithm gives the worst performance, but it is the least complex algorithm. It was shown in Section 3.6.3 that imperfect estimation of the channel reliability value had no effect on the performance of the Max-Log-MAP and SOVA algorithms. However, the Log-MAP algorithm performs badly, if the estimation of the channel reliability value is imperfect. The effect of puncturing was investigated in Section 3.6.4 and as expected, it was found that puncturing degrades the performance of turbo BCH codes. Hence it was concluded that no puncturing should be applied to the parity bits of turbo BCH codes. In Section 3.6.5, we showed that, again, as anticipated the performance of turbo BCH codes improves as the turbo interleaver length increases and a near-optimum performance was achieved, when the interleaver length exceeded about 5,000 bits. We proposed a novel interleaver design in Section 3.6.6. This interleaver design was referred to as the random-in-column block interleaver and it was shown to outperform other conventional block and random interleavers, if the interleaver had a dimension of $k \times k$, i.e. it was rectangular. Different turbo BCH codes which employ different BCH component codes were investigated in Section 3.6.7. It was found that turbo BCH codes perform impressively at near-unity code rates. The turbo BCH(63,51,5) code was found to perform within about 0.8 dB from the Shannon limit. In Section 3.6.9, two different BCH component codes were employed in the turbo encoder. The performance of this turbo BCH code was found to be between that of turbo BCH codes, which employ two identical BCH codes. The modified MAP algorithm was employed for decoding extended BCH codes and it was shown in Section 3.6.10 that the extended turbo BCH(32,26,4) code outperformed the turbo BCH(31,26,3) code by approximately 0.5 dB at a BER of 10^{-5}. Finally, the performance of BCH product codes was found to be similar to that of turbo BCH codes in Section 3.6.11.

Chapter 4

Residue Number System

4.1 Introduction

We have things of which we do not know the number,
If we count them by threes, the remainder is 2,
If we count them by fives, the remainder is 3,
If we count them by sevens, the remainder is 2,
How many things are there?
The answer is 23.

The verse [40] is quoted from a third-century book, *Suan-Ching*, by Sun Tzu. Motivated by the publication of the book [40], the study of the residue number system begins. In Sun Tzu's historic work, he presents a formula for manipulating the remainders of an integer after division by 3, 5 and 7. Today, his contribution is referred to as the Chinese Remainder Theorem (CRT) [40, 41], which is one of the common rules of converting remainders, or residues into integers.

The residue number system (RNS) [40–42] represents a further departure from two established and well known number systems, the decimal and binary number systems. In many ways, the RNS is different from the decimal and binary number systems. Due to their basic differences, the RNS exhibits an unusual set of characteristics, which intrigued many researchers. The most interesting one is that the RNS provides the ability to add, subtract or multiply in parallel, regardless of the size of the numbers involved, without recourse to intermediate carry digits or internal processing delays [40, 41]. In addition, there is a lack of ordered significance amongst the residues. Hence, without adding more logic circuits, it is possible in principle to produce a parallel computer.

However, a drawback of the RNS is the awkward nature of some operations, such as division [42, 120–122], magnitude comparison [41, 123–125], sign detection [41, 126], scaling [42, 127, 128], additive and multiplicative overflow detection [41, 126, 128–130], etc, which are significantly much simpler in the decimal and binary number systems. Furthermore, significant complexity is involved in the conversion from the RNS to the decimal or binary number system [42, 131, 132].

Due to the above-mentioned disadvantages, until recently the RNS arithmetic has not proved popular in general purpose computers. Instead, research interest has shifted to the fault tolerance characteristics of the RNS [41, 42, 42, 133, 134], for applications in digital signal processing (DSP) [126, 135–139], in modulation schemes [140] , etc, for the correction of both computational errors and transmission errors. In digital signal processing, the carry-free and fault tolerance properties of the RNS render them attractive for the implementation of digital filtering, that of the Fast Fourier Transform (FFT) spectral analysis, correlation, matrix operations and image processing [126, 131, 135–139, 141, 142]. A RNS-based M-ary modulation scheme has been proposed and analysed in [140].

Using the RNS, Szabo *et al.* [41] are amongst the earliest researchers, who have derived a method for single error detection and correction. However, the error correction procedure proposed by Szabo is computationally inefficient and it is implementationally complex. Watson and Hastings [42] exploited the properties of the redundant residue number system (RRNS) for detecting or correcting a single error and also for detecting multiple errors. The RRNS is based on a RNS, which has a number of redundant residues added to it. Watson's method for error correction needs a correction table, which may have a high memory requirement, thereby rendering the proposed technique impractical for the correction of more than a single error. Mandelbaum [143] showed how single error correction can be accomplished in a RRNS with the aid of less redundancy than that required in [42]. Later, Barsi and Maestrini [133] derived the necessary and sufficient conditions for the minimal amount of redundancy allowing the correction of an arbitrary single residue error. Watson's method was also used by Yau and Liu [134], but Watson's correction table was replaced by appropriate computations. Hence, the implementation proposed by Yau and Liu required less memory.

Recently, a coding theoretic approach to error control has been developed in [43, 44] in the context of the RRNS. The concepts of Hamming weight, minimum free distance, weight distribution, error detection capabilities and error correction capabilities was also introduced. A computationally efficient procedure was described in [43] for correcting a single error. In [44], the procedure was extended for correcting double errors and simultaneously correcting single and detecting multiple errors.

In this chapter, we combine our discourse by a rudimentary introduction to the RNS in Section 4.2. Some theorems associated with the RNS will be discussed in detail. Then, in Section 4.3 we present the coding theory of the RRNS. The options of implementing a channel codec using the RNS will be investigated in Section 4.7. Finally, our simulation results are given in Section 4.9.

4.2 Background

4.2.1 Conventional Number System

The decimal number system is the most widely used number system, in which any integer X can be represented by

$$\begin{aligned} X &= a_k 10^k + a_{k-1} 10^{k-1} + ... + a_1 10 + a_0 \\ &= \sum_{j=0}^{k-1} a_j r^j \ , \end{aligned} \tag{4.1}$$

where r is the radix of the system, which is equal to 10 for the decimal number system. The decimal digits a_j are integers in the interval [0,9]. Thus, the integer $X = 17$ can be represented in the decimal number system as

$$17 = 1 \times 10^1 + 7 \times 10^0 \ . \tag{4.2}$$

Apparently, the decimal number system is of *unlimited range*; i.e., any integer can be expressed in the system regardless of its magnitude. Furthermore, this is a *unique representation*, since each integer has only one representation and this representation is *non-redundant.* A non-redundant system is defined as the system in which every combination of the digits a_j represents a number and there is no two different sets of digits a_j, which correspond to the same number.

In Equation 4.1, each digit a_j is multiplied by a constant, hence the decimal number system is a *weighted number system.* In this instance, the constant or weight is 10^j, which is a power of the radix 10. Since all the weights used in this system are powers of the same base or radix, the decimal number system is referred to as a *fixed radix system*, where the fixed radix is 10.

In modern computers the binary number system is used to represent the operands. Similarly to the decimal number system, the binary number system has an unlimited range, and a unique representation. It is a non-redundant, fixed-radix number system. However, the radix is 2 and the coefficients a_j are integers in the interval [0,1]. The integer $X = 17$

is represented as

$$17 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \ . \tag{4.3}$$

In a weighted number system there are many advantages:

- Magnitude comparison of two numbers can be readily carried out by comparing the most significant digits.
- The range of the number system is easily extended by adding more digits.
- Overflow can be detected by checking the carry of the most significant digit.
- Simple polarity detection.
- Multiplication and division by a power of the radix (e.g. 2 for a binary number system) can be accomplished by simple arithmetic shifts.

The attributes which lead to these advantages impose a limitation on the speed with which the arithmetic operations can be performed. During the arithmetic operations, the carry information must be passed from digits of lesser significance to those of higher significance. Thus, it is impossible to process all digits in parallel. Furthermore, any errors encountered during the process will be propagated through to the most significant digit.

4.2.2 Residue Number System

The RNS is defined in terms of a k-tuple of pairwise relatively prime positive integers, $m_1, m_2, ..., m_k$, where each individual member is referred to as a modulus. Hence the greatest common divisor of (m_i, m_j) is 1 for $i \neq j$. The product of the moduli represents the dynamic range, M, of the RNS which is formulated as:

$$M = \prod_{j=1}^{k} m_j \ . \tag{4.4}$$

Any positive integer X in the range of $0 \leq X < M$ can be uniquely represented by a k-tuple residue sequence given by

$$X \longleftrightarrow (x_1, x_2, ..., x_k) \ , \tag{4.5}$$

where the quantity x_j is the least positive integer remainder of the division of X by m_j, which is expressed as the residue of X modulo m_j or $|X|_{m_j}$. The positive integer x_j is also referred to as the j-th residue digit of X.

Let us consider a three-modulus RNS, having moduli of $m_1 = 2$, $m_2 = 3$ and $m_3 = 5$. The range M of the RNS is

$$\begin{aligned} M &= m_1 \times m_2 \times m_3 \\ &= 2 \times 3 \times 5 \\ &= 30 \ , \end{aligned} \tag{4.6}$$

which implies that the system can represent any integers in the interval [0, 29] uniquely. For example, an integer $X = 17$ can be represented by

$$17 \longleftrightarrow (1, 2, 2) \ . \tag{4.7}$$

If another integer of $X = 47$ is represented by the system, one would find that the integer 47 has the same residue representation as the integer 17, namely, (1,2,2). The ambiguity of the residue representation is avoided, if only numbers from X to $X + 29$ are considered in this system, where X is an integer, which is normally 0. Table 4.1 shows the residue representation of integers -4 to $+31$. Clearly, we can see that the pattern of the residue representation repeats itself after 30 different patterns. For example, the integers 1 and 31 have the same residue representation.

Integers	Residues			Integers	Residues			Integers	Residues			Integers	Residues		
	Moduli				Moduli				Moduli				Moduli		
	2	3	5		2	3	5		2	3	5		2	3	5
-4	0	2	1	-3	1	0	2	-2	0	1	3	-1	1	2	4
0	0	0	0	+1	1	1	1	+2	0	2	2	+3	1	0	3
+4	0	1	4	+5	1	2	0	+6	0	0	1	+7	1	1	2
+8	0	2	3	+9	1	0	4	+10	0	1	0	+11	1	2	1
+12	0	0	2	+13	1	1	3	+14	0	2	4	+15	1	0	0
+16	0	1	1	+17	1	2	2	+18	0	0	3	+19	1	1	4
+20	0	2	0	+21	1	0	1	+22	0	1	2	+23	1	2	3
+24	0	0	4	+25	1	1	0	+26	0	2	1	+27	1	0	2
+28	0	1	3	+29	1	2	4	+30	0	0	0	+31	1	1	1

Table 4.1: Residue representation of the integers -4 to $+31$ using the moduli 2, 3 and 5

For a *signed number system*, we can represent any integers from $-M/2+1$ to $M/2$. Again, we used the previous number system, which has the moduli of 2, 3 and 5. Hence, we can represent any integers from -14 to 15. Each number has an n-tuple representation where,

$$x_j = \begin{cases} |X|_{m_j} & X \geq 0 \\ |M - |X||_{m_j} & X < 0 \ . \end{cases} \tag{4.8}$$

The signed RNS system is often referred to as a symmetric system.

4.2.3 Mixed Radix Number System

In Section 4.2.1 we have seen that the conventional decimal and binary number systems are fixed radix number systems, since all the weights used in Equation 4.1 are powers of the same radices. In a mixed radix system, the weights are products of a number of the radices. An integer X may be expressed in mixed radix form as [41]:

$$X = a_k \prod_{j=1}^{k-1} r_j + a_{k-1} \prod_{j=1}^{k-2} r_j + ... + a_3 r_1 r_2 + a_2 r_1 + a_1 \ , \tag{4.9}$$

where r_j are the radices and a_j, $0 \le a_j < r_j$, are the mixed radix digits. Explicitly, we can see that any positive integer in the interval $\left[0, \prod_{j=1}^{k} r_j - 1\right]$ may be represented by the system and that each number has an unique representation. Given a set of radices, $r_1, r_2, ...r_k$, the mixed radix representation of an integer X is given as follows:

$$X \longleftrightarrow (a_k, a_{k-1}, ..., a_2, a_1) \ . \tag{4.10}$$

Let us assume that we have a three-radix system, where $r_1 = 2$, $r_2 = 3$ and $r_3 = 5$, respectively. Hence, we can represent any integer in the interval $[0, 29]$, which are expressed as:

$$X = a_3 \times 6 + a_2 \times 2 + a_1 \ . \tag{4.11}$$

In Section 4.2.1 we have shown how the integer $X = 17$ is represented in the decimal and binary number systems with the aid of Equation 4.2 and 4.3, respectively. Applying the same principle, we can represent the integer $X = 17$ in the above-mentioned mixed radix number system as:

$$\begin{aligned} 17 &= 2 \times 6 + 2 \times 2 + 1 \\ 17 &\longleftrightarrow (2, 2, 1) \ . \end{aligned} \tag{4.12}$$

Table 4.2 shows the mixed radix representation of the integers 0 to 29.

4.2.4 Residue Arithmetic Operations

Let us assume that we have two integers, namely X_1 and X_2. The corresponding RNS representations are $X_1 \longleftrightarrow (x_{11}, x_{12}, ..., x_{1k})$ and $X_2 \longleftrightarrow (x_{21}, x_{22}, ..., x_{2k})$, respectively. Then, $(x_{11}, x_{12}, ..., x_{1k}) \odot (x_{21}, x_{22}, ..., x_{2k})$, where $\odot$ denotes addition, subtraction or multiplication, results in another unique k-tuple residue sequence namely $X_3 \longleftrightarrow (x_{31}, x_{32}, ..., x_{3k})$, as long as the number X_3 is in the range $[0, M-1]$. This is expressed as:

$$\begin{aligned} X_3 &= X_1 \odot X_2 \\ X_3 &\longleftrightarrow \left(|x_{11} \odot x_{21}|_{m_1}, |x_{12} \odot x_{22}|_{m_2}, ..., |x_{1k} \odot x_{2k}|_{m_k} \right) \ . \end{aligned} \tag{4.13}$$

Number	a_3	a_2	a_1	Number	a_3	a_2	a_1	Number	a_3	a_2	a_1
0	0	0	0	1	0	0	1	2	0	1	0
3	0	1	1	4	0	2	0	5	0	2	1
6	1	0	0	7	1	0	1	8	1	1	0
9	1	1	1	10	1	2	0	11	1	2	1
12	2	0	0	13	2	0	1	14	2	1	0
15	2	1	1	16	2	2	0	17	2	2	1
18	3	0	0	19	3	0	1	20	3	1	0
21	3	1	1	22	3	2	0	23	3	2	1
24	4	0	0	25	4	0	1	26	4	1	0
27	4	1	1	28	4	2	0	29	4	2	1

Table 4.2: Mixed radix representation of the integers 0 to 29 for radices 2, 3, 5

Again, let us consider the previous example using moduli $m_1 = 2$, $m_2 = 3$ and $m_3 = 5$. The dynamic range M is 30. The three arithmetic operations, namely addition, subtraction and multiplication, are illustrated below as:

$$\begin{array}{cccccccc}
 & 7 & \longleftrightarrow & (1, & 1, & 2) & & \\
+ & 4 & \longleftrightarrow & (0, & 1, & 4) & & \\
\hline
 & 11 & \longleftrightarrow & (1 \bmod 2, & 2 \bmod 3, & 6 \bmod 5) & \equiv (1,\ 2,\ 1) &
\end{array}$$

$$\begin{array}{cccccccc}
 & 7 & \longleftrightarrow & (1, & 1, & 2) & & \\
- & 4 & \longleftrightarrow & (0, & 1, & 4) & & \\
\hline
 & 3 & \longleftrightarrow & (1 \bmod 2, & 0 \bmod 3, & -2 \bmod 5) & \equiv (1,\ 0,\ 3) &
\end{array}$$

$$\begin{array}{cccccccc}
 & 7 & \longleftrightarrow & (1, & 1, & 2) & & \\
\times & 4 & \longleftrightarrow & (0, & 1, & 4) & & \\
\hline
 & 28 & \longleftrightarrow & (0 \bmod 2, & 1 \bmod 3, & 8 \bmod 5) & \equiv (0,\ 1,\ 3) & .
\end{array}$$

It can be seen from the above example that the j-th residue digit, namely x_{3j}, is uniquely and unambiguously defined in terms of $(x_{1j} \odot x_{2j})$ modulo m_j. That is, no carry information has to be communicated between the residue digits. Hence, the overhead of manipulating carry information in weighted number system, can be avoided. This results in high speed parallel operations and it makes RNS attractive. In the event of an error occurring in a residue digit operation, the error is confined within the operation and it does not affect the result of other operations.

It should be noted that the previous examples satisfy the condition $0 \leq X_1 \odot X_2 < M$. If this conditions is not satisfied, the correct results will not be obtained. For example,

$$\begin{array}{rrclll}
 & 7 & \longleftrightarrow & (1, & 1, & 2) \\
\times & 5 & \longleftrightarrow & (1, & 2, & 0) \\
\hline
 & 35 & \not\longleftrightarrow & (1 \bmod 2, & 2 \bmod 3, & 0 \bmod 5) \quad \equiv (1,2,0) \longleftrightarrow 5 \ .
\end{array}$$

Explicitly, the result represented by the residues is wrong, a condition, which is termed as overflow. In the RNS system, an overflow is difficult to detect, since the residue digits have the same significance. Other arithmetic operations such as, magnitude comparison, sign detection, division, etc. require more sophisticated procedures. Sometimes, we have to convert the residue digits to the decimal number system in order to accomplish the above-mentioned operations.

4.2.4.1 Multiplicative Inverse

In certain applications of the RNS the multiplicative inverse of an operand has to be determined. If $0 \leq L < m$ and $|X.L|_m = 1$, L is referred to as the *multiplicative inverse of* X *modulo* m. Table 4.3 shows the multiplicative inverse of X for moduli 2, 3 and 5. Observe that there exists no multiplicative inverse of $X = 2$ for modulus $m = 4$, since no L can be found, for which $|X.L|_m = 1$

$m = 3$		$m = 4$		$m = 5$	
X	L	X	L	X	L
1	1	1	1	1	1
2	2	2	-	2	3
-	-	3	3	3	2
-	-	-	-	4	4

Table 4.3: Multiplicative inverse of various values of X modulo 3, 4, 5

The multiplicative inverse L is useful for example for converting the division of a residue digit x_j by a number X, to the multiplication of the residue digit x_j by the multiplicative inverse L of the number X modulo m. It is important to note however that this only applies, if the remainder of the integer division $\frac{x_j}{X}$ is zero.

Let us now illustrate the above concepts further using an example. Let us assume that we have a modulus of $m = 5$ and the residue is $x_j = 4$. The result of $\frac{x_j}{X}$, $X = 2$, is to be calculated using the multiplicative inverse L of X modulo m. From Table 4.3, we know that the multiplicative inverse L of $X = 2$ modulo $m = 5$ is equal to 3. Therefore the required division can be carried out with the aid of a multiplication by the multiplicative inverse L

as follows:

$$\begin{aligned}\frac{4}{2} &= 2, \qquad\qquad \text{Remainder} = 0 \\ \frac{4}{2} &\equiv |4 \times 3|_5 \\ &= 2\,. \end{aligned} \tag{4.14}$$

As stated above, the division can only be replaced with the aid of multiplication by the multiplicative inverse, if the remainder of the integer division $\frac{x_j}{X}$ is zero. Hence in the following example we demonstrate that for a remainder of 1 the division cannot be replaced by the above-mentioned multiplication. Explicitly, if $X = 3$, the multiplicative inverse L of $X = 3$ modulo $m = 5$ is equal to 2,

$$\begin{aligned}\frac{4}{3} &= 1, \qquad\qquad \text{Remainder} = 1 \\ \frac{4}{3} &\not\equiv |4 \times 2|_5 \\ &= 3\,. \end{aligned} \tag{4.15}$$

The concept of the multiplicative inverse is important, when we have to convert the RNS representation of an integer to the decimal number system, using the Mixed Radix Conversion (MRC) method, which will be introduced in Section 4.2.5.2.

4.2.5 Residue to Decimal Conversion

There are two methods of residue to decimal conversion; the Chinese Remainder Theorem (CRT) [40, 41] and the MRC [41].

4.2.5.1 Chinese Remainder Theorem

The classic CRT uses the following expression for residue to decimal conversion:

$$\begin{aligned} X &= \left[M_1 \times |x_1 L_1|_{m_1} + M_2 \times |x_2 L_2|_{m_2} + \ldots + M_k \times |x_k L_k|_{m_k}\right] \bmod M \\ &= \left[\sum_{j=1}^{k} M_j |x_j L_j|_{m_j}\right] \bmod M\ , \end{aligned} \tag{4.16}$$

where $M = \prod_{j=1}^{k} m_j$ and $M_j = \frac{M}{m_j}$, while L_j is the multiplicative inverse of M_j mod m_j, which was formulated as $|(L_j M_j)|_{m_j} = 1$.

In order to show the validity of Equation 4.16, we take the modulo value m_i of both sides of the equation, yielding:

$$\begin{aligned} X \bmod m_i &= \left\{ \left[\sum_{j=1}^{k} M_j |x_j L_j|_{m_j} \right] \bmod M \right\} \bmod m_i \\ &= M_i |x_i L_i|_{m_i} \bmod m_i \ , \qquad (4.17) \end{aligned}$$

since $M_j \bmod m_i = 0$, except for $j = i$. Since we have shown previously that $(L_j M_j) \bmod m_j = 1$, we can reduce Equation 4.17 to

$$\begin{aligned} X \bmod m_j &= x_j \bmod m_j \\ &= x_j \ , \qquad (4.18) \end{aligned}$$

which is simply the residue of the integer X upon division by m_j and hence it is valid by definition. Therefore, Equation 4.16 has been proven.

Let us now use an example in order to explain the operation of the CRT more clearly. A three-modulus RNS employing the moduli $m_1 = 3$, $m_2 = 4$ and $m_3 = 5$ is used. The residue representation of an unknown integer X is $(1, 3, 4)$. The corresponding values M_j and L_j are:

$$\begin{array}{lll} M = 3 \times 4 \times 5 & = 60 & \\ M_1 = 20 & M_2 = 15 & M_3 = 12 \\ |M_1|_{m_1} = 2 & |M_2|_{m_2} = 3 & |M_3|_{m_3} = 2 \\ L_1 = 2 & L_2 = 3 & L_3 = 3, \end{array}$$

where $(L_j \times |M_j|_{m_j})$ modulo $m_j = 1$. Using Equation 4.16, we arrive at:

$$\begin{aligned} X &= \left[M_1 |x_1 L_1|_{m_1} + M_2 |x_2 L_2|_{m_2} + M_3 |x_3 L_3|_{m_3} \right] \bmod M \\ &= [20 \times |1 \times 2|_3 + 15 \times |3 \times 3|_4 + 12 \times |4 \times 3|_5] \bmod 60 \\ &= [40 + 15 + 24] \bmod 60 \\ &= 19 \ . \qquad (4.19) \end{aligned}$$

In principle, this method is fairly simple and straightforward. However, the associated operations are not readily implementable if the range M is high. For instance, the summation of $M_j(x_j L_j)$, in a ten-modulus RNS system having moduli ranging from 100 to 128, could easily exceed the typical dynamic range of 2^{64} of the computer used. We can solve this problem by modifying Equation 4.16 according to:

$$X = \left[\sum_{j=1}^{k} \left| M_j |x_j L_j|_{m_j} \right|_M \right] \bmod M \ . \qquad (4.20)$$

However, this representation requires more computational power and the dynamic range M has to be less than that of the computer carrying out this conversion. By contrast, the MRC can be readily implemented, since it requires operations modulo m_j only.

4.2.5.2 Mixed Radix Conversion

In Section 4.2.3 we have highlighted the fundamental philosophy of the mixed radix number system and that of the multiplicative inverse. Let us now consider the conversion of the residues x_j, in the RNS to the mixed radix digits a_j. Using the mixed radix digits a_j, the decimal number represented by the RNS can be readily calculated with the aid of Equation 4.9.

If a set of moduli $m_1, m_2, ..., m_k$ and a set of radices $r_1, r_2, ..., r_k$ are chosen so that $m_j = r_j$, the mixed radix system and the residue number system are said to be associated. In this case both systems have the same dynamic range of values, that is, $\left[0, \prod_{j=1}^{k} m_j - 1\right]$. Below, we present the process of converting an operand from the residue number system to the mixed radix number system.

Since $m_j = r_j$, we can rewrite Equation 4.9 as:

$$X = a_k \prod_{j=1}^{k-1} m_j + a_{k-1} \prod_{j=1}^{k-2} m_j + ... + a_3 m_1 m_2 + a_2 m_1 + a_1 \ , \tag{4.21}$$

where a_j are the mixed radix coefficients. Let us first take Equation 4.21 modulo m_1. As we can see in Equation 4.21, all the terms, except for the last one, are multiples of the modulus m_1. Hence, we have

$$|X|_{m_1} = x_1 = a_1 \ . \tag{4.22}$$

Explicitly Equation 4.22 indicates that a_1 is the same as the first residue digit, namely x_1, of the RNS.

In order to obtain the coefficient a_2, we have to subtract a_1 from Equation 4.21:

$$X - a_1 = a_k \prod_{j=1}^{k-1} m_j + a_{k-1} \prod_{j=1}^{k-2} m_j + ... + a_3 m_1 m_2 + a_2 m_1 \ . \tag{4.23}$$

Equation 4.23 is now divisible by m_1, yielding:

$$\frac{X - a_1}{m_1} = a_n \prod_{j=2}^{k-1} m_j + a_{k-1} \prod_{j=2}^{k-2} m_j + ... + a_3 m_2 + a_2 \ . \tag{4.24}$$

Again, all the terms in Equation 4.24 are multiples of the modulus m_2. Hence we take Equation 4.24 modulo m_2 and we arrive at:

$$\left|\frac{X - a_1}{m_1}\right|_{m_2} = a_2 \ . \tag{4.25}$$

If we repeat the above procedures, namely subtracting, dividing and taking modulo, all the mixed radix digits may be obtained. Once all the mixed radix digits, a_j, have been found, the mixed radix conversion is accomplished using Equation 4.9.

It is interesting to note that

$$\begin{aligned}
a_1 &= |X|_{m_1} \\
a_2 &= \left|\left[\frac{X}{m_1}\right]\right|_{m_2} \\
a_3 &= \left|\left[\frac{X}{m_1 m_2}\right]\right|_{m_3} \\
&\cdot \\
&\cdot \\
a_k &= \left|\left[\frac{X}{m_1 m_2 \ldots m_{k-1}}\right]\right|_{m_k} ,
\end{aligned} \tag{4.26}$$

where $\left[\frac{X}{m_j}\right]$ is the integer value of the quotient $\frac{X}{m_j}$.

In Section 4.2.5.1 we have shown using an example, how to invoke the CRT for converting residue digits to a decimal number. Here, we use the same example, but we employ the mixed radix conversion method of Equation 4.21. The RNS has the moduli of $m_1 = 3$, $m_2 = 4$ and $m_3 = 5$. The residue representation of the unknown integer X is $(1, 3, 4)$. We substitute m_j into Equation 4.21 and we get

$$X = a_3(3 \times 4) + a_2(3) + a_1 , \tag{4.27}$$

which will be used to calculate the decimal representation of the unknown integer X. However, we have to determine the mixed radix coefficients a_j in Equation 4.27.

Applying Equation 4.22,

$$a_1 = x_1 = 1 . \tag{4.28}$$

We subtract $a_1 = 1$ from $X = 19$, yielding $19 - 1 = 18$, which is formulated in the residue domain as:

$$\begin{array}{rcccc}
X & \overset{3,4,5}{\longleftrightarrow} & (1, & 3, & 4) \\
- \quad a_1 & \overset{3,4,5}{\longleftrightarrow} & (1, & 1, & 1) \\
\hline
X - a_1 & \overset{3,4,5}{\longleftrightarrow} & (0, & 2, & 3)
\end{array} ,$$

where $\overset{3,4,5}{\longleftrightarrow}$ denotes the residue representation of a number in the RNS using moduli 3, 4 and 5.

In Equation 4.24, we have shown that $X - a_1$ is divisible by m_1 and the remainder is equal to zero. Therefore, instead of employing the more cumbersome division operation of $\frac{X-a_1}{m_1}$ we can invoke the multiplication of $X - a_1$ with the multiplicative inverse L_j of m_1 modulo m_j. The required multiplicative inverse values are shown in Table 4.3, yielding:

$$\begin{array}{rcccc} & X - a_1 & \overset{4,5}{\longleftrightarrow} & (2, & 3) \\ \times & L_j & & 3 & 2 \\ \hline & \frac{X-a_1}{m_1} & \overset{4,5}{\longleftrightarrow} & (|6|_4, & |6|_5) \longleftrightarrow (2, \quad 1). \end{array}$$

As shown in Equation 4.25, $\left|\frac{X-a_1}{m_1}\right|_{m_2} = a_2$. Hence, a_2 is equal to the residue of $\frac{X-a_1}{m_1} = 6$ modulo 4, giving $a_2 = 2$.

We repeat the above process again and we subtract $a_2 = 2$ from $\frac{X-a_1}{m_1} = 6$, yielding 4, which is formulated in the residue domain as:

$$\begin{array}{rcccc} & \frac{X-a_1}{m_1} & \overset{4,5}{\longleftrightarrow} & (2, & 1) \\ - & a_2 & \overset{4,5}{\longleftrightarrow} & (2, & 2) \\ \hline & \frac{X-a_1}{m_1} - a_2 & \overset{4,5}{\longleftrightarrow} & (|0|_4, & |-1|_5) \longleftrightarrow (0, 4). \end{array}$$

Then again, instead of carrying out the division we multiply with L_j, yielding:

$$\begin{array}{rccc} & \frac{X-a_1}{m_1} - a_2 & \overset{5}{\longleftrightarrow} & (4) \\ \times & L_j & & 4 \\ \hline & \frac{\frac{X-a_1}{m_1} - a_2}{m_2} & \overset{5}{\longleftrightarrow} & (|16|_5) \longleftrightarrow (1), \end{array}$$

and hence a_3 is found to be 1.

Finally, we substitute all mixed radix coefficients a_j $j = 1, 2, 3$ into Equation 4.27, giving:

$$\begin{aligned} X &= 12 \times 1 + 3 \times 2 + 1 \\ &= 19 \,. \end{aligned} \tag{4.29}$$

Having exemplified the various conversion processes between the residue and decimal domains, let us now consider the construction of redundant RNSs, which are used in RNS-based error correction coding.

4.2.6 Redundant Residue Number System

As we have highlighted in Section 4.2.2, the RNS is defined by the choice of k moduli, namely $m_1, m_2, ..., m_k$. The dynamic range M of the RNS is $\prod_{j=1}^{k} m_j$. In a Redundant Residue

Number System (RRNS), extra moduli, namely $m_{k+1}, m_{k+2}, ..., m_{k+n}$, are incorporated into the RNS. As in the RNS, the moduli, $m_1, m_2, ..., m_k, m_{k+1}, ..., m_n$, are chosen to be pairwise relatively prime positive integers and $m_{k+j} \geq max\{m_1, m_2, ..., m_k\}$. Moduli $m_1, m_2, ..., m_k$ are considered to be non-redundant moduli and moduli $m_{k+1}, m_{k+2}, ..., m_n$ are the redundant moduli. The redundant moduli are not considered to increase the dynamic range, $M = \prod_{j=1}^{k} m_j$, even though the redundant residues related to an integer operand now become part of the residue representation of that integer. The interval $[0, M - 1]$ is referred to as the *legitimate range*, where $M = M_k = \prod_{j=1}^{k} m_j$, and the interval $[M_k, M_n - 1]$ is the *illegitimate range*, where $M_n = \prod_{j=1}^{n} m_j$.

In Section 4.2.5.1 the RNS having moduli $m_1 = 3$, $m_2 = 4$ and $m_3 = 5$ was used. The corresponding RNS representation of the integer $X = 19$ is $(1, 3, 4)$. If we incorporate a redundant modulus of $m_4 = 7$, the integer will be represented by the RRNS as:

$$X \longleftrightarrow (1, 3, 4, 5) \ . \tag{4.30}$$

In general, any k of the n residue digits could be used to calculate the decimal representation of the integer X, if and only if $m_{k+j} \geq max\{m_1, m_2, ..., m_k\}$ [41]. The conversion method could be either the CRT or the MRC. Let us assume that a computer uses the above-mentioned RRNS as its number representation system. Due to a fault of a module in the computer, the residue x_2 associated with the modulus m_2 is no longer valid. However, we can still calculate the decimal representation of the integer X using the residues x_1, x_3 and x_4. These residues constitute the residue representation of a new RNS, which has m_1, m_3 and m_4 as its moduli. The dynamic range M of the new RNS is 105, which is more than the legitimate range of the RRNS. This implies that any number, within the legitimate range of the RRNS, is represented unambiguously by the new RNS. Furthermore, the new RNS and the RRNS have the same residues with respect to the moduli m_1, m_3 and m_4. Hence, we can calculate the original decimal number, even though we only have the residue digits x_1, x_3 and x_4. Employing the CRT, the corresponding values M_j and L_j of the new RNS are:

$$\begin{array}{lll} M = 3 \times 5 \times 7 & = 105 & \\ M_1 = 35 & M_3 = 21 & M_4 = 15 \\ |M_1|_{m_1} = 2 & |M_3|_{m_3} = 1 & |M_4|_{m_4} = 1 \\ L_1 = 2 & L_3 = 1 & L_4 = 1, \end{array}$$

where $(L_j \times |M_j|_{m_j})$ modulo $m_j = 1$. Using Equation 4.16, we arrive at:

$$\begin{aligned} X &= [35 \times |1 \times 2|_3 + 21 \times |4 \times 1|_5 + 15 \times |5 \times 1|_7] \bmod 105 \\ &= [70 + 84 + 75] \bmod 105 \\ &= 19 \ . \end{aligned} \tag{4.31}$$

The above simple example demonstrated an application of the RRNS. The RRNS is used extensively also in the field of error detection and correction schemes. Our detailed discussions on RRNS-based error correction coding are postponed to Section 4.3.

4.2.7 Base Extension

In the various applications of RNS arithmetics [41] it is often necessary to find the residue digits with respect to a new set of moduli, given the residue digits related to another set of moduli. In most cases the new set of moduli will be an extension of the original set; that is, one or more additional moduli are incorporated in the original set. This is the case for example, when creating the redundant moduli from the non-redundant moduli of a RNS. Normally, the CRT is employed to find the decimal number represented by the original set of moduli. This decimal number is then used to calculate the residues of the new set of moduli. However, there is a simpler procedure known as *base extension* (BEX). It is related to the MRC, with an additional final step, as highlighted below.

Consider a RNS consisting of moduli $m_1, m_2, ..., m_k$. The dynamic range of the RNS is $M = \prod_{j=1}^{k} m_j$. If another modulus, m_{k+1}, is incorporated into the RNS, the dynamic range will be extended and becomes $M = \prod_{j=1}^{k+1} m_j$. Therefore, we have to add another term to Equation 4.9, yielding the mixed radix representation of the integer X in the form of:

$$X = a_{k+1} \prod_{j=1}^{k} m_j + a_k \prod_{j=1}^{k-1} m_j + ... + a_3 m_1 m_2 + a_2 m_1 + a_1 \ . \tag{4.32}$$

Any integer X, represented by the original k moduli, will be in the interval $\left[0, \prod_{j=1}^{k} m_j - 1\right]$. If the integer X is represented by the extended moduli in the mixed radix form, as shown in Equation 4.32, a_{k+1} clearly will be equal to zero. In performing the MRC, the fact that $a_{k+1} = 0$ will be used to find $|X|_{m_{k+1}}$. The method is best illustrated with the aid of a numerical example.

In Section 4.2.6, we found the residue representation of the integer $X = 19$ in the RRNS. The process is straightforward and the residue representation of the integer $X = 19$ is $(1, 3, 4, 5)$, given that the moduli are $m_1 = 3$, $m_2 = 4$, $m_3 = 5$ and $m_4 = 7$. Let us assume that in this example we have no prior knowledge of the decimal representation of the integer

X. The residue representation of the integer X is $(1, 3, 4)$, given that the moduli are $m_1 = 3$, $m_2 = 4$ and $m_3 = 5$. A redundant modulus, $m_4 = 7$, is appended to the RNS and hence we have to find $|X|_7$.

In the RRNS, the residue representation of X will be $(1, 3, 4, |X|_7)$. The process of determining $|X|_7$ is initiated by performing the MRC in Section 4.2.5.2 in the usual manner, but including the $|X|_7$ value in the operations. We commence by recalling that

Moduli:	3	4	5	7	
Residue Representation	1	3	4	$\lvert X\rvert_7$	$a_1 = 1$
Subtract $a_1 = 1$	1	1	1	1	
	0	2	3	$\lvert X\rvert_7 + 6$	
Multiply by L_j, $\lvert 3.L_j\rvert_{m_j} = 1$		3	2	5	
$\frac{X-a_1}{m_1}$		2	1	$5\lvert X\rvert_7 + 2$	$a_2 = 2$
Subtract $a_2 = 2$		2	2	2	
		0	4	$5\lvert X\rvert_7$	
Multiply by L_j, $\lvert 4.L_j\rvert_{m_j} = 1$			4	2	
$\frac{\frac{X-a_1}{m_1}-a_2}{m_2}$			1	$3\lvert X\rvert_7$	$a_3 = 1$
Subtract $a_3 = 1$			1	1	
			0	$3\lvert X\rvert_7 + 6$	
Multiply by L_j, $\lvert 5.L_j\rvert_{m_j} = 1$				3	
$\frac{\frac{\frac{X-a_1}{m_1}-a_2}{m_2}-a_3}{m_3}$				$2\lvert X\rvert_7 + 4$	

Since a_4 corresponds to a_{k+1} in Equation 4.32, a_4 is equal to zero. Therefore,

$$a_4 = |2|X|_7 + 4|_7 = 0 \ , \tag{4.33}$$

and

$$\begin{aligned} |2|X|_7|_7 &= |-4|_7 \\ &= 3 \ . \end{aligned} \tag{4.34}$$

Multiplying by the multiplicative inverse of 2 modulo 7, $|L \times 2|_7 = 1 \Rightarrow L = 4$ yields:

$$|X|_7 = |3 \times 4|_7 = 5 \ . \tag{4.35}$$

Hence, the residue representation of integer X in the RRNS is $(1, 3, 4, 5)$.

The BEX operation is fundamental to several important arithmetic operations, such as scaling [40, 41], dynamic range extension, magnitude comparison, overflow detection and sign determination [41, 126].

4.3 Coding Theory of Redundant Residue Number Systems

As described earlier, RRNSs have been studied extensively for the protection of arithmetic and data transmission operations in general purpose computers [41–44, 133, 134, 143], in digital filters [135] and in modulation schemes [140]. In this section, we will discuss a coding theoretic approach to error control using the RRNS [43, 44]. The concepts of Hamming weight, minimum free distance, error detection and correction capabilities of RRNS based codes are introduced. The necessary and sufficient conditions for the desired error control capability are derived from minimum distance point of view. In a special case, we are capable of generating the maximum distance separable RRNS (MDS-RRNS).

4.3.1 Minimum Free Distance of RRNS Based Codes

The minimum distance is a fundamental parameter associated with any error control code and it very much affects the performance of the code. As shown in Section 4.2.6, the integer number X, which is within the legitimate dynamic range M, can be represented in the RRNS by a set of residues or a valid codeword $\underline{x}$. The number of non-zero residues of $\underline{x}$ is the *Hamming weight*, weight$(\underline{x})$ of the codeword. The *Hamming distance* between two codewords $\underline{x}_i$ and $\underline{x}_j$, $dist(\underline{x}_i, \underline{x}_j)$ is the number of residue positions in which $\underline{x}_i$ and $\underline{x}_j$ differ. Hence, we can define the minimum distance d_{min} of the RRNS based codes as:

$$d_{min} = \min \left\{ dist(\underline{x}_i, \underline{x}_j) \right\} , \tag{4.36}$$

where $\underline{x}_i \neq \underline{x}_j$ and $\underline{x}_i$, $\underline{x}_j$ are legitimate codewords in the RRNS. However, if the number of possible codewords $\underline{x}$ in the RRNS is high, it may not be possible to compute d_{min} using a full-search based on Equation 4.36.

In [43], Krishna *et al.* derived the necessary and sufficient conditions imposed on the redundant moduli in order for an RRNS code to have a minimum distance equal to d_{min}. The minimum free distance of an RRNS code is d_{min} if and only if the product of the redundant moduli satisfies the following relation [43]:

$$\max \left\{ \prod_{i=1}^{d_{min}} m_{j_i} \right\} > M_{n-k} \geq \max \left\{ \prod_{i=1}^{d_{min}-1} m_{j_i} \right\} , \tag{4.37}$$

where the illegitimate range is given by $M_{n-k} = \prod_{j=k+1}^{n} m_j$ and m_{j_i} $1 \leq j_i \leq n$, is an arbitrary modulus of the RRNS code. In simple terms a good code aims to attain the highest possible minimum distance, since this maximises the error correction capabilities of the code, whilst maximising the code rate. Maximising the code rate is achieved by maximising the useful information dynamic range M_k and hence minimising the illegitimate range M_{n-k}, since the total dynamic range $M_n = M_k \cdot M_{n-k} = \prod_{j=1}^{n} m_j$ is constant.

Proof: We will show the validity of Equation 4.37 in two steps, namely the validity of the right hand side inequality first and then the validity of the left hand side inequality. Consider a codeword having a Hamming weight of α, which implies having non-zero residues in positions $j_1, j_2, ..., j_\alpha$ of the codeword, and zero residues elsewhere. As a result, X which represents the set of residues describing a codeword is a multiple of $m_j, j = 1, 2, ..., n; j \neq j_1, j_2, ..., j_\alpha$. Thus, we can write the integer X as:

$$X = X' \prod_{\substack{j=1 \\ j \neq j_1, j_2, \ldots, j_\alpha}}^{n} m_j \ , \tag{4.38}$$

where X' is an arbitrary integer satisfying

$$0 < X' < m_{j_1} m_{j_2} ... m_{j_\alpha} \ . \tag{4.39}$$

For an RRNS code to have a minimum distance of d_{min}, the following conditions must be satisfied [43]:

1. There is no valid codeword of Hamming weight $d_{min} - 1$ or less, except for the all-zero codeword;
2. There is at least one valid codeword having a Hamming weight of d_{min}.

The first condition implies that if the Hamming weight α of codeword $\underline{x}$ obeys $\alpha \leq d_{min} - 1$, then $X \geq M_k = \prod_{j=1}^{k} m_j$ which means that X is in the illegitimate range of the RRNS, because except for the all-zero codeword there are no legitimate codewords having a Hamming weight of $\alpha \leq d_{min} - 1$. Considering now all the legitimate codewords, for which the condition $\alpha > d_{min} - 1$ is trivially satisfied, we have $X < M_k$, which implies that X is in the legitimate range. Hence $\underline{x}$ is a valid codeword. The first condition is satisfied trivially if and only if $X \geq M_k$ and the number of non-zero residues obeys $\alpha \leq d_{min} - 1$. Let us therefore show that even the smallest possible value of X is outside the legitimate range M_k, i.e. it falls in the illegitimate range. The smallest possible value of X according to Equation 4.38 is obtained upon setting $X' = 1$, and including the smallest $(n - \alpha)$ number of moduli for the set of n moduli. Consequently, the moduli must satisfy [43]:

$$\begin{aligned}
\min \left\{ \prod_{\substack{j=1 \\ j \neq j_1, j_2, \ldots, j_{d_{min}-1}}}^{n} m_{j_i} \right\} &\geq M_k \\
\frac{M_k M_{n-k}}{\max \left\{ \prod_{i=1}^{d_{min}-1} m_{j_i} \right\}} &\geq M_k \\
M_{n-k} &\geq \max \left\{ \prod_{i=1}^{d_{min}-1} m_{j_i} \right\} .
\end{aligned} \tag{4.40}$$

The second condition implies that there is at least one codeword $\underline{x}$, representing the integer $X < M_k$, that has a Hamming weight of $\alpha = d_{min}$. Again, we set $X' = 1$ and include the smallest $(n - \alpha)$ number of moduli from the set of n moduli in Equation 4.38. Hence, we have

$$\begin{aligned}
\min \left\{ \prod_{\substack{j=1 \\ j \neq j_1, j_2, \ldots, j_{d_{min}}}}^{n} m_{j_i} \right\} &< M_k \\
\frac{M_k M_{n-k}}{\max \left\{ \prod_{i=1}^{d_{min}} m_{j_i} \right\}} &< M_k \\
M_{n-k} &< \max \left\{ \prod_{i=1}^{d_{min}} m_{j_i} \right\} ,
\end{aligned} \tag{4.41}$$

which completes the proof $\square$.

Let us now consider an example of the RRNS using the following moduli [43, 133]:

$$(m_1, m_2, m_3, m_4, m_5, m_6) = (3, 7, 11, 13, 16, 17) . \tag{4.42}$$

Here we show how various RRNS codes exhibiting different distance properties and error correction capabilities, can be derived using these moduli. As mentioned before, our aim is to maximise the minimum distance and the useful information dynamic range M_k. This implies minimising M_{n-k}, since $M_n = M_{n-k} \cdot M_k = \prod_{j=1}^{n} m_j = C$, where C is a constant. Specifically, according to Equation 4.37, the minimum free distance of the RRNS becomes for example $d_{min} = 3$ if and only if the redundant moduli of the code satisfy:

$$\begin{aligned}
\max \{m_{j_1} m_{j_2} m_{j_3}\} &> M_{n-k} \geq \max \{m_{j_1} m_{j_2}\} \\
3536 &> M_{n-k} \geq 272 .
\end{aligned} \tag{4.43}$$

Therefore, we can have an RRNS code which has a minimum free distance of $d_{min} = 3$ by using the following set of moduli as the redundant moduli: $\{m_5, m_6 : M_{n-k} = 272\}$,

$\{m_1, m_2, m_4 : M_{n-k} = 273\}$, $\{m_1, m_2, m_5 : M_{n-k} = 336\}$, and so on. One can also readily verify that if the redundant moduli of $\{m_1, m_2, m_3, m_5 : M_{n-k} = 3696\}$ are chosen, then the RRNS has a minimum free distance of $d_{min} = 4$.

It is plausible from Equation 4.37 and with the aid of the above example that for a given RRNS and for the minimum distance of d_{min}, the choice of the redundant moduli is not unique. However, we can find a set of redundant moduli, which requires the minimum redundant range of M_{n-k} for maintaining a given minimum distance d_{min}. The corresponding RRNS is termed the optimal RRNS. Note that minimising M_{n-k} means maximising the useful information dynamic range M_k, since $M_n = M_k \times M_{n-k}$ is a constant. This implies that the number of possible codewords $\underline{x}$ or the *dynamic range* of the code is maximised. In other words, the code rate is maximised, while maintaining a given minimum distance and error correction capability. Since the lowest illegitimate range is associated with the choice of $\{m_5, m_6 : M_{n-k} = 272\}$, this is our preferred option, since it ensures the highest useful information dynamic range M_k. Furthermore, it requires only two redundant moduli for achieving $d_{min} = 3$ and hence four information moduli can be used. The associated code can be denoted as a RRNS(6,4,3) code.

From Equation 4.40, the smallest value of M_{n-k} for a minimum distance of d_{min} is obtained by setting

$$M_{n-k} = \max \left\{ \prod_{i=1}^{d_{min}-1} m_{j_i} \right\} , \tag{4.44}$$

where $1 \le j_i \le n$. Equation 4.44, which was derived from the right hand side of Equation 4.37, shows that the left hand side of the inequality in Equation 4.37 is satisfied trivially. It also shows that an optimal RRNS having a minimum distance of d_{min} uses the $(d_{min} - 1)$ number of largest moduli from the set of n moduli as its redundant moduli. Therefore, we can write

$$\begin{aligned} d_{min} - 1 &= n - k \\ n &= k + d_{min} - 1 \ . \end{aligned} \tag{4.45}$$

Using a coding theoretic terminology, we will refer to an RRNS that satisfies Equation 4.45 as the maximum distance separable RRNS. In our previous example, if moduli m_5 and m_6 are chosen as the redundant moduli, we obtain the maximum distance separable RRNS code having a minimum distance of 3. The useful information dynamic range of this RRNS is $[0, 3003]$. At this stage it is worth noting that RRNS codes exhibit strong similarities with the well-known class of Reed-Solomon (RS) code. They are both non-binary codes. RS codes convey fixed number of bits per symbol, while RRNS codes may have a different number of bits per residue, as we will show in more depth during our further discourse. They also have similar distance properties.

4.3.2 Linearity of RRNS Codes

An RRNS(n,k) code is a block code, since it accepts k information symbols and generates n coded symbols. There are other block codes, such as the family of BCH and Reed-Solomon codes, which are linear block codes. The fundamental properties of linear block codes are [86]:

- All-zero vector is a valid codeword;
- The sum of two valid codewords is also a valid codeword.

Assume that there are two codewords $\underline{x}_1$ and $\underline{x}_2$ in an RRNS(n,k) code. The corresponding integers are X_1 and X_2, respectively, which are in the range of $[0, M_k - 1]$. Let $X_3 = X_1 + X_2$. If $X_3 < M_k$, then vector $\underline{x}_3$ is a valid codeword. However, if $M_k \leq X_3 < M_n$, then $\underline{x}_3$ is not a valid codeword. Clearly, this violates the definition of a linear code. Similarly, for a given scalar α such that $M_k \leq \alpha X_1 < M_n$, the vector $\alpha\underline{x}_1$ is not a valid codeword either. Hence, in [43] the RRNS codes are termed as the *semi-linear* block codes. The term *semi-linearity* or *conditional linearity* implies that the property of linearity is satisfied under certain appropriate predefined conditions.

4.3.3 Error Detection and Correction in RRNS Codes

In this section we will relate the minimum free distance d_{min} of the RRNS code to the error detection and error correction capabilities of the code. In our forthcoming discourse, the triangular inequality will be used repeatedly. Consider three arbitrary residue vectors $\underline{A}$, $\underline{B}$ and $\underline{C}$ in the RRNS codes, where the corresponding integer values are X_A, X_B and X_C, respectively. The Hamming distance among the residue vectors or RRNS codewords satisfies the triangular inequality [43, 96]:

$$dist(\underline{A},\underline{B}) + dist(\underline{B},\underline{C}) \geq dist(\underline{A},\underline{C}) \ . \tag{4.46}$$

Proof: Let us assume that $X_C > X_A$. Therefore, recalling the definition of Hamming distance and weight from the beginning of Section 4.3.1, we have:

$$dist(\underline{A},\underline{C}) = \text{weight}(\underline{C} - \underline{A}) \ , \tag{4.47}$$

which indicates the number of residue positions where $\underline{A}$ and $\underline{C}$ differ.

There are three cases, depending on the value of X_B:

1. $X_C > X_B > X_A$
2. $X_C > X_A > X_B$
3. $X_B > X_C > X_A$

For case 1, we can write

$$\begin{aligned} dist(\underline{A}, \underline{C}) &= \text{weight}(\underline{C} - \underline{A}) \\ &= \text{weight}\,\{(\underline{C} - \underline{B}) + (\underline{B} - \underline{A})\} \\ &= \text{weight}(\underline{C} - \underline{B}) + \text{weight}(\underline{B} - \underline{A}) + \alpha_1 + 2\alpha_2\ , \end{aligned} \tag{4.48}$$

where α_1 is the number of residue positions, where the non-zero residue digits of $(\underline{C} - \underline{B})$ and $(\underline{B} - \underline{A})$ add to a non-zero number, and α_2 is the number of residue positions, where the non-zero residue digits of $(\underline{C} - \underline{B})$ and $(\underline{B} - \underline{A})$ add to zero. Since $\alpha_1 + 2\alpha_2 \geq 0$, we have:

$$\text{weight}(\underline{C} - \underline{B}) + \text{weight}(\underline{B} - \underline{A}) \geq \text{weight}(\underline{C} - \underline{A})\ . \tag{4.49}$$

For case 2, we can write

$$\begin{aligned} dist(\underline{A}, \underline{C}) &= \text{weight}(\underline{C} - \underline{A}) \\ &= \text{weight}(\underline{C} - \underline{A} + \underline{B} + \underline{m} - \underline{B}) \\ &= \text{weight}\,\{(\underline{C} - \underline{B}) + \underline{m} - (\underline{A} - \underline{B})\} \\ &= \text{weight}(\underline{C} - \underline{B}) + \text{weight}\,\{\underline{m} - (\underline{A} - \underline{B})\} - \alpha_3 - 2\alpha_4 \\ &\leq \text{weight}(\underline{C} - \underline{B}) + \text{weight}\,\{\underline{m} - (\underline{A} - \underline{B})\} \\ &= \text{weight}(\underline{C} - \underline{B}) + \text{weight}(\underline{A} - \underline{B})\ , \end{aligned} \tag{4.50}$$

where $\underline{m}$ represents the vector of all moduli, α_3 is the number of residue positions in which the non-zero residue digits of $(\underline{C} - \underline{B})$ and weight $\{\underline{m} - (\underline{A} - \underline{B})\}$ add to a non-zero number, and α_4 is the number of residue positions where the non-zero residue digits $(\underline{C} - \underline{B})$ and weight $\{\underline{m} - (\underline{A} - \underline{B})\}$ add to zero. Finally, case 3 is similar to case 2 and hence we have shown the validity of the triangular inequality in Equation 4.46.

At the receiver of an RRNS coded data transmission system, the demodulator provides the received residues $\underline{z}$ in response to a transmitted codeword $\underline{x}$, which can be modelled as:

$$\underline{z} = \underline{x} + \underline{e} \tag{4.51}$$

where $\underline{e}$ is the error vector imposed by the channel, which may have an arbitrary number of non-zero components in the range between 0 and n. Each received residue can be written as follows:

$$z_i = (x_i + e_i) \bmod m_i \ , \tag{4.52}$$

where $1 \leq i \leq n$ and e_i $0 \leq e_i \leq m_i$, is the error magnitude of each residue. More explicitly, since the residue x_i may assume m_i different values, the error values are also m_i-ary. If $e_i = 0$ for all i, then the received residues are error free.

Let us denote the Hamming weight of the error vector $\underline{e}$ by α, which quantifies the number of non-zero e_i positions in the received residue vector $\underline{z}$. In an RRNS, there is no error vector $\underline{e}$, which has a weight of $0 < \alpha < d_{min}$, that can change codeword $\underline{x}$ into another valid codeword $\hat{\underline{x}}$. Since $dist(\underline{z}, \underline{x}) = \alpha$ and $0 < \alpha < d_{min}$, upon applying Equation 4.46, we can write

$$\begin{aligned} dist(\underline{z}, \hat{\underline{x}}) + dist(\underline{z}, \underline{x}) &\geq dist(\underline{x}, \hat{\underline{x}}) \\ dist(\underline{z}, \hat{\underline{x}}) &\geq d_{min} - \alpha \\ &\geq 0 \ , \end{aligned} \tag{4.53}$$

for all possible codewords $\hat{\underline{x}}$ in RRNS and $\hat{\underline{x}} \neq \underline{x}$. Therefore, $\underline{z}$ cannot be a valid codeword. Let us assume that there is another error vector $\underline{e}$, which satisfies:

$$\underline{e} = \hat{\underline{x}} - \underline{x} \ , \tag{4.54}$$

where $\hat{\underline{x}}$ is an arbitrary codeword in the RRNS satisfying $\hat{\underline{x}} \neq \underline{x}$, and $dist(\hat{\underline{x}}, \underline{x}) = d_{min}$. Then, the received residue vector $\underline{z}$ is equal to codeword $\hat{\underline{x}}$. Therefore, all error vectors having weight$(\underline{e}) = d_{min}$ are non-detectable. We can then characterise the error detection capability of a RRNS code as:

$$l = d_{min} - 1 \ , \tag{4.55}$$

which is the highest possible number of errors for which the residue vector $\underline{z}$ is not a valid codeword.

The error correction capability t of an RRNS code is defined as the highest possible number of errors that the decoder is capable of correcting. The error correction capability of an RRNS code is given by [43, 96]:

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \ , \tag{4.56}$$

where $\lfloor i \rfloor$ means the largest integer not exceeding i. Here we note that l and t in Equations 4.55 and 4.56 are identical to the corresponding quantities in RS codes. Again, let $\hat{\underline{x}}$

be a codeword other than $\underline{x}$ in the RRNS code. The Hamming distance among $\underline{x}$, $\underline{\hat{x}}$ and $\underline{z}$ satisfies the triangular inequality:

$$dist(\underline{z}, \underline{x}) + dist(\underline{z}, \underline{\hat{x}}) \geq dist(\underline{x}, \underline{\hat{x}}) \ . \tag{4.57}$$

Since $dist(\underline{x}, \underline{\hat{x}}) \geq d_{min}$ and an error vector has a weight of $dist(\underline{z}, \underline{x}) = \alpha$, we can rewrite Equation 4.57 as:

$$dist(\underline{z}, \underline{\hat{x}}) > d_{min} - \alpha \ . \tag{4.58}$$

If the weight of the error vector is:

$$\alpha \leq \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor , \tag{4.59}$$

then

$$d_{min} - \alpha > \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \geq \alpha \ . \tag{4.60}$$

Therefore, we have shown that:

$$dist(\underline{z}, \underline{x}) < dist(\underline{z}, \underline{\hat{x}}) \ , \tag{4.61}$$

which implies that the received residue vector $\underline{z}$ is closer to $\underline{x}$ than to any other valid codeword $\underline{\hat{x}}$.

4.4 Multiple Error Correction Procedure

Based on the properties of modulus projection and the MRC, an algorithm was proposed in [133] for single residue error correction. It was then used in [135, 144, 145] for correcting single errors in digital filters and error checkers. Recently, this algorithm was extended [44] for detecting and correcting multiple residue errors.

In an RRNS having n moduli, we define the following quantity:

$$M^{\alpha} = \prod_{i=1}^{\alpha} m_{j_i} \ , \tag{4.62}$$

where $1 \leq j_i \leq n$ and $\alpha \leq n$. Hence, we can define the M^{α}-projection of integer X, denoted by $X_{M^{\alpha}}$, as:

$$X_{M^{\alpha}} \equiv X \left(\mathrm{mod} \ \frac{M_n}{M^{\alpha}} \right) \ . \tag{4.63}$$

If $\frac{M_n}{M^{\alpha}} \geq M_k$, it follows from Equation 4.63 that the M^{α}-projection of any legitimate number X in the RRNS is still the same legitimate number, that is $X_{M^{\alpha}} = X$.

For an RRNS(n,k) code which has a minimum free distance d_{min}, as described in Section 4.3.1, any valid codeword $\underline{x}_i$ represents an unique integer X_i which falls in the legitimate range of the code, $0 \leq X_i < M_k$. Conversely, any invalid codewords are in the illegitimate range, $M_k \leq X_i < M_n$. It was also shown in Section 4.3.1 that any integer X' differing from X, $0 \leq X < M_k$, in at least one but no more than $d_{min} - 1$ residue digits is an illegitimate number. By using our previous arguments, we are able to detect if a set of residues $\underline{x}$ is erroneous, as long as the number of residue errors is less than or equal to $l = d_{min} - 1$.

Let us assume that no more than t number of errors occurred in a valid codeword $\underline{x}$, where t is the error correction capability of the RRNS code. The altered codeword can be represented as:

$$\begin{aligned} Z &\equiv X + E \pmod{M_n} && (4.64) \\ &\longleftrightarrow (z_1, z_2, ..., z_n) \\ &= (x_1, x_2, ..., x_n) + (0, ..., 0, e_{j_1}, 0, ..., 0, e_{j_2}, 0, ..., 0, e_{j_t}, 0, ..., 0)\ , && (4.65) \end{aligned}$$

where $X \longleftrightarrow (x_1, x_2, ..., x_n)$ and $E \longleftrightarrow (0, ..., 0, e_{j_1}, 0, ..., 0, e_{j_2}, 0, ..., 0, e_{j_t}, 0, ..., 0)$. We have $E \equiv 0 \pmod{m_j}$ for all $j \neq j_i, i = 1, 2, ..., t$. Hence E is a multiple of all moduli except $m_{j_1}, m_{j_2}, ..., m_{j_t}$ and we can write

$$\begin{aligned} E &= e\frac{M_n}{m_{j_1} m_{j_2} ... m_{j_t}} \\ &= e\frac{M_n}{M^t}\ , && (4.66) \end{aligned}$$

where $0 < e < M^t$.

Let us now substitute Equation 4.66 into Equation 4.64 and take the M^t-projection of Z,

$$\begin{aligned} Z_{M^t} &\equiv Z \left(\text{mod } \frac{M_n}{M^t}\right) \\ &\equiv X + e\frac{M_n}{M^t} \left(\text{mod } \frac{M_n}{M^t}\right) \\ &\equiv X \left(\text{mod } \frac{M_n}{M^t}\right) \\ &= X_{M^t} = X < M_k\ . && (4.67) \end{aligned}$$

Therefore, we have shown that the valid codeword $\underline{x}$ can be recovered from the received vector $\underline{z}$, even though it has been corrupted by t errors. However, we have to show that for any other combination of t moduli, $m_{j_{1'}}, m_{j_{2'}}, ..., m_{j_{t'}}$, which is denoted by $M^{t'}$ and $M^{t'} \neq M^t$, the $M^{t'}$-projection of the integer Z results in an integer $Z_M^{t'}$, which represents an invalid codeword in the illegitimate range of $M_k \leq Z_M^{t'} < M_n$. Note that the total number of moduli combinations is equal to ${}^nC_t = \frac{n!}{t!(n-t)!}$. We define the $M^{t'}$-projection of

Z, namely $Z_M^{t'}$ as its *illegitimate projection* and conversely, Z_{M^t} as its *legitimate projection.* The illegitimate projection $Z_M^{t'}$ can be treated as a number originating from $X_M^{t'}$ which was projected with the aid of the moduli $m_{j_1}, m_{j_2}, ..., m_{j_t}$. We can then express $Z_M^{t'}$ as follows:

$$\begin{aligned} Z_M^{t'} &\equiv X_M^{t'} + e\frac{M_n}{M^t M^{t'}} \left(\text{mod } \frac{M_n}{M^t}\right) \\ &= X_M^{t'} + e\frac{M_n}{\prod_{i=1}^t m_{j_i} \prod_{i=1}^t m_{j_{i'}}} \ , \end{aligned} \tag{4.68}$$

where $0 < e < \prod_{i=1}^t m_{j_{i'}}$. With the objective of quantifying the range to which an illegitimate projection of Z, namely $Z_M^{t'}$, we find the minimum of $Z_M^{t'}$. Hence we choose $X_M^{t'} = 0$, $e = 1$ and we have

$$\begin{aligned} \min\left(Z_M^{t'}\right) &= \frac{M_n}{\prod_{i=1}^t m_{j_i} \prod_{i=1}^t m_{j_{i'}}} \\ &= \frac{M_k M_{n-k}}{\max\left(\prod_{i=1}^t m_{j_i} \prod_{i=1}^t m_{j_{i'}}\right)} \\ &= \frac{M_k M_{n-k}}{M_{n-k}} \\ &= M_k \ . \end{aligned} \tag{4.69}$$

This shows that an illegitimate projection of integer Z is always larger than $M_k - 1$, $Z_M^{t'} \geq M_k$.

Let us now consider an example. We have an RRNS code based on the moduli $m_1 = 3$, $m_2 = 4$, $m_3 = 5$, $m_4 = 7$, $m_5 = 11$ and $m_6 = 13$, where m_3, m_4, m_5 and m_6 are the redundant moduli. Therefore, $n = 6$, $k = 2$ and

$$\begin{aligned} M_k &= 3 \times 4 & \qquad M_n &= 3 \times 4 \times 5 \times 7 \times 11 \times 13 \\ &= 12 & &= 60,060 \end{aligned} \ . \tag{4.70}$$

The minimum free distance of the RRNS code is $d_{min} = 5$, since by applying Equation 4.37, we found

$$\begin{aligned} 4 \times 5 \times 7 \times 11 \times 13 &> M_{n-k} \geq 5 \times 7 \times 11 \times 13 \\ 20,020 &> M_{n-k} \geq 5,005 \ . \end{aligned} \tag{4.71}$$

The values of n, k and d_{min} satisfy Equation 4.45 and hence the designed RRNS code is a maximum distance separable RRNS code. The error correction capability is $t = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor = 2$.

Let $X = 10$, which is represented in the RRNS domain by,

$$X \longleftrightarrow (1, 2, 0, 3, 10, 10) \ . \tag{4.72}$$

Since $0 \leq X < M_k = 12$, $\underline{x}$ is a valid codeword. Assume $E = 5,720$ and with the aid of Equation 4.64, we can write

$$\begin{aligned} Z &\equiv X + E \bmod (M_n) \\ &= 10 + 5720 \bmod (60060) \\ &= 5730 \\ (0,2,0,4,10,10) &\longleftrightarrow |(1,2,0,3,10,10) + (2,0,0,1,0,0)|_{(m_1,m_2,m_3,m_4,m_5,m_6)} \ . \end{aligned} \quad (4.73)$$

In order to correct the errors, their positions have to be found. Hence the integer Z is first calculated using the CRT of Section 4.2.5.1 and it is found that $Z \geq M_k$, which means that $\underline{z}$ is an invalid codeword. Errors have been detected and the error locations will have to be found in the next step. Since $n = 6$ and $t = 2$, we would have ${}^nC_t = \binom{6}{2} = 15$ different combinations of the residue error positions. Let us for example assume that the erroneous residue positions correspond to m_1 and m_2. The projection of Z using $M^{t'} = 3 \times 4 = 12$, is then,

$$\begin{aligned} Z_{M^{t'}} &\equiv Z \left(\bmod \frac{M_n}{M^{t'}}\right) \\ &= Z \left(\bmod \frac{60060}{12}\right) \\ &= 5730 \ (\bmod 5005) \\ &= 725 > M_k \ , \end{aligned} \quad (4.74)$$

which yields an illegitimate projection. Using all 15 different possible residue error moduli combinations and the steps described, we find the moduli projection of each combination. The results are shown in Table 4.4. It can be seen from the table that the moduli projection $M^{t'}$ of all moduli combinations are larger than M_k, except that corresponding to moduli m_1 and m_4. Hence, the residue positions 1 and 4 are declared to be in error. The moduli projection Z_{M^t} for moduli m_1 and m_4 is equal to 10, which is the same as integer X. The correct residues in position 1 and 4 are Z_{M^t} modulo m_1 and m_4, i.e. 10 mod 3 = 1 and 10 mod 4 = 2. Alternatively, we can apply the BEX algorithm in order to find the correct residues for both positions.

Above we have shown a simple example for correcting two residues errors. Let us now show using the same example that the same procedure can be used to correct one residue error in the RRNS. Again, let $X = 10 \longleftrightarrow (1,2,0,3,10,10)$ and assume $E = 40,040$. With the aid of Equation 4.64, we can write

$$\begin{aligned} Z &\equiv 10 + 40040 \bmod (60060) \\ &= 40050 \\ (0,2,0,3,10,10) &\longleftrightarrow (1,2,0,3,10,10) + (2,0,0,0,0,0) \ . \end{aligned} \quad (4.75)$$

j_1	j_2	m_{j_1}	m_{j_2}	$M^{t'2}$	$\frac{M_n}{M^{t'2}}$	$Z_{M^{t'}}$	j_1	j_2	m_{j_1}	m_{j_2}	$M^{t'2}$	$\frac{M_n}{M^{t'2}}$	$Z_{M^{t'}}$
1	2	3	4	12	5005	725	2	6	4	13	52	1155	1110
1	3	3	5	15	4004	1726	3	4	5	7	35	1716	582
1	**4**	**3**	**7**	**21**	**2860**	**10**	3	5	5	11	55	1092	270
1	5	3	11	33	1820	270	3	6	5	13	65	924	186
1	6	3	13	39	1540	1110	4	5	7	11	77	780	270
2	3	4	5	20	3003	2727	4	6	7	13	91	660	450
2	4	4	7	28	2145	1440	5	6	11	13	143	420	270
2	5	4	11	44	1365	270	-	-	-	-	-	-	-

Table 4.4: Results of the 15 different moduli projections of integer $Z = 5730$

During the first step of the procedure we calculate the integer Z, which is found to be in error. In order to locate the errors, 15 different moduli combinations are to be considered for finding the projection of Z. The corresponding results are shown in Table 4.5. From the table, we can see that there is more than one moduli combination which results in moduli projection $Z_{M^t} < M_k = 12$. Indeed, the moduli projections $Z_{M^t} < M_k = 12$ are all the same integers, namely 10. Actually, all moduli combinations which include modulus m_1 will produce a moduli projection of $Z_{M^t} = 10 < M_k = 12$. Hence, the procedure used to correct t residue errors can be applied to correct less than t errors as well.

j_1	j_2	m_{j_1}	m_{j_2}	$M^{t'2}$	$\frac{M_n}{M^{t'2}}$	$Z_{M^{t'}}$	j_1	j_2	m_{j_1}	m_{j_2}	$M^{t'2}$	$\frac{M_n}{M^{t'2}}$	$Z_{M^{t'}}$
1	**2**	**3**	**4**	**12**	**5005**	**10**	2	6	4	13	52	1155	780
1	**3**	**3**	**5**	**15**	**4004**	**10**	3	4	5	7	35	1716	582
1	**4**	**3**	**7**	**21**	**2860**	**10**	3	5	5	11	55	1092	738
1	**5**	**3**	**11**	**33**	**1820**	**10**	3	6	5	13	65	924	318
1	**6**	**3**	**13**	**39**	**1540**	**10**	4	5	7	11	77	780	270
2	3	4	5	20	3003	1011	4	6	7	13	91	660	450
2	4	4	7	28	2145	1440	5	6	11	13	143	420	150
2	5	4	11	44	1365	465	-	-	-	-	-	-	-

Table 4.5: Results of the 15 different moduli projections of integer $Z = 40050$

In the previous two examples we have shown that the multiple errors correction procedure can be accomplished by using the CRT. However, if the number of moduli is increased in an effort to create larger and/or stronger RRNS codes, the range M_k and M_n increases as well, which may lead to overflows. Therefore, special procedures have to be used for carrying out large integer operations, such as those involved in the CRT. This will impose extra complexity upon the decoding algorithm. In order to avoid large integer operations, the MRC was used by a number of authors [44, 133, 135] for correcting single or multiple errors in RRNS codes.

In Section 4.2.7, we explained that for any integer $X < M_k$ represented in the mixed radix form, as in Equation 4.32, the redundant mixed radix digits, $a_{k+j}, j = 1, 2, ..., n-k$, will be equal to zero. Hence, we can apply the MRC to a given residue representation $\underline{z}$ in order to generate the redundant residues. By checking the redundant mixed radix digits, we are able to tell whether $Z < M_k$, i.e. whether $\underline{z}$ is a valid codeword. If $Z \geq M_k$, moduli projection can be used to locate the errors. In Equation 4.63, we defined the moduli projection of an integer. It can be shown that the M^t-projection of Z can also be represented as a reduced residue representation of Z with the residues $z_{j_1}, z_{j_2}, ..., z_{j_\alpha}$ deleted. Using the reduced residue representation, we are able to find the mixed radix representation of Z_{M^t}, as follows [41, 43, 44]:

$$Z_{M^t} = \sum_{\substack{i=1 \\ i \neq j_1, j_2, ..., j_t}}^{n} a_i \prod_{\substack{r=1 \\ r \neq j_1, j_2, ..., j_t}}^{i-1} m_r \ , \tag{4.76}$$

which reflects the structure of the MRC definition in Equation 4.21. The legitimate and illegitimate range of the reduced RRNS are $[0, M_k' - 1]$ and $[M_k', \frac{M_n}{M^t} - 1]$, respectively. Let us assume that $j_1 < j_2 < ... < j_r < ... < j_t$ and $j_r \leq k$. We can then specify the legitimate range of the reduced RRNS separately for the specific cases, where the dropped moduli are from the set of non-redundant moduli, i.e. from the range $j_1 \leq k$, or where some of the redundant moduli may have been dropped. This is expressed explicitly as:

$$\begin{aligned} M_k' &= \prod_{\substack{i=1 \\ i \neq j_1, j_2, ..., j_r}}^{k+r} m_i, & j_1 \leq k \\ M_k' &= M_k = \prod_{i=1}^{k} m_i, & j_1 > k \ . \end{aligned} \tag{4.77}$$

We can see from the equation that $M_k' \geq M_k$ and $M_k' - 1$ is the highest integer that can be represented by Equation 4.76 with all the redundant mixed radix digits set to zero. Therefore, even if all the redundant mixed radix digits are zero, one can still argue that any legitimate projection Z_{M^t} can be larger than M_k since $M_k' \geq M_k$. However, we are able to show that any illegitimate projection of Z will result in $Z_{M^{t'}} \geq M_k'$. In order to find the minimum of $Z_M^{t'}$, we rewrite Equation 4.68 as:

$$\begin{aligned} Z_M^{t'} &= X_M^{t'} + e\frac{M_k M_{n-k}}{m_{j_{1'}} ... m_{j_{r'}} ... m_{j_{t'}} \prod_{i=1}^{t} m_{j_i}} \\ &= X_M^{t'} + e\frac{M_k}{m_{j_{1'}} ... m_{j_{r'}}} \frac{M_{n-k}}{m_{j_{r+1'}} ... m_{j_{t'}} \prod_{i=1}^{t} m_{j_i}} \end{aligned} \tag{4.78}$$

Since $n-k \geq t+t' = 2t$, we have

$$\frac{M_{n-k}}{m_{j_{r+1'}}...m_{j_{t'}}\prod_{i=1}^{t} m_{j_i}} \geq m_{k+1}...m_{k+r} \ . \tag{4.79}$$

Following from Equation 4.79, we can then write

$$\begin{aligned} Z_M^{t'} &\geq X_M^{t'} + e\frac{M_k}{m_{j_{1'}}...m_{j_{r'}}} m_{k+1}...m_{k+r} \\ &= X_M^{t'} + eM_k' \ , \end{aligned} \tag{4.80}$$

where M_k' is the legitimate range of the reduced RRNS. Clearly, in Equation 4.80, any illegitimate projection of Z always results in $Z_M^{t'} > M_k'$. Hence, if not all the redundant mixed radix digits are equal to zero, the projection is illegitimate and vice versa. Once a legitimate projection is found, we can apply BEX to find the correct residues for the corresponding positions.

In order to augment our previous discussions let us explain in detail the multiple error correction procedure employing MRC, using the above example. Previously, we have used moduli $m_1 = 3$, $m_2 = 4$, $m_3 = 5$, $m_4 = 7$, $m_5 = 11$ and $m_6 = 13$. The total number of moduli is $n = 6$ and the number of information moduli is $k = 2$. Therefore, we have $M_k = 12$ and $M_n = 60,060$. It was shown in Equation 4.71 that the RRNS code has $d_{min} = 5$. Again, we have an integer message of $X = 10$, which has been corrupted by $E = 5,520$ and became Z given by:

$$\begin{aligned} Z &\equiv X + E \bmod (M_n) \\ &= 5730 \\ (0,2,0,4,10,10) &\longleftrightarrow |(1,2,0,3,10,10) + (2,0,0,1,0,0)|_{(m_1,m_2,m_3,m_4,m_5,m_6)} \ . \end{aligned} \tag{4.81}$$

We first assume that the erroneous residue positions correspond to m_1 and m_2. Then we find a reduced representation of Z with the residues z_1 and z_2 deleted. Since the moduli m_1 and m_2 are deleted, we have a new RRNS based on the moduli $m_1 = 5$, $m_2 = 7$, $m_3 = 11$ and $m_4 = 13$, where m_3 and m_4 are the redundant moduli and hence $M_k' = 5 \times 7 = 35$. Using the reduced residue representation, we can express the mixed radix representation of the reduced representation of Z with the residues z_1 and z_2 deleted, using the $M^{t'}$-projection of Z $Z_{M^{t'}}$ as follows:

$$\begin{aligned} Z_{M^{t'}} &= a_5 m_1 m_2 m_3 m_4 + a_4 m_1 m_2 m_3 + a_3 m_1 m_2 + a_2 m_1 + a_1 \\ &= 5005a_5 + 385a_4 + 35a_3 + 5a_2 + a_1 \ . \end{aligned} \tag{4.82}$$

Employing the procedures outlined in Section 4.2.5.2, we are able to calculate the values of a_i $i = 1, 2, .., 5$ given the residues z_3, z_4, z_5 and z_6. However, for the reader's convenience, we can assume that we have $Z_{M^{t'}} = 725$ and hence rewrite Equation 4.82 as:

$$725 = 5005a_5 + 385a_4 + 35a_3 + 5a_2 + a_1 \ . \tag{4.83}$$

Using Equation 4.83, we readily find the values $a_1 = 0$, $a_2 = 5$, $a_3 = 9$, $a_4 = 1$ and $a_5 = 0$. Since $a_3 \neq a_4 \neq a_5 \neq 0$, we know that:

$$\begin{aligned} Z_{M^{t'}} &> M'_k \\ 725 &> 35 \ . \end{aligned} \tag{4.84}$$

We can therefore conclude that $Z_{M^{t'}}$ is an illegitimate projection.

Let us now delete another set of moduli, namely m_1 and m_4 from the original set of moduli given by $m_1 = 3$, $m_2 = 4$, $m_3 = 5$, $m_4 = 7$, $m_5 = 11$ and $m_6 = 13$. Hence the original RRNS was reduced to a new set of RRNS based on the moduli $m_1 = 4$, $m_2 = 5$, $m_3 = 11$ and $m_4 = 13$, where m_3 and m_4 are the redundant moduli and the dynamic range is given by $M'_k = 4 \times 5 = 20$. Using the reduced residue representation, we can express the mixed radix representation of the reduced representation of Z with the residues z_1 and z_4 deleted, using the $M^{t'}$-projection of Z, $Z_{M^{t'}}$ as follows:

$$\begin{aligned} Z_{M^{t'}} &= a_5m_1m_2m_3m_4 + a_4m_1m_2m_3 + a_3m_1m_2 + a_2m_1 + a_1 \\ &= 2860a_5 + 220a_4 + 20a_3 + 4a_2 + a_1 \ . \end{aligned} \tag{4.85}$$

Given the residues z_1, z_2, z_3 and z_4, we can apply the somewhat tedious procedures of Section 4.2.5.2 for calculating the values of a_i $i = 1, 2, .., 5$. Again, for the reader's convenience we assume that we have $Z_{M^{t'}} = 10$ and hence we arrive at:

$$10 = 2860a_5 + 220a_4 + 20a_3 + 4a_2 + a_1 \ . \tag{4.86}$$

Using Equation 4.86, we calculated $a_1 = 2$, $a_2 = 2$, $a_3 = 0$, $a_4 = 0$ and $a_5 = 0$. Since $a_3 = a_4 = a_5 = 0$, $Z_{M^{t'}}$ is a legitimate projection. Once the legitimate projection is found, we can apply the BEX algorithm for finding the correct residues for the corresponding positions, namely for positions 1 and 4.

The flow-chart of multiple error correction procedures using the MRC method is shown in Figure 4.1. Initially, the corrupted residues $\underline{z}$ are received and MRC is applied. The redundant residues $a_{k+1}, ..., a_n$ are checked whether they are all zeros. If $a_{k+1} = a_{k+2} = ... = a_n = 0$, the received residues $\underline{z}$ are declared error free. If not all of them are zero, a set of t moduli is generated in an effort to find up to t error positions. MRC is applied to the reduced residue representation, where the set of t chosen moduli is deleted. The redundant mixed radix digits are checked, whether all the redundant mixed radix digits are zero. If so,

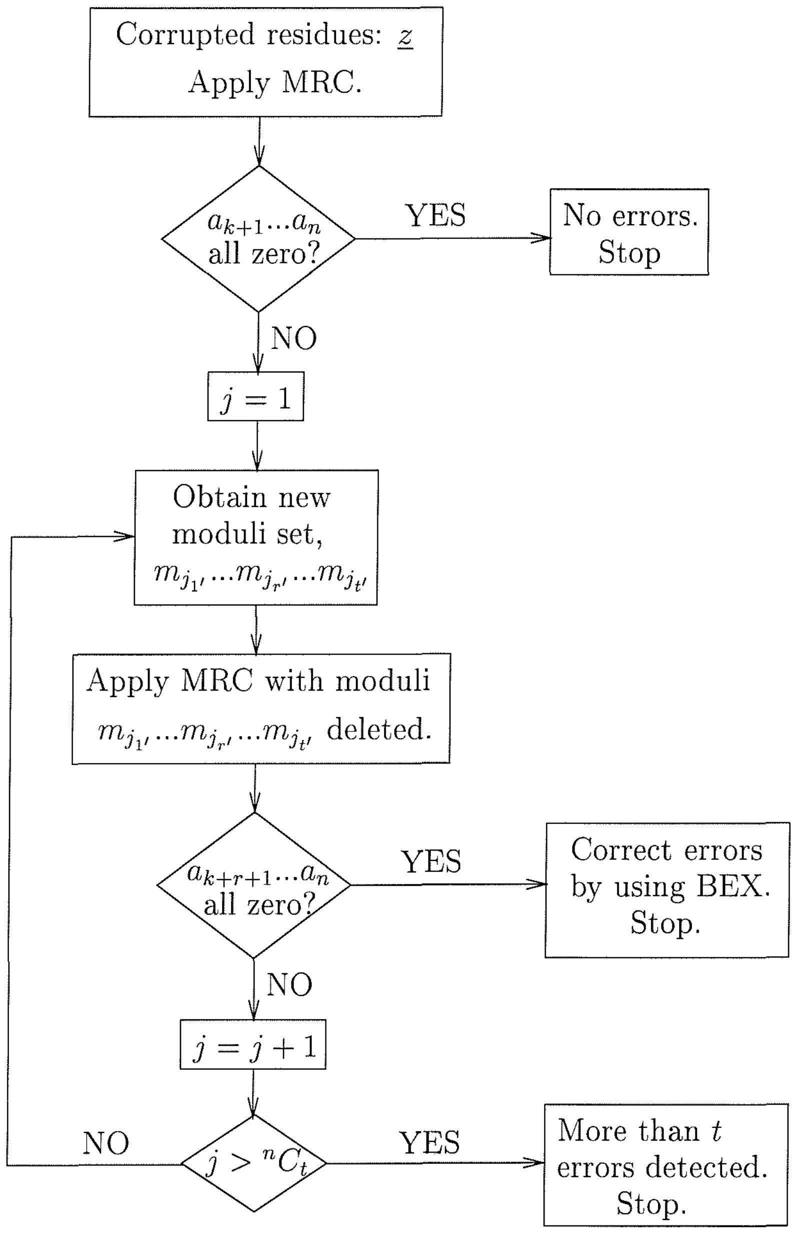

Corrupted residues: $\underline{z}$
Apply MRC.
$a_{k+1}...a_n$ all zero?
YES
No errors. Stop
NO
$j = 1$
Obtain new moduli set, $m_{j_{1'}}...m_{j_{r'}}...m_{j_{t'}}$
Apply MRC with moduli $m_{j_{1'}}...m_{j_{r'}}...m_{j_{t'}}$ deleted.
$a_{k+r+1}...a_n$ all zero?
YES
Correct errors by using BEX. Stop.
NO
$j = j + 1$
NO
$j > {}^nC_t$
YES
More than t errors detected. Stop.

Figure 4.1: Flow-chart of multiple error correction RRNS decoding.

the error positions have been found and the errors can be corrected using the BEX. If either of the redundant mixed radix digits is non-zero, another set of t moduli is obtained and the MRC procedure is repeated again. The above process is repeated, until all possible set of t moduli combinations have been tested and hence the received residue vector is declared to have more than t errors.

4.5 RRNS Encoder

In the previous section we stated that an RRNS code used is constituted by a set of residues with respect to a pre-defined set of moduli. Since the moduli and the residues can assume any positive integer value - representing an arbitrary number of binary bits - the RRNS code is a non-binary code, based on transmitting the residues conveying a number of bits. In this section, we propose two different mapping methods transforming the binary source bits to the non-binary RRNS code, which result in a so-called non-systematic and systematic RRNS code.

4.5.1 Non-systematic RRNS Code

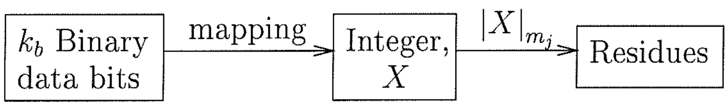


Figure 4.2: Non-systematic encoding procedures.

Here we commence by summarising the non-systematic encoding process of Figure 4.2. The non-systematic encoder encodes k_b number of binary data bits per RRNS(n, k) codeword, where the integer 2^{k_b} must not be higher than the legitimate range M_k, hence:

$$2^{k_b} \le M_k \; . \tag{4.87}$$

In other words, the data bits are mapped to an integer X, which has to be in the range of $[0, 2^{k_b} - 1]$. Note that the full legitimate range M_k of the RRNS may not be actively exploited, since M_k is typically not an integer power of 2. Considering now the mapping of the integer X to residues for transmission in Figure 4.2 and using the moduli in the RRNS, the residues x_j are simply obtained by invoking the conventional modulus operation:

$$x_j = |X|_{m_j} \; , \tag{4.88}$$

where $j = 1, 2, ..., n$. In order to represent an integer X in the RRNS seen in Figure 4.2, each residue x_j, non-redundant or redundant, has to be represented uniquely for transmission in terms of bits. Therefore, we have to ensure that

$$2^{n_{bj}} \geq m_j \ , \tag{4.89}$$

where $j = 1, 2, ..., n$ and n_{bj} is the number of bits representing the residue x_j in Figure 4.2. The total number of coded bits per RRNS(n, k) codeword is therefore:

$$n_b = \sum_{j=1}^{n} n_{bj} \ . \tag{4.90}$$

The rate R of the code is then $\frac{k_b}{n_b}$. Since the residues $x_j, j = 1, ..., k$ do not directly represent the binary data bits, we refer to the above encoding process as non-systematic encoding.

Let us for example consider an RRNS based on the moduli $m_1 = 53$, $m_2 = 55$, $m_3 = 59$, $m_4 = 61$, $m_5 = 63$ and $m_6 = 64$, where m_5 and m_6 are the redundant moduli. We have $n = 6$, $k = 4$ and

$$\begin{aligned} M_k &= 53 \times 55 \times 59 \times 61 \\ &= 10,491,085 \ , \end{aligned} \tag{4.91}$$

where M_k is the legitimate dynamic range of the RRNS. In this case, we can represent an integer from 0 to $10,491,084$ uniquely by the RRNS. Since $8,388,608 = 2^{23} < M_k < 2^{24} = 16,777,216$, $k_b = 23$ is the number of the binary data bits encoded by the RRNS code. Hence, the dynamic range of the RRNS is not fully utilised. Applying Equation 4.89, we calculate the number of coded bits by taking into account that each of the residues requires six bits for its unique representation, yielding:

$$\begin{aligned} n_b &= 6 + 6 + 6 + 6 + 6 + 6 \\ &= 36 \ . \end{aligned} \tag{4.92}$$

Therefore, the code rate is,

$$\begin{aligned} R &= \frac{k_b}{n_b} \\ &= \frac{23}{36} = 0.639 \end{aligned} \tag{4.93}$$

4.5.2 Systematic RRNS Code

In contrast to the non-systematic encoder of Figure 4.2, Figure 4.3 characterises the systematic encoding process. Unlike the non-systematic encoder, which maps all the data bits

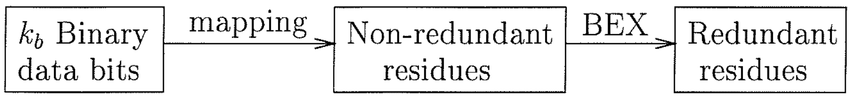


Figure 4.3: Systematic encoding procedures.

to be transmitted to a single integer X, the systematic encoder divides the bit sequence to be encoded into shorter groups of bits, each of which represents a non-redundant residue x_j. In contrast to the non-systematic mapping of Equation 4.87, in order to render this systematic mapping unique, the number of bits k_{b_j} mapped to residue x_j has to satisfy:

$$2^{k_{b_j}} < m_j \ , \tag{4.94}$$

The total number of data bits that the systematic encoder encodes into each RRNS codeword becomes:

$$k_b = \sum_{j=1}^{k} k_{b_j} \ . \tag{4.95}$$

Accordingly, as shown in Figure 4.3, the data bit sequences are mapped to the non-redundant residues directly. Then the so-called BEX algorithm of Section 4.2.7 can be invoked, in order to compute the redundant residues. Similarly to the non-systematic encoder, the number of bits needed to represent the redundant residues has to satisfy Equation 4.89 for $j = k+1, ..., n$. The number of bits required for the unique representation of the non-redundant residues has been specified in Equation 4.94. Hence, we can write the number of coded bits n_{b_j} for each residue x_j as:

$$n_{b_j} = \begin{cases} 2^{k_{b_j}} < m_j & j = 1, 2, ..., k \\ 2^{n_{b_j}} \geq m_j & j = k+1, ..., n \end{cases} \ . \tag{4.96}$$

The total number of coded bits can then be calculated using Equation 4.90.

We consider again the same moduli set, namely $m_1 = 53$, $m_2 = 55$, $m_3 = 59$, $m_4 = 61$, $m_5 = 63$ and $m_6 = 64$, as for our non-systematic coding example in Section 4.5.1. Applying Equation 4.94 and 4.95, the total number of data bits is:

$$k_b = 5 + 5 + 5 + 5 = 20 \ . \tag{4.97}$$

The number of coded bits, calculated using Equation 4.96, is then:

$$n_b = 5 + 5 + 5 + 5 + 6 + 6 = 32 \ . \tag{4.98}$$

The code rate is $R = \frac{k_b}{n_b} = \frac{20}{32} = 0.625$. If we compare the code rate of the non-systematic and systematic encoders, we can see that the code rate of the systematic encoder is lower than that of the non-systematic encoder. Furthermore, the dynamic range of the RRNS code is only $\left(2^5\right)^4 = 1,048,576$ as compared to the corresponding range of $8,388,608$ for the non-systematic code. In order to increase the code rate and the dynamic range of the systematic RRNS encoder, we propose a more efficient mapping method, which is outlined in the next section.

4.5.2.1 Modified Systematic RRNS Code

As mentioned earlier, the legitimate dynamic range of a systematic encoder of Figure 4.3 is more limited than that of the non-systematic encoder seen in Figure 4.2. This also causes a reduction in the code rate compared to that of the non-systematic encoder. Here, we propose a modification to the mapping method used in the systematic encoder of the previous section. Instead of using Equation 4.94, the number of binary data bits mapped to each non-redundant residue is now increased by one, yielding:

$$2^{k_{b_j}} \geq m_j \ . \tag{4.99}$$

However, as a consequence of the new allocation of data bits, there may exist integers X, which are equal to or greater than the modulus, i.e. $X \geq m_j$. Hence, we define the new mapping method as follows:

$$x_j = \begin{cases} X & \text{if } X < m_j \\ 2^{k_{b_j}} - 1 - X & \text{if } X \geq m_j \end{cases} , \tag{4.100}$$

where k_{b_j} is the number of data bits mapped to the integer X for transmission in Figure 4.3. The mapping in the second line of Equation 4.100 has to be implemented on the basis of bitwise complement, if $X \geq m_j$. Although this implies that the mapping to the integers is ambiguous for some of the values, Equation 4.100 ensures the maximum Hamming distance separation of the ambiguous values. We will show in the following that this maximum Hamming distance separation allows the decoder to recognise the original integer messages.

We use the modulus $m_4 = 61$ as an example for further illustration. In the previous section, only 5 bits were assigned to this residue. Since $2^5 = 32$, $0, 1, ..., 31$ are the possible residues. Hence, the previous bit assignment policy does not fully utilise the dynamic range of the modulus $m_4 = 61$, which is $0, 1, ..., 60$. Applying Equation 4.99, we now map 6 bits to the residue of modulus $m_4 = 61$ and $0, 1, ..., 63$ are the possible integers. However, integers $61, ..., 63$ are not valid residues. By applying Equation 4.100, we map the integers $61, 62, 63$

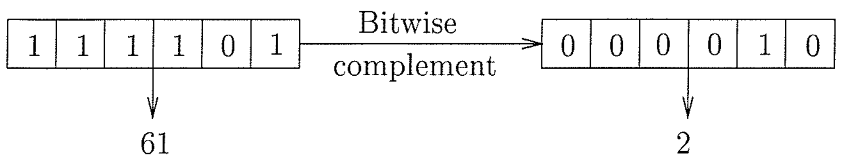


Figure 4.4: An example of modified systematic mapping.

to the integers 2, 1, 0, respectively. Figure 4.4 shows the mapping of integer 61 to 2 by implementing its bitwise complement.

Using the same moduli set as in the previous two sections, and applying Equation 4.99 and 4.89, we arrive at $k_b = 6 + 6 + 6 + 6 = 24$ and $n_b = 6 + 6 + 6 + 6 + 6 + 6 = 36$. Consequently, the code rate is now $R = \frac{24}{36} = 0.667$, which is more than in the previous systematic and non-systematic encoding cases. Furthermore, the dynamic range of the systematic RRNS code was increased from 1,048,576 to $(2^6)^4 = 16,777,216$, which is more than $M_k = 10,491,085$.

4.6 RRNS Decoder

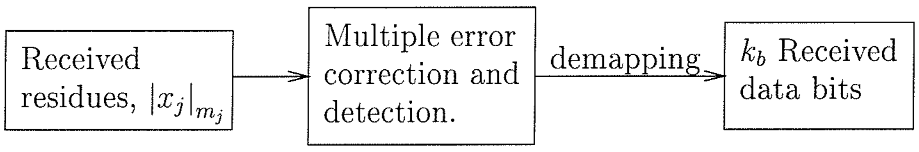

Received residues, $|x_j|_{m_j}$
Multiple error correction and detection.
demapping
k_b Received data bits

Figure 4.5: A block diagram of the RRNS decoder.

The structure of the systematic and non-systematic RRNS decoders is similar. Figure 4.5 shows the simplified block diagram of the RRNS decoder. The modulo of the received residues, $|x_j|_{m_j}$, is taken, where $j = k + 1, ..., n$ for the systematic RRNS decoder and $j = 1, ..., n$ for the non-systematic RRNS decoder. Multiple residue error correction and detection is invoked, as described in Section 4.4, for correcting a maximum of t errors in the codeword. Then, the residues are de-mapped to the original k_b data bits according to the mapping method of the systematic and non-systematic RRNS encoders, respectively.

For the modified systematic RRNS code of Section 4.5.2.1, the decoder structure needs an extra step, as shown in Figure 4.6. Due to the potentially ambiguous mapping defined in Equation 4.100, a residue x_j may represent two integers, which exhibit maximum Hamming distance separation. Hence, after the multiple error correction and detection procedures

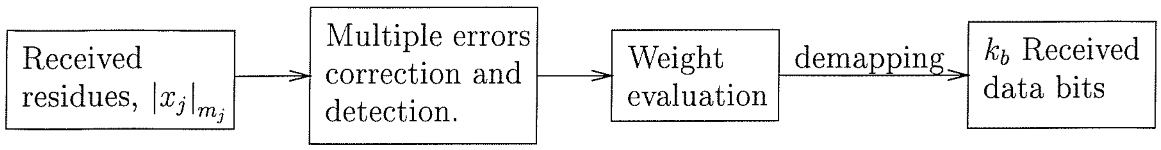

Received residues, $|x_j|_{m_j}$
Multiple errors correction and detection.
Weight evaluation
demapping
k_b Received data bits

Figure 4.6: A block diagram of the modified systematic RRNS decoder.

of Section 4.4, we have to determine which integer has to be used for extracting the k_{b_j} transmitted data bits. Soft decision of the received bits y_{ji} can be used for calculating their decision metric W, which in turn determines the integer message transmitted. The soft decision metric W of the residue x_j is calculated as follows:

$$W = \sum_{i=1}^{k_{b_j}} \text{assign}(x_{ji}) \times y_{ji} \ , \tag{4.101}$$

where

$$\text{assign}(x_{ji}) = \begin{cases} +1 & \text{if } x_{ji} = 1 \\ -1 & \text{if } x_{ji} = 0 \end{cases} . \tag{4.102}$$

Depending on the polarity of the soft decision metric W, the transmitted integer is then inferred from:

$$X = \begin{cases} x_j & \text{if } W \geq 0.0 \\ 2^{k_{b_j}} - 1 - x_j & \text{if } W < 0.0 \end{cases} , \tag{4.103}$$

which is the reverse operation of Equation 4.100.

Again, we use the modulus $m_4 = 61$ as our example in augmenting the associated operations. After the multiple residue error correction and detection procedures, residue x_4 is found to be $x_4 = 2$. From Equation 4.103 and Figure 4.4 we know that there are two possible transmitted integers X associated with residue $x_4 = 2$. Therefore, soft decision of the received bits y_{ji} can be used for determining the transmitted integer by exploiting the maximum Hamming distance separation of the associated residues. Explicitly, let us assume that the received soft decision bits are $+1.2, \ +1.5, \ -0.3, \ +0.8, \ -1.0, \ +0.2$. By applying Equation 4.101, we calculate the soft decision metric as follows:

$$W = -1.2 - 1.5 + 0.3 - 0.8 - 1.0 - 0.2 = -4.4 < 0.0 \ . \tag{4.104}$$

Therefore, with the aid of Equation 4.103 the transmitted integer is decided to be $X = 2^6 - 1 - 2 = 61$. By contrast, for example the soft decision values of $W \geq 0.0$ lead to a decision of $X = 2$.

4.7 Soft Input and Soft Output RRNS Decoder

If the output of the hard-decision based demodulator is binary, the RRNS decoder is incapable of exploiting the potential advantages accruing from the soft outputs. In this section we contrive the soft decoding of RRNS codes by combining the classic Chase algorithm [28] of Section 2.4 with the hard-decision based RRNS decoder. Hence only a brief account of the Chase algorithm will be given, since a detailed exposure to the Chase algorithm has been provided in Section 2.4. Recently, Pyndiah *et al.* [62,81,146,147] extended the Chase algorithm so that it became capable of providing soft outputs of the decoded bits. This modified algorithm was then invoked for RRNS codes [107] in order to provide the soft output of the decoded bits. Consequently, we contrived the Soft Input Soft Output (SISO) RRNS decoder. This advance allowed us to employ RRNS codes as component codes in turbo codes.

4.7.1 Soft Input RRNS Decoder

Let us consider the transmission of block coded binary symbols $\{-1,+1\}$ using BPSK modulation over an AWGN channel. At the receiver, the demodulator provides the soft decision values $\underline{y}$ for the RRNS decoder. A maximum-likelihood decoder is capable of finding the codeword that satisfies:

$$\min_{m} \text{weight}(|\underline{y} - \underline{x}_j|^2) \ , \tag{4.105}$$

where $x_{ji} \in \{-1,+1\}$ are the transmitted binary or bit representations of the RRNS coded symbols and the range of j is over all possible legitimate RRNS codewords. The decision given by Equation 4.105 is optimum in the minimum BER sense, but the associated computational complexity increases exponentially with k and becomes prohibitive for block codes with $k > 6$. As a remedy, the reduced-complexity Chase algorithm [28] can be invoked for near maximum-likelihood decoding of block codes. Again, the algorithm is sub-optimum, but it offers a significantly reduced complexity.

As it was already highlighted in Section 2.4 in the context of binary BCH codes, at the demodulator, the soft decision outputs $\underline{y}$ are subjected to a tentative hard decision, yielding the binary sequence $\underline{z}$ and the associated soft decision confidence values $|\underline{y}|$ are fed to the Chase algorithm. The tentative binary sequence $\underline{z}$ is perturbed with the aid of a set of test patterns TP, which are also binary sequences that contain binary 1s in the bit positions that are to be tentatively inverted. By adding this test pattern, modulo two, to the tentative

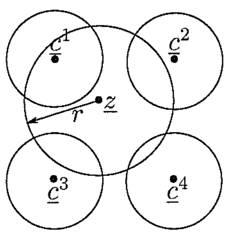


Figure 4.7: Simple coding-space illustration of the Chase algorithm.

binary sequence $\underline{z}$ a new sequence $\underline{z}'$ is obtained, where:

$$\underline{z}' = \underline{z} \oplus TP \ . \tag{4.106}$$

Using the different test patterns, the perturbed received sequence $\underline{z}'$ falls within the decoding sphere of a number of different valid codewords, namely in that of $\underline{c}^1 \ldots \underline{c}^4$ for example in Figure 4.7. In the figure, r represents the maximum Hamming distance of the perturbed binary sequence $\underline{z}'$ from the original tentative binary sequence $\underline{z}$. Hence the value of r can be adjusted by varying the maximum Hamming weight of the TPs. If we increase r, the perturbed received sequence $\underline{z}'$ will fall within the decoding sphere of more valid RRNS codewords. In order to reduce the associated implementational complexity, typically only a small set of l bit positions associated with the least reliable confidence values $|\underline{y}|$ is perturbed. The number of test patterns for which tentative decoding is invoked is then equal to 2^l.

Specifically, if the perturbed received sequence $\underline{z}'$ falls within the decoding sphere of a valid codeword, a new error pattern $\underline{e}'$ is obtained with the aid of a tentative hard decision RRNS decoding , which may be an all-zero or a non-zero tuple. Explicitly, the resultant error pattern is an all-zero tuple, if the original hard decision was erroneous, but the corruption by the TP in the low-reliability bit positions succeeded in bringing the hard decision based sequence $\underline{z}$ within the decoding sphere of the original transmitted codeword. By contrast, the resultant error pattern is a non-zero tuple, if the tentative corruption of the received codewords failed to move it into the decoding sphere of the original transmitted codeword. Instead, TP may have corrupted the received hard decision based codeword into the decoding sphere of another legitimate codewords. In this case the minimum number of decoding errors is d_{min}. The actual error pattern $\underline{e}$ associated with the received sequence $\underline{z}$ is given by

$$\underline{e} = \underline{e}' \oplus TP, \tag{4.107}$$

which may or may not be different from the original test pattern TP, depending on whether or not the perturbed binary sequence $\underline{z}'$ falls into the decoding sphere of a valid codeword.

However, only those perturbed binary sequences $\underline{z}'$ are tentatively RRNS decoded that fall into the decoding sphere of a valid codeword. Those $\underline{z}'$ binary sequences that do not fall within a legitimate decoding sphere cannot lead to a legitimate RRNS codeword and hence are discarded from our further procedures. More specifically, we derive the error patter $\underline{e}'$ for all perturbed binary sequence $\underline{z}'$ within the decoding sphere of a valid RRNS codeword and find that particular one, which is the most likely transmitted one, since it is the closest one to the received binary codeword $\underline{z}$. In this case, we are concerned with finding the error pattern $\underline{e}$ of minimum 'analogue weight', where the analogue weight of an error sequence $\underline{e}$ is defined as:

$$W(\underline{e}) = \sum_{i=1}^{n} e_i|y_i| \ . \tag{4.108}$$

The generated test pattern TP will be stored, if the associated analogue weight W is found to be lower, than the previously registered analogue weights associated with the other TPs. The above procedure will be repeated for the maximum number of test patterns, which is tolerable in complexity terms. Upon completing this loop, the memory is checked in order to determine, whether any error pattern has been stored, and if so, the corrected decoded sequence will be $\underline{z} \oplus \underline{e}$. Otherwise, the decoded sequence is the same as the received sequence $\underline{z}$.

4.7.2 Soft Output RRNS Decoder

In the previous section, we have highlighted the philosophy of soft decoding RRNS codes. Following the philosophy of Section 3.3, which was cast in the context of turbo BCH codes, with the aim of contriving the turbo RRNS code, here let us now determine the Log Likelihood Ratio (LLR) of each decoded bit u_k, given that the demodulator's soft output sequence is $\underline{y}$. This is equivalent to finding the LLR of

$$L(u_k|\underline{y}) = \ln \frac{P(u_k = +1|\underline{y})}{P(u_k = -1|\underline{y})} \ , \tag{4.109}$$

where k is a bit position in a RRNS codeword. Since the probability of $u_k = +1$ is equal to the sum of all the probabilities of all codewords $\underline{x}_i$, which have $u_k = +1$, we can rewrite the numerator of Equation 4.109 as follows:

$$P(u_k = +1|\underline{y}) = \sum_{\underline{x}_i \in \alpha^{+k}} P(\underline{x}_i|\underline{y}) \ , \tag{4.110}$$

where α^{+k} is the set of codewords $\underline{x}_i$ such that $u_k = +1$. By applying Bayes' rules, which was detailed in Section 3.3.3, we can rewrite Equation 4.110 as:

$$P(u_k = +1|\underline{y}) = \sum_{\underline{x}_i \in \alpha^{+k}} \frac{P(\underline{y}|\underline{x}_i)P(\underline{x}_i)}{P(\underline{y})} . \tag{4.111}$$

Similarly, the probability of $u_k = -1$ is equal to the sum of all the probabilities of all codewords $\underline{x}_i$, which have $u_k = -1$. Hence, the denominator of Equation 4.109 can be written as:

$$\begin{aligned} P(u_k = -1|\underline{y}) &= \sum_{\underline{x}_i \in \alpha^{-k}} P(\underline{x}_i|\underline{y}) \\ &= \sum_{\underline{x}_i \in \alpha^{-k}} \frac{P(\underline{y}|\underline{x}_i)P(\underline{x}_i)}{P(\underline{y})} , \end{aligned} \tag{4.112}$$

where α^{-k} is the set of codewords $\underline{x}_i$ such that $u_k = -1$.

Substituting Equation 4.111 and 4.112 into Equation 4.109, and assuming that all codewords are equi-probable, we arrive at:

$$\begin{aligned} L(u_k|\underline{y}) &= \ln \frac{\sum_{\underline{x}_i \in \alpha^{+k}} \frac{P(\underline{y}|\underline{x}_i)P(\underline{x}_i)}{P(\underline{y})}}{\sum_{\underline{x}_i \in \alpha^{-k}} \frac{P(\underline{y}|\underline{x}_i)P(\underline{x}_i)}{P(\underline{y})}} \\ &= \ln \frac{\sum_{\underline{x}_i \in \alpha^{+k}} P(\underline{y}|\underline{x}_i)P(\underline{x}_i)}{\sum_{\underline{x}_i \in \alpha^{-k}} P(\underline{y}|\underline{x}_i)P(\underline{x}_i)} . \end{aligned} \tag{4.113}$$

Let us assume that the transmitted bit x_k has been sent over an AWGN channel using BPSK modulation. Then, as we have seen in Section 3.3.2 in the context of turbo BCH codes, the probability density function of the demodulator soft output y_k conditioned on the transmitted bit x_k can be expressed as:

$$P(\underline{y}|\underline{x}) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \exp^{-\frac{E_B}{2\sigma^2}|\underline{y}-a\underline{x}|^2} \tag{4.114}$$

where,

n	is the number of coded bits which is equal to n_b in RRNS code
σ^2	is the noise variance
E_B	is the energy per bit
a	is the fading amplitude (=1 for a non-fading AWGN channel).

Since the probability of a specific codeword, $P(\underline{x})$, is equal to the product of all probabilities of its constituent coded bits x_j, $j = 1, 2, ..., n$, we can then write

$$\begin{aligned} P(\underline{x}) &= P(x_1)P(x_2)...P(x_n) \\ &= C\exp^{x_1\frac{L(x_1)}{2}}C\exp^{x_2\frac{L(x_2)}{2}}...C\exp^{x_n\frac{L(x_n)}{2}} \\ &= C^n\exp^{\underline{x}\frac{L(\underline{x})}{2}} , \end{aligned} \tag{4.115}$$

where $P(x_j) = C\exp^{x_j\frac{L(x_j)}{2}}$ and C is a constant, which will be cancelled out in Equation 4.113. The derivation of $P(x_j)$ can be found in Section 3.3.1, where the same problem was cast in the context of binary turbo BCH codes.

Using Equation 4.114 and 4.115, we can rewrite Equation 4.113 as:

$$L(u_k|\underline{y}) = \ln\frac{\sum_{\underline{x}_i\in\alpha^{+k}}\exp^{-\frac{E_B}{2\sigma^2}|\underline{y}-a\underline{x}_i|^2}\exp^{\underline{x}_i\frac{L(\underline{x}_i)}{2}}}{\sum_{\underline{x}_i\in\alpha^{-k}}\exp^{-\frac{E_B}{2\sigma^2}|\underline{y}-a\underline{x}_i|^2}\exp^{\underline{x}_i\frac{L(\underline{x}_i)}{2}}} . \tag{4.116}$$

Let $\underline{x}^{+k} \in \alpha^{+k}$ and $\underline{x}^{-k} \in \alpha^{-k}$ be the codewords, which are at minimum Euclidean distance from the demodulator's soft output sequence $\underline{y}$. Then, upon using the approximation [50,51]:

$$\ln\left(\sum_j \exp^{A_j}\right) \approx \max_j(A_j) , \tag{4.117}$$

where $\max_j(A_j)$ means the maximum value of A_j, and assuming that there were no transmission errors, i.e. $\underline{u} = \underline{x}$, then we have $L(\underline{x}) = L(\underline{u})$. Hence we can approximate Equation 4.116 as:

$$L(u_k|\underline{y}) \approx -\frac{E_B}{2\sigma^2}|\underline{y} - a\underline{x}^{+k}|^2 + \frac{1}{2}\underline{x}^{+k}L(\underline{u}) + \frac{E_B}{2\sigma^2}|\underline{y} - a\underline{x}^{-k}|^2 - \frac{1}{2}\underline{x}^{-k}L(\underline{u}) . \tag{4.118}$$

Since $|\underline{y} - a\underline{x}^{\pm k}|^2$ is the Euclidean distance between the demodulator's soft output sequences $\underline{y}$ and the legitimate transmitted codewords $\underline{x}^{\pm k}$, we can write:

$$|\underline{y} - a\underline{x}^{\pm k}|^2 = \sum_{j=1}^{n}\left(y_j - ax_j^{\pm k}\right)^2 . \tag{4.119}$$

Upon substituting Equation 4.119 into Equation 4.118, we arrive at:

$$\begin{aligned} L(u_k|\underline{y}) &\approx \frac{E_b}{2\sigma^2}\left[\sum_{j=1}^{n}\left(y_j - ax_j^{-k}\right)^2 - \sum_{j=1}^{n}\left(y_j - x_j^{+k}\right)^2\right] + \frac{L(\underline{u})}{2}\left(\underline{x}^{+k} - \underline{x}^{-k}\right) \\ &= \frac{E_b}{2\sigma^2}\left[\sum_{j=1}^{n}\left\{y_j^2 - 2ay_jx_j^{-k} + \left(ax_j^{-k}\right)^2\right\} - \sum_{j=1}^{n}\left\{y_j^2 - 2ay_jx_j^{+k} + \left(ax_j^{+k}\right)^2\right\}\right] \\ &\quad + \sum_{j=1}^{n}\frac{L(u_j)}{2}\left(x_j^{+k} - x_j^{-k}\right) . \end{aligned} \tag{4.120}$$

Since $x^{\pm k} \in \{-1,+1\}$, $\left(x_j^{-k}\right)^2 = \left(x_j^{+k}\right)^2$, and $Lc = \frac{2E_b}{\sigma^2}a$, we can simplify Equation 4.120 to:

$$\begin{aligned}
L(u_k|\underline{y}) &\approx \frac{E_b}{\sigma^2}a\left[\sum_{j=1}^{n} y_j x_j^{+k} - \sum_{j=1}^{n} y_j x_j^{-k}\right] + \sum_{j=1}^{n}\frac{L(u_j)}{2}\left(x_j^{+k} - x_j^{-k}\right) \\
&= \frac{L_c}{2}\sum_{j=1}^{n} y_j\left(x_j^{+k} - x_j^{-k}\right) + \frac{1}{2}\sum_{j=1}^{n} L(u_j)\left(x_j^{+k} - x_j^{-k}\right) \qquad (4.121) \\
&= \frac{L_c}{2}y_k\left(x_k^{+k} - x_k^{-k}\right) + \frac{1}{2}L(u_k)\left(x_k^{+k} - x_k^{-k}\right) \\
&\quad + \frac{L_c}{2}\sum_{\substack{j=1\\ i\neq k}}^{n} y_j\left(x_j^{+k} - x_j^{-k}\right) + \frac{1}{2}\sum_{\substack{j=1\\ i\neq k}}^{n} L(u_j)\left(x_j^{+k} - x_j^{-k}\right) \\
&= L_c y_k + L(u_k) + \frac{1}{2}\sum_{\substack{j=1\\ i\neq k}}^{n}\left[L_c y_j + L(u_j)\right]\left(x_j^{+k} - x_j^{-k}\right) \\
&= L_c y_k + L(u_k) + \sum_{\substack{j=1\\ i\neq k}}^{n} e_j\left[L_c y_j + L(u_j)\right] \ , \qquad (4.122)
\end{aligned}$$

where

$$e_j = \begin{cases} 0 & \text{if } x_j^{+k} = x_j^{-k} \\ 1 & \text{if } x_j^{+k} \neq x_j^{-k} \end{cases} . \qquad (4.123)$$

In harmony with the corresponding turbo BCH coding formula of Equation 3.36, here we define the extrinsic information of bit u_k as:

$$L_e(u_k) = \sum_{\substack{j=1\\ i\neq k}}^{n} e_j\left[L_c y_j + L(u_j)\right] \ , \qquad (4.124)$$

which allows us to approximate the RRNS decoder's soft output as:

$$L(u_k|\underline{y}) \approx L_c y_k + L(u_k) + L_e(u_k) \ , \qquad (4.125)$$

constituted by the sum of the soft channel output $L_c y_k$, the intrinsic information $L(u_k)$ and the extrinsic information $L_e(u_k)$.

4.7.3 Algorithm Implementation

Previously, we have shown in Equation 4.118 that in order to approximate the soft output $L(u_k|\underline{y})$, two codewords $\underline{x}^{+k}$ and $\underline{x}^{-k}$ which are nearest to $L_c\underline{y}+L(\underline{u})$ have to be found. Using the Chase algorithm described in Section 4.7.1, we can find a surviving codeword $\underline{x}$, which generates x_k on the basis of finding the codeword $\underline{x}$ having the lowest Euclidean distance from $L_c\underline{y} + L(\underline{u})$. The algorithm can be readily extended to finding another competing (or discarded) codeword $\hat{\underline{x}}$ which decodes to $\hat{x}_k \neq x_k$ and has the minimum Euclidean distance compared to all the other codewords, which decodes to $\hat{x}_k \neq x_k$. From Equation 4.121, we derive:

$$\begin{aligned} L(u_k|\underline{y}) &\approx \frac{1}{2}\left[\sum_{j=1}^{n} x_j^{+k}\{L_c y_j + L(u_j)\} - \sum_{j=1}^{n} x_j^{-k}\{L_c y_j + L(u_j)\}\right] \\ &= \frac{1}{2}\left[\sum_{j=1}^{n} x_j^{+k} y_j' - \sum_{j=1}^{n} x_j^{-k} y_j'\right] \\ &= \frac{1}{2}\left[\underline{x}^{+k}\underline{y}' - \underline{x}^{-k}\underline{y}'\right] , \end{aligned} \tag{4.126}$$

where $y_j' = L_c y_j + L(u_j)$. Given the surviving and discarded codewords and Equation 4.126, we approximate the soft output as:

$$\begin{aligned} L(x_k|\underline{y}) &\approx \frac{x_k}{4}\left[(\underline{y}')^2 - 2\hat{\underline{x}}\underline{y}' + \hat{\underline{x}}^2 - (\underline{y}')^2 + 2\underline{x}\underline{y}' - \underline{x}^2\right] \\ &= x_k\left[\frac{|\underline{y}' - \hat{\underline{x}}|^2 - |\underline{y}' - \underline{x}|^2}{4}\right] . \end{aligned} \tag{4.127}$$

This expression can be interpreted physically as the difference between the Euclidean distances of the surviving codeword $\underline{x}$ and the discarded codeword $\hat{\underline{x}}$ from the sequence $\underline{y}$ constituted by $y_j' = L_c y_j + L(u_j)$. In order to find the transmitted codeword $\underline{x}$ with a high probability, we have to increase the radius of perturbation in Figure 4.7. Therefore, we increase the number of least reliable bit positions l considered in the Chase algorithm and also the number of test patterns TP. It is clear that the probability of finding the most likely codeword $\underline{x}$ increases with l. However, the complexity of the decoder increases exponentially with l and hence we must find a tradeoff between complexity and performance. This also implies that in some cases, we shall be unable to find a discarded codeword $\hat{\underline{x}}$, which decodes to $\hat{x}_k \neq x_k$, given the l test positions. If no discarded codeword $\hat{\underline{x}}$ is found, we have to find another method of approximating the soft output. Pyndiah [148] suggested that the soft output can be approximated as:

$$L(u_k|\underline{y}) \approx y_k' + \beta \times L_c x_k , \tag{4.128}$$

where $y_k' = L_c y_k + L(u_k)$ and β is a reliability factor, which increases with the iteration index and that can be optimised by simulation. This rough approximation of the soft output is justified by the fact that if no discarded codewords $\hat{\underline{x}}$ were found by the Chase algorithm which decode to $\hat{x}_k \neq x_k$, then the discarded codewords $\hat{\underline{x}}$ which decode to $\hat{x}_k \neq x_k$ are probably far from $\underline{y}'$ in terms of the Euclidean distance. Since the discarded codewords $\hat{\underline{x}}$

are far from $\underline{y}'$, the probability that the decision u_k is correct is relatively high and hence the reliability of u_k, $L(u_k)$, is also high.

We note here that there is a distinct similarity between this algorithm and the Soft Output Viterbi Algorithm (SOVA) proposed by Hagenauer [53, 54], which was covered in Section 3.3.5. According to Equation 3.51 in the SOVA, the surviving path $\underline{s}$ is decided on the basis of the demodulator's soft output sequence $\underline{y}$ and the intrinsic information $L(\underline{u})$. The surviving path $\underline{y}$ determines the surviving codeword $\underline{x}$ in this case. Then, the soft output of the SOVA is given by Equation 3.57 in Section 3.3.5. Explicitly this soft output is proportional to the minimum path metric difference between the surviving path $\underline{s}$, which decodes to x_k, and a discarded path $\hat{\underline{s}}$, which decodes to $\hat{x}_k \neq x_k$. Observe that Equation 3.57 is similar to Equation 4.127. Specifically, Equation 4.127 identifies the codewords having the *minimum Euclidean distance difference* and evaluates the weight difference between the surviving codeword $\underline{x}$ and the discarded codeword $\hat{\underline{x}}$.

It was also proposed by Pyndiah [62] that a weighting factor α should be introduced in Equation 4.125, as follows:

$$L(u_k|\underline{y}) \approx L_c y_k + \alpha L(u_k) + L_e(u_k) \ . \tag{4.129}$$

The weighting factor α takes into account that the standard deviation of the demodulator's soft output sequence $\underline{y}$ from its expected value and that of the intrinsic information $L(\underline{y})$ are different [12, 13, 62]. The standard deviation of the extrinsic information $L_e(u_k)$ is comparatively high in the first few decoding steps and decreases during future iterations as the associated reliability increases. This scaling factor α is also used to reduce the effect of the extrinsic information in the decoder during the first decoding steps, when the BER is relatively high. The value of α is small in the initial stages of decoding and it increases, as the BER tends to zero.

The parameter α in Equation 4.129 and β in Equation 4.128 can be determined experimentally, in order to achieve an optimum performance. Both α and β were given in [62], which are reproduced in Table 4.6, where the decoding index j in Table 4.6 is the index of the decoding steps given by:

$$j = 2 \times \text{Number of iterations} + \text{Decoder Index} \ . \tag{4.130}$$

	Decoding index j							
	1	2	3	4	5	6	7	8
$\alpha(j)$	0.0	0.2	0.3	0.5	0.7	0.9	1.0	1.0
$\beta(j)$	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0

Table 4.6: The weighting factors α and reliability factors β versus the decoding index j.

4.8 Complexity

In Sections 4.4, 4.7.1 and 4.7.2, we have described the hard decision, soft decision and turbo decoding of RRNS codes. Let us now estimate the complexity of each decoding algorithm. It was shown in Section 4.7 that both the soft decision decoding and turbo decoding of RRNS codes is dependent on their hard decision decoding, which is the multiple error correction procedure highlighted in Section 4.4. In Figure 4.1, we showed the flow chart of the associated multiple error correction procedures, suggesting that for each new moduli set, the decoder has to perform MRC with t' number of moduli deleted. In Section 4.2.3, we have shown that in order to determine the mixed radix coefficients, one integer subtraction, one integer multiplication and one integer modulus operations has to be performed recursively. Let us denote the combined operation constituted by one integer subtraction, one integer multiplication and one integer modulus by δ. Since computer simulations have shown that MRC is the bottleneck of the decoding algorithm, the number of combined operations δ is used as the basis of our estimated complexity comparison in our forthcoming discussions.

Since the combined operation δ is invoked recursively in the MRC for determining the mixed radix coefficients, the algorithm can be summarised by the flowchart of Figure 4.8.

In Figure 4.8 n is the codeword length and t is the error correction capability of the RRNS code. Hence, we estimate the complexity of the MRC algorithm as:

$$comp(\mathrm{MRC}) = \left[n(n-t-1) - \sum_{i=1}^{n-t-1} i \right] \times \delta \; . \tag{4.131}$$

The estimated complexity of the hard decision decoder is very much dependent on the number of moduli combinations to be employed with the MRC algorithm in Equation 4.131. In Section 4.4, we stated that the total number of moduli combinations is equal to ${}^{n}C_{t}$. The decoder has to search through the ${}^{n}C_{t}$ number of moduli, until a combination of moduli is found, which matches the position of t errors, as we also have seen in the context of our numerical example of Section 4.4. Assuming that t residue errors are randomly distributed among the n residues of a codeword, the probability of each moduli combination matching the position of t errors, is equal. Statistically speaking, the number of moduli combinations

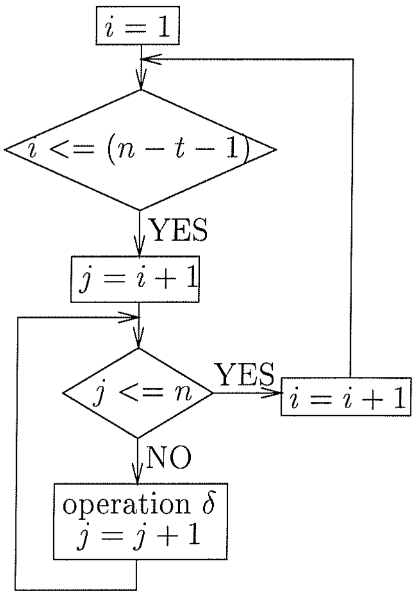


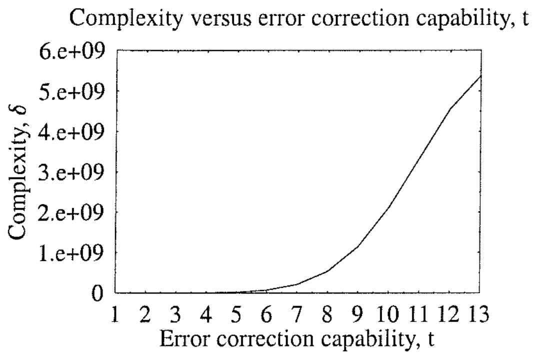
Figure 4.8: Flow chart of MRC algorithm.

to be tested for every codeword is equal to $\frac{{}^{n}C_t}{2}$. Since each moduli combination involves one MRC operation, we estimate the complexity of the hard decision decoder as:

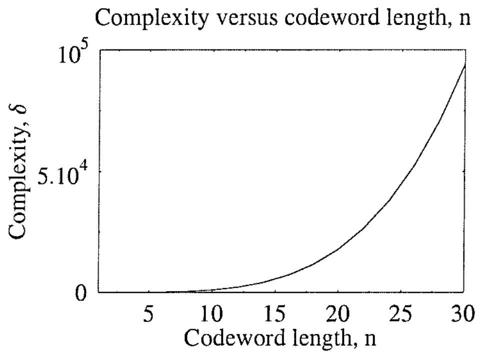
$$comp(\text{Hard decision}) = \frac{{}^{n}C_t}{2}\left[n(n-t-1) - \sum_{i=1}^{n-t-1} i\right] \times \delta \,. \tag{4.132}$$

Equation 4.132 suggests that the estimated complexity of the hard decision decoder depends only on the codeword length n and on the error correction capability t. Since ${}^{n}C_t$ increases exponentially as n or t increases, the value of the ${}^{n}C_t$ term is the dominant factor in Equation 4.132.

Figure 4.9 shows our estimated complexity comparisons for different hard decision based RRNS decoders. The estimated complexity was computed using Equation 4.132. Figure 4.9(a) shows the estimated complexity comparison of hard decision RRNS decoders having a different error correction capability t. The codeword length n was fixed at $n = 26$. As expected, the estimated complexity of the decoder increases exponentially with the error correction capability t. Figure 4.9(b) also shows our estimated complexity comparisons for the hard decision decoder using different codeword lengths n. The error correction capability t was fixed at $t = 2$. Similarly, we can see that the estimated complexity of the decoder increases exponentially with the codeword length n. However, the estimated complexity increase rate per codeword length n appears to be slow compared to the estimated complexity increase rate per error correction capability t. This is due to the dominance of the ${}^{n}C_t$ term in Equation 4.132, since ${}^{n}C_t$ increases more rapidly with t than with n.



(a) Estimated complexity versus the error correction capability t. The codeword length was fixed at $n = 26$.



(b) Estimated complexity versus the codeword length n. The error correction capability was fixed at 2.

Figure 4.9: Estimated complexity comparison of different hard decision RRNS decoders.

Let us now consider the estimated complexity of the soft decision and turbo decoding based RRNS decoder, which is directly proportional to the estimated complexity of the hard decision decoder. For a soft decision decoder, we have seen in Section 4.7.1 that 2^l number of test patterns are generated, where the tentatively corrupted l bit positions are associated with the least reliable confidence values in the received sequence. For each test pattern, initially the hard decision decoder is invoked for decoding the perturbed binary codeword. Therefore, we can estimate the complexity of the soft decision coded RRNS decoder as:

$$\begin{aligned} comp(\text{Soft decision}) &= 2^l \times comp(\text{Hard decision}) \\ &= 2^{l-1} \times^n C_t \left[n(n-t-1) - \sum_{i=1}^{n-t-1} i \right] \times \delta \,. \end{aligned} \tag{4.133}$$

Likewise, 2^l test patterns are generated for turbo RRNS decoding. Since the turbo RRNS decoding involves iterative decoding, the number of iterations has to be considered in calculating the decoding complexity. Furthermore, there are two component decoders for each iteration. Hence, the complexity of turbo RRNS decoding is estimated as:

$$\begin{aligned} comp(\text{Turbo decoding}) &= 2 \times \text{Iteration no} \times 2^l \times comp(\text{Hard decision}) \\ &= \text{Iteration no} \times 2^l \times^n C_t \left[n(n-t-1) - \sum_{i=1}^{n-t-1} i \right] \times \delta \,. \end{aligned} \tag{4.134}$$

Let us consider an example for comparing the estimated complexity of hard decision, soft decision and turbo RRNS decoding of two RRNS(28,26) and RRNS(28,24) codes. They have an error correction capability of $t = 1$ and $t = 2$, respectively. Assuming that $l = 4$ and the number of iterations is four, we can apply Equations 4.132, 4.133 and 4.134 for finding the estimated complexity of the three decoding algorithms. The estimated complexity of each algorithm is shown in Table 4.7 for the RRNS(28,26) and RRNS(28,24) codes.

Code	t	Complexity, δ		
		Hard decision	Soft Decision	Turbo decoding
RRNS(28,26)	1	5,278	84,448	675,584
RRNS(28,24)	2	70,875	1,134,000	9,072,000

Table 4.7: Estimated complexity of hard decision, soft decision and turbo decoding for the RRNS(28,26) and RRNS(28,24) codes. The number of turbo decoding iterations was four and $l = 4$ was used for both soft decision based and turbo decoding.

Using the values tabulated in Table 4.7, we plotted the estimated complexity bar chart of hard decision, soft decision and turbo decoding in Figure 4.10. We can see in the figure that the estimated complexity of soft decision decoding is $2^4 = 16$ times higher than that of hard decision decoding. For turbo decoding, the estimated complexity is about $2^4 \times 2 \times 4 = 128$ times higher, than that of hard decision decoding and it is $2 \times 4 = 8$ times higher, than that of soft decision decoding. Again, we can see that the error correction capability t plays an important role in determining the estimated complexity of the decoding algorithm. Let us consider turbo decoding as an example, where the estimated complexity of the turbo RRNS(28,24) code, which corrects $t = 2$ errors, is about 13.4 times higher, than that of the turbo RRNS(28,26) code, where $t = 1$.

Finally, we conclude that error correction capability t is the major factor affecting the estimated complexity of all three decoding algorithms; namely that of hard decision, soft decision and turbo decoding. In order to obtain simulation results for $t > 1$, we attempted to simplify the considered operation δ. Explicitly, we replaced the operation with one integer subtraction and one two dimensional table lookup operation. Explicitly, we invested memory for increasing the computational speed. Our computer simulations have shown that with the aid of this new technique, we were capable of reducing the computational complexity by about 70%.

4.9 Simulation Results

In this section, we present our simulation results for RRNS hard decision, soft decision and turbo decoding. All simulation results were obtained using BPSK over AWGN channels. We will demonstrate how the performance of RRNS codes is affected by the following

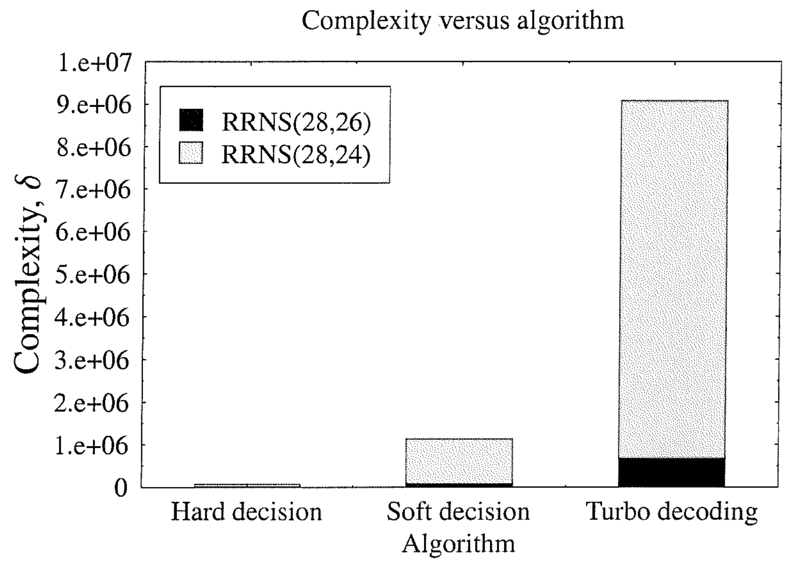


Figure 4.10: Estimated complexity comparison for hard decision, soft decision and turbo decoding of the RRNS(28,26) and RRNS(28,24) codes. The number of iterations was four and $l = 4$ was used for both soft decision based and turbo decoding.

parameters:

- The RRNS encoding algorithm - non-systematic, systematic and modified systematic;
- Error correction capability, t and length n of the codeword;
- The decoding algorithm - hard decision, soft decision and turbo decoding;
- The number of turbo decoding iterations used;
- The design of the turbo RRNS interleaver;

All the simulation results were generated using the encoder and decoder structures described in this chapter. The following parameters were the same for all simulations, unless otherwise stated:

- The RRNS codes used were maximum distance separable;
- The RRNS encoders used modified systematic bit-to-residue mapping;
- The number of bits k_{b_j}, used to represent all residues in a codeword was the same;
- The weighting factors α and reliability factors β used were specified in Table 4.6;

- The number of turbo decoding iterations was four;
- No puncturing of the parity bits of the upper and lower encoders was used.

For a fixed number of Bits Per Symbol (BPS), k_{b_j}, used to represent all the residues in a codeword, the moduli set was chosen such that it maximised the code's legitimate dynamic range. Therefore, the moduli were chosen to be as large as possible, which are shown in Table 4.8.

BPS	max n	Moduli
4	4	11, 13, 15, 16.
5	8	17, 19, 23, 25, 27, 29, 31, 32.
6	10	37, 41, 43, 47, 53, 55, 59, 61, 63, 64.
7	18	67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 119, 121, 123, 125, 127, 128.
8	28	131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 217, 223, 227, 229, 233, 239, 241, 247, 251, 253, 255, 256.
9	50	257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 437, 439, 443, 449, 457, 461, 463, 467, 473, 479, 487, 491, 493, 499, 503, 505, 507, 509, 511, 512.
10	82	521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 949, 953, 967, 971, 977, 983, 989, 991, 997, 1003, 1007, 1009, 1013, 1015, 1019, 1021, 1023, 1024.
11	150	1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381, 1399, 1409, 1423, 1427, 1429, 1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499, 1511, 1523, 1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621, 1627, 1637, 1657, 1663, 1667, 1669, 1681, 1693, 1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783, 1787, 1789, 1801, 1811, 1823, 1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889, 1891, 1901, 1907, 1913, 1931, 1933, 1943, 1949, 1951, 1961, 1973, 1979, 1987, 1991, 1993, 1997, 1999, 2003, 2011, 2017, 2021, 2023, 2027, 2029, 2033, 2039, 2041, 2043, 2045, 2047, 2048.

Table 4.8: List of the largest moduli for a given fixed number of BPS.

4.9.1 Hard Decision Decoding

4.9.1.1 Encoder Types

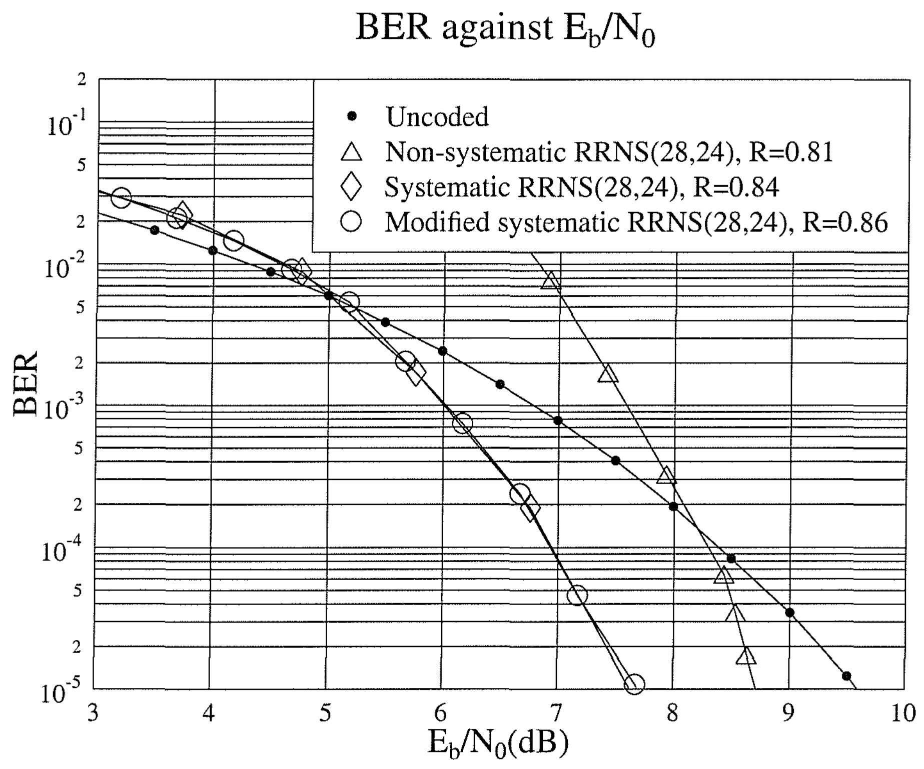


Figure 4.11: Performance comparison between the non-systematic, systematic and modified systematic 8 BPS RRNS(28,24) encoders over AWGN channels. The moduli set used is shown in Table 4.8.

In Section 4.5, we have proposed two different methods for mapping the binary source bits to the non-binary RRNS code, which result in non-systematic and systematic RRNS codes, respectively. Furthermore, in Section 4.5.2.1 we modified the systematic RRNS mapping rule, in order to increase the associated code rate and to achieve the same number of BPS for every residue in a codeword. This has practical advantages in many applications and allows us to highlight the similarities with the well-known family of RS codes. In Figure 4.11, we show the performance of the non-systematic, systematic and modified systematic 8 BPS RRNS(28,24) code over AWGN channels. Due to the associated different mapping methods used, the code rate of the different encoders varied. Since the number of BPS mapped to every residue in a codeword was the same for the modified systematic RRNS code, it exhibited the highest code rate.

In the figure, we can see that the BER performance of the non-systematic encoder is worse than that of the uncoded scenario for $E_b/N_0 < 8$ dB. However, there is a coding gain of about 0.8 dB at BER=10^{-5}. Both the systematic and the modified systematic RRNS(28,24) code exhibit a similar performance and they are about 1 dB better in E_b/N_0 terms, than the non-systematic code. If there is a decoding failure, there will be about 50% bit-decoding errors for the non-systematic decoder which is the reason for its poor performance for $E_b/N_0 < 8$ dB. By contrast, due to its systematic nature the systematic code, simply outputs the bits associated with the k information moduli, if a decoder failure was detected. Hence, there will be significantly less than 50% bit-decoding errors. Therefore, typically the systematic code is preferred to its non-systematic counterpart.

Throughout the rest of this chapter, the modified systematic RRNS code was used rather than the systematic RRNS code of Section 4.5.2. This is because the modified systematic RRNS code always has a higher code rate, while the BER performance of both codes is similar. Moreover, the modified systematic RRNS code has fixed number of BPS, which is again, directly comparable to systematic Reed-Solomon (RS) codes.

4.9.1.2 Comparison of Redundant Residue Number System codes and Reed-Solomon Codes

As argued earlier, RRNS codes which use the $d_{min} - 1$ number of largest moduli as their redundant moduli, are maximum distance separable. Hence, they are similar to RS codes in terms of their performance. Figure 4.12 compares the performance of the modified systematic RRNS code and that of systematic RS codes over AWGN channels. Both the RRNS(28,24) and the RS(28,24) codes constructed over GF(256) transmit 8 BPS. It can be seen from the figure that the performance of both codes is similar. For the RRNS(48,46) code, we used moduli, which transmit 6, 7 and 8 BPS, which was compared to the 8 BPS RS(48,46) code. Again as expected, the performance of both codes is similar.

There are a number of advantages when using RRNS codes instead of RS codes. For example, if a code having a short codeword length n and high number of BPS is required, we have to shorten the RS code. This implies that we have to incorporate dummy symbols at both the encoder and decoder. Then the decoder has to decode the full-length padded RS codeword, which is wasteful in terms of decoding complexity. More explicitly, the algebraic decoding of a shortened RS code implies generating $2t$ number of syndromes, in order to determine t number of error locations. These syndromes are typically calculated by substituting the first $2t$ number of primitive elements into the received polynomial and this operation cannot be readily simplified in the case of shortened RS codes [90]. By contrast, a wide range of codeword length can be generated using RRNS codes without introducing

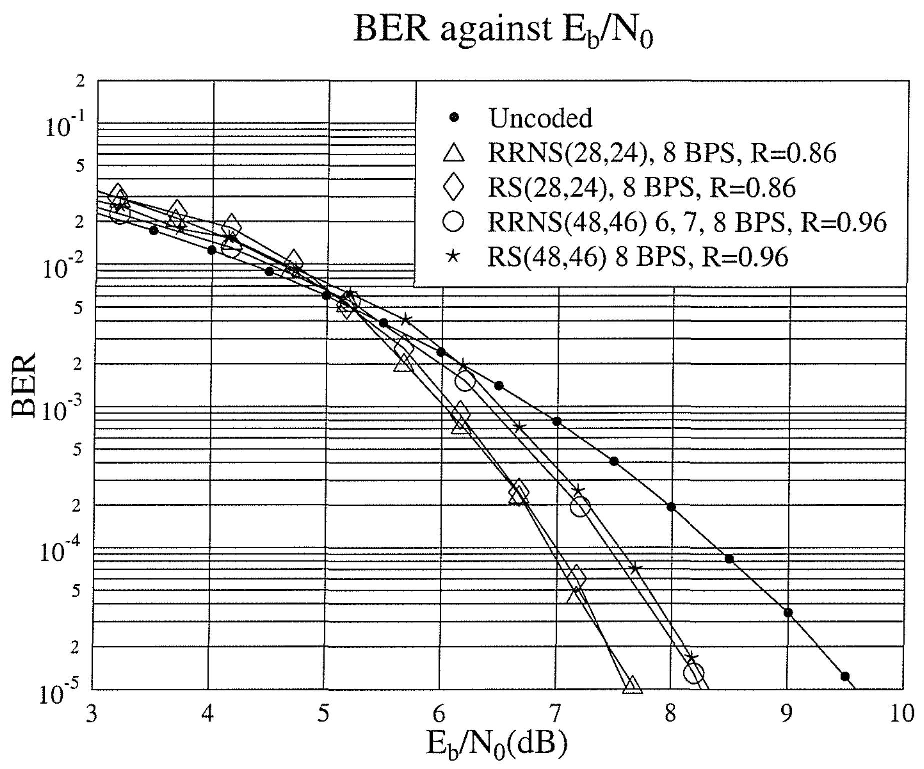


Figure 4.12: Performance comparison between modified systematic RRNS and systematic RS codes over AWGN channels. The RRNS(28,24) code is compared to the RS(28,24) code constructed over GF(256). Both codes transmit 8 BPS. Furthermore, the performance of the RRNS(48,46) code transmitting 6, 7 and 8 BPS is compared to the 8 BPS RS(48,46) code.

dummy symbols. This potentially reduces the decoding complexity.

In an automatic request (ARQ) system [149], a stronger RS code will be required for retransmission if a weaker RS code fails. This implies the transmission of another full codeword. However, for RRNS codes, a stronger code is obtained by introducing more redundant residues. If a RRNS code fails, stronger RRNS code can be obtained by transmitting more redundant residues without transmitting the whole codeword [150, 151].

However, as argued in Section 4.8, the estimated complexity of the RRNS decoder increases exponentially with the codeword length n and with the error correction capability t. Furthermore, it is extremely complex to implement a RRNS code with $n \approx 2^j$, where j is the number of BPS.

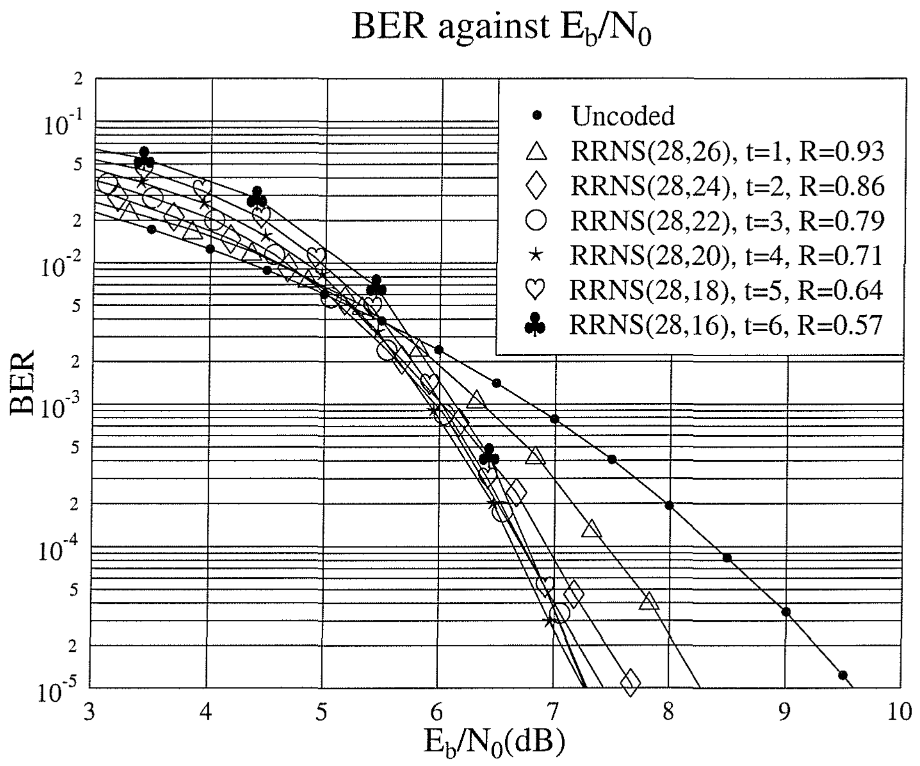


Figure 4.13: Performance comparison between different 8 BPS RRNS(28,k) codes having an error correction capability of t over AWGN channels.

4.9.1.3 Comparison Between Different Error Correction Capabilities t

Figure 4.13 shows our performance comparison between different error correction capabilities t over AWGN channels. The codes used are 8 BPS RRNS(28,k) codes, which have a fixed codeword length. We can see in the figure that as t increases, the coding gain also increases. However, the performance improvements saturate, when $t = 4$ and the best design trade-off is constituted by the RRNS(28,20) code. Since the complexity of the RRNS decoder is excessive for $t > 6$, no simulation results were provided. The associated theoretical results [152] suggest that the achievable coding gain decreases, once the code rate R is below 0.6, that is for $t > 4$.

The coding gain of the RRNS(28,k) codes in Figure 4.13 were evaluated and tabulated in Table 4.9. The estimated complexity of the decoder was calculated using Equation 4.132 and it was also shown in the table. Figure 4.14 shows the trend of the coding gain versus estimated complexity characteristic of the decoder. As we can see from the figure, increasing the estimated complexity does not necessary increase the coding gain and the best coding gain was achieved, when the estimated complexity of the decoder was relatively low.

Code	t	R	Complexity, δ	Coding gain (dB)
RRNS(28,26)	1	0.93	5,278	1.3
RRNS(28,24)	2	0.86	70,875	1.9
RRNS(28,22)	3	0.79	609,336	2.2
RRNS(28,20)	4	0.71	3,767,400	2.3
RRNS(28,18)	5	0.64	17,837,820	2.2
RRNS(28,16)	6	0.57	67,248,090	2.2

Table 4.9: Coding gain and estimated complexity of 8 BPS RRNS(28,k) codes using hard decision decoding for transmission over AWGN channels.

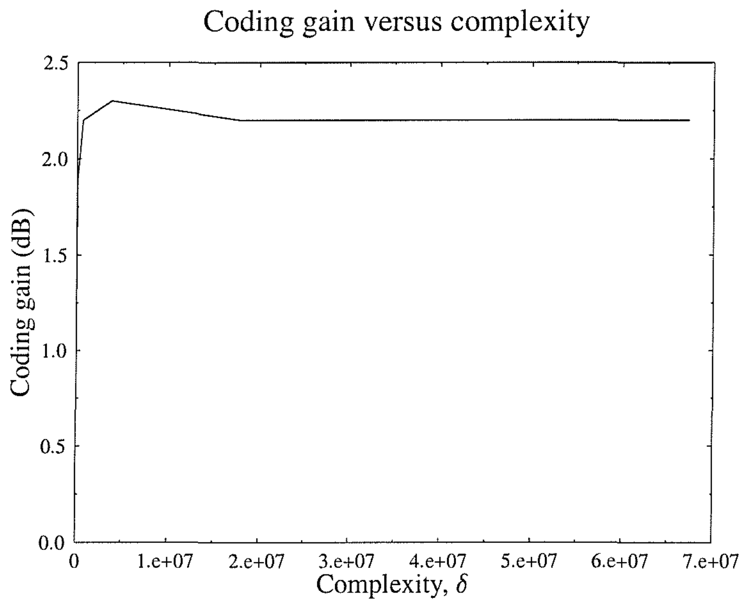


Figure 4.14: Coding gain versus estimated complexity for hard decision decoding of various 8 BPS RRNS(28,k) codes for $k = 26, 24, 22, 20, 18$ and 16.

4.9.2 Soft Decision Decoding

4.9.2.1 Effect of the Number of Test Positions

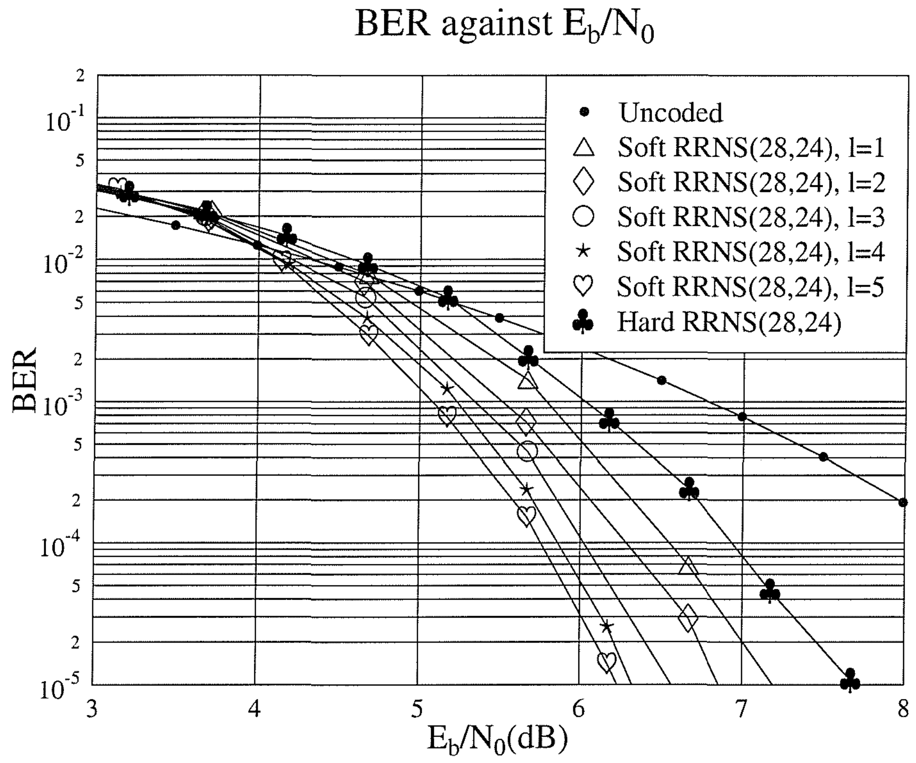


Figure 4.15: Performance of the soft decision assisted 8 BPS RRNS(28,24) decoder for different number of test positions l over AWGN channels. The performance of the hard decision decoder is also shown for comparison.

In Section 4.7.1, we outlined how we can employ the Chase algorithm to exploit the soft outputs of the demodulator. In the decoder, only a limited number l bit positions associated with the least reliable confidence values was considered. Figure 4.15 shows the performance of the soft decision assisted 8 BPS RRNS(28,24) decoder for different number of test positions l over AWGN channels. The performance of the hard decision decoder is also shown in the figure for comparison. We can see from the figure that the performance of the soft decision decoder improves, as we increase the number of test positions l. However, the incremental improvement becomes more limited, when $l > 4$. It is surmised that the best-case performance of the soft decision RRNS decoder is similar to that of the soft decision Viterbi algorithm. The estimated complexity of the decoder increases exponentially, since the number of test patterns is equal to 2^l. In Figure 4.15, the performance of the soft decision RRNS(28,24) decoder using $l = 4$ is about 1.5 dB better, than that of the hard decision assisted RRNS(28,24) decoder at a BER of 10^{-5}.

4.9.2.2 Soft Decision RRNS(10,8) Decoder

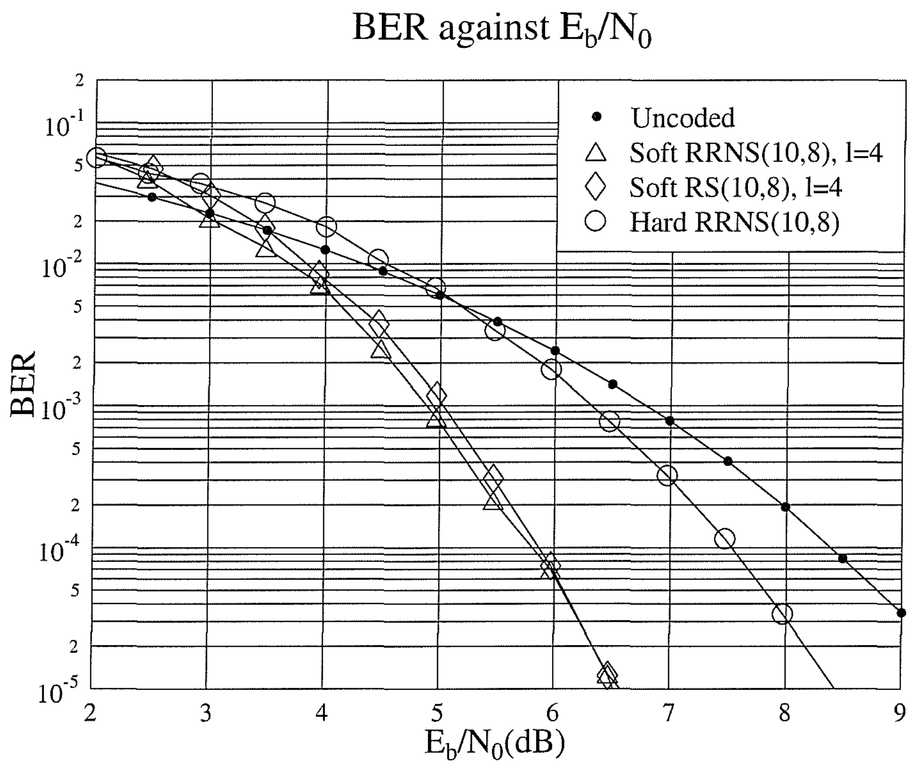


Figure 4.16: Performance comparison between hard decision assisted 6 BPS RRNS(10,8) decoder and the soft decision assisted 6 BPS RRNS(10,8) decoder for $l = 4$ over AWGN channels. The performance of the RS(10,8) code constructed over GF(64) for $l = 4$ is also shown for comparison.

Figure 4.16 compares the performance of the hard decision assisted 6 BPS RRNS(10,8) decoder and that of the soft decision aided 6 BPS RRNS(10,8) decoder for $l = 4$ over AWGN channels. The soft decision aided decoder is about 2 dB better in terms of E_b/N_0 than that of the hard decision based decoder at a BER of 10^{-5}. We applied the Chase algorithm for decoding the RS(10,8) code constructed over GF(128), where for $l = 4$ we can see that the performance of the soft decision assisted RS(10,8) code constructed over GF(64) is similar to that of the soft decision aided RRNS(10,8) decoder.

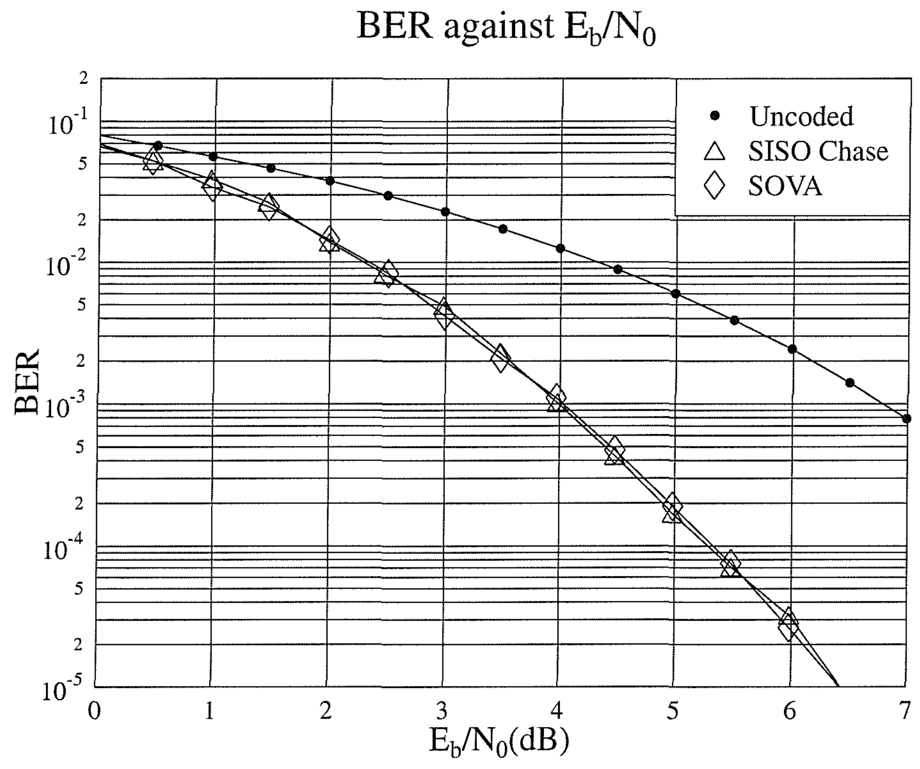


Figure 4.17: Performance comparison between the SOVA and the SISO Chase decoding algorithm using the turbo BCH(7,4) code over AWGN channels. For the SISO Chase algorithm, the weighting factors were set to $\alpha(j) = 1$ and the reliability factors to $\beta(j) = 1$ for every decoding index j and the number of test positions was $l = 5$. There were six decoding iterations and a 4×4 bit block interleaver was used.

4.9.3 Turbo RRNS Decoding

4.9.3.1 Algorithm Comparison

In Section 4.7.3, we stated that there is a similarity between the SOVA and the SISO Chase algorithm. In the SOVA, the surviving path is chosen and the soft outputs are derived on the basis of the minimum path metric differences. Similarly, the SISO Chase algorithm searches for the surviving codeword and the associated soft outputs are proportional to the minimum weight difference. Therefore, if the weighting factors $\alpha(j) = 1$ in Equation 4.129 and the reliability factors $\beta(j) = 1$ in Equation 4.128 for all are significantly high values of j and than all the valid codewords are considered during the decoding process and hence the SISO Chase algorithm is fairly similar to the SOVA. In this section, we use the BCH(7,4) code as the component code of the turbo BCH(7,4) code. BCH codes were favoured since a Viterbi decoder can also be invoked for their decoding and hence the Log-MAP, Max-Log-MAP and SOVA can be applied as benchmarkers of the SISO Chase algorithm.

In Figure 4.17, we compare the performance of the SOVA and the SISO Chase decoding algorithm using the turbo BCH(7,4) code over AWGN channels. Since there are only $2^4 = 16$ valid codewords in the BCH(7,4) code, $2^{l=5} = 32$ test patterns are sufficient to perturb the received sequences to all the 16 valid codewords. The weighting factors $\alpha(j)$ and reliability factors $\beta(j)$ of Equations 4.129 and 4.128 are 1 for all decoding indices. Both algorithms have a total of six decoding iterations and a 4×4 bit block interleaver was used. From Figure 4.17, we can see that the performance of the algorithms is identical. Hence, we have shown that the concept of the SISO Chase algorithm is similar to that of the SOVA.

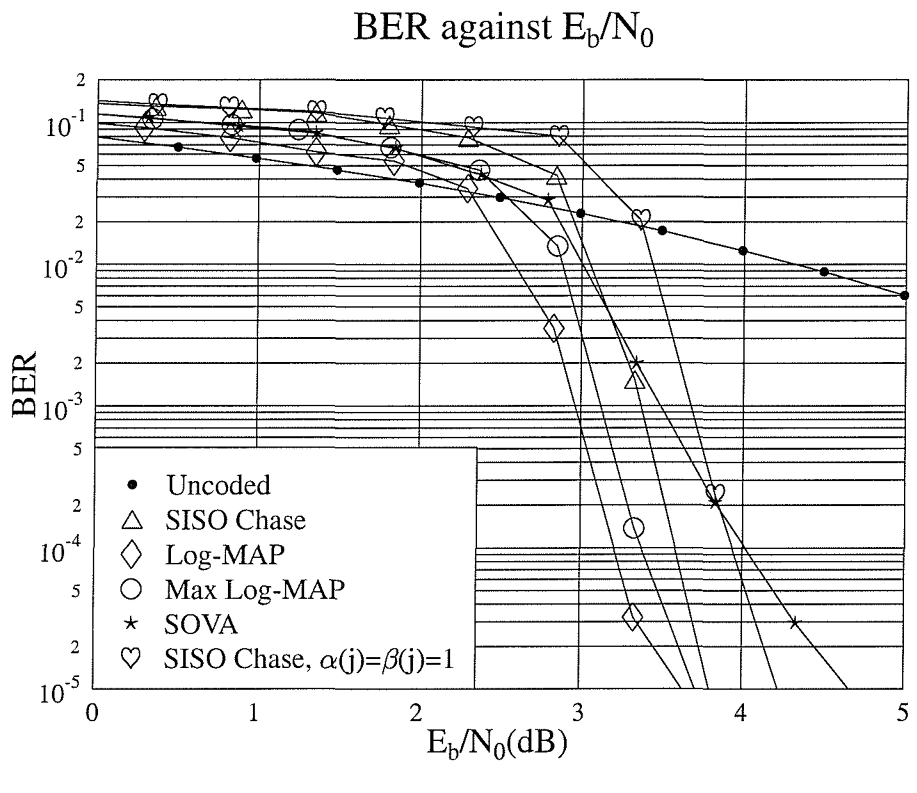


Figure 4.18: Performance comparison between different decoding algorithms using the turbo BCH(63,57) code over AWGN channels. There were six decoding iterations and a 57×57 bit block interleaver was used. For the SISO Chase algorithm, $\alpha(j)$ and $\beta(j)$ were specified in Table 4.6 and we had $l = 4$.

Let us now compare the performance of the SISO Chase algorithm to that of other well-known algorithms, such as the Log-MAP, Max-Log-MAP and SOVA in the context of binary turbo BCH codes. Figure 4.18 shows our performance comparison between the different decoding algorithms using the turbo BCH(63,57) code over AWGN channels. There were six decoding iterations and a 57×57 bit block interleaver was used. For the SISO Chase algorithm, $\alpha(j)$ and $\beta(j)$ were specified in Table 4.6 and we had $l = 4$. Since the Log-MAP decoding algorithm is optimum, its BER performance is the best in Figure 4.18. The Max Log-MAP decoding algorithm suffers from a slight degradation of 0.1 dB at BER= 10^{-5} compared to the Log-MAP decoding algorithm. With the optimum values of $\alpha(j)$ and $\beta(j)$ given by Pyndiah [62] as shown in Table 4.6, the SISO Chase algorithm exhibits a slight E_b/N_0 performance degradation of 0.2 and 0.1 dB at BER= 10^{-5} compared to the Log-MAP and the Max Log-MAP decoding algorithms, respectively. As compared to the SOVA, the SISO Chase algorithm performs better, having a 0.8 dB E_b/N_0 advantage at a BER of 10^{-5}. Also shown in Figure 4.18 is the performance of the SISO Chase algorithm in conjunction with $\alpha(j) = \beta(j) = 1$ for every decoding index. It can be seen that its E_b/N_0 performance is about 0.4 dB worse, than that of the SISO Chase algorithm using the $\alpha(j)$ and $\beta(j)$ values shown in Table 4.6. Hence, we have shown that the BER performance can be improved by optimising the values of $\alpha(j)$ and $\beta(j)$.

4.9.3.2 Number of Iterations Used

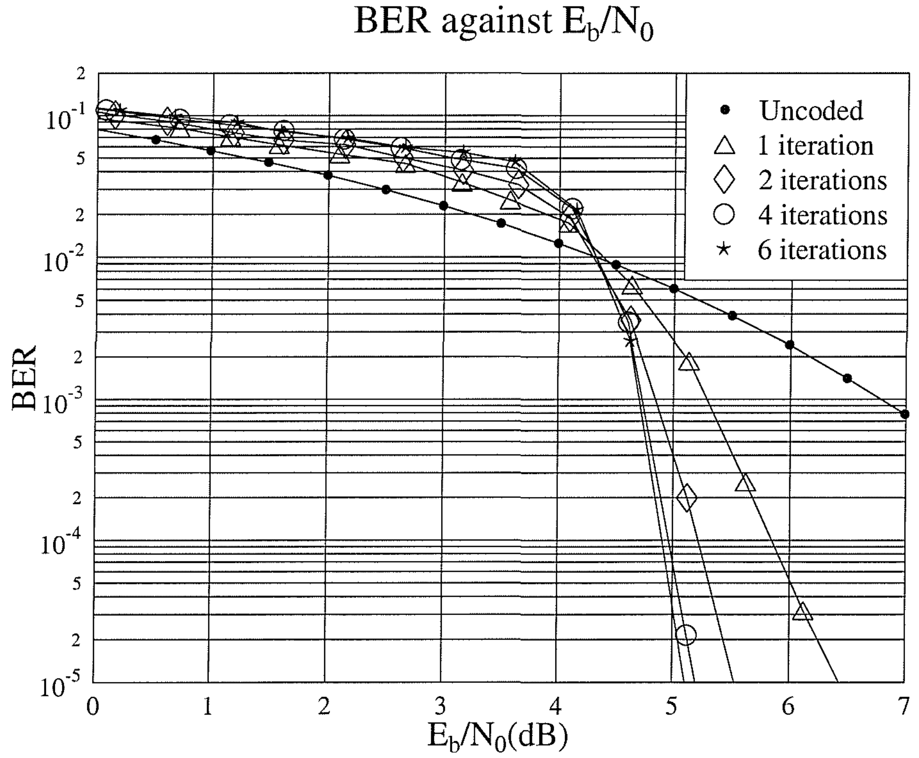


Figure 4.19: Performance comparison for different number of iterations using the rate $R0.87$ 8 BPS turbo RRNS(28,26) code, a 26×26 symbol block interleaver, and the $\alpha(j)$ and $\beta(j)$ values shown in Table 4.6, over AWGN channels.

Figure 4.19 shows the performance of different number of iterations using the 8 BPS turbo RRNS(28,26) code, over AWGN channels. The interleaver used is a 26×26 symbol block interleaver, while the $\alpha(j)$ and $\beta(j)$ values used in our simulations are shown in Table 4.6. Since the parity symbols of both the upper and lower encoder are transmitted through the channel, the resultant code rate is 0.86. As the number of iterations performed by turbo decoder increases, the performance of the turbo code improves. However, after four iterations the improvement becomes insignificant.

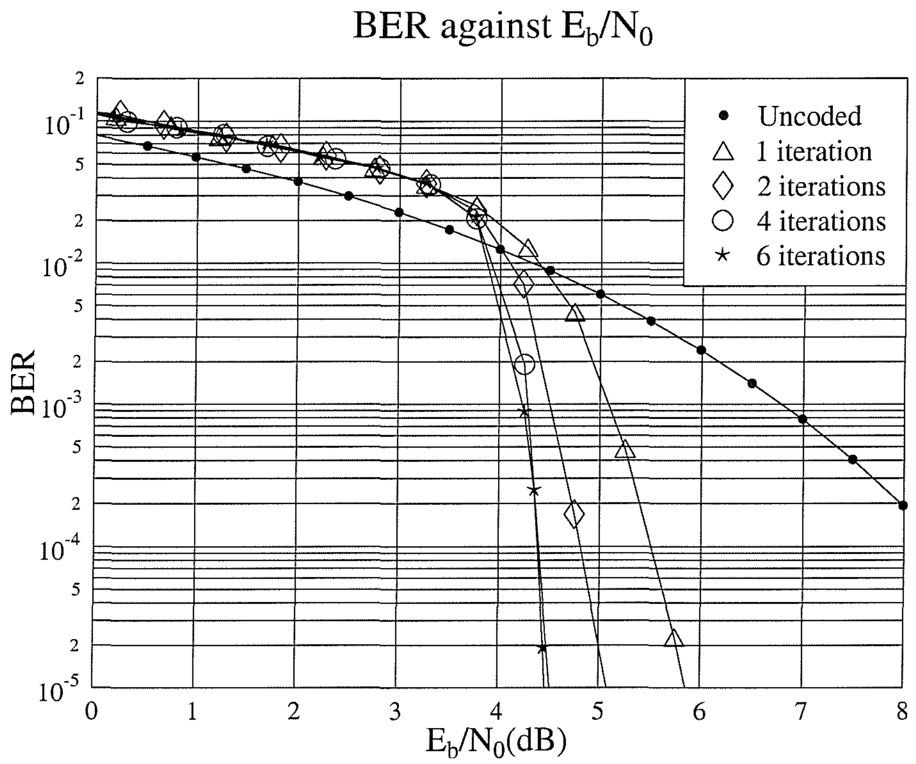


Figure 4.20: Performance comparison for different number of iterations using the rate $R = 0.75$ 8 BPS turbo RRNS(28,24) code, a 24×24 symbol block interleaver, and the $\alpha(j)$ and $\beta(j)$ values shown in Table 4.6, over AWGN channels.

Figure 4.20 also shows our performance comparisons for different number of iterations over AWGN channels. The turbo component codes are constituted by the 8 BPS RRNS(28,24) code and the code rate is 0.75. The values of $\alpha(j)$ and $\beta(j)$ used were shown in Table 4.6 and the turbo interleaver is a 24×24 symbol block interleaver. It can be seen from the figure that the performance improvement due to each extra iteration is higher compared to that of the turbo RRNS(28,26) code. For instance, at a BER of 10^{-5}, the E_b/N_0 performance of the turbo RRNS(28,24) code using two iterations is about 2 dB better, than that of the first iteration. By contrast, for the turbo RRNS(28,26) code, the performance of the second iteration is only 1 dB better in terms of E_b/N_0, than that of the first iteration. However, the performance of the turbo RRNS(28,24) code does not improve significantly after four iterations. Hence, the simulation results shown in the following sections will be based on four iterations.

4.9.3.3 Imperfect Estimation of the Channel Reliability Value L_c

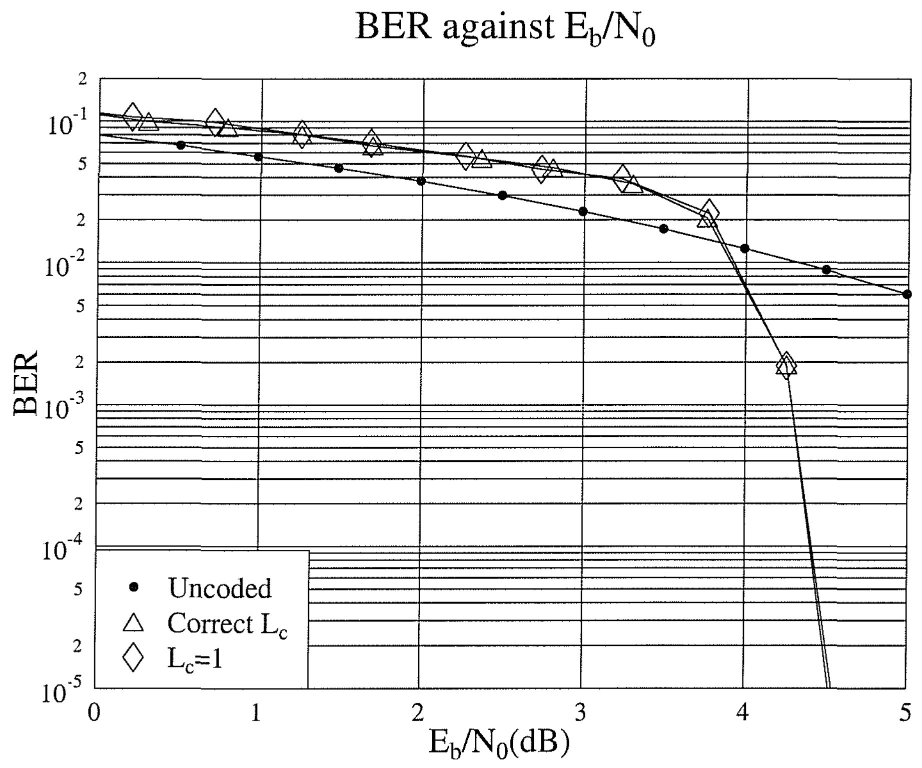


Figure 4.21: Effect of using incorrect channel reliability value L_c on the performance of the turbo RRNS(28,24) code employing four iterations, and a 24×24 symbol block interleaver. The values of $\alpha(j)$ and $\beta(j)$ were shown in Table 4.6.

In Equation 4.122 of Section 4.7.2, we have shown that we can approximate the soft output as:

$$L(u_k|\underline{y}) \approx L_c y_k + L(u_k) + \sum_{\substack{j=1 \\ i \neq k}}^{n} e_j \left[L_c y_j + L(u_j) \right] \; . \qquad (4.135)$$

In the first iteration of the iterative decoding process, there is no intrinsic information

$L(u_j) = L(u_k) = 0$ for the data bits. Therefore, we can simplify Equation 4.135 as follows:

$$L(u_k|\underline{y}) \approx L_c \left[y_k + \sum_{\substack{j=1 \\ i \neq k}}^{n} e_j y_j \right] . \tag{4.136}$$

Over AWGN channels the channel reliability value L_c becomes a constant in Equation 4.136. Therefore, we can omit the channel reliability value in deriving the soft outputs of the data bit u_k. Equation 4.135 is then simplified and rewritten as follows:

$$L_e(u_k) \approx y_k + L(u_k) + \sum_{\substack{j=1 \\ i \neq k}}^{n} e_j \left[y_j + L(u_j) \right] . \tag{4.137}$$

Figure 4.21 shows the associated performance, when using the correct L_c value and when setting $L_c = 1$ for the 8 BPS turbo RRNS(28,24) code over AWGN channels. The number of iterations was four and a 24×24 symbol block interleaver was used. The values of $\alpha(j)$ and $\beta(j)$ were shown in Table 4.6. The performance of the 8 BPS turbo RRNS(28,24) code using the correct values of L_c is shown in the figure for comparison purpose. It can be seen that the SISO Chase algorithm performs equally well whether or not the correct values of L_c are known. A similar observation is valid for the SOVA and the Max-Log-MAP algorithms, which were detailed in Section 3.6.3.

4.9.3.4 The Effect of the Turbo Interleaver

Conventionally, the turbo interleaver is used to disperse the parity information incorporated by the encoder into the transmitted sequence. In turbo RRNS codes the parity information is transmitted twice. Even if a number of bits are corrupted by channel errors, the second set of interleaved parity information may assist the decoder in removing these errors, provided that these bits are not corrupted by the channel. For binary codes, such as the turbo BCH codes of Chapter 3, bit-based turbo interleavers were used. In this section we provide performance results of a variety of interleavers.

Specifically, Figure 4.22 shows our performance comparison between a 12480 symbol (8 BPS) random interleaver and a 99840 bit random interleaver over AWGN channels. More explicitly, the interleaving depth of both interleavers is about 100,000 bits. At a BER of 10^{-5} the performance of the bit interleaver is about 0.2 dB better in terms of E_b/N_0, than that of the symbol interleaver. This result might suggest that using a bit interleaver is more beneficial, than a symbol interleaver in terms of scrambling the input data bits and spreading the parity information. This assists in passing independent information between the two decoders, which is vital for decoding turbo codes iteratively.

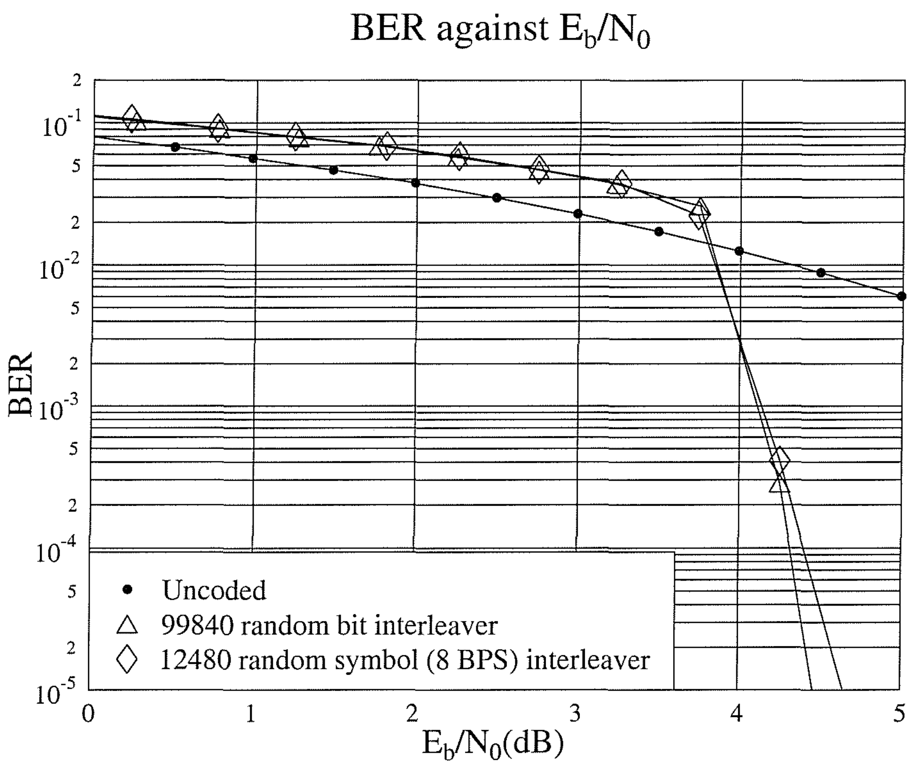


Figure 4.22: Performance comparison of the 8 BPS turbo RRNS(28,24) code for an 12480 random symbol (8 BPS) random interleaver and a 99840 bit random interleaver over AWGN channels. The number of iterations was four and values of $\alpha(j)$ and $\beta(j)$ were shown in Table 4.6.

It is well known that the design of the turbo interleaver affects the performance of the turbo code [12,13,69]. One of the most important factors is the size of the turbo interleaver. Impressive performance results have been obtained using large interleavers [12,13,69]. However, a large interleaver implies a long delay, which is an impediment in interactive speech and video transmissions. Hence below we will characterise the performance of low-latency interleavers.

We show in Figure 4.23, how the interleaver length affects the performance of the 8 BPS turbo RRNS(28,24) code employing four iterations over AWGN channels. The smallest interleaver size that can accommodate one RRNS(28,24) codeword is the 8 BPS 6×4 symbol block interleaver, resulting in a delay of 192 bits. Due to its small interleaver depth, the performance of this scheme is moderate, achieving a BER of 10^{-5} at $E_b/N_0 = 6.8$ dB. If we increase the interleaver size to a 12×10 8 BPS block symbol interleaver, the performance of the turbo code is about 1.5 dB better in terms of the required E_b/N_0, than in the previous case at a BER of 10^{-5}. A further increase of 0.8 dB E_b/N_0 improvement is observed at

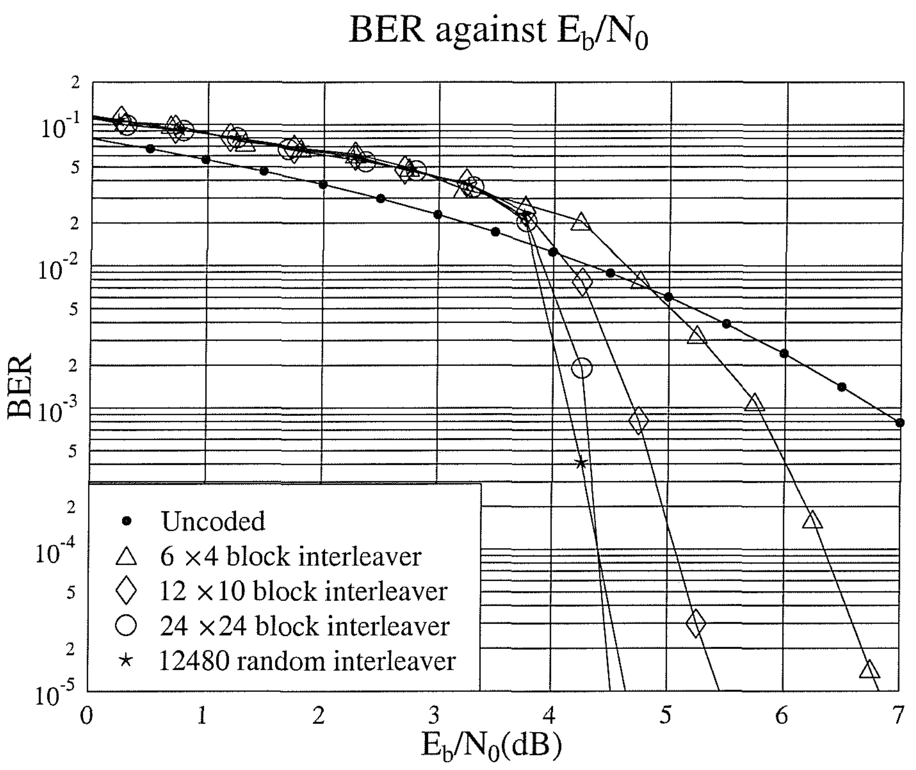


Figure 4.23: Performance comparison between different length symbol interleavers for the 8 BPS RRNS(28,24) employing four iterations over AWGN channels. The values of $\alpha(j)$ and $\beta(j)$ were shown in Table 4.6.

BER$=10^{-5}$ when the interleaver size is increased to using a 24×24 8 BPS block interleaver. However, beyond this interleaver size there is no significant coding gain improvement, when the interleaver size increases for example to 12480 symbols or about 100,000 bits.

4.9.3.5 The Effect of the Number of Bits Per Symbol

Figure 4.24 shows the performance of various 8, 9, 10 and 11 BPS turbo RRNS(28,24) codes employing four iterations over AWGN channels. The interleavers were implemented using random symbol interleaving and their size was chosen such that the interleaver depth was approximately 100,000 bits. Figure 4.24 shows that the BER performance degrades as we increase the number of bits per symbol (BPS). This is because over AWGN channels the errors occur randomly and independently. Therefore, as the number of BPS increases, the probability of symbol errors increases.

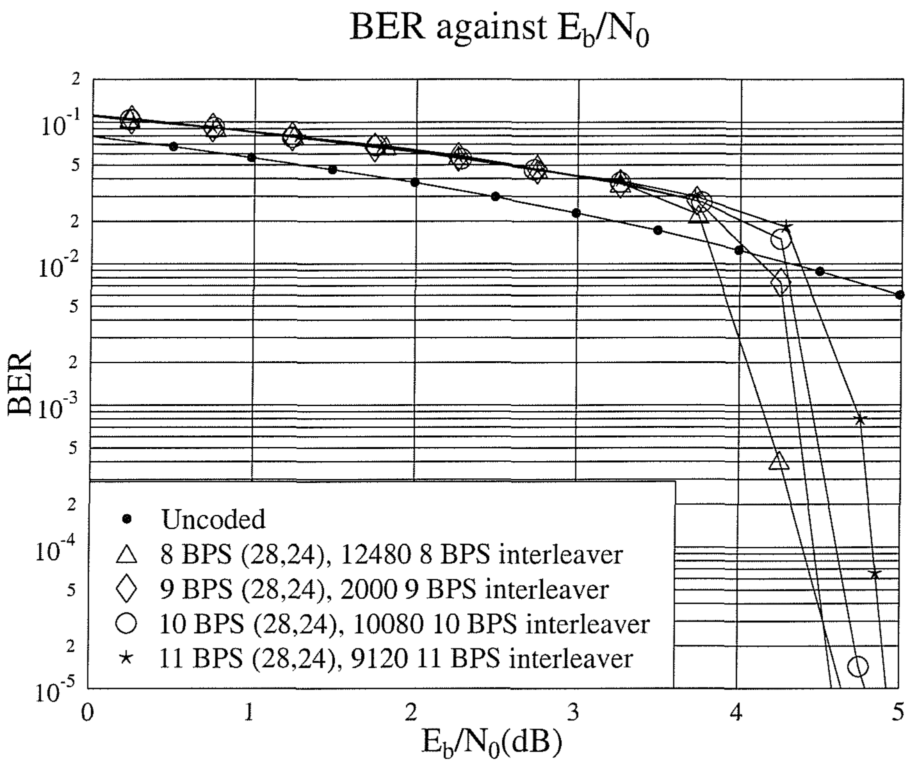


Figure 4.24: Performance comparison of different number of BPS turbo RRNS(28,24) codes employing four iterations over AWGN channels. All interleavers used were random and the values of $\alpha(j)$ and $\beta(j)$ were shown in Table 4.6.

4.9.3.6 Coding Gain Versus Estimated Complexity

In Section 4.8, we have studied the estimated complexity of hard decision, soft decision and turbo decoding assisted RRNS codes. The estimated complexity of the decoding algorithms was also shown in Figure 4.10 for the RRNS(28,26) and RRNS(28,24) codes. Let us now investigate the amount of coding gain achievable by investing more complexity at the decoder.

Figure 4.25 shows our performance comparison for the hard decision, soft decision and turbo decoding assisted RRNS(28,24) codes over AWGN channels. Both the soft decision and turbo decoding aided schemes had the same number of test positions, namely $l = 4$. For turbo decoding, the number of iterations was four and a 12480 8 BPS random interleaver was used. As shown in the figure, the BER performance of the RRNS(28,24) code improves as we invest more complexity. Specifically, with the aid of turbo decoding, the RRNS(28,24) code achieves a maximum coding of 5.0 dB at a BER of 10^{-5}. The estimated complexity versus coding gain performance of the three decoding algorithms is tabulated in Table 4.10.

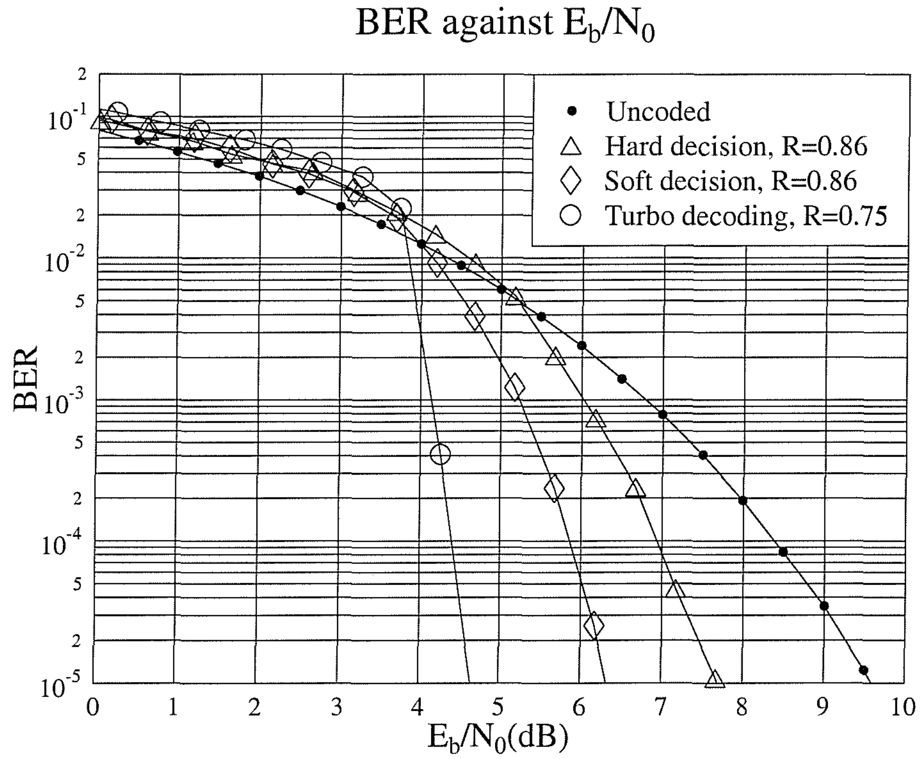


Figure 4.25: Performance comparison of hard decision, soft decision and turbo decoding of the RRNS(28,24) code over AWGN channels. For soft decision, the number of test positions was $l = 4$. Similarly, $l = 4$ was used for turbo decoding and the number of iterations was four. The turbo interleaver was a 12480 symbol 8 BPS random interleaver and values of $\alpha(j)$ and $\beta(j)$ were shown in Table 4.6.

Using the values seen in Table 4.10, we plotted the coding gain versus estimated complexity of this code in Figure 4.26. As we can see in the figure, the estimated complexity of hard decision decoding is the lowest and hence the coding gain is only 1.9 dB. With a moderate increase of estimated complexity, a coding gain of 3.3 dB is achieved by soft decision decoding. However, we have to invest a significantly higher complexity, in order to achieve a coding gain of 5 dB by turbo decoding.

4.10 Summary and Conclusion

The ancient theory of the RNS was reviewed in Section 4.1. This was followed by a brief introduction in Section 4.2.1 to the conventional number systems, namely to the decimal and the binary number systems. Then the residue number system was introduced in more detail

Algorithm	R	Complexity, δ	Coding gain (dB)
Hard decision	0.86	70,875	1.9
Soft decision	0.86	1,134,000	3.3
Turbo decoding	0.75	9,072,000	5.0

Table 4.10: Estimated decoding complexity versus coding gain at BER=10^{-5} for hard decision, soft decision and turbo decoding assisted RRNS(28,24) codes over AWGN channels.

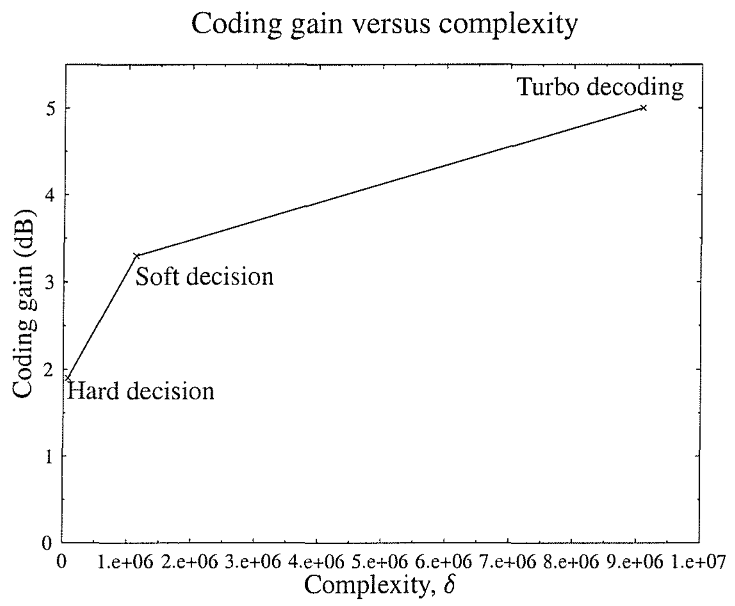


Figure 4.26: Coding gain versus estimated complexity for hard decision, soft decision and turbo decoding of RRNS(28,24) codes. The simulation results are obtained over the AWGN channels.

in Section 4.2.2. A simple example was given and the differences between the conventional number systems and the residue number system were highlighted. In Section 4.2.3, we also discussed the differences between fixed radix number systems and mixed radix number systems. The conventional decimal number system is a fixed radix number system, whereas the residue number system is a mixed radix system.

The various RNS-based arithmetic operations such as addition, subtraction and multiplication were demonstrated in Section 4.2.4 with the aid of simple examples. In Section 4.2.4.1 the multiplicative inverse was defined. Two known techniques were described in Section 4.2.5 for the conversion of operands from the residue number system to the decimal number system, which are the CRT and the MRC. In Section 4.2.6, we showed that by incorporating extra moduli in the existing residue number system, a redundant residue number system is obtained. The BEX technique was suggested for finding the residue digits of an extended set of moduli, given the residue digits of the existing moduli.

In Section 4.3, we provided a brief overview of the coding theory of RRNS codes. Given a set of moduli, we showed that the minimum free distance d_{min} of the RRNS code was given by Equation 4.37. Furthermore we also showed that maximum distance separable RRNS codes are obtained, if Equation 4.45 is satisfied. In Section 4.3.2, we highlighted, why RRNS codes are semi-linear block codes. Then, we related the minimum free distance d_{min} of RRNS codes to their error detection and error correction capabilities. The procedure for multiple error correction was summarised in Section 4.4. In Section 4.5, we demonstrated that different mapping methods result in systematic and non-systematic RRNS codes. Furthermore, we proposed a novel mapping method, which results in an increased code rates for systematic RRNS codes. The performance of the novel mapping method was found to be similar to that of the systematic RRNS codes. In Section 4.7, we modified the Chase algorithm so that it became capable of incorporating soft inputs and providing soft outputs. The estimated complexity of the hard decision and soft decision assisted decoder as well as that of iterative decoding was evaluated in Section 4.8. It was found that the estimated complexity of the algorithms depends on the codeword length n and on the error correction capability t. Moreover, the increase in estimated complexity is faster, when t is increased as compared to n.

Finally in Section 4.9 we presented our simulation results for the proposed RRNS codes using BPSK over AWGN channels. It was found in Section 4.9.1.1 that systematic RRNS codes outperform their non-systematic counterparts. Modified systematic RRNS codes give a similar performance to systematic RRNS codes at a higher code rate. The performance of the RRNS codes was then compared to that of Reed-Solomon codes in Section 4.9.1.2. It was found that analogous codes of these code families give a similar performance. Fixing the codeword length n, the performance of the RRNS codes was evaluated by varying the error correction capability t. It was shown in Figure 4.14 that the increase in error correction capability t results in an increased estimated complexity, although no extra coding gain was obtained. In Section 4.9.2.1, a soft decision aided RRNS decoder was employed. It was found that the performance of the soft decision assisted decoder improves as the the number test position is increased. We also concluded that the best-case performance of the Chase algorithm is the same as that of the soft decision Viterbi algorithm. The performance of the soft decoding of RS codes was found to be similar to that of RRNS codes. The performance of various decoding algorithms was compared in Figure 4.18 for turbo BCH(63,57) code. It was shown in the figure that the SISO Chase algorithm, which offers a significantly reduced complexity, suffers only a small performance degradation. In Section 4.9.3.2, we concluded that the optimum number of decoding iterations is four. We then extended our research into the effects of the turbo interleaver in Section 4.9.3.4. Finally, we showed in Figure 4.26 that the coding gain can be increased by increasing the estimated complexity.

In conclusion, in this chapter, we have investigated a new class of codes referred to as RRNS codes. Their performance was found to be similar to that of the analogous Reed-Solomon codes, with some additional advantages over the well-known class of Reed-Solomon codes. For example, short non-binary block codes could be readily designed using RRNS codes without having to shorten long mother codes, which would be inevitable in the context of RS codes. Furthermore, in ARQ assisted systems stronger RRNS codes are readily obtained for retransmission by transmitting extra redundant residues, without having to re-transmit the whole codeword. In [150,151,153], we exploited the same principle in designing adaptive-rate RRNS codes for adaptive OFDM transmissions over mobile communication channels.

Chapter 5

Space-Time Block Codes

5.1 Introduction

The third generation (3G) mobile communications standards [154] are expected to provide a wide range of bearer services, spanning from voice to high-rate data services, supporting rates of at least 144 kb/s in vehicular, 384 kb/s in outdoor-to-indoor and 2 Mb/s in indoor as well as picocellular applications [154].

In an effort to support such high rates, the bit/symbol capacity of band-limited wireless channels can be increased by employing multiple antennas [155]. The classic approach is to use multiple antennas at the receiver and invoke Maximum Ratio Combining (MRC) [156–158] of the received signals for improving the performance. However, applying receiver diversity at the Mobile Stations (MS) increases their complexity. Hence receiver diversity techniques typically have been applied at the Base Stations (BS). The BSs provide services for many MSs and hence up-grading the BSs is economically viable. However, the drawback of this scheme is that it only provides diversity gain for the BSs' receivers.

In the past, different transmit diversity techniques have been introduced, in order to provide diversity gain for MSs by upgrading the BSs. These transmit diversity techniques can be classified into three main categories: schemes using information feedback [159, 160], arrangements invoking feedforward or training information [161–163] and blind schemes [164, 165]. Recently, Tarokh *et al.* proposed space-time trellis coding [70, 80, 166–169] by jointly designing the channel coding, modulation, transmit diversity and the optional receiver diversity scheme. The performance criteria for designing space-time trellis codes were derived in [70], under the assumption that the channel is fading slowly and that the fading is frequency nonselective. These advances were then also extended to fast fading channels. The encoding and decoding complexity of these *space-time trellis codes* is comparable to that of conventional trellis codes [46–48] often employed in practice over non-dispersive Gaussian channels.

Space-time trellis codes [70, 80, 166–169] perform extremely well at the cost of relatively high complexity. In addressing the issue of decoding complexity, Alamouti [71] discovered a remarkable scheme for transmissions using two transmit antennas. A simple decoding algorithm was also introduced by Alamouti [71], which can be generalised to an arbitrary number of receiver antennas. This scheme is significantly less complex, than space-time trellis coding using two transmitter antennas, although there is a loss in performance [72]. Despite the associated performance penalty, Alamouti's scheme is appealing in terms of its simplicity and performance. This proposal motivated Tarokh *et al.* [72, 73] to generalise Alamouti's scheme to an arbitrary number of transmitter antennas, leading to the concept of *space-time block codes.*

Intrigued by the decoding simplicity of the space-time block codes proposed in [71–73], in this chapter, we commence our discourse by detailing their encoding and decoding process. Subsequently, we investigate the performance of the space-time block codes over perfectly interleaved, non-dispersive Rayleigh fading channels. A system which consists of space-time block codes and different channel coders will be proposed. Finally, the performance and estimated complexity of the different systems will be compared and tabulated.

Following a rudimentary introduction to space-time block codes in Section 5.3 and to channel coded space-time codes in Section 5.4, the associated estimated complexity issues and memory requirements are addressed in Section 5.4.3. The bulk of this chapter is constituted by the performance study of various space-time and channel coded transceivers in Section 5.5. Our aim is firstly to identify a space-time code, channel code combination constituting a good engineering trade-off in terms of its effective throughput, BER performance and estimated complexity in Section 5.5.1. Specifically, the issue of bit-to-symbol mapping is addressed in the context of convolution codes and convolutional coding as well as Bose-Chaudhuri-Hocquenghem (BCH) coding based turbo codes in conjunction with an attractive unity-rate space-time code and multi-level modulation in Section 5.5.2. These schemes are also benchmarked against a range of powerful trellis coded modulation (TCM) and turbo trellis coded modulation (TTCM) schemes. Our conclusions concerning the merits of the various schemes are drawn in Section 5.5.4 in the context of their coding gain versus estimated complexity.

5.2 Background

In this section, we present a brief overview of space-time block codes by considering the classical Maximum Ratio Combining (MRC) technique [71, 102, 170]. The introduction of this classical technique is important, since at a later stage it will assist us in highlighting the philosophy of space-time block codes.

5.2.1 Maximum Ratio Combining

In conventional transmission systems we have a single transmitter, which transmits information to a single receiver. In Rayleigh fading channels the transmitted symbols experience severe magnitude fluctuation and phase rotation. In order to mitigate this problem, we can employ several receivers that receive replicas of the same transmitted symbol through independent fading paths. Even if a particular path is severely faded, we may still be able to recover a reliable estimate of the transmitted symbols through other propagation paths. However, at the station we have to combine the received symbols of the different propagation paths, which involves additional complexity. Again, the classical method often used in practice is referred to as the MRC technique [71, 102, 170].

Figure 5.1 shows the baseband representation of the classical MRC technique in conjunction with two receivers. At a particular instant, a symbol x is transmitted. As we can see from the figure, the transmitted symbol x propagates through two different channels, namely h_1 and h_2. For simplicity, all channels are assumed to be constituted by a single propagation path and can be modelled as complex multiplicative distortion, which consists of a magnitude and phase response given as follows:

$$h_1 = |h_1|e^{j\theta_1} \tag{5.1}$$

$$h_2 = |h_2|e^{j\theta_2} , \tag{5.2}$$

where $|h_1|$, $|h_2|$ are the fading magnitudes and θ_1, θ_2 are the phase values. Noise is added by each receiver, as shown in Figure 5.1. Hence, the resulting received baseband signals are:

$$y_1 = h_1x + n_1 \tag{5.3}$$

$$y_2 = h_2x + n_2 , \tag{5.4}$$

where n_1 and n_2 are complex noise samples. In matrix form this can be written as follows:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = x \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} + \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} . \tag{5.5}$$

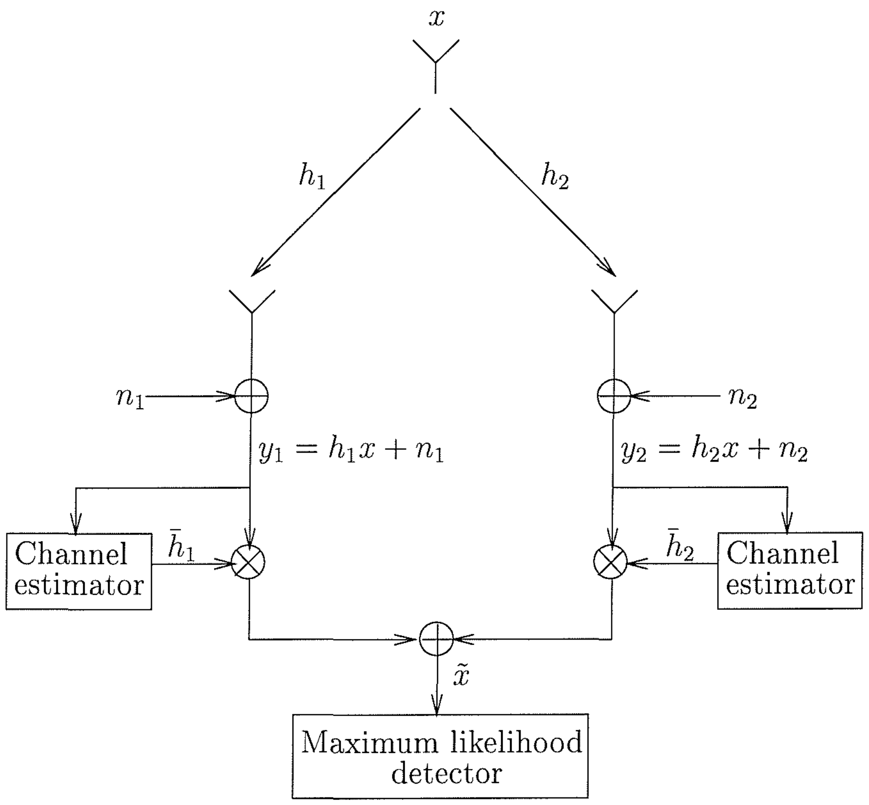


Figure 5.1: Baseband representation of the MRC technique using two receivers.

Assuming that we have perfect channel information, i.e. a perfect channel estimator, the received signals y_1 and y_2 can be multiplied by the conjugate of the complex channel transfer functions $\bar{h}_1$ and $\bar{h}_2$, respectively, in order to remove the channel's effects. Then the corresponding signals are combined at the input of the maximum likelihood detector of Figure 5.1 as follows:

$$\begin{aligned} \tilde{x} &= \bar{h}_1 y_1 + \bar{h}_2 y_2 \\ &= \bar{h}_1 h_1 x + \bar{h}_1 n_1 + \bar{h}_2 h_2 x + \bar{h}_2 n_2 \\ &= \left(|h_1|^2 + |h_2|^2\right) x + \bar{h}_1 n_1 + \bar{h}_2 n_2 \ . \end{aligned} \tag{5.6}$$

The combined signal $\tilde{x}$ is then passed to the maximum likelihood detector, as shown in Figure 5.1. Based on the Euclidean distances between the combined signal $\tilde{x}$ and all possible transmitted symbols, the most likely transmitted symbol is determined by the maximum likelihood detector. The simplified decision rule is based on choosing x_i if and only if

$$dist(\tilde{x}, x_i) \leq dist(\tilde{x}, x_j), \quad \forall i \neq j, \tag{5.7}$$

where $dist(A, B)$ is the Euclidean distance between signals A, B and the index j spans all possible transmitted signals. From Equation 5.7 we can see that maximum likelihood transmitted symbol is the one having the minimum Euclidean distance from the combined signal $\tilde{x}$.

5.3 Space-Time Block Codes

In the previous section we have briefly introduced the classic MRC technique. In this section we will present the basic principles of space-time block codes. In analogy to the MRC matrix formula of Equation 5.5, a space-time block code describing the relationship between the original transmitted signal x and the signal replicas artificially created at the transmitter for transmission over various diversity channels, is defined by an $n \times p$ dimensional transmission matrix. The entries of the matrix are constituted by linear combinations of the $k-$ary input symbols $x_1, x_2, ..., x_k$ and their conjugates. The $k-$ary input symbols x_i $i = 1...k$, are used to represent the information-bearing binary bits to be transmitted over the transmit diversity channels. In a signal constellation having 2^b constellation points, b number of binary bits are used to represent a symbol x_i. Hence, a block of $k \times b$ binary bits are entered into the space-time block encoder at a time and it is therefore referred to as a space-time block code. The number of transmitter antennas is p and n represents the number of time slots used to transmit k input symbols. Hence, a general form of the transmission matrix of a space-time block code is as follows:

$$\begin{pmatrix} g_{11} & g_{21} & \cdots & g_{p1} \\ g_{12} & g_{22} & \cdots & g_{p2} \\ . & . & . & . \\ . & . & . & . \\ g_{1n} & g_{2n} & \cdots & g_{pn} \end{pmatrix}, \tag{5.8}$$

where the entries g_{ij} represent linear combinations of the symbols $x_1, x_2, ..., x_k$ and their conjugates. More specifically, the entries g_{ij}, where $i = 1...p$, are transmitted simultaneously from transmit antennas $1, ..., p$ in each time slot $j = 1, ..., n$. For example, in time slot $j = 2$, signals $g_{12}, g_{22}, ..., g_{p2}$ are transmitted simultaneously from transmit antennas $Tx\ 1, Tx\ 2, ..., Tx\ p$. We can see in the transmission matrix defined in Equation 5.8 that encoding is carried out in both space and time; hence the term space-time coding.

The $n \times p$ transmission matrix in Equation 5.8 – which defines the space-time block code – is based on a complex generalised orthogonal design, as defined in [71–73]. Since there are k symbols transmitted over n time slots, the code rate of the space-time block code is given by:

$$R = k/n. \tag{5.9}$$

At the receiving end, we can have an arbitrary number of q receivers. It was shown in [71] that the associated diversity order is $p \times q$. A combining technique [71,73] similar to MRC can be applied at the receiving end, which may be generalised to q number of receivers. At the current state-of-the-art the associated diversity channels are often assumed to be flat fading channels. A possible approach to satisfying this condition for high-rate transmissions over frequency-selective channels is to split the high-rate bit stream into a high number of low-rate streams transmitted over flat-fading subchannels This can be achieved with the aid of Orthogonal Frequency Division Multiplexing (OFDM) [153], It is also typically assumed that the complex fading envelope is constant over n consecutive time slots.

5.3.1 A Twin-Transmitter Based Space-Time Block Code

As mentioned above, the simplest form of space-time block codes was proposed by Alamouti in [71], which is a simple twin-transmitter based scheme associated with $p = 2$. The transmission matrix is defined as follows:

$$\mathbf{G}_2 = \begin{pmatrix} x_1 & x_2 \\ -\bar{x}_2 & \bar{x}_1 \end{pmatrix} . \tag{5.10}$$

We can see in the transmission matrix $\mathbf{G}_2$ that there are $p = 2$ (number of columns in the matrix $\mathbf{G}_2$) transmitters, $k = 2$ possible input symbols, namely x_1, x_2, and the code spans over $n = 2$ (number of rows in the matrix $\mathbf{G}_2$) time slots. Since $k = 2$ and $n = 2$, the code rate given by Equation 5.9 is unity. The associated encoding and transmission process is shown in Table 5.1. At any given time instant T, two signals are simultaneously

Time	antenna	
slot, T	Tx 1	Tx 2
1	x_1	x_2
2	$-\bar{x}_2$	$\bar{x}_1$

Table 5.1: The encoding and transmission process for the $\mathbf{G}_2$ space-time block code of Equation 5.10.

transmitted from the antennas Tx 1 and Tx 2. For example, in the first time slot associated with $T = 1$, signal x_1 is transmitted from antenna Tx 1 and *simultaneously* signal x_2 is transmitted from antenna Tx 2. In the next time slot corresponding to $T = 2$, signals $-\bar{x}_2$ and $\bar{x}_1$ (the conjugates of symbols x_1 and x_2) are simultaneously transmitted from antennas Tx 1 and Tx 2, respectively.

5.3.1.1 The Space-Time Code $\mathbf{G}_2$ Using One Receiver

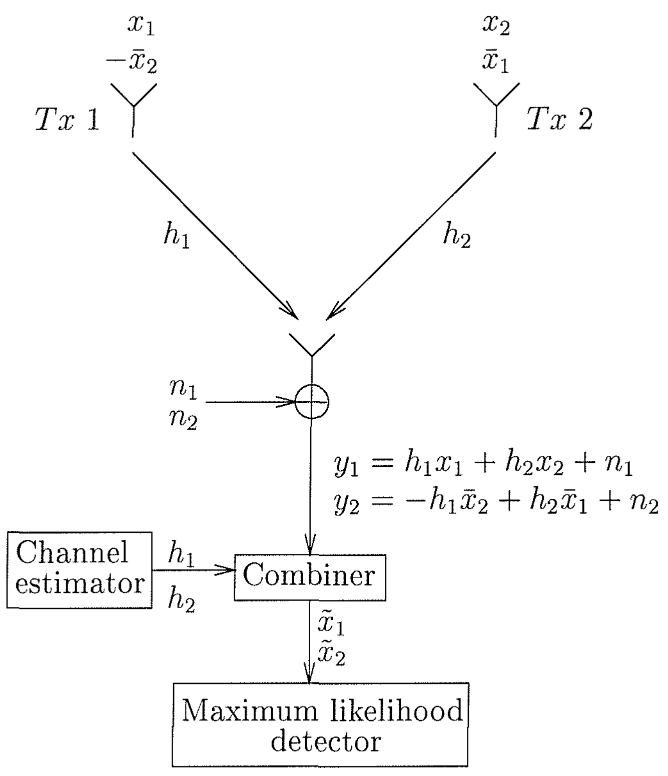


Figure 5.2: Baseband representation of the simple twin-transmitter space-time block code $\mathbf{G}_2$ of Equation 5.10 using one receiver.

Lets us now consider an example of encoding and decoding the $\mathbf{G}_2$ space-time block code of Equation 5.10 using one receiver. This example can be readily extended to an arbitrary number of receivers. In Figure 5.2 we show the baseband representation of a simple two-transmitter space-time block code, namely that of the $\mathbf{G}_2$ code seen in Equation 5.10 using one receiver. We can see from the figure that there are two transmitters, namely Tx 1 as well as Tx 2 and they transmit two signals simultaneously. As mentioned earlier, the complex fading envelope is assumed to be constant across the corresponding two consecutive time slots. Therefore, we can write

$$h_1 = h_1(T=1) = h_1(T=2) \tag{5.11}$$

$$h_2 = h_2(T=1) = h_2(T=2)\ . \tag{5.12}$$

Independent noise samples are added at the receiver in each time slot and hence the received signals can be expressed with the aid of Equation 5.10 as:

$$\begin{aligned} y_1 &= h_1 x_1 + h_2 x_2 + n_1 & (5.13)\\ y_2 &= -h_1 \bar{x}_2 + h_2 \bar{x}_1 + n_2 \ , & (5.14) \end{aligned}$$

where y_1 is the first received signal and y_2 is the second. Notice that the received signal y_1 consists of the transmitted signal x_1 and x_2, while y_2 of their conjugates. In order to determine the transmitted symbols, we have to extract the signals x_1 and x_2 from the received signals y_1 and y_2. Therefore, both signals y_1 and y_2 are passed to the combiner, as shown in Figure 5.2. In the combiner – aided by the channel estimator, which provides perfect estimation of the diversity channels in this example – simple signal processing is performed in order to separate the signals x_1 and x_2. Specifically, in order to extract the signal x_1, we combine the received signals y_1 and y_2 as follows:

$$\begin{aligned} \tilde{x}_1 &= \bar{h}_1 y_1 + h_2 \bar{y}_2 \\ &= \bar{h}_1 h_1 x_1 + \bar{h}_1 h_2 x_2 + \bar{h}_1 n_1 - h_2 \bar{h}_1 x_2 + h_2 \bar{h}_2 x_1 + h_2 \bar{n}_2 \\ &= \left(|h_1|^2 + |h_2|^2\right) x_1 + \bar{h}_1 n_1 + h_2 \bar{n}_2 \ . & (5.15) \end{aligned}$$

Similarly, for signal x_2 we generate:

$$\begin{aligned} \tilde{x}_2 &= \bar{h}_2 y_1 - h_1 \bar{y}_2 \\ &= \bar{h}_2 h_1 x_1 + \bar{h}_2 h_2 x_2 + \bar{h}_2 n_1 + h_1 \bar{h}_1 x_2 - h_1 \bar{h}_2 x_1 - h_1 \bar{n}_2 \\ &= \left(|h_1|^2 + |h_2|^2\right) x_2 + \bar{h}_2 n_1 - h_1 \bar{n}_2 \ . & (5.16) \end{aligned}$$

Clearly, from Equation 5.15 and 5.16 we can see that we have separated the signals x_1 and x_2 by simple multiplications and additions. Due to the orthogonality of the space-time block code $\mathbf{G}_2$ in Equation 5.10 [72], the unwanted signal x_2 is cancelled out in Equation 5.15 and vice versa, signal x_1 is removed from Equation 5.16. Both signals $\tilde{x}_1$ and $\tilde{x}_2$ are then passed to the maximum likelihood detector of Figure 5.2, which applies Equation 5.7 to determine the most likely transmitted symbols.

From Equations 5.15 and 5.16 we can derive a simple rule of thumb for manipulating the received signal in order to extract a symbol x_i. For each received signal y_j, we would have a linear combination of the transmitted signals x_i convolved with the corresponding Channel Impulse Response (CIR) h_i. The non-dispersive CIR is assumed to be constituted by a single CIR tap corresponding to a complex multiplicative factor. The conjugate of the CIR $\bar{h}_i$ should be multiplied with the received signal y_j, if x_i is in the expression of the received signal y_j. However, if the conjugate of x_i, namely $\bar{x}_i$ is present in the expression, we should then multiply the CIR h_i with the conjugate of the received signal y_j, namely $\bar{y}_j$. The product should then be added to or subtracted from the rest, depending on the sign of the term in the expression of the received signal y_j.

5.3.1.2 The Space-Time Code $\mathbf{G}_2$ Using Two Receivers

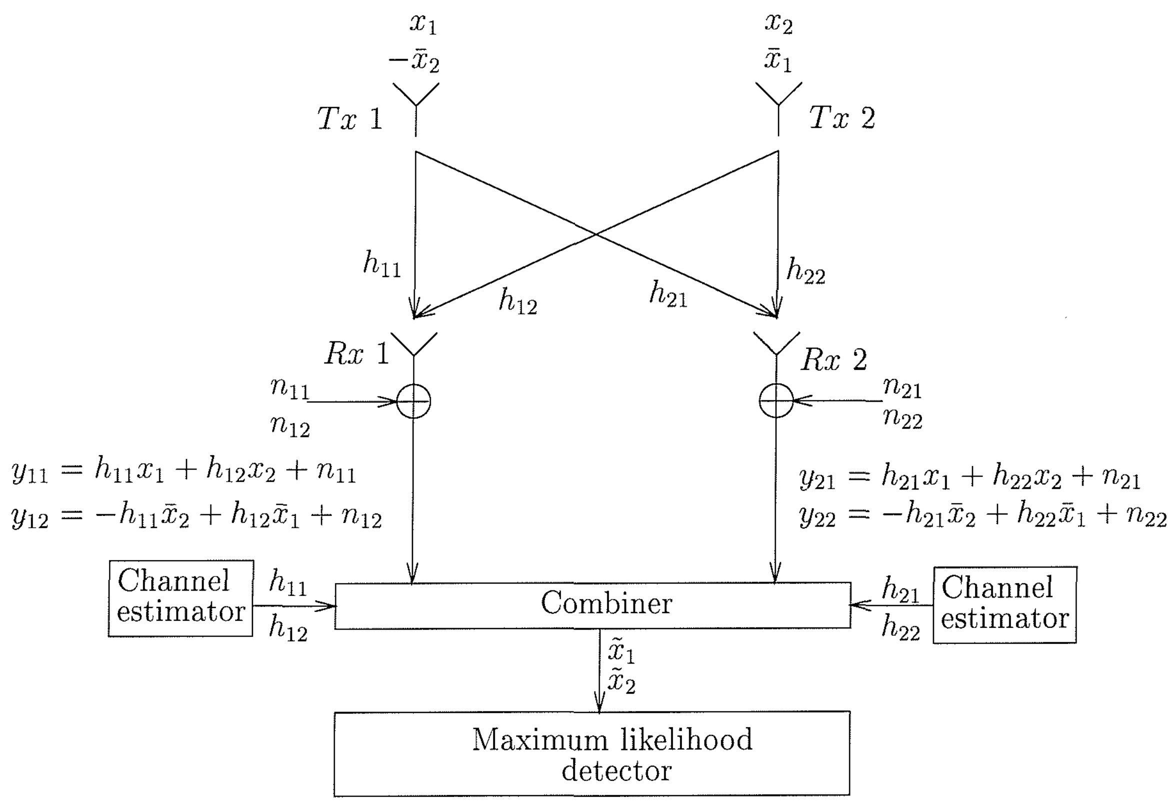


Figure 5.3: Baseband representation of the simple twin-transmitter space-time block code $\mathbf{G}_2$ of Equation 5.10 using two receiver.

In Section 5.3.1.1 we have shown an example of the encoding and decoding process for the $\mathbf{G}_2$ space-time block code of Equation 5.10 using one receiver. However, this example can be readily extended to an arbitrary number of receivers. The encoding and transmission sequence will be identical to the case of a single receiver. For illustration, we discuss the specific case of two transmitters and two receivers, as shown in Figure 5.3. We will show however that the generalisation to q receivers is straightforward. In Figure 5.3, the subscript i in the notation h_{ij}, n_{ij} and y_{ij} represents the receiver index. By contrast, the subscript j denotes the transmitter index in the CIR h_{ij}, but it denotes the time slot T in n_{ij} and y_{ij}. Therefore, at the first receiver Rx 1 we have:

$$y_{11} = h_{11}x_1 + h_{12}x_2 + n_{11} \tag{5.17}$$

$$y_{12} = -h_{11}\bar{x}_2 + h_{12}\bar{x}_1 + n_{12}\ , \tag{5.18}$$

while at receiver Rx 2 we have

$$\begin{aligned} y_{21} &= h_{21}x_1 + h_{22}x_2 + n_{21} && (5.19)\\ y_{22} &= -h_{21}\bar{x}_2 + h_{22}\bar{x}_1 + n_{22}\ . && (5.20)\end{aligned}$$

We can, however, generalise these equations to:

$$\begin{aligned} y_{i1} &= h_{i1}x_1 + h_{i2}x_2 + n_{i1} && (5.21)\\ y_{i2} &= -h_{i1}\bar{x}_2 + h_{i2}\bar{x}_1 + n_{i2}\ , && (5.22)\end{aligned}$$

where $i = 1, ..., q$ and q is the number of receivers, which is equal to two in this example. At the combiner of Figure 5.3, the received signals are combined to extract the transmitted signals x_1 and x_2 from the received signals y_{11}, y_{12}, y_{21} and y_{22}, as follows:

$$\begin{aligned} \tilde{x}_1 &= \bar{h}_{11}y_{11} + h_{12}\bar{y}_{12} + \bar{h}_{21}y_{21} + h_{22}\bar{y}_{22} && (5.23)\\ \tilde{x}_2 &= \bar{h}_{12}y_{11} - h_{11}\bar{y}_{12} + \bar{h}_{22}y_{21} - h_{21}\bar{y}_{22}\ . && (5.24)\end{aligned}$$

Again, we can generalise the above expressions to q receivers, yielding:

$$\begin{aligned} \tilde{x}_1 &= \sum_{i=1}^{q}\left(\bar{h}_{i1}y_{i1} + h_{i2}\bar{y}_{i2}\right) && (5.25)\\ \tilde{x}_2 &= \sum_{i=1}^{q}\left(\bar{h}_{i2}y_{i1} - h_{i1}\bar{y}_{i2}\right)\ . && (5.26)\end{aligned}$$

Finally, we can simplify Equations 5.23 and 5.24 to:

$$\begin{aligned} \tilde{x}_1 &= \left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right)x_1 + \bar{h}_{11}n_{11} + h_{12}\bar{n}_{12} + \bar{h}_{21}n_{21} + h_{22}\bar{n}_{22} && (5.27)\\ \tilde{x}_2 &= \left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right)x_2 + \bar{h}_{12}n_{11} - h_{11}\bar{n}_{12} + \bar{h}_{22}n_{21} - h_{21}\bar{n}_{22}\ . && (5.28)\end{aligned}$$

In the generalised form of q receivers we have:

$$\begin{aligned} \tilde{x}_1 &= \sum_{i=1}^{q}\left[\left(|h_{i1}|^2 + |h_{i2}|^2\right)x_1 + \bar{h}_{i1}n_{i1} + h_{i2}\bar{n}_{i2}\right] && (5.29)\\ \tilde{x}_2 &= \sum_{i=1}^{q}\left[\left(|h_{i1}|^2 + |h_{i2}|^2\right)x_2 + \bar{h}_{i2}n_{i1} - h_{i1}\bar{n}_{i2}\right]\ . && (5.30)\end{aligned}$$

Signals $\tilde{x}_1$ and $\tilde{x}_2$ are finally derived and passed to the maximum likelihood detector seen in Figure 5.3. Again, Equation 5.7 is applied to determine the maximum likelihood transmitted symbols.

We observe in Equation 5.29 that signal x_1 is multiplied by a term related to the fading amplitudes, namely $|h_{i1}|^2 + |h_{i2}|^2$. Hence, in order to acquire a high reliability signal $\tilde{x}_1$, the amplitudes of the CIRs must be high. If the number of receivers is equal to one, i.e. $q = 1$, then Equation 5.29 is simplified to Equation 5.15. In Equation 5.15, we can see that there are two fading amplitude terms, i.e. two independent paths associated with transmitting the symbol x_1. Therefore, if either of the paths is in a deep fade, the other path still may provide a high-reliability for the transmitted signal x_1. This explains, why the performance of a system having two transmitters and one receiver is better, than that of the system employing one transmitter and one receiver. On the other hand, in the conventional single-transmitter, single-receiver system there is only a single propagation path, which may be severely attenuated by a deep fade. To elaborate further, if the number of receivers is increased to $q = 2$, Equation 5.27 accrues from Equation 5.29. We can see in Equation 5.27 that there are now twice as many propagation paths, as in Equation 5.15. This increases the probability of providing a high reliability for the signal $\tilde{x}_1$.

5.3.2 Other Space-Time Block Codes

In Section 5.3.1 we have detailed Alamouti's simple two-transmitter space-time block code namely the $\mathbf{G}_2$ code of Equation 5.10. This code is significantly less complex, than the space-time trellis codes of [70, 80, 166–169] using two transmit antennas. However, again, there is a performance loss compared to the space-time trellis codes of [70, 80, 166–169]. Despite its performance loss, Alamouti's scheme [71] is appealing in terms of its simplicity. This motivated Tarokh *et al.* [72] to search for similar schemes using more than two transmit antennas. In [72] the theory of orthogonal code design was invoked, in order to construct space-time block codes having more than two transmitters. The half-rate space-time block code employing three transmitters was defined as [72]:

$$\mathbf{G}_3 = \begin{pmatrix} x_1 & x_2 & x_3 \\ -x_2 & x_1 & -x_4 \\ -x_3 & x_4 & x_1 \\ -x_4 & -x_3 & x_2 \\ \bar{x}_1 & \bar{x}_2 & \bar{x}_3 \\ -\bar{x}_2 & \bar{x}_1 & -\bar{x}_4 \\ -\bar{x}_3 & \bar{x}_4 & \bar{x}_1 \\ -\bar{x}_4 & -\bar{x}_3 & \bar{x}_2 \end{pmatrix}, \tag{5.31}$$

and the four-transmitter half-rate space-time block code was specified as [72]:

$$\mathbf{G}_4 = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \\ \bar{x}_1 & \bar{x}_2 & \bar{x}_3 & \bar{x}_4 \\ -\bar{x}_2 & \bar{x}_1 & -\bar{x}_4 & \bar{x}_3 \\ -\bar{x}_3 & \bar{x}_4 & \bar{x}_1 & -\bar{x}_2 \\ -\bar{x}_4 & -\bar{x}_3 & \bar{x}_2 & \bar{x}_1 \end{pmatrix}. \tag{5.32}$$

By employing the space-time block codes $\mathbf{G}_3$ and $\mathbf{G}_4$, we can see that the bandwidth efficiency has been reduced by a factor of two compared to the space-time block code $\mathbf{G}_2$. Besides, the number of transmission slots across which the channels is required to have a constant fading envelope is eight, namely four times higher, than that of the space-time code $\mathbf{G}_2$.

In order to increase the associated bandwidth efficiency, Tarokh *et al.* constructed the rate 3/4 so-called generalised complex orthogonal sporadic codes [72,73]. The corresponding rate 3/4 three-transmitter space-time block code is given by [72]:

$$\mathbf{H}_3 = \begin{pmatrix} x_1 & x_2 & \frac{x_3}{\sqrt{2}} \\ -\bar{x}_2 & \bar{x}_1 & \frac{x_3}{\sqrt{2}} \\ \frac{\bar{x}_3}{\sqrt{2}} & \frac{\bar{x}_3}{\sqrt{2}} & \frac{(-x_1-\bar{x}_1+x_2-\bar{x}_2)}{2} \\ \frac{\bar{x}_3}{\sqrt{2}} & -\frac{\bar{x}_3}{\sqrt{2}} & \frac{(x_2+\bar{x}_2+x_1-\bar{x}_1)}{2} \end{pmatrix}, \tag{5.33}$$

while the rate 3/4 four-transmitter space-time block code is defined as [72]:

$$\mathbf{H}_4 = \begin{pmatrix} x_1 & x_2 & \frac{x_3}{\sqrt{2}} & \frac{x_3}{\sqrt{2}} \\ -\bar{x}_2 & \bar{x}_1 & \frac{x_3}{\sqrt{2}} & -\frac{x_3}{\sqrt{2}} \\ \frac{\bar{x}_3}{\sqrt{2}} & \frac{\bar{x}_3}{\sqrt{2}} & \frac{(-x_1-\bar{x}_1+x_2-\bar{x}_2)}{2} & \frac{(-x_2-\bar{x}_2+x_1-\bar{x}_1)}{2} \\ \frac{\bar{x}_3}{\sqrt{2}} & -\frac{\bar{x}_3}{\sqrt{2}} & \frac{(x_2+\bar{x}_2+x_1-\bar{x}_1)}{2} & \frac{(-x_1-\bar{x}_1-x_2+\bar{x}_2)}{2} \end{pmatrix}. \tag{5.34}$$

In Table 5.2 we summarise the parameters associated with all space-time block codes proposed by Alamouti [71] and Tarokh, Jafarkhani as well as Calderbank [72, 73]. The decoding algorithms and the corresponding performance results of the space-time block codes were given in [73].

5.3.3 MAP Decoding of Space-Time Block Codes

Recently Bauch [171] derived a simple symbol-by-symbol Maximum-A-Posteriori (MAP) decoding rule for space-time block codes. The soft-outputs provided by the space-time

Space-time code	Rate	No. of transmitters, p	No. of input symbols, k	Code span, n
$\mathbf{G}_2$	1	2	2	2
$\mathbf{G}_3$	1/2	3	4	8
$\mathbf{G}_4$	1/2	4	4	8
$\mathbf{H}_3$	3/4	3	3	4
$\mathbf{H}_4$	3/4	4	3	4

Table 5.2: Table of different space-time block codes.

MAP decoder can be used as the input to channel decoders, such as for example turbo codes, which may be concatenated for further improving the system's performance.

By using Bayes' rule, the a-posteriori probability of the transmitted k-ary symbols $x_1, ..., x_k$ given the received signals $y_{11}, ..., y_{1n}, y_{21}, ..., y_{qn}$ can be expressed as [171]:

$$P\left(x_1, ..., x_k | y_{11}, ..., y_{qn}\right) = P\left(y_{11}, ..., y_{qn} | x_1, ..., x_k\right) \cdot P\left(x_1, ..., x_k\right) \ , \qquad (5.35)$$

where $P\left(x_1, ..., x_k\right)$ is the associated *a-priori* information, of the transmitted symbols which can be obtained from other independent sources, for example from channel decoders. Furthermore, according to Bauch [171], over non-dispersive Rayleigh fading channels we have:

$$P\left(y_{11}, ..., y_{qn} | x_1, ..., x_k\right) = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^{qn}} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{l=1}^{q} \sum_{i=1}^{n} \left| y_{li} - \sum_{j=1}^{p} h_{lj} g_{ji} \right|^2 \right\} \ , \qquad (5.36)$$

where σ is the noise variance and g_{ij} are the entries of the transmission matrix in Equation 5.8. We can, however, simplify Equation 5.35 and obtain the expression for the a-posteriori probability of each transmitted symbol x_i as [171]:

$$P\left(x_i | y_{11}, ..., y_{qn}\right) = P\left(y_{11}, ..., y_{qn} | x_i\right) \cdot P(x_i) \ , \qquad (5.37)$$

where $i = 1, ..., k$.

Let us now consider as an example the simplest possible space-time code, namely $\mathbf{G}_2$ associated with $k = 2$, $n = 2$ and $p = 2$. Assuming that there is no a-priori information, i.e. that $P(x_1, ..., x_k) = C$ where C is a constant, we obtain the a-posteriori information of the transmitted k-ary symbols from Equation 5.35 and 5.36, as [171]:

$$\begin{aligned}
&P\left(x_1,...,x_k|y_{11},...,y_{qn}\right)\\
&= C\cdot\frac{1}{\left(\sigma\sqrt{2\pi}\right)^{qn}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[\left|y_{l1}-\sum_{j=1}^{p}h_{lj}g_{j1}\right|^2+\left|y_{l2}-\sum_{j=1}^{p}h_{lj}g_{j2}\right|^2\right]\right\}\\
&= C'\cdot\exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[\left|y_{l1}-h_{l1}g_{11}-h_{l2}g_{21}\right|^2+\left|y_{l2}-h_{l1}g_{12}-h_{l2}g_{22}\right|^2\right]\right\}\\
&= C'\cdot\exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[\left|y_{l1}-h_{l1}x_1-h_{l2}x_2\right|^2+\left|y_{l2}+h_{l1}\bar{x}_2-h_{l2}\bar{x}_1\right|^2\right]\right\}, \quad (5.38)
\end{aligned}$$

where $C' = C\cdot\frac{1}{\left(\sigma\sqrt{2\pi}\right)^{qn}}$. In order to obtain the expression of the a-posteriori probability for symbol x_1, x_2-related terms can be eliminated in Equation 5.38 due to the orthogonality of the code, arriving at:

$$\begin{aligned}
&P\left(x_1|y_{11},...,y_{q2}\right)\\
&= C'\cdot\exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[\left|y_{l1}-h_{l1}x_1\right|^2+\left|y_{l2}-h_{l2}\bar{x}_1\right|^2\right]\right\}\\
&= C''\cdot\exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[-h_{l1}x_1\bar{y}_{l1}-\bar{h}_{l1}\bar{x}_1y_{l1}-h_{l2}\bar{x}_1\bar{y}_{l2}-\bar{h}_{l2}x_1y_{l2}+|x_1|^2\sum_{i=1}^{2}|h_{li}|^2\right]\right\}, \quad (5.39)
\end{aligned}$$

where $|y_{l1}|^2$ and $|y_{l2}|^2$ are constants, which do not depend on x_1 and hence incorporated into C''. Following a few further manipulations, we can simplify Equation 5.39 to:

$$\begin{aligned}
&P\left(x_1|y_{11},...,y_{q2}\right)\\
&= C\cdot\exp\left\{-\frac{1}{2\sigma^2}\left[\left|\left[\sum_{l=1}^{q}\left(\bar{h}_{l1}y_{l1}+h_{l2}\bar{y}_{l2}\right)\right]-x_1\right|^2+\left(-1+\sum_{l=1}^{q}\sum_{i=1}^{2}|h_{li}|^2\right)|x_1|\right]\right\}. \quad (5.40)
\end{aligned}$$

Similarly, we can eliminate the x_1-related terms in Equation 5.38 and simplify it to:

$$\begin{aligned}
&P\left(x_2|y_{11},...,y_{q2}\right)\\
&= C\cdot\exp\left\{-\frac{1}{2\sigma^2}\left[\left|\left[\sum_{l=1}^{q}\left(\bar{h}_{l2}y_{l1}-h_{l1}\bar{y}_{l2}\right)\right]-x_2\right|^2+\left(-1+\sum_{l=1}^{q}\sum_{i=1}^{2}|h_{li}|^2\right)|x_2|\right]\right\}. \quad (5.41)
\end{aligned}$$

It can be seen that Equations 5.40 and 5.41 resemble the equations given in [73] for the maximum likelihood decoding of the space-time code $\mathbf{G}_2$. Besides considering the space-time code $\mathbf{G}_2$, the maximum likelihood decoding algorithms were also given for the space-time codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ in [73]. It can be shown that Bauch's MAP algorithms

[171] applicable to the space-time codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ also resemble the maximum likelihood algorithms given in [73].

5.4 Channel Coded Space-Time Block Codes

In Section 5.3, we have given a detailed illustration of the concept of space-time block codes. The MAP decoding rules were also applied to space-time block codes in Section 5.3.3. This enables the space-time decoder to provide soft-outputs, which in turn can be used by the concatenated channel decoders. Hence, in this section we concatenate space-time block codes with Convolutional Codes (CC) [3,172,173], Turbo Convolutional (TC) codes [12,13], Turbo BCH (TBCH) codes [61], Trellis Coded Modulation (TCM) [46,47] and Turbo Trellis Coded Modulation (TTCM) [57]. The performance and estimated complexity of each scheme will be studied and compared. We will also address the issue of mapping channel coded bits of the TC and TBCH schemes to different protection classes in multilevel modulation [153].

Convolutional codes (CC) were first suggested by Elias [3] in 1955. The so-called Viterbi Algorithm (VA) was proposed by Viterbi [8,9] in 1967 for the maximum likelihood decoding of convolutional codes. As an alternative decoder, the more complex Maximum A-Posteriori (MAP) algorithm was proposed by Bahl [11], which provided the optimum Bit Error Rate (BER) performance, although this was not significantly better than that of the Viterbi algorithm. In the early 1970s, convolutional codes were used in deep-space and satellite communications. They were then also adopted by the Global System of Mobile communications (GSM) [49] for the pan-European digital cellular mobile radio system.

In 1993, Berrou *et al.* [12,13] proposed a novel channel code, referred to as a turbo code. As detailed in Chapter 3, the turbo encoder consists of two component encoders. Generally, convolutional codes are used as the component encoders, and the corresponding turbo codes are termed here as a TC code. However, BCH [49,86] codes can also be employed as their component codes, resulting in the so-called turbo BCH codes (TBCH). They have been shown for example by Hagenauer [61,63] to perform impressively at near-unity coding rates, although at a higher decoding complexity than that of the corresponding-rate TCs.

In 1987, Ungerboeck [46,47] invented trellis coded modulation (TCM) by combining the design of channel coding and modulation. TCM optimises the Euclidean distance between codewords and hence maximises the coding gain. In [57], Robertson *et al.* applied the basic idea of turbo codes [12,13] to TCM by retaining the important properties and advantages of both structures. In the resultant TTCM scheme, two Ungerboeck codes [46,47] are employed in combination with TCM as component codes in an overall structure similar to that of turbo codes.

5.4.1 System Overview

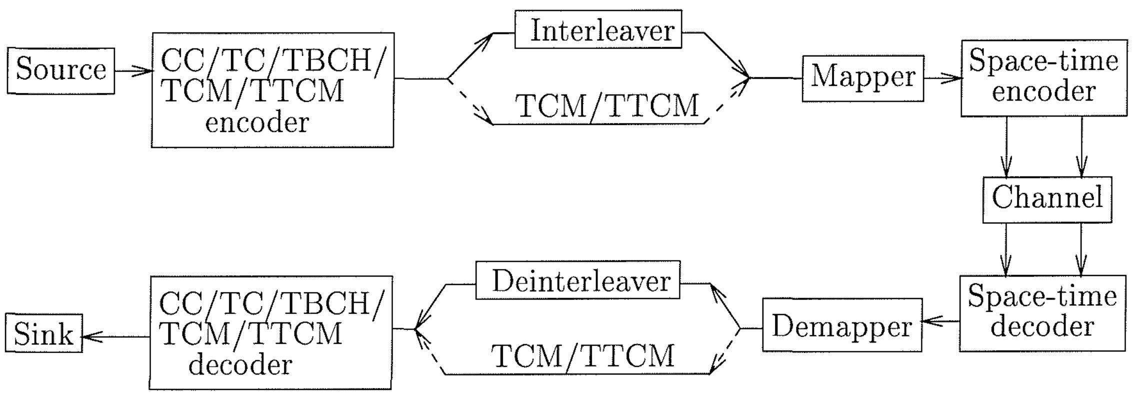


Figure 5.4: System overview of space-time block codes and different channel coding schemes.

The schematic of the proposed concatenated space-time block codes and the different channel coding schemes is shown in Figure 5.4. As mentioned above, the investigated channel coding schemes are CC, TC codes, TBCH codes, TCM and TTCM. The information source at the transmitter of Figure 5.4 generates random data bits. The information bits are then encoded by each of the above five different channel coding schemes. However, only the output binary bits of the CC, TBCH and TC coding schemes are channel interleaved, as seen in Figure 5.4. The role of the interleaver will be detailed in Section 5.5.2.

The output bits of the TCM and TTCM scheme are passed directly to the mapper in Figure 5.4, which employs two different mapping techniques. Gray-mapping [55, 86] is used for the CC, TBCH and TC schemes, whereas set-partitioning [46–48, 57] is utilised for the TCM and TTCM scheme. Different modulation schemes are employed, namely Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), 8-level Phase Shift Keying (8PSK), 16-level Quadrature Amplitude Modulation (16QAM) and 64-level Quadrature Amplitude Modulation (64QAM) [153].

Following the mapper, the channel coded symbols are passed to the space-time block encoder, as shown in Figure 5.4. Below, we will investigate the performance of all the previously mentioned space-time block codes, namely that of the $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ codes proposed in [71–73]. The corresponding transmission matrices are given in Equations 5.10, 5.31, 5.32, 5.33 and 5.34, respectively. The coding rate and number of transmitters of the associated space-time block codes is shown in Table 5.2. The channels are uncorrelated or – synonymously – perfectly interleaved narrow-band or non-dispersive Rayleigh fading channels. This assumption does not contradict the requirement for a constant channel magnitude and phase over p (number of rows in the transmission matrix) consecutive symbols, since upon applying a sufficiently high channel interleaving depth the channels' fading envelope can become indeed near-uncorrelated. We assumed that the narrow-band fading amplitudes received from each transmitter antenna were mutually uncorrelated Rayleigh distributed processes. The average signal power received from each transmitter antenna was the same. Furthermore, we assumed that the receiver had a perfect estimate of the channels' fading amplitudes. In practice, the channels' fading amplitude can be estimated for example with the aid of pilot symbols [153].

At the receiver, the number of receiver antennas constitutes a design parameter, which was fixed to one, unless specified otherwise. The space-time block decoders then apply the MAP or Log-MAP decoding algorithm of Section 5.3.3 for the decoding of the signals received from the different antennas. Due to its implementational simplicity, the Log-MAP decoding algorithm is preferred in the proposed system. The soft outputs associated with the received bits or symbols are passed through the channel deinterleaver or directly to the TCM/TTCM decoder, respectively, as seen in Figure 5.4. The channel-deinterleaved soft outputs of the received bits are then passed to the CC, TC or TBCH decoders. The Viterbi algorithm [8, 9] is applied in the CC and TCM decoder. By contrast, all turbo decoder schemes apply the Log-MAP [13, 57, 61] decoding algorithm. The decoded bits are finally passed to the information sink for calculation of the BER, as shown in Figure 5.4.

5.4.2 Channel Codec Parameters

In Figure 5.4, we have given an overview of the proposed system. As we can see in the figure, there are different channel encoders to be considered, namely the CC, TC, TBCH, TCM and TTCM schemes. In this section, we present the parameters of all the channel codecs to be used in our investigations.

Table 5.3 shows the parameters of each channel encoder proposed in the system. We commence with the most well-known channel code, namely the convolutional code. A convolutional code is described by three parameters n, k and K and it is denoted as CC(n,k,K). At each instant, a CC(n,k,K) encoder accepts k input bits and outputs n coded bits. The constraint length of the code is K and the number of encoder states is equal to 2^{K-1}. The

Code	Octal generator polynomial	No. of states	Decoding algorithm	No. of iterations
Convolutional Code (CC)				
CC(2,1,5)	23,33	16	VA	–
CC(2,1,7)	171,133	64	VA	–
CC(2,1,9)	561,753	256	VA	–
Turbo Convolutional Code (TC)				
TC(2,1,3)	7,5	4	Log-Map	8
TC(2,1,4)	13,15	8	Log-Map	8
TC(2,1,5)	23,35	16	Log-Map	8
Turbo BCH Code (TBCH)				
TBCH(31,26)	45	32	Log-Map	8
TBCH(32,26)	45	64	Log-Map	8
TBCH(31,21)	3551	1024	Log-Map	8
TBCH(63,57)	103	64	Log-Map	8
TBCH(127,120)	211	128	Log-Map	8
Trellis Coded Modulation (TCM)				
8PSK-TCM	103,30,66	64	VA	–
16QAM-TCM	101,16,64	64	VA	–
Turbo Trellis Coded Modulation (TTCM)				
8PSK-TTCM	11,2,4	8	Log-Map	8
16QAM-TTCM	23,2,4,10	16	Log-Map	8

Table 5.3: Parameters of the different channel encoders used in Figure 5.4.

channel coded rate is given by

$$R = \frac{k}{n} . \tag{5.42}$$

However, different code rates can be obtained by suitable puncturing [174] and we will elaborate on this issue later in the section. The first entry of Table 5.3 is the convolutional code CC$(2,1,5)$, which was adopted by the Groupe Speciale Mobile (GSM) committee in 1982 [49, 175]. Then in 1996, a more powerful convolutional code, the CC$(2,1,7)$ arrangement was employed by the Digital Video Broadcasting (DVB) [39] standard for television, sound and data services. Recently, the Universal Mobile Telecommunication System (UMTS) proposed the use of the CC$(2,1,9)$ scheme, which is also shown in Table 5.3. The implementation of this scheme is about 16 times more complex than that of the CC$(2,1,5)$ scheme adopted by GSM some 15 years ago. This clearly shows that the advances of integrated circuit technology have substantially contributed towards the performance improvement of mobile communication systems.

As mentioned earlier, a turbo encoder consists of two component encoders. Generally, two identical Recursive Systematic Convolutional (RSC) codes are used. Berrou *et al.* [12,13] used two constraint length $K = 3$, RSC codes, each having four trellis states. We denote a turbo convolutional code as TC(n, k, K) where n, k and K have their usual interpretations, as in CC. In [12,13], the MAP algorithm [11] was employed for iterative decoding. However, in our systems the Log-MAP decoding algorithm [52] was utilised. The Log-MAP algorithm is a more attractive version of the MAP algorithm, since it operates in the logarithmic domain, in order to reduce the computational complexity and to mitigate the numerical problems associated with the MAP algorithm [52]. The number of turbo iterations was set to eight, since this yielded a performance close to the optimum performance associated with an infinite number of iterations. In our investigations we will consider the turbo convolutional code TC$(2, 1, 3)$, as proposed in [12,13]. However, the more complex TC$(2, 1, 4)$ code [176] was proposed by UMTS to be employed in the third generation (3G) mobile communication systems [49,154,177]. The TC$(2, 1, 5)$ code is also interesting, since it is expected to provide further significant coding gains over that of the TC$(2, 1, 3)$ and TC$(2, 1, 4)$ code.

BCH codes [49] are used as the component codes in the TBCH codes of Table 5.3. Again, TBCH codes have been shown for example by Hagenauer [61,63] to perform impressively at near-unity coding rates, although at high complexity. Hence in our study the BCH component codes BCH$(31, 26)$, BCH$(31, 21)$, BCH$(63, 57)$ and BCH$(127, 120)$ are employed, as shown in Table 5.3. Finally, we also investigate TCM and TTCM. Both of them are employed in 8PSK and 16QAM modulation modes. This results in 8PSK-TCM, 16QAM-TCM and 8PSK-TTCM, 16QAM-TTCM, respectively.

In Table 5.3, we have given the encoding and decoding parameters of the different channel encoders employed. However, as mentioned earlier, we can design codes of variable code rates R by employing suitable puncturing patterns. By combining puncturing with different modulation modes, we could design a system having a range of various throughputs, expressed in terms of the number of Bits Per Symbol (BPS), as shown in Table 5.4 and 5.5. Some of the parameters in Table 5.4 and 5.5 are discussed in depth during our further discourse, but significantly more information can be gleaned concerning these systems by carefully studying both tables.

In Table 5.4 we summarised the simulation parameters of the CC and TCM schemes employed. Since there are two coded bits ($n = 2$) for each data bit ($k = 1$), we have two possible puncturing patterns, as shown in the table. A binary 1 means that the coded bit is transmitted, whereas a binary 0 implies that the coded bit is punctured. Accordingly, the puncturing pattern $(1, 1)$ simply implies that no puncturing is applied and hence results in a half-rate CC. However, for example in the DVB standard [39] different puncturing patterns were proposed for the CC$(2, 1, 7)$ code, which result in different coding rates. These are

Code	Code Rate R	Puncturing Pattern	Modulation Mode	BPS	Random interleaver depth
Convolutional Code (CC)					
CC(2,1,5)	0.50	1,1	QPSK	1.00	20,000
CC(2,1,7)	0.50	1,1	QPSK	1.00	20,000
	0.75	101, 110	64QAM	4.50	13,320
	0.83	10101, 11010	64QAM	5.00	12,000
CC(2,1,9)	0.50	1,1	QPSK	1.00	20,000
			16QAM	2.00	20,000
			64QAM	3.00	20,004
Trellis Coded Modulation (TCM)					
8PSK-TCM	0.67	1,1	8PSK	2.00	–
16QAM-TCM	0.75	1,1	16QAM	3.00	–

Table 5.4: Simulation parameters associated with the CC and TCM channel encoders in Figure 5.4.

also shown in Table 5.4.

In Table 5.5 we showed the simulation parameters of three different turbo schemes, namely that of the TC, TBCH and TTCM arrangements. Again, different code rates can be designed using suitable puncturing patterns, where the puncturing patterns seen in Table 5.5 consist of two parts. Specifically, the associated different puncturing patterns represent the puncturing patterns of the parity bits emanating from the first and the second encoder, respectively. These patterns are different from the puncturing patterns seen in Table 5.4. For the TC$(2, 1, 3)$ scheme different puncturing patterns are employed for the various code rates R. The puncturing patterns were optimised experimentally by simulations, in order to attain the best possible BER performance. The design procedure for punctured turbo codes was proposed by Acikel *et al.* [64] in the context of BPSK and QPSK.

5.4.3 Complexity Issues and Memory Requirements

In this section we address the complexity issues and memory requirements of the proposed system. We will mainly focus on the relative estimated complexity and memory requirements of the proposed channel decoders rather than attempting to determine their exact complexity. Therefore, in order to simplify our comparative study, several assumptions are made. In our simplified approach the estimated complexity of the whole system is deemed to depend only on that of the channel decoders. In other words, the complexity associated with the modulator, demodulator, space-time encoder and decoder as well as channel

Code	Code Rate R	Puncturing Pattern	Modulation Mode	BPS	Random turbo interleaver depth	Random (separation) interleaver depth
Turbo Convolutional Code (TC)						
TC(2,1,3)	0.50	$10, 01$	16QAM	2.00	10,000	20,000
TC(2,1,4)	0.33	$1, 1$	64QAM	2.00	10,000	30,000
	0.50	$10, 01$	16QAM	2.00	10,000	20,000
			64QAM	3.00	10,002	20,004
	0.67	$1000, 0001$	64QAM	4.00	10,000	15,000
	0.75	100000,	16QAM	3.00	9,990	13,320
		000001	64QAM	4.50	9,990	13,320
	0.83	1000000000, 0000000001	64QAM	5.00	10,000	12,000
	0.90	100000000000000000, 000000000000000001	64QAM	5.40	10,044	11,160
TC(2,1,5)	0.50	$10, 01$	16QAM	2.00	10,000	20,000
Turbo BCH Code (TBCH)						
TBCH(31,26)	0.72	1,1	16QAM	2.89	9,984	13,824
			64QAM	4.33	9,984	13,824
TBCH(32,26)	0.68	1,1	8PSK	2.05	9,984	14,592
TBCH(31,21)	0.51	1,1	16QAM	2.04	9,996	19,516
TBCH(63,57)	0.83	1,1	64QAM	4.96	10,032	12,144
TBCH(127,120)	0.90	1,1	64QAM	5.37	10,080	11,256
Turbo Trellis Coded Modulation (TTCM)						
8PSK-TTCM	2/3	$10, 01$	8PSK	2.00	10,000	–
16QAM-TTCM	3/4	$10, 01$	16QAM	3.00	13,332	–

Table 5.5: Simulation parameters associated with the TC, TBCH and TTCM channel encoders in Figure 5.4.

encoders are assumed to be insignificant compared to the complexity of channel decoders.

Since the estimated complexity of the channel decoders depends directly on the number of trellis transitions, the number of trellis transitions per information data bit will be used as the basis of our comparison. Several channel encoders schemes in Table 5.3 are composed of convolutional codes. For the binary convolutional code $\mathrm{CC}(2,1,K)$, two trellis transitions diverge from each of the 2^{K-1} states. Hence, we can approximate the complexity of a $\mathrm{CC}(2,1,K)$ code as:

$$\begin{aligned} comp\{\mathrm{CC(2,1,K)}\} &= 2 \times 2^{K-1} \\ &= 2^{K} . \end{aligned} \tag{5.43}$$

The number of trellis transitions in the Log-MAP decoding algorithm is assumed to be three times higher, than that of the conventional Viterbi algorithm, since the Log-MAP algorithm has to perform forward as well as backward recursion and soft output calculations, which results in traversing through the trellis three times. The reader is referred to Section 3.3.3.4 for further details of the algorithm. For TC codes we apply the Log-MAP decoding algorithm for iterative decoding, assisted by the two component decoders. Upon taking into account the number of turbo decoding iterations as well, the complexity of TC decoding is then approximated by:

$$\begin{aligned} comp\{TC(2,1,K)\} &= 3 \times 2 \times 2^{K-1} \times 2 \times \text{No. of Iterations} \\ &= 3 \times 2^{K+1} \times \text{No. of Iterations}\ . \end{aligned} \qquad (5.44)$$

In TCM we construct a non-binary decoding trellis [48]. The TCM schemes of Table 5.3 have 2^{BPS-1} trellis branches diverging from each trellis state, where BPS is the number of transmitted bits per modulation symbol. However, for each trellis transition we would have $BPS - 1$ transmitted information data bits, since the TCM encoder typically adds one parity bit per non-binary symbol. Therefore, we can estimate the complexity of the proposed TCM schemes as:

$$comp\{\text{TCM}\} = 2^{BPS-1} \times \frac{\text{No. of States}}{BPS-1}\ . \qquad (5.45)$$

Similarly to TC, TTCM consists of two TCM codes and the Log-MAP decoding algorithm [57] is employed for iterative decoding. The associated TTCM complexity is then estimated as:

$$\begin{aligned} comp\{\text{TTCM}\} &= 3 \times 2^{BPS-1} \times \frac{\text{No. of States}}{BPS-1} \times 2 \times \text{No. of Iterations} \\ &= \frac{3 \times 2^{BPS} \times \text{No. of States} \times \text{No of Iterations}}{BPS-1}\ . \end{aligned} \qquad (5.46)$$

For TBCH(n,k) codes the estimated complexity calculation is not as straightforward as in the previous cases. Its component codes are BCH(n,k) codes and the decoding trellis can be divided into three sections [18]. Assuming that $k > n-k$, for every decoding instant j the number of trellis states is given as [18]:

$$\text{No. of States}_j = \begin{cases} 2^j & j = 0,1,...,n-k-1 \\ 2^{n-k} & j = n-k, n-k+1,...,k \\ 2^{n-j} & j = k+1, k+2,...,n \end{cases}\ . \qquad (5.47)$$

It can be readily shown that:

$$2^{n-k} - 1 = \sum_{j=0}^{n-k-1} 2^j \qquad (5.48)$$

$$= \sum_{j=k+1}^{n} 2^{n-j}\ . \qquad (5.49)$$

Upon using the approximation $\sum_{j=0}^{n-k-1} 2^j = \sum_{j=k+1}^{n} 2^{n-j} = 2^{n-k} - 1 \approx 2^{n-k}$, we can write the number of decoding trellis states per information data bit as:

$$\begin{aligned} \text{No. of States} &= \frac{2 \times 2^{n-k} + \{k - (n-k)\} \times 2^{n-k}}{k} \\ &= \frac{(2k - n + 2) \times 2^{n-k}}{k} \ . \end{aligned} \qquad (5.50)$$

Having derived the number of decoding trellis states per information data bit, we can approximate the complexity of TBCH codes as:

$$\begin{aligned} comp\,\{\text{TBCH}(n,k)\} &= 3 \times 2 \times \frac{(2k-n+2) \times 2^{n-k}}{k} \times 2 \times \text{No. of Iterations} \\ &= \frac{3 \times (2k-n+2) \times 2^{n-k+2} \times \text{No. of Iterations}}{k} \ . \end{aligned} \qquad (5.51)$$

Having approximated the complexity of each channel decoder, we will now derive their approximate memory requirements. Typically, the memory requirement of a channel decoder depends directly on the number of trellis states in the entire coded block. Therefore in this section the number of trellis states per coded block serves as the basis of a relative memory requirement comparison between the channel decoders studied. For a binary convolutional code, observation of the VA has shown that typically all surviving paths of the current trellis state emerge from trellis states not 'older' than approximately five times the constraint length, K. Therefore at any decoding instant, only a section of $5 \times K$ trellis transitions has to be stored. We can then approximate the associated memory requirement as:

$$mem\,\{CC(2,1,K)\} = 2^{K-1} \times 5 \times K \ . \qquad (5.52)$$

Again, as highlighted in Section 3.3.3.4, the Log-MAP algorithm requires the storage of γ, α and β values. Hence for the same number of decoding trellis states, the Log-MAP algorithm would require about three times more memory, than the classic Viterbi algorithm. Consequently, we can estimate the memory requirement of the TC code as:

$$mem\,\{\text{TC}(2,1,K)\} = 3 \times 2^{K-1} \times \text{Block Length} \ . \qquad (5.53)$$

Similarly to CCs, we can approximate the memory requirements of TCM as:

$$mem\,\{\text{TCM}\} = \text{No. of States} \times \text{Block Length} \ . \qquad (5.54)$$

Following similar arguments, the memory requirements of TTCM employing the Log-MAP algorithm can be approximated as:

$$mem\,\{\text{TTCM}\} = 3 \times \text{No. of States} \times \text{Block Length} \ . \qquad (5.55)$$

The estimation of the memory requirements of TBCH codes is again different from that of the other channel codes considered. Specifically, their memory requirement does not directly depend on the number of decoding trellis states in a coded TBCH block. Instead, it depends on the number of decoding trellis states in the constituent BCH codewords. From Equation 5.50, we can estimate the associated memory requirements as:

$$mem\{\text{TBCH}(n,k)\} = 3 \times (2k - n + 2) \times 2^{n-k} \ . \tag{5.56}$$

Applying Equations 5.43 to 5.56, we summarised the estimated complexity and memory requirements of the channel decoders characterised in Table 5.3. Explicitly, assuming that there are 10,000 information data bits per coded block, the associated estimated complexity and memory requirements are then given in Table 5.6. Note that the block length of TCM and TTCM is expressed in terms of the number of symbols per coded block, since these schemes are symbol-oriented rather than bit-oriented.

Code	No. of states	No. of states per data bit	Iteration No	Block length	Complexity	Memory requirement
Convolutional Code (CC)						
CC(2,1,5)	16	16	–	10,000	32	400
CC(2,1,7)	64	64	–	10,000	128	2,240
CC(2,1,9)	256	256	–	10,000	512	11,520
Turbo Convolutional Code (TC)						
TC(2,1,3)	4	4	8	10,000	384	120,000
TC(2,1,4)	8	8	8	10,000	768	240,000
TC(2,1,5)	16	16	8	10,000	1,536	480,000
Turbo BCH Code (TBCH)						
TBCH(31,26)	32	28	8	31	2,718	2,208
TBCH(32,26)	64	54	8	32	5,199	4,224
TBCH(31,21)	1,024	634	8	31	60,855	39,936
TBCH(63,57)	64	60	8	63	5,713	10,176
TBCH(127,120)	128	123	8	127	11,776	44,160
Trellis Coded Modulation (TCM)						
8PSK-TCM	64	32	–	5,000	128	320,000
16QAM-TCM	64	21	–	3,333	171	213,312
Turbo Trellis Coded Modulation (TTCM)						
8PSK-TTCM	8	4	8	5,000	768	120,000
16QAM-TTCM	16	5	8	3,333	2,048	159,984

Table 5.6: Complexity and memory requirements of the different channel decoders in characterised Table 5.3.

5.5 Performance Results

In this section, unless otherwise stated, all simulation results are obtained over uncorrelated or – synonymously – perfectly interleaved narrow-band or non-dispersive Rayleigh fading channels. As stated before, this does not contradict the requirement for a constant channel magnitude and phase over n consecutive time slots in Equation 5.8, since upon applying a sufficiently high interleaving depth the channel's fading envelope can be indeed uncorrelated. Our assumptions were that:

1. The fading amplitudes were constant across n consecutive transmission slots of the space-time block codes' transmission matrix;
2. The average signal power received from each transmitter antenna was the same;
3. The receiver had a perfect knowledge of the channels' fading amplitudes.

We note that the above assumptions are unrealistic, yielding the best-case performance, nonetheless, facilitating the performance comparison of the various techniques under identical circumstances.

In the following sections, we compare the performance of various combinations of space-time block codes and channel codes. As mentioned earlier, various code rates can be used for both the space-time block codes and for the associated channel codes. The different modulation schemes employed result in various effective throughput. Hence, for a fair comparison, all different systems are compared on the basis of the same effective BPS throughput given by:

$$\text{BPS} = R_{st} \times R_{cc} \times \text{modulation throughput}, \tag{5.57}$$

where R_{st} and R_{cc} are the code rates of the space-time block code and the channel code, respectively.

5.5.1 Performance Comparison Of Various Space-Time Block Codes Without Channel Codecs

In this section, the performance of various space-time block codes without channel codes is investigated and compared. All the investigated space-time block codes, namely the $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ codes [71–73] have their corresponding transmission matrices given in Equation 5.10, 5.31, 5.32, 5.33 and 5.34, respectively. The encoding parameters are summarised in Table 5.2.

5.5.1.1 Maximum Ratio Combining and the Space-Time Code G_2

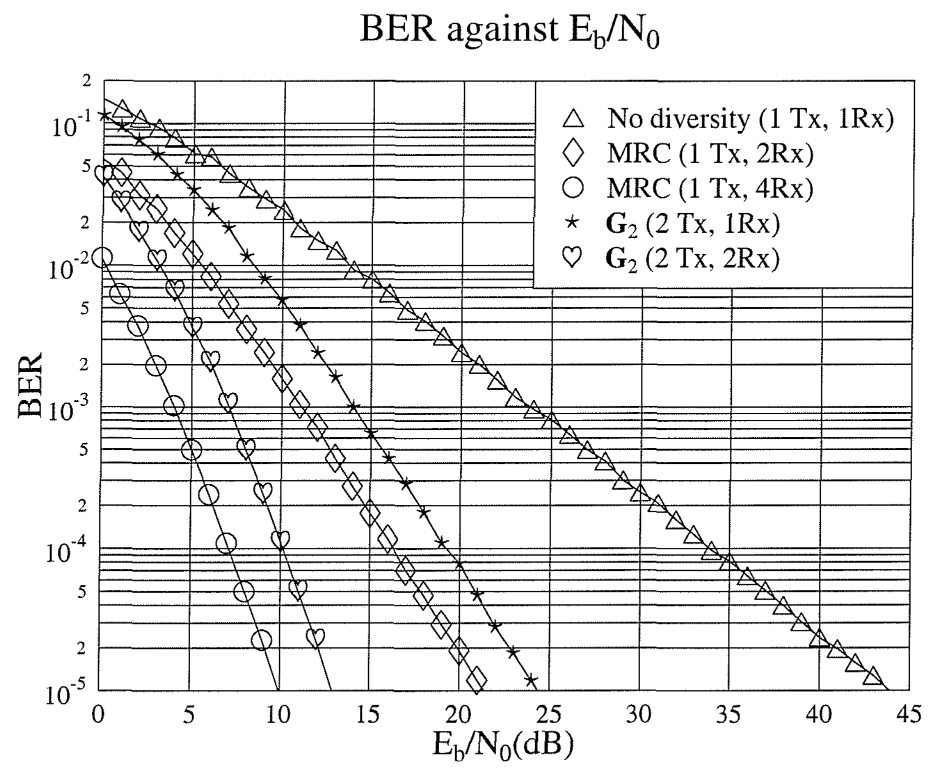


Figure 5.5: Performance comparison of the MRC technique and space-time code G_2 using **BPSK** over uncorrelated Rayleigh fading channels.

Figure 5.5 shows the performance of MRC and the space-time code G_2 using BPSK over uncorrelated Rayleigh fading channels. It is assumed that the total power received from both transmit antennas in the space-time coded system using G_2 of Equation 5.10 is the same as the transmit power of the single transmit antenna assisted MRC system. It can be seen in Figure 5.5 that the performance of the space-time code G_2 is about 3 dB worse, than that of the MRC technique using two receivers, even though both systems have the same diversity order of two. The 3 dB penalty is incurred, because the transmit power of each antenna in the G_2 space-time coded arrangement is only half of the transmit power in the MRC assisted system. It is shown in Figure 5.5 however that at a BER of 10^{-5} a diversity gain of 20 dB is achieved by the space-time code G_2. If we increase the diversity order to four by using two receivers, the space-time code G_2 achieves a diversity gain of 32 dB. However, there is still a 3 dB performance penalty as compared to the conventional MRC technique using four receivers. The advantage of the space-time coded scheme is nonetheless that the increased complexity of the space-time coded transmitter is more affordable at the BS than at the MS, where the MRC receiver would have to be located.

5.5.1.2 Performance of 1 BPS Schemes

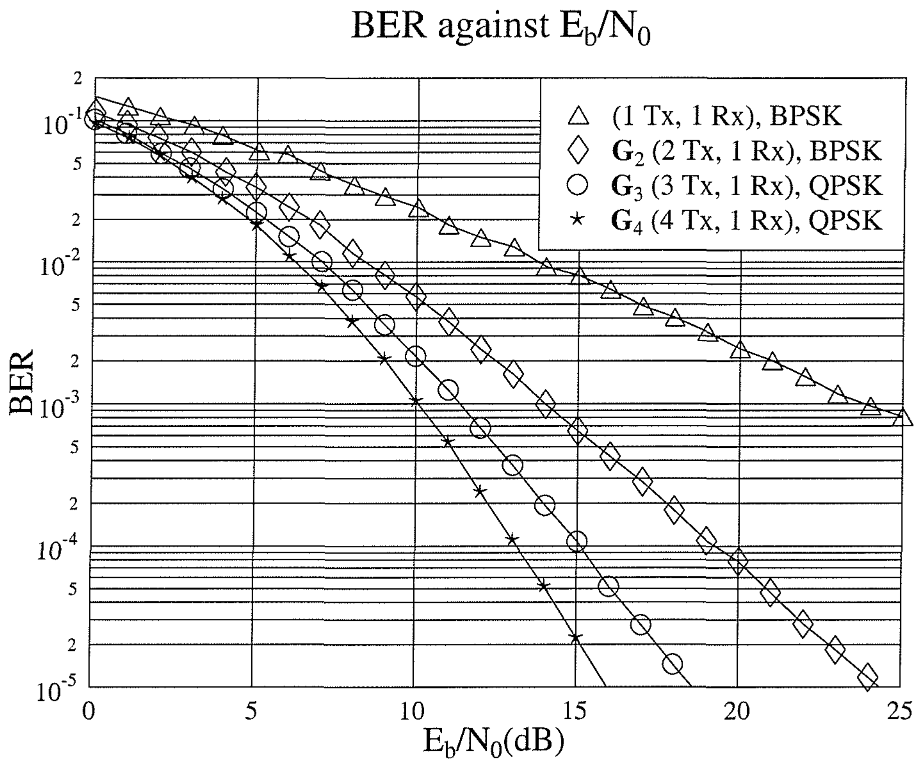


Figure 5.6: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ of Table 5.2 at an effective throughput of **1 BPS** using **one receiver** over uncorrelated Rayleigh fading channels.

Figures 5.6 and 5.7 compare the performance of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ having an effective throughput of 1 BPS over uncorrelated Rayleigh fading channels using one and two receivers, respectively. BPSK modulation was employed in conjunction with the space-time code $\mathbf{G}_2$. As shown in Table 5.2, the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ are half-rate codes. Therefore, QPSK modulation was used in the context of $\mathbf{G}_3$ and $\mathbf{G}_4$ in order to retain a throughput of 1 BPS. It can be seen in Figure 5.6 that at a BER of 10^{-5} the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ give about 2.5 and 7.5 dB gain over the $\mathbf{G}_2$ code, respectively. If the number of receivers is increased to two, as shown in Figure 5.7, the associated E_b/N_0 gain reduces to about 1 and 3.5 dB, respectively. The reason is that over the perfectly interleaved flat-fading channel encountered much of the attainable diversity gain is already achieved using the $\mathbf{G}_2$ code and two receivers. The associated gains of the various schemes at a BER of 10^{-5} are summarised in Table 5.7.

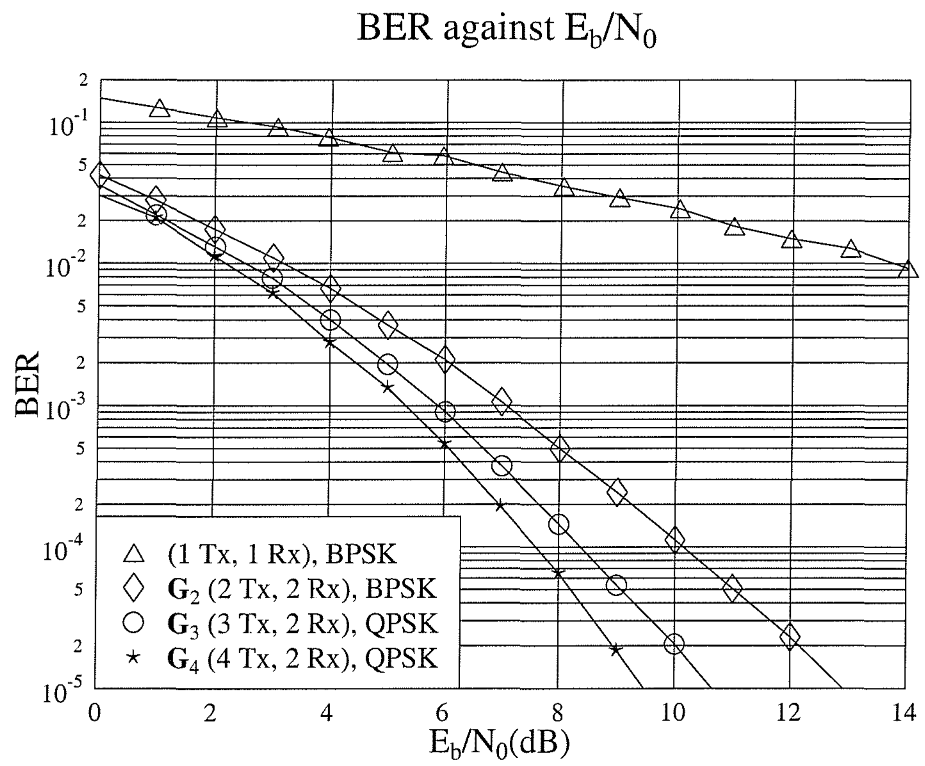


Figure 5.7: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ of Table 5.2 at an effective throughput of **1 BPS** using **two receivers** over uncorrelated Rayleigh fading channels.

5.5.1.3 Performance of 2 BPS Schemes

In Figure 5.8 we compare the performance of the space-time codes $\mathbf{G}_2, \mathbf{G}_3, \mathbf{G}_4, \mathbf{H}_3$ and $\mathbf{H}_4$ proposed in [71–73] using the encoding parameters summarised in Table 5.2. The performance results were obtained over uncorrelated Rayleigh fading channels using one receiver and the effective throughput of the system is about 2 BPS. For the $\mathbf{G}_2$ code QPSK modulation was used, while the $\mathbf{G}_3$ and $\mathbf{G}_4$ codes employ 16QAM conveying 4 BPS. Hence the effective throughput is 2 BPS, since $\mathbf{G}_3$ and $\mathbf{G}_4$ are half-rate codes. Since the code rate of the $\mathbf{H}_3$ and $\mathbf{H}_4$ codes is $\frac{3}{4}$, 8PSK modulation was employed in this context, resulting in a throughput of $3 \times 3/4 = 2.25$ BPS, which is approximately 2 BPS. We can see in Figure 5.8 that at high BERs or low E_b/N_0 values the $\mathbf{G}_2$ code slightly outperforms the others. However, the situation is reversed, when the system is operated at a low BER or high E_b/N_0 values. At a BER of 10^{-5} the code $\mathbf{G}_4$ only gives a diversity gain of 5 dB over the $\mathbf{G}_2$ code. This is a 2.5 dB loss compared to the 7.5 dB gain achieved by the system transmitting at an effective throughput of 1 BPS in the previous section. This is because the more vulnerable 16QAM scheme was used for the space-time code $\mathbf{G}_4$. Since

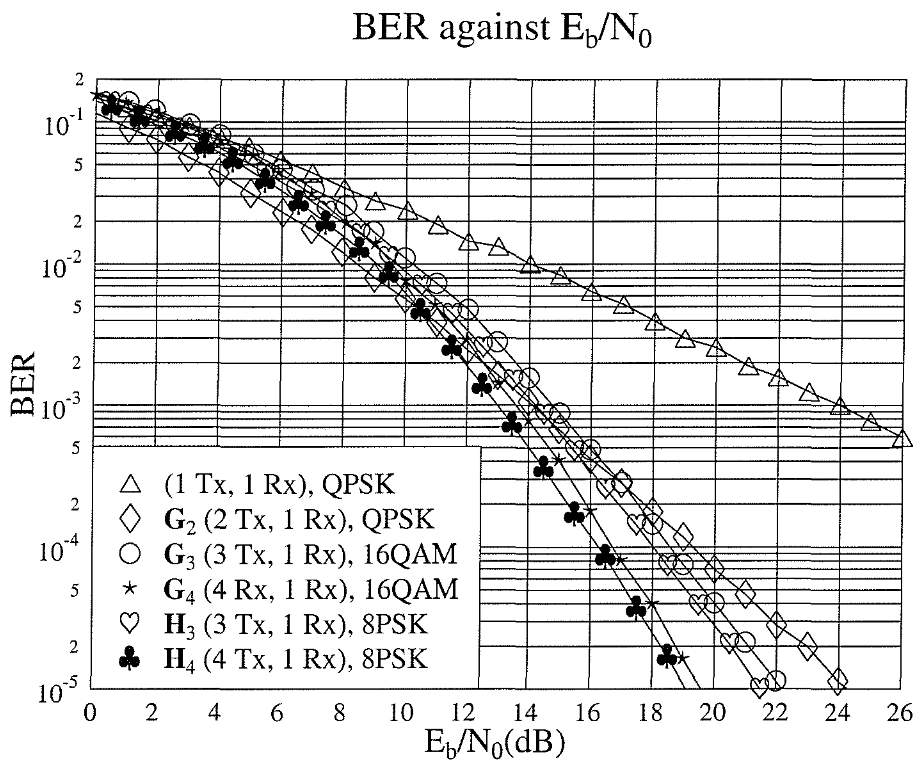


Figure 5.8: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ at an effective throughput of approximately **2 BPS** using **one receiver** over uncorrelated Rayleigh fading channels. The associated parameters of the space-time codes are summarised in Table 5.2.

the 16QAM signal constellation is more densely packed compared to QPSK, it is more prone to errors. Moreover, the space-time code $\mathbf{G}_4$ has no error correction capability to correct the extra errors induced by employing a more vulnerable, higher-order modulation scheme. Hence, this results in a poorer performance. If the throughput of the system is increased by employing an even higher-order modulation scheme, the space-time code $\mathbf{G}_4$ will suffer even higher performance degradations, as it will be shown in the next section. Since the space-time code $\mathbf{G}_3$ of Table 5.2 is also a half-rate code, similarly to the $\mathbf{G}_4$ code, it suffers from the same drawbacks.

In Figure 5.8, we also show the performance of the rate $\frac{3}{4}$ space-time codes $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 5.2. Both the $\mathbf{H}_4$ and $\mathbf{G}_4$ codes have the same diversity order of four in conjunction with one receiver. However, at a BER of 10^{-5} the performance of the $\mathbf{H}_4$ code is about 0.5 dB better, than that of the $\mathbf{G}_4$ code. This is again due to the higher-order modulation employed in conjunction with the half-rate code $\mathbf{G}_4$, in order to maintain the same throughput. As alluded to earlier, the higher-order modulation schemes are more susceptible to errors and hence the performance of the system in conjunction with the **G**$_3$ or **G**$_4$ code of Table 5.2 is worse, than that of the **H**$_3$ or **H**$_4$ code having the same diversity orders, respectively. The associated gains of the various schemes at a BER of 10^{-5} are summarised in Table 5.7.

5.5.1.4 Performance of 3 BPS Schemes

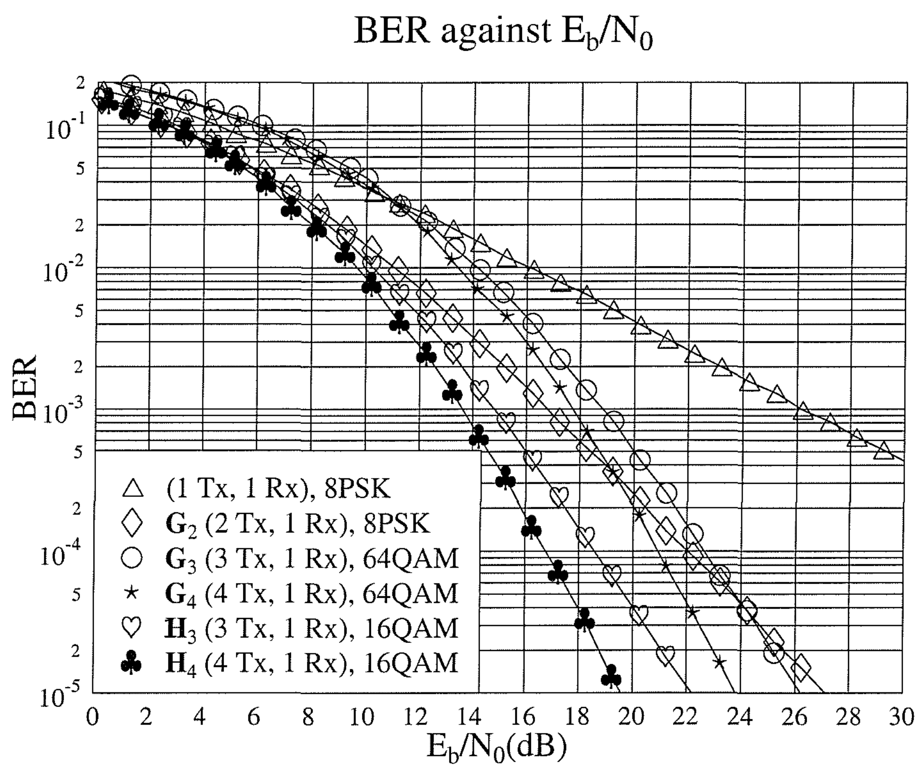


Figure 5.9: Performance comparison of the space-time codes **G**$_2$, **G**$_3$, **G**$_4$, **H**$_3$ and **H**$_4$ of Table 5.2 at an effective throughput of **3 BPS** using **one receiver** over uncorrelated Rayleigh fading channels.

Figures 5.9 and 5.10 show our performance comparisons for the space-time codes **G**$_2$, **G**$_3$, **G**$_4$, **H**$_3$ and **H**$_4$ of Table 5.2 at an effective throughput of 3 BPS over uncorrelated Rayleigh fading channels using one and two receivers, respectively. When using the **G**$_2$ code we employed 8PSK modulation. Since **G**$_3$ and **G**$_4$ are half-rate codes, 64QAM was employed, in order to obtain an effective throughput of 3 BPS. By contrast, for the **H**$_3$ and **H**$_4$ codes, which have a code rate of $\frac{3}{4}$, 16QAM was used in order to ensure the same throughput of $4/4 = 3$ BPS.

In Figure 5.9 we can see that at a BER of 10^{-5} the diversity gain of the **G**$_4$ code over the

$\mathbf{G}_2$ code is further reduced to about 3 dB. There is only a marginal diversity gain for the $\mathbf{G}_3$ code over the $\mathbf{G}_2$ code. As alluded to in the previous section, 64QAM in conjunction with the space-time code $\mathbf{G}_3$ or $\mathbf{G}_4$, has a densely packed signal constellation and hence this scheme is prone to errors. At the higher BER of 10^{-2} the $\mathbf{G}_2$ code outperforms the $\mathbf{G}_3$ and $\mathbf{G}_4$ codes by approximately 3 and 4 dB, respectively.

Due to the associated higher-order modulation scheme employed, we can see in Figure 5.9 that at a BER of 10^{-5} the $\mathbf{H}_3$ and $\mathbf{H}_4$ codes of Table 5.2 outperform both the $\mathbf{G}_3$ and the $\mathbf{G}_4$ codes. Specifically, we can see that the $\mathbf{H}_3$ code attains about 2 dB gain over the $\mathbf{G}_4$ code, even though it has a lower diversity order.

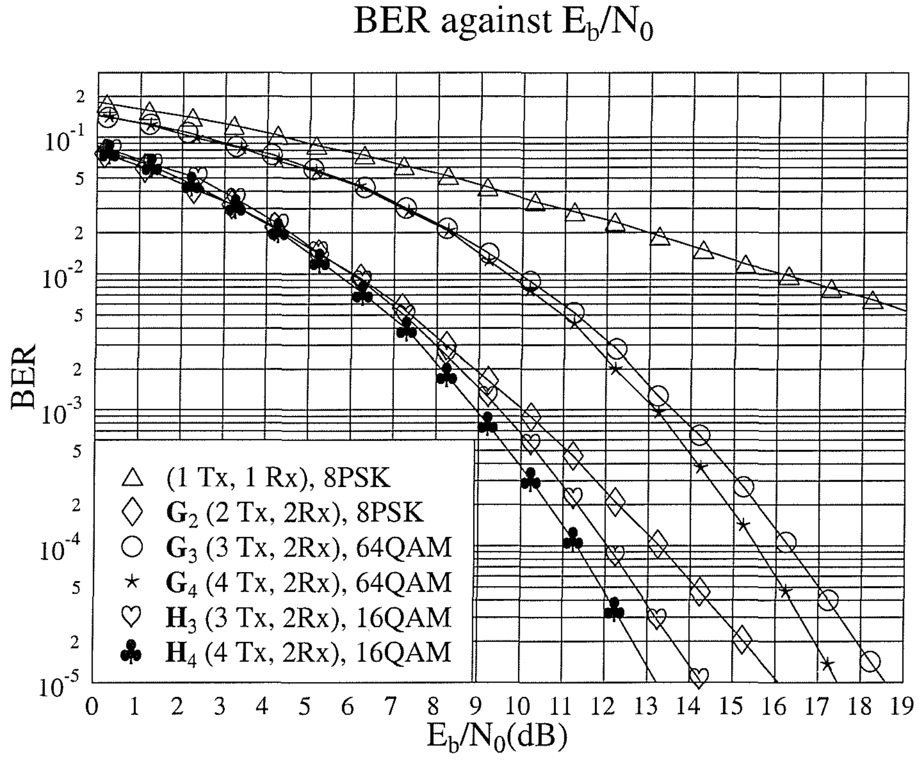


Figure 5.10: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 5.2 at an effective throughput of **3 BPS** using **two receivers** over uncorrelated Rayleigh fading channels.

If we increase the number of receivers to two, a scenario characterised in Figure 5.10, the performance degradation of the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ is even more pronounced. At a BER of 10^{-5} the performance gain of the $\mathbf{H}_4$ code over the $\mathbf{G}_4$ code is approximately 4 dB compared to the 0.5 dB gain, when the system's effective throughput is only 2 BPS, as it was shown in Figure 5.8 of the previous section.

Having studied Figures 5.6 to 5.10, we may conclude two important points. Firstly, the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ of Table 5.2 suffer from having a code-rate of half, since this significantly reduces the effective throughput of the system. In order to maintain the same throughput as the unity-rate $\mathbf{G}_2$ code, higher-order modulation schemes, such as for example 64QAM have to employed. This results in a preponderance of channel errors, since the constellation points of the higher-order modulation schemes are more densely packed. Due to their lack of error correcting capability, the $\mathbf{G}_3$ and $\mathbf{G}_4$ codes suffer performance losses compared to the $\mathbf{G}_2$ code. Secondly, if the number of receivers is increased to two, the performance gain of the $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ or $\mathbf{H}_4$ codes over the $\mathbf{G}_2$ code becomes lower. The reason behind this phenomenon is that much of the attainable diversity gain was already achieved using the $\mathbf{G}_2$ code and two receivers. The associated gains of the various schemes at a BER of 10^{-5} are summarised in Table 5.7.

		Coding gain (dB)					
		One receiver			Two receivers		
Code	Rate	1 BPS	2 BPS	3 BPS	1 BPS	2 BPS	3 BPS
$\mathbf{G}_2$	1	19.5	19.6	19.1	30.9	30.9	30.1
$\mathbf{G}_3$	1/2	25.2	21.8	20.0	33.2	29.6	27.6
$\mathbf{G}_4$	1/2	27.9	24.3	22.4	34.3	30.7	28.8
$\mathbf{H}_3$	3/4	–	22.4	24.0	–	30.1	31.9
$\mathbf{H}_4$	3/4	–	24.8	22.6	–	31.2	33.0

Table 5.7: Coding gain (dB) of the space-time block codes of Table 5.2 over uncorrelated Rayleigh fading channels.

5.5.1.5 Channel Coded Space-Time Block Codes

In the previous sections, we have shown that without channel coding the performance of the unity-rate space-time $\mathbf{G}_2$ code is inferior to the lower rate space-time codes, namely to that of the $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ schemes. Since the space-time code $\mathbf{G}_2$ has a unity code rate, half-rate turbo codes can be employed for improving the performance of the system. In Figure 5.11, we compare the performance of the half-rate TC(2,1,4) code concatenated with the space-time code $\mathbf{G}_2$ and with the space-time block codes $\mathbf{G}_4$ and $\mathbf{H}_4$. Both the space-time codes $\mathbf{G}_4$ and $\mathbf{H}_4$ have a diversity gain of four and a code rate of $\frac{1}{2}$ and $\frac{3}{4}$, respectively. The associated parameters are shown in Tables 5.2, 5.3 and 5.5. Suitable modulation schemes were chosen so that all systems had the same throughput of 3 BPS. All simulation results were obtained over uncorrelated Rayleigh fading channels.

From Figure 5.11, we can see that a huge performance improvement is achieved by concatenating the space-time code $\mathbf{G}_2$ with the half-rate code TC(2,1,4). At a BER of 10^{-5}

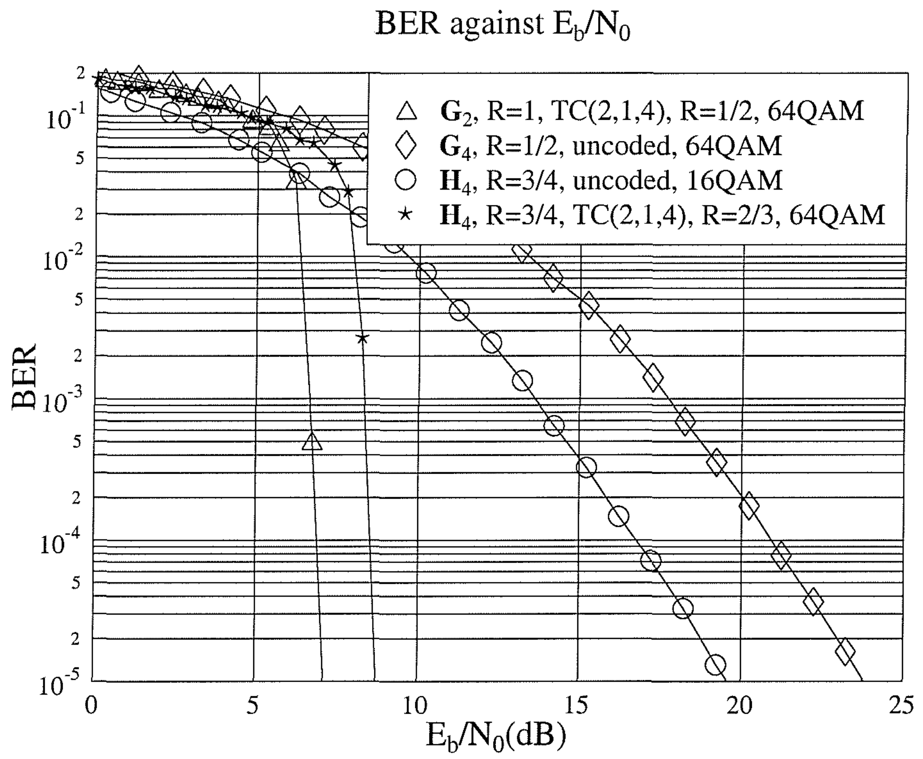


Figure 5.11: Performance comparison of the half-rate TC(2,1,4) code concatenated with the space-time code $\mathbf{G}_2$ and the space-time block codes $\mathbf{G}_4$ and $\mathbf{H}_4$. The associated parameters are shown in Tables 5.2, 5.3, and 5.5. All simulation results were obtained at an effective throughput of **3 BPS** over uncorrelated Rayleigh fading channels.

this concatenated scheme attains a coding gain of 16 dB and 13 dB compared to the space-time codes $\mathbf{G}_4$ and $\mathbf{H}_4$, respectively. This clearly shows that it is better to invest the parity bits associated with the code-rate reduction in the concatenated turbo code, rather than in non-unity-rate space-time block codes. In Figure 5.11 we also show the performance of the space-time code $\mathbf{H}_4$ concatenated with the punctured two-third rate code TC(2,1,4). The figure shows that the TC(2,1,4) code improves the performance of the system tremendously, attaining a coding gain of 11 dB compared to the non-turbo-coded space-time code $\mathbf{H}_4$, at BER= 10^{-5}. However, its performance is still inferior to that of the half-rate TC(2,1,4) coded space-time code $\mathbf{G}_2$.

In conclusion, in Figure 5.11 we have seen that the reduction in coding rate is best assigned to turbo channel codes, rather to space-time codes. Therefore, in all our forthcoming simulations, all channel codecs of Table 5.3 are concatenated with the unity-rate space-time code $\mathbf{G}_2$, instead of the non-unity-rate space-time codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 5.2.

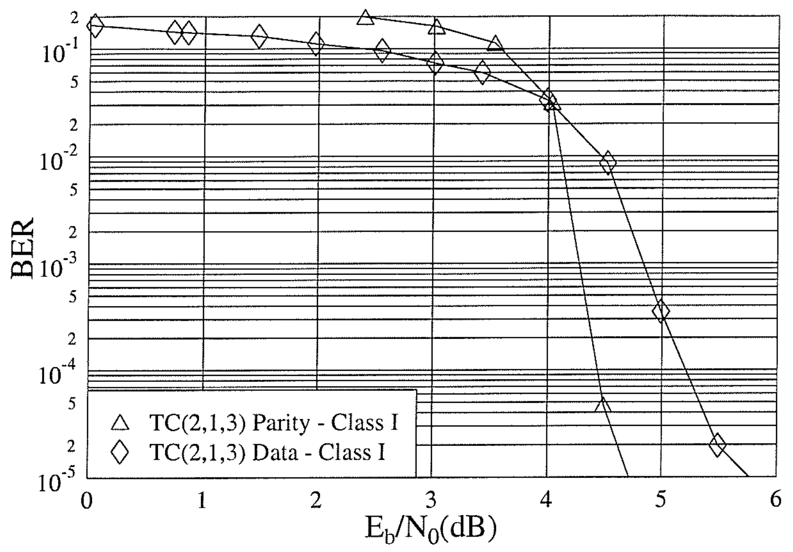
5.5.2 Mapping Binary Channel Codes to Multilevel Modulation

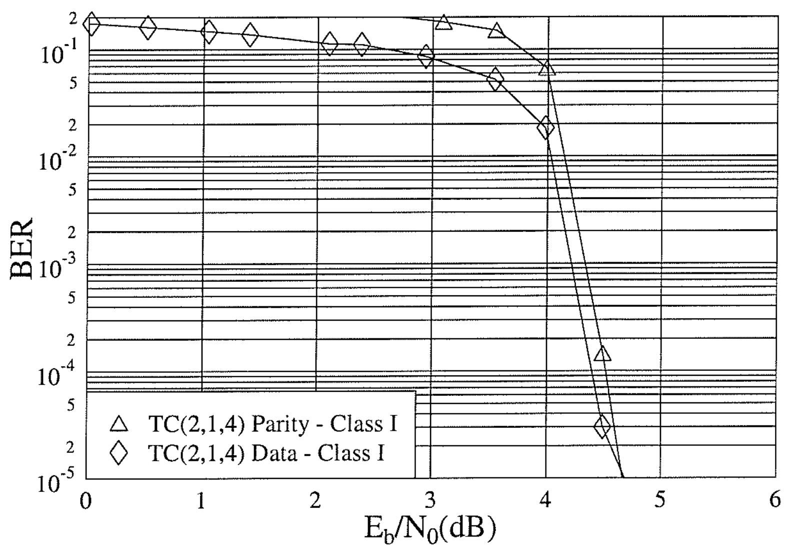
As mentioned earlier, in our investigations different modulation schemes are employed in conjunction with the binary channel codecs CC, TC and TBCH. Specifically, the modulation schemes used are BPSK, QPSK, 8PSK, 16QAM and 64QAM. Gray-mapping [86, 102, 153] is employed to map the bits to the QPSK, 8PSK, 16QAM and 64QAM symbols. In higher-order modulation schemes, such as 8PSK, 16QAM and 64QAM we have several transmitted bits per constellation point. However, the different bit positions of the constellation points have different noise-protection distances [153]. More explicitly, the protection distance is the Euclidean distance from one constellation point to another, which results in the corruption of a particular bit. A larger noise-protection distance results in a higher integrity of the bit and vice-versa. Therefore, for the different bit positions in the symbol we have different protection for the transmitted bits within the phaser constellation of the non-binary modulation schemes. It can be readily shown that in 8PSK and 16QAM we have two protection classes, namely class I and II [86, 153], where the class I transmitted bits are more protected. Similarly, in 64QAM we have three protection classes namely I, II and III [153], where the transmitted bits in class I are most protected, followed by class II and class III.

In our system the parity bits are generated by binary channel encoders, such as the CC, TC and TBCH schemes for protecting the binary data bits. However it is not intuitive, whether the integrity of the data or parity bits is more important in yielding a better overall BER performance. For example, if the parity bits are more important, it is better to allocate the parity bits to the better protection classes in higher-order modulation scheme and vice-versa. Therefore, in this section, we will investigate the performance of different channel codes along with different bit mapping schemes. The effect of the bit interleaver seen in Figure 5.4 is studied in conjunction with binary channel codes as well.

5.5.2.1 Turbo Convolutional Codes - Data and Parity Bit Mapping

We commence here by studying half-rate turbo convolutional codes, which are characterised in Table 5.3. An equal number of parity and data bits are generated by the half-rate TC codes and they are then mapped to the protection classes of the 16QAM scheme considered. Again, in the Gray-mapping assisted 16-QAM constellation there are two protection classes [153], class I and II, depending on the bit position. Explicitly, there are four bits per symbol in the 16-QAM constellation and two of the bit positions are more protected, than the remaining two bits.

(a)


(b)


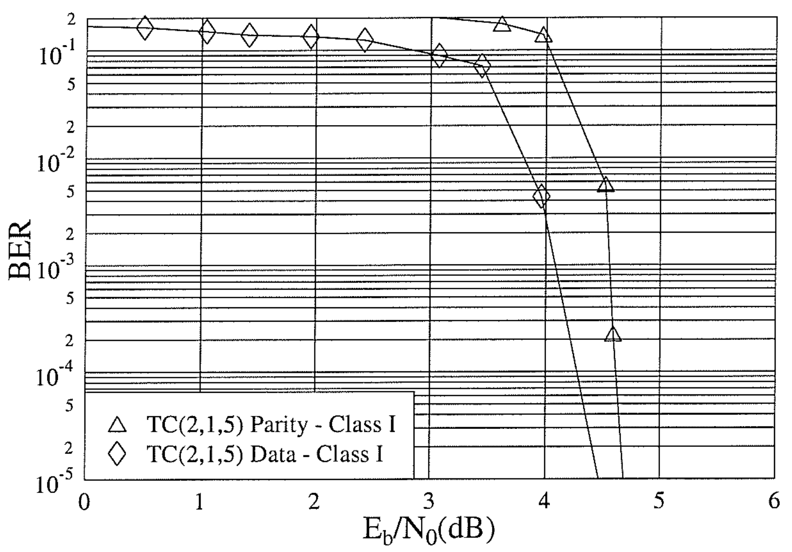
(c)


Figure 5.12: Performance comparison of various data and parity bit allocation schemes for the (a) TC(2,1,3), (b) TC(2,1,4) and (c) TC(2,1,5) codes, where the parameters are shown in Table 5.3. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver and 16QAM over uncorrelated Rayleigh fading channels at an effective throughput of 2 BPS.

In Figure 5.12 we compare the performance of various parity and data bit mapping schemes for the (a) TC(2,1,3), (b) TC(2,1,4) and (c) TC(2,1,5) codes. The curve marked by triangles represents the performance of the TC codes, when allocating the parity bits to the higher-integrity protection class I and the data bits to the lower-integrity protection class II. On the other hand, the performance curve marked by diamonds indicates the allocation of data bits to protection class I, while the parity bits are assigned to protection class II.

In Figure 5.12(a), we can see that at low E_b/N_0 values the performance of the TC(2,1,3) code, when allocating the parity bits to protection class I is worse, than upon allocating the data bits to protection class I. However, for E_b/N_0 values in excess of about 4 dB, the situation is reversed. At a BER of 10^{-5}, there is a performance gain of about 1 dB when using the TC(2,1,3) arrangement with the parity bits allocated to protection class I. We surmise that by protecting the parity bits better, we render the TC(2,1,3) code more powerful. It is common that stronger channel codes perform worse, than weaker codes at low E_b/N_0 values, but outperform their less powerful counterparts for higher E_b/N_0 values.

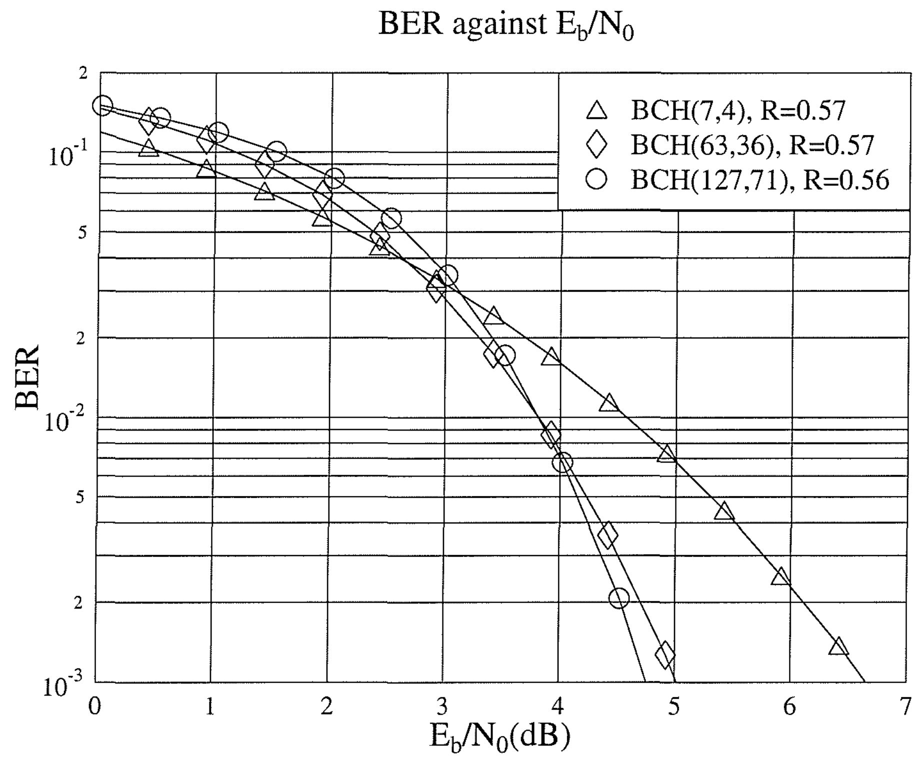


Figure 5.13: Performance comparison of hard decision algebraic decoding of different BCH codes having approximately the same code rate of $R = 0.57$, using BPSK over AWGN channels.

This is further justified in Figure 5.13. Here, we showed the performance of hard decision algebraic decoding of the BCH(7,4), BCH(63,36) and BCH(127,71) codes using BPSK over AWGN channels. All BCH codes characterised in the figure have approximately the same code rate, which is $R = 0.57$. From the figure we can see that at a BER of 10^{-3} the performance of the BCH codes improves with an increasing codeword length n. However, at a high BER or low E_b/N_0 value we can see that the performance of the BCH(7,4) code is better, than that of the BCH(63,36) and BCH(127,71) codes, which are stronger channel codes. This is, because stronger codes have many codewords having a large free distance. At low SNRs we have bad channel conditions and hence the channel might corrupt even those codewords having a large free distance. Once they are corrupted, they produce many erroneous information bits, a phenomenon which results in a poorer BER performance.

In Figure 5.12(b) we showed the performance of the TC(2,1,4) code using the same data and parity bit allocation, as in Figure 5.12(a). The figure clearly shows that the TC(2,1,4) scheme exhibits a better performance for E_b/N_0 values below about 4.7 dB, if the data bits are more strongly protected than the parity bits. It is also seen from the figure that the situation is reversed for E_b/N_0 values above this point. This phenomenon is different from the behaviour of the TC(2,1,3) scheme, since the crossing point of both curves occurs at a significantly lower BER. The same situation can be observed for the BCH codes characterised in Figure 5.13, where we can see that the performance curve of the BCH(127,71) code crosses the performance curve of the BCH(63,36) scheme at $E_b/N_0 \approx 4$ dB. This value is lower, than the crossing point of the performance curves of the BCH(63,36) and BCH(7,4) codes. Hence the trend is that the crossing point of stronger codes is shifted to right of the figure. Hence the crossing point of the performance curves of stronger codes will occur at lower BERs and shifted to the right on the E_b/N_0 scale. From the above argument we can speculate also in the context of TC codes that since the TC(2,1,4) scheme is a stronger code than the TC(2,1,3) arrangement, the crossing point of the associated performance curves for TC(2,1,4) is at a lower BER, than that of the TC(2,1,3) code and appears to be shifted to right of the E_b/N_0 scale.

Let us now consider the same performance curves in the context of the significantly stronger TC(2,1,5) code in Figure 5.12(c). The figure clearly shows that better performance is yielded in the observed range, when the data bits are more strongly protected. Unlike in Figure 5.12(a) and 5.12(b), there is no visible crossing point in Figure 5.12(c). However, judging from the gradient of both curves, if we were to extrapolate the curves in Figure 5.12(c), they might cross at BER$\approx 10^{-6}$. The issue of data and parity bit mapping to multilevel modulation schemes was also addressed by Goff *et al.* [55]. However, the authors only investigated the performance of the TC(2,1,5) code and stated that better performance is achieved by protecting more strongly the data bits. Additionally, we note here that the situation was reversed for the TC(2,1,3) code, where better performance was achieved by protecting the parity bits more strongly.

Hence, from the three subfigures of Figure 5.12 we can draw the following conclusions for the mapping of the data and parity bits to the different protection classes of the modulated symbol. For weaker half-rate turbo codes, such as the TC(2,1,3) arrangement, it is better to protect the parity bits more strongly. On the other hand, for stronger half-rate turbo codes, such as the TC(2,1,4) and TC(2,1,5) schemes, better performance is achieved by protecting more strongly the data bits. From our simulation results, we found that the same scenario also applies to turbo codes having code rates lower or higher than half-rates, as shown in Table 5.5. Based on these facts, we continue our investigations into the effect of interleavers, in an effort to achieve an improved performance.

5.5.2.2 Turbo Convolutional Codes – Interleaver Effects

In Figure 5.4 we have seen that a bit-based interleaver is employed for the CC, TC and TBCH codes. Since our performance results are obtained over uncorrelated Rayleigh fading channels, the purpose of the bit-based interleaver is to disperse bursts of channel errors within a modulated symbol, when it experiences a deep fade. This is vital for TC codes, because according to the turbo code structure proposed by Berrou *et al.* in [12, 13], at the output of the turbo encoder, a data bit is followed by the parity bits generated for its protection against errors. Therefore in multi-level modulation schemes a particular modulated symbol could consist of the data bit and its corresponding parity bits generated for its protection. If the symbol experiences a deep fade, the demodulator would provide low reliability values for both the data bit and the associated parity bits. In conjunction with low reliability information the turbo decoder may fail to correct errors induced by the channel. However, we can separate both the data bit and the parity bits generated for its protection into different modulation symbols. By doing so, there is a better chance that the demodulator can provide high-reliability parity bits, which are represented by another modulation symbol, even if the data bit experienced a deep fade and vice-versa. This will assist the turbo decoder in correcting errors.

More explicitly, the random interleaver shown in Figure 5.4 has two different effects on the binary channel codes, namely:

1. It separates the data bit and the parity bits generated for its protection into different modulated symbols;

2. It randomly maps the data and parity bits into different protection classes in multilevel modulation schemes.

The first effect of the random interleaver will improve the performance of the binary channel codecs. By contrast, the second effect might have a negative impact on the performance of the channel codecs, because the data and parity bits are randomly mapped to the different protection classes, rather than assigning the more vulnerable bits consistently to the higher-integrity protection class.

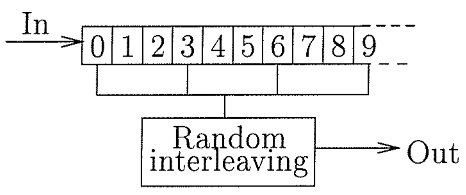


Figure 5.14: Random separation based interleaving.

In order to eliminate the potentially detrimental second effect of the random interleaver, we propose to invoke a so-called random separation based interleaver. Explicitly, Figure 5.14 shows an example of the random separation based interleaving employed. The objective of random separation based interleaving is to randomly interleave the bits within the same protection class of the multilevel modulated symbols. If 8PSK modulation is used, 3 bits per symbol are transmitted. Hence, for every 3-bit spaced position, the bits will be randomly interleaved. For example, in Figure 5.14 we randomly interleaved the bit positions $0, 3, 6, 9, \ldots$ Similarly, bit positions $1, 4, 7, \ldots$ and $2, 5, 8, \ldots$ will be randomly interleaved as well.

In Figure 5.15 we investigated the effects of both a random interleaver and those of a random separation based interleaver on the performance of the TC(2,1,3) code. The encoding parameters of the TC(2,1,3) code are shown in Table 5.3. The simulation results were obtained in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and 16QAM over uncorrelated Rayleigh fading channels. The performance curves marked by the triangles and diamonds were obtained by protecting the parity bits and data bits more strongly, respectively. Recall that the same performance curves were also shown in Figure 5.12(a).

As mentioned earlier, the random interleaver has two different effects on the performance of binary channel codes. It randomly maps the data and parity bits into different protection classes which might have a negative impact on the performance of the channel codecs. Additionally, it may separate the data bits and parity bits generated for their protection into different modulated symbols, which on the other hand might improve the performance.

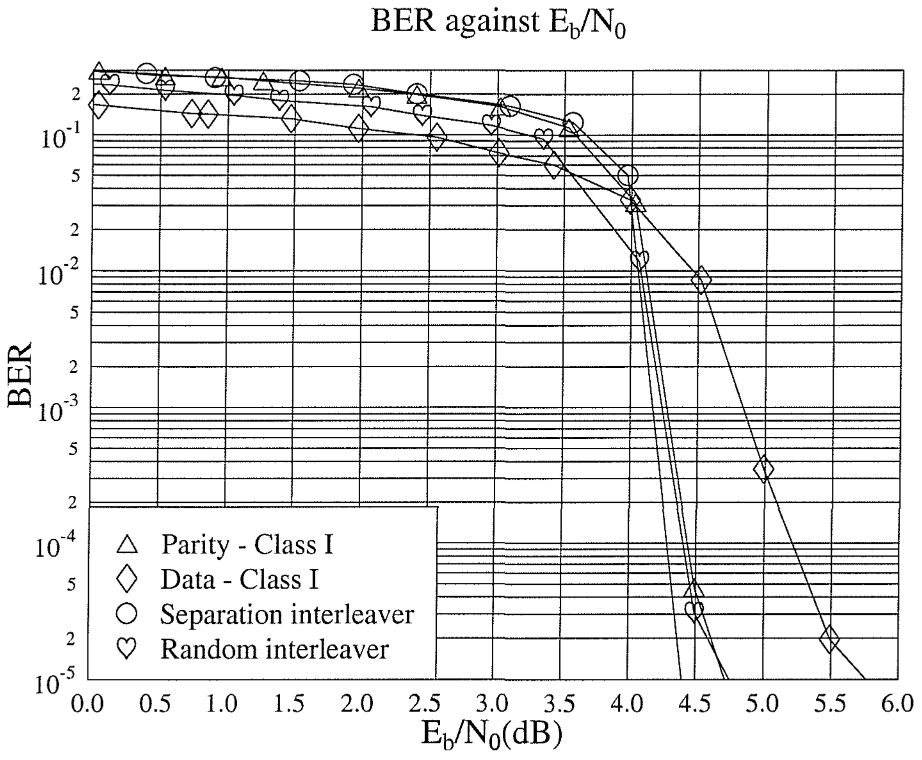


Figure 5.15: Performance comparison between different bit-to-symbol mapping methods for the **TC(2,1,3)** code in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and **16QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TC(2,1,3) code are shown in Table 5.3.

In Figure 5.15 the random interleaver based performance curve is marked by the hearts, which is similar to that of the TC(2,1,3) coded scheme protecting the parity bits more strongly. This suggest that the above-mentioned positive effect of the random interleaver is more pronounced than the negative effect in the context of the TC(2,1,3) coded scheme. On the other hand, based on the evidence of Figure 5.12(a) the random separation based interleaver was ultimately applied in conjunction with the allocation of the parity bits, rather than the data bits into protection class I. The interleaver randomly interleaved the coded bits within the same protection class of a block of transmitted symbols. Therefore, the parity bits remained more protected compared to the data bits and yet they have been randomly interleaved within the set of parity. In Figure 5.15 the performance of the random separation based interleaver is marked by circles, which is about 0.5 dB better, than that of the TC(2,1,3) coded scheme with the parity bits allocated to protection class I.

Similarly to Figure 5.15, in Figures 5.16 and 5.17 we show the performance of the

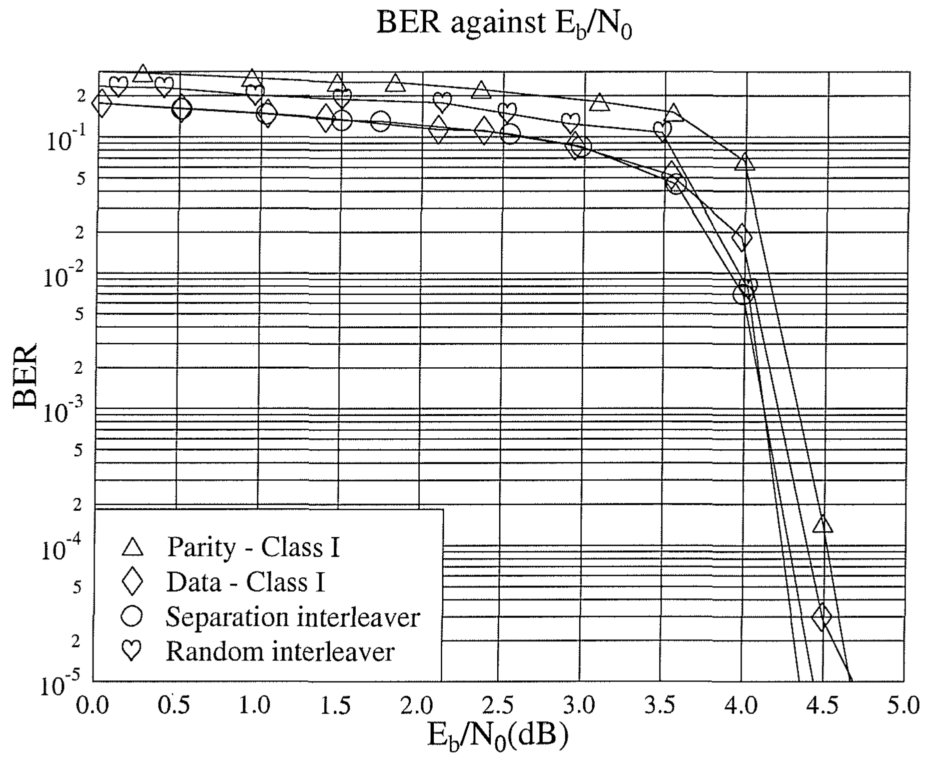


Figure 5.16: Performance comparison between different bit-to-symbol mapping methods for the **TC(2,1,4)** code in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and **16QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TC(2,1,4) code are shown in Table 5.3.

TC(2,1,4) and TC(2,1,5) codes, respectively, using different bit-to-symbol mapping methods. All simulation results were obtained in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and 16QAM over uncorrelated Rayleigh fading channels. The encoding parameters of the TC(2,1,4) and TC(2,1,5) codes are shown in Table 5.3. Unlike in Figure 5.15, the random separation based interleaver was applied in conjunction with the allocation of the data bits, rather than the parity bits to protection class I. It can be seen from Figures 5.16 and 5.17 that the performance of the random interleaver and random separation based interleaver is similar. This again suggest that the above-mentioned positive effect yielded by the random based interleaver is more pronounced than its detrimental effect in the context of both the TC(2,1,4) and TC(2,1,5) schemes.

In conclusion, our simulation results presented in this section demonstrated that at a BER of 10^{-5} the half-rate turbo codes using a random separation based interleaver attain the best performance, albeit for certain schemes only by a small margin. Therefore in our forthcoming performance comparisons we will be employing the random separation based

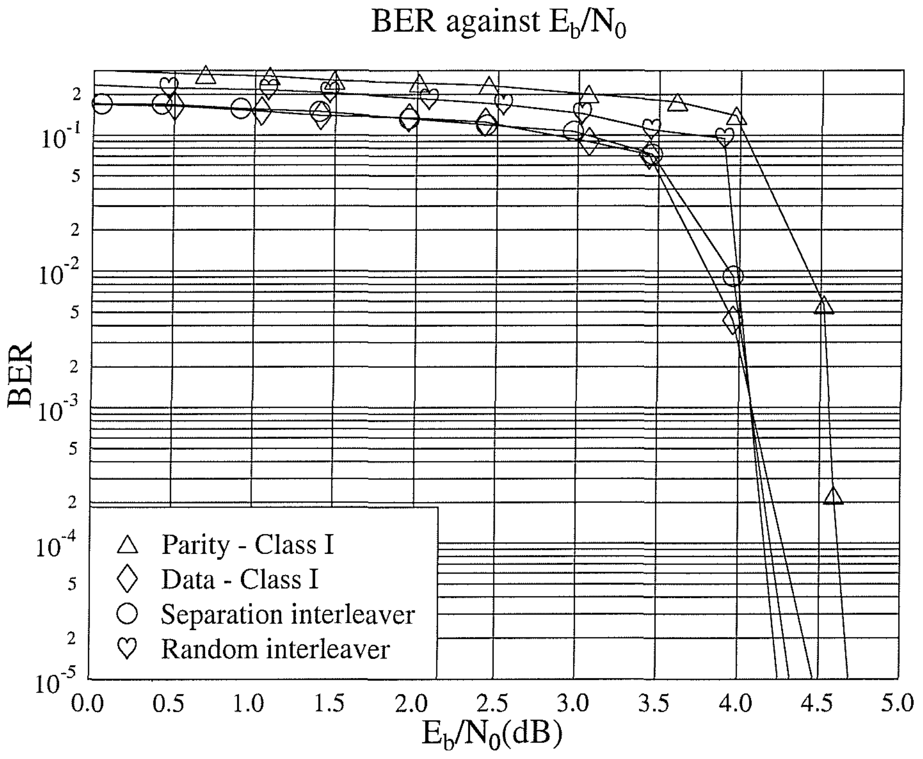


Figure 5.17: Performance comparison between different bit-to-symbol mapping methods for the **TC(2,1,5)** code in conjunction with the space-time code $\mathbf{G_2}$ using one receiver and **16QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TC(2,1,5) code are shown in Table 5.3.

interleaver in conjunction with the various TC codes.

5.5.2.3 Turbo BCH Codes

Figure 5.18 characterises the performance of the TBCH(32,26) code in conjunction with different bit-to-symbol mapping to the two protection classes of 8PSK. All simulation results were obtained with the aid of the space-time code $\mathbf{G_2}$ using one receiver and 8PSK over uncorrelated Rayleigh fading channels. Again, the encoding parameters of the TBCH(32,26) code are shown in Tables 5.3 and 5.5. The TBCH(32,26) code was chosen for our investigations, because the parity bits of the constituent encoders were not punctured and hence this resulted in a code rate of $R \approx \frac{2}{3}$. Roughly speaking, for every two data bits, there is one parity bit. Similarly to 16QAM, in the Gray-mapping assisted 8-PSK constellation there are also two protection classes, depending on the bit position in the 3-bit symbols. From the three bits of the 8-PSK constellation two of the bit positions are more protected, than the remaining bit. In Figure 5.18, we portray the performance of the TBCH(32,26)

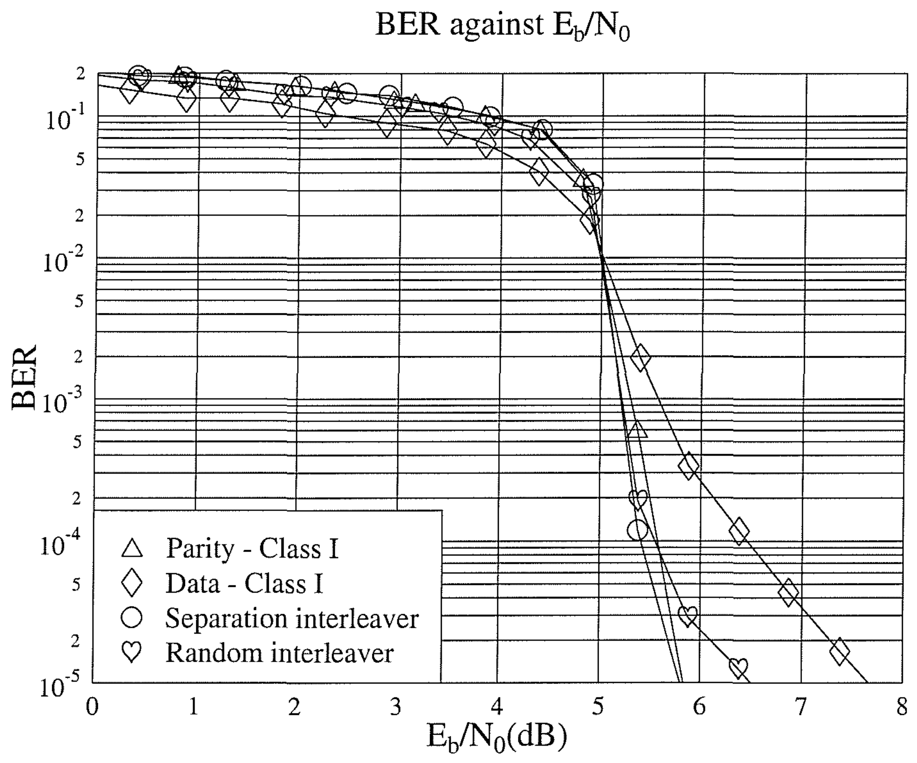


Figure 5.18: Performance comparison between different bit-to-symbol mapping methods for the TBCH(32,26) code in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and **8PSK** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TBCH(32,26) code are shown in Tables 5.3 and 5.5.

scheme for four different bit-to-symbol mapping methods. Firstly, one data bit and one parity bit was mapped to the two better protected 8-PSK bit positions. The corresponding BER curve was marked by the triangles in Figure 5.18. According to the second method, the data bits were mapped to the two better protected bit positions of the 8-PSK symbol. This scenario was marked by the diamonds in Figure 5.18. As we can see from the figure, the first mapping method yields a substantial E_b/N_0 gain of 1.5 dB at a BER of 10^{-5} over the second method. By applying the random separation based interleaver of Figure 5.14, while still better protecting one of the data bits and the parity bit than the remaining data bits, we disperse the bursty bit errors associated with a transmitted symbol over several BCH codewords of the turbo BCH code. As shown in Figure 5.18, the performance curve marked by the circles shows a slight improvement compared to the above-mentioned first method, although the difference is marginal. Finally, we show the performance of applying random interleaving, which randomly distributes the data and parity bits between the two 8-PSK protection classes. It can be seen that the associated performance is worse, than that of the first bit-to-symbol mapping method.

In Figure 5.18, we have shown that it is better to protect the parity bits more strongly for the TBCH(32,26) code and a slight further improvement can be achieved by applying a random separation based interleaver. More simulation results were obtained in conjunction with the other TBCH codes shown in Tables 5.3 and 5.5 with the aid of the space-time code $\mathbf{G}_2$ and 64QAM over uncorrelated Rayleigh fading channels. From the simulation results we have found that all TBCH codes shown in Tables 5.3 and 5.5 perform better, if the parity bits are more protected. In general, a slight further improvement can be obtained for TBCH codes, when a random separation based interleaver is applied. A possible explanation is that the component encoders of the TBCH codes are BCH encoders, where a block of parity bits is generated by a block of data bits. Hence, every parity bit has an influence on the whole codeword. Moreover, we used high-rate TBCH codes and hence there are more data bits compared to the parity bits. Hence, in our forthcoming TBCH comparisons, we will use bit-to-symbol mappers protecting the parity bits better.

5.5.2.4 Convolutional Codes

Let us now investigate the space-time code $\mathbf{G}_2$ in conjunction with the half-rate convolutional code CC(2,1,9) proposed for UMTS. The CC(2,1,9) code is a non-systematic non-recursive convolutional code, where the original information bits cannot be explicitly recognised in the encoded sequence. Its associated performance curve is shown in Figure 5.19 marked by the triangles. A random interleaver was then applied, in order to disperse the bursty channel errors and the associated performance curve is marked by the diamonds in Figure 5.19. At a BER of 10^{-5} there is a performance gain of 2.5 dB, if the random interleaver is applied. As a further scheme we invoked a systematic CC(2,1,9) code, which was obtained using a recursive convolutional code [49, 102]. Hence, in this scenario we have explicitly separable data bits and parity bits. In Figure 5.19 the performance curve marked by the circles is obtained by mapping the data bits of the systematic CC(2,1,9) code to protection class I of the associated 16QAM scheme in conjunction with the random separation based interleaver of Figure 5.14. From the figure we can see that there is only a marginal performance improvement over the non-systematic CC(2,1,9) code using the random interleaver.

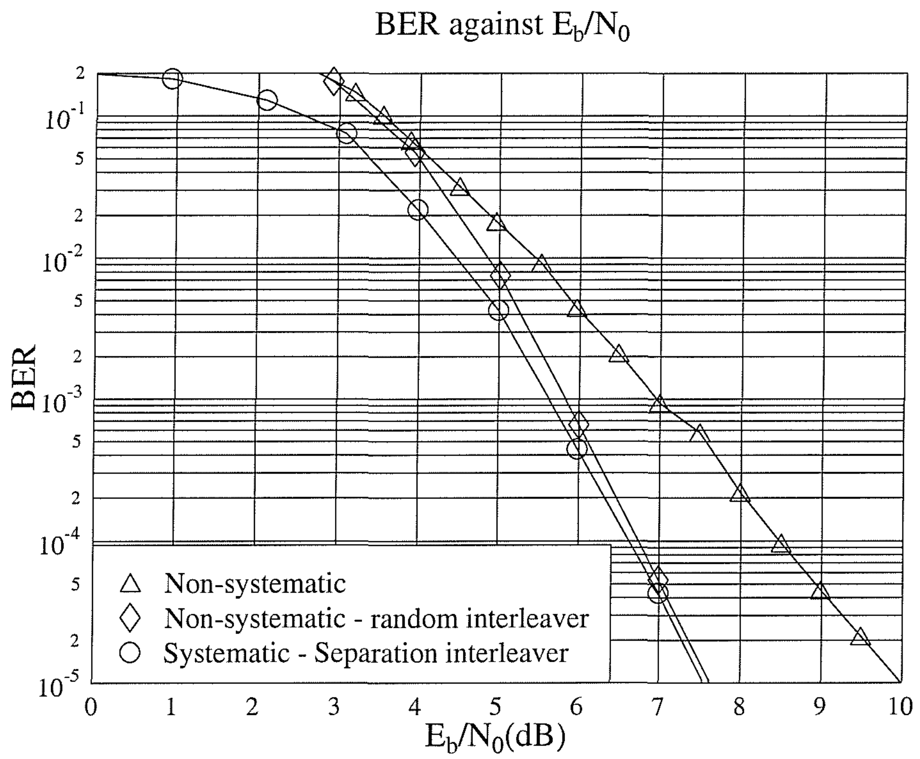


Figure 5.19: Performance comparison between the **systematic and non-systematic half-rate CC(2,1,9)** code in conjunction with the space-time code $\mathbf{G_2}$ and **16QAM** over uncorrelated Rayleigh fading channels at a throughput of **2 BPS**. The encoding parameters of the CC(2,1,9) code are shown in Tables 5.3 and 5.4.

5.5.3 Performance Comparison of Various Channel Codecs Using the G_2 Space-time Code and Multi-level Modulation

In this section we compare the $\mathbf{G_2}$ space-time coded performance of all channel codecs summarised in Table 5.3. In order to avoid having an excessive number of curves in one figure, only one channel codec will be characterised from each group of the CC, TC, TBCH, TCM and TTCM schemes. The choice of the channel codec considered depends on its performance, complexity and code rate. Unless otherwise stated, all channel codecs are concatenated with the space-time code $\mathbf{G_2}$ using one receiver. All comparison are carried out on the basis of the same BPS throughput over uncorrelated Rayleigh fading channels. Let us now briefly discuss in the forthcoming sections, how each channel codec is selected from the codec families considered.

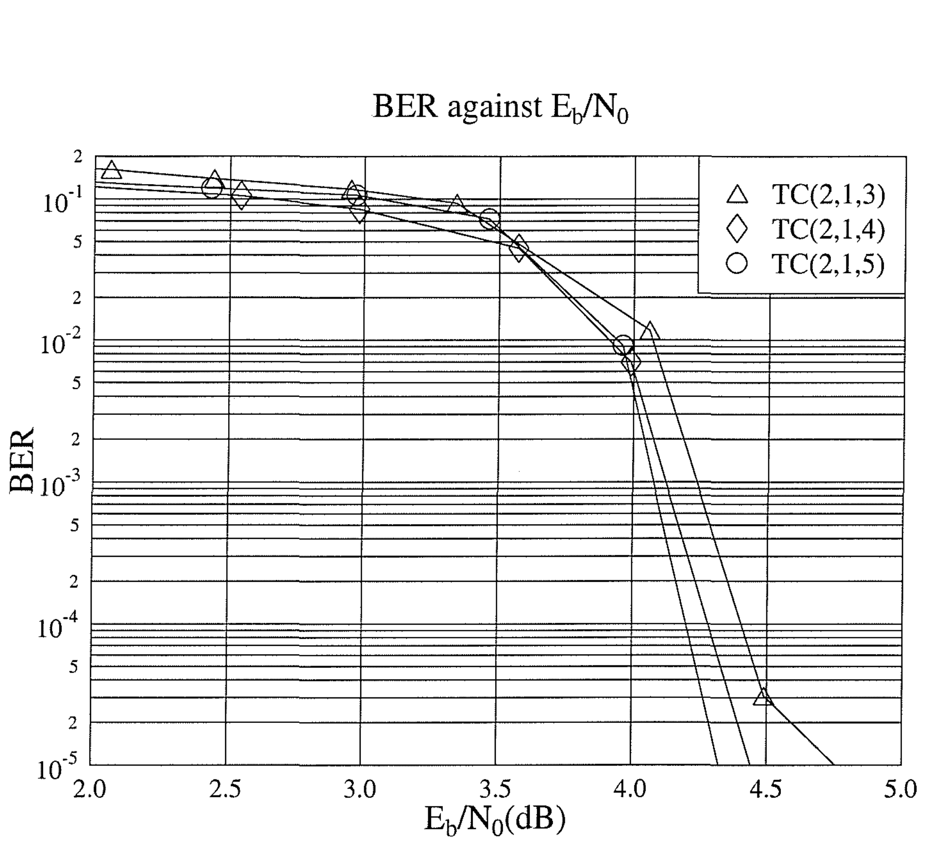


Figure 5.20: Performance comparison between the half-rate codes TC(2,1,3), TC(2,1,4) and TC(2,1,5), where the encoding parameters are shown in Table 5.3 and 5.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using **16QAM** over uncorrelated Rayleigh fading channels and the throughput was **2 BPS**.

5.5.3.1 Comparison of Turbo Convolutional Codes

In Figure 5.20 we compare the performance of the half-rate turbo codes TC(2,1,3), TC(2,1,4) and TC(2,1,5), where the encoding parameters are shown in Tables 5.3 and 5.5. The simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using 16QAM over uncorrelated Rayleigh fading channels. The three performance curves in the figure are the best performance curves chosen from Figures 5.17, 5.16 and 5.15 for the half-rate codes TC(2,1,5), TC(2,1,4) and TC(2,1,3), respectively. It can be seen from the figure that the performance of the turbo codes improves, when we increase the constraint length of the component codes from 3 to 5. However, this performance gain is obtained at the cost of a higher decoding complexity. At a BER of 10^{-5} the TC(2,1,4) code has an E_b/N_0 improvement of approximately 0.25 dB over the TC(2,1,3) scheme at a penalty of twice the complexity. However, at the cost of the same complexity increment over that of the TC(2,1,4) arrangement the TC(2,1,5) scheme only achieves a marginal performance gain of 0.1 dB at BER= 10^{-5}. Therefore, in our following investigations only the TC(2,1,4) scheme will be characterised as it exhibits a significant coding gain at a moderate complexity. Furthermore, the TC(2,1,4) code has been adopted by the 3G UTRA mobile communication system [49].

5.5.3.2 Comparison of Different Rate TC(2,1,4) Codes

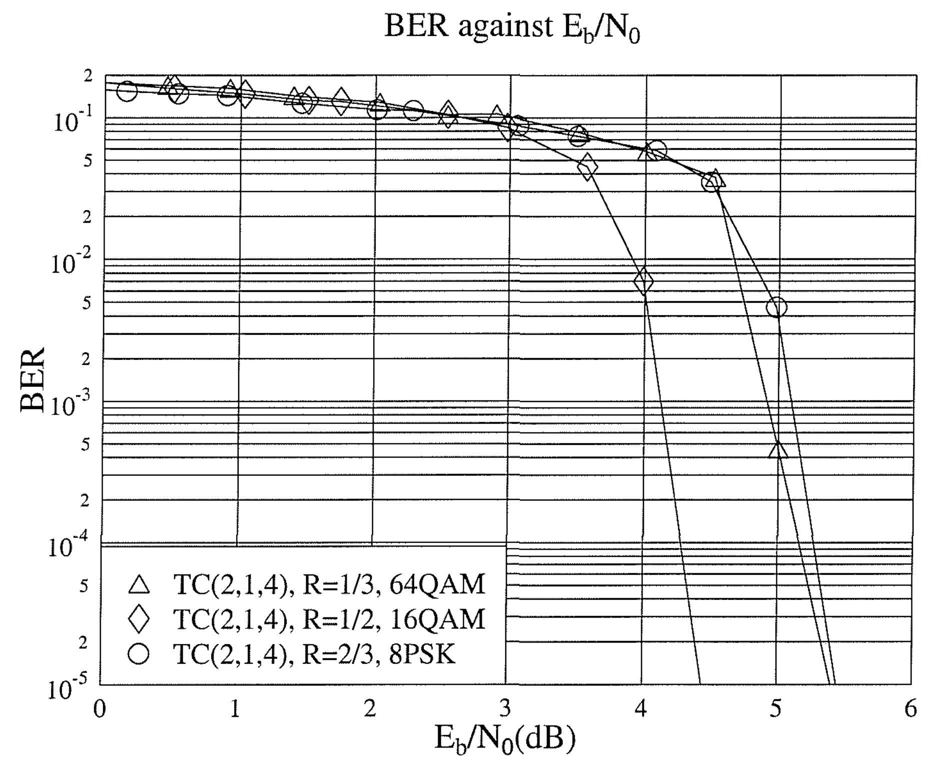


Figure 5.21: Performance of the TC(2,1,4) code using coding rates of $\frac{1}{3}$, $\frac{1}{2}$ and $\frac{2}{3}$, where the associated encoding parameters are shown in Tables 5.3 and 5.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at an effective throughput of **2 BPS** over uncorrelated Rayleigh fading channels.

In their seminal paper on turbo coding [12,13], Berrou *et al.* applied alternate puncturing of the parity bits. This results in half-rate turbo codes. However, additionally a range of different puncturing patterns can be applied, which results in different code rates [64]. In Figure 5.21 we portray the performance of the punctured TC(2,1,4) code having coding rates of $\frac{1}{3}$, $\frac{1}{2}$ and $\frac{2}{3}$. The associated coding parameters are shown in Tables 5.3 and 5.5. Suitable multi-level modulation schemes are chosen so that all systems have the same effective throughput of 2 BPS. Explicitly, 64QAM, 16QAM and 8PSK are used. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ over uncorrelated Rayleigh fading channels. As expected, from Figure 5.21 we can clearly see that the best performance is achieved by the half-rate TC(2,1,4) scheme. At a BER of 10^{-5} the half-rate TC(2,1,4) code achieved a performance gain of approximately 1 dB over the third-rate and the two-third-rate TC(2,1,4) codes. Even though the third-rate TC(2,1,4) code has a higher amount of redundancy than the half-rate TC(2,1,4) scheme, its performance is worse, than that of the half-rate TC(2,1,4) arrangement. We speculate that this is because the constellation points in 64QAM are more densely packed, than those of 16QAM. Therefore, they are more prone to errors and hence the extra coding power of the third-rate TC(2,1,4) code is insufficient to correct the extra errors. This results in a poorer performance. On the other hand, there are less errors induced by 8PSK, but the two-third-rate TC(2,1,4) code is a weak code due to the puncturing of the parity bits. Again, this results in an inferior performance.

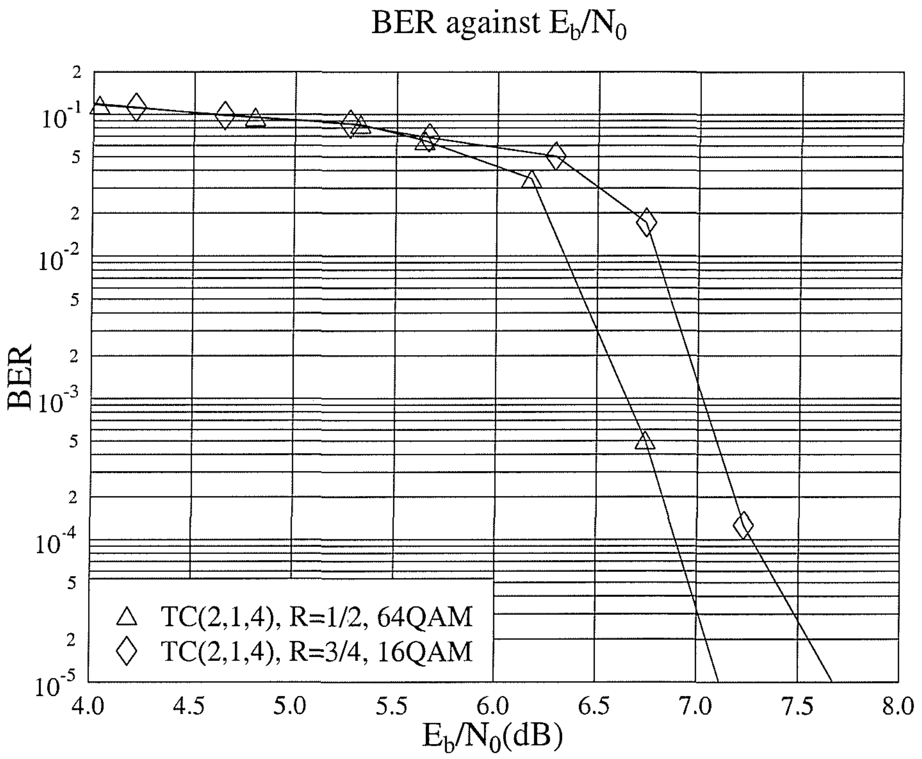


Figure 5.22: Performance of the punctured TC(2,1,4) code at coding rates of $\frac{1}{2}$ and $\frac{3}{4}$, where the associated parameters are shown in Tables 5.3 and 5.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at an effective throughput of **3 BPS** over uncorrelated Rayleigh fading channels.

In Figure 5.22 we show the performance of the TC(2,1,4) code at coding rates of $\frac{1}{2}$ and $\frac{3}{4}$. The associated coding parameters were shown in Tables 5.3 and 5.5. Again, suitable modulation schemes were chosen so that both systems have the same effective throughput, namely 3 BPS. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ over uncorrelated Rayleigh fading channels. As compared to Figure 5.21, the throughput of the systems in Figure 5.22 has been increased from 2 BPS to 3 BPS. In order to maintain a high BPS throughput, 64QAM was employed in conjunction with the half-rate TC(2,1,4) code. We can see from the figure that the performance gain of the half-rate TC(2,1,4) code over the three-quarter-rate TC(2,1,4) code has been reduced to only 0.5 dB, as compared to 1 dB over the two-third-rate TC(2,1,4) code characterised in Figure 5.21. Moreover, the three-quarter-rate TC(2,1,4) code is weaker, than the two-third-rate TC(2,1,4) code, since less parity bits are transmitted over the channel. Based on the fact that the performance gain of the half-rate TC(2,1,4) code has been reduced, we surmise that high-rate turbo codes will outperform the half-rate TC(2,1,4) code, if the throughput of the system is increased to 4 BPS or even further.

From Figures 5.21 and 5.22 we can see that the best performance is achieved by the half-rate TC(2,1,4) code for an effective throughput of 2 and 3 BPS. However, we are also interested in the system's performance at higher effective BPS throughputs. Hence, during our later discourse in Section 5.5.3.6 the performance of high-rate TC and TBCH codes will be studied for throughput values in excess of 5 BPS.

5.5.3.3 Convolutional Codes

In Figure 5.23 we compare the performance of the $\mathbf{G}_2$ space-time coded non-recursive half-rate convolutional codes CC(2,1,5), CC(2,1,7) and CC(2,1,9). These schemes were standardised in the GSM [49, 175], DVB [39] and the 3G UTRA systems [49, 154, 177], respectively. The associated coding parameters were shown in Tables 5.3 and 5.4. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using QPSK over uncorrelated Rayleigh fading channels. We can see from the figure that at a BER of 10^{-5} the performance of the non-recursive convolutional codes improves by approximately 1 dB, if the complexity is increased by a factor of $2^2 = 4$. However, the extra performance gain attainable becomes smaller, as the affordable complexity further increases. In our forthcoming channel code comparisons, only the CC(2,1,9) code will be used, since it has the best performance amongst the above three schemes and it has a comparable complexity to that of the turbo convolutional codes studied. Moreover, the CC(2,1,9) code is also proposed for the third generation UTRA mobile communication system [49].

5.5.3.4 $\mathbf{G}_2$ Coded Channel Codec Comparison – Throughput of 2 BPS

Having narrowed down the choice of the $\mathbf{G}_2$ space-time coded convolutional codes and the turbo codes, we are now ready to compare the performance of the different proposed channel codecs belonging to different codec families. Our comparison is carried out on the basis of

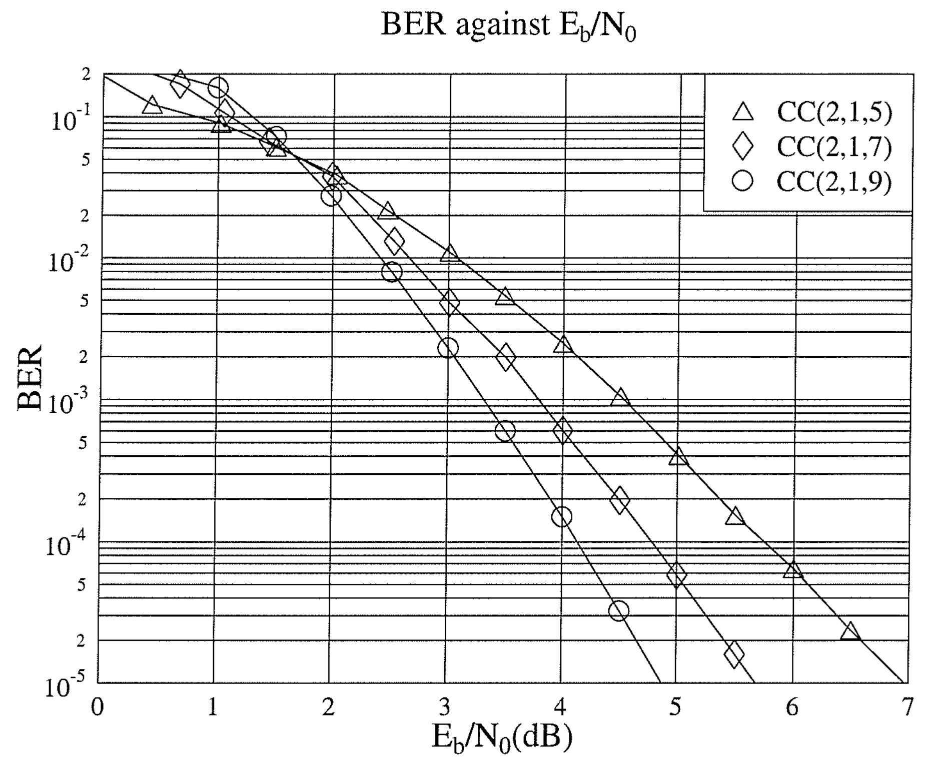


Figure 5.23: Performance comparison between the non-recursive half-rate convolutional codes CC(2,1,5), CC(2,1,7) and CC(2,1,9), where the coding parameters are shown in Tables 5.3 and 5.4. All simulation results were obtained with the aid of the space-time code $\mathbf{G_2}$ using **QPSK** over uncorrelated Rayleigh fading channels. The effective throughput is **1BPS**

the same throughput and all channel codecs are concatenated with the space-time code $\mathbf{G_2}$, when transmitting over uncorrelated Rayleigh fading channels. Figure 5.24 shows the performance of our channel codecs selected from the CC, TC, TBCH, TCM and TTCM families on the basis of the same throughput of 2 BPS, regardless of their coding rates. The associated coding parameters are shown in Tables 5.3, 5.4 and 5.5. The throughput is 2 BPS.

From Figure 5.24 we can see that the half-rate TC(2,1,4) code outperforms the other channel codecs. At a BER of 10^{-5} the TC(2,1,4) code achieves a gain of approximately 0.5 dB over the TBCH(31,21) scheme at a much lower complexity. At the same BER, the TC(2,1,4) code also outperforms 8PSK-TTCM by approximately 1.5 dB. The poor performance of TTCM might be partially due to using generator polynomials, which are optimum for AWGN channels [57]. However, to date only limited research has been carried out on finding optimum generator polynomials for TTCM over fading channels [178].

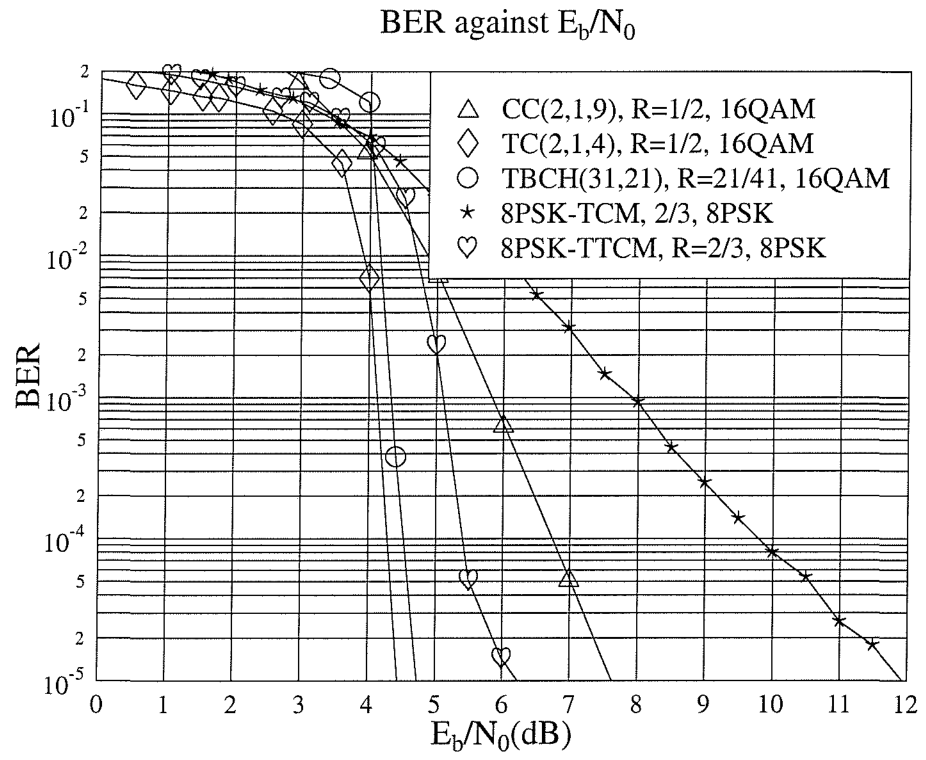


Figure 5.24: Performance comparison between different CC, TC, TBCH, TCM and TTCM schemes where the coding parameters are shown in Tables 5.3, 5.4 and 5.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at a throughput of **2 BPS** over uncorrelated Rayleigh fading channels.

In Figure 5.24 we also characterise the performance of the CC(2,1,9) and 8PSK-TCM schemes. The figure clearly demonstrates that the invention of turbo codes invoked in our TC, TBCH and TTCM $\mathbf{G}_2$-coded schemes, resulted in substantial improvements over the conventional $\mathbf{G}_2$-coded channel codecs, such as the CC and TCM schemes considered. At a BER of 10^{-5}, the TC(2,1,4) code outperforms the CC(2,1,9) and 8PSK-TCM arrangements by approximately 3.0 dB and 7.5 dB, respectively.

5.5.3.5 $\mathbf{G}_2$-Coded Channel Codec Comparison – Throughput of 3 BPS

In Figure 5.25 we portray the performance of various channel codecs belonging to the CC, TC, TBCH, TCM and TTCM codec families on the basis of a constant throughput of 3 BPS, regardless of their coding rates. The associated coding parameters are shown in Tables 5.3, 5.4 and 5.5. The simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ over uncorrelated Rayleigh fading channels.

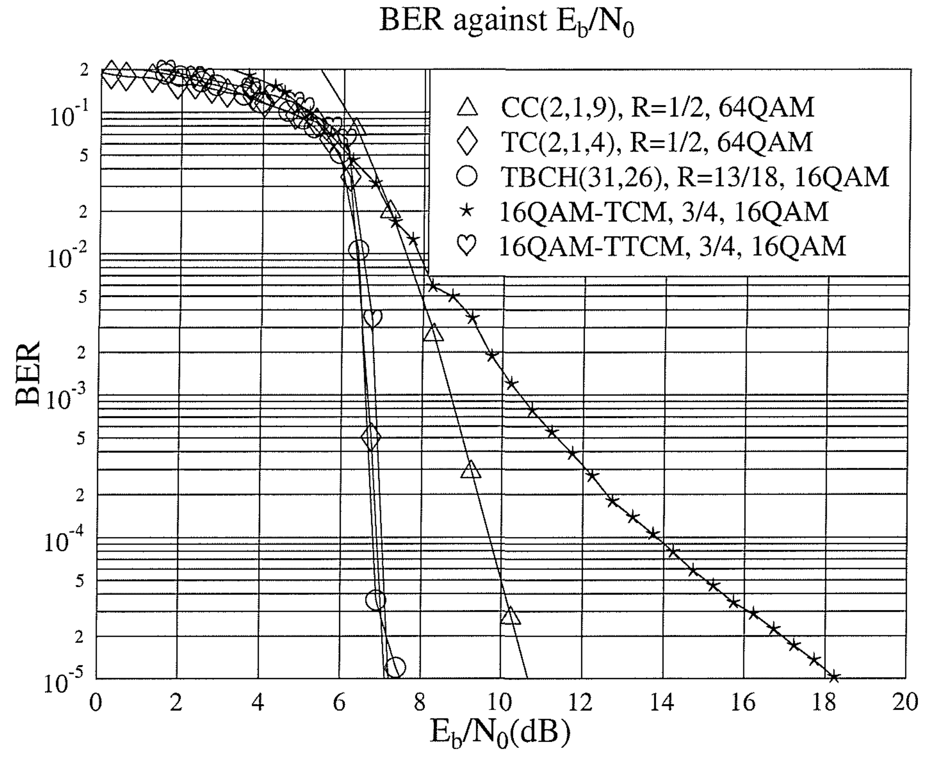


Figure 5.25: Performance comparison between different CC, TC, TBCH, TCM and TTCM schemes where the coding parameters are shown in Tables 5.3, 5.4 and 5.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at an effective throughput of **3 BPS** over uncorrelated Rayleigh fading channels.

From Figure 5.25 we can infer a few interesting points. As mentioned earlier, the half-rate TC(2,1,4) code suffers from the effects of puncturing as we increase the throughput of the system. In order to maintain a throughput of 3 BPS, 64QAM has to be employed in the systems using the half-rate TC(2,1,4) code. The rather vulnerable 64QAM modulation scheme appears to over-stretch the coding power of the half-rate TC(2,1,4) code attempting to saturate the available channel capacity. At a BER of 10^{-5} there is no obvious performance gain over the TBCH(31,26)/16QAM and 16QAM-TTCM schemes. Hence, we have reasons to postulate that if the throughput of the system is increased beyond 3 BPS, high-rate turbo codes should be employed for improving the performance, rather than invoking a higher throughput modulation scheme.

5.5.3.6 Comparison of $\mathbf{G}_2$-Coded High-Rate TC and TBCH Codes

In the previous section we have shown that at the BER of 10^{-5}, the required E_b/N_0 is increased by about 2.5 dB for the half-rate turbo code TC(2,1,4), as the throughput of the

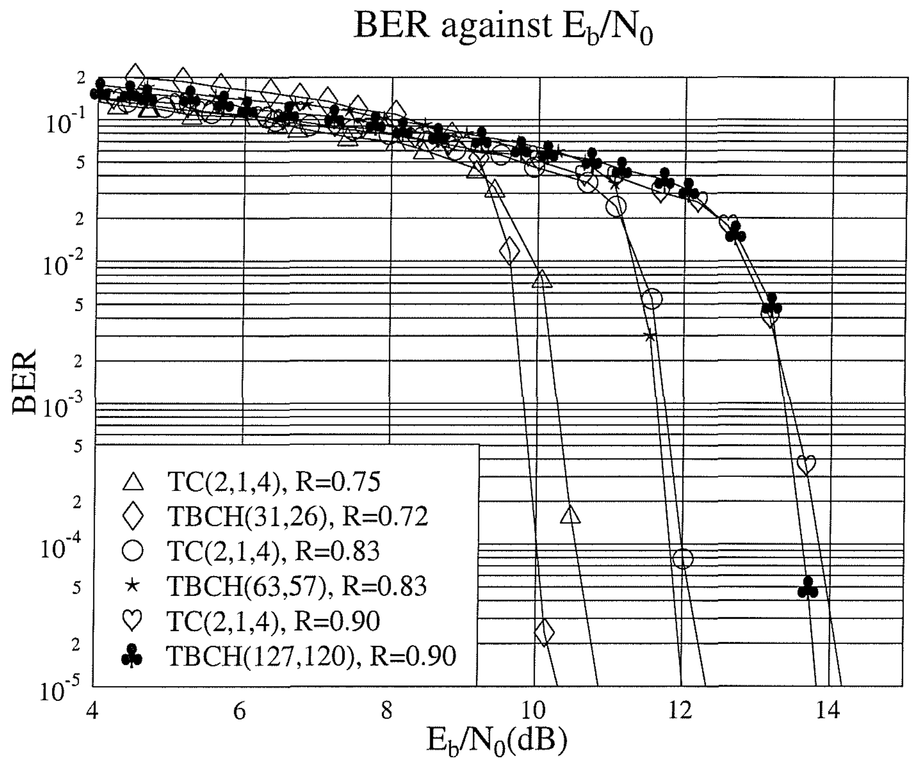


Figure 5.26: Performance comparison between high-rate TC and TBCH codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The parameters of the TC and TBCH codes were shown in Tables 5.3 and 5.5.

system is increased from 2 BPS to 3 BPS. A range of schemes having a throughput in excess of 5 BPS is characterised in Figure 5.26. Specifically, the figure shows the performance of high-rate TC and TBCH codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The parameters of the TC and TBCH codes used are shown in Tables 5.3 and 5.5. The performance of half-rate turbo codes along with such a high throughput is not shown, because a modulation scheme having at least 1024 constellation points would be needed, which is practically infeasible over non-stationary wireless channels. Moreover, the turbo codes often would be overloaded with the plethora of errors induced by the densely packed constellation points.

In Figure 5.26 we can clearly see that there is not much difference in performance terms between the high-rate TC(2,1,4) and TBCH codes employed, although the TBCH codes exhibit marginal gains. This gain is achieved at a cost of high decoding complexity, as evidenced by Table 5.6. The slight performance improvement of the TBCH(31,26) code over the three-quarter rate TC(2,1,4) scheme is probably due to its slightly lower code rate of $R = 0.72$, compared to the rate of $R = 0.75$ associated with the TC(2,1,4) code. It is important to note that all BCH component codes used in the TBCH codes have a minimum distance d_{min} of 3. We speculate that the performance of the TBCH codes might improve, if d_{min} is increased to 5. However, due to the associated complexity we will refrain from employing $d_{min} = 5$ BCH component codes in the TBCH schemes studied.

5.5.3.7 Comparison of High-Rate TC and Convolutional Codes

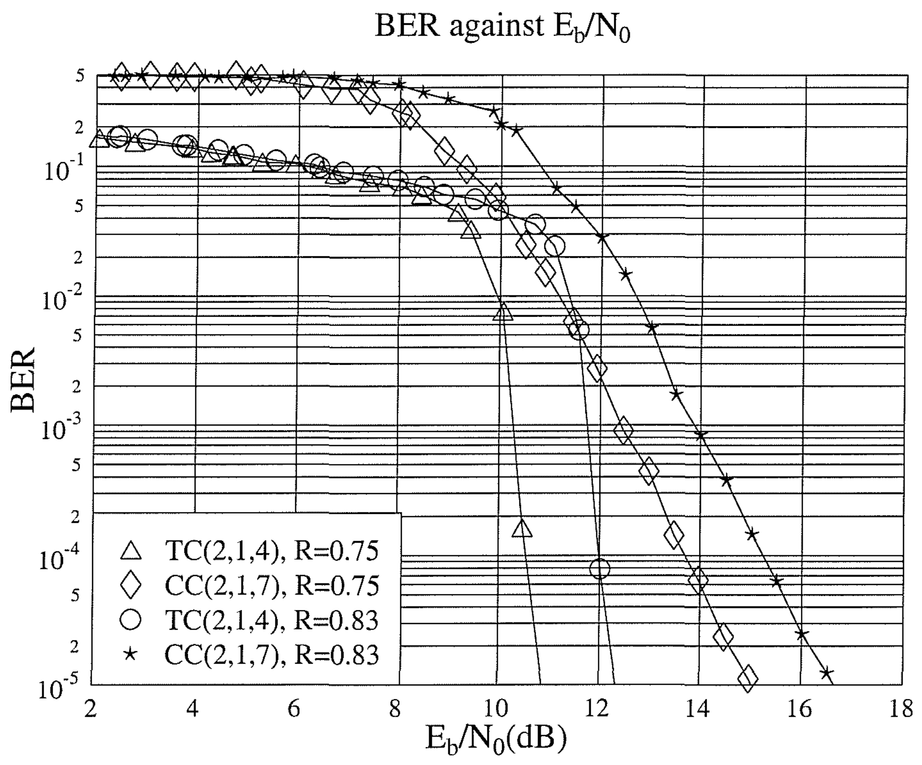


Figure 5.27: Performance comparison between high-rate TCs and convolutional codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The parameters of the TC and CC codes were shown in Tables 5.3, 5.4 and 5.5.

In Figure 5.27, we compare the performance of the high-rate punctured TC(2,1,4) and CC(2,1,7) codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The puncturing patterns employed for the CC(2,1,7) scheme were proposed in the DVB standard [39]. The parameters of the TC(2,1,4) and CC(2,1,7) codes are shown in Tables 5.3, 5.4 and 5.5. From the figure we can see that both high-rate TC(2,1,4) codes outperform their equivalent rate CC(2,1,7) counterparts by about 2 dB at a BER of 10^{-5}, whilst maintaining a similar estimated decoding complexity, as it was evidenced by Table 5.6. This fact indicates that at a given tolerable complexity, better BER performance can be attained by an iterative turbo decoder. These findings motivated the investigations of our next section, where the performance of the various schemes was studied in the context of the achievable coding gain versus the estimated decoding complexity.

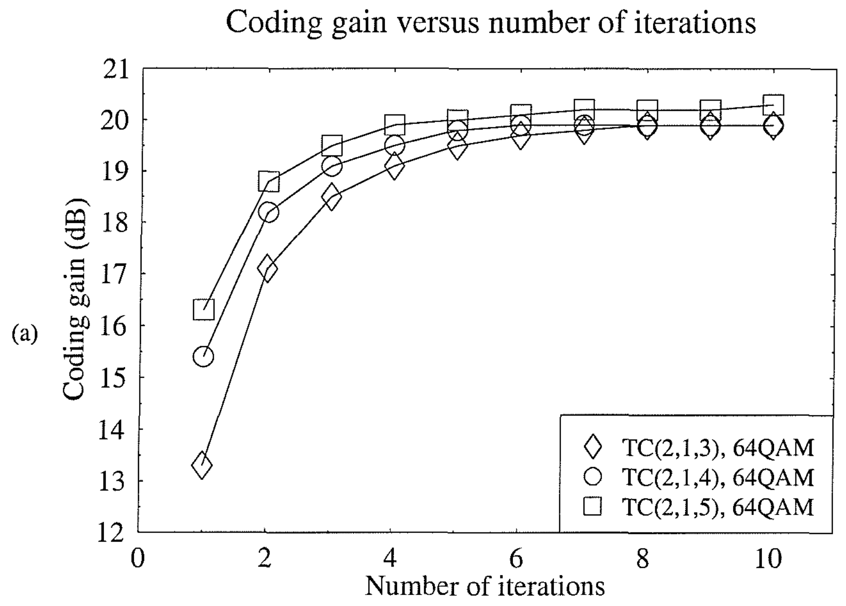
5.5.4 Coding Gain Versus Complexity

In Section 5.4.3 we have estimated the various channel decoders' complexity based on a few simplifying assumptions. All the complexities estimated in our forthcoming discourse were calculated based on Equations 5.43 to 5.51. Again, our performance comparison of the channel codes was made on the basis of the coding gain defined as the E_b/N_0 difference, expressed in decibels, at BER$= 10^{-5}$ between the various channel coded and uncoded systems having the same throughput, while using the space-time code $\mathbf{G}_2$.

5.5.4.1 Complexity Comparison of Turbo Convolutional Codes

Figure 5.28 shows the (a) coding gain versus the number of iterations and (b) the coding gain versus estimated complexity for the TC(2,1,3), TC(2,1,4) and TC(2,1,5) codes, where the coding parameters used are shown in Tables 5.3, 5.5 and 5.6. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver and 64QAM over uncorrelated Rayleigh fading channels at an effective throughput of 3 BPS. We can see from Figure 5.28(a) that there is a huge performance improvement of approximately $3 - 4$ dB between the first and second turbo decoding iteration. However, the further coding gain improvements become smaller, as the number of iterations increases. It can be seen from the figure that the performance of turbo codes does not significantly improve after 8 iterations, as indicated by the rather flat coding gain curve. Figure 5.28(a) also shows that as we increase the constraint length K of the turbo codes from 3 to 5, the associated performance improves.

In Figure 5.28(b) the coding gains of the various turbo codes using different number of iterations were compared on the basis of their estimated complexity. This was necessary, since we have seen in Section 5.4.3 that the estimated complexity of turbo codes depends exponentially on the constraint length K, but only linearly on the number of iterations. From Figure 5.28(b), we can see that the estimated complexity of the TC(2,1,5) code ranges from approximately 200 to 2000, when using one to ten iterations. On the other hand, the estimated complexity of the TC(2,1,3) scheme ranges only from approximately 50 to 500 upon invoking one to ten iterations. This clearly shows that the estimated complexity of the turbo codes is dominated by the constraint length K. Figure 5.28(b) also shows that



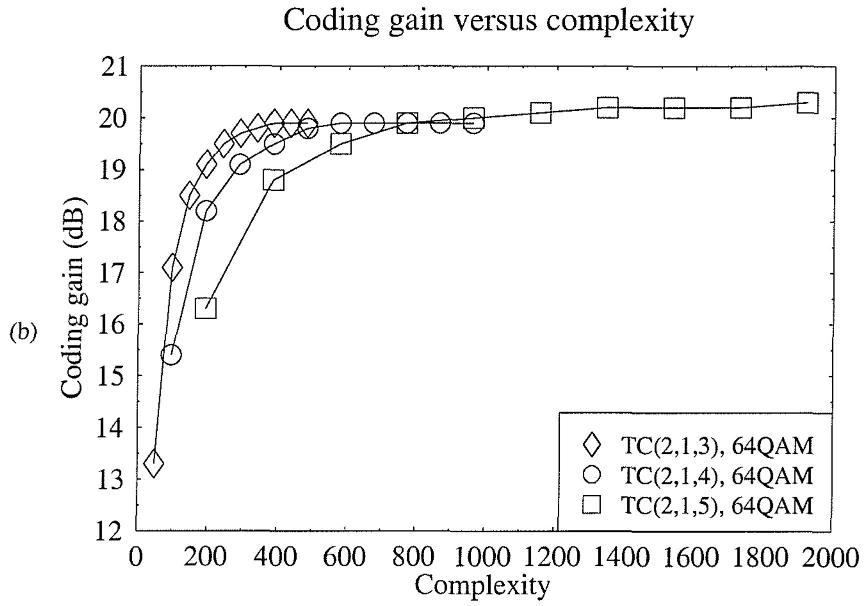


Figure 5.28: Coding gain versus (a) the number iterations and versus (b) estimated complexity for the TC(2,1,3), TC(2,1,4) and TC(2,1,5) codes, where the coding parameters are shown in Tables 5.3, 5.5 and 5.6. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver and **64QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **3 BPS**.

the coding gain curve of the TC(2,1,3) code saturates faster, which is demonstrated by the steep increase in coding gain, as the estimated complexity increases. For achieving the same coding gain of 19 dB, we can see that the TC(2,1,3) scheme requires the lowest estimated complexity. We would require 2-3 times higher computational power for the TC(2,1,5) code to achieve the above-mentioned coding gain of 19 dB.

5.5.4.2 Complexity Comparison of Channel Codes

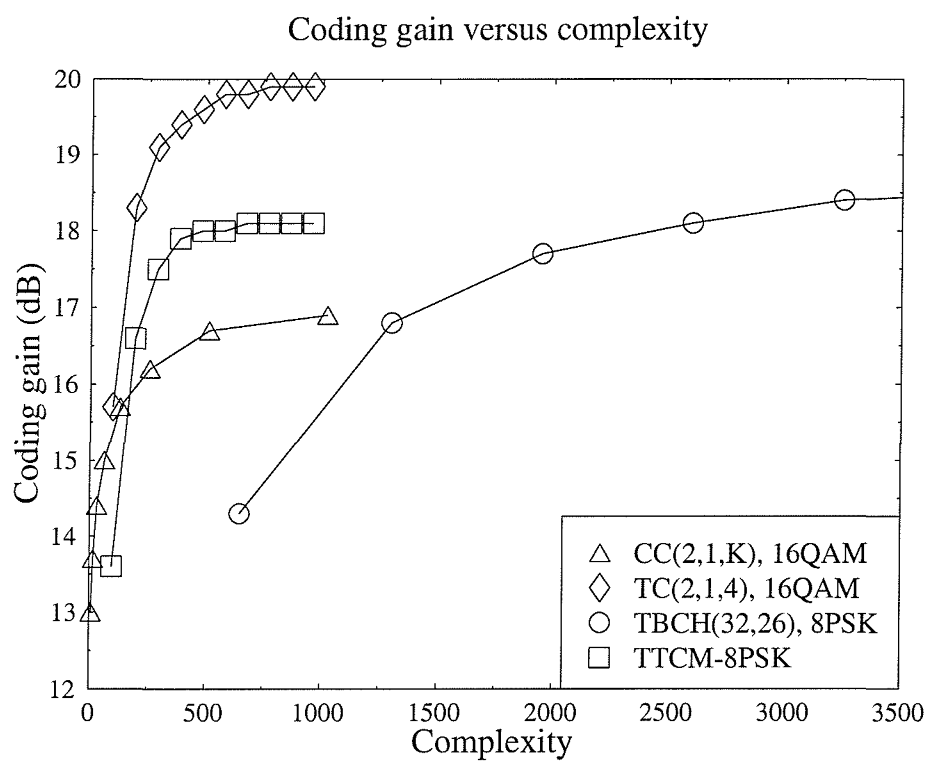


Figure 5.29: Coding gain versus estimated complexity for the CC(2,1,K), TC(2,1,4), TBCH(32,26) and TTCM-8PSK where the parameters are shown in Table 5.3, 5.4, 5.5 and 5.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**.

In the previous section we have compared the coding gain versus estimated complexity of the $\mathbf{G}_2$-coded turbo schemes TC(2,1,3), TC(2,1,4) and TC(2,1,5). Here we compare the TC(2,1,4) arrangement that faired best amongst them to the CC(2,1,9) code and to the TBCH(32,26)/8PSK as well as to the TTCM-8PSK arrangements, representing the other codec families studied. Specifically, Figure 5.29 shows the coding gain versus estimated complexity for the CC(2,1,K), TC(2,1,4), TBCH(32,26) and TTCM-8PSK schemes, where the associated parameters are shown in Tables 5.3, 5.4, 5.5 and 5.6. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of 2 BPS. For the turbo schemes TC(2,1,4), TBCH(32,26) and TTCM-8PSK the increased estimated complexity is achieved by increasing the number of iterations from 1 to 10. However, convolutional codes are decoded non-iteratively. Therefore in Figure 5.29 we vary the constraint length K of the convolutional codes from 3 to 10, which results in increased estimated complexity. The generator polynomials of the CC(2,1,K) codec, where $K = 3...10$, are given in [102] and they define the corresponding maximum minimum free distance of the codes. From Figure 5.29 we can see that there is a steep increase in the coding gain achieved by the TC(2,1,4) code, as the estimated complexity is increased. Moreover, the TC(2,1,4) scheme asymptotically achieves a maximum coding gain of approximately 20 dB. At a low estimated complexity of approximately 200, the TC(2,1,4) code attains a coding gain of approximately 18 dB, which exceeds that of the other channel codes studied. The TBCH(32,26) arrangement is the least attractive one, since a huge estimated complexity is incurred, when aiming for a high coding gain.

In contrast to the 2 BPS schemes of Figure 5.29, Figure 5.30 shows the corresponding coding gain versus estimated complexity curves for the CC(2,1,K), TC(2,1,4), TBCH(31,26) and TTCM-16QAM 3 BPS arrangements, where the coding parameters are shown in Tables 5.3, 5.4, 5.5 and 5.6. Again, all simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of 3 BPS. As before, the increased estimated complexity of the turbo schemes is incurred by increasing the number of iterations from 1 to 10. For the convolutional codes the constraint length K is varied from 3 to 10. Similarly to Figure 5.29, the TC(2,1,4) scheme achieves a considerable coding gain at a relatively low estimated complexity. For example, in order to achieve a coding gain of 18 dB, the TTCM and TBCH(31,26) arrangements would require an approximately 3 and 4 times higher computational power compared to the TC(2,1,4) code.

From Figures 5.29 and 5.30 we can clearly see that turbo codes are the most attractive one of all the channel codes studied in conjunction with the space-time code $\mathbf{G}_2$, offering an impressive coding gain at a moderate estimated decoding complexity.

In Figure 5.31, we show the E_b/N_0 value required for maintaining BER$= 10^{-5}$ versus the effective throughput BPS for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,4) code where the coding parameters are shown in Tables 5.3, 5.5 and 5.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using **one receiver** over uncorrelated Rayleigh fading channels. Half-rate TC(2,1,4) code was employed for BPS

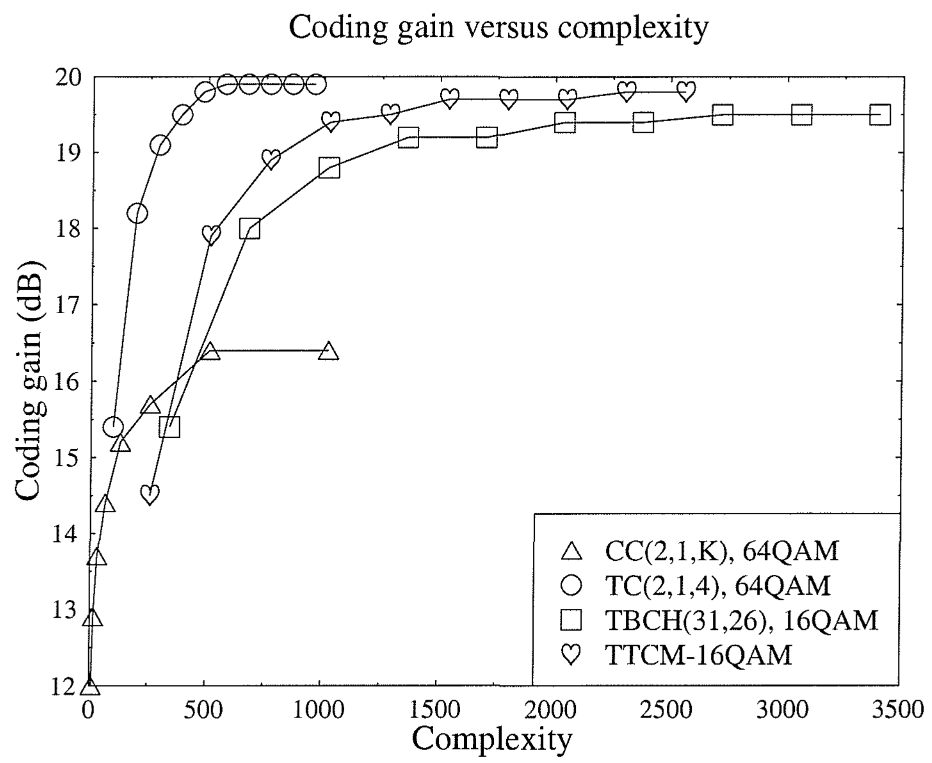


Figure 5.30: Coding gain versus estimated complexity for the CC(2,1,K), TC(2,1,4), TBCH(31,26) and TTCM-16QAM schemes where the coding parameters are shown in Tables 5.3, 5.4, 5.5 and 5.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of **3 BPS**.

up to three. Then TC(2,1,4) code with various rates was employed with 64QAM in order to achieve increasing effective throughput BPS. It can be seen from the figure that the E_b/N_0 value required for maintaining BER$= 10^{-5}$ increases linearly as the effective throughput BPS increases.

5.6 Summary and Conclusions

The state-of-the-art of transmission schemes based on multiple transmitters and receivers was reviewed in Section 5.1 This was followed by a rudimentary introduction to MRC [71] technique, using a simple example in Section 5.2.1. Space-time block codes were introduced in Section 5.3 employing the unity-rate space-time code $\mathbf{G}_2$. In Sections 5.3.1.1 and 5.3.1.2 two examples of employing the space-time code $\mathbf{G}_2$ were provided using one and two receivers, respectively. The transmission matrix of a range of different-rate space-time codes,

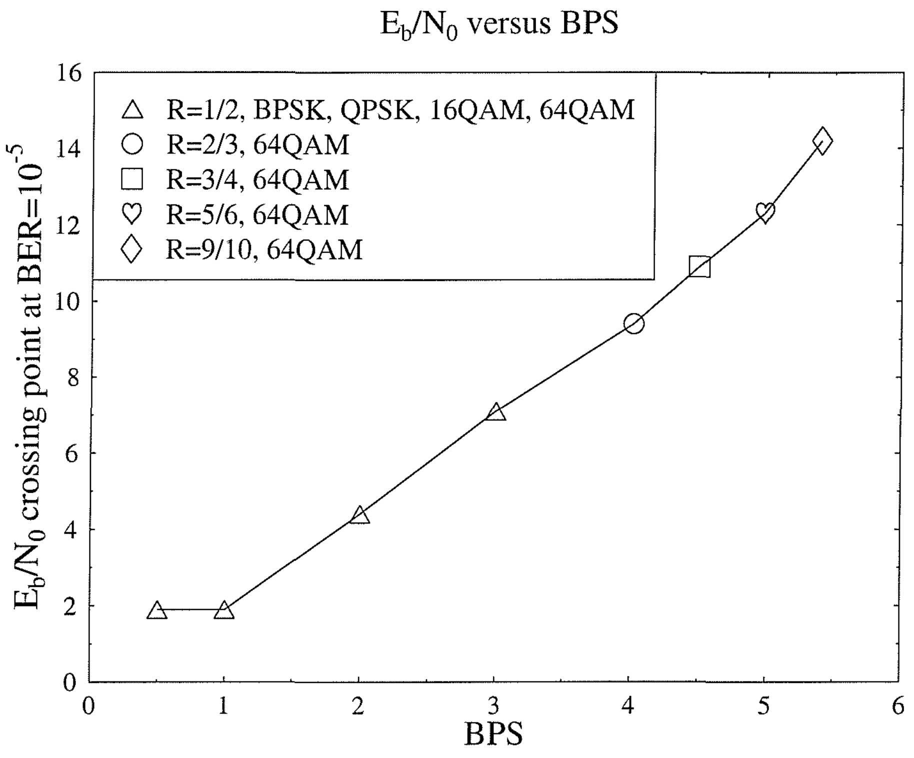


Figure 5.31: The E_b/N_0 value required for maintaining BER= 10^{-5} versus the effective throughput BPS for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,4) code where the coding parameters are shown in Tables 5.3, 5.5 and 5.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using **one receiver** over uncorrelated Rayleigh fading channels.

namely that of the codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 5.2 were also given. Additionally, a brief description of the MAP decoding algorithm [171] was provided in Section 5.3.3 in the context of space-time block codes.

In Section 5.4 we proposed a system, which consists of the concatenation of the above-mentioned space-time block codes and a range of different channel codes. The channel coding schemes investigated were convolutional codes, turbo convolutional codes, turbo BCH codes, trellis coded modulation and turbo trellis coded modulation. The estimated complexity and memory requirement of the channel decoders were summarised in Section 5.4.3.

Finally, we presented our simulation results in Section 5.5, which were divided into four categories. In Section 5.5.1, we first compared the performance results of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ without using channel codecs. It was found that as we increased the effective throughput of the system, the performance of the half-rate space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ degraded in comparison to that of the unity rate space-time code $\mathbf{G}_2$. This was because in order to maintain the same effective throughput, higher modulation schemes had to be employed in conjunction with the half-rate space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$, which were more prone to errors and hence degraded the performance of the system. On the other hand, for the sake of maintaining the same diversity gain and same effective throughput we found that the performance of the space-time codes $\mathbf{H}_3$ and $\mathbf{H}_4$ was better, than that of the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$, respectively. Since the space-time code $\mathbf{G}_2$ has a code rate of unity, we were able to concatenate it with half-rate TC codes, while maintaining the same effective throughput, as the half-rate space-time code using no channel coding. Hence for the same effective throughput, the unity-rate $\mathbf{G}_2$ space-time coded and half-rate channel coded scheme provided substantial performance improvement over the three-quarter rate space-time code $\mathbf{H}_4$ and half-rate space-time code $\mathbf{G}_4$, which were unable to benefit from channel coding. We concluded that the reduction in coding rate was best invested in turbo channel codes, rather than space-time block codes. Therefore, all channel codes studied were concatenated with the unity-rate space-time code $\mathbf{G}_2$ only.

In the second category of our investigations in Section 5.5.1.5 we studied the effect of the binary channel codes' data and parity bits mapped into different protection classes of multi-level modulation schemes. It was found that TC codes having different constraint lengths K require different mapping methods, as evidenced by Figure 5.12. By contrast, in the turbo BCH codes studied mapping of the parity bits to the higher-integrity protection class of a multi-level modulation scheme yielded a better performance. The so-called random separation based interleaver was proposed, in order to improve the performance of the system.

The third set of results compared the performance of all proposed channel codes in conjunction with the space-time code $\mathbf{G}_2$. In order to avoid confusion, we only selected one channel code from each group of channel codes in Table 5.3. Specifically, only half-rate TC codes were studied, as they gave better coding gain performance compared to other TC codes having lower and higher rates. It was then found that the performance of the half-rate TC codes was better than that of the CC, TBCH, TCM and TTCM codes. Then, we compared the performance of high rates TC codes with high-rate turbo BCH codes in conjunction with 64QAM. It was found that the turbo BCH codes provided a slight performance improvement over high rate TC codes, but at the cost of high complexity. Finally, the chapter was concluded by comparing the $\mathbf{G}_2$ space-time coded channel codes upon taking their estimated complexity into consideration. In Figures 5.29 and 5.30, we can clearly see that the half-rate TC codes give the best coding gain at a moderate estimated complexity.

Chapter 6

Space-Time Trellis Codes

6.1 Introduction

In the previous chapter, we have detailed the encoding and decoding processes of *space-time block codes* [71–73]. Various proposed space-time block codes [72, 73] have been discussed and their performance was investigated over perfectly interleaved, non-dispersive Rayleigh fading channels. A range of systems consisting of space-time block codes and different channel codecs were proposed. The performance versus estimated complexity trade-off of the different systems was investigated and compared.

In an effort to provide as comprehensive a technology road-map as possible and to identify the most promising schemes in the light of their performance versus estimated complexity, in this chapter we shall explore the family of *space-time trellis codes* [70,80,166–169], which were proposed by Tarokh *et al.* Space-time trellis codes incorporate jointly designed channel coding, modulation, transmit diversity and optional receiver diversity. The performance criteria for designing space-time trellis codes were outlined in [70], under the assumption that the channel is fading slowly and that the fading is frequency non-selective. It was shown in [70] that the system's performance is determined by matrices constructed from pairs of distinct code sequences. Both the *diversity gain* and *coding gain* of the codes are determined by the minimum rank and the minimum determinant [70, 179] of the matrices, respectively. The results were then also extended to fast fading channels. The space-time trellis codes proposed in [70] provide the best tradeoff between data rate, diversity advantage and trellis complexity.

The performance of both space-time trellis and block codes over narrowband Rayleigh fading channels was investigated by numerous researchers [70, 71, 73, 79, 80]. The investigation of space-time codes was then also extended to the class of practical wideband fading channels. The effect of multiple paths on the performance of space-time trellis codes was studied in [169] for transmission over slowly varying Rayleigh fading channels. It was shown in [169] that the presence of multiple paths does not decrease the diversity order guaranteed by the design criteria used to construct the space-time trellis codes. The evidence provided in [169] was then also extended to rapidly fading dispersive and non-dispersive channels. As a further performance improvement, turbo equalisation was employed in [74] in order to mitigate the effects dispersive channels. However space-time coded turbo equalisation involved an enormous complexity. In addressing the complexity issues, Bauch *et al.* [75] derived finite-length multi-input multi-output (MIMO) channel filters and used them as prefilters for turbo equalisers. These prefilters significantly reduce the number of turbo equaliser states and hence mitigate the decoding complexity. As an alternative solution, the effect of Inter Symbol Interference (ISI) could be eliminated by employing Orthogonal Frequency Division Multiplexing (OFDM) [153]. A system using space-time trellis coded OFDM is attractive, since the decoding complexity reduced, as demonstrated by the recent surge of research interests [76–79]. In [76,78,79], non-binary Reed-Solomon (RS) codes were employed in the space-time trellis coded OFDM systems for improving its performance.

Similarly, the performance of space-time block codes was also investigated over frequency selective Rayleigh fading channels. In [180], a multiple input multiple output equaliser was utilised for equalising the dispersive multipath channels. Furthermore, the advantages of OFDM were also exploited in space-time block coded systems [79,181,182].

We commence our discussion with a detailed description of the encoding and decoding processes of the space-time trellis codes in Section 6.2. The state diagrams of a range of other space-time trellis codes are also given in Section 6.2.2. In Section 6.3, a specific system is proposed, which enables the comparison of space-time trellis codes and space-time block codes over wideband channels. Our simulation results are then given in Section 6.4. We continue our investigations by proposing space-time coded adaptive modulation based OFDM in Section 6.5. Finally, we conclude the chapter in Section 6.6.

6.2 Space-Time Trellis Codes

In this section, we will detail the encoding and decoding processes of space-time trellis codes. Space-time trellis codes are defined by the number of transmitters p, by the associated state diagram and the modulation scheme employed. For ease of explanation, as an example we shall use the simplest 4-state, 4-level Phase Shift Keying (4PSK) space-time trellis code, which has $p = 2$ two transmit antennas.

6.2.1 The 4-State, 4PSK Space-Time Trellis Encoder

At any time instant k, the 4-state 4PSK space-time trellis encoder transmits symbols $x_{k,1}$ and $x_{k,2}$ over the transmit antennas Tx 1 and Tx 2, respectively. The output symbols at time instant k are given by [70]:

$$x_{k,1} = 0.d_{k,1} + 0.d_{k,2} + 1.d_{k-1,1} + 2.d_{k-1,2} \tag{6.1}$$

$$x_{k,2} = 1.d_{k,1} + 2.d_{k,2} + 0.d_{k-1,1} + 0.d_{k-1,2} \tag{6.2}$$

where $d_{k,i}$ represents the current input bits, whereas $d_{k-1,i}$ the previous input bits and $i = 1, 2$. More explicitly, we can represent Equation 6.2 with the aid of a shift register, as shown in Figure 6.1, where $\oplus$ represents modulo 4 addition. Let us explain the operation

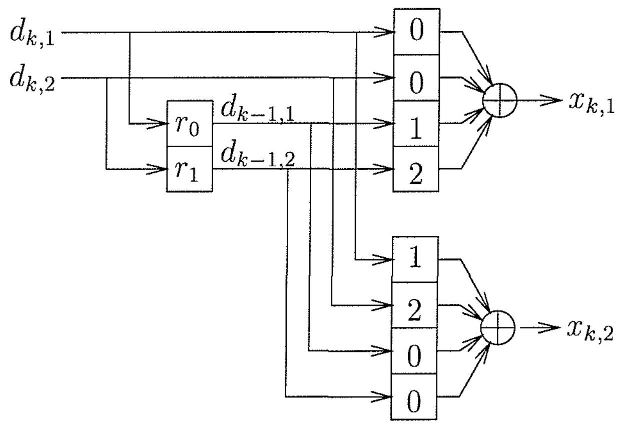


Figure 6.1: The 4-state, 4PSK space-time trellis encoder.

of the shift register encoder for the random input data bits 01111000. The shift register stages r_0 and r_1 must be reset to zero before the encoding of a transmission frame starts. They represent the state of the encoder. The operational steps are summarised in Table 6.1. Again, given the register stages $d_{k-1,1}$ and $d_{k-1,2}$ as well as the input bits $d_{k,1}$ and $d_{k,2}$, the output symbols seen in the table are determined according to Equation 6.2 or Figure 6.1. Note that the last two binary data bits in Table 6.1 are intentionally set to zero in order to force the 4-state 4PSK trellis encoder back to the zero state which is common practice at the end of a transmission frame. Therefore, the transmit antenna Tx 1 will transmit symbols $0, 2, 3, 1$. By contrast, symbols $2, 3, 1, 0$ are then transmitted by the antenna Tx 2.

According to the shift register encoder shown in Figure 6.1, we can find all the legitimate subsequent states, which result in transmitting the various symbols $x_{k,1}$ and $x_{k,2}$, depending on a particular state of the shift register. This enables us to construct the state diagram for the encoder. The 4PSK constellation points are seen in Figure 6.2, while the

Input queue	Instant k	Input bits $(d_{k,1}; d_{k,2})$	Shift register $(d_{k-1,1}; d_{k-1,2})$	State S_k	Transmitted symbols $(x_{k,1}; x_{k,2})$
00011110	0	- -	0 0	0	- -
000111	1	0 1(2)	0 0	0	0 2
0001	2	1 1(3)	0 1	2	2 3
00	3	1 0(1)	1 1	3	3 1
	4	0 0(2)	1 0	1	1 0
	5	- -	0 0	0	- -

Table 6.1: Operation of the space-time encoder of Figure 6.1.

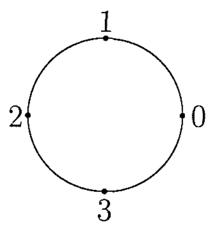


Figure 6.2: The 4PSK constellation points.

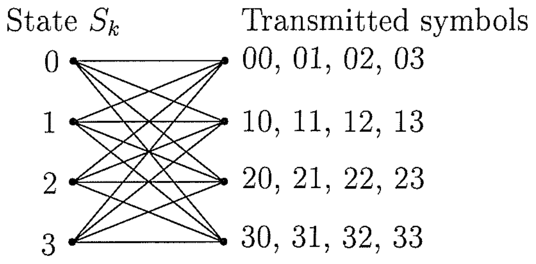


Figure 6.3: The 4-state, 4PSK space-time trellis code.

corresponding state diagram of the 4-state 4PSK space-time trellis code [70] is shown in Figure 6.3. In Figure 6.3, we can see that for each current state there are four possible trellis transitions to the states $0, 1, 2$ and 3, which correspond to the legitimate input symbols of $0(d_{k,1} = 0, d_{k,2} = 0), 1(d_{k,1} = 1, d_{k,2} = 0), 2(d_{k,1} = 0, d_{k,2} = 1)$ and $3(d_{k,1} = 1, d_{k,2} = 1)$, respectively. Correspondingly, there are four sets of possible transmitted symbols associated with the four trellis transitions, shown at right of the state diagram. Each trellis transition is associated with two transmitted symbols, namely with x_1 and x_2, which are transmitted by the antennas Tx 1 and Tx 2, respectively. In Figure 6.4, we have highlighted the trellis transitions from state zero $S_k = 0$ to various states. The associated input symbols and the transmitted symbols of each trellis transitions are shown on top of each trellis transition. If

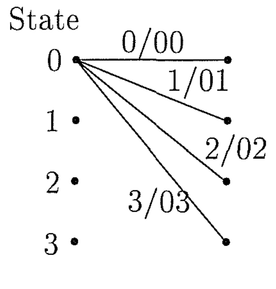


Figure 6.4: The trellis transitions from state $S_k = 0$ to various states.

the input symbol is 0, then the symbol $x_1 = 0$ will be sent by the transmit antenna *Tx* 1, and symbol $x_2 = 0$ by the transmit antenna *Tx* 2 as seen in Figure 6.4 or Figure 6.3. The next state remains $S_{k+1} = 0$. However, if the input symbol is 2 associated with $d_{k,1} = 0$, $d_{k,2} = 1$ in Table 6.1 then, the trellis traverses from state $S_k = 0$ to state $S_{k+1} = 2$ and the symbols $x_1 = 0$ and $x_2 = 2$ are transmitted over the antennas *Tx* 1 and *Tx* 2, respectively. Again, the encoder is required to be in the zero state both at the beginning and at the end of the encoding process.

6.2.1.1 The 4-State, 4PSK Space-Time Trellis Decoder

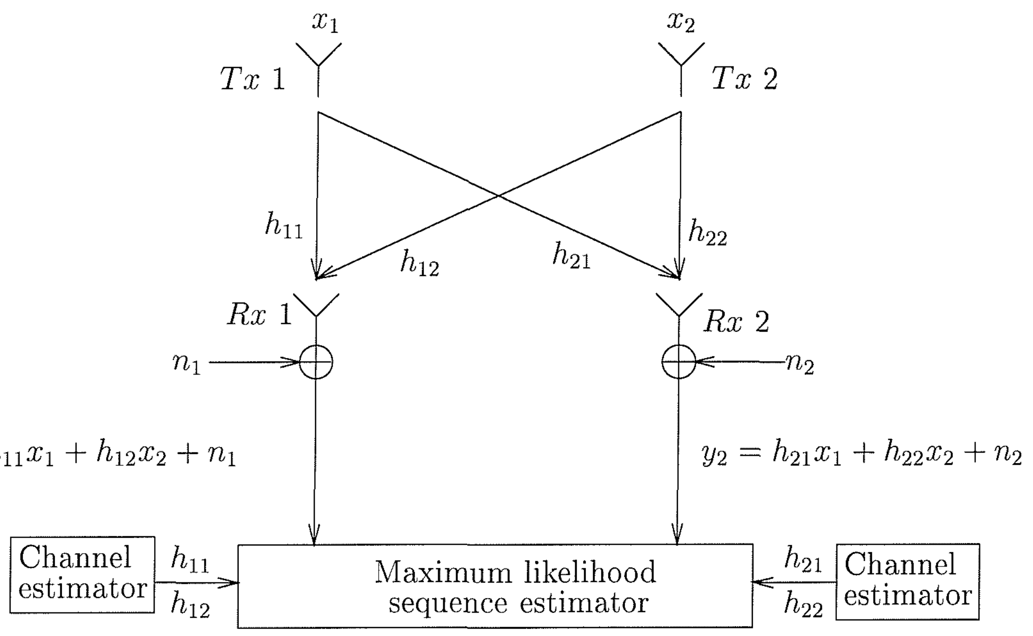


Figure 6.5: Baseband representation of the 4-state, 4PSK space-time trellis code using two receivers.

In Figure 6.5 we show the baseband representation of the 4-state, 4PSK space-time trellis code using two receivers. At any transmission instant, we have symbols x_1 and x_2 transmitted by the antennas *Tx* 1 and *Tx* 2, respectively. At the receivers *Rx* 1 and *Rx* 2, we would have:

$$y_1 = h_{11}x_1 + h_{12}x_2 + n_1 \quad (6.3)$$

$$y_2 = h_{21}x_1 + h_{22}x_2 + n_2 \ , \quad (6.4)$$

where h_{11}, h_{12}, h_{21} and h_{22} represent the corresponding complex time-domain channel transfer factors. Aided by the channel estimator, the Viterbi Algorithm based maximum likelihood sequence estimator [70] first finds the branch metric associated with every transition in the decoding trellis diagram, which is identical to the state diagram shown in Figure 6.3. For each trellis transition, we have two estimated transmit symbols, namely $\tilde{x}_1$ and $\tilde{x}_2$, for which the branch metric BM is given by:

$$\begin{aligned} BM &= |y_1 - h_{11}\tilde{x}_1 - h_{12}\tilde{x}_2 + y_2 - h_{21}\tilde{x}_1 - h_{22}\tilde{x}_2|^2 \\ &= \sum_{i=1}^{2} |y_i - h_{i1}\tilde{x}_1 - h_{i2}\tilde{x}_2|^2 \\ &= \sum_{i=1}^{2} \left| y_i - \sum_{j=1}^{2} h_{ij}\tilde{x}_j \right|^2 . \end{aligned} \tag{6.5}$$

We can however generalise Equation 6.5 to p transmitters and q receivers, as follows:

$$BM = \sum_{i=1}^{p} \left| y_i - \sum_{j=1}^{q} h_{ij}\tilde{x}_j \right|^2 . \tag{6.6}$$

When all the transmitted symbols were received and the branch metric of each legitimate transition was calculated, the maximum likelihood sequence estimator invokes the Viterbi Algorithm (VA) in order to find the maximum likelihood path associated with the best accumulated metric.

6.2.2 Other Space-Time Trellis Codes

In Section 6.2.1, we have shown the encoding and decoding process of the simple 4-state, 4PSK space-time trellis code. More sophisticated 4PSK space-time trellis codes were designed by increasing the number of trellis states [70], which are reproduced in Figures 6.6 to 6.8. With an increasing number of trellis states the number of tailing symbols required for terminating the trellis at the end of a transmitted frame is also increased. Two zero-symbols are needed to force the trellis back to state zero for the space-time trellis codes shown in Figures 6.6 and 6.7. By contrast, three zero-symbols are required for the space-time trellis code shown in Figure 6.8.

Space-time trellis codes designed for the higher-order modulation scheme of 8PSK were also proposed in [70]. In Figure 6.9, we showed the constellation points employed in [70]. The proposed 8-state, 16-state and 32-state 8PSK space-time trellis codes were reproduced from [70] in Figures 6.10, 6.11 and 6.12, respectively. One zero-symbol is required to terminate the 8-state, 8PSK space-time trellis code, whereas two zero-symbols are needed for both the 16-state and 32-state 8PSK space-time trellis codes.

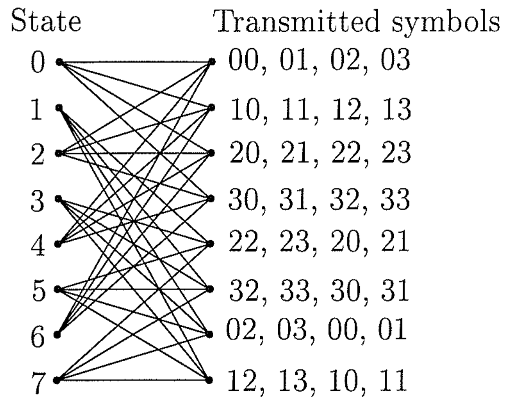


Figure 6.6: The 8-state, 4PSK space-time trellis code.

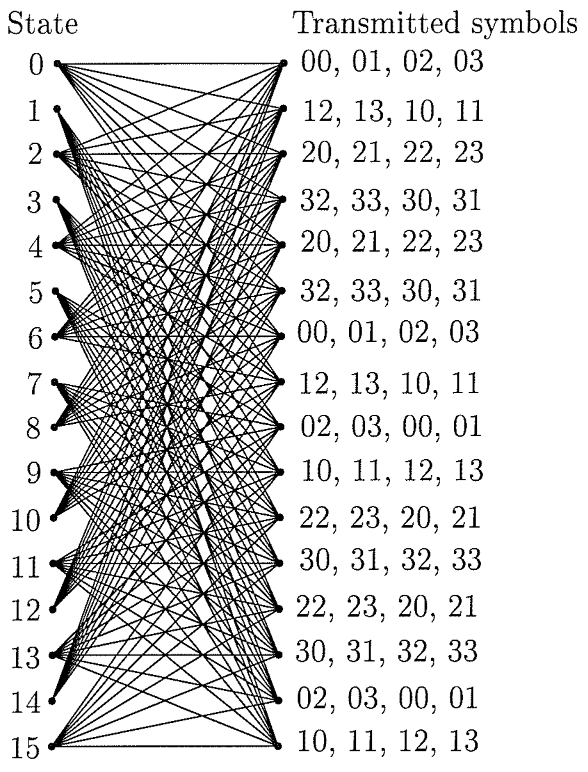


Figure 6.7: The 16-state, 4PSK space-time trellis code.

6.3 Space-Time Coded Transmission Over Wideband Channels

In Section 6.2, we have detailed the concept of space-time trellis codes. Let us now elaborate further by investigating the performance of space-time codes over dispersive wideband fading channels. As mentioned in Section 6.1, Bauch's approach [74,75] of using turbo equalisation for mitigating the ISI exhibits a considerable complexity. Hence we argued that using space-time coded OFDM constitutes a more favourable approach to transmission over dispersive wireless channels, since the associated decoding complexity is significantly lower. Therefore, in this chapter OFDM is employed for mitigating the effects of dispersive channels.

It is widely recognised that space-time trellis codes [70] perform well at the cost of high complexity. However, Alamouti's $\mathbf{G}_2$ space-time block code [71] could be invoked instead of space-time trellis codes. The space-time block code $\mathbf{G}_2$ is appealing in terms of its simplicity, although there is a slight loss in performance. Therefore, we concatenate the space-time block code $\mathbf{G}_2$ with Turbo Convolutional (TC) codes in order to improve the performance of the system. The family of TC codes was favoured, because it was shown in Section 5.5.3 of Chapter 5 and in [108,183] that TC codes achieve an enormous coding gain

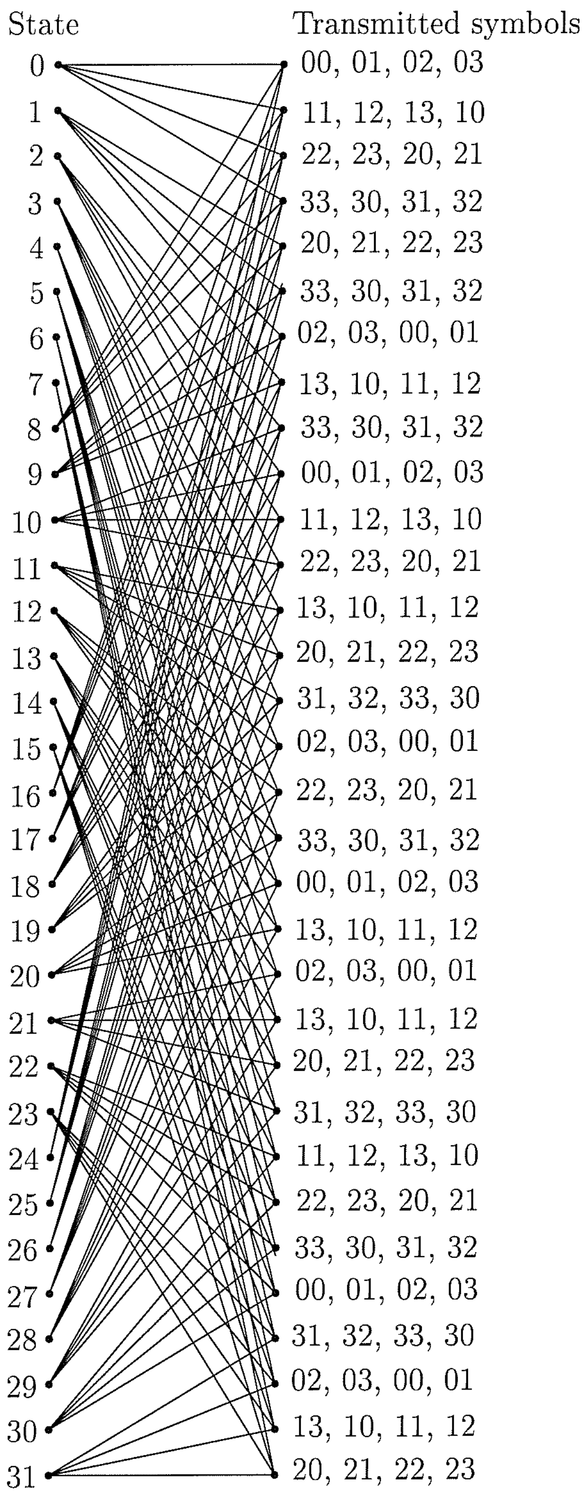


Figure 6.8: The 32-State, 4PSK space-time trellis code.

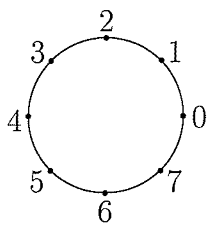


Figure 6.9: The 8PSK constellation points.

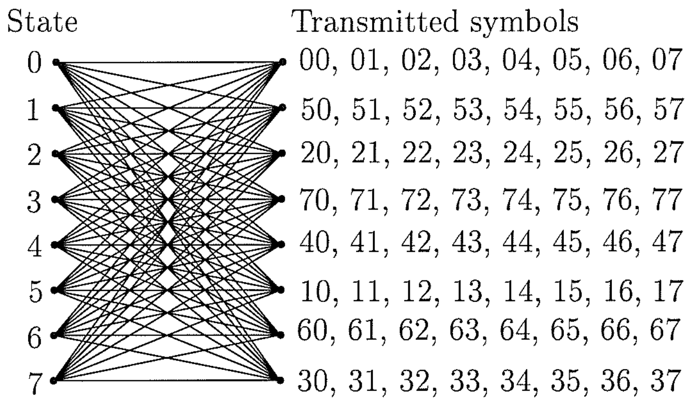


Figure 6.10: The 8-state, 8PSK space-time trellis code.

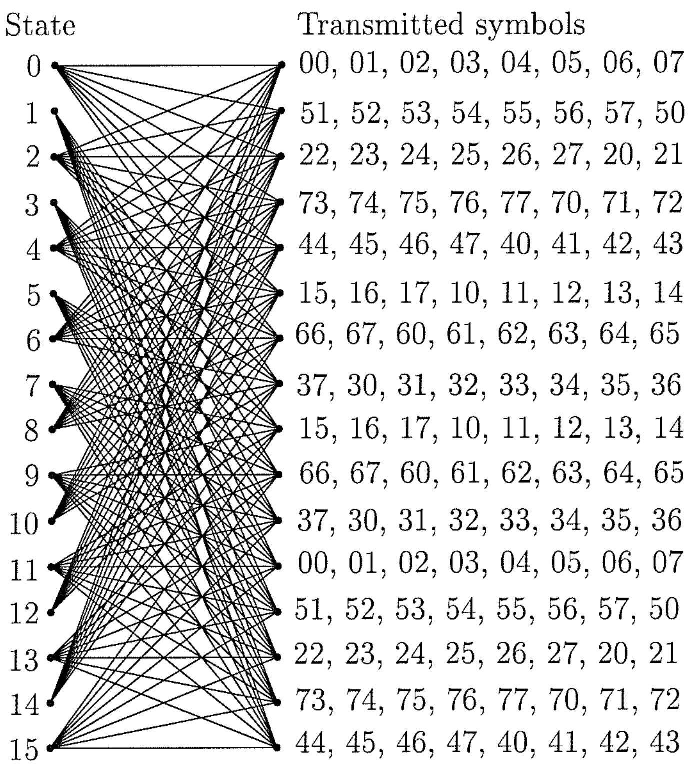


Figure 6.11: The 16-State, 8PSK space-time trellis code.

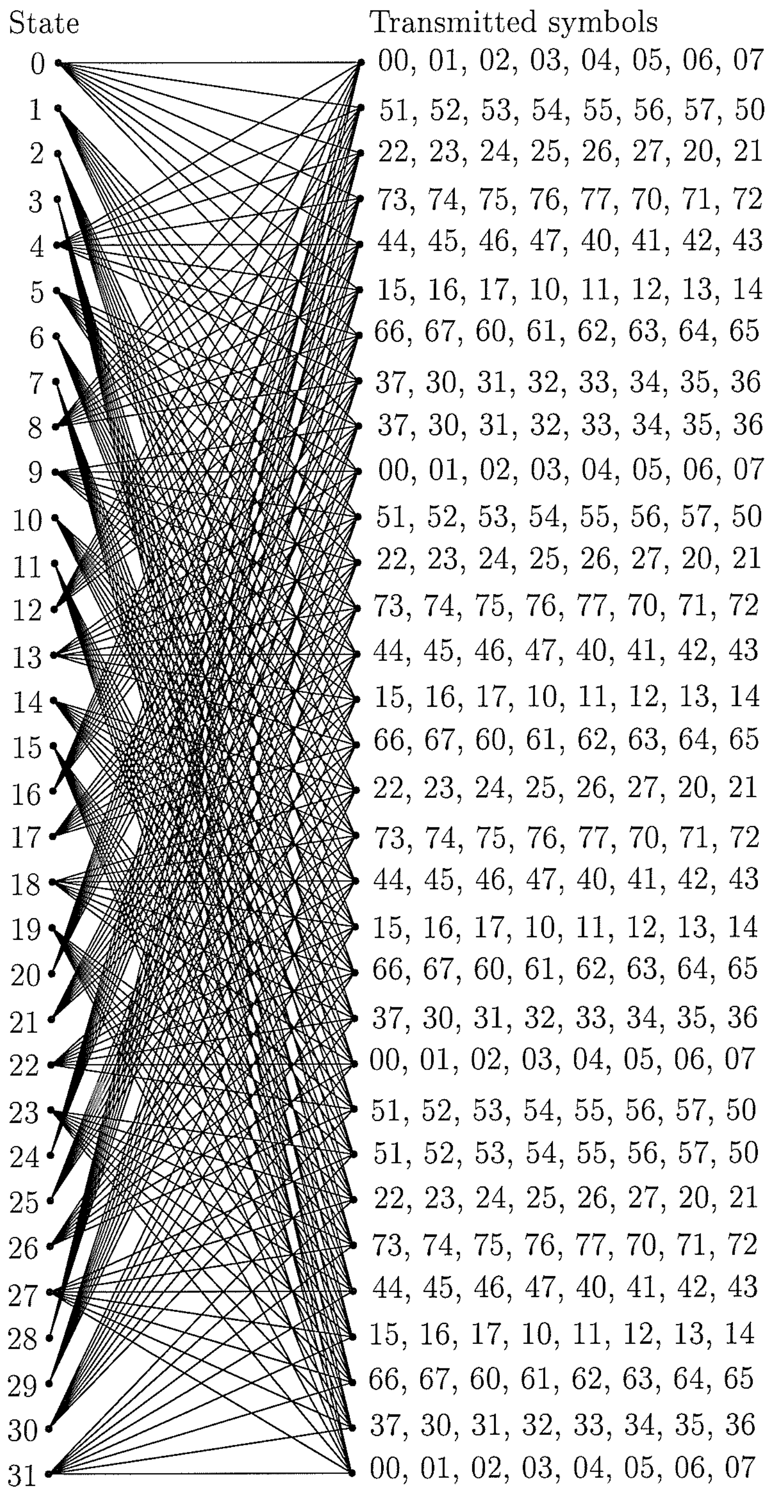


Figure 6.12: The 32-State, 8PSK space-time trellis code.

at a moderate complexity, when compared to convolutional codes, turbo BCH codes, trellis coded modulation and turbo trellis coded modulation. The performance of concatenated space-time block codes and TC codes will then be compared to that of space-time trellis codes. Conventionally, Reed-Solomon (RS) codes have been employed in conjunction with the space-time trellis codes [76, 78, 79] for improving the performance of the system. In our forthcoming discussion, we will concentrate on comparing the performance of space-time block and trellis codes in conjunction with various channel coders.

6.3.1 System Overview

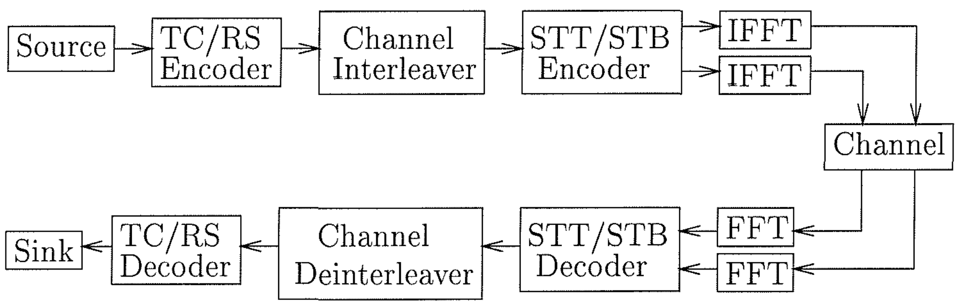


Figure 6.13: System overview.

Figure 6.13 shows the schematic of our system. At the transmitter, the information source generates random information data bits. The information bits are then encoded by TC codes, RS codes or left uncoded. The coded or uncoded bits are then channel interleaved, as shown in Figure 6.13. The output bits of the channel interleaver are then passed to the Space-Time Trellis (STT) or Space-Time Block (STB) encoder. We will investigate all the previously mentioned space-time trellis codes proposed in [70], where the associated state diagrams are shown in Figures 6.3, 6.6, 6.7, 6.10, 6.11 and 6.12. The modulation schemes employed are 4PSK as well as 8PSK and the corresponding trellis diagrams were shown in Figures 6.2 and 6.9, respectively. On the other hand, from the family of space-time block codes only Alamouti's $\mathbf{G}_2$ code is employed in our system, since we have shown in Figure 5.11 and in [183] that the best performance is achieved by concatenating the space-time block code $\mathbf{G}_2$ with TC codes. For convenience, the transmission matrix of the space-time block code $\mathbf{G}_2$ is reproduced here as follows:

$$\mathbf{G}_2 = \begin{pmatrix} x_1 & x_2 \\ -\bar{x}_2 & \bar{x}_1 \end{pmatrix} . \tag{6.7}$$

The reader is referred to Chapter 5 for an indepth discussion on space-time block codes. Different modulation schemes could be employed [49], such as Binary Phase Shift Keying

(BPSK), Quadrature Phase Shift Keying (QPSK), 16-level Quadrature Amplitude Modulation (16QAM) and 64-level Quadrature Amplitude Modulation (64QAM). Gray-mapping of the bits to symbols was applied and this resulted in different protection classes in higher-order modulation schemes [153]. The mapping of the data bits and parity bits of the TC encoder was chosen such that it yielded the best achievable performance along with the application of the random separation channel interleaver [108] seen in Figure 5.14. The output of the space-time encoder was then OFDM [153] modulated and transmitted by the corresponding antenna. The number of transmit antennas was fixed to two, while the number of receive antennas constituted a design parameter. Dispersive wideband channels were used and the associated channels' profiles will be discussed later.

At the receiver the signal of each receive antenna is OFDM demodulated. The demodulated signals of the receiver antennas are then fed to the space-time trellis or space-time block decoder. The space-time decoders apply the MAP [11] or Log-MAP [52,171] decoding algorithms for providing soft outputs for the channel decoders. If no channel codecs are employed in the system, the space-time decoders apply the VA [9,70], which gives similar performance to the MAP decoder at a lower complexity. The decoded bits are finally passed to the sink for the calculation of the Bit Error Rate (BER) or Frame Error Rate (FER).

6.3.2 Space-Time and Channel Codec Parameters

In Figure 6.13, we have given an overview of the proposed system. In this section, we present the parameters of the space-time codes and the channel codecs employed in the proposed system. We will employ the set of various space-time trellis codes shown in Figures 6.3, 6.6, 6.7, 6.8, 6.10, 6.11 and 6.12. The associated space-time trellis coding parameters are summarised in Table 6.2. On the other hand, from the family of space-time block codes

Modulation scheme	BPS	Decoding algorithm	No. of states	No. of transmitters	No. of termination symbols
4PSK	2	VA	4	2	1
			8	2	2
			16	2	2
			32	2	3
8PSK	3	VA	8	2	1
			16	2	2
			32	2	2

Table 6.2: Parameters of the space-time trellis codes shown in Figures 6.3, 6.6, 6.7, 6.8, 6.10, 6.11 and 6.12

only Alamouti's $\mathbf{G}_2$ code is employed, since we have shown in Section 5.5.3 of the previous chapter and in [183] that the best performance in the set of investigated schemes was yielded by concatenating the space-time block code $\mathbf{G}_2$ with TC codes. The transmission matrix of the code is shown in Equation 6.7, while the number of transmitters used by the space-time block code $\mathbf{G}_2$ is two, which is identical to the number of transmitters in the space-time trellis codes shown in Table 6.2.

Let us now briefly consider the TC channel codes used. The reader is referred to Chapters 3 and 5 for further information on the Log-MAP decoding algorithm and for a brief introduction to various TC codes, respectively. In this chapter, we will concentrate on using the simple half-rate $TC(2,1,3)$ code. Its associated parameters are shown in Table 6.3. As seen in Table 6.4, different modulation schemes are employed in conjunction

Code	Octal generator polynomial	No. of states	Decoding algorithm	Puncturing pattern	No. of iterations
$TC(2,1,3)$	7,5	4	Log-MAP	10,01	8

Table 6.3: The associated parameters of the $TC(2,1,3)$ code.

with the concatenated space-time block code $\mathbf{G}_2$ and the TC(2,1,3) code. Since the half-rate TC(2,1,3) code is employed, higher-order modulation schemes such as 16QAM and 64QAM were chosen, so that the throughput of the system remained the same as that of the system employing the space-time trellis codes without channel coding. It is widely recognised that the performance of TC codes improves upon increasing the turbo interleaver size and near-optimum performance can be achieved using large interleaver sizes exceeding 10,000 bits. However, this performance gain is achieved at the cost of high latency, which is impractical for a delay-sensitive real-time system. On the other hand, space-time trellis codes offer impressive coding gains [70] at low latency. The decoding of the space-time trellis codes is carried out on a transmission burst-by-burst basis. In order to make a fair comparison between the systems investigated, the turbo interleaver size was chosen such that all the coded bits were hosted by one transmission burst. This enables burst-by-burst turbo decoding at the receiver. Since we employ an OFDM modem, latency may also be imposed by a high number of subcarriers in an OFDM symbol. Therefore, the turbo interleaver size was increased, as the number of sub-carriers increased in our investigations. In Table 6.4, we summarised the modulation schemes and interleaver sizes used for different number of OFDM subcarriers in the proposed system. The random separation based channel interleaver of Figure 5.14 was used. The mapping of the data bits and parity bits into different

Code	Code Rate R	Modulation Mode	BPS	Random turbo interleaver depth	Random separation interleaver depth
		128 carriers			
TC$(2,1,3)$	0.50	16QAM	2	256	512
		64QAM	3	384	768
		512 carriers			
		QPSK	1	512	1024
		16QAM	2	1024	2048

Table 6.4: The simulation parameters associated with the TC$(2,1,3)$ code.

protection classes of the higher-order modulation scheme was carried out such that the best possible performance was attained. This issue was addressed in Section 5.5.2.

Code	Galois Field	Rate	Correctable symbol errors
RS(105,51)	2^{10}	0.49	27
RS(153,102)	2^{10}	0.67	25

Table 6.5: The coding parameters of the Reed-Solomon codes employed.

Reed-Solomon codes were employed in conjunction with the space-time trellis codes. Hard decision decoding was utilised and the coding parameters of the Reed-Solomon codes employed are summarised in Table 6.5.

6.3.3 Complexity Issues

In this section, we will address the implementational complexity issues of the proposed system. We will however focus mainly on the relative complexity of the proposed systems, rather than attempting to quantify their exact complexity. In order to simplify our comparative study, several assumptions were stipulated. In our simplified approach, the estimated complexity of the system is deemed to depend only on that of the space-time trellis decoder and turbo decoder. In other words, the complexity associated with the modulator, demodulator, space-time block encoder and decoder as well as that of the space-time trellis encoder and turbo encoder are assumed to be insignificant compared to the complexity of space-time trellis decoder and turbo decoder.

In Section 5.4.3, we have detailed our complexity estimates for the TC decoder and the reader is referred to this section for further details. The estimated complexity of the TC decoder is assumed to depend purely on the number of trellis transitions per information data bit and this simple estimated complexity measure was also used in Section 5.4.3 as the basis of our comparisons. Here, we adopt the same approach and evaluate the estimated complexity of the space-time trellis decoder on the basis of the number of trellis transitions per information data bit.

In Figures 6.3, 6.6, 6.7, 6.8, 6.10, 6.11 and 6.12, we have shown the state diagrams of the 4PSK and 8PSK space-time trellis codes. From these state diagrams, we can see that the number of trellis transitions leaving each state is equivalent to 2^{BPS}, where BPS denotes the number of transmitted bits per modulation symbol. Since the number of information bits is equal to BPS, we can approximate the complexity of the space-time trellis decoder as:

$$\begin{aligned} comp\{\text{STT}\} &= \frac{2^{BPS} \times \text{No. of States}}{BPS} \\ &= 2^{BPS-1} \times \text{No. of States}\ . \end{aligned} \tag{6.8}$$

Applying Equation 6.8 and assuming that the Viterbi decoding algorithm was employed, we tabulated the approximated complexities of the space-time trellis decoder in Table 6.6.

Modulation scheme	BPS	No. of states	Complexity
4PSK	2	4	8
		8	16
		16	32
		32	64
8PSK	3	8	21.33
		16	42.67
		32	85.33

Table 6.6: Estimated complexity of the space-time trellis decoders shown in Figures 6.3, 6.6, 6.7, 6.8, 6.10, 6.11 and 6.12

6.4 Simulation Results

In this section, we will present our simulation results characterising the proposed OFDM-based system. As mentioned earlier, we will investigate the proposed system over dispersive wideband Rayleigh fading channels. We will commence our investigations using a simple two-ray channel impulse response (CIR) having equal tap weights, followed by a more realistic Wireless Asynchronous Transfer Mode (WATM) channel [153]. The CIR of the two-ray model is shown in Figure 6.14. From the figure we can see that the reflected path

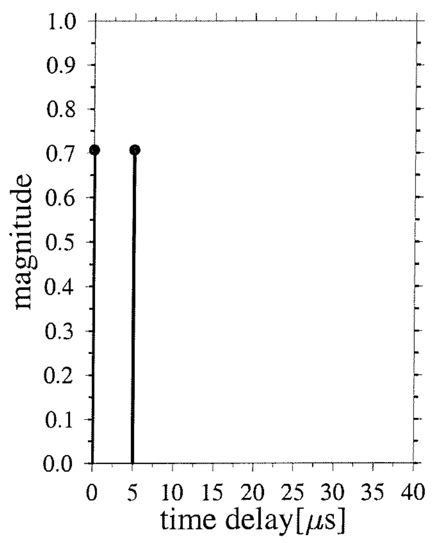


Figure 6.14: Two-ray channel impulse response having equal amplitudes.

has the same amplitude as the Line Of Sight (LOS) path, although arriving $5\mu s$ later. However, in our simulations we also present results over two-ray channels separated by various delay spreads, up to $40\mu s$. Jakes' model [184] was adapted for modelling the fading channels. In Figure 6.15, we portray the 128-subcarrier OFDM symbol employed, having a guard period of $40\mu s$. The guard period of $40\mu s$ or cyclic extension of 32 samples was employed to overcome the inter-OFDM symbol interference due to the channel's memory.

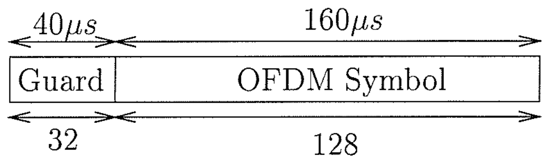


Figure 6.15: Stylised plot of 128-subcarrier OFDM time-domain signal using a cyclic extension of 32 samples.

In order to obtain our simulation results, several assumptions were stipulated:

- The average signal power received from each transmitter antenna was the same;

- All multipath components undergo independent Rayleigh fading;
- The receiver has a perfect knowledge of the CIR.

We note that the above assumptions are unrealistic, yielding the best-case performance, nonetheless, facilitating the performance comparison of the various techniques under identical circumstances.

6.4.1 Space-Time Coding Comparison – Throughput of 2 BPS

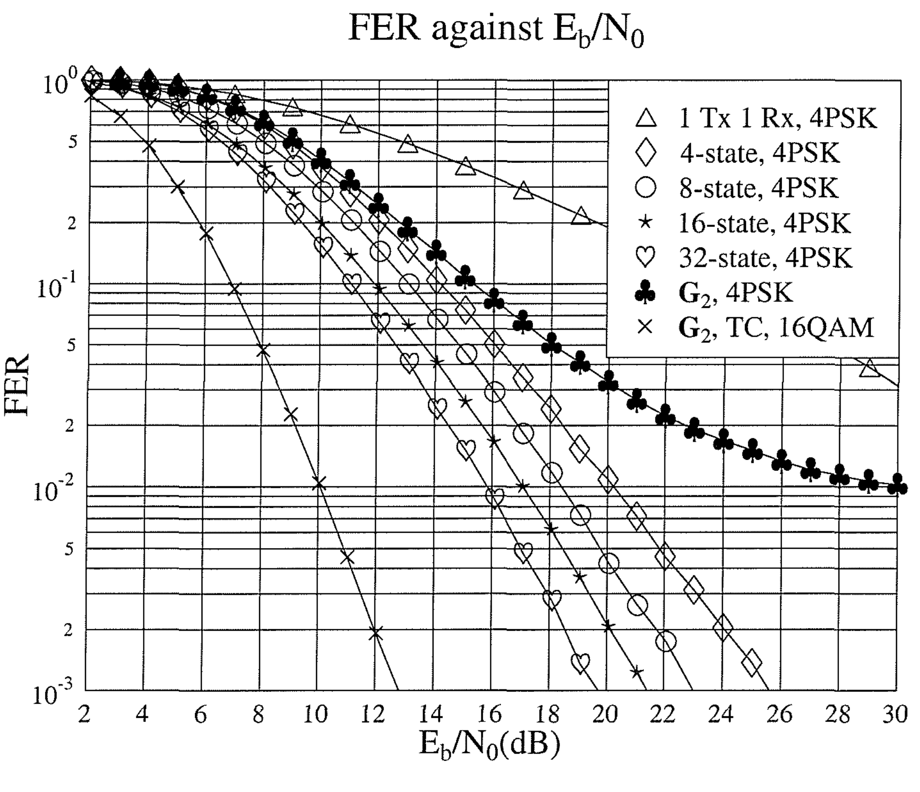


Figure 6.16: **FER** performance comparison between various 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **one receiver** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **2 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

In Figure 6.16, we show our frame error rate (FER) performance comparison between 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver and the 128-subcarrier OFDM modem. The CIR had two equal-power rays separated by a delay spread of $5\mu s$ and the maximum Doppler frequency was 200 Hz. The TC$(2,1,3)$ code is a half-rate code and hence 16QAM was employed, in order to support the same 2 BPS throughput, as the 4PSK space-time trellis codes using no channel codes. We can clearly see that at FER=10^{-3} the performance of the concatenated scheme is at least 7 dB better, than that of the space-time trellis codes.

The performance of the space-time block code $\mathbf{G}_2$ without TC$(2,1,3)$ channel coding is also shown in Figure 6.16. It can be seen in the figure that the space-time block code $\mathbf{G}_2$ does not perform well, exhibiting a residual BER. Moreover, at high E_b/N_0 values, the performance of the single-transmitter, single-receiver system is better than that of the space-time block code $\mathbf{G}_2$. This is because the assumption that the fading is constant over the two consecutive transmission instants is no longer valid in this situation. Here, the two consecutive transmission instants are associated with two adjacent subcarriers in the OFDM symbol and the fading variation is relatively fast in the frequency domain. Therefore, the orthogonality of the space-time code has been destroyed by the frequency-domain variation of the fading envelope. At the receiver, the combiner can no longer separate the two different transmitted signals, namely x_1 and x_2. More explicitly, the signals interfere with each other. The increase in SNR does not improve the performance of the space-time block code $\mathbf{G}_2$, since this also increases the power of the interfering signal. We will address this issue more explicitly in Section 6.4.4. By contrast, the TC$(2,1,3)$ channel codec succeeds in overcoming this problem. However, we will show later in Section 6.4.4 that the concatenated channel coded scheme exhibits the same residual BER problem, if the channel's variation becomes more rapid.

In Figure 6.17, we provide the corresponding BER performance comparison between the 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver and the 128-subcarrier OFDM modem over a channel characterised by two equal-power rays separated by a delay spread of $5\mu s$ and having a maximum Doppler frequency of 200 Hz. Again, we show in the figure that the 2 BPS throughput concatenated $\mathbf{G}_2$/TC(2,1,3) scheme outperforms the 2 BPS space-time trellis codes using no channel coding. At a BER of 10^{-4}, the concatenated channel coded scheme is at least 2 dB superior in SNR terms to the space-time trellis codes using no channel codes. At high E_b/N_0 values, the space-time block code $\mathbf{G}_2$, again exhibits a residual BER. On the other hand, at low E_b/N_0 values the latter outperforms the concatenated $\mathbf{G}_2$/TC(2,1,3) channel coded scheme as well as the space-time trellis codes using no channel coding.

Following the above investigations, the number of receivers was increased to two. In Figure 6.18, we show our FER performance comparison between the various 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using two receivers and the 128-subcarrier OFDM modem. As before, the CIR had two

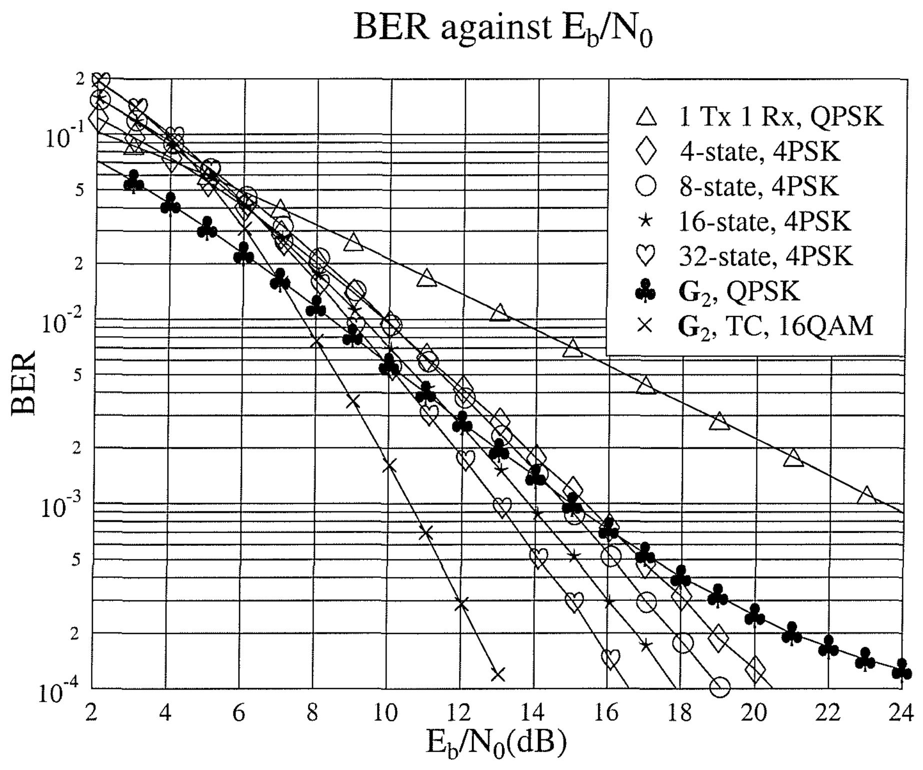


Figure 6.17: **BER** performance comparison between various 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **one receiver** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **2 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

equal-power rays separated by a delay spread of $5\mu s$. Again, we can see that the concatenated $\mathbf{G}_2$/TC(2,1,3) channel coded scheme outperforms the space-time trellis codes using no channel coding. However, the associated difference is lower and at a FER of 10^{-3} the concatenated channel coded scheme is about 4 dB better in E_b/N_0 terms than the space-time trellis codes using no channel codes. On the other hand, by employing two receivers the performance of the space-time block code $\mathbf{G}_2$ improved and the performance flattening effect happens at a lower FER.

In Sections 5.4.3 and 6.3.3, we have derived the complexity estimates of the TC decoders and space-time trellis decoders, respectively. By employing Equations 5.44 and 6.8, we compare the performance of the proposed schemes by considering their approximate complexity. Our performance comparison of the various schemes was carried out on the basis of the coding gain defined as the E_b/N_0 difference, expressed in decibels (dB), at FER$= 10^{-3}$ between the proposed schemes and the uncoded single-transmitter, single-receiver system having the

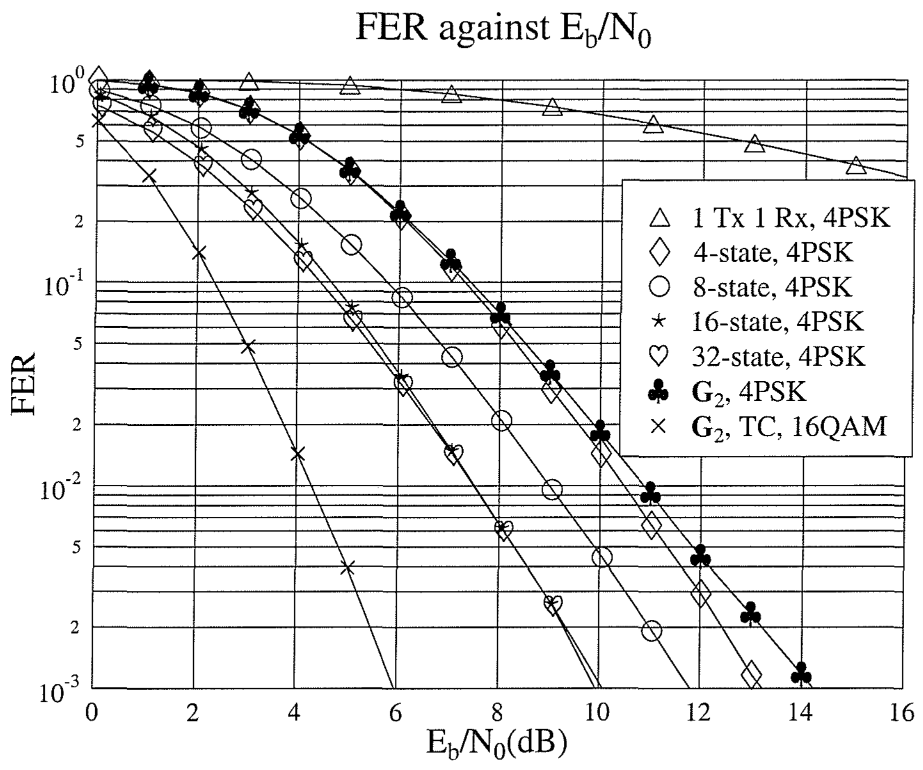


Figure 6.18: **FER** performance comparison between various 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **two receivers** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **2 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

same throughput of 2 BPS. In Figure 6.19, we show our coding gain versus estimated complexity comparison for the various 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one as well as two receivers. The 128-subcarrier OFDM modem transmitted over the channel having a CIR of two equal-power rays separated by a delay spread of $5\mu s$ and a maximum Doppler frequency of 200 Hz. The estimated complexity of the space-time trellis codes was increased by increasing the number of trellis states. By contrast, the estimated complexity of the TC$(2,1,3)$ code was increased by increasing the number of turbo iterations. The coding gain of the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme using one, two, four and eight iterations is shown in Figure 6.19. It can be seen that the concatenated scheme outperforms the space-time trellis codes using no channel coding, even though the number of turbo iterations was only one. Moreover, the improvement in coding gain was obtained, at a estimated complexity comparable to that of the 32-state 4PSK space-time trellis code using no channel coding. From the figure we

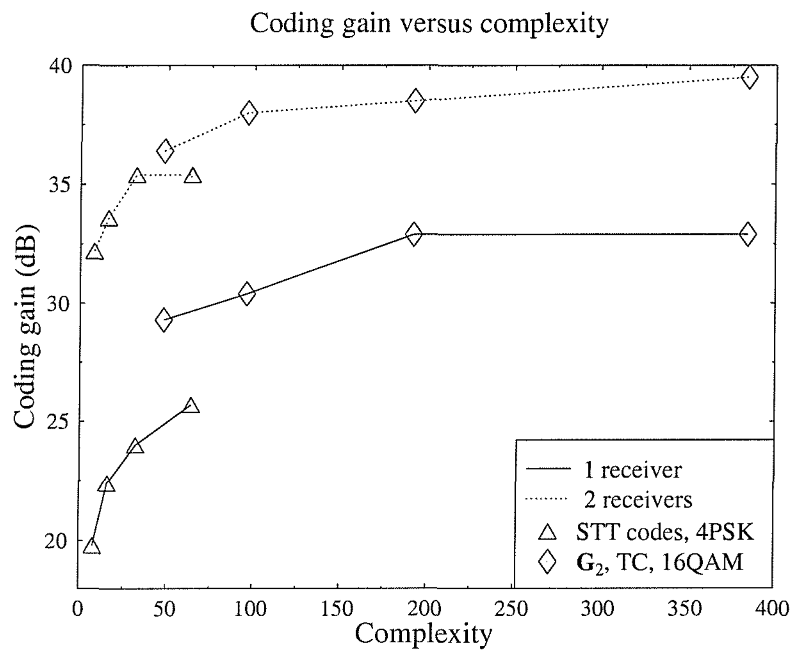


Figure 6.19: Coding gain versus estimated complexity for the various 4PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code **using one as well as two receivers** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **2 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

can also see that the performance gain of the concatenated $\mathbf{G}_2$/TC(2,1,3) channel coded scheme over the space-time trellis codes becomes lower, when the number of receivers is increased to two.

6.4.2 Space-Time Coding Comparison – Throughput of 3 BPS

In Figure 6.20, we show our FER performance comparison between the various 8PSK space-time trellis codes of Table 6.2 and space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver and the 128-subcarrier OFDM modem. The CIR exhibited two equal-power rays separated by a delay spread of $5\mu s$ and a maximum Doppler frequency of 200 Hz. Since the TC(2, 1, 3) scheme is a half-rate code, 64QAM was employed in order to ensure the same 3 BPS throughput, as the 8PSK space-time trellis codes using no channel coding. We can clearly see that at FER=10^{-3} the performance of the concatenated channel coded scheme is at least 7 dB better in terms of the required E_b/N_0 than that of the

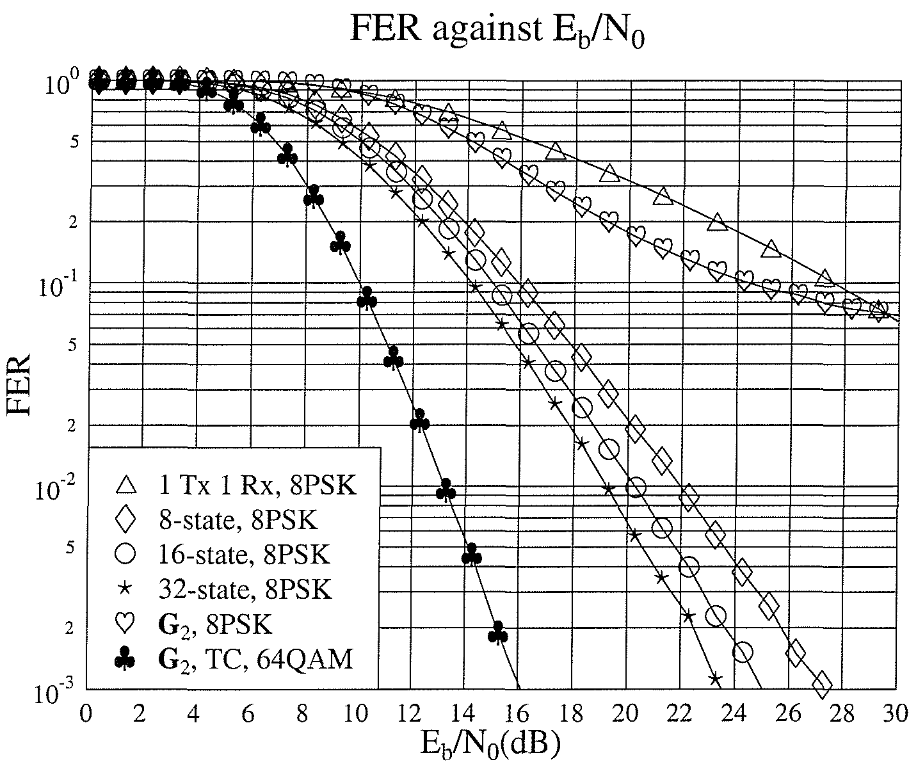


Figure 6.20: **FER** performance comparison between various 8PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **one receiver** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **3 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

space-time trellis codes. The performance of the space-time block code $\mathbf{G}_2$ without the concatenated $\mathrm{TC}(2,1,3)$ code is also shown in the figure. In Table 6.4, we can see that although there is an increase in the turbo interleaver size, due to employing a higher-order modulation scheme, nonetheless, no performance gain is observed for the concatenated $\mathrm{TC}(2,1,3)$-$\mathbf{G}_2$ scheme over the space-time trellis codes using no channel coding. We speculate that this is because the potential gain due to the increased interleaver size has been offset by the vulnerable 64QAM scheme.

We also show in Figure 6.20 that the performance of the 3 BPS 8PSK space-time block code $\mathbf{G}_2$ without the concatenated $\mathrm{TC}(2,1,3)$ scheme is worse, than that of the other proposed schemes. It exhibits the previously noted flattening effect, which becomes more pronounced near FER= 10^{-1}. The same phenomenon was observed near FER= 10^{-2} for the corresponding $\mathbf{G}_2$-coded 4PSK scheme, which has a throughput of 2 BPS.

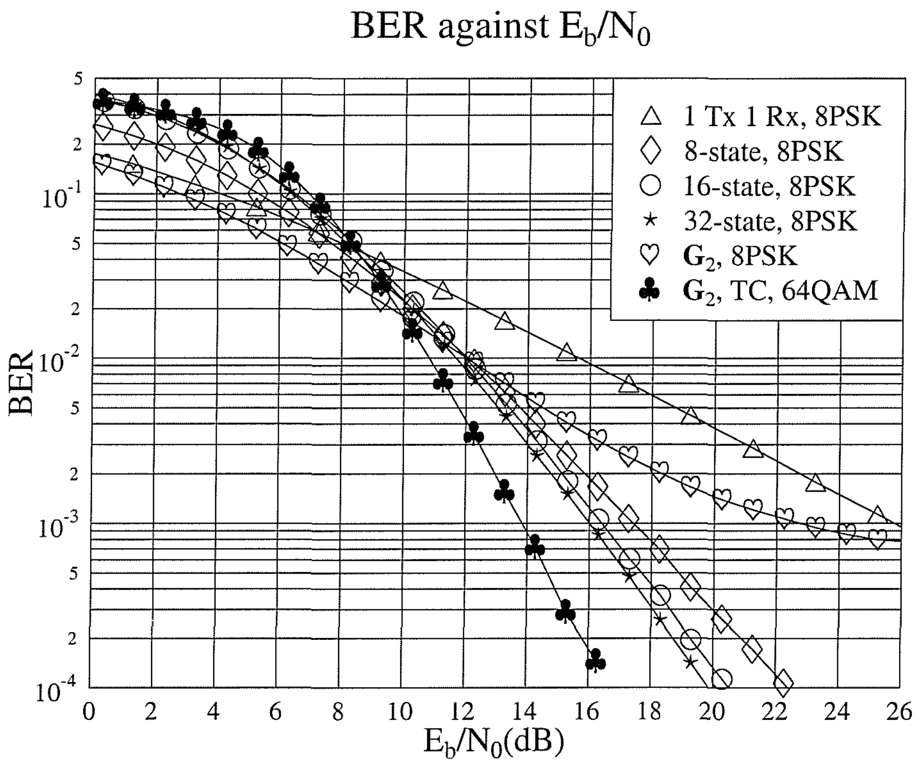


Figure 6.21: **BER** performance comparison between various 8PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **one receiver** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **3 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

In Figure 6.21, we portray our BER performance comparison between the various 8PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) scheme using one receiver and the 128-subcarrier OFDM modem. The CIR exhibited two equal-power rays separated by a delay spread of $5\mu s$ and the maximum Doppler frequency was 200 Hz. Again, we observe in the figure that the concatenated $\mathbf{G}_2$/TC(2,1,3)-coded scheme outperforms the space-time trellis codes using no channel coding. At a BER of 10^{-4}, the concatenated scheme is at least 2 dB better in terms of its required E_b/N_0 value, than the space-time trellis codes. The performance of the space-time block code $\mathbf{G}_2$ without TC(2,1,3) channel coding is also shown in Figure 6.21. As before, at high E_b/N_0 values, the space-time block code $\mathbf{G}_2$ exhibits a flattening effect. On the other hand, at low E_b/N_0 values it outperforms the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme as well as the space-time trellis codes.

In Figure 6.22, we compare the FER performance of the 8PSK space-time trellis codes

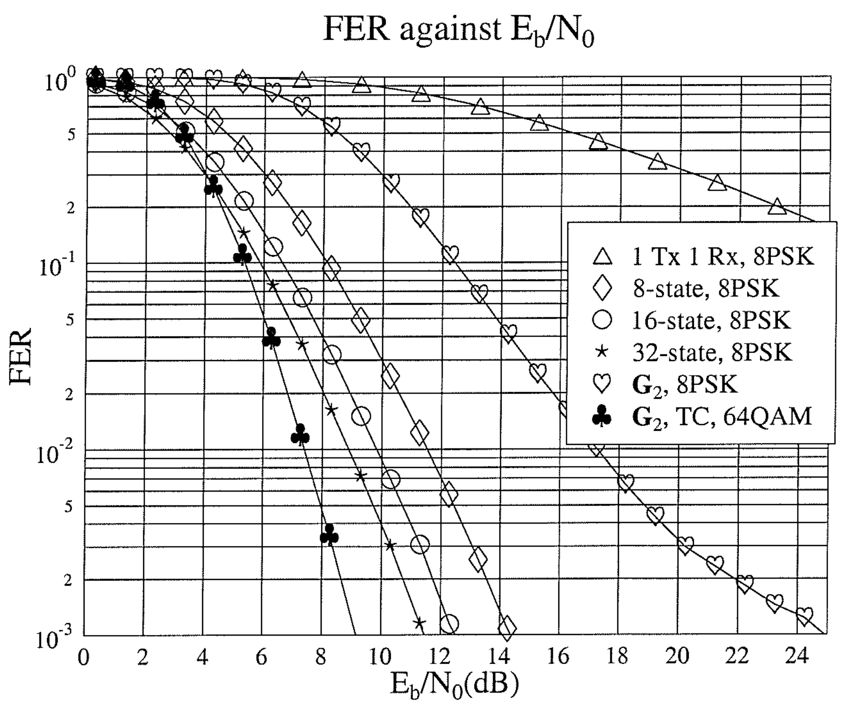


Figure 6.22: **FER** performance comparison between various 8PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **two receivers** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **3 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) channel codec using two receivers and the 128-subcarrier OFDM modem. As before, the CIR has two equal-power rays separated by a delay spread of $5\mu s$ and exhibits maximum Doppler frequency of 200 Hz. Again, with the increase in the number of receivers the performance gap between the concatenated channel coded scheme and the space-time trellis codes using no channel coding becomes smaller. At a FER of 10^{-3} the concatenated channel coded scheme is only about 2 dB better in terms of its required E_b/N_0, than the space-time trellis codes using no channel coding.

With the increase in the number of receivers, the previously observed flattening effect of the space-time block code $\mathbf{G}_2$ has been substantially mitigated, dipping to values below FER= 10^{-3}. However, it can be seen in Figure 6.22 that its performance is about 10 dB worse, than that of the 8-state 8PSK space-time trellis code. In our previous system characterised in Figure 6.18, which had an effective throughput of 2 BPS, the performance of the space-time block code $\mathbf{G}_2$ was only about 1 dB worse in E_b/N_0 terms, than that of the 4-state 4PSK space-time trellis code, when the number of receivers was increased to two. This observation clearly shows that higher-order modulation schemes have a tendency to saturate the channel's capacity and hence result in a poorer performance, than the identical-throughput space-time trellis codes using no channel coding.

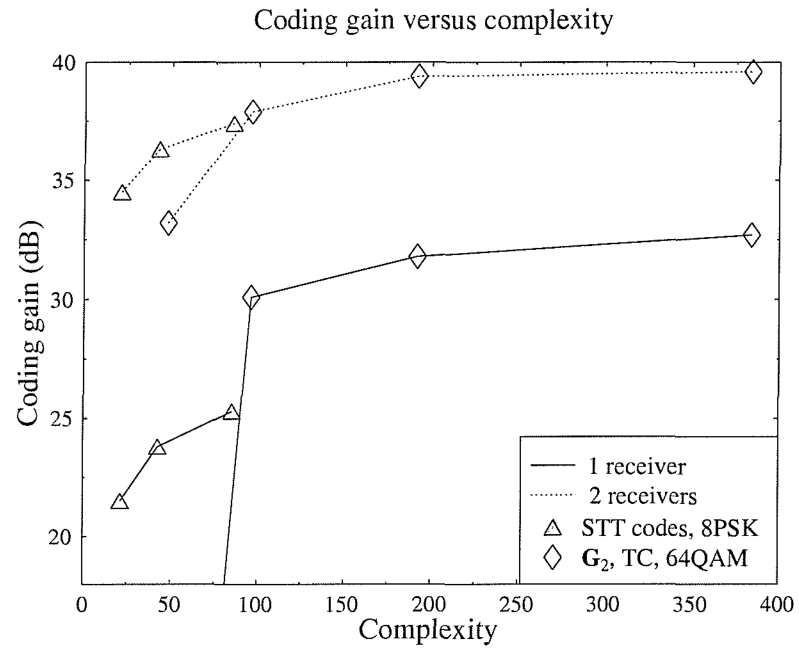


Figure 6.23: Coding gain versus estimated complexity for the various 8PSK space-time trellis codes and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code **using one and two receivers** and the 128-subcarrier OFDM modem over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $5\mu s$. The maximum Doppler frequency was 200 Hz. The effective throughput was **3 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

Similarly to the 2 BPS schemes of Figure 6.19, we compare the performance of the proposed 3 BPS throughput schemes by considering their approximate decoding complexity. The derivation of the estimated complexity has been detailed in Sections 5.4.3 and 6.3.3. As mentioned earlier, the performance comparison of the various schemes was made on the basis of the coding gain defined as the E_b/N_0 difference, expressed in decibels, at a FER$= 10^{-3}$ between the proposed schemes and the uncoded single-transmitter, single-receiver system having a throughput of 3 BPS. In Figure 6.23, we show the associated coding gain versus estimated complexity curves for the 8PSK space-time trellis codes using no channel coding and the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one and two receivers and the 128-subcarrier OFDM modem. For the sake of consistency, the CIR, again, exhibited two equal-power rays separated by a delay spread of $5\mu s$ and a maximum Doppler frequency of 200 Hz. Again, the estimated complexity of the space-time trellis codes was increased by increasing the number of states. On the other hand, the estimated complexity of the $\mathrm{TC}(2,1,3)$ code was increased by increasing the number of iterations. The coding gain of the concatenated channel coded scheme invoking one, two, four and eight iterations is shown in Figure 6.23. Previously in Figure 6.19 we have shown that the concatenated TC(2,1,3)-coded scheme using one iteration outperformed the space-time trellis codes using no channel coding. However, in Figure 6.23 the concatenated scheme does not exhibit the same performance trend. For the case of one receiver, the concatenated scheme using one iteration has a negative coding gain and exhibits a saturation effect. This is again, due to the employment of the high-order 64QAM scheme, which has a preponderance to exceed the channel's capacity. Again, we can also see that the performance gain of the concatenated $\mathbf{G}_2$/TC(2,1,3)-coded scheme over the space-time trellis codes using no channel coding becomes smaller, when the number of receivers is increased to two. Having studied the performance of the proposed schemes over the channel characterised by the two-path, 5μ-dispersion CIR at a fixed Doppler frequency of 200Hz, let us in the next section study the effects of varying the Doppler frequency.

6.4.3 The Effect of Maximum Doppler Frequency

In our further investigations we have generated the FER versus E_b/N_0 curves similar to those in Figure 6.16, when the Doppler frequency was fixed to 5, 10, 20, 50 and 100 Hz. In order to present these results in a compact form, we then extracted the required E_b/N_0 values for maintaining a FER of 10^{-3}. In Figure 6.24, we show the E_b/N_0 crossing point at FER=10^{-3} versus the maximum Doppler frequency for the 32-state 4PSK space-time trellis code using no channel coding and for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver and the 128-subcarrier OFDM modem. As before, the CIR exhibited two equal-power rays separated by a delay spread of $5\mu s$. We conclude from the near-horizontal curves shown in the figure that the maximum Doppler frequency does not significantly affect the performance of the space-time trellis codes and the concatenated scheme. Furthermore, the performance of the concatenated scheme is always better, than that of the space-time trellis codes using no channel coding. Having studied the effects of various Doppler frequencies, let us now consider the impact of varying the delay spread.

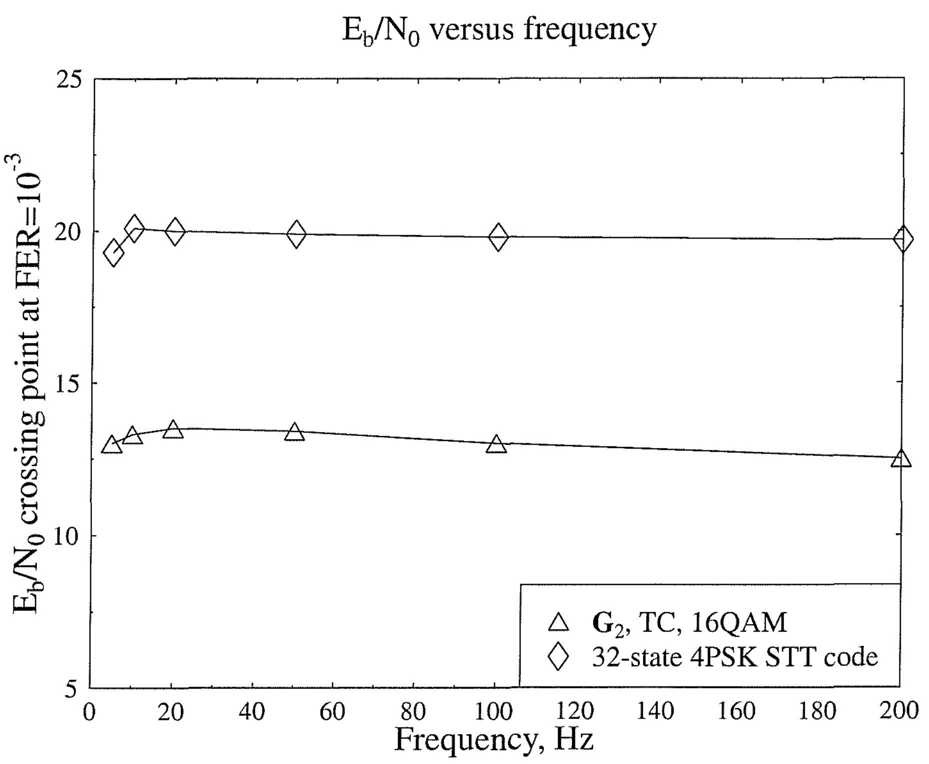


Figure 6.24: The E_b/N_0 value required for maintaining FER=10^{-3} versus the maximum Doppler frequency for the 32-state 4PSK space-time trellis code and for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **one receiver** and the 128-subcarrier OFDM modem. The CIR exhibited two equal-power rays separated by a delay spread of $5\mu s$. The effective throughput was **3 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4.

6.4.4 The Effect of Delay Spreads

In this section, we will study how the variation of the delay spread between the two paths of the channel affects the system performance. By varying the delay spread, the channel's frequency-domain response varies as well. In Figure 6.25, we show the fading amplitude variation of the 128 subcarriers in an OFDM symbol for a delay spread of (a) $5\mu s$, (b) $10\mu s$, (c) $20\mu s$ and (d) $40\mu s$. It can be seen from the figure that the fading amplitudes vary more rapidly, when the delay spread is increased. For the space-time block code $\mathbf{G}_2$ we have shown in Section 5.3.1 that the fading envelopes of the two consecutive transmission instants of antennas Tx 1 and Tx 2 are assumed to be constant. In Figure 6.25 (d), we can see that the variation of the frequency-domain fading amplitudes is so dramatic that we can no longer assume that the fading envelopes are constant for two consecutive transmission instants. The variation of the frequency-domain fading envelope will eventually destroy the

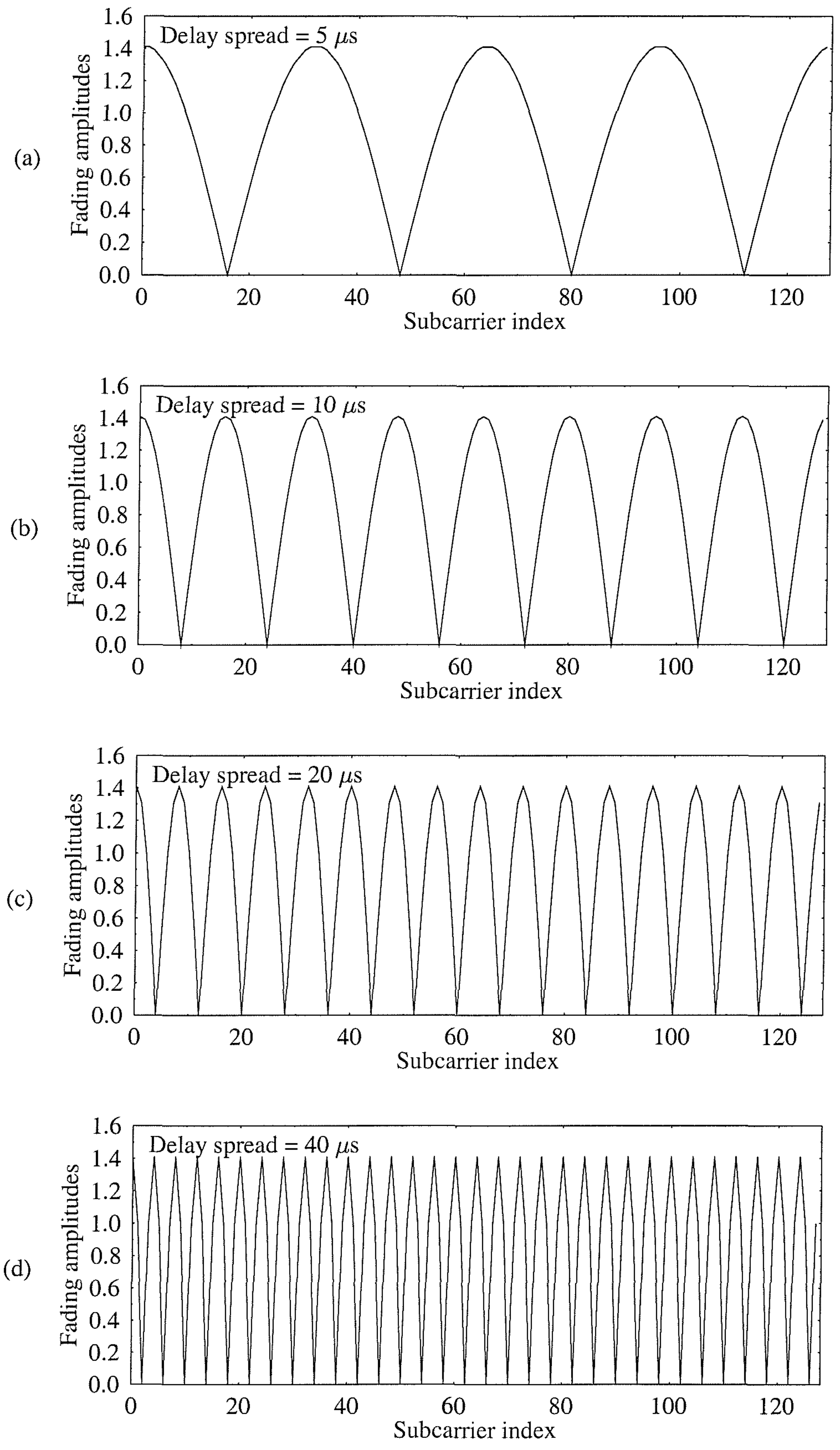


Figure 6.25: Frequency-domain fading amplitudes of the 128 subcarriers in an OFDM symbol for a delay spread of (a) $5\mu s$, (b) $10\mu s$, (d) $20\mu s$ and (c) $40\mu s$.

orthogonality of the space-time block code $\mathbf{G}_2$.

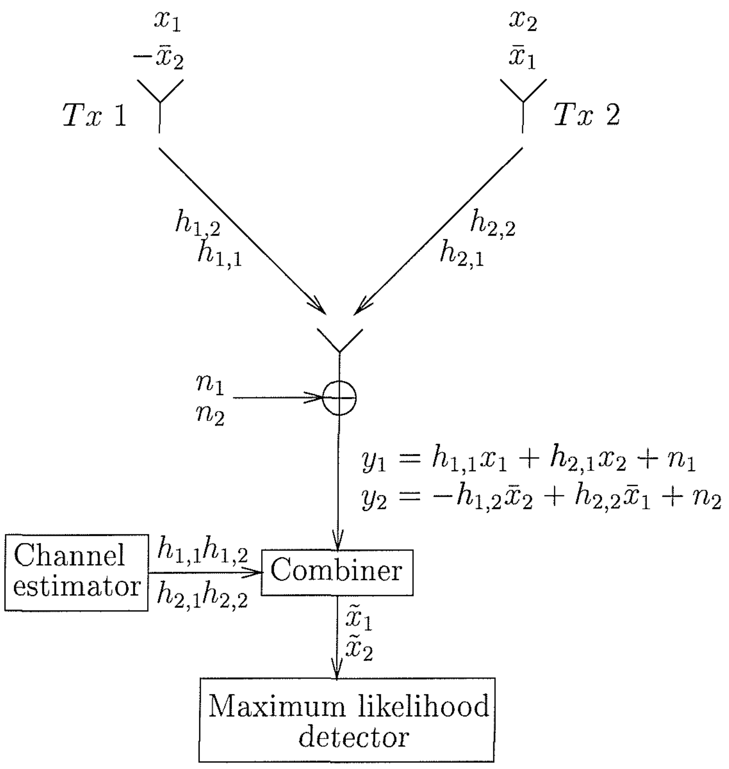


Figure 6.26: Baseband representation of the simple twin-transmitter space-time block code $\mathbf{G}_2$ of Equation 6.7 using one receiver over varying fading conditions.

We reproduce Figure 5.2 in Figure 6.26 with the modification that the two transmission instants are no longer assumed to be associated with the same complex transfer function values. The figure shows the baseband representation of the simple twin-transmitter space-time block code $\mathbf{G}_2$ of Equation 6.7 using one receiver. At the receiver, we have

$$y_1 = h_{1,1}x_1 + h_{2,1}x_2 + n_1 \tag{6.9}$$

$$y_2 = -h_{1,2}\bar{x}_2 + h_{2,2}\bar{x}_1 + n_2 , \tag{6.10}$$

where y_1 is the first received signal and y_2 is the second. Both signals y_1 and y_2 are passed to the combiner in order to extract the signals x_1 and x_2. Aided by the channel estimator, which in this example provides perfect estimation of the diversity channels' frequency-domain transfer functions, the combiner performs simple signal processing in order to separate the signals x_1 and x_2. Specifically, in order to extract the signal x_1, it combines signals y_1 and y_2 as follows:

$$\begin{aligned}\tilde{x}_1 &= \bar{h}_{1,1}y_1 + h_{2,2}\bar{y}_2 \\ &= \bar{h}_{1,1}h_{1,1}x_1 + \bar{h}_{1,1}h_{2,1}x_2 + \bar{h}_{1,1}n_1 - h_{2,2}\bar{h}_{1,2}x_2 + h_{2,2}\bar{h}_{2,2}x_1 + h_{2,2}\bar{n}_2 \\ &= \left(|h_{1,1}|^2 + |h_{2,2}|^2\right)x_1 + \left(\bar{h}_{1,1}h_{2,1} - h_{2,2}\bar{h}_{1,2}\right)x_2 + \bar{h}_{1,1}n_1 + h_{2,2}\bar{n}_2 \ . \qquad (6.11)\end{aligned}$$

Similarly, for signal x_2 the combiner generates:

$$\begin{aligned}\tilde{x}_2 &= \bar{h}_{2,1}y_1 - h_{1,2}\bar{y}_2 \\ &= \bar{h}_{2,1}h_{1,1}x_1 + \bar{h}_{2,1}h_{2,1}x_2 + \bar{h}_{2,1}n_1 + h_{1,2}\bar{h}_{1,1}x_2 - h_{1,2}\bar{h}_{2,2}x_1 - h_{1,2}\bar{n}_2 \\ &= \left(|h_{2,1}|^2 + |h_{1,2}|^2\right)x_2 + \left(\bar{h}_{2,1}h_{1,1} - h_{1,2}\bar{h}_{2,2}\right)x_1 + \bar{h}_{2,1}n_1 - h_{1,2}\bar{n}_2 \ . \qquad (6.12)\end{aligned}$$

In contrast to the prefect cancellation scenario of Equations 5.15 and 5.16, we can see from Equations 6.11 and 6.12 that the signals x_1 and x_2 now interfere with each other. We can no longer cancel the cross-coupling of signals x_2 and x_1 in Equations 6.11 and 6.12, respectively, unless the fading envelopes satisfy the condition of $h_{1,1} = h_{1,2}$ and $h_{2,1} = h_{2,2}$.

At high SNRs the noise power is insignificant compared to the transmitted power of the signals x_1 and x_2. Therefore, we can ignore the noise terms n in Equations 6.11 and 6.12. However, the interference signals' power increases, as we increase the transmission power. Assuming that both the signals x_1 and x_2 have an equivalent signal power, we can then express the signal to interference ratio (SIR) for signal x_1 as:

$$SIR = \frac{|h_{1,1}|^2 + |h_{2,2}|^2}{\bar{h}_{1,1}h_{2,1} - h_{2,2}\bar{h}_{1,2}} \ , \qquad (6.13)$$

and similarly for signal x_2 as:

$$SIR = \frac{|h_{2,1}|^2 + |h_{1,2}|^2}{\bar{h}_{2,1}h_{1,1} - h_{1,2}\bar{h}_{2,2}} \ . \qquad (6.14)$$

In Figure 6.27, we show the FER performance of the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver and the 128-subcarrier 16QAM OFDM modem. The CIR has two equal-power rays separated by various delay spreads and a maximum Doppler frequency of 200 Hz. As we can see in Equations 6.13 and 6.14, we have $SIR \to \infty$, if $h_{1,1} = h_{1,2}$ and $h_{2,1} = h_{2,2}$. On the other hand, we encounter $SIR \to 1$, if $h_{1,1} = \delta h_{1,2}$ and $h_{2,1} = \delta h_{2,2}$, where $\delta \to \infty$. Since the SIR decreases, when the delay spread increases due to the rapidly fluctuating frequency-domain fading envelopes, as shown in Figure 6.25, we can see in Figure 6.27 that the performance of the concatenated scheme degrades, when increasing the delay spread. When the delay spread is more than $15\mu s$, we can see from the figure that the concatenated scheme exhibits the previously observed

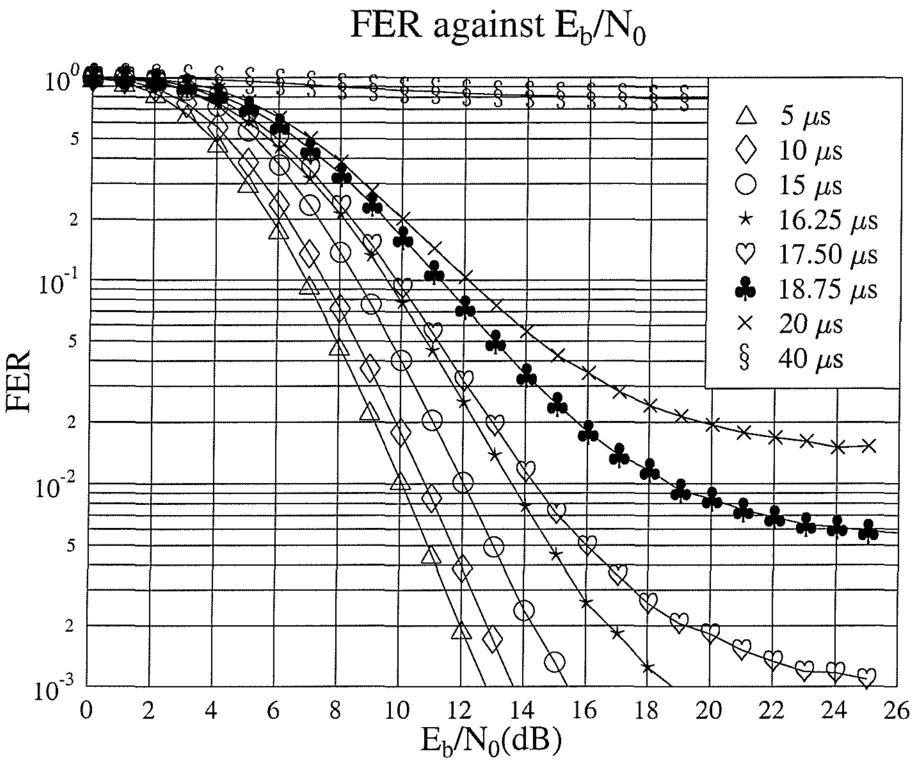


Figure 6.27: **FER** performance of the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver, the 128-subcarrier OFDM modem and 16QAM. The CIR exhibits two equal-power rays separated by various delay spreads and a maximum Doppler frequency of 200 Hz. The coding parameters are shown in Tables 6.2, 6.3 and 6.4.

flattening effect. Furthermore, the error floor of the concatenated scheme becomes higher, as the delay spread is increased.

Similarly to Figure 6.24, where the Doppler frequency was varied, we show in Figure 6.28 the E_b/N_0 value required for maintaining FER=10^{-3} versus the delay spread for the 32-state 4PSK space-time trellis code and for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver and the 128-subcarrier OFDM modem. The CIR exhibited two equal-power rays separated by various delay spreads. The maximum Doppler frequency was 200 Hz. We can see in the figure that the performance of the 32-state 4PSK space-time trellis code does not vary significantly with the delay spread. However, the concatenated TC(2,1,3)-coded scheme suffers severe performance degradation upon increasing the delay spread, as evidenced by the associated error floors shown in Figure 6.27. The SIR associated with the various delay spreads was obtained using computer simulations and the associated SIR values are also shown in Figure 6.28, denoted by the hearts. As we have expected, the calculated SIR decreases with the delay spread. We can see in the figure that the

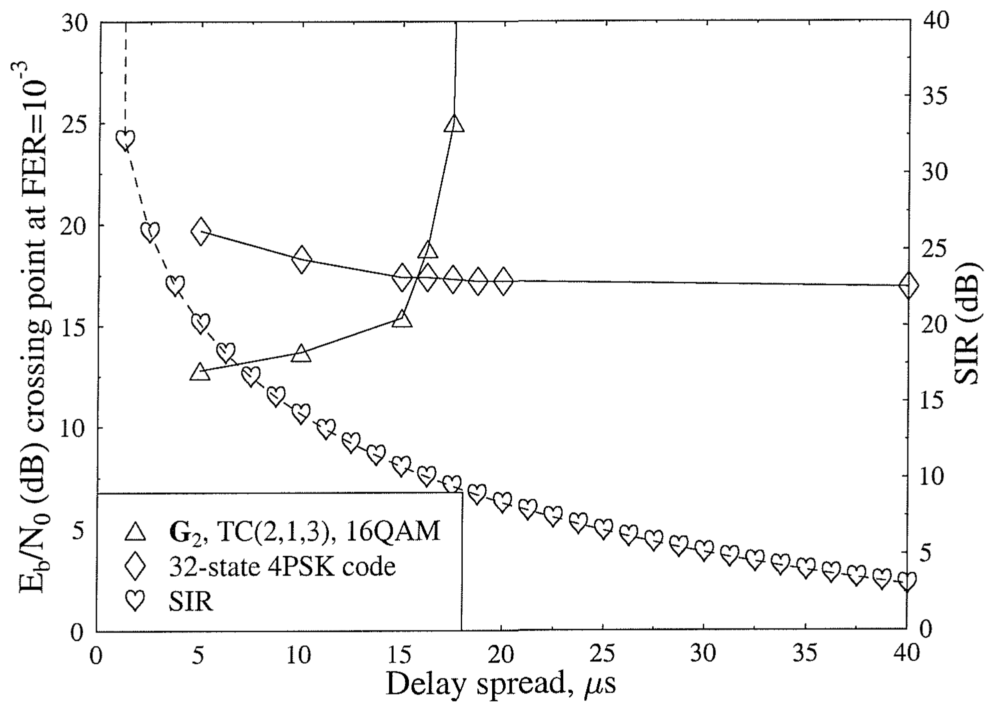


Figure 6.28: The E_b/N_0 values required for maintaining FER=10^{-3} versus delay spreads for the 32-state 4PSK space-time trellis code and for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **one receiver** and the 128-subcarrier OFDM modem. The CIR exhibited two equal-power rays separated by various delay spreads and a maximum Doppler frequency of 200 Hz. The effective throughput was **2 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4. The *SIR* of various delay spreads are shown as well.

performance of the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme suffers severe degradation, when the delay spread is in excess of $15\mu s$, as indicated by the near-vertical curve marked by triangles. If we relate this curve to the SIR curve marked by the hearts, we can see on the right-hand side scale of the figure that the *SIR* is approximately 10 dB. Hence the *SIR* of the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme has to be more than 10 dB, in order for it to outperform the space-time trellis codes using no channel coding.

6.4.5 Delay Non-sensitive System

Previously, we have provided simulation results for a delay-sensitive, OFDM symbol-by-symbol decoded system. More explicitly, the received OFDM symbol had to be demodulated and decoded on a symbol-by-symbol basis, in order to provide decoded bits for example for

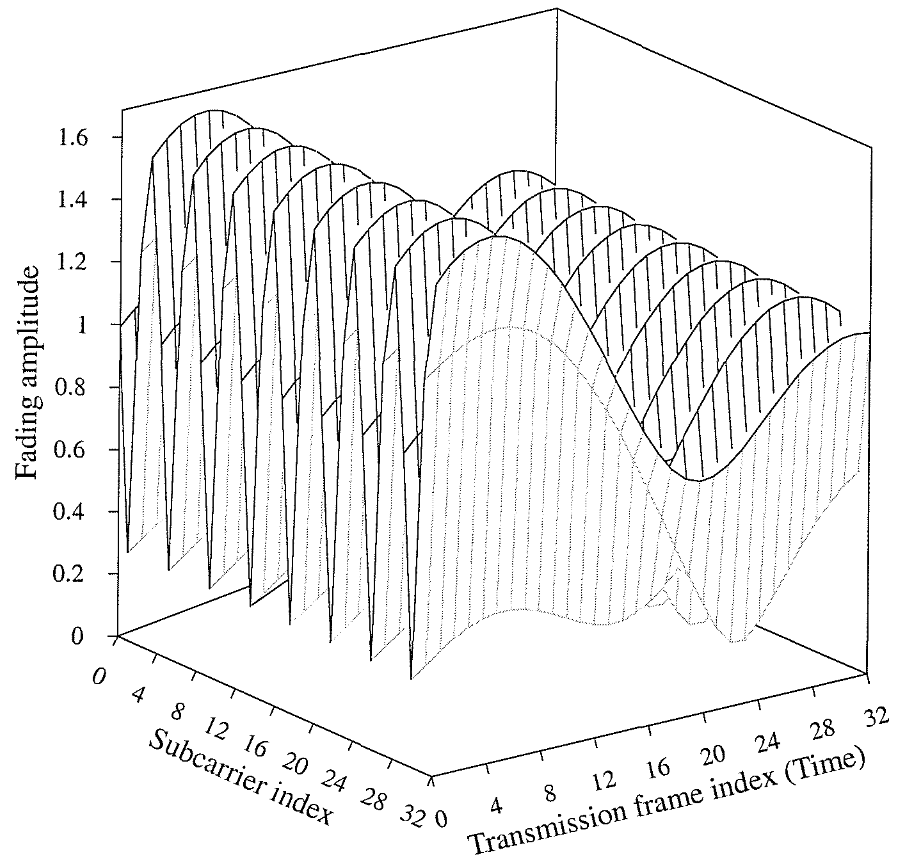


Figure 6.29: The fading amplitude versus time and frequency for various 128-subcarrier OFDM symbols over the two-path channel exhibiting two equal-power rays separated by a delay spread of $40\mu s$ and maximum Doppler frequency of 100 Hz.

a low-delay source decoder. Therefore, the two transmission instants of the space-time block code $\mathbf{G}_2$ had to be in the same OFDM symbol. They were allocated to the adjacent subcarriers in our previous studies. Moreover, we have shown in Figure 6.25 that the variation of the frequency-domain fading amplitudes along the subcarriers becomes more severe, as we increase the delay spread of the two rays. In Figure 6.29 we show both the frequency-domain and time-domain fading amplitudes of the channels' fading amplitudes for a fraction of the subcarriers in the 128-subcarrier OFDM symbols over the previously used two-path channel having two equal-power rays separated by a delay spread of $40\mu s$. The maximum Doppler frequency was set here to 100 Hz. It can be clearly seen from the figure that the fading amplitude variation versus time is slower, than that versus the subcarrier index within the OFDM symbols. This implies that the SIR attained would be higher, if we were to allocate the two transmission instants of the space-time block code

$\mathbf{G}_2$ to the same subcarrier of consecutive OFDM symbols. This increase in *SIR* is achieved by doubling the delay of the system, since in this scenario two consecutive OFDM symbols have to be decoded, before all the received data becomes available.

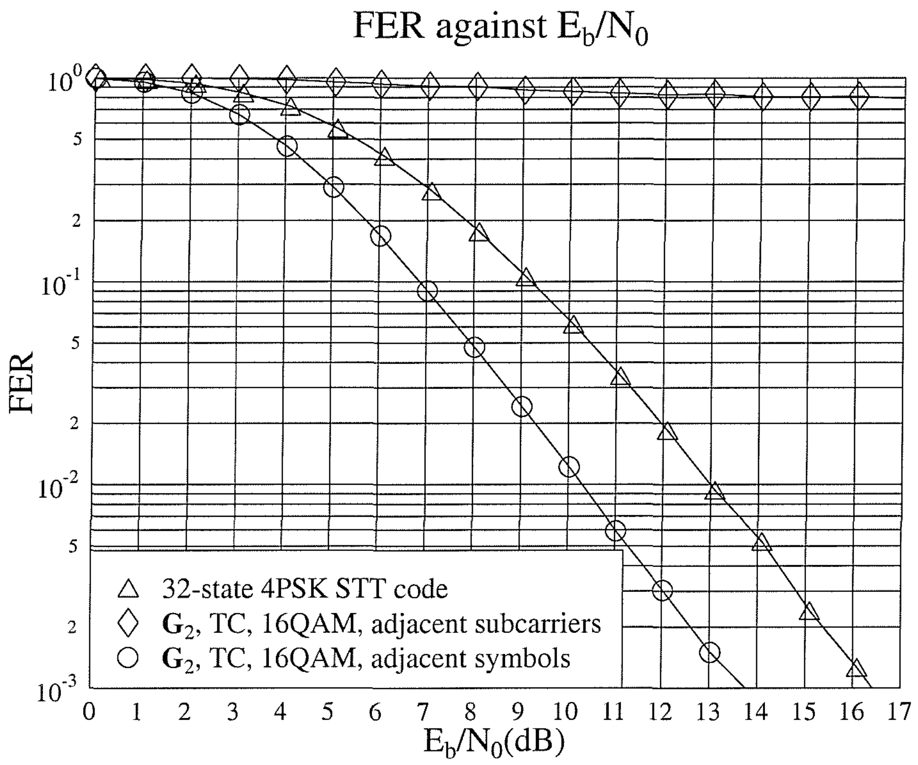


Figure 6.30: **FER** performance comparison between adjacent subcarriers and adjacent OFDM symbols allocation for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one receiver, the 128-subcarrier OFDM modem and 16QAM over a channel having a CIR characterised by two equal-power rays separated by a delay spread of $40\mu s$. The maximum Doppler frequency was 100 Hz. The coding parameters are shown in Tables 6.2, 6.3 and 6.4.

In Figure 6.30, we show our FER performance comparison for the above two scenarios, namely using two adjacent subcarriers and the same subcarrier in two consecutive OFDM symbols for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one 128-subcarrier 16QAM OFDM receiver. As before, the CIR exhibited two equal-power rays separated by a delay spread of $40\mu s$ and a maximum Doppler frequency of 100 Hz. It can be seen from the figure that there is a severe performance degradation, if the two transmission instants of the space-time block $\mathbf{G}_2$ are allocated to two adjacent subcarriers. This is evidenced by the near-horizontal curve marked by diamonds across the figure. On the other hand, upon assuming that having a delay of two OFDM-symbol durations does not pose any problems in terms of real-time interactive communications, we can allocate the two transmission instants of the space-time block code $\mathbf{G}_2$ to the same subcarrier of two consecutive OFDM symbols. From Figure 6.30, we can observe a dramatic improvement over the previous allocation method. Furthermore, the figure also indicates that by tolerating a two OFDM-symbol delay, the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme outperforms the 32-state 4PSK space-time trellis code by approximately 2 dB in terms of the required E_b/N_0 value at a FER of 10^{-3}.

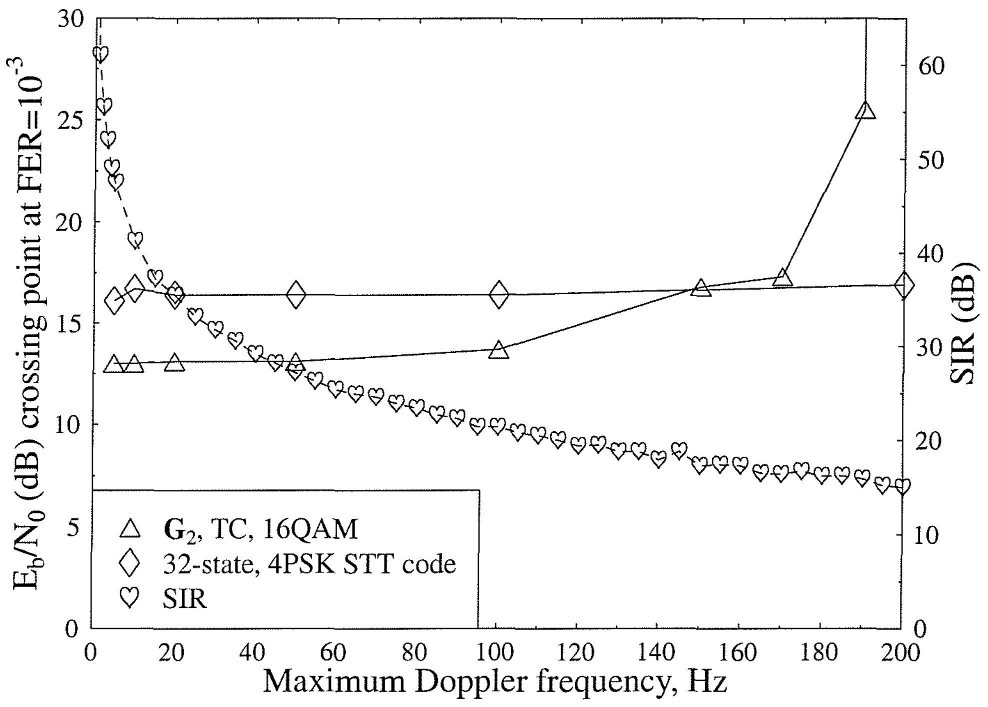


Figure 6.31: The E_b/N_0 value required for maintaining FER=10^{-3} versus the maximum Doppler frequency for the 32-state 4PSK space-time trellis code and for the adjacent OFDM symbols allocation of the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using **one receiver** and the 128-subcarrier OFDM modem. The CIR exhibited two equal-power rays separated by a delay spread of $40\mu s$. The effective throughput was **2 BPS** and the coding parameters are shown in Tables 6.2, 6.3 and 6.4. The *SIR* of various maximum Doppler frequencies are shown as well.

Since the two transmission instants of the space-time block code $\mathbf{G}_2$ are allocated to the same subcarrier of two consecutive OFDM symbols, it is the maximum Doppler frequency that would affect the performance of the concatenated scheme more gravely, rather than the delay spread. Hence we extended our studies to consider the effects of the maximum Doppler frequency on the performance of the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme. Specifically, Figure 6.31 shows the E_b/N_0 values required for maintaining FER=10^{-3} versus the

Doppler frequency for the 32-state 4PSK space-time trellis code, and for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,3) code using one 128-subcarrier 16QAM OFDM receiver, when mapping the two transmission instants to the same subcarrier of two consecutive OFDM symbols. The channel exhibited two equal-power rays separated by a delay spread of $40\mu s$ and various maximum Doppler frequencies. The SIR achievable at various maximum Doppler frequencies is also shown in Figure 6.31. Again, we can see that the performance of the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme suffers severely, if the maximum Doppler frequency is above 160 Hz. More precisely, we can surmise that in order for the concatenated scheme to outperform the 32-state 4PSK space-time trellis code, the SIR should be at least 15 dB, which is about the same as the required SIR in Figure 6.28. From Figure 6.28 and 6.31, we can conclude that the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme performs better, if the SIR is in excess of about 10-15 dB.

6.4.6 The Wireless Asynchronous Transfer Mode System

We have previously investigated the performance of different schemes over two-path channels having two equal-power rays. In this section, we investigate the performance of the proposed systems over indoor Wireless Asynchronous Transfer Mode (WATM) channels. The WATM system used 512 subcarriers and each OFDM symbol was extended with a cyclic prefix of length 64. The sampling rate was 225 Msamples/s and the carrier frequency was 60 GHz. In [153] two WATM CIRs were used, namely a five-path and a three-path model, where the latter one was referred to as the shortened WATM CIR. This CIR was used also in our investigations here.

The shortened WATM channel's impulse response is depicted in Figure 6.32, where the longest-delay path arrived at a delay of 48.9 ns, which corresponds to 11 sample periods at the 225 Msamples/s sampling rate. The 512-subcarrier OFDM time-domain transmission frame having a cyclic extension of 64 samples is shown in Figure 6.33.

6.4.6.1 Channel Coded Space-Time Codes – Throughput of 1 BPS

Previously, we have compared the performance of the space-time trellis codes to that of the TC(2,1,3)-coded space-time block code $\mathbf{G}_2$. We now extend our comparisons to Reed-Solomon (RS) coded space-time trellis codes, which were used in [76, 78, 79]. In Figure 6.34 we show our FER performance comparison between the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ and the RS(102,51) GF(2^{10}) coded 16-state 4PSK space-time trellis code using one 512-subcarrier OFDM receiver over the shortened WATM CIR of Figure 6.32 at an

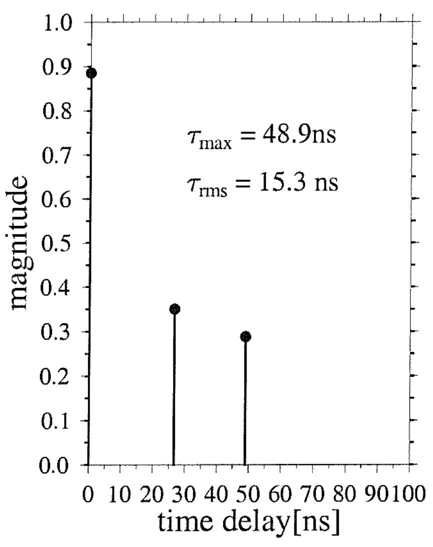


Figure 6.32: Short WATM channel impulse response.

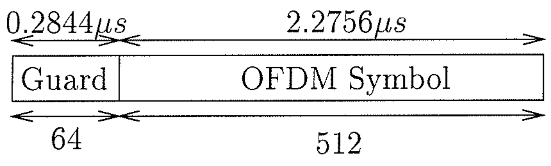


Figure 6.33: Short WATM plot of 512-subcarrier OFDM time domain signal with a cyclic extension of 64 samples.

effective throughput of 1 BPS. We can see from the figure that the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ outperforms the RS(102,51) GF(2^{10}) coded 16-state 4PSK space-time trellis code by approximately 5 dB in E_b/N_0 terms at a FER of 10^{-3}. The performance of the RS(102,51) GF(2^{10}) coded 16-state 4PSK space-time trellis code would be improved by about 2 dB, if the additional complexity of maximum likelihood decoding were affordable. However, even assuming this improvement, the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ would outperform the RS(102,51) GF(2^{10}) coded 16-state 4PSK space-time trellis code.

6.4.6.2 Channel Coded Space-Time Codes – Throughput of 2 BPS

In our next experiment, the throughput of the system was increased to 2 BPS by employing a higher-order modulation scheme. In Figure 6.35 we show our FER performance comparison

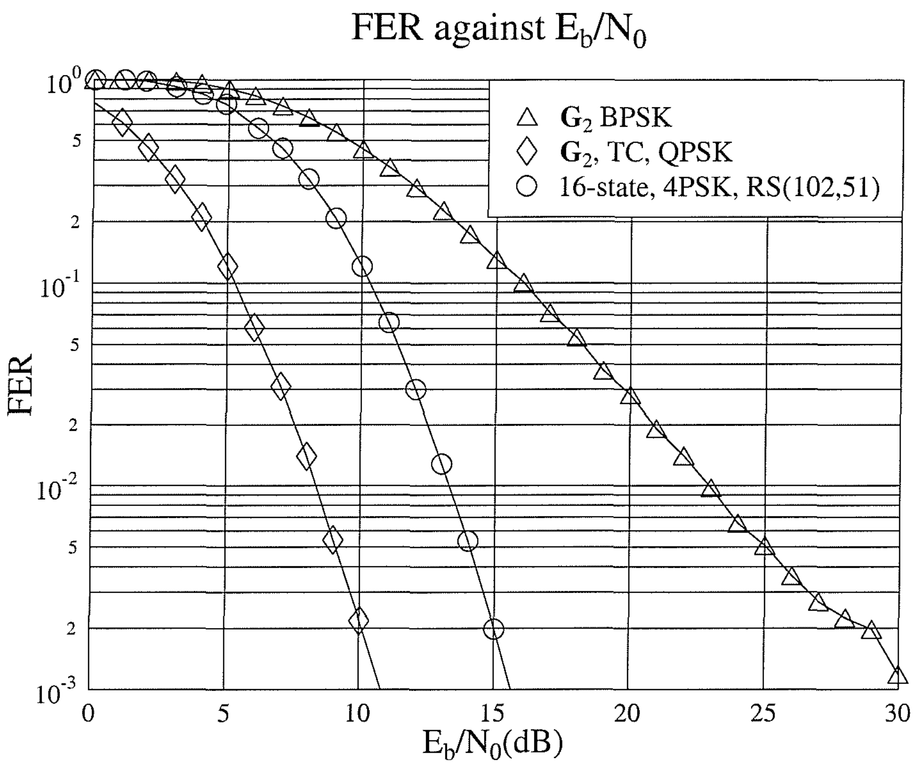


Figure 6.34: **FER** performance comparison between the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ and the RS(102,51) GF(2^{10}) coded 16-state 4PSK space-time trellis code using one 512-subcarrier OFDM receiver over the shortened WATM channel at an effective throughput of **1 BPS**. The coding parameters are shown in Tables 6.2, 6.3, 6.4 and 6.5

between the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ and the RS(153,102) GF(2^{10}) coded 16-state 8PSK space-time trellis code using one 512-subcarrier OFDM receiver over the shortened WATM channel of Figure 6.32 at an effective throughput of 2 BPS. Again, we can see that the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ outperforms the RS(153,102) GF(2^{10}) coded 16-state 8PSK space-time trellis code by approximately 5 dB in terms of E_b/N_0 at a FER of 10^{-3}. The corresponding performance of the 32-state 4PSK space-time trellis code is also shown in the figure. It can be seen that its performance is about 13 dB worse in E_b/N_0 terms, than that of the TC(2,1,3) coded space-time block code $\mathbf{G}_2$. Let us now continue our investigations by considering, whether channel-quality controlled adaptive space-time coded OFDM is capable of providing further performance benefits.

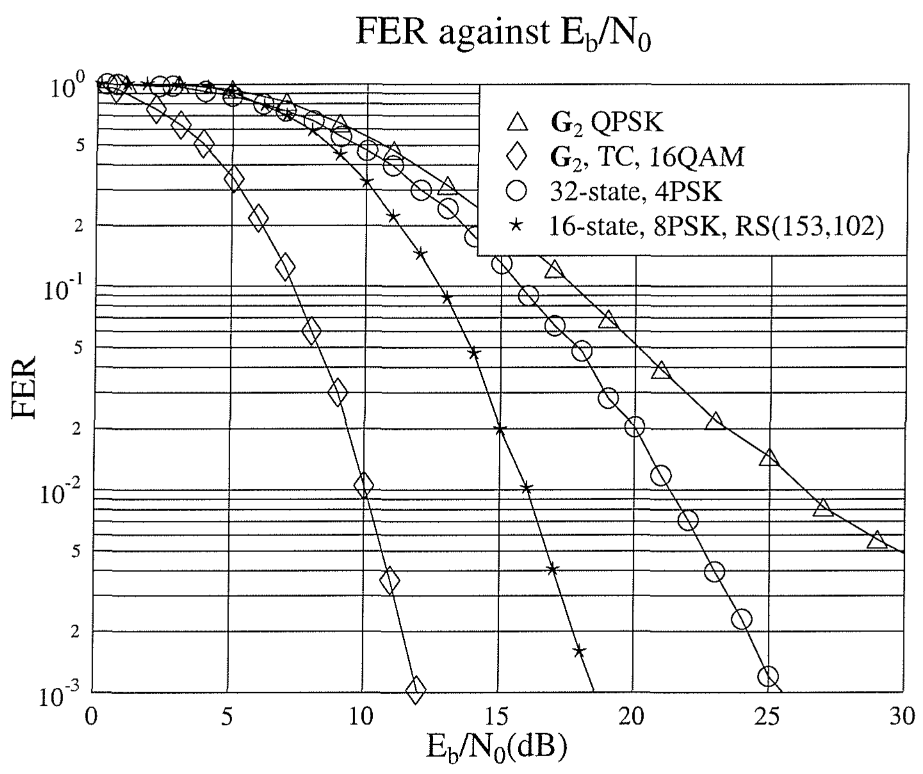


Figure 6.35: **FER** performance comparison between the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ and the RS(153,102) GF(2^{10}) coded 16-state 8PSK space-time trellis code using one 512-subcarrier OFDM receiver over the shortened WATM channel at an effective throughput of 2 BPS. The coding parameters are shown in Tables 6.2, 6.3, 6.4 and 6.5.

6.5 Space-Time Coded Adaptive Modulation for OFDM

6.5.1 Introduction

Adaptive modulation was proposed by Steele and Webb [185, 186], in order to combact the time-variant fading of mobile channels. The main idea of adaptive modulation is that when the channel quality is favourable, higher-order modulation modes are employed, in order to increase the throughput of the system. On the other hand, more robust but lower-throughput modulation modes are employed, if the channel quality is low. This simple but elegant idea has motivated a number of researchers to probe further [153, 187–194].

Recently adaptive modulation was also proposed for OFDM, which was termed adaptive OFDM (AOFDM) [153, 190, 191, 195]. AOFDM exploits the variation of the signal quality both in the time domain as well as in the frequency domain. In what is known as sub-band adaptive OFDM transmission, all subcarriers in an AOFDM symbol are split into blocks of adjacent subcarriers, referred to as sub-bands. The same modulation scheme is employed for all subcarriers of the same sub-band. This substantially simplifies the task of signalling the modulation modes, since there are typically four modes and for example 32 sub-bands, requiring a total of 64 AOFDM mode signalling bits.

6.5.2 Turbo-Coded and Space-Time-Coded Adaptive OFDM

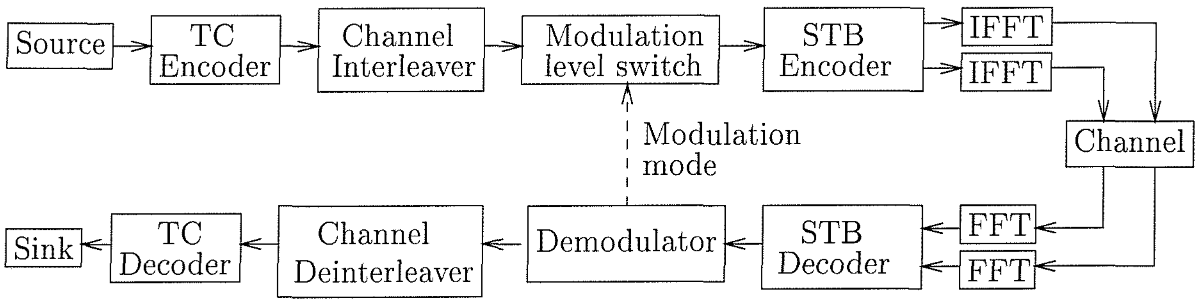


Figure 6.36: System overview of the turbo-coded and space-time-coded adaptive OFDM.

In this section, the adaptive OFDM philosophy proposed by Keller *et al.* [153,190,191] is extended, in order to exploit the advantages of multiple transmit and receive antennas. Besides, turbo coding is also employed in order to improve the performance of the system. In Figure 6.36, we show the system schematic of the turbo-coded and space-time-coded adaptive OFDM system. Similarly to Figure 6.13, random data bits are generated and encoded by the TC(2,1,3) encoder using an octal generator polynomial of $(7,5)$. Various TC(2,1,3) coding rates were used for the different modulation schemes. The encoded bits were channel interleaved and passed to the modulator. The choice of the modulation scheme to be used by the transmitter for its next OFDM symbol is determined by the channel quality estimate of the receiver based on the current OFDM symbol. In this study, we assumed perfect channel quality estimation and perfect signalling of the required modem mode of each sub-band based on the channel quality estimate acquired during the current OFDM symbol. Aided by the perfect channel quality estimator, the receiver determines the highest-throughput modulation mode to be employed by the transmitter for its next transmission while maintaining the system's target BER. Five possible transmission modes were employed in our investigations, which are no transmission (NoTx), BPSK, QPSK, 16QAM and 64QAM. In order to simplify the task of signalling the required modulation modes, we employed the sub-band adaptive OFDM transmission scheme proposed by Keller *et al.* [153,190,191]. The modulated signals were then passed to the encoder of the space-time block code $\mathbf{G}_2$. The space-time encoded signals were OFDM modulated and transmitted by the corresponding antennas. The shortened WATM channel was used, where the CIR profile and the OFDM transmission frame are shown in Figures 6.32 and 6.33, respectively.

The number of receivers invoked constitutes a design parameters. The received signals were OFDM demodulated and passed to the space-time decoders. Log-MAP [171] decoding of the received space-time signals was performed, in order to provide soft-outputs for the TC(2,1,3) decoder. Assuming that the demodulator of the receiver has perfect knowledge of the instantaneous channel quality, this information is passed to the transmitter in order to determine its next AOFDM modulation mode allocation. The received bits were then channel deinterleaved and passed to the TC decoder, which again, employs the Log-MAP decoding algorithm [52]. The decoded bits were finally passed to the sink for calculation of the BER.

6.5.3 Simulation Results

As mentioned earlier, all the AOFDM based simulation results were obtained over the shortened WATM channel. The channels' profile and the OFDM transmission frame structure are shown in Figures 6.32 and 6.33, respectively. Again, Jakes' model [184] was adopted for modelling the fading channels.

In order to obtain our simulation results, several assumptions were stipulated:

- The average signal power received from each transmitter antenna was the same;
- All multipath components undergo independent Rayleigh fading;
- The receiver has a perfect knowledge of the CIR;
- Perfect signalling of the AOFDM modulation modes.

Again, we note that the above assumptions are unrealistic, yielding the best-case performance, nonetheless, they facilitate the performance comparison of the various techniques under identical circumstances.

6.5.3.1 Space-Time Coded Adaptive OFDM

In this section, we employ the fixed threshold based modem mode selection algorithm, which was also used in [153], adapting the technique proposed by Torrance [188, 196, 197] for serial modems. Torrance assumed that the channel quality is constant for all the symbols in a transmission burst, i.e. that the channel's fading envelope varied slowly across the transmission burst. Under these conditions, all the transmitted symbols are modulated using the same modulation mode, chosen according to the predicted SNR. Torrance optimised the modem mode switching thresholds [188, 196, 197] for the target BERs of 10^{-2} and 10^{-4}, which will be appropriate for a high-BER speech system and for a low-BER data system, respectively. The resulting SNR switching thresholds for activating a given modu-

System	NoTx	BPSK	QPSK	16QAM	64QAM
Speech	$-\infty$	3.31	6.48	11.61	17.64
Data	$-\infty$	7.98	10.42	16.76	26.33

Table 6.7: Optimised switching levels quoted from [188] for adaptive modulation over Rayleigh fading channels for the speech and data systems, shown in instantaneous channel SNR (dB)

lation mode in a slowly Rayleigh fading narrowband channel are given in Table 6.7 for both systems. Assuming perfect channel quality estimation, the instantaneous channel SNR is measured by the receiver and the information is passed to the modulation mode selection switch at the transmitter, as shown in Figure 6.36 using the system's control channel. This side-information signalling does not constitute a problem, since state-of-the-art wireless systems, such as for example IMT-2000 [49] have a high-rate, low-delay signalling channel. This modem mode signalling feedback information is utilised by the transmitter for selecting the next modulation mode. Specifically, a given modulation mode is selected, if the instantaneous channel SNR perceived by the receiver exceeds the corresponding switching levels shown in Figure 6.7, depending on the target BER.

As mentioned earlier, the proposed adaptation algorithm [188,196,197] assumes constant instantaneous channel SNR over the whole transmission burst. However, in the case of an OFDM system transmitting over frequency selective channels, the channels' quality varies across the different subcarriers. Keller *et al.* proposed employing the lowest-quality subcarrier in the sub-band for controlling the adaptation algorithm based on the switching thresholds given in Table 6.7. Again, this approach significantly simplifies the signalling and therefore it was also adopted in our investigations.

In Figure 6.37, we show the BER and BPS performance of the 16 sub-band AOFDM scheme employing the space-time block code $\mathbf{G}_2$ in conjunction with multiple receivers and a target BER of 10^{-4} over the shortened WATM channel shown in Figure 6.32. The switching thresholds are shown in Table 6.7. The performance of the conventional AOFDM scheme using no diversity [153] is also shown in the figure. From Figure 6.37, we can see that the BPS performance of the space-time coded AOFDM scheme using one receiver is

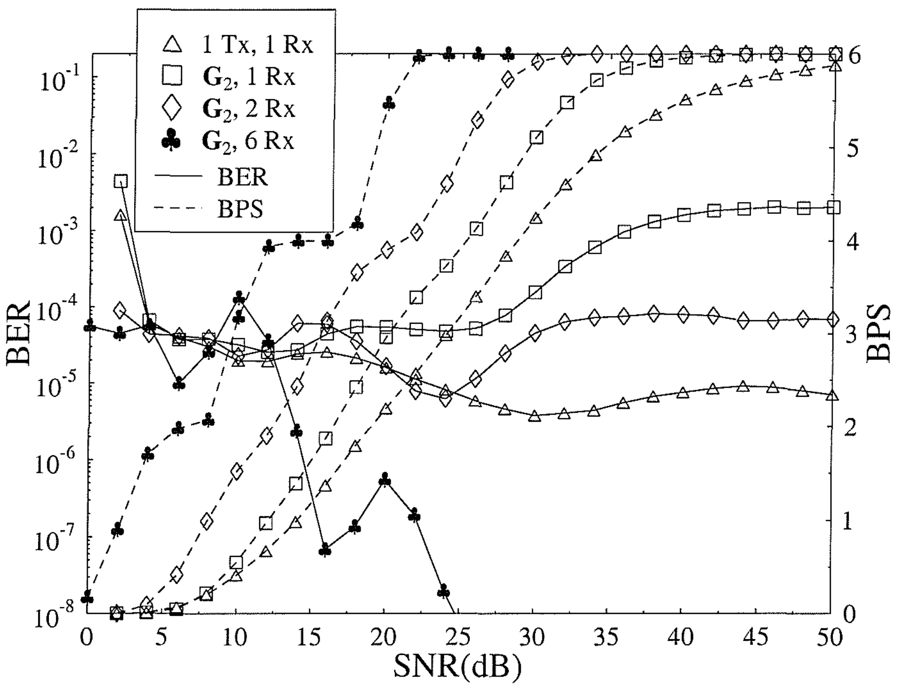


Figure 6.37: BER and BPS performance of 16 sub-band AOFDM employing the space-time block code $\mathbf{G}_2$ using multiple receivers for a target BER of 10^{-4} over the shortened WATM channel shown in Figure 6.32 and the transmission format of Figure 6.33. The switching thresholds are shown in Table 6.7.

better, than that of the conventional AOFDM scheme. The associated performance gain improves, as the throughput increases. At a throughput of 6 BPS, the space-time coded scheme outperforms the conventional scheme by at least 10 dB in E_b/N_0 terms. However, we notice in Figure 6.37 that as a secondary effect, the BER performance of the space-time coded AOFDM scheme using one receiver degrades, as we increase the average channel SNR. Again, this problem is due to the interference of signals x_1 and x_2 caused by the rapidly varying frequency-domain fading envelope across the subcarriers. At high SNRs, predominantly 64QAM was employed. Since the constellation points in 64QAM are densely packed, this modulation mode is more sensitive to the 'cross-talk' of the signals x_1 and x_2. This limited the BER performance to 10^{-3} even at high SNRs. However, at SNRs lower than 30 dB typically more robust modulation modes were employed and hence the target BER of 10^{-4} was readily met. We will show in the next section that this problem can be overcome by employing turbo channel coding in the system.

In Figure 6.37 we also observe that the BER and BPS performance improves, as we increase the number of AOFDM receivers, since the interference between the signals x_1 and x_2 is eliminated. Upon having six AOFDM receivers, the BER of the system drops below 10^{-8}, when the average channel SNR exceeds 25 dB and there is no sign of the BER flattening effect. At a throughput of 6 BPS, the space-time coded AOFDM scheme using six receivers outperforms the conventional system by more than 30 dB.

Figure 6.38 shows the probability of each AOFDM sub-band modulation mode for (a) conventional AOFDM and for space-time coded AOFDM using (b) 1, (c) 2 and (d) 6 receivers over the shortened WATM channel shown in Figure 6.32. The transmission format obeyed Figure 6.33. The switching thresholds were optimised for the data system having a target BER of 10^{-4} and they are shown in Table 6.7. By employing multiple transmitters and receivers, we increase the diversity gain and we can see in the figure that this increases the probability of the most appropriate modulation mode at a certain average channel SNR. This is clearly shown by the increased peaks of each modulation mode at different average channel SNRs. As an example, in Figure 6.38 (d) we can see that there is an almost 100% probability of transmitting in the QPSK and 16QAM modes at average channel SNR of approximately 6 dB and 15 dB, respectively. This strongly suggest that it is a better solution, if fixed modulation based transmission is employed in space-time coded OFDM, provided that we can afford the associated complexity of using six receivers. We shall investigate these issues in more depth at a later stage.

On the other hand, the increased probability of a particular modulation mode at a certain average channel SNR also means that there is less frequent switching amongst the various modulation modes. For example, we can see in Figure 6.38 (b) that the probability of employing 16QAM increased to 0.8 at an average channel SNR of 25 dB compared to 0.5 in Figure 6.38 (a). Furthermore, there are almost no BPSK transmissions at SNR=25 dB in Figure 6.38 (b). This situation might be an advantage in the context of the AOFDM system, since most of the time the system will employ 16QAM and only occasionally switches to the QPSK and 64QAM modulation modes. This can be potentially exploited to reduce AOFDM modem mode the signalling traffic and to simplify the system.

The characteristics of the modem mode probability density functions in Figure 6.38 in conjunction with multiple transmit antennas can be further explained with the aid of Figure 6.39. In Figure 6.39 we show the instantaneous channel SNR experienced by the 512-subcarrier OFDM symbols for a single-transmitter, single-receiver scheme and for the space-time block code $\mathbf{G}_2$ using one, two and six receivers over the shortened WATM channel. The average channel SNR is 10 dB. We can see in Figure 6.39 that the variation of the instantaneous channel SNR for a single transmitter and single receiver is fast and severe. The

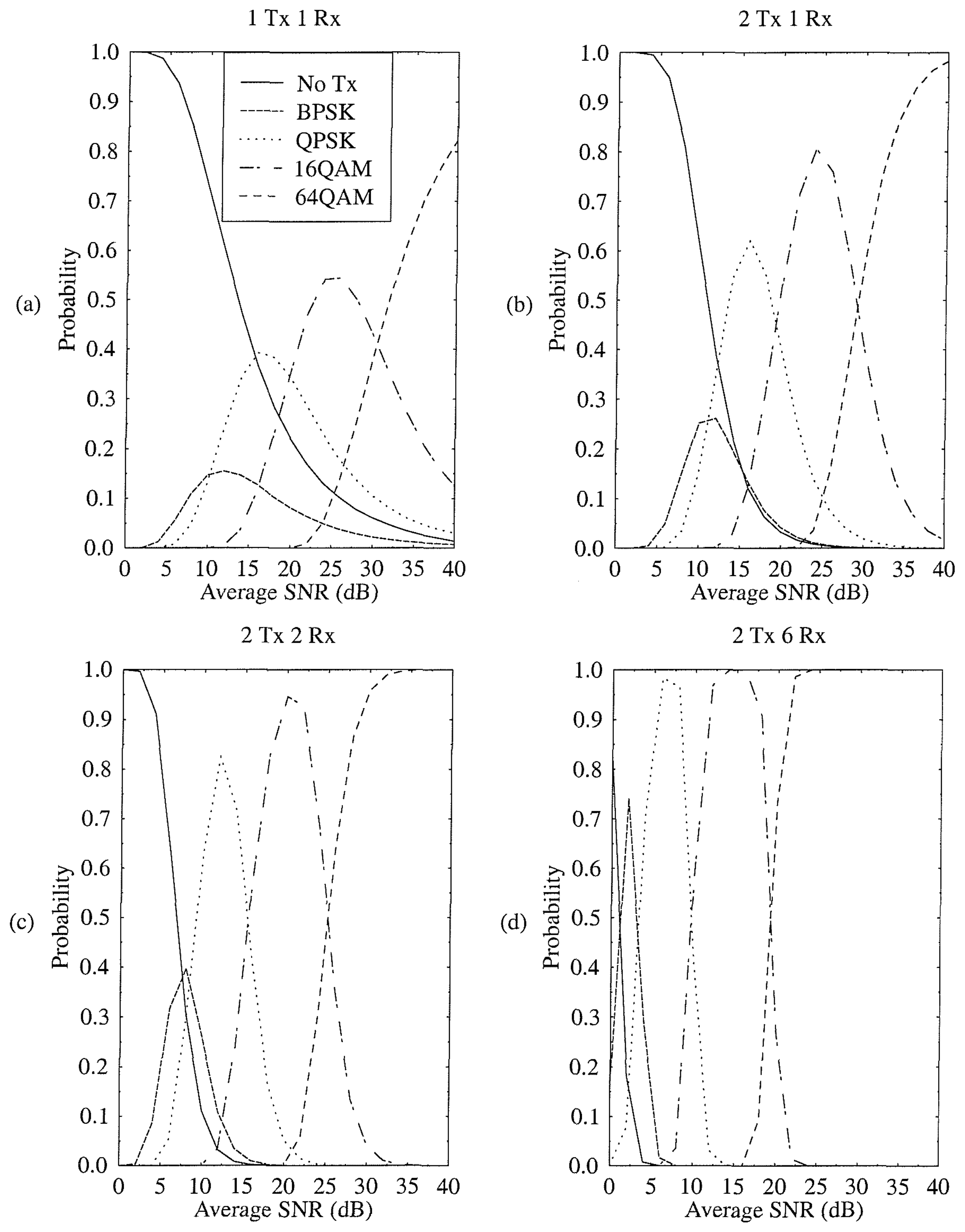


Figure 6.38: Probability of each modulation mode for (a) conventional AOFDM and for space-time coded AOFDM using (b) 1, (c) 2 and (d) 6 receivers over the shortened WATM channel shown in Figure 6.32 and using the transmission frame of Figure 6.33. The thresholds were optimised are for the data system and they are shown in Table 6.7. All sub-figures share the legends seen in Figure 6.38 (a).

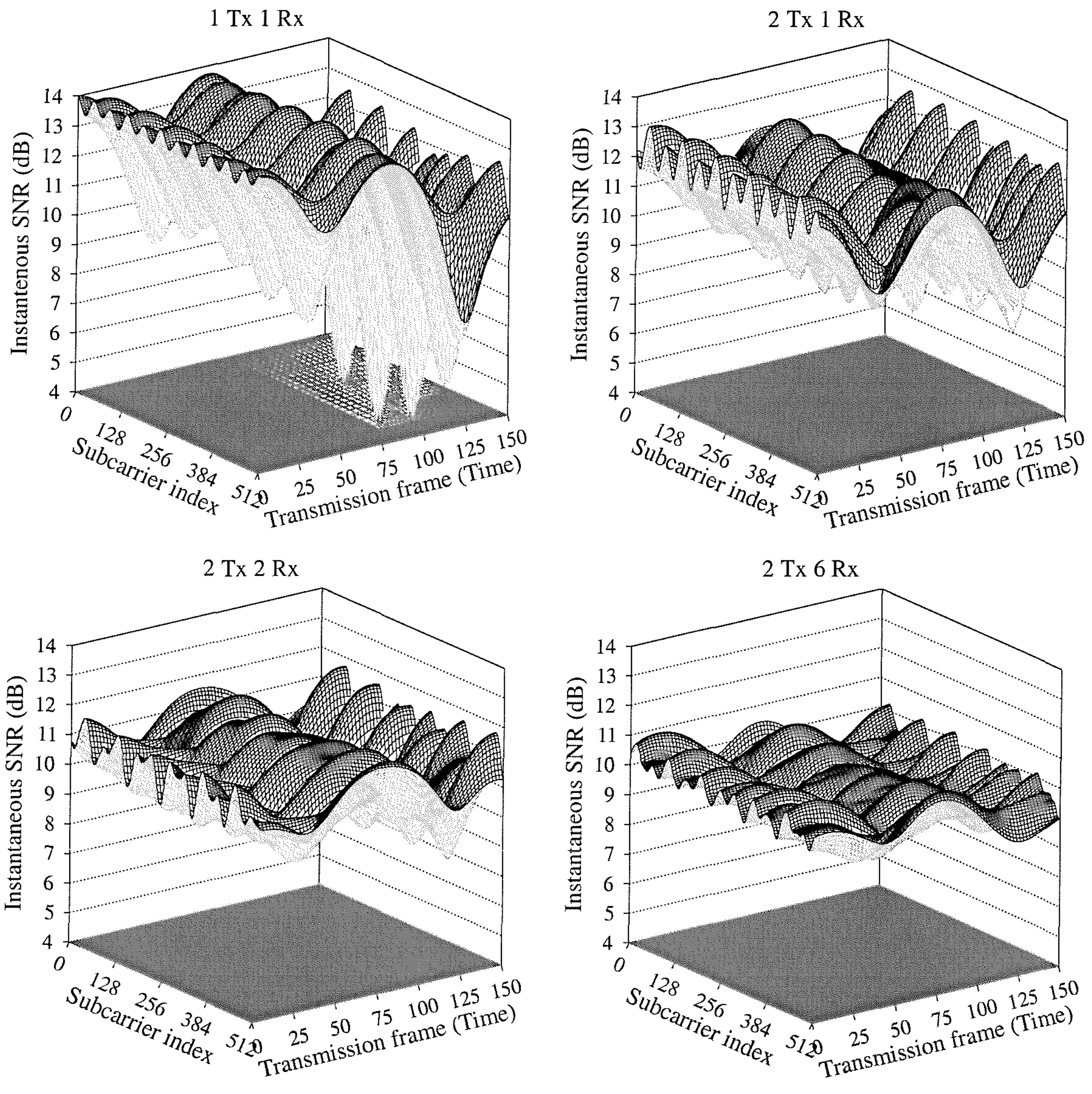


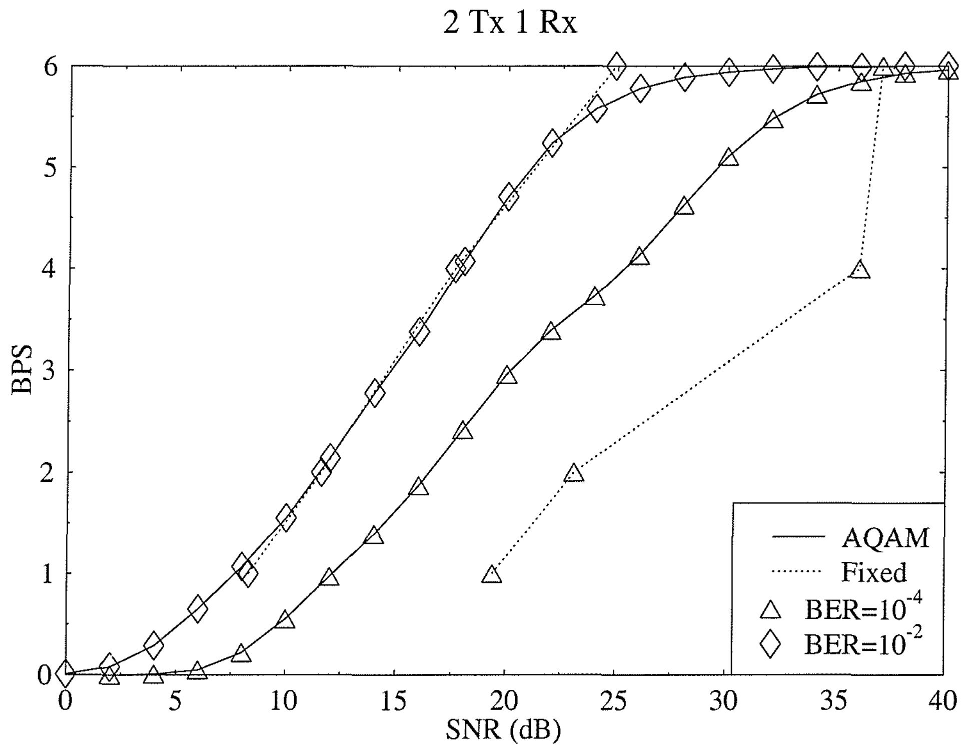
Figure 6.39: Instantaneous channel SNR of 512-subcarrier OFDM symbols for single-transmitter single-receiver and for the space-time block code $\mathbf{G}_2$ using one, two and six receivers over the shortened WATM channel shown in Figure 6.32 and using the transmission format of Figure 6.33. The average channel SNR is 10 dB.

instantaneous channel SNR may become as low as 4 dB due to deep fades of the channel. On the other hand, we can see that for the space-time block code $\mathbf{G}_2$ using one receiver the variation in the instantaneous channel SNR is slower and less severe. Explicitly, by employing multiple transmit antennas as shown in Figure 6.39, we have reduced the effect of the channels' deep fades significantly. This is advantageous in the context of adaptive modulation schemes, since higher-order modulation modes can be employed, in order to increase the throughput of the system. However, as we increase the number of receivers, i.e. the diversity order, we observe that the variation of the channel becomes slower. Effectively, by employing higher-order diversity, the fading channels have been converted to AWGN-like channels, as evidenced by the space-time block code $\mathbf{G}_2$ using six receivers. Since adaptive modulation only offers advantages over fading channels, we argue that using adaptive modulation might become unnecessary, as the diversity order is increased.

To elaborate a little further, from Figure 6.38 and 6.39 we surmise that fixed modulation schemes might become more attractive, when the diversity order increases, which is achieved in this case by employing more receivers. This is because for a certain average channel SNR, the probability of a particular modulation mode increases. In other words, the fading channel has become an AWGN-like channel, as the diversity order is increased. In Figure 6.40 we show our throughput performance comparison between AOFDM and fixed modulation based OFDM in conjunction with the space-time block code $\mathbf{G}_2$ employing (a) one receiver and (b) two receivers over the shortened WATM channel. The throughput of fixed OFDM was 1, 2, 4 and 6 BPS and the corresponding E_b/N_0 values were extracted from the associated BER versus E_b/N_0 curves of the individual fixed-mode OFDM schemes. It can be seen from Figure 6.40 (a) that the throughput performance of the adaptive and fixed OFDM schemes is similar for a 10^{-2} target BER system. However, for a 10^{-4} target BER system, there is an improvement of 5-10 dB in E_b/N_0 terms at various throughputs for the adaptive OFDM scheme over the fixed OFDM scheme. At high average channel SNRs the throughput performance of both schemes converged, since 64QAM became the dominant modulation mode for AOFDM.

On the other hand, if the number of receivers is increased to two, we can see in Figure 6.40 (b) that the throughput performance of both adaptive and fixed OFDM is similar for both the 10^{-4} and 10^{-2} target BER systems. We would expect similar trends, as the number of receivers is increased, since the fading channels become AWGN-like channels. From Figure 6.40, we conclude that AOFDM is only beneficial for the space-time block code $\mathbf{G}_2$ using one receiver in the context of the 10^{-2} target BER system.

(a)



(b)

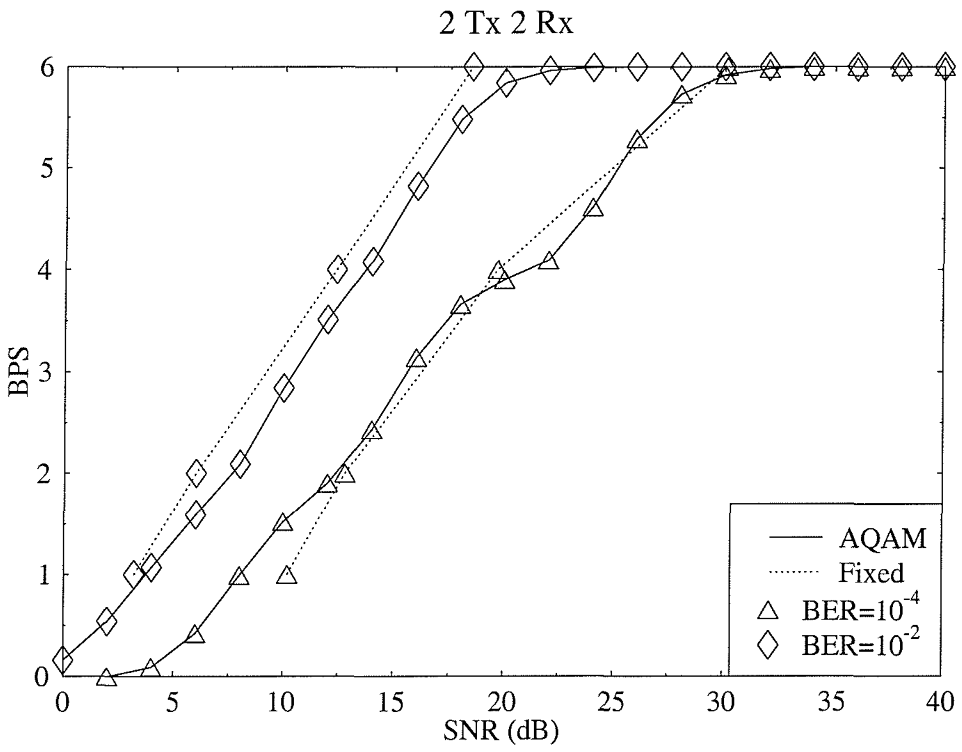


Figure 6.40: BPS throughput performance comparison between adaptive OFDM and fixed modulation based OFDM using the space-time block code $\mathbf{G}_2$ employing (a) one receiver and (b) two receivers over the shortened WATM channel shown in Figure 6.32 and the transmission format of Figure 6.33.

6.5.3.2 Turbo and Space-Time Coded Adaptive OFDM

In the previous section we have discussed the performance of space-time coded adaptive OFDM. Here we extend our study by concatenating turbo coding with the space-time coded AOFDM scheme in order to improve both the BER and BPS performance of the system. As earlier, the turbo convolutional code TC(2,1,3) having a constraint length of 3 and octal generator polynomial of $(7,5)$ was employed. Since the system was designed for high-integrity, low-BER data transmission, it was delay non-sensitive. Hence a random turbo interleaver size of approximately 10,000 bits was employed. The random separation channel interleaver [108] of Figure 5.14 was utilised in order to disperse the bursty channel errors. The Log-MAP [52] decoding algorithm was employed, using eight iterations.

We proposed two TC coded schemes for the space-time coded AOFDM system. The first scheme is a fixed half-rate turbo and space-time coded adaptive OFDM system. It achieves a high BER performance, but at the cost of a maximum throughput limited to 3 BPS due to half-rate channel coding. The second one is a variable-rate turbo and space-time coded adaptive OFDM system. This scheme sacrifices the BER performance in exchange for an increased system throughput. Different puncturing patterns are employed for the various code rates R. The puncturing patterns were optimised experimentally by simulations. The design procedure for punctured turbo codes was proposed by Acikel *et al.* [64] in the context of BPSK and QPSK. The optimum AOFDM mode switching thresholds were obtained by computer simulations over the shortened WATM channel of Figure 6.32 and they are shown in Table 6.8.

	NoTx	BPSK	QPSK	16QAM	64QAM
Half-rate TC(2,1,3)					
Rate	–	0.50	0.50	0.50	0.50
Thresholds (dB)	$-\infty$	-4.0	-1.3	5.4	9.8
Variable-rate TC(2,1,3)					
Rate	–	0.50	0.67	0.75	0.90
Thresholds (dB)	$-\infty$	-4.0	2.0	9.70	21.50

Table 6.8: Coding rates and switching levels (dB) for TC(2,1,3) and space-time coded adaptive OFDM over the shortened WATM channel of Figure 6.32 for a target BER of 10^{-4}.

In Figure 6.41, we show the BER and BPS performance of 16 sub-band AOFDM employing the space-time block code $\mathbf{G}_2$ concatenated with both half-rate and variable-rate TC(2,1,3) coding at a target BER of 10^{-4} over the shortened WATM channel of Figure 6.32.

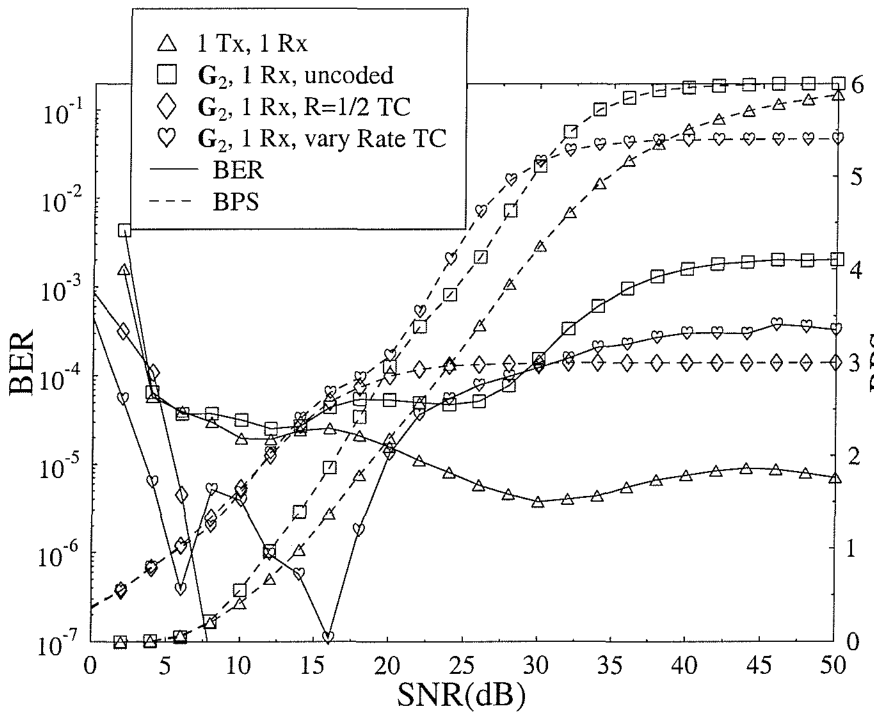


Figure 6.41: BER and BPS performance of 16 sub-band AOFDM employing the space-time block code $\mathbf{G}_2$ concatenated with both half-rate and variable-rate TC(2,1,3) at a target BER of 10^{-4} over the shortened WATM channel shown in Figure 6.32 and using the transmission format of Figure 6.33. The switching thresholds and coding rates are shown in Table 6.7.

We can see in the figure that by concatenating fixed half-rate turbo coding with the space-time coded adaptive OFDM scheme, the BER performance of the system improves tremendously, indicated by a steep dip of the associated BER curve marked by the solid line and diamonds. There is an improvement in the BPS performance as well, exhibiting an E_b/N_0 gain of approximately 5 dB and 10 dB at an effective throughput of 1 BPS, compared to space-time coded AOFDM and conventional AOFDM, respectively. However, again, the maximum throughput of the system is limited to 3 BPS, since half-rate channel coding was employed. In Figure 6.41, we can see that at an E_b/N_0 value of about 30 dB the maximum throughput of the turbo coded and space-time adaptive OFDM system is increased from 3.0 BPS to 5.4 BPS by employing the variable-rate TC(2,1,3) code. Furthermore, the BPS performance of the variable-rate turbo coded scheme is similar to that of the half-rate turbo coded scheme at an average channel SNR below 15 dB. The BER curve marked by the solid line and clubs drops, as the average channel SNR is increased from 0 dB to 15 dB. Due to the increased probability of the 64QAM transmission mode, the variable-rate turbo coded scheme was overloaded by the plethora of channel errors introduced by the 64QAM mode. Therefore, we can see in Figure 6.41 that the BER increases and stabilises at 10^{-4}. Again, the interference of the signals x_1 and x_2 in the context of the space-time block code $\mathbf{G}_2$ prohibits further improvements in the BER performance, as the average channel SNR is increased. However, employing the variable-rate turbo codec has lowered the BER floor, as demonstrated by the curve marked by the solid line and squares.

6.6 Summary and Conclusion

Space-time trellis codes [70,80,166–169] and space-time block codes [71–73] constitute state-of-the-art transmission schemes based on multiple transmitters and receivers. Both codes have been introduced in Section 6.1. Since we have detailed the encoding and decoding processes of the space-time block codes in Chapter 5, the detailed description of the codes was left out of this chapter. Instead, space-time trellis codes were introduced in Section 6.2 by utilising the simplest possible 4-state, 4PSK space-time trellis code as an example. The state diagrams for other 4PSK and 8PSK space-time trellis codes were also given. The branch metric of each trellis transition was derived, in order to facilitate their maximum likelihood (ML) decoding.

In Section 6.3, we proposed to employ an OFDM modem for mitigating the effects of dispersive multipath channels due to its simplicity compared to other approaches [74, 75]. Turbo codes and Reed-Solomon codes were invoked in Section 6.3.1 for concatenation with the space-time block code $\mathbf{G}_2$ and the various space-time trellis codes, respectively. The estimated complexity of the various space-time trellis codes was derived in Section 6.3.3.

We presented our simulation results for the proposed schemes in Section 6.4. The first scheme studied was the TC(2,1,3) coded space-time block code $\mathbf{G}_2$, whereas the second one was based on the family of space-time trellis codes. It was found that the FER and BER performance of the TC(2,1,3) coded space-time block $\mathbf{G}_2$ was better than that of the investigated space-time trellis codes at a throughput of 2 and 3 BPS over the channel exhibiting two equal-power rays separated by a delay spread of $5\mu s$ and having a maximum Doppler frequency of 200 Hz. Our comparison between the two schemes was performed by also considering the estimated complexity of both schemes. It was found that the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme still outperformed the space-time trellis codes using no channel coding, even though both schemes exhibited a similar complexity.

The effect of the maximum Doppler frequency on both schemes was also investigated in Section 6.4.3. It was found that the maximum Doppler frequency had no significant impact on the performance of both schemes. By contrast, in Section 6.4.4 we investigated the effect of the delay spread on the system. Initially, the delay-spread dependent SIR of the space-time block code $\mathbf{G}_2$ was quantified. It was found that the performance of the concatenated TC(2,1,3)-$\mathbf{G}_2$ scheme degrades, as the delay spread increases due to the decrease in the associated SIR. However, varying the delay spread had no significant effect on the space-time trellis codes. We proposed in Section 6.4.5 an alternative mapping of the two transmission instants of the space-time block code $\mathbf{G}_2$ to the same subcarrier of two consecutive OFDM symbols, a solution which was applicable to a delay non-sensitive system. By employing this approach, the performance of the concatenated scheme was no longer limited by the delay spread, but by the maximum Doppler frequency. We concluded that a certain minimum SIR has to be maintained for attaining the best possible performance of the concatenated scheme.

The shortened WATM channel was introduced in Section 6.4.6. In this section, space-time trellis codes were concatenated with Reed-Solomon codes, in order to improve the performance of the system. Once again, both channel coded space-time block and trellis codes were compared at a throughput of 1 and 2 BPS. It was also found that the TC(2,1,3) coded space-time block code $\mathbf{G}_2$ outperforms the RS coded space-time trellis codes.

Space-time block coded AOFDM was proposed in the Section 6.5, which is the last section of this chapter. It was shown in Section 6.5.3.1 that only the space-time block code $\mathbf{G}_2$ using one AOFDM receiver outperformed the conventional single-transmitter, single-receiver AOFDM system designed for a data transmission target BER of 10^{-4} over the shortened WATM channel. We also confirmed that upon increasing the diversity order, the fading channels become AWGN-like channels. This explains, why fixed-mode OFDM transmission constitutes a better trade-off, than AOFDM, when the diversity order is increased. In Section 6.5.3.2, we continued our investigations into AOFDM by concatenating turbo coding with the system. Two schemes were proposed: half-rate turbo and space-time coded AOFDM as well as variable-rate turbo and space-time coded AOFDM. Despite the impressive BER performance of the half-rate turbo and space-time coded scheme, the maximum throughput of the system was limited to 3 BPS. However, by employing the variable-rate turbo and space-time coded scheme, the BPS performance improved, achieving a maximum throughput of 5.4 BPS. However, the improvement in BPS performance was achieved at the cost of a poorer BER performance.

Chapter 7

Conclusions and Future Work

This concluding chapter gives a summary of the thesis. This will be followed by a range of ideas on future research.

7.1 Summary and Conclusions

This thesis commenced by a brief historical perspective on channel coding following Shannon's pioneering work [1]. Thanks to the invention of turbo codes by Berrou [12, 13], researchers have been able to approach the Shannon limit within 0.27 dB [63],

In Chapter 2, we commenced our discussions with a brief introduction to conventional BCH codes. This was followed by the description of the Viterbi algorithm [8] using the simple BCH(7,4,3) code. Several examples on hard decision and soft decision Viterbi decoding were given. Then in Section 2.3.5, we presented simulation results for both the Viterbi and the Berlekamp-Massey decoding algorithms. It was found that the performance of the soft decision Viterbi decoding algorithm is about 2 dB better, than that of the corresponding hard decision decoding algorithm. The coding gain of various BCH codes using the same codeword length n was plotted in Figure 2.19 and 2.20. We observed from the figures that the maximum coding gain of the various BCH codes was achieved typically at coding rates of $0.5 - 0.7$. Since a high complexity is incurred by employing the Viterbi decoding of BCH codes, the reduced complexity Chase algorithm was introduced in Section 2.4.2 which offers a significantly lower complexity. It was found that the Chase algorithm gives effectively the same performance, as Viterbi decoding. We note however that the complexity of the Chase algorithm increases exponentially with the minimum free distance d_{min} of the code, as well as with the number of test patterns.

After the introduction to conventional BCH codes, in Chapter 3 we embarked on investigating the novel turbo BCH codes. The structure of the turbo encoder and decoder was outlined and the principle of iterative decoding was highlighted. In Section 3.3.3, the derivation of the MAP [11] algorithm was presented. This is was followed by the portrayal of the reduced complexity Max-Log-MAP [50, 51], Log-MAP [52] algorithms and the SOVA [53, 54]. A simple example on iterative decoding of the turbo BCH(7,4,3) code was given in Section 3.4. We highlighted, how iterative decoding can improve the reliability of the decoded bits and eventually correct an increased number of errors. In Section 3.5 the MAP algorithm was modified in order to incorporate the parity check bit of extended BCH codes for enhancing the decoder's performance. This was vital in the iterative decoding of the family of extended turbo BCH codes.

Various parameters can affect the performance of turbo BCH codes and these effects were investigated in Section 3.6. We first studied the effect of iterative decoding on the performance of turbo BCH codes. It was found that the performance of turbo BCH codes does not improve significantly after four iterations. Various decoding algorithms were compared and it was found that the Log-MAP and Max-Log-MAP algorithms give similar performance. However, the Log-MAP algorithm performs badly, if the estimate of the channel reliability value L_c is imperfect. Puncturing was found to degrade the performance of turbo BCH codes. Hence we concluded that no puncturing should be applied to turbo BCH codes. The performance of turbo BCH codes employing various interleaver lengths was investigated as well. It was found that the best possible performance can be achieved by employing an interleaver depth of 5,000 bits over AWGN channels. In Section 3.6.6 a new interleaver design was proposed. This interleaver was referred to as the random-in-column block interleaver and it was shown to outperform other conventional block and random interleavers, when the interleaver dimension is $k \times k$. Furthermore, turbo BCH codes employing different BCH component codes were investigated and it was found that the turbo BCH(63,51,5) code performs within about 0.8 dB of the Shannon limit. The proposed modified MAP algorithm [106] was then employed, invoking extended BCH codes and it was shown in Section 3.6.10 that the extended turbo BCH(32,26,4) code outperforms the turbo BCH(31,26,3) code by approximately 0.5 dB at a BER of 10^{-5}.

In Chapter 4 we extended our research to the novel class of non-binary block codes referred to as Redundant Residue Number System (RRNS) codes. An RRNS code is a maximum-minimum distance block code, exhibiting identical distance properties to Reed-Solomon (RS) codes. In order to understand the basic principles of RRNS codes, we commenced the chapter with a brief introduction to the ancient theory of the Residue Number System

(RNS). This theory provide the basis for a promising way of supporting fast arithmetic operations [40,41] such as addition, subtraction and multiplication. After a brief introduction to the RNS, we showed in Section 4.2.6 that a RRNS can be obtained by incorporating extra moduli into the existing RNS. Then the coding theory of the RRNS codes was developed. We showed that the minimum free distance d_{min} of the RRNS code could be determined using Equation 4.37. The procedure for multiple error correction was laid out in Section 4.4. In Section 4.5, we introduced different bit-to-residue mapping methods resulting in systematic and non-systematic RRNS codes. Furthermore, we proposed a novel systematic mapping method, which results in higher code rates for systematic RRNS codes, than the conventional mapping. Then we modified the Chase algorithm so that the RRNS decoder became capable of processing soft inputs and providing soft outputs. This facilitated the iterative decoding of turbo RRNS codes. Our simulation results for the proposed RRNS codes using BPSK over AWGN channels were then given in Section 4.9. It was found that systematic RRNS codes outperform their non-systematic counterparts. The proposed mapping method which results in higher code rates for systematic RRNS codes gave a similar performance to that of the corresponding conventional systematic RRNS codes. Besides hard decision decoding, soft decision decoding of the proposed RRNS codes was also employed in order to improve the performance of the codes. It was found that the soft decoding performance of RS codes is similar to that of the RRNS codes. Performance results were also provided for turbo RRNS codes. We began with the performance comparison between the SISO Chase algorithm and other trellis decoding algorithms, using the turbo BCH(63,57) code as an example. It was shown in Figure 4.18 that the performance of the SISO Chase algorithm suffers only a small performance degradation compared to Log-MAP algorithm at a significantly reduced complexity. Then, similarly to Chapter 3, the effects of various parameters affecting the performance of turbo RRNS codes were investigated. We concluded that the best compromise in terms of the number of decoding iterations was four. In Chapter 4, we concluded that the performance of RRNS codes was similar to that of Reed-Solomon codes, while offering some advantages over Reed-Solomon codes. Specifically, short non-binary block codes can be readily designed using RRNS codes without having to shorten long codes, which is not an option in RS codes. Besides, in ARQ systems stronger RRNS codes can be obtained by transmitting more redundant residues without having to re-transmit the whole codeword. In [150,151,153], we exploited this principle in designing adaptive rate RRNS codes for near-instantaneously adaptive OFDM transmission over mobile communication channels.

In our last two chapters, Chapter 5 and 6, we expanded our research into a new area. The design of the channel codes was no longer considered to be independent from the modulation and transmit diversity scheme used. Instead, channel coding, modulation and transmit diversity were jointly designed, resulting in space-time codes. We studied two distinct types of space-time codes: space-time block codes [71–73] and space-time trellis codes [70, 80, 166–169]. Despite the slight associated performance penalty of space-time block codes in comparison to space-time trellis codes, space-time block codes are appealing in terms of their simplicity and performance. Allowed by the associated low complexity offered by the space-time block code $\mathbf{G}_2$, we concatenated it with turbo convolutional codes, in order to improve its performance. The performance of the concatenated scheme was then compared to that of space-time trellis codes. Specifically, in Chapter 5 we investigated space-time block codes, whereas space-time trellis codes were detailed in Chapter 6.

We commenced in Chapter 5 with a rudimentary introduction to Maximum Ratio Combining [71], which constituted the basis of our further studies into space-time block codes. This was followed by an introduction to Alamouti's simple space-time block code $\mathbf{G}_2$. Examples were given for the space-time block code $\mathbf{G}_2$ using one and two receivers, followed by various other space-time block codes. In Section 5.4 we proposed a system, which consists of the concatenation of the above-mentioned space-time block codes and a range of different channel codes. The channel coding schemes investigated were convolutional codes, turbo convolutional codes, turbo BCH codes, trellis coded modulation and turbo trellis coded modulation. The estimated complexity and memory requirements of the channel decoders were summarised in Section 5.4.3. Simulation results characterising the proposed concatenated system were presented in Section 5.5. We first compared the performance of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ without using channel codecs. It was found that as we increased the effective throughput of the system, the performance of the half-rate space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ degraded in comparison to that of the unity rate space-time code $\mathbf{G}_2$. This was because in order to maintain the same effective throughput, higher order modulation schemes had to be employed in conjunction with the half-rate space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$, which were more prone to errors and hence degraded the performance of the system. On the other hand, for the sake of maintaining the same diversity gain and same effective throughput we found that the performance of the space-time codes $\mathbf{H}_3$ and $\mathbf{H}_4$ was better, than that of the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$, respectively. Since the space-time code $\mathbf{G}_2$ has a code rate of unity, we were able to concatenate it with half-rate TC codes, while maintaining the same effective throughput, as that of the half-rate space-time code using no channel coding. We found that for the same effective throughput, the unity-rate $\mathbf{G}_2$ space-time coded and half-rate TC coded scheme provided substantial performance improvement over other space-time block codes. We concluded that the reduction in coding rate was best invested in turbo channel codes, rather than in space-time block codes.

In the second category of our investigations of channel coded space-time systems, we studied the effect of the binary channel codes' data and parity bits being mapped into different protection classes of multi-level modulation schemes. It was found that TC codes having different constraint lengths K require different mapping methods, as evidenced by Figure 5.12. By contrast, in the turbo BCH codes studied mapping of the parity bits to the higher-integrity protection class of a multi-level modulation scheme yielded a better performance. The so-called random separation based interleaver as shown in Figure 5.14 was proposed, in order to improve the performance of the system. The third set of results compared the performance of all proposed channel codes in conjunction with the space-time code $\mathbf{G}_2$. It was then found that the performance of the half-rate TC codes was better than that of the CC, TBCH, TCM and TTCM codes. The chapter was then concluded by comparing the $\mathbf{G}_2$ space-time coded channel codes upon taking their complexity into consideration. In Figures 5.29 and 5.30, we can clearly see that the half-rate TC codes give the best coding gain at a moderate complexity.

Finally, in Chapter 6, we explored the power of various space-time trellis codes, where the performance of the space-time trellis codes was compared to that of the TC coded space-time block code of Chapter 5. Space-time trellis codes were introduced in Section 6.2 by utilising the simplest possible 4-state, 4PSK space-time trellis code as an example. The state diagrams for other 4PSK and 8PSK space-time trellis codes were also provided. The branch metric of each trellis transition was derived, in order to facilitate maximum likelihood decoding. In Section 6.3, we proposed to employ an OFDM modem for mitigating the effects of dispersive multipath channels. This technique was invoked due to its simplicity compared to other approaches. Turbo convolutional codes and Reed Solomon codes were invoked in Section 6.3.1 for concatenation with the space-time block code $\mathbf{G}_2$ and the various space-time trellis codes, respectively. The estimated complexity of the various space-time trellis codes was derived in Section 6.3.3. We then presented our simulation results for the proposed schemes in Section 6.4. The first scheme studied was the TC(2,1,3) coded space-time block code $\mathbf{G}_2$, whereas the second one was based on the family of space-time trellis codes. It was found that the FER and BER performance of the TC(2,1,3) coded space-time block $\mathbf{G}_2$ was better than that of the investigated space-time trellis codes at a throughput of 2 and 3 BPS over the channel exhibiting two equal-power rays separated by a delay spread of $5\mu s$ and having a maximum Doppler frequency of 200 Hz. Our comparison between the two schemes was performed by also considering the estimated complexity of both schemes. It was found that the concatenated $\mathbf{G}_2$/TC(2,1,3) scheme still outperformed the space-time trellis codes using no channel coding, even though both schemes exhibited a similar complexity. The effect of the maximum Doppler frequency on both schemes was also investigated in Section 6.4.3. It was found that the maximum Doppler frequency had no significant impact on the performance of either scheme. By contrast, in Section 6.4.4, it was found that the performance of the concatenated TC(2,1,3)-$\mathbf{G}_2$ scheme degrades, when the Doppler frequency is increased. Then, the Doppler-dependent channel-induced interference of the $\mathbf{G}_2$ space-time coded system was quantified in terms of the SIR. It was found that the SIR decreases, as the delay spread increases and this phenomenon degrades the performance of the concatenated scheme. We proposed in Section 6.4.5 an alternative mapping of the two transmission instants of the space-time block code $\mathbf{G}_2$ to the same subcarrier of two consecutive OFDM symbols, a solution which was applicable to a delay non-sensitive system. By employing this approach, the performance of the concatenated scheme was no longer limited by the delay spread, but rather by the maximum Doppler frequency. We concluded that a certain minimum SIR has to be maintained for attaining the best possible performance of the concatenated scheme.

Space-time block coded adaptive OFDM was proposed in Section 6.5. It was shown in Figure 6.40 that the space-time block code $\mathbf{G}_2$ using only one AOFDM receiver outperformed the conventional single-transmitter, single-receiver AOFDM system designed for a data transmission target BER of 10^{-4} over a WATM channel. We also confirmed that upon increasing the diversity order, the fading channels become AWGN-like channels. In Section 6.5.3.2, we continued our investigations into AOFDM by concatenating turbo coding with the system. Two schemes were proposed: half-rate turbo and space-time coded AOFDM as well as variable-rate turbo and space-time coded AOFDM. Despite the impressive BER performance of the half-rate turbo and space-time coded scheme, the maximum throughput of the system was limited to 3 BPS. However, by employing the proposed variable-rate turbo and space-time coded scheme, the BPS performance improved, achieving a maximum throughput of 5.4 BPS. This improvement in BPS performance terms was achieved at the cost of a poorer BER performance.

7.2 Future Work

In Chapter 3 we showed that the performance of turbo BCH codes is impressive. However, in Section 5.5.3, we demonstrated that similar performance can be obtained using punctured TC codes at the cost of a lower complexity. Nonetheless, the SISO Chase algorithm could be employed in decoding turbo BCH codes, which significantly reduced the complexity at the cost of a small performance degradation, as shown in Figure 4.18. The complexity and performance of employing the SISO Chase algorithm in the context of turbo BCH codes should be further investigated. Carrying out a comparative study of high rate turbo convolutional codes employing the Log-MAP algorithm and turbo BCH codes employing the SISO Chase algorithm would be beneficial.

It is widely recognised that space-time coded OFDM constitutes an attractive approach to transmission over dispersive wireless channels, since the associated decoding complexity is significantly lower than that of other approaches [74, 75]. Further research is needed for finding lower complexity solutions for space-time coded systems over wideband channels.

It was shown in Section 6.4.4 that the transmitted space-time coded signals will interfere with each other, if the fading envelope fluctuates rapidly. A certain minimum SIR has to be maintained in order to ensure that the system performs well. By employing turbo convolutional codes, the required SIR can be reduced enabling transmissions over rapidly fading channels. Alternative techniques have to be investigated, in order to reduce the required SIR of the system employing space-time block codes.

Iterative decoding could be applied for the decoding of space-time trellis codes. Channel codes, such as trellis coded modulation could be readily concatenated with space-time trellis codes, potentially enabling iterative decoding of the received sequence upon exchanging information between the trellis coded modulation and space-time trellis coding schemes. Besides, turbo space-time trellis codes could be constructed, allowing iterative decoding using two space-time trellis decoders.

List of Symbols

General notation

- The notation $\underline{y}$ represents a vector of y values.
- The notation $\underline{y}_{j<k}$ represents a range of values $y_0, y_1, ..., y_{k-1}$.
- The notation y_k represents the k-th element of the variable $\underline{y}$.
- The notation $|y|$ represents the magnitude of the variable y.
- The notation $|X|_j$ represents the positive integer remainder of the division X by j.
- The notation $\bar{h}$ represents the conjugate of h.
- The notation $\tilde{x}$ represents the estimate of the symbol x.
- $P(i \wedge j)$ represents the joint probability of i and j.
- $P(i|j)$ represents the conditional probability of i given j.

Special symbols

a:	Fading amplitude.
BM:	Branch metric.
C:	Constant.
$comp$:	Estimated complexity.
d_{min}:	Minimum free distance of a codeword.
$dist(i,j)$:	Euclidean distance between i and j.
E_b:	Energy per bit.
E_b/N_0:	Ratio of bit energy to noise power spectral density.
e:	Error pattern.
$f_c(\delta)$:	Correction term in the Log-MAP algorithm.
$g(x)$:	Octal generator polynomial.
h:	Channel impulse response.
i:	Index.
j:	Index.
K:	Constraint length of a convolutional code.
k:	Number of information data bits in a codeword.
L:	Multiplicative inverse in the context of the RRNS.
L_c:	Channel reliability value in Equation 3.8.
$L(u_k)$:	Intrinsic information.
$L_e(u_k)$:	Extrinsic information.
$L(u_k\|\underline{y})$:	A-Posteriori information.
M:	Dynamic range of the RNS.
M_{n-k}:	Product of moduli $m_{k+1}, m_{k+2}, ..., m_n$.
$M(\underline{s})$:	Probability of a trellis path.

m:	Modulus in the RNS.
max(x,y):	A function which return the maximum value x or y.
n:	Number of coded bits in a codeword.
P:	Probability.
p:	Number of transmitters in space-time codes.
q:	Number of receivers in space-time codes.
Rx:	Receiver.
r:	Residues in the RNS.
S:	A state in the trellis diagram.
SIR:	Signal to interference ratio.
t:	Error correcting capability.
T:	Time instant.
TP:	Test pattern in the Chase algorithm.
Tx:	Transmitter.
u:	Decoded data bit.
W:	Analogue weight in Equation 2.7.
X:	An integer in the RNS.
x:	Transmitted symbol.
y:	Received symbol.
α:	Forward recursion probability in Equation 3.25.
β:	Backward recursion probability in Equation 3.29.
δ:	Magnitude of the difference between two variables in Equation 3.45.
Δ:	Path metric difference.
$\gamma(\grave{s}, s)$:	Transition probability in Equation 3.23.
σ^2:	Noise variance.

$\oplus$:	Modulo addition.
$\otimes$:	Multiplication.

Glossary

16QAM	16-level Quadrature Amplitude Modulation
3G	Third Generation
4PSK	4-level Phase Shift Keying
64QAM	64-level Quadrature Amplitude Modulation
8PSK	8-level Phase Shift Keying
ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit Error Rate
BMA	Berlekamp-Massey Algorithm
BPS	Bits Per Symbol
BPSK	Binary Phase Shift Keying
BS	Base Station
CC	Convolutional Code
CD	Compact Disc
CIR	Channel Impulse Response
CRT	Chinese Remainder Theorem
DSP	Digital Signal Processing
DVB	Digital Video Broadcasting

FEC	Forward Error Correction
FER	Frame error rate
GSM	Global System of Mobile Communications Standard
ISI	Intersymbol Interference
LLR	Log Likelihood Ratio
LOS	Line Of Sight
MAP	Maximum A Posteriori
MDS-RRNS	Maximum Distance Separable Redundant Residue Number System
MIMO	Multi-Input Multi-Output
ML	Maximum Likelihood
MRC	Mixed Radix Conversion
MS	Mobile Station
OFDM	Orthogonal Frequency Division Multiplexing
PGZ	Peterson-Gorenstein-Zierler
QPSK	Quadrature Phase Shift Keying
RNS	Residue Number System
RRNS	Redundant Residue Number System
RS	Reed-Solomon
RSC	Recursive Systematic Convolutional
SISO	Soft Input Soft Output
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
STB	Space-Time Block
STT	Space-Time Trellis
TBCH	Turbo BCH

TC	Turbo Convolutional
TCM	Trellis Coded Modulation
TTCM	Turbo Trellis Coded Modulation
UMTS	Universal Mobile Telecommunication System
VA	Viterbi Algorithm
WATM	Wireless Asynchronous Transfer Mode
XOR	Exclusive-or

Bibliography

[1] C. Shannon, "A mathematical theory of communication," *The Bell system Technical Journal*, vol. 27, pp. 379–656, July 1948.

[2] R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, 1950.

[3] P. Elias, "Coding for noisy channels," *IRE Conv. Rec. pt. 4*, pp. 37–47, 1955.

[4] J. Wozencraft, "Sequential decoding for reliable communication," *IRE Natl. Conv. Rec.*, vol. 5, pt.2, pp. 11–25, 1957.

[5] J. Wozencraft and B. Reiffen, *Sequential Decoding.* Cambridge, MA, USA: MIT Presss, 1961.

[6] R. Fano, "A heuristic discussion of probabilistic coding," *IEEE Transactions on Information Theory*, vol. 9, pp. 64–74, April 1963.

[7] J. Massey, *Threshold Decoding.* Cambridge, MA, USA: MIT Press, 1963.

[8] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, April 1967.

[9] G. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–277, March 1973.

[10] J. Heller and I. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Transactions on Communication Technology*, vol. 19, pp. 835–848, October 1971.

[11] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

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

[13] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[14] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, September 1959.

[15] R. Bose and D. Ray-Chaudhuri, "On a class of error-correcting binary group codes," *Information and Control*, vol. 3, pp. 68–79, March 1960.

[16] R. Bose and D. Ray-Chaudhuri, "Further results on error correcting binary group codes," *Information and Control*, vol. 3, pp. 279–290, September 1960.

[17] W. Peterson, "Encoding and error-correction procedures for the Bose-Chaudhuri-Hocquenghem codes," *IRE Transactions on Information Theory*, vol. 6, pp. 459–470, 1960.

[18] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Transactions on Information Theory*, vol. 24, pp. 76–80, January 1978.

[19] B. Honary and G. Markarian, *Trellis Decoding of Block Codes.* Kluwer Academic Publishers Group, Distribution Centre, Post Office Box 322, 3300 AH Dordrecht, The Netherlands: Kluwer Academic Publishers, 1997.

[20] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and Trellis-based Decoding Algorithms for Linear Block Codes.* Norwell, Massachusetts 02061 USA: Kluwer Academic, 1998.

[21] G. Forney, "Coset codes-part II: Binary lattices and related codes," *IEEE Transactions on Information Theory*, vol. 34, pp. 1152–1187, September 1988.

[22] H. Manoukian and B. Honary, "BCJR trellis construction for binary linear block codes," *IEE Proc-Comms*, vol. 144, pp. 367–371, December 1997.

[23] B. Honary, G. Markarian, and P. Farrell, "Generalised array codes and their trellis structure," *Electronics Letters*, vol. 29, pp. 541–542, March 1993.

[24] B. Honary and G. Markarian, "Low-complexity trellis decoding of Hamming codes," *Electronics Letters*, vol. 29, pp. 1114–1116, June 1993.

[25] B. Honary, G. Markarian, and M. Darnell, "Low-complexity trellis decoding of linear block codes," *IEE Proceeding Communication*, vol. 142, pp. 201–209, August 1995.

[26] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On complexity of trellis structure of linear block codes," *IEEE Transactions on Information Theory*, vol. 39, pp. 1057–1937, May 1993.

[27] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On the optimum bit orders with respects to the state complexity of trellis diagrams for binary linear codes," *IEEE Transactions on Information Theory*, vol. 39, pp. 242–245, January 1993.

[28] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions of Information Theory*, vol. 18, pp. 170–182, January 1972.

[29] D. Gorenstein and N. Zierler, "A class of cyclic linear error-correcting codes in p^m symbols," *J. Soc. Ind. Appl. Math*, vol. 9, pp. 107–214, June 1961.

[30] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, pp. 300–304, June 1960.

[31] E. Berlekamp, "On decoding binary Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 11, pp. 577–579, 1965.

[32] E. Berlekamp, *Algebraic Coding Theory.* McGraw-Hill Book Company, 1968.

[33] J. Massey, "Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 11, pp. 580–585, 1965.

[34] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. 15, pp. 122–127, 1969.

[35] M. Oh and P. Sweeney, "Bit-level soft-decision sequential decoding for Reed Solomon codes," in *Workshop on Coding and Cryptography*, (Paris, France), January 1999.

[36] M. Oh and P. Sweeney, "Low complexity soft-decision sequential decoding using hybrid permutation for rs codes," in *Seventh IMA Conference on Cryptography and Coding*, (Royal Agricultural College, Cirencester, UK), December 1999.

[37] D. Burgess, S. Wesemeyer, and P. Sweeney, "Soft-decision decoding algorithms for RS codes," in *Seventh IMA Conference on Cryptography and Coding*, (Royal Agricultural College, Cirencester, UK), December 1999.

[38] Consultative Committee for Space Data Systems, *Blue Book: Recommendations for Space Data System Standards: Telemetry Channel Coding*, May 1984.

[39] European Telecommunication Standard Institute (ETSI), *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for MVDS at 10GHz and above*, ETS 300 748 ed., October 1996. http://www.etsi.org/.

[40] F. Taylor, "Residue arithmetic: A tutorial with examples." IEEE Computer Magazine, May 1984.

[41] N. Szabo and R. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology.* New York: McGraw-Hill Book Company, 1967.

[42] R. Watson and C. Hastings, "Self-checked computation using residue arithmetic," *Proceedings of the IEEE*, vol. 54, pp. 1920–1931, December 1966.

[43] H. Krishna, K. Lin, and J. Sun, "A coding theory approach to error control in redundant residue number systems - Part I: Theory and single error correction," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, pp. 8–17, January 1992.

[44] J. Sun and H. Krishna, "A coding theory approach to error control in redundant residue number systems - Part II: Multiple error detection and correction," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, pp. 18–34, January 1992.

[45] T. Liew, L. Yang, and L. Hanzo, "Soft-decision redundant residue number system based error correction coding," in *Proceedings of VTC 1999 Fall*, (Amsterdam, Holland), pp. 2546–2550, 19-22 September 1999.

[46] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets Part I: Introduction," *IEEE Communications Magazine*, vol. 25, pp. 5–11, February 1987.

[47] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets Part II: State of the art," *IEEE Communications Magazine*, vol. 25, pp. 12–21, February 1987.

[48] C. Schlegel, *Trellis Coding.* The Institute of Electrical and Electronics Engineers, Inc., New York: IEEE Press, 1997.

[49] R. Steele and L. Hanzo, *Mobile Radio Communications.* John Wiley & Son Ltd., 1999.

[50] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data distributed by time-varying inter-symbol interference," *IEEE Globecom*, pp. 1679–1684, December 1990.

[51] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures for ISI channels," *IEEE Transactions on Communications*, vol. 42, pp. 1661–1671, February/March/April 1994.

[52] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings of the International Conference on Communications*, pp. 1009–1013, June 1995.

[53] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *GLOBECOM '89*, (Dallas, Texas, USA), pp. 47.1.1–47.1.7, November 1989.

[54] J. Hagenauer, "Source-controlled channel decoding," *IEEE Transactions on Communications*, vol. 43, pp. 2449–2457, September 1995.

[55] S. Goff, A. Glavieux, and C. Berrou, "Turbo-codes and high spectral efficiency modulation," in *Proc IEEE International Conference on Communications*, (New Orleans and USA), pp. 645–649, May 1994.

[56] U. Wachsmann and J. Huber, "Power and bandwidth efficient digital communication using turbo codes in multilevel codes," *European Transactions on Telecommunications (ETT)*, vol. 6, pp. 557–567, Sept-Oct 1995.

[57] P. Robertson and T. Wörz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 206–218, February 1998.

[58] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Transactions on Communications*, vol. 44, pp. 591–600, May 1996.

[59] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 409–428, March 1996.

[60] L. Perez, J. Seghers, and D. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1698–1709, November 1996.

[61] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, March 1996.

[62] R. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Transactions on Communications*, vol. 46, pp. 1003–1010, August 1998.

[63] H. Nickl, J. Hagenauer, and F. Burkert, "Approaching Shannon's capacity limit by 0.27 dB using simple Hamming codes," *IEEE Communications Letters*, vol. 1, pp. 130–132, September 1997.

[64] O. Acikel and W. Ryan, "Punctured turbo-codes for BPSK/QPSK channels," *IEEE Transactions on Communications*, vol. 47, pp. 1315–1323, September 1999.

[65] P. Jung and M. Nasshan, "Performance evaluation of turbo codes for short frame transmission systems," *Electronics Letter*, vol. 30, pp. 111–113, January 1994.

[66] P. Jung, "Comparison of turbo-code decoders applied to short frame transmission systems," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 530–537, April 1996.

[67] M. N. P. Jung and J. Blanz, "Application of turbo-codes to a CDMA mobile radio system using joint detection and antenna diversity," in *Proceedings of IEEE Vehicular Technology Conference*, pp. 770–774, 1994.

[68] A. Barbulescu and S. Pietrobon, "Interleaver design for turbo codes," *Electronics Letters*, vol. 30, pp. 2107–2108, December 1994.

[69] B. Sklar, "A primer on turbo code concepts," *IEEE Communications Magazine*, vol. 35, pp. 94–101, December 1997.

[70] V. Tarokh, N. Seshadri, and A. Calderbank, "Space-time codes for high data rate wireless communication: Performance criterion and code construction," *IEEE Transactions on Information Theory*, vol. 44, pp. 744–765, March 1998.

[71] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 1451–1458, October 1998.

[72] V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Transactions on Information Theory*, vol. 45, pp. 1456–1467, July 1999.

[73] V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-time block coding for wireless communications: Performance results," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 451–460, March 1999.

[74] G. Bauch, A. Naguib, and N. Seshadri, "MAP equalization of space-time coded signals over frequency selective channels," in *Proceedings of Wireless Communications and Networking Conference*, (New Orleans, USA), September 1999.

[75] G. Bauch and N. Al-Dhahir, "Reduced-complexity turbo equalization with multiple transmit and receive antennas over multipath fading channels," in *Proceedings of Information Sciences and Systems*, (Princeton, USA), pp. WP3 13–18, March 2000.

[76] D. Agrawal, V. Tarokh, A. Naguib, and N. Seshadri, "Space-time coded OFDM for high data-rate wireless communication over wideband channels," in *Proceedings of IEEE Vehicular Technology Conference*, (Ottawa, Canada), pp. 2232–2236, May 1998.

[77] Y. Li, N. Seshadri, and S. Ariyavisitakul, "Channel estimation for OFDM systems with transmitter diversity in mobile wireless channels," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 461–471, March 1999.

[78] Y. Li, J. Chuang, and N. Sollenberger, "Transmitter diversity for OFDM systems and its impact on high-rate data wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1233–1243, July 1999.

[79] A. Naguib, N. Seshadri, and A. Calderbank, "Increasing data rate over wireless channels," *IEEE Signal Processing Magazine*, vol. 17, pp. 76–92, May 2000.

[80] A. Naguib, V. Tarokh, N. Seshadri, and A. Calderbank, "A space-time coding modem for high-data-rate wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 1459–1478, October 1998.

[81] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *GLOBECOM 94*, (San Francisco, California), pp. 339–343, November 1994.

[82] W. Peterson and E. Weldon Jr., *Error Correcting Codes.* Cambridge, MA, USA: MIT Press, 2nd ed., 1972.

[83] T. Kasami, *Combinational Mathematics and Its Applications.* University of North Carolina Press, 1969.

[84] W. Peterson, *Error Correcting Codes.* Cambridge, MA, USA: MIT Press, 1st ed., 1961.

[85] F. Macwilliams and N. Sloane, *The Theory of Error Correcting Codes*, vol. 16. Bell Laboratories, Murray Hill, NJ 07974 USA: North-Holland publishing company, 1978.

[86] B. Sklar, *Digital Communications Fundamentals and Applications.* Prentice Hall, Englewood Cliffs: Prentice-Hall International Editions, 1988.

[87] I. Blake, *Algebraic Coding Theory: History and Development.* Dowden, Hutchinson and Ross, 1973.

[88] G. Clark Jr. and J. Cain, *Error Correction Coding for Digital Communications.* New York, USA: Plenum Press, 1981. ISBN: 0306406152.

[89] V. Pless, *Introduction to the Theory of Error-correcting Codes.* New York, USA: John Wiley and Sons, 1982.

[90] R. Blahut, *Theory and Practice of Error Control Codes.* Reading, MA, USA: Addison-Wesley, 1983. ISBN 0-201-10102-5.

[91] C. Shannon, *Mathematical Theory of Communication.* University of Illinois Press, 1963.

[92] C. Heegard and S. Wicker, *Turbo Coding.* The Netherlands: Kluwer Academic Publishers, 1999.

[93] M. Bossert, *Channel Coding for Telecommunications.* New York, USA: John Wiley and Sons, 1999. ISBN 0-471-98277-6.

[94] B. Vucetic and J. Yuan, *Turbo Codes Principles and Applications.* The Netherlands: Kluwer Academic Publishers, 2000.

[95] R. Lidl and H. Niederreiter, *Finite Fields.* Addison Wesley, 1983.

[96] S. Lin and D. Costello Jr, *Error Control Coding: Fundamentals and Applications.* Inc. Englewood Cliffs, New Jersey 07632: Prentice-Hall, 1983.

[97] A. Michelson and A. Levesque, *Error Control Techniques for Digital Communication.* New York, USA: John Wiley and Sons, 1985.

[98] D. Hoffman, D. Leonard, C. Lindner, K. Phelps, C. Rodger, and J. Wall, *Coding Theory.* New York, USA: Marcel Dekker, Inc., 1991.

[99] J. Huber, *Trelliscodierung.* Berlin: Springer Verlag, 1992.

[100] J. Anderson and S. Mohan, *Source and Channel Coding — An Algorithmic Approach.* Dordrech: Kluwer Academic Publishers, 1993.

[101] S. Wicker, *Error control systems for digital Communication and storage.* Englewood Cliffs, NJ, USA: Prentice-Hall, 1994.

[102] J. Proakis, *Digital Communications.* McGraw-Hill, Inc: McGraw-Hill International Editions, 1995.

[103] P. Sweeney, *Error Control Coding: An Introduction.* New York, USA: Prentice Hall, 1991.

[104] M. Simon and M.-S. Alouini, *Digital Communication over Fading Channels: A Unified Approach to Performance Analysis.* New York, USA: John Wiley and Sons, 2000. ISBN 0471317799.

[105] M. Nakagami, "The m-distribution-a general formula of intensity distribution of fading," *Statistical Methods in Radio Wave Propagation*, 1960.

[106] T. Liew, B. Yeap, J. Woodard, and L. Hanzo, "Modified MAP algorithm for extended turbo BCH codes and turbo equalisers," in *3G Mobile Communications Technologies*, (London, UK), pp. 185–189, 27-29 March 1999.

[107] T. Liew, L. Yang, and L. Hanzo, "Turbo decoded redundant residue number system codes," in *Proceedings of VTC 2000 Spring*, (Tokyo, Japan), pp. 576–580, 15-18 May 2000.

[108] T. Liew, J. Pliquett, B. Yeap, L.-L. Yang, and L. Hanzo, "Comparative study of space time block codes and various concatenated turbo coding schemes," in *PIMRC 2000*, (London, UK), pp. 741–745, 18-21 September 2000.

[109] T. Liew, B. Choi, and L. Hanzo, "Comparative study of concatenated turbo coded and space-time block coded as well as space-time trellis coded OFDM," submitted to VTC 2001 Spring.

[110] J. Stenbit, "Table of generators for Bose-Chaudhuri codes," *IEEE Transactions on Information Theory*, vol. 10, pp. 390–391, October 1964.

[111] B. Sklar, *Digital Communications Fundamentals and Applications*, ch. 5.6.5, pp. 294–296. Prentice Hall, Englewood Cliffs: Prentice-Hall International Editions, 1988.

[112] R. Steele and L. Hanzo, eds., *Mobile Radio Communications*, ch. 4.4.4, pp. 425–428. IEEE Press, 445 Hoes Lane, Piscataway, NJ, 08855, USA: IEEE Press and Pentech Press, 1992.

[113] R. Blahut, *Theory and Practice of Error Control Codes*, ch. 6, pp. 130–160. IBM Corporation, Owego, NY 13827, USA: Addison-Wesley Publishing Company, 1983.

[114] G. Forney, "On decoding binary Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 11, pp. 549–557, 1965.

[115] P. Adde, R. Pyndiah, O. Raoul, and J. Inisan, "Block turbo decoder design," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 166–169, September 1997.

[116] A. Goalic and R. Pyndiah, "Real-time turbo-decoding of product codes on a digital signal processor," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 267–270, September 1997.

[117] R. Pyndiah, "Iterative decoding of product codes: Block turbo codes," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 71–79, September 1997.

[118] F. Macwilliams and N. Sloane, *The theory of error correcting codes*, vol. 16, ch. 18, pp. 567–580. Bell Laboratories, Murray Hill, NJ 07974 USA: North-Holland publishing company, 1978.

[119] S. Ng, T. Liew, L. Yang, and L. Hanzo, "Turbo coding performance using block component codes," in *Proceedings of VTC 2000 Spring*, (Tokyo, Japan), pp. 849–853, 15-18 May 2000.

[120] C. Hastings, "Automatic detection and correction of errors in digital computers using residue arithmetic," in *Proceeding of IEEE Conference*, (Region Six), pp. 465–490, 1966.

[121] R. Watson and C. Hastings, *Residue Arithmetic and Reliable Computer Design*. Washington, D. C.: Spartan Books, 1967.

[122] Y. Keir, P. Cheney, and M. Tannenbaum, "Division and overflow detection in residue number systems," *IRE Transactions on Electronic Computers*, vol. EC-11, pp. 501–507, August 1962.

[123] A. Baraniecka and G. Jullien, "On decoding techniques for residue number system realizations of digital signal processing hardware," *IEEE Transactions on Circuits System*, vol. CAS-25, November 1978.

[124] W. Jenkins, "A new algorithm for scaling in residue number systems with applications to recursive digital filtering," in *IEEE International Symposium on Circuits and Systems*, (Phoenix, Arizona), pp. 56–59, 1977.

[125] H. Huang and F. Taylor, "High speed DFT's using residue numbers," in *IEEE International Conference on Acoustics, Speech, Signal Processing*, (Denver, CO), pp. 238–242, April 1980.

[126] W. Jenkins and B. Leon, "The use of residue number system in the design of finite impulse response filters," *IEEE Transactions on Circuits Systems*, vol. CAS-24, pp. 191–201, April 1977.

[127] C. Su and H. Lo, "An algorithm for scaling and single residue error correction in residue number system," *IEEE Transactions on Computers*, vol. 39, pp. 1053–1064, August 1990.

[128] F. Taylor and C. Huang, "An autoscale residue multiplier," *IEEE Transactions on Computers*, vol. 31, pp. 321–325, April 1982.

[129] G. Jullien, "Residue number scaling and other operations using ROM arrays," *IEEE Transactions on Computers*, vol. C-27, pp. 325–337, April 1978.

[130] W. Jenkins, "A highly efficient residue-combinatorial architecture for digital filters," *Proceeding of the IEEE*, vol. 66, pp. 700–702, June 1978.

[131] F. Taylor and A. Ramnarayanan, "An efficient residue-to-decimal converter," *IEEE Transactions on Circuits and Systems*, vol. 28, pp. 1164–1169, December 1981.

[132] A. Shenoy and R. Kumaresan, "Residue to binary conversion for RNS arithmetic using only modular look-up table," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1158–1162, September 1988.

[133] F. Barsi and P. Maestrini, "Error correcting properties of redundant residue number systems," *IEEE Transactions on Computers*, vol. C-22, pp. 307–315, March 1973.

[134] S. Yau and Y. Liu, "Error correction in redundant residue number systems," *IEEE Transactions on Computers*, vol. C-22, pp. 5–11, January 1984.

[135] M. Etzel and W. Jenkins, "Redundant residue number systems for error detection and correction in digital filters," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-28, pp. 538–544, October 1980.

[136] C. Huang, D. Peterson, H. Rauch, J. Teague, and D. Fraser, "Implementation of a fast digital processor using the residue number system," *IEEE Transactions on Circuits Systems*, vol. CAS-28, pp. 32–38, January 1981.

[137] W. Jenkins, "Recent advances in residue number system techniques for recursive digital filtering," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-27, pp. 19–30, February 1979.

[138] M. Soderstrand, "A high-speed, low-cost, recursive digital filter using residue number arithmetic," *Proceeding IEEE*, vol. 65, pp. 1065–1067, July 1977.

[139] M. Soderstrand, W. Jenkins, and G. Jullien, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing.* New York: IEEE Press, 1986.

[140] L. Yang and L. Hanzo, "Residue number system arithmetic assisted M-ary modulation," *IEEE Communications Letters*, vol. 3, pp. 28–30, February 1998.

[141] A. Baraniecka and G. Jullien, "Residue number system implementations of number theoretic transforms," *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. ASSP-28, pp. 285–291, June 1980.

[142] M. Soderstrand and E. Fields, "Multipliers for residue number arithmetic digital filters," *Electronic Letters*, vol. 13, pp. 164–166, March 1977.

[143] D. Mandelbaum, "Error correction in residue arithmetic," *IEEE Transactions on Computers*, vol. C-21, pp. 538–543, June 1972.

[144] W. Jenkins and E. Altman, "Self-checking properties of residue number error checkers based on mixed radix conversion," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 159–167, February 1988.

[145] W. Jenkins, "The design of error checkers for self-checking residue number arithmetic," *IEEE Transactions on Computers*, vol. 32, pp. 388–396, April 1983.

[146] O. Aitsab and R. Pyndiah, "Performance of Reed-Solomon block turbo code," in *GLOBECOM '96*, (London, U.K.), pp. 121–125, November 1996.

[147] O. Aitsab and R. Pyndiah, "Performance of concatenated Reed-Solomon/Convolutional codes with iterative decoding," in *GLOBECOM '97*, (New York, USA), pp. 644–648, 1997.

[148] R. Pyndiah, P. Combelles, and P. Adde, "A very low complexity block turbo decoder for product codes," in *GLOBECOM '96*, (London, U.K.), pp. 101–105, November 1996.

[149] S. Lin, D. Constello Jr., and M. Miller, "Automatic-repeat-request error-control schemes," *IEEE Communications Magazine*, vol. 22, pp. 5–17, December 1984.

[150] T. Keller, T. Liew, and L. Hanzo, "Adaptive rate RRNS coded OFDM transmission for mobile communication channels," in *Proceedings of VTC 2000 Spring*, (Tokyo, Japan), pp. 230–234, 15-18 May 2000.

[151] T. Keller, T. Liew, and L. Hanzo, "Adaptive redundant residue number system coded multicarrier modulation," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2292–2301, November 2000.

[152] T. James, "Study into redundant residue number system codes," tech. rep., University of Southampton, May 1999.

[153] L. Hanzo, W. Webb, and T. Keller, eds., *Single- and Multi-carrier Quadrature Amplitude Modulation.* Wessex, England: John Wiley & Sons, Ltd, 3rd ed., 2000. ISBN 0471492396.

[154] P. Chaudhury, "The 3GPP proposal for IMT-2000," *IEEE Communications Magazine*, vol. 37, pp. 72–81, December 1999.

[155] G. Foschini Jr. and M. Gans, "On limits of wireless communication in a fading environment when using multiple antennas," *Wireless Personal Communications*, vol. 6, pp. 311–335, March 1998.

[156] B. Glance and L. Greestein, "Frequency-selective fading effects in digital mobile radio with diversity combining," *IEEE Transactions Communications*, vol. COM-31, pp. 1085–1094, September 1983.

[157] F. Adachi and K. Ohno, "BER performance of QDPSK with postdetection diversity reception in mobile radio channels," *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 237–249, February 1991.

[158] H. Zhou, R. Deng, and T. Tjhung, "Performance of combined diversity reception and convolutional coding for QDPSK land mobile radio," *IEEE Transactions on Vehicular Technology*, vol. 43, pp. 499–508, August 1994.

[159] J. Winters, "Switched diversity with feedback for DPSK mobile radio systems," *IEEE Transactions on Information Theory*, vol. 32, pp. 134–150, February 1983.

[160] G. Raleigh and J. Cioffi, "Spatio-temporal coding for wireless communications," in *GLOBECOM '96*, (London, U.K.), pp. 533–537, November 1996.

[161] A. Wittneben, "Base station modulation diversity for digital SIMULCAST," in *Proceedings of IEEE Vehicular Technology Conference*, pp. 505–511, May 1993.

[162] N. Seshadri and J. Winters, "Two signalling schemes for improving the error performance of frequency-division-duplex (FDD) transmission systems using transmitter antenna diversity," *International Journal of Wireless Information Networks*, vol. 1, pp. 49–60, January 1994.

[163] J. Winters, "The diversity gain of transmit diversity in wireless systems with Rayleigh fading," *IEEE Transactions on Vehicular Technology*, vol. 47, pp. 119–123, February 1998.

[164] T. Hattori and K. Hirade, "Multitransmitter simulcast digital signal transmission by using frequency offset strategy in land mobile radio-telephone system," *IEEE Transactions on Vehicular Technology*, vol. 27, pp. 231–238, 1978.

[165] A. Hiroike, F. Adachi, and N. Nakajima, "Combined effects of phase sweeping transmitter diversity and channel coding," *IEEE Transactions on Vehicular Technology*, vol. 41, pp. 170–176, May 1992.

[166] N. Seshadri, V. Tarokh, and A. Calderbank, "Space-time codes for high data rate wireless communications: Code construction," in *Proceedings of IEEE Vehicular Technology Conference '97*, (Phoenix, Arizona), pp. 637–641, 1997.

[167] V. Tarokh, N. Seshadri, and A. Calderbank, "Space-time codes for high data rate wireless communications: Performance criterion and code construction," in *Proc IEEE International Conference on Communications '97*, (Montreal, Canada), pp. 299–303, 1997.

[168] V. Tarokh, A. Naguib, N. Seshadri, and A. Calderbank, "Space-time codes for high data rate wireless communications: Mismatch analysis," in *Proc IEEE International Conference on Communications '97*, (Montreal, Canada), pp. 309–313, 1997.

[169] V. Tarokh, A. Naguib, N. Seshadri, and A. Calderbank, "Space-time codes for high data rate wireless communication: Performance criteria in the presence of channel estimation errors, mobility, and multiple paths," *IEEE Transactions on Communications*, vol. 47, pp. 199–207, February 1999.

[170] D. Brennan, "Linear diversity combining techniques," *Proc. IRE*, vol. 47, pp. 1075–1102, 1959.

[171] G. Bauch, "Concatenation of space-time block codes and Turbo-TCM," in *Proceedings of IEEE International Conference on Communications*, (Vancouver, Canada), pp. 1202–1206, June 1999.

[172] G. Forney, "Convolutional codes: I. Algebraic structure," *IEEE Transactions on Information Theory*, vol. 16, pp. 720–738, November 1970.

[173] G. Forney, "Burst-correcting codes for the classic burst channel," *IEEE Transactions on Communication Technology*, vol. 19, pp. 772–781, October 1971.

[174] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, pp. 389–400, April 1988.

[175] S. Red, M. Oliphant, and M. Weber, *An Introduction to GSM.* Artech House, 1995.

[176] 3GPP, *Multiplexing and channel coding (TDD).* 3G TS 25.222, http://www.3gpp.org.

[177] T. Ojanperä and R. Prasad, *Wideband CDMA for Third Generation Mobile Communications.* London, UK: Artech House, 1998.

[178] S. Al-Semari and T. Fuja, "I-Q TCM: Reliable communication over the rayleigh fading channel close to the cutoff rate," *IEEE Transactions on Information Theory*, vol. 43, pp. 250–262, January 1997.

[179] R. Horn and C. Johnson, *Matrix Analysis.* New York: Cambridge University Press, 1988.

[180] W.-J. Choi and J. Cioffi, "Space-time block codes over frequency selective fading channels," in *Proceedings of VTC 1999 Fall*, (Amsterdam, Holland), pp. 2541–2545, 19-22 September 1999.

[181] Z. Liu, G. Giannakis, A. Scaglione, and S. Barbarossa, "Block precoding and transmit-antenna diversity for decoding and equalization of unknown multipath channels," in *Proc 33rd Asilomar Conference Signals, Systems and Computers*, (Pacific Grove, Canada), pp. 1557–1561, 1-4 November 1999.

[182] Z. Liu and G. Giannakis, "Space-time coding with transmit antennas for multiple access regardless of frequency-selective multipath," in *Proc 1st Sensor Array and Multichannel SP Workshop*, (Boston, USA), 15-17 March 2000.

[183] T. Liew, J. Pliquett, B. Yeap, L.-L. Yang, and L. Hanzo, "Concatenated space time block codes and TCM, turbo TCM, convolutional as well as turbo codes," in *GLOBECOM 2000*, (San Francisco, USA), pp. 2292–2301, 27 Nov -1 Dec 2000.

[184] W. Jakes, *Microwave Mobile Communications.* Piscataway: IEEE Press, 1993.

[185] R. Steele and W. Webb, "Variable rate QAM for data transmission over Rayleigh fading channels," in *Proceedings of Wireless '91*, (Calgary, Alberta), pp. 1–14, 1991.

[186] W. Webb and R. Steele, "Variable rate QAM for mobile radio," *IEEE Transactions on Communications*, vol. 43, pp. 2223–2230, July 1995.

[187] J. Torrance and L. Hanzo, "Latency and networking aspects of adaptive modems over slow indoors Rayleigh fading channels," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 4, pp. 1237–1251, 1998.

[188] J. Torrance and L. Hanzo, "Performance upper bound of adaptive QAM in slow Rayleigh fading environments," in *ISPACS '96*, (Singapore), pp. 1653–1657, November 1996.

[189] J. Torrance and L. Hanzo, "Interference aspects of adaptive modems over slow Rayleigh fading channels," *IEEE Transactions on Vehicular Technology*, vol. 48, pp. 1527–1545, September 1999.

[190] T. Keller and L. Hanzo, "Adaptive orthogonal frequency division multiplexing schemes," in *Proceedings of ACTS Mobile Communications Summit '98*, (Rhodos, Greece), pp. 794–799, June 1998.

[191] T. Keller and L. Hanzo, "Blind-detection assisted sub-band adaptive turbo-coded OFDM schemes," in *Proceedings of VTC '99*, (Houston, USA), pp. 127–130, March 1999.

[192] H. Matsuako, S. Sampei, N. Morinaga, and Y. Kamio, "Adaptive modulation systems with variable coding rate concatenated code for high quality multi-media communication systems," in *Proceedings of IEEE Vehicular Technology Conference*, (Atlanta, USA), pp. 487–491, April 1996.

[193] S.-G. Chua and A. Goldsmith, "Variable-rate variable-power mQAM for fading channels," in *Proceedings of IEEE VTC '96*, (Atlanta, GA, USA), pp. 815–819, 1996.

[194] V. Lau and M. Macleod, "Variable rate trellis coded QAM for high bandwidth efficiency applications in Rayleigh fading channels," in *Proceedings of IEEE VTC '98*, (Ottawa, Canada), pp. 348–352, May 1998.

[195] I. Kalet, "The multitone channel," *IEEE Transactions on Communications*, vol. 37, pp. 119–124, February 1989.

[196] J. Torrance and L. Hanzo, "Optimization of switching levels for adaptive modulation in a slow Rayleigh fading channel," *IEE Electronic Letters*, pp. 1167 – 1169, June 1996.

[197] J. Torrance and L. Hanzo, "On the upper bound performance of adaptive QAM in a slow Rayleigh fading," *IEE Electronic Letters*, pp. 169–171, April 1996.

Author Index

Index