UNIVERSITY OF SOUTHAMPTON
FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Iterative Source Decoding, Channel Decoding and Channel Equalisation

by

Jin Wang
BEng, MEng

A doctoral thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy at the University of Southampton

SUPERVISORS:
Professor Lajos Hanzo
Dip Ing, MSc, PhD, FREng, FIEEE, FIEE, DSc
Chair of Telecommunications
Dr. Lie-Liang Yang
BEng, MEng, PhD, SIEEE

Southampton, SO17 1BJ
United Kingdom
January 2007

This thesis is dedicated to:

my parents and my wife

with all my love and respect...

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE

SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Iterative Source Decoding, Channel Decoding and Channel Equalisation

by

Jin Wang

In this thesis, we apply EXtrinsic Information Transfer (EXIT) charts for analysing iterative decoding and detection schemes. We first consider a two-stage iterative source/channel decoding scheme, which is then extended to a three-stage serially concatenated turbo equalisation scheme. As a result, useful design guidelines are obtained for optimising the system's performance.

More explicitly, in Chapter 2 various source codes such as Huffman codes, Reversible Variable-Length Codes (RVLCs) and Variable-Length Error-Correcting (VLEC) codes are introduced and a novel algorithm is proposed for the construction of efficient RVLCs and VLEC codes. In Chapter 3, various iterative source decoding, channel decoding and channel equalisation schemes are investigated and their convergence behaviour is analysed by using EXIT charts. The effects of different source codes and channel precoders are also considered. In Chapter 4, a three-stage serially concatenated turbo equalisation scheme consisting of an inner channel equaliser, an intermediate channel code and an outer channel code separated by interleavers is proposed. With the aid of the novel concept of EXIT modules, conventional 3D EXIT chart analysis may be simplified to 2D EXIT chart analysis. Interestingly, it is observed that for the three-stage scheme relatively weak convolutional codes having short memories result in lower convergence thresholds than strong codes having long memories. Additionally, it is found that by invoking the outer and the intermediate decoder more frequently the total number of decoder activations is reduced, resulting in a relatively lower decoding complexity. Furthermore, the three-stage turbo equalisation schemes employing non-unity rate intermediate codes or IRregular Convolutional Codes (IRCCs) as the outer constituent codes are investigated. The performance of the resultant schemes is found to become gradually closer to the channel's capacity at the expense of the increase of decoding complexity.

Contents

Acknowledgements

First of all, I would like to thank my supervisor Professor Lajos Hanzo from the bottom of my heart for his substantial assistance to me. His efficient guidance, insightful inspiration and unflagging support have significantly benefitted me not only in my work but also in my personal life. His loyalty to research has exhibited the right way to be an excellent researcher, which will continue to stimulate me to make progress in my future career. Meanwhile, I am also very grateful to my supervisor Dr. Lie-Liang Yang for his generous guidance and invaluable encouragement. His remarkable range of reading and mathematical thinking exhibit the characteristics that a successful researcher should have. All in all, this thesis would never have gone this far without them.

I would also like to thank all my colleagues in the Communications Group of University of Southampton, both past and present, for all their supports and helps throughout my PhD study. Special thanks to Dr. Soon Xin Ng for his guidance on my first paper and for his continuous assistance in the following projects. I am also indebted to Wei Liu, Bee Leong Yeap, Feng Guo, Hua Wei, Bin Hu, Ming Jiang, Soog Ni, Jin Yee Chung, Xiang Liu, Choo L Koh (Augustine), Kai-Wen Lien (Kevin), Ronald Tee, Rob Maunder, Andreas Wolfgang, Osamah Alamri and Yosef (Jos) Akhtman for their wonderful friendship. I want to say a big thank you to Denise Harvey for providing us practical day-to-day assistance.

The financial support of the British ORS award scheme and the University of Southampton is gratefully acknowledged.

I would also like to thank my parents and my sister in China for their love and support. Finally, to my beloved wife, Lili, for her love, care and support.

Publication List

Journal Papers

1) R. G. Maunder, **J. Wang**, S. X. Ng, L.-L. Yang and L. Hanzo, "Iteratively Decoded Irregular Variable Length Coding and Trellis Coded Modulation", submitted to IEEE Transactions on Wireless Communications.

2) O. Alamri, **J. Wang**, S. X. Ng, L.-L. Yang and L. Hanzo, "Near-Capacity Three-Stage Turbo Detection of Irregular Convolutional Coded Joint Sphere-Packing Modulation and Space-Time Coding", submitted to IEEE Transactions on Communications.

3) S. X. Ng, **J. Wang**, M. Tao, L.-L. Yang and L. Hanzo, "Iteratively Decoded Variable Length Space-Time Coded Modulation: Code Construction and Convergence Analysis", accepted for IEEE Transactions on Wireless Communications.

4) R. G. Maunder, J. Kliewer, S. X. Ng, **J. Wang**, L.-L. Yang and L. Hanzo, "A Wireless Video Scheme Employing the Joint Iterative Decoding of Trellis-Based Vector Quantization and Trellis-Coded Modulation", accepted for IEEE Transactions on Wireless Communications.

5) **J. Wang**, L-L. Yang, and L. Hanzo, "Iterative Construction of Reversible Variable Length Codes and Variable Length Error Correcting Codes," IEEE Communications Letters 8(11):pp. 671-673, 2004.

6) S. X. Ng , F. Guo, **J. Wang**, L-L. Yang, and L. Hanzo, "Joint source-Coding, Channel Coding and Modulation Schemes for AWGN and Rayleigh Fading Channels," IEE Electronic Letters 39(17):pp. 1259-1261, 2003.

Conference Papers

1) O. Alamri, **J. Wang**, S. X. Ng, L.-L. Yang and L. Hanzo, "Near-Capacity Transceiver Design Using EXIT-Curve Fitting: Three-Stage Turbo Detection of Irregular Convolutional Coded Joint Sphere-Packing Modulation and Space-Time Coding", accepted for IEEE International Conference on Communications (ICC), Glasgow, Scotland, June 2007.

2) S. X. Ng, Wei Liu, **J. Wang**, M. Tao, L.-L. Yang and L. Hanzo, "Performance Analysis of Iteratively Decoded Variable-Length Space-Time Coded Modulation", accepted for IEEE International Conference on Communications (ICC), Glasgow, Scotland, June 2007.

3) A.Q. Pham, **J. Wang**, L.-L. Yang and L. Hanzo, “An Iterative Detection Aided Irregular Convolutional Coded Wavelet Videophone Scheme Using Reversible Variable-Length Codes and Map Equalization“, accepted for 65th IEEE Vehicular Technology Conference (VTC), Dublin, Ireland, April 2007.

4) **J. Wang**, L.-L. Yang and L. Hanzo, ”Combined Serially Concatenated Codes and MMSE Equalisation: an EXIT Chart Aided Perspective”, In Proceedings of the 64th IEEE Vehicular Technology Conference (VTC), Montréal, Canada, 25-28 September, 2006 [CDROM].

5) L.-L. Yang, **J. Wang**, R. Maunder and L. Hanzo, ”Iterative Channel Equalization and Source Decoding for Vector Quantization Sources”, In Proceedings of the 63rd IEEE Vehicular Technology Conference, Melbourne, Austria, pp. 2349 - 2353, 7-10 May, 2006.

6) A.Q. Pham, **J. Wang**, L.-L. Yang and L. Hanzo, ”An Iterative Detection Aided Unequal Error Protection Wavelet Video Scheme Using Irregular Convolutional Codes”, In Proceedings of the 63rd IEEE Vehicular Technology Conference (VTC), Melbourne, Austria, pp. 2484 - 2488, 7-10 May, 2006.

7) **J. Wang**, S. X. Ng, A. Wolfgang, L.-L. Yang, S. Chen and L. Hanzo, ”Near-Capacity Three-Stage MMSE Turbo Equalization Using Irregular Convolutional Codes”, In the 4th International Symposium on Turbo Codes (ISTC’06) in connection with the 6th International ITG-Conference on Source and Channel Coding, Munich, Germany, 3-7 April 2006.

8) **J. Wang**, N. Othman, J. Kliewer, L-L. Yang, and L. Hanzo, ”Turbo-Detected Unequal Error Protection Irregular Convolutional Codes Designed for the Wideband Advanced Multi-rate Speech Codec,” In Proceedings of the 62nd IEEE Vehicular Technology Conference (VTC), Dallas, TX, USA, pp. 927 - 931, September, 2005.

9) S. X. Ng, **J. Wang**, L-L. Yang and L. Hanzo, ”Variable Length Space Time Coded Modulation”, In Proceedings of the 62nd IEEE Vehicular Technology Conference (VTC), Dallas, TX,USA, pp. 1049 - 1053, September, 2005.

10) R. Maunder, J. Kliewer, S. X. Ng, **J. Wang**, L-L. Yang and L. Hanzo, ”Iterative Joint Video and Channel Decoding in a Trellis-Based Vector-Quantized Video Codec and Trellis-Coded Modulation Aided Wireless Videophone,” In Proceedings of the 62nd IEEE Vehicular Technology Conference (VTC), Dallas, TX, USA, pp. 922 - 926, September, 2005.

11) **J. Wang**, L-L. Yang, L. Hanzo "Iterative channel equalization, channel decoding and source decoding," In Proceedings of the 61st IEEE Vehicular Technology Conference (VTC), Sweden, pp. 518 - 522, May, 2005.

12) S. X. Ng, R. G. Maunder, **J. Wang**, L-L. Yang and L. Hanzo, " Joint Iterative-Detection of Reversible Variable-Length Coded Constant Bit Rate Vector-Quantized Video and Coded Modulation ", Invited paper, In Proceedings the European Signal Processing Conference (EUSIPCO), Vienna, Austria, September, 2004 [CDROM].

13) S. X. Ng, F. Guo, **J. Wang**, L-L. Yang, and L. Hanzo, "Jointly Optimised Iterative Source-coding, Channel-coding and Modulation for Transmission over Wireless Channels," In Proceedings of the 59th IEEE Vehicular Technology Conference (VTC), pp. 313-317, Milan, Italy, May, 2004.

Chapter 1

Introduction

1.1 Historical Overview

Multimedia transmission over time-varying wireless channels presents a number of challenges beyond the existing capabilities of the third-generation (3G) networks. The prevalent design principles stemming from Shannon's source and channel separation theorem [6] have to be reconsidered. The separation theorem, stating that optimal performance bounds of source and channel decoding can be approached as close as desired by independently designing source and channel coding, holds only under asymptotic conditions, where both codes may have a length and complexity tending to infinity, and under the conditions of stationary sources. In practice, the design of the system is heavily constrained in terms of both complexity and delay, hence source and channel codecs are typically suboptimal [7]. Fortunately, the "lossy" nature of the audio and video codecs may be concealed with the aid of perceptual masking by exploiting the psycho-acoustic and psycho-visual properties of the human ear and eye. The output of practical source encoders, such as speech codecs [8] and image/video codecs [9], may contain a significant amount of redundancy. Moreover, having a non-zero residual channel error rate may lead to dramatically increased source symbol error rates [7]. In these circumstances, the assumptions of the Shannon's separation theorem no longer hold, neither can they be used as a good approximation. One may potentially improve the attainable performance by considering the source and channel code designs jointly, allowing us to exploit both the residual redundancy of the source coded stream and the redundancy introduced by the channel code, or optimally allocate redundancy (in the best form) within the source and channel coding chain [7, 10–12].

1.1.1 Source Coding and Decoding

1.1.1.1 Variable-Length Coding

Source coding constitutes a mapping from a sequence of symbols of an information source to a sequence of codewords (usually bits), so that the source symbols can be exactly recovered from the binary bits (lossless source coding) or recovered within some distortion (lossy source coding). Claude Shannon initiated this subject (along with other subjects in the foundations of information theory) in his ground-breaking paper [6]. The key concept of entropy was formulated here. Shannon then filled the concept of entropy with practical significance by proving his noiseless source coding theorem. Shannon's theorem ([6], Theorem 9) states that the entropy rate of a stationary ergodic finite-alphabet Markov source is the optimum rate at which the source can be encoded using "almost noiseless" fixed-rate block codes or using noiseless variable-rate block codes. To prove the "fixed-rate half" of the theorem, Shannon constructed fixed-rate block codes using an auxiliary theorem which later became known as the Shannon-McMillan Theorem [13]. In the proof of the "variable-rate half" of the theorem, Shannon devised a noiseless variable-rate block code (nowadays referred to as a Shannon-Fano code) which encodes a source symbol having a probability of occurrence p into a codeword having a length within one bit of $-log_2p$. The Shannon-Fano coding algorithm is capable of producing fairly efficient variable-length prefix codes based on a set of symbols and their probabilities (estimated or measured), but it is nonetheless suboptimal. As a remedy, Huffman coding [14] named after its inventor, D. A. Huffman, is always capable of producing optimal prefix codes, which approach the lower bound of the expected codeword length under the constraint that each symbol is represented by a codeword formed of an integer number of bits. This constraint, however, is often unnecesary, since the codewords will be packed end-to-end in long sequences. If we consider groups of codes at a time, symbol-by-symbol Huffman coding is only optimal if the probabilities of the symbols are independent and assume a value of $1/2^n$, where n is an integer. In most situations, arithmetic coding [15] is capable of producing a higher overall compression than either Huffman or Shannon-Fano coding, since it is capable of encoding in terms of fractional non-integer numbers of bits which more closely approximate the actual information content of the symbol. However, arithmetic coding has not superseded Huffman coding, as Huffman coding superseded Shannon-Fano coding, partly because arithmetic coding is more computationally expensive and partly because it is covered by multiple patents. An excellent tutorial on the history of source coding can be found in [16].

In addition to coding efficiency, the robustness of the variable length codes (VLCs) to transmission errors is another design concern. Takishima *et al.* [17] proposed Reversible Variable-Length

Codes (RVLCs) for providing both forward and backward decoding capability so that the correct data can be recovered with as high a probability as possible from the received data, when transmission errors occurred. For example, a decoder can commence by processing the received bitstream in the forward direction and upon detecting an error, proceed immediately to the end of the bitstream and decode in the reverse direction. This technique and a range of other related strategies can be used for significantly reducing the effects of bit errors on the fidelity of the decoded data. Wen and Villasenor [18, 19] proposed a new class of asymmetric RVLCs having the same codeword length distribution as the Golomb-Rice codes [20, 21] as well as exp-Golomb codes [22] and applied them to the video coding framework of the H.263+ [23] and MPEG-4 [24] standards. Later on, in the quest for constructing optimal RVLCs, further improved algorithms were proposed in [25–29]. More particularly, Lakovic and Villasenor [27] designed a family of new RVLCs having a free distance of $d_f = 2$, which exhibit a better error correcting performance than the RVLCs and Huffman codes having a lower free distance. The conditions for the existence of RVLCs and their properties were also studied in [30–32]. For the sake of attaining an even stronger error correction capability, Buttigieg and Farrell [33, 34] proposed a new class of Variable-Length Error-Correcting (VLEC) codes, which generally have higher free distances ($d_f > 2$). It was shown in [34] that the free distance is the most important parameter that determines the decoding performance at high E_b/N_0 values. Detailed treatment of various VLEC code properties, their construction, decoding and performance bounds can be found in Buttigieg's PhD thesis [35]. Lamy and Paccaut [36] also investigated a range of low-complexity construction techniques designed for VLEC codes. In fact, both Huffman codes and RVLCs may be viewed as special VLEC codes having a free distance of $d_f < 2$ [1]. Hence a generic construction algorithm may be applied for designing both RVLCs and VLEC codes [1]. For the reader's convenience, we have summarised the major contributions on variable-length coding in Table 1.1 at a glance.

1.1.1.2 Variable-Length Decoding

Owing to their prefix property, source messages encoded by Huffman codes as well as RVLCs and VLEC codes may be decoded on a bit by bit basis with the aid of their code trees or look-up tables [7, 35, 37]. Given the wide employment of the ITU-T H.26x and ISO/IEC MPEG series video coding standards, numerous efficient hardware architectures have been proposed [38, 39] for achieving low-power, memory-efficient and high-throughput decoding. These methods aim for low-latency, near-instantaneous decoding and generally assume having an error-free input bit stream. However, owing to their variable-length nature, their decoding is prone to the loss of synchronisation, when transmission errors occur and the errors are likely to propagate during the decoding of the subsequent

Year	Author(s)	Contribution
1948	Shannon [6]	invented the first variable-length coding algorithm referred to as Shannon-Fano coding along with the creation of information theory.
1952	Huffman [14]	proposed the optimal prefix codes, Huffman codes.
1979	Rissanen [15]	proposed arithmetic coding.
1994	Buttigieg and Farrell [33–35]	proposed VLEC codes and investigated various VLEC code properties, construction, decoding and performance bounds.
1995	Takishima *et al.* [17]	proposed RVLCs for providing both a forward and backward decoding capability.
1997	Wen and Villasenor [18, 19]	proposed a new class of asymmetric RVLCs and applied them to the video coding framework of the H.263+ [23] and MPEG-4 [24] standards.
2001	Tsai and Wu [25, 26]	proposed the maximum symmetric suffix length (MSSL) metric [25] for designing symmetric RVLCs and the minimum repetition gap (MRG) [26] metric for asymmetric RVLCs.
2002	Lakovic and Villasenor	proposed a new family of RVLCs having a free distance of $d_f = 2$.
2003	Lakovic and Villasenor	proposed the so-called "affix index" metric for constructing asymmetric RVLCs.
	Lamy and Paccaut [36]	proposed low-complexity construction techniques for VLEC codes.

Table 1.1: Major contributions on variable-length coding.

bit stream. In other words, should a decoding error be encountered in the current codeword, a relatively large number of subsequent erroneously decoded codewords are likely to follow, even if in the meantime there were no further errors imposed by the channel. Hence, although these codes may be decoded instantaneously, it is clear that if we were to wait for several codewords before making a decision, a better decoding performance (fewer decoding errors) may be expected. Indeed, the decoding of VLCs in error-prone environments is a joint segmentation (i.e. recovering source-symbol boundaries) and estimation problem.

Numerous source decoding algorithms capable of generating a source estimate based on a sequence of received data have been developed [12, 33, 34, 40–50]. In parallel, various trellis representations [34, 41, 42, 44, 45, 48] have been devised for describing the source (coder). In [33, 34] a Maximum Likelihood (ML) sequence estimation algorithm was proposed, which selected the most likely symbol sequence instead of symbol-by-symbol decoding. More specifically, the decoder invoked a modified form of the Viterbi Algorithm (VA) [51] based on the trellis structure of VLEC codes, where each state of the trellis corresponded to the number of bits produced so far and each transition between the states represented the transmission of a legitimate codeword (or source symbol). Maximum *a posteriori* (MAP) sequence estimation based on the same trellis structure was also studied in [35]. Both algorithms are based on hard decoding, where the inputs of the source decoder are binary 0s as well as 1s and the path metric is calculated based on the Hamming distance between the transmitted bits and the received bits. Soft information may be used for improving the attainable decoding performance with the aid of soft-input soft-output (SISO) decoding algorithms. The term "soft" implies that the decoder is capable of accepting as well as of generating not only binary ("hard") decisions, but also reliability information (i.e. probabilities) concerning the bits.

The explicit knowledge of the total number of transmitted source-symbols or transmitted bits can be incorporated into the trellis representations as a termination constraint in order to assist the decoder in resynchronising at the end of the sequence. Park and Miller [41] designed a symbol-based trellis structure for MAP sequence estimation, where the number of transmitted bits is assumed to be known at the receiver, resulting in the so-called bit-constrained directed graph representation. Each state of the trellis corresponds to a specific source-symbol and the number of symbols produced so far. By contrast, when the total number of transmitted symbols is assumed to be known at the receiver, Demir and Sayood [42] proposed a symbol-constrained directed graph representation, where each state of the trellis corresponds to a specific symbol and the corresponding number of bits that have been produced. Moreover, when both constraints are known at the receiver, Bauer and Hagenauer [44] designed another symbol-trellis representation for facilitating symbol-by-symbol MAP decoding of VLCs, which is a modified version of the BCJR algorithm [52]. Later they [46] applied their BCJR-type MAP decoding technique to a bit-level trellis representation of VLCs, which was originally devised by Balakirsky [45], where the trellis was derived from the corresponding code tree by mapping each tree node to a specific trellis state. The symbol-based trellis has the drawback that for long source sequences, the number of trellis states increases quite drastically, while the bit-based trellis has a constant number of states, which only depends on the depth of the VLC tree.

For sources having memory, the inter-symbol correlation is usually modelled by a Markov chain [7], which can be directly represented by a symbol-based trellis structure [40–42, 47]. In order to fully exploit all the available *a-priori* source information such as the inter-symbol correlation, or the knowledge of the total number of transmitted bits and symbols, a three-dimensional symbol-based trellis representation was also proposed by Thobaben and Kliewer in [48]. In [49], the same bit-trellis as that in [46] was used, but the APP decoding algorithm was modified for the sake of exploiting the inter-symbol correlation. As a low-complexity solution, sequential decoding [53,54] was also adopted for VLC decoding. In addition to classic trellis-based representations, Bayesian networks were also employed for assisting the design of VLC decoders [50], where it was shown that the decoding of the Markov source and that of the VLC can be performed separately.

1.1.1.3 Classification of Source Decoding Algorithms

Generally, the source decoding problem may be viewed as a hidden Markov inference problem, i.e. that of estimating the sequence of hidden states of a Markovian source/source-encoder through observations of the output of a memoryless channel. The estimation algorithms may employ different estimation criteria and may carry out the required computations differently. The decision rules can be either optimum with respect to a sequence of symbols or with respect to individual symbols. More explicitly, the source decoding algorithms can be broadly classified into the following categories [12].

- **MAP/ML Sequence Estimation:** The estimate of the entire source sequence is calculated using the MAP or ML criterion based on all available measurements. When the *a priori* information concerning the source sequence is uniformly distributed implying that all sequences have the same probabilities or simply ignored in order to reduce the computational complexity, the MAP estimate is equivalent to the ML estimate. The algorithms outlined in [33,34,40–42, 45,50] belong to this category.

- **Symbol-by-Symbol MAP Decoding:** This algorithm searches for the estimate that maximises the *a posteriori* probability for each source symbol rather than for the entire source sequence based on all available measurements. The algorithms characterised in [44,46–50] fall into this category.

- **MMSE Decoding:** This algorithm aims for minimising the expected distortion between the original signal and the reconstructed signal, where the expected reconstructed signal may be obtained using APP values from the MAP estimation. The algorithm presented in [43] falls in this category.

1.1.2 Joint Source/Channel Decoding

Let us consider a classic communication chain using a source encoder aiming for removing the redundancy from the source signal followed by a channel encoder that aims for re-introducing the redundancy in the transmitted stream in an efficiently controlled manner in order to cope with transmission errors [55] (Chapter 1, page 2). Traditionally, the source encoder and the channel encoder are designed and implemented separately according to Shannon's source and channel coding separation theorem [6], which holds only under idealistic conditions. Therefore joint source-channel decoding (JSCD) have gained considerable attention as potentially attractive alternatives to the separate decoding of source and channel codes. The key idea of JSCD is to exploit jointly the residual redundancy of the source-coded stream (i.e. exploiting the sub-optimality of the source encoder) and the redundancy deliberately introduced by the channel code in order to correct the bit errors and to find the most likely source-symbol estimates. This requires modelling of all the dependencies present in the complete source-channel coding chain.

In order to create an exact model of dependencies amenable to optimal estimation, one can construct the amalgam of the source, source encoder and channel encoder models. This amalgamated model of the three elements of the chain has been proposed in [56]. The set of states in the trellis-like joint decoder graph contains state information about the source, the source code and the channel code. The amalgamated decoder may then be used to perform a MAP, ML or MMSE decoding. This amalgamated approach allows for optimal joint decoding, however, its complexity may remain excessive for realistic applications, since the state-space dimension of the amalgamated model explodes in most practical cases. Instead of building the Markov chain of the amalgamated model, one may consider the serial or parallel connection of two hidden Markov models (HMMs), one for the source as well as source encoder and one for the channel encoder, in the spirit of serial and parallel turbo codes. The dimension of the state space required for each of the separate models is then reduced. Furthermore, the adoption of the turbo principle [57] led to the design of iterative estimators operating alternatively on each factor of the concatenated model. The estimation performance attained is close to the optimal performance only achievable by the amalgamated model [12].

In [41, 42, 44–50, 58, 59], the serial concatenation of a source and a channel encoder was considered for the decoding of both fixed-length and variable-length codes. *Extrinsic* information was iteratively exchanged between the channel decoder and the source decoder at the receiver. The convergence behaviour of iterative source/channel decoding using fixed-length source codes and a serially concatenated structure was studied in [60], while in [49] for VLCs. The gain provided by the iterations is very much dependent on the amount of correlation present at both sides of the interleaver. The variants of the algorithms proposed for JSCD of VLC-encoded sources relate to various trellis representations forms derived for the source encoder, as described in Section 1.1.1.2, as well as to the different underlying assumptions with respect to the knowledge of the length of the symbol or bit sequences. Moreover, a parallel-concatenated source-channel coding and decoding structure designed for VLC-encoded sources is described in [61]. In comparison to a turbo code, the explicit redundancy inherent in one of the component channel codes is replaced by the residual redundancy left in the source-compressed stream after VLC encoding. It was shown in [61] that the parallel concatenated structure is superior to the serial concatenated structure for the RVLCs considered and for the AWGN channel experiencing low signal-to-noise ratios (SNRs).

Another possible approach is to modify the channel decoder in order to take into account the source statistics and the model describing both the source and the source encoder, resulting in the so-called source-controlled channel decoding (SCCD) technique [62]. A key idea presented in [62] is to introduce a slight modification of a standard channel decoding technique in order to take advantage of the source statistics. This idea has been explored [62] first in the case of Fixed-Length Coding (FLC) and has been validated using convolutional codes in the context of transmitting coded speech frames over the Global System of Mobile telecommunication (GSM) [63]. Source-controlled channel decoding has also been combined with both block and convolutional turbo codes, considering FLC for hidden Markov sources [64] or images [65–67]. Additionally, this approach has been extended to JSCD of VLC sources, such as source-controlled convolutional decoding of VLCs in [68, 69] and combined decoding of turbo codes and VLCs in [68–71]. The main contributions on JSCD of VLCs are summarised in Table 1.2.

1.1.3 Iterative Decoding and Convergence Analysis

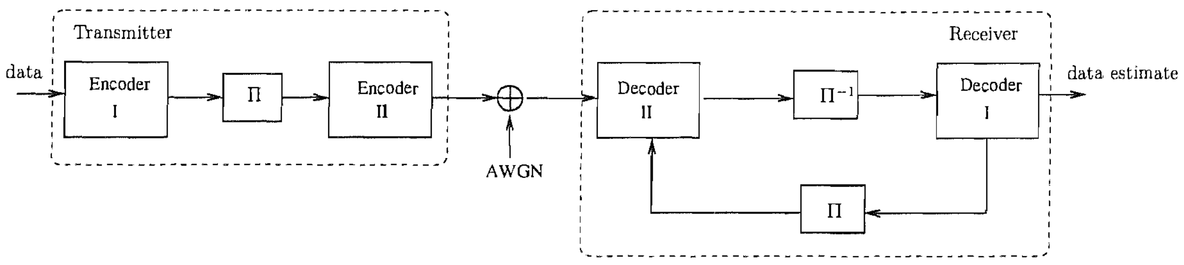
The invention of turbo codes over a decade ago [73] constitutes a milestone in error-correction coding designed both for digital communications and storage. The parallel concatenated structure employing interleavers for decorrelating the constituent encoders and the associated iterative decoding procedure exchanging only *extrinsic* information between the constituent decoders are the key elements that are capable of offering a performance approaching the previously elusive Shannon limit, although they also impose an increasing complexity upon inching closer to the limit, in particular beyond the so-called Shannonian cut-off [74]. Reliable communications for all channel capacity rates slightly in excess of the source' entropy rate becomes approachable. The practical success of the iterative turbo decoding algorithm has inspired its adaptation to other code classes, notably serially concatenated codes [75], Turbo Product Codes (TPCs) which may be constituted from Block Turbo Codes (BTCs) [76–78], and Repeat-Accumulate (RA) codes [79], and has rekindled interest [80–82] in Low-Density Parity-Check (LDPC) codes [83], which constitute a historical milestone in iterative decoding.

The serially concatenated configuration of components [75] commands particular interest in communication systems, since the "inner encoder" of such a configuration can be given more general interpretations [57, 84, 85], as seen in Fig. 1.1, such as an "encoder" constituted by a dispersive channel or by the spreading codes used in Code-Division Multiple-Access (CDMA). The corresponding iterative decoding algorithm can then be extended into new areas, giving rise to Turbo Equalisation

Year	Author(s)	Contribution
1991	Sayood and Borken-hagen [9]	considered the exploitation of the residual redundancy found in the fix-length coded quantised-source indices in the design of joint source/channel coders.
1994	Phamdo and Farvardin [72]	studied the optimal detection of discrete Markov sources over discrete memoryless channels.
1995	Hagenauer [62]	proposed the first SCCD scheme with application in speech transmissions.
1997	Balakirsky [45]	proposed a bit-level trellis representation of VLCs and applied it in a joint source-channel VLC sequence estimation scheme.
1998	Murad and Fuja [56]	proposed a joint source-channel VLC decoding scheme based on a combined model of the source, source encoder and the channel encoder.
	Miller and Park [43]	proposed a sequence-based approximate MMSE decoder for JSCD.
	Demir and Sayood [42]	investigated a JSCD scheme using MAP sequence estimation based on trellis structure constrained by the number of transmitted bits.
1999	Park and Miller [41]	studied a JSCD scheme using both exact and approximate MAP sequence estimation based on a trellis structure constrained by the number of transmitted symbols.
2000	Bauer and Hagenauer [44]	proposed a symbol-level trellis representation of VLCs and derived a symbol-by-symbol MAP VLC decoding algorithm.
	Guivarch *et al.* [68]	considered SCCD of Huffman codes combined with turbo codes.
2001	Bauer and Hagenauer [46]	investigated MAP VLC decoding based on the bit-level trellis proposed in [45].
	Guyader *et al.* [50]	studied JSCD schemes using Bayesian networks.
	Garcia-Frias and Villasenor [64]	applied SCCD to the decoding of hidden Markov sources.
	Peng *et al.* [67]	investigated a SCCD scheme using turbo codes for image transmission.
2002	Kliewer and Thobaben [47]	adopted the JSCD scheme of [44] to the decoding of correlated sources.
2003	Thobaben and Kliewer [48]	proposed a 3D symbol-based trellis for the sake of fully exploiting the *a priori* source information.
2004	Jaspar and Vandendorpe [71]	proposed a SCCD scheme using a hybrid concatenation of three SISO modules.
2005	Thobaben and Kliewer [49]	proposed a modified BCJR-algorithm [52] incorporating intersymbol correlation based on the bit-level trellis of [45].
	Jeanne *et al.* [69]	studied SCCD schemes using both convolutional codes and turbo codes.

Table 1.2: Major contributions on JSCD of VLCs.

(TE) [86], Bit-interleaved Coded Modulation (BiCM) [87,88], serial concatenated trellis coded modulation [89,90], turbo multiuser detection [91,92], turbo synchronisation [93] and coded CDMA [94]. Connection of turbo decoding algorithm to the belief propagation algorithm [95] has also been discussed in [96]. Moreover, factor graphs [97, 98] were employed [99, 100] for providing an unified framework that allows us to understand the connections and commonalities among seemingly different iterative receivers.



Configuration	En-/Decoder I (Outer Code)	En-/Decoder II (Inner Code)
Serial Concatenated Code	FEC En-/Decoder	FEC En-/Decoder
LDPC Codes	Check Nodes	Variable Nodes
Turbo Equalisation	FEC En-/Decoder	Multipath Channel/Equaliser
Turbo BiCM	FEC En-/Decoder	Mapper/Demapper
Turbo Source-Channel Decoding	Source En-/Decoder	FEC En-/Decoder
Turbo CDMA	FEC En-/Decoder	Spreading Code/MUD
Turbo MIMO	FEC En-/Decoder	Mapper/MIMO Detector

Figure 1.1: A serial concatenated system with iterative detection/decoding [101], where Π denotes the interleaver and Π^{-1} represents the de-interleaver.

Typically, Bit-Error Rate (BER) charts of iterative decoding schemes can be divided into three regions [102]: 1) the region of low E_b/N_0 values with negligible iterative BER reduction, 2) the "turbo-cliff" region, which is also referred to as the "waterfall" region with consistent iterative BER reduction over many iterations, and 3) the BER floor region often recorded at moderate to high E_b/N_0 values, where a rather low BER can be reached after just a few iterations. Efficient analytical bounding techniques have been found for moderate to high E_b/N_0 values in [75, 103–106], where design rules were also derived for the sake of lowering the BER floors. On the other hand, a long-standing open problem was understanding of the convergence behaviour of iterative decoding algorithms in the turbo-cliff region. A successful approach to this problem referred to as *density evolution* was pioneered in [81, 82], which investigates the probability density functions (PDFs) of the information communicated within the iterative decoding algorithms. Based on the evolution of these PDFs during iterative decoding, convergence thresholds have been derived for the iterative decoding of LDPC codes [81]. Simplified approaches were also proposed by assuming that PDFs considered above are Gaussian [107] or by observing a single parameter of them such as the SNR measure in [108], a reliability measure in [109] and the mutual information measure in [102, 110]. The convergence prediction capability of six different measures were compared in [111], where the mutual information measure used in the EXtrinsic Information Transfer (EXIT) chart [102, 110] analysis was found to have a high accuracy, regardless of the specific iterative decoding schemes considered. Additionally, the cross-entropy [112] was employed by Luo and Sweeney in [113] as a measure used for determining the convergence threshold of turbo decoding algorithms, where neither the knoweledge of the source information nor the Gaussian approximation was required, while the relationship between the Minimum Cross-Entropy (MCE) decoding algorithm and the iterative decoding algorithm was discussed in [114].

EXIT charts have been successfully applied for predicting the convergence behaviour of various concatenated systems, such as parallel [102, 115] and serially concatenated codes [116], turbo equalisation [117], Bit-interleaved Coded Modulation using Iterative Decoding (BiCM-ID) [110, 118] and JSCD [49, 60]. A unified framework derived for applying EXIT chart analysis for various serially concatenation schemes was presented in [119]. EXIT functions were defined with the aid of a universal decoding model in [120] and several properties of the related functions were derived for erasure channels. A specific property allows us to quantify the area under an EXIT function in terms of the conditional entropy. A useful consequence of this result is that the design of capacity-approaching codes reduces to a curve-fitting problem in the context of EXIT charts [120]. For example, EXIT charts have been applied for designing IRregular Convolutional Codes (IRCCs) [84, 85] in order to approach the AWGN channel's capacity, for designing irregular LDPC codes [121] and Irregular Repeat-Accumulate (IRA) codes [122] for approaching the Multiple-Input Multiple-Output (MIMO) fading channel's capacity.

EXIT charts have then also been extended from 2D to 3D for the sake of analysing the convergence behaviour of multi-stage iterative decoding schemes consisting of more than two components, such as for example PCCs consisting of three constituent codes [123], three-stage turbo equalisation [124] and three-stage iterative decoding and demodulation [125, 126]. A 3D to 2D projection technique was presented in [124–126], which simplifies the convergence analysis of multi-stage systems. Optimising the activation order of the component decoders was also investigated in [125, 126]. Furthermore, the philosophy of EXIT chart analysis has been extended from binary decoders to non-binary decoders [127, 128]. The major contributions on iterative decoding and its convergence analysis are summarised in Table 1.3 and Table 1.4.

Year	Author(s)	Contribution
1962	Galager [83]	invented LDPC codes.
1966	Forney [129]	proposed the first concatenated coding scheme using a convolutional code as the inner code and a Reed-Solomon code as the outer code.
1993	Berrou *et al.* [73]	invented turbo codes, which achieved near-Shannon limit performance by employing iterative decoding.
1995	Douillard *et al.* [86]	proposed turbo equalisation, which was capable of eliminating channel-induced inter-symbol-interference by performing channel equalisation and channel decoding iteratively.
	Divsalar and Pollara [130]	extended the turbo principle to multiple parallel concatenated codes.
1996	Benedetto and Montorsi [103, 131, 132]	proposed an analytical bounding technique for evaluating the bit error probability of a parallel concatenated coding scheme [103] as well as extended the turbo principle to serially concatenated block and convolutional codes [131, 132].
	Hagenauer *et al.* [76]	studied the log-likelihood algebra of iterative decoding algorithms, which were generalised for employment in any linear binary systematic block code.
	MacKay and Neal [80]	"re-discovered" LDPC codes [83] and proposed a new class of LDPC codes constructed on the basis of sparse random parity check matrices, which was capable of achieving near-Shannon limit performance.
1997	Benedetto *et al.*	investigated iterative decoding of an outer convolutional decoder and an inner TCM decoder, originally devised by Ungerboeck in 1982 [133].
1998	Benedetto *et al.* [75, 134]	derived design guidelines for serially concatenated codes with the aid of the union-bound technique [75] as well as extended the turbo principle to multiple serially concatenated codes [134].
	Moher [91]	proposed an iterative multiuser decoder for near-capacity communications.
	Reed *et al.* [92]	investigated iterative multiuser detection designed for DS-CDMA.
	Kschischang and Frey [99]	studied iterative decoding based on factor graph models.
	Divsalar *et al.* [79]	proposed repeat-accumulate (RA) codes, where turbo-like performance was achieved at a low-complexity.
	Pyndiah [77]	proposed turbo product codes.
	ten Brink *et al.*	studied iterative decoding between an outer channel decoder and an inner soft-demapper in the context of BICM.
	McEliece *et al.* [96]	discovered the connection between turbo-decoding algorithms and Pearl's belief propagation algorithm [95].
1999	Wang and Poor [94]	proposed iterative interference cancellation and channel decoding for coded CDMA schemes.

Table 1.3: Major contributions on iterative decoding and convergence analysis (Part 1).

Year	Author(s)	Contribution
1999	ten Brink [110]	proposed EXIT charts for analysing the convergence behaviour of iterative decoding and demapping [110].
2000	ten Brink [115, 116, 119]	applied EXIT charts to the convergence analysis of various iterative decoding algorithms, such as iterative decoding of PCCs [115] and SCCs [116], turbo equalisation [119] as well as proposed a common framework of EXIT chart analysis in [119].
2001	Richardson and Urbanke [81]	proposed the density evolution algorithm for analysing the convergence behaviour of LDPC decoding.
	Richardson *et al.* [82]	employed the density evolution algorithm for the sake of designing capacity-achieving irregular LDPCs.
	Kschischang *et al.* [97]	studied the sum-product algorithm with the aid of factor graphs.
	ten Brink [123]	extended EXIT charts from 2D to 3D for analysing the convergence behaviour of three-stage parallel concatenated codes.
	Grant [127]	extended EXIT charts to non-binary decoders, where mutual information was calculated based on histogram measurements.
2002	Tüchler and Hagenauer [84]	proposed irregular convolutional codes as well as a simplified method for computing mutual information in EXIT chart analysis.
	Tüchler [124]	applied 3D EXIT charts to the convergence analysis of three-stage serially concatenated turbo equalisation schemes.
	Tüchler *et al.* [117, 135]	proposed turbo MMSE equalisation [135] and applied EXIT chart analysis to it [117].
2003	Brännström [125, 126]	proposed a 3D-to-2D projection technique for facilitating the EXIT chart analysis of multiple serially concatenated codes. Additionally, a search algorithm was proposed for finding the optimal decoder activation order.
2004	Ashikhmin *et al.* [120]	presented a universal model for conducting EXIT chart analysis as well as various properties of EXIT functions, which form the theoretical foundation of EXIT-curve matching techniques.
	Land *et al.* [136]	provided a low-complexity mutual information estimation method, which significantly simplifies the EXIT chart analysis.
2006	Kliewer *et al.* [128]	proposed an efficient and low-complexity method of computing non-binary EXIT charts from index-based *a posteriori* probabilities, which may be considered as a generalisation of the approach presented in [136].

Table 1.4: Major contributions on iterative decoding and convergence analysis (Part2).

1.2 Motivation and Methodology

In the spirit of the remarkable performance of turbo codes, as described in Section 1.1, the turbo principle has been successfully applied to various generalised decoding/detection problems, resulting in diverse turbo-transceivers [74]. In the meantime, new challenges have arisen. For example, how to choose the parameters of a turbo transceiver consisting of multiple components, so that near-capacity performance can be achieved? Owing to iterative decoding, different components of the turbo-transceivers are combined and affect each other during each iteration. Hence, in contrast to the design of conventional non-iterative transceivers the optimisation should be considered jointly rather than individually. Moreover, as argued in Section 1.1, EXIT charts have shown their power in analysing the convergence behaviour of iterative receivers. Therefore, in this treatise we mainly rely on this semi-analytical tool as well as on the conventional Monte Carlo simulation method in our attempt to tackle the problem of optimising multi-stage turbo-transceivers and JSCD schemes.

1.3 Outline of the Thesis

Having briefly reviewed the literature of source coding, joint source/channel decoding, concatenated coding and iterative decoding as well as convergence analysis, let us now outline the organisation of the thesis.

- **Chapter 2: Sources and Source Codes**

  This chapter introduces various entropy coding schemes designed for memoryless sources having known alphabets along with several trellis-based soft-decision decoding methods in contrast to conventional hard-decision decoding. Section 2.2 describes the source models used in this treatise; various source coding schemes designed for memoryless sources are introduced in Section 2.3, such as Huffman codes, Reversible Variable-Length Codes (RVLCs) and Variable-Length Error-Correcting (VLEC) codes. A generic algorithm contrived for efficiently constructing both RVLCs and VLEC codes having different free distances is also presented. In Section 2.4.1, two different trellis representations of source codes are introduced in addition to their conventional binary tree representation. Based on these trellis representations, the Maximum Likelihood Sequence Estimation (MLSE) and MAP decoding algorithm are derived for soft-decision aided source decoding in Section 2.4.2.1 and Section 2.4.2.2, respectively. Finally, the source Symbol Error Rate (SER) performance of these decoding algorithms is presented in Secion 2.4.3.

- **Chapter 3: Iterative Source/Channel Decoding**

Based on the soft VLC decoding algorithms described in Chapter 2, various iterative source-channel decoding (ISCD) schemes may be formed. The SER performance of these schemes contrived for communicating over both AWGN channels and dispersive channels is quantified and their convergence behaviour is analysed with the aid of EXIT charts. More explicitly, Section 3.1 provides an overview of various code concatenation schemes and the so-called "turbo principle". A generic SISO *a posteriori* Probability (APP) decoding module is described in Section 3.2, which serves as the "kernel" of iterative decoding schemes. Additionally, the semi-analytical EXIT chart tool for convergence analysis is introduced in Section 3.2.2. Section 3.3 evaluates the performance of ISCD schemes designed for transmission over AWGN channels, along with the EXIT chart analysis of their convergence behaviour. More particularly, the effects of different source codes are considered. Furthermore, the performance of ISCD schemes is investigated for transmission over dispersive channels in Section 3.4, which necessitates the employment of channel equalisation. Both scenarios with and without channel precoding are considered in Section 3.4.2 and Section 3.4.3, respectively. A three-stage iterative channel equalisation, channel decoding and source decoding scheme is presented in Section 3.4.4.

- **Chapter 4: Three-Stage Serially Concatenated Turbo Equalisation**

In this chapter, a three-stage serially concatenated turbo MMSE equalisation scheme is investigated. The different componets of the iterative receiver are jointly optimised with the aid of EXIT charts for the sake of achieving a near-capacity performance. More specifically, Section 4.2 provides a short introduction to SISO MMSE equalisation. The conventional two-stage turbo equalisation scheme is described in Section 4.3, while Section 4.4 presents a three-stage turbo equalisation scheme. Both 3D and 2D EXIT chart analysis are carried out in Section 4.4.2, where the channel code is optimised for obtaining the lowest possible E_b/N_0 convergence threshold and the activation order of the component decoders is optimised for achieving the fastest convergence, while maintaining a low decoding complexity. Various methods of visualising the three-stage iterative decoding process are described in Section 4.4.4. The effects of different interleaver block lengths on the attainable system performance are discussed in Section 4.4.5. Section 4.5 demonstrates the technique of EXIT chart matching and the design of IRregular Convolutional Codes (IRCCs), which results in a capacity-approaching three-stage turbo equalisation scheme. In addition to unity-rate intermediate codes, the employment of non-unity rate intermediate codes in three-stage schemes is investigated in Section 4.6.

- **Chapter 5: Conclusions and Future Work**

The major findings of our work are summarised in this chapter along with our suggestions for future research.

1.4 Novel Contributions

The thesis is based on the publications and manuscript submissions of [1–5, 137]. The novel contribution of this thesis includes the following:

- In Chapter 2, a novel and high-efficiency algorithm was proposed for constructing both RVLCs and VLEC codes [1]. For RVLCs the algorithm results in codes of higher efficiency and/or shorter maximum codeword length than the algorithms previously disseminated in the literature. For VLEC codes it significantly reduces the construction complexity imposed in comparison to the original algorithm of [34, 35].
- In Chapter 2, soft VLC decoding based on both symbol-level and bit-level trellis representation was investigated. The performance of different decoding criteria such as MLSE and MAP was evaluated. In particular, the effect of different free distances of the source codes was quantified.
- In Chapter 3, a novel iterative channel equalisation, channel decoding and source decoding scheme was proposed [2, 3]. It was shown that without using channel coding, the redundancy in the source code alone was capable of eliminating the channel-induced ISI, provided that the channel equalisation and source decoding was performed iteratively at the receiver. When employing channel coding, a three-stage joint source/channel detection scheme was formed, which was capable of outperforming the separate source/channel detection scheme by 2 dB in terms of the E_b/N_0 value required for achieving the same BER. Furthermore, with the aid of EXIT charts, the systems' convergence thresholds and the design of channel precoders were optimised.
- In Chapter 4, a novel three-stage serially concatenated MMSE turbo equalisation scheme was proposed [4]. The performance of conventional two-stage turbo equalisation schemes is generally limited by a relatively high BER "shoulder" [138]. However, when employing a unity-rate recursive convolutional code as the intermediate component, the three-stage turbo equalisation scheme was capable of achieving a steep BER turbo "cliff", beyond which an infinitesimally low BER was obtained.
- In Chapter 4, the concept of EXIT modules was proposed, which significantly simplified the EXIT chart analysis of multiple concatenated systems, casting them from the 3D domain to the

2D domain. As a result, the outer constituent code of the three-stage turbo equalisation scheme was optimised for achieving the lowest possible convergence threshold. The activation order of the component decoders was optimised for obtaining the convergence at the lowest possible decoding complexity using the lowest total number of iterations or decoder activations.

- In Chapter 4, a near-channel-capacity three-stage turbo equalisation scheme was proposed [5]. The proposed scheme was capable of achieving a performance within 0.2 dB of the dispersive channel's capacity with the aid of EXIT chart matching and IRCCs. This scheme has been further generalised to MIMO turbo detection [137].

- In Chapter 4, the employment of non-unity-rate intermediate codes in a three-stage turbo equalisation scheme was also investigated. The maximum achievable information rates of such schemes were determined and the serial concatenation of the outer constituent code and the intermediate constituent code was optimised for achieving the lowest possible decoding convergence threshold.

Chapter 2

Sources and Source Codes

2.1 Introduction

This chapter primarily investigates the design of VLCs constructed for compressing discrete sources as well as the choice of soft-decoding methods for VLCs. First, in Section 2.2 appropriate source models are introduced for representing both discrete memoryless sources as well as discrete sources having memory. In order to map discrete source symbols to binary digits, source coding is invoked, which may employ various source codes, such as Huffman codes [14], RVLCs [17] as well as VLEC codes [35]. Unlike channel codes such as convolutional codes, VLCs constitute nonlinear codes, hence the superposition of legitimate codewords does not necessarily yield a legitimate codeword. The design methods of VLCs are generally heuristic. Similar to channel codes, the coding rate of VLCs is also an important design criterion in the context of source codes, although some redundancy may be intentionally imposed in order to generate reversible VLCs, where decoding may be commenced from both ends of the codeword, or to incorporate error-correction capability. A novel code construction algorithm is proposed in Section 2.3, which typically constructs more efficient source codes in comparison to the state-of-the-art methods found in the literature.

Moreover, the soft-decoding of VLCs is presented in Section 2.4. In contrast to traditional hard-decoding methods processing binary ones and zeros, the family of soft-decoding methods is capable of processing soft input bits, i.e. hard bits associated with reliability information, which are optimal in the sense of maximising the *a posteriori* probability. Finally, the performance of soft-decoding methods is quantified in Section 2.4.3, when communicating over AWGN channels.

2.2 Source Models

Practical information sources can be broadly classified into the family of analog and digital sources. The output of an analog source is a continuous function of time. With the aid of sampling and quantization an analog source signal can be transformed to its digital representation.

The output of a digital source is constitued by a finite set of ordered, discrete symbols often referred to as an alphabet. Digital sources are usually described by a range of characteristics, such as the source alphabet, the symbol rate, the individual symbol probabilities and the probabilistic interdependence of symbols.

At the current state-of-the-art, most communication systems are implemented using digital transmission and implementation techniques as a benefit of a host of advantages, such as their ease of implementation using Very-Large-Scale Integration (VLSI) technology and their superior performance in hostile propagation environments. Digital transmissions invoke a whole range of diverse signal processing techniques, which would otherwise be impossible or difficult to implement in analogue forms. In this thesis we mainly consider digital sources.

2.2.1 Memoryless Sources

Following Shannon's approach [6], let us consider a source A emitting one of N_s possible symbols from the alphabet $\mathscr{A} = \{a_1, a_2, \ldots, a_i \ldots a_{N_s}\}$ having symbol probabilities of $\{p_i, i = 1, 2, \ldots N_s\}$. Here the probability of occurrence for any of the source symbols is assumed to be independent of previously emitted symbols, hence this type of source is referred to as a "memoryless" source . The information carried by symbol a_i is

$$I_i = -\log_2 p_i \quad \text{[bits]}. \tag{2.1}$$

The average information per symbol is referred to as the source entropy, which is given by

$$H = -\sum_{i=1}^{N_s} p_i \log_2 p_i \quad \text{[bits/symbol]}, \tag{2.2}$$

where H is bounded according to:

$$0 \leq H \leq \log_2 N_s, \tag{2.3}$$

and we have $H = \log_2 N_s$, when the symbols are equiprobable.

Then the average source information rate can be defined as the product of the information carried by a source symbol, given by the entropy H, and the source emission rate R_s [symbols/sec]:

$$R_I = R_s \cdot H \text{ [bits/sec]}. \tag{2.4}$$

2.2.2 Sources with Memory

Most analog source signals, such as speech and video, are inherently correlated, owing to various physical restrictions imposed on the analog sources [7]. Hence all practical analog sources possess some grade of memory, which constitutes a property that is also retained after sampling and quantization. An important feature of sources having memory is that they are predictable to a certain extent, hence, they can usually be more efficiently encoded than unpredictable memoryless sources.

Predictable sources that exhibit memory can be conveniently modeled by *Markov processes.* A source having a memory of one symbol interval directly "remembers" only the previously emitted source symbol and depending on this previous symbol it emits one of its legitimate symbols with a certain probability, which depends explicitly on the state associated with this previous symbol. A one-symbol-memory model is often referred to as a first-order model.

In general, Markov models are characterised by their state probabilities $P_i = P(X = X_i), i = 1 \dots N$, where N is the number of states, as well as by the transition probability $p_{ij} = P(X_j/X_i)$, where p_{ij} explicitly indicates the probability of traversing from state X_i to state X_j. At every state transition, they emit a source symbol. Fig. 2.1 shows a two-state first-order Markov model.

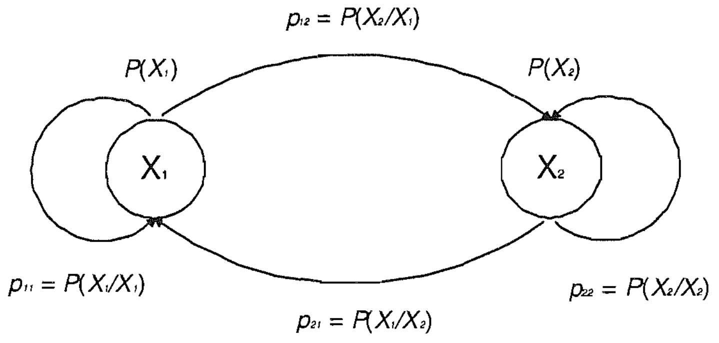


Figure 2.1: A two-state Markov model.

The entropy of a source having memory is defined as the weighted average of the entropy of the individual symbols emitted from each state, where weighting is carried out by taking into account the probability of occurrence of the individual states, namely P_i. In analytical terms, the symbol entropy for state $X_i, i = 1 \dots N$ is given by:

$$H_i = -\sum_{j=1}^{N} p_{ij} \cdot \log_2 p_{ij} \quad i = 1 \dots N. \tag{2.5}$$

The symbol entropies weighted by their probability of occurrence and over all symbols give the source entropy:

$$\begin{aligned} H &= \sum_{i=1}^{N} P_i H_i \\ &= -\sum_{i=1}^{N} P_i \sum_{j=1}^{N} p_{ij} \log_2 p_{ij}. \end{aligned} \tag{2.6}$$

Finally, assuming a source symbol rate of R_s, the average information emission rate R of the source is given by:

$$R_I = R_s \cdot H \;\; [\text{bits/sec}]. \tag{2.7}$$

A widely used analytical Markov model is the *Autoregressive(AR) model.* A zero-mean random sequence $y(n)$ is referred to as *an AR process of order N*, if it is generated as follows:

$$y(k) = \sum_{n=1}^{N} \rho_n y(k-n) + \varepsilon(k), \forall k, \tag{2.8}$$

where $\varepsilon(k)$ is an uncorrelated zero-mean, random input sequence with variance σ^2. A variety of practical information sources are adequately modeled by the analytically tractable first-order Markov/AR model given by:

$$y(k) = \varepsilon(k) + \rho y(k-1), \tag{2.9}$$

where ρ is the correlation coefficient of the process $y(k)$.

2.3 Source Codes

Source coding is the process by which each source symbol is mapped to a binary codeword, hence the output of an information source is converted to a sequence of binary digits. For a non-uniform discrete memoryless source (DMS), the transmission of a highly probable symbol carries little information and hence assignning equal number of bits to every symbol in the context of fixed-length encoding is inefficient. Therefore various variable length coding methods are proposed. The general principle is that the source symbols which occur more frequently are assigned short codewords and those that occur infrequently are assigned long codewords, which generally results in a reduced average codeword length. Hence variable length coding may also be viewed as a data compression method. *Shannon's source coding theorem* suggests that by using a *source encoder* before transmission the efficiency of a system transmitting equiprobable source symbols can be arbitrarily approached [6].

Code efficiency can be defined as the ratio of the source information rate and the average output bit rate of the source encoder, which is usually referred to as *coding rate* in the channel coding community [13]. The most efficient VLC is Huffman code [14]. Besides code efficiency, there are other VLC design criteria that may be important in the application environment, such as having a high error resilience or a symmetric codeword construction. Thus both RVLCs [17,25,26] and VLEC codes [35] are proposed in the literature, which we will describe in the following sections.

Before proceeding further, we now provide some definitions.

2.3.1 Definitions and Notions

Let X be a code alphabet with cardinality[1] q. In what follows, we shall assume that we have $q = 2$, although the results may readily be extended to the general case of $q > 2$. A finite sequence $\mathbf{x} = x_1 x_2 \dots x_l$ of code symbols (or bits for binary alphabets) is referred to as a codeword over X of length $\ell(\mathbf{x}) = l$, where each code symbol satisfys $x_i \in X$, for all $i = 1, 2, \dots, l$. A set C of legitimate words is referred to as a code. Let the code C have N_s legitimate codewords $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{N_s}$ and let the length of $\mathbf{c}_j$ be given by $l_j = \ell(\mathbf{c}_j)$, $j = 1, 2, \dots, N_s$. Without loss of generality, assume that we have $l_1 \leq l_2 \leq \dots \leq l_{N_s}$. Furthermore, let K denote the number of different codeword lengths in the code C and let these lengths be given by $L_1, L_2, \dots, L_K$, where we have $L_1 < L_2 < \dots < L_K$, and the corresponding number of codewords having specific lengths is given by $n_1, n_2, \dots, n_K$, where we have $\sum_{k=1}^{K} n_k = N_s$. Where applicable, we use the VLC $\mathscr{C} = \{00, 11, 010, 101, 0110\}$ as an example.

Definition 1 The *Hamming weight* (or simply *weight*) of a codeword $\mathbf{c}_i$, $w(\mathbf{c}_i)$, is the number of nonzero symbols in $\mathbf{c}_i$. The *Hamming distance* between two equal-length words, $d_h(\mathbf{c}_i, \mathbf{c}_j)$, is the number of positions in which $\mathbf{c}_i$ and $\mathbf{c}_j$ differ. For example, the Hamming distance between the codeword '00' and '11' is $d_h(00, 11) = 2$, while $d_h(010, 101) = 3$.

Definition 2 The *minimum block distance* d_{b_k} at the length of L_k for a VLC C is defined as the minimum Hamming distance between all codewords having the same length L_k, i.e. we have [34]

$$d_{b_k} = \min\{d_h(\mathbf{c}_i, \mathbf{c}_j) : \mathbf{c}_i, \mathbf{c}_j \in C, i \neq j \text{ and } \ell(\mathbf{c}_i) = \ell(\mathbf{c}_j) = L_k\}.$$

There are λ different minimum block distances, one for each different codeword length. However, if

[1]The number of elements in a set.

for some length L_k there is only one codeword, i.e. we have $n_k = 1$, then in this case the minimum block distance for length L_k is undefined. For example, the minimum block distance at length 2 for the VLC $\mathscr{C} = \{00, 11, 010, 101, 0110\}$ is $d_{b_2} = 2$, while $d_{b_3} = 3$ and d_{b_4} is undefined.

Definition 3 The *overall minimum block distance*, b_{min}, of a VLC C is defined as the minimum value of d_{b_k} over all possible VLC codeword lengths $k = 1, 2, \cdots, K$, i.e. we have [34]

$$b_{min} = \min\{d_{b_k} : k = 1, 2, \cdots, K\}.$$

According to this definition, the overall minimum block distance of the VLC $\mathscr{C} = \{00, 11, 010, 101, 0110\}$ is $b_{min} = \min(d_{b_2}, d_{b_3}) = 2$.

Definition 4 The Buttigieg-Farrell (B-F) *diverging distance*[2] $d_d(\mathbf{c}_i, \mathbf{c}_j)$ between two different-length codewords $\mathbf{c}_i = c_{i_1}c_{i_2}\cdots c_{i_{l_i}}$ and $\mathbf{c}_j = c_{j_1}c_{j_2}\cdots c_{j_{l_j}}$, where we have $\mathbf{c}_i, \mathbf{c}_j \in C, l_i = \ell(\mathbf{c}_i)$ and $l_j = \ell(\mathbf{c}_j)$, is defined as the Hamming distance between the codeword sections spanning the common support of these two codewords, when they are left-aligned, i.e. assuming $l_j < l_i$ we have [34]

$$d_d(\mathbf{c}_i, \mathbf{c}_j) = d_h(c_{i_1}c_{i_2}\cdots c_{i_{l_j}}, c_{j_1}c_{j_2}\cdots c_{j_{l_j}}).$$

For example, the B-F diverging distance between the codewords '00' and '010' is the Hamming distance between '00' and '01', i.e. we have a common-support distance of $d_d(00, 010) = 1$. Similarly, we have $d_d(00, 101) = 1$, $d_d(00, 0110) = 1$, $d_d(11, 010) = 1$, $d_d(11, 101) = 1$, $d_d(11, 0110) = 1$, $d_d(010, 0110) = 1$ and $d_d(101, 0110) = 2$.

Definition 5 The *minimum B-F diverging distance* d_{min} of the VLC C is the minimum value of all the B-F diverging distances between all possible pairs of unequal length codewords of C, i.e. we have [34]

$$d_{min} = \min\{d_d(\mathbf{c}_i, \mathbf{c}_j) : \mathbf{c}_i, \mathbf{c}_j \in C, \ell(\mathbf{c}_i) > \ell(\mathbf{c}_j)\}.$$

According to this definition, the minimum diverging distance of the VLC $\mathscr{C} = \{00, 11, 010, 101, 0110\}$ is $d_{min} = 1$.

Definition 6 The *B-F converging distance* $d_c(\mathbf{c}_i, \mathbf{c}_j)$ between two different-length codewords $\mathbf{c}_i = c_{i_1}c_{i_2}\cdots c_{i_{l_i}}$ and $\mathbf{c}_j = c_{j_1}c_{j_2}\cdots c_{j_{l_j}}$, where we have $\mathbf{c}_i, \mathbf{c}_j \in C, l_i = \ell(\mathbf{c}_i)$ and $l_j = \ell(\mathbf{c}_j)$, is defined as the Hamming distance between the codeword sections spanning the common support of these two codewords, when they are right-aligned, i.e. assuming $l_j < l_i$ we have [34]

$$d_c(\mathbf{c}_i, \mathbf{c}_j) = d_h(c_{i_{l_i-l_j+1}}c_{i_{l_i-l_j+2}}\cdots c_{i_{l_i}}, c_{j_1}c_{j_2}\cdots c_{j_{l_j}}).$$

[2]We will also refer to the Buttigieg-Farrell (B-F) diverging distance [34] as the 'left-aligned common-support distance'.

For example, the B-F converging distance between the codewords '00' and '010' is the Hamming distance between 00 and 10, i.e. $d_c(00, 010) = 1$. Similarly, we have $d_c(00, 101) = 1$, $d_c(00, 0110) = 1$, $d_c(11, 010) = 1$, $d_c(11, 101) = 1$, $d_c(11, 0110) = 1$, $d_c(010, 0110) = 1$ and $d_c(101, 0110) = 2$.

Definition 7 The *minimum B-F converging distance* c_{min} of the VLC C, is the minimum value of all the B-F converging distances between all possible pairs of unequal length codewords of C, i.e. we have [34]

$$c_{min} = \min\{d_c(\mathbf{c}_i, \mathbf{c}_j) : \mathbf{c}_i, \mathbf{c}_j \in C, \ell(\mathbf{c}_i) > \ell(\mathbf{c}_j)\}.$$

According to this definition, the minimum B-F diverging distance of the VLC $\mathscr{C} = \{00, 11, 010, 101, 0110\}$ is $c_{min} = 1$.

Definition 8 The set of VLC codeword sequences consisting of a total number of N code symbols (or bits for binary codes) is defined as the *extended code* of order N of a VLC. Note that such codeword sequences may consist of an arbitrary number of VLC codewords. Let the set F_N be the extended code of order N of the VLC C and integer η_i be the number of VLC codewords, which constitute an element of the set F_N, then we have [34]

$$F_N = \{\mathbf{f}_i : \mathbf{f}_i = \mathbf{c}_{i_1}\mathbf{c}_{i_2}\cdots\mathbf{c}_{i_{\eta_i}}, \ell(\mathbf{f}_i) = N, \mathbf{c}_{i_j} \in C, \forall j = 1, 2, \ldots, \eta_i\}.$$

For example, $F_4 = \{0011, 1100, 0110\}$ is the extended code of order 2 for the VLC $\mathscr{C} = \{00, 11, 010, 101, 0110\}$.

Definition 9 The *free distance*, d_f, of a VLC C is the minimum Hamming distance between any two equal-length VLC codeword sequences, which may consist of an arbitrary number of VLC codewords. With the aid of the definition of extended codes, the free distance of a VLC may be defined as the minimum Hamming distance in the set of all arbitrary extended codes, i.e. we have [34]

$$d_f = \min\{d_h(\mathbf{f}_i, \mathbf{f}_j) : \mathbf{f}_i, \mathbf{f}_j \in F_n, i \neq j, n = 1, 2, 3, \ldots\},$$

where F_n is the extended code of order n of C. It was found in [34,35] that d_f is the most important parameter that determines the error-correcting capability of VLEC codes at high channel signal-to-noise ratios (SNRs). The relationship between d_f, b_{min}, d_{min}, and c_{min} is described by the following theorem:

Theorem 2.1 The free distance d_f of a VLC is given by [34]

$$d_f = \min(b_{min},\ d_{min} + c_{min}).$$

For the proof of the theorem please refer to [34]. According to this theorem, the free distance of the VLC $\mathscr{C} = \{00, 11, 010, 101, 0110\}$ is $d_f = \min(2, 1+1) = 2$.

Definition 10 Let A be a memoryless data source having an alphabet of $\{a_1, a_2, \cdots, a_{N_s}\}$, where the elements have a probability of occurrence given by $p_i, i = 1, 2, \cdots, N_s$. The source A is encoded using the code C by mapping symbol a_i to codeword c_i for all indices $i = 1, 2, \cdots, N_s$. Then the *average codeword length* $\bar{L}$ is given by

$$\bar{L} = \sum_{i=1}^{N_s} l_i p_i,$$

where $l_i = \ell(\mathrm{c}_i)$ represents the length of the codeword c_i.

Definition 11 A VLC's *efficiency*, η, is defined as

$$\eta = \frac{H(A)}{\bar{L}},$$

where $H(A)$ is the entropy of source A, while the VLC *redundancy*, ϱ, is defined as

$$\varrho = 1 - \eta = \frac{\bar{L} - H(A)}{\bar{L}}.$$

2.3.2 Some Properties of Variable-Length Codes

Non-Singular Codes

A code C is said to be *non-singular*, if all the codewords in the code are distinct. Both fixed and variable-length codes must satisfy this property in order to be useful.

Unique Decodability

A code C is said to be *uniquely decodable* if we can map a string of codewords uniquely and unambiguously back to the original source symbols. It is plausible that all fixed-length codes, which are non-singular are also uniquely decodable. However, this is not true in general for variable-length codes.

A necessary and sufficient condition for the existence of a uniquely decodable code is provided by the McMillan inequality [139].

Theorem 2.2: A necessary and sufficient condition for the existence of a uniquely decodable code having codeword lengths of $l_1, l_2, \cdots, l_s$ is that we have [139]

$$\sum_{i=1}^{s} q^{-l_i} \leq 1,$$

where q is the cardinality of the code alphabet.

Instantaneously Decodability

For practical applications, it is required that the decoding delay should be as low as possible in order to facilitate real-time interactive communications. The minimum possible decoding delay is achieved, when a codeword becomes immediately decodable, as soon as it was completely received. Any delay lower than this would imply that the code is redundant. A code satisfying this property is referred to as an *instantaneously decodable code* [140]. It is plausible that for a code to satisfy this property, none of its codewords may constitute a prefix of another codeword. Therefore, these codes are also known as *prefix codes* [140]. Hence prefix codes are uniquely decodable and have a decoding delay of at most l_s (maximum codeword length) bit duration.

Interestingly enough, McMillan's inequality given in [139] is also a necessary and sufficient condition for the existence of a prefix code having codeword lengths of $l_1, l_2, \cdots, l_s$, which was stated as the *Kraft inequality* [141]. Chronologically, the proof of this inequality was provided in [141] before that of McMillan's [139].

2.3.3 Huffman Code

Huffman codes were named after their inventor, D. A. Huffman [14] and as a benefit of their high coding efficiency they have been widely adopted in both still image and video coding standards [7]. Huffman codes are optimum in the sense that the average number of binary digits required for representing the source symbols is as low as possible, subject to the constraint that the codewords satisfy the above-mentioned prefix condition, which allows the received sequence to be uniquely and instantaneously decodable.

The construction of Huffman codes is fairly straightforward and is described in many textbooks [7, 140], and thus is omitted here. It is noteworthy nonetheless that there are more than one possible Huffman trees for a given source, i.e. the Huffman code construction is not necessarily unique.

Amongst the diverse variants a meritorious one is the canonical Huffman code, which is defined as the code having the following characteristics [142]:

- Shorter codewords have a numerically higher value than longer codewords, provided that short codewords are filled with zeros to the right so that all codewords have the same length
- For a specific codeword length, the corresponding numerical values increase monotonically upon the decrease of the source symbol probabilities.

For example, the code $\{11, 10, 01, 001, 000\}$ is *the* canonical Huffman code for the source given in Table 2.1. It can be seen that we have $(11\underline{0})_2 = (6)_{10} > (10\underline{0})_2 = (4)_{10} > (01\underline{0})_2 = (2)_{10} > (001)_2 = (1)_{10} > (000)_2 = (0)_{10}$, where $\underline{0}$ denotes the extra zeros that are appended to the short codewords. Hence the above-mentioned properties of canonical Huffman codes are satisfied. In general, the canonical Huffman code has the minimum total sum of codeword-lengths $\sum l_i$ and simultaneously the minimum maximum codeword-length L_{max} in addition to the minimum average codeword-length $\bar{L}$, while allowing fast instantaneous decoding [37]. The construction of this code was described for example, in [142].

2.3.4 Reversible Variable Length Code

Due to the variable-length nature of Huffman codes, they are very sensitive to errors occurred in noisy environments. Even a single bit error is extremely likely to cause loss of synchronization, which is the term often used to describe the phenomenon of losing track of the beginning and end of VLCs, when a single error renders a VLC codeword unrecognisable and hence the data received after the bit error position becomes useless, until a unique synchonisation word is found.

As a potential remedy, RVLCs [17–19, 25–29] have been developed for the sake of mitigating the effects of error propagation. As the terminology suggests, RVLCs often have a symmetric construction, i.e. they have the same bit-pattern both at the beginning and at the end of RVLCs, which facilitates the instantaneous decoding in both forward and backward directions, so that we can recover the correct data with a high probability from the received data even when errors occur. To achieve this, a RVLC must satisfy both the so-called prefix and suffix conditions, which implies that no codeword is allowed to be a prefix or a suffix of any other codeword. Thus RVLCs are also known in parlance as "fix-free" codes. The conditions for the existence of RVLCs and their properties were studied in [30–32].

Apart from the above-mentioned symmetric RVLC construction, asymmetric RVLCs also exist. For the same source, a symmetric RVLC generally has a higher average code length and thus becomes less efficient than an asymmetric RVLC. However, a symmetric RVLC has the benefit of requiring a single decoding table for bidirectional decoding, while an asymmetric RVLC employs two. Hence symmetric RVLCs might be preferable for memory-limited applications. Table 2.1 shows an example of both symmetric and asymmetric RVLCs. In the following sections, we discuss the construction of RVLCs for a given source.

2.3.4.1 Construction of RVLCs

The construction of RVLCs was studied in [17, 25–29]. Most RVLC constructions commence from a Huffman code designed for the source concerned and then replace the Huffman codewords by identical-length codewords that satisfy both the above-mentioned prefix and suffix conditions as necessitated. If the number of valid reversible codewords is insufficient, longer reversible codewords have to be generated and assigned, resulting in an increased redundancy. Hence a lower code-

Table 2.1: An example of RVLCs

Symbol	Probability	C_{HUFF}	$C_{\mathrm{sym.RVLC}}$	$C_{\mathrm{asym.RVLC}}$
a	0.33	00	00	00
b	0.30	01	11	01
c	0.18	11	010	10
d	0.10	100	101	111
e	0.09	101	0110	11011
Average code length		2.19	2.46	2.37
Free distance		1	2	1

efficiency is expected. Viewing this process of mapping RVLCs to Huffman codes from a different perspective, it is plausible that at a given codeword length, we are likely to run out of legitimate RVLCs owing to their "fix-free" constraint, typically requiring length-extension. By contrast, if there are more RVLC candidate codewords than necessary, different codeword selection mechanisms may be applied. For example, the maximum symmetric-suffix length (MSSL) scheme [25] may be used for designing symmetric RVLCs, or the minimum repetition gap (MRG) [26] metric and the so-called "affix index" [28] metric may be invoked for designing asymmetric RVLCs.

The number of codewords of a certain length that satisfy both the prefix and suffix condition depends on the previously assigned (shorter) RVLC codewords. This dependency was not considered in [17], which hence resulted in reduced-efficient RVLCs. In [26] it was conjectured that the number of available codewords of any length depends on the above-mentioned MRG metric associated with the shorter RVLC codewords. The codeword assignment policy in [26] was base on MRG and typically produced more efficient codes than the algorithm advocated in [17]. In [28] the formal relationship between the number of available RVLC codewords of any length, and the structure of shorter RVLC codewords was established. Based on this relationship, the authors proposed a new codeword selection mechanism, which rendered the approach "length-by-length" optimal. However, the computational complexity imposed is generally quite high. More details of the MRG-based algorithm [26], which is denoted as Algorithm A, and the affix index-based algorithm [28] denoted as Algorithm B can be found in Appendix A.

Both Algorithm A [26] and Algorithm B [28] attempt to match the codeword length distribution of the RVLC to that of the corresponding Huffman code. Since a RVLC has to satisfy the suffix condition in addition to the prefix condition, the desirable codeword length distribution of a Huffman code is usually not matched by that of the RVLC for the same source.

Moreover, both the value of the minimal average codeword length and the optimal codeword length distribution of the RVLC are unknown for a given source. If we find the necessary and sufficient condition for the existence of a RVLC, finding the optimal codeword length distribution would be a constrained minimization problem. More explicitly, given the source symbol probabilities, find a codeword length distribution that minimises the average codeword length, while satisfying the necessary and sufficient condition at the same time. Some results related to this existence condition have been reported in [30–32, 143], however, even the existence of a general necessary and sufficient condition is uncertain. Therefore, we propose a heuristic method for the sake of finding an improved codeword length distribution.

The proposed method, i.e. Algorithm C, bases its codeword selection procedure on that of the procedures proposed in [26] or [28] and attempts to optimize the codeword length distribution of the resultant RVLC length-by-length. The construction procedure[3] is as follows:

Algorithm C:

Step 1: Employ Algorithm A or Algorithm B of Appendix A for constructing an initial RVLC and calculate the length vector n, which records the number of codewords at different lengths, i.e. the histogram of the VLC codewords' lengths.

Step 2: Increase the number of required codewords at length l by one, and decrease the number of required codewords at the maximum codeword length L_{max} by one, yielding

$$\mathrm{n}(l) := \mathrm{n}(l) + 1, \qquad \mathrm{n}(L_{max}) := \mathrm{n}(L_{max}) - 1.$$

Use Algorithm A or Algorithm B Appendix A for constructing an RVLC based on the modified length distribution.

Step 3: Recalculate the average codeword length $\bar{L}$, and if $\bar{L}$ was increased, restore the previous length distribution, then proceed to the next level; otherwise update the length vector n and go to Step 2.

Step 4: Repeat Step 2 and Step 3 until the last level is reached.

2.3.4.2 An Example Construction of a RVLC

In the following, we will demonstrate the construction procedure of our Algorithm C using an example. The specific source for which our RVLC example is designed is the English Alphabet shown in Table 2.3. The construction procedure is summarised in Table 2.2. At the first iteration, the length vector is initialised to that of the Huffman code, namely n =[0 0 2 7 7 5 1 1 1 2], and Tasi's algorithm [26] (Algorithm A) is invoked for constructing an initial RVLC, resulting in a length distribution of n =[0 0 2 7 4 3 5 2 1 2] and an average codeword length of $\bar{L} = 4.30675$. Then, our length-distribution

[3]The proposed algorithm is also applicable to the construction of the symmetric RVLCs considered in [25].

optimisation procedure commences. Firstly, the number of codewords $\mathbf{n}(3)$ required at the length of $l = 3$ is increased by one. For the sake of maintaining the total number of codewords, the number of codewords $\mathbf{n}(10)$ required at the length of $l = 10$ is decreased by one. Therefore the desired length distribution becomes $\mathbf{n}$ =[0 0 3 7 4 3 5 2 1 1], as shown in Table 2.2. Then Algorithm A is invoked again for constructing a RVLC according to the new length distribution and the resultant average codeword length becomes $\bar{L} = 4.25791$. Since the average codeword length of the new RVLC is lower than that of the old one, the code construction procedure continues to increase the number of codewords required at the length of $l = 3$ during the third iteration. However, the resultant RVLC now has an increased average codeword length of $\bar{L} = 4.26231$, hence the code construction procedure has to restore the length vector to that of the RVLC obtained at the end of the second iteration, which is $\mathbf{n}$ =[0 0 3 6 5 2 2 2 2 3 1]. Then the code construction procedure attempts to increase the number of codewords required at the next level, i.e. at the length of $l = 4$. Hence the desired length distribution at the beginning of the 4th iteration becomes $\mathbf{n}$ =[0 0 3 7 5 2 2 2 2 3 0]. This procedure is repeated, until the last level has been reached. As a result, the best RVLC having the minimum average codeword length is found at the end of the 12th iterations.

Table 2.2: An example construction of a RVLC designed for the English Alphabet. The expected length vector repesents the desired RVLC length distribution, while the resultant length vector denotes the actual length distribution of the resultant RVLC after each iteration. The arrows pointing upwards imply that the values at those positions have been increased by one, while arrows pointing downwards imply the opposite. The underlines indicate the numbers that will be changed in the next iteration. The control logic decides the next step of the code generator, which depends on the resultant average codeword length.

Iteration Number	Expected Length Vector	Resultant Length Vector	Average Length	Control Logic
#1	[0 0 2 7 7 5 1 1 1 2]	[0 0 $\underline{2}$ 7 4 3 5 2 1 $\underline{2}$]	4.30675	start
#2	[0 0 3$\uparrow$ 7 4 3 5 2 1 1$\downarrow$]	[0 0 $\underline{3}$ 6 5 2 2 2 2 3 $\underline{1}$]	4.25791	continue
#3	[0 0 4$\uparrow$ 6 5 2 2 2 2 3 0$\downarrow$]	[0 0 4 4 4 4 3 2 2 3]	4.26231	restore, next level
#4	[0 0 3 7$\uparrow$ 5 2 2 2 2 3 0$\downarrow$]	[0 0 3 $\underline{6}$ 6 2 2 2 2 2 0 $\underline{1}$]	4.18785	continue
#5	[0 0 3 7$\uparrow$ 6 2 2 2 2 2 0 0$\downarrow$]	[0 0 3 6 6 2 2 2 2 2 0 0 1]	4.18839	restore, next level
⋮	⋮	⋮	⋮	⋮
#12	[0 0 3 6 6 2 2 2 2 2 1$\uparrow$ 0$\downarrow$]	[0 0 3 6 6 2 2 2 2 2 1]	4.18732	end

2.3.4.3 Experimental Results

Table 2.3 compares the RVLCs designed for the English alphabet, constructed using the proposed algorithm (invoking Algorithm A of Appendix A for the initial construction), and the algorithms published in [17,26,28]. As a benefit of its improved length distribution, Algorithm C yields a RVLC construction of the highest efficiency. For example, compared with the algorithm in [17] and [26], the average codeword length is reduced by 2.8% and 1.5%, respectively.

Table 2.3: VLCs for the English Alphabet

u	p(u)	Huffman code	Takishima's [17] RVLC	Tsai's [26] RVLC	Lakovic's [28] RVLC	Proposed RVLC
E	0.14878	111	001	000	000	000
T	0.09351	110	110	111	001	111
A	0.08833	1011	0000	0101	0100	010
O	0.07245	1010	0100	1010	0101	0110
R	0.06872	1001	1000	0010	0110	1001
N	0.06498	1000	1010	1101	1010	1011
H	0.05831	0111	0101	0100	1011	1101
I	0.05644	0110	11100	1011	1100	0011
S	0.05537	0101	01100	0110	1101	1100
D	0.04376	01001	00010	11001	01110	10101
L	0.04124	01000	10010	10011	01111	00100
U	0.02762	00111	01111	01110	10010	01110
P	0.02575	00110	10111	10001	10011	10001
F	0.02455	00101	11111	001100	11110	00101
M	0.02361	00100	111101	011110	11111	10100
C	0.02081	00011	101101	100001	100010	011110
W	0.01868	000101	000111	1001001	100011	100001
G	0.01521	000011	011101	0011100	1000010	0111110
Y	0.01521	000100	100111	1100011	1000011	1000001
B	0.01267	000010	1001101	0111110	1110111	01111110
V	0.01160	000001	01110011	1000001	10000010	10000001
K	0.00867	0000001	00011011	00111100	10000011	011111110
X	0.00146	00000001	000110011	11000011	11100111	100000001
J	0.00080	000000001	0001101011	100101001	100000010	0111111110
Q	0.00080	0000000001	00011010011	0011101001	1000000010	1000000001
Z	0.00053	0000000000	000110100011	1001011100	1000000111	01111111110
Avg. length		4.15572	4.36068	4.30678	4.25145	4.18732
Max length		10	12	10	10	11

In [29] Lin *et al.* proposed two so-called backtracking based construction algorithms for asymmetric RVLCs. In comparison to the algorithm of [26], one of the algorithms in [29] is capable of achieving a lower average codeword length at the cost of an equal or higher maximum codeword length, while the other algorithm of [29] is capable of reducing the maximum codeword length at the expense of an increased average codeword length. Table 2.4 compares the achievable performance of the algorithm proposed here as well as that of various benchmarkers based on the Canterbury Corpus file set (available in http://corpus.canterbury.ac.nz), which was designed specifically for testing new compression algorithms. It is shown in Table 2.4 that our Algorithm C of Section 2.3.4.1 is potentially capable of reducing both the average codeword length and the maximum codeword length at the same time.

In [27] Lakovic *et al.* proposed a construction algorithm based on the MRG metric for design-

Table 2.4: VLCs for Canterbury Corpus[†]

File	Number of Codewords	Huffman Code		Tsai's [26] RVLC		Lin's [29] RVLC-1		Lin's [29] RVLC-2		Proposed RVLC	
		Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
asyoulik.txt	68	4.84465	15	5.01142	15	5.00954	15	5.13624	11	4.92273	11
alice29.txt	74	4.61244	16	4.80326	17	4.68871	18	4.86762	11	4.70569	11
xargs.1	74	4.92382	12	5.07334	13	5.16087	15	5.27537	11	5.00166	11
grammar.lsp	76	4.66434	12	4.85461	12	4.78581	17	4.96130	11	4.80247	12
plrabn12.txt	81	4.57534	19	4.80659	19	4.64910	17	4.84043	11	4.71036	12
lcet10.txt	84	4.69712	16	4.87868	16	4.74177	17	4.93372	11	4.80024	12
cp.html	86	5.26716	14	5.37113	14	5.77080	16	5.74580	11	5.29342	14
fields.c	90	5.04090	13	5.26987	13	5.20278	13	5.36233	11	5.18341	11
ptt5	159	1.66091	17	1.71814	17	1.70401	15	1.72843	13	1.69580	13
sum	255	5.36504	14	5.49767	13	6.01870	15	5.78572	13	5.47330	13
kennedy.xls	256	3.59337	12	3.89401	13	3.85384	14	3.86296	14	3.79759	14

[†] Results of Tsai's RVLCs and Lin's RVLCs are from [29].

ing RVLCs having a free distance of $d_f = 2$, which significantly outperformed the regular RVLCs having a free distance of $d_f = 1$, when using a joint source/channel decoding scheme. Since the RVLC construction of [27] was also based on the codeword length distribution of Huffman codes, the proposed optimisation procedure of Algorithm C may also be invoked in this context for generating more efficient RVLCs having a free distance of $d_f = 2$. The corresponding results are shown in Table 2.5.

Table 2.5: Free Distance 2 RVLCs for Canterbury Corpus and English Alphabet

File	Number of Codewords	Lakovic' [27] RVLC ($d_f = 2$)		Proposed RVLC ($d_f = 2$)	
		Avg. Length	Max. Length	Avg. Length	Max Length
asyoulik.txt	68	4.99469	15	4.96686	12
alice29.txt	74	4.75122	16	4.73989	11
xargs.1	74	5.09108	13	5.06624	12
grammar.lsp	76	4.79387	12	4.79118	11
plrabn12.txt	81	4.67692	19	4.65932	12
lcet10.txt	84	4.79175	16	4.78348	12
cp.html	86	5.40999	14	5.28915	14
fields.c	90	5.30430	13	5.24592	12
ptt5	159	1.73990	17	1.71901	13
sum	255	5.54921	14	5.54772	13
kennedy.xls	256	3.94279	13	3.89779	14
English Alphabet	26	4.34534	10	4.33957	8

2.3.5 Variable Length Error Correcting Code

VLEC codes [33, 34] were invented for combing source coding and error correction for employment in specific scenarios, when the assumptions of the Shannonian source and channel coding separation theorm do not apply. In terms of their construction, VLEC codes are similar to classic Huffman codes and RVLCs, in that shorter codewords are assigned to more probable source symbols. Additionally, in order to provide an error-correcting capability, VLEC codes typically have better distance properties, i.e., higher free distances in exchange for their increased average codeword length. The idea of VLEC codes were first discussed in [144], and subsequently in [145–147], where they were considered to be instantaneously decodable, resulting in sub-optimal performance [34]. In [33, 34] Buttigieg *et al.* proposed a maximum likelihood (ML) decoding algorithm, which significantly improves the decoding performance. It was shown that the free distance is the most important parameter that determines the decoding performance at high E_b/N_0. Detailed treatment on various code properties, decoding of VLEC codes and their performance bounds can be found in [35] for example.

The family of VLECs may be characterized by three different distances, namely the minimum block distance b_{min}, the minimum diverging distance d_{min}, and the minimum converging distance c_{min} [35]. It may be shown that Huffman codes and RVLCs may also be viewed as VLECs. For Huffman codes, we have $b_{min} = d_{min} = 1, c_{min} = 0$. By contrast, for RVLCs, we have $d_{min} = c_{min} = 1$ and $b_{min} = 1$ for RVLCs having a free distance of $d_f = 1$, while $b_{min} = 2$ for RVLCs having a free distance of $d_f = 2$. Table 2.6 shows two VLECs having free distances of $d_f = 3$ and $d_f = 4$, respectively.

Table 2.6: An example of VLEC

Symbol	Probability	C_{HUFF}	C_{VLEC}	C_{VLEC}
0	0.33	00	000	0000
1	0.30	01	0110	1111
2	0.18	11	1011	01010
3	0.10	100	11010	10101
4	0.09	101	110010	001100
Average code length		2.19	3.95	4.46
Free distance		1	3	4

2.3.5.1 Construction of VLEC codes

The construction of an optimal VLEC code requires finding a variable length code, which satisfies the specific distance requirements and additionally has a minimum average codeword length. As for a VLEC code having $d_f > 2$, the code structure becomes substantially different from that of the corresponding Huffman code for the same source. Therefore it is inadequate to use the length distribution of the Huffman code as a starting point for constructing a VLEC code. According to Theorem 2.1, the free distance of a VLC is determined by the minimum block distance b_{min} as well as the sum of the minimum diverging distance d_{min} and the minimum converging distance c_{min}, $(d_{min} + c_{min})$, whichever is the minimum. Hence, in order to construct a VLEC code having a given free distance of d_f, it is reasonable to choose $b_{min} = d_{min} + c_{min} = d_f$ and $d_{min} = \lceil d_f/2 \rceil$ as well as $c_{min} = d_f - d_{min}$. In [35] Buttigieg proposed a heuristic VLEC construction method, which is sumarized as follows:

Algorithm D:

Step 1: Initialize the target VLEC codeword set C to be designed to a fixed-length code of length L_1 having a minimal distance b_{min}. This step, as well as all following steps resulting in sets of identical-length codewords having a given distance, are carried out with the aid of either the greedy algorithm (GA) or the majority voting algorithm (MVA) of Appendix A.3.

Step 2: Create a set W that contains all L_1-tuples having at least a distance d_{min} from each codeword in C. If the set W is not empty, an extra bit is affixed at the end of all its words. This new set having twice the number of words replaces the previous set W.

Step 3: Delete all words in the set W that do not have at least a distance c_{min} from all codewords of C. At this point, the set W satisfies both the d_{min} and c_{min} minimum distance requirements with respect to the set C.

Step 4: Select the specific codewords from the set W that are at a distance of b_{min} using the GA or the MVA. The selected codewords are then added to set C.

The procedure of Algorithm D is repeated, until it finds the required number of codewords or has no more options to explore. If no more codewords of the required properties can be found, or if the maximal codeword length is reached, shorter codewords are deleted until one finds an adequate VLEC code structure.

Table 2.7 shows an example of a VLEC code's construction for the English Alphabet using Algorithm D. For reasons of brevity, only the results of the first four steps are listed. The target VLEC code has a free distance of $d_f = 3$, hence the minimum block distance should be at least $b_{min} = 3$ and we choose the minimum B-F diverging distance to be $d_{min} = 2$ as well as the minimum B-F converging distance to be $c_{min} = 1$. Let the minimum codeword length be $L_{min} = 4$. At the first step, the GA is invoked for generating a fixed-length code of length $l = 4$, while having a minimum block distance of $b_{min} = 3$, which is shown in the first column of Table 2.7. Our target code C is initialised with this fixed-length code. In the first part of Step 2 corresponding to column 2.1 of Table 2.7, all length-4 codewords are generated and only those having at least a distance of $d = 2$ from each codeword in the set C are assigned to a temporary codeword set W. Then, in the second part of Step 2 as depicted in column 2.2 of Table 2.7, an extra bit of either bit '0' or '1' is affixed at the end of all the codewords in the set W. At Step 3, all codewords in the set W that do not have at least a distance of $d = 1$ from all codewords of the set C are deleted, as shown in column 2.3 of Table 2.7. As a result, all codewords in the set W now satisfy both the minimum B-F diverging distance and minimum B-F converging distance requirement with respect to the codewords in the set C. Finally, at Step 4 the GA is invoked for selecting codewords from the set W that are at a distance of at least $b_{min} = 3$ from each other. The selected codewords are then incorporated in the set C, as shown in Table 2.7.

Table 2.7: An example construction of a VLEC code having a free distance of $d_f = 3$ for the English Alphabet using Algorithm D.

Set C	Set W			Set C
Step 1	Step 2.1	Step 2.2	Step 3	Step 4
0 0 0 0	1 0 0 1	1 0 0 1 0	1 0 0 1 0	0 0 0 0
0 1 1 1	1 0 1 0	1 0 0 1 1	1 0 0 1 1	0 1 1 1
	1 0 1 1	1 0 1 0 0	1 0 1 0 0	1 0 0 1 0
	1 1 0 0	1 0 1 0 1	1 0 1 0 1	1 0 1 0 1
	1 1 0 1	1 0 1 1 0	1 0 1 1 0	
	1 1 1 0	1 0 1 1 1	1 1 0 0 0	
		1 1 0 0 0	1 1 0 0 1	
		1 1 0 0 1	1 1 0 1 0	
		1 1 0 1 0	1 1 0 1 1	
		1 1 0 1 1	1 1 1 0 0	
		1 1 1 0 0	1 1 1 0 1	
		1 1 1 0 1		

The basic construction of Algorithm D attempts to assign as many short codewords as possible in order to minimize the average codeword length without considering the source statistics. However, assigning too many short codewords typically decreases the number of codewords available at a higher length. Therefore the maximum codeword length may have to be increased in order to generate the required number of codewords, which will probably result in an increased average codeword length.

Therefore Buttigieg [35] proposed an exhaustive search procedure for arriving at a better codeword length distribution. After generating a sufficiently high number of codewords using the basic construction Algorithm D, the optimization procedure starts by deleting the last group of codewords of the same length and one codeword from the next last group of codewords. The rest of the codewords are retained and the basic construction Algorithm D is invoked to add new codewords. After finding the required number of codewords, the same optimisation procedure is applied to the new code. The optimisation is repeated, until it reaches the first group of codewords (i.e., the group of codewords with the minimum length) and there is only one codeword in the first group. At every iteration, the average codeword length is calculated, and the code with minimum average codeword length is recorded. In this way, every possible codeword length distribution is tested and the "best" one is found. This technique is capable of improving the achievable code efficiency, however, its complexity increases exponentially with the size of the source alphabet.

Hence here we propose a codeword length distribution optimization procedure similar to that for the construction of RVLCs as an alternative to the exhaustive search:

Algorithm E:

Step 1: Use Algorithm D for constructing an initial VLEC code, and calculate the length vector $\mathbf{n}$.

Step 2: Decrease the number of codewords required at length l, $\mathbf{n}(l)$, by one. The values of $\mathbf{n}(k)$, $k \leq l$ are retained and no restrictions are imposed on the number of codewords at higher lengths. Then use Algorithm D again for constructing a VLEC code based on the modified length distribution.

Step 3: Recalculate the average codeword length $\bar{L}$, and if $\bar{L}$ was increased, restore the previous length distribution, and proceed to the next codeword length; otherwise update the bit length vector $\mathbf{n}$ and go to Step 2.

Step 4: Repeat Step 2 and Step 3, until the last codeword length is reached.

2.3.5.2 An Example Construction of a VLEC Code

In the following, we will demonstrate the construction procedure of our Algorithm E using an example. The source for which the VLEC code is designed is the English Alphabet as shown in Table 2.9. The construction procedure is summarised in Table 2.8, which is similar to the procedure described in Section 2.3.4.2 and devised for the construction of RVLCs, except that no length distribution restrictions are imposed beyond the current level, as well as except for the fact that the number of codewords required at the current level is expected to be decreased.

At the first iteration, the length vector is initialised to NULL, namely no restrictions are imposed, and Algorithm D is invoked for constructing an initial VLEC code, resulting in a length distribution of $\mathbf{n}$ =[0 0 0 2 1 2 1 2 5 7 6] and an average codeword length of $\bar{L} = 6.94527$. Then our length-distribution optimisation procedure commences. Firstly, the number of codewords $\mathbf{n}(4)$ required at the length of $l = 4$ is decreased by one and the expected length distribution becomes $\mathbf{n}$ =[0 0 0 1], as shown in Table 2.8. Then Algorithm D is invoked again for constructing a VLEC code according to the new length distribution and the resultant average codeword length becomes $\bar{L} = 6.64447$. Since the average codeword length of the new RVLC is lower than that of the old one, the code generator continues to increase the number of codewords required at the length of $l = 4$ during the third iteration. However, the resultant RVLC has an increased average codeword length of $\bar{L} = 6.71211$, hence the code generator has to restore the length vector to that of the VLEC code obtained at the end of the second iteration, which is $\mathbf{n} =[0\ 0\ 0\ 1\ 4\ 1\ 2\ 2\ 2\ 8\ 6]$. Then the code generator attempts to increase the number of codewords required at the next level, i.e. at the length of $l = 4$. Hence the desired length distribution devised at the beginning of the 4th iteration becomes $\mathbf{n} =[0\ 0\ 0\ 1\ 3]$. This procedure is repeated, until the last level has been reached. As seen from Table 2.8, after 12 iterations all levels of the codeword lengths have been optimised. Finally, the best VLEC code having the minimum average codeword length found at the 9th iteration is selected.

Table 2.8: An example construction of a VLEC code having a free distance of $d_f = 3$ for the English Alphabet. The expected length vector repesents the desired VLEC code length distribution, while the resultant length vector denotes the actual length distribution of the resultant VLEC code after each iteration. The arrows pointing downwards imply that the values at those positions have been decreased by one, while the underlines indicate the numbers that will be changed in the next iteration. The control logic decides the next step of the code generator, which depends on the resultant average codeword length.

Iteration Number	Expected Length Vector	Resultant Length Vector	Average Length	Control Logic
#1	NULL	$[0\ 0\ 0\ \underline{2}\ 1\ 2\ 1\ 2\ 5\ 7\ 6]$	6.94527	start
#2	$[0\ 0\ 0\ 1{\downarrow}]$	$[0\ 0\ 0\ \underline{1}\ 4\ 1\ 2\ 2\ 2\ 8\ 6]$	6.64447	continue
#3	$[0\ 0\ 0\ 0{\downarrow}]$	$[0\ 0\ 0\ 4\ 3\ 1\ 3\ 3\ 6\ 6]$	6.71211	restore, next level
#4	$[0\ 0\ 0\ 1\ 3{\downarrow}]$	$[0\ 0\ 0\ 1\ \underline{3}\ 3\ 2\ 3\ 5\ 7\ 2]$	6.39466	continue
		⋮		
#9	$[0\ 0\ 0\ 1\ 2\ 4\ 4{\downarrow}]$	$[0\ 0\ 0\ 1\ 2\ 4\ \underline{4}\ 6\ 5\ 2\ 2]$	6.2053	continue
		⋮		
#12	$[0\ 0\ 0\ 1\ 2\ 4\ 4\ 6\ 5\ 1{\downarrow}]$	$[0\ 0\ 0\ 1\ 2\ 4\ 4\ 6\ 4\ 1\ 4]$	6.21624	end

2.3.5.3 Experimental Results

In [35] Buttigieg only considered the construction of VLEC codes having $d_f > 2$. The experimental results of Table 2.9 demonstrate that the proposed Algorithm E of Section 2.3.5.1 may also be used for constructing VLEC codes having $d_f = 1, 2$, which are the equivalents of the RVLCs having $d_f = 1, 2$ in Section 2.3.4.3. It is worth noting that the VLEC code having $d_f = 2$ seen in Table 2.9 results in an even shorter average codeword length, than the corresponding RVLC having $d_f = 2$ in Table 2.5, which indicates that this is a meritorious way of constructing such codes.

Table 2.10 compares the VLEC codes having $d_f = 3, 5, 7$ constructed by the proposed Algorithm E of Section 2.3.5.1 to those of [35]. By using the optimisation procedure of Algorithm E instead of

Table 2.9: Proposed VLECs for the English Alphabet

u	p(u)	VLEC $d_f = 1$	VLEC $d_f = 2$	VLEC $d_f = 3$
E	0.14878	000	000	0000
T	0.09351	001	011	00110
A	0.08833	010	101	11000
O	0.07245	0110	110	010100
R	0.06872	0111	0010	100101
N	0.06498	1011	0100	011001
H	0.05831	1100	1001	101000
I	0.05644	1101	1111	0101110
S	0.05537	1110	00111	0110100
D	0.04376	1111	01010	1111010
L	0.04124	10011	10001	1001100
U	0.02762	10100	11100	11110010
P	0.02575	10101	001100	01111110
F	0.02455	100011	010111	11011111
M	0.02361	100100	100001	10111011
C	0.02081	100101	111010	10111100
W	0.01868	1000011	0011010	11101101
G	0.01521	1000100	0101100	011110110
Y	0.01521	1000101	1000001	110110101
B	0.01267	10000011	1110111	111110011
V	0.01160	10000100	00110111	101011110
K	0.00867	10000101	01011010	111011101
X	0.00146	100000011	10000001	1111100001
J	0.00080	100000100	11101100	1110101100
Q	0.00080	100000101	001101100	11101010010
Z	0.00053	1000000011	010110111	11101011111
Avg. length		4.19959	4.23656	6.20530
Max length		10	9	11

the exhaustive search of [35], only a small fraction of all possible codeword length distributions are tested, hence the construction complexity is significantly reduced without compromising the achievable performance. For sources of larger alphabets, the "accelerated" search techniques of [36] may be invoked for further complexity reduction.

Table 2.10: Comparison of VLECs for the English Alphabet†

	Buttigieg's VLEC [35]		Proposed VLEC	
Free Distance	Avg. Length	Max Length	Avg. Length	Max Length
$d_f = 3$	6.370	13	6.3038	11
$d_f = 5$	8.467	12	8.4752	12
$d_f = 7$	10.70	15	10.7594	15

† The probability distribution in [35] was used, which is different from that in Table 2.9.

2.4 Soft-Decoding of Variable Length Codes

Classic VLCs, such as Huffman codes and RVLCs have been widely employed in image and video coding standards, such as the JPEG/JPEG2000 [148, 149], the H.263/H.264 [23, 150] and MPEG-1/2/4 [24, 151, 152] for the sake of source compression. Traditionally, VLCs are instantaneously decoded using hard decisions, implying that after demodulation and possible channel decoding, the input of the VLC decoder is a stream of binary 0s and 1s. The VLC decoder then searches the binary code tree or look up the code table in order to decode the bit stream. If there are residual errors in the input bit stream, it is very likely that the VLC decoder will lose synchronization and hence has to discard the rest of the bit stream.

On the other hand, the demodulator or channel decoder is often capable of providing more information for the VLC decoder than the hard decision based bits, such as the probability that a transmitted bit assumes a particular value from the set $\{0, 1\}$. The principle of using probabilities (soft information) rather than hard decisions is often referred to as "soft decoding." According to one of Viterbi's three lessons on Shannonian information theory [153]: "Never discard information prematurely that may be useful in making a decision, until after all decisions related to that information have been completed." Hence, one may argue that soft-decoding could outperform hard-decoding. Before describing the soft-decoding of VLCs, we now introduce their trellis representations.

2.4.1 Trellis Representation

2.4.1.1 Symbol-level Trellis

In [44], Bauer and Hagenauer proposed a symbol-level trellis representation of VLCs, where the trellis was constructed based on two types of length information related to a VLC sequence, namely the number of source symbols (or VLC codewords) and the number of bits. Let (n, k) represent a node in the trellis of Fig. 2.2 at symbol index k, where the source symbols is represented by n bits of VLC codewords. Assume that we have a VLC sequence consisting of N bits and K symbols. Then the trellis of Fig. 2.2 diverges from node $(0, 0)$ as the symbol index k increases and converges to node (N, K) at the end of the sequence. Each path through the trellis represents a legitimate combination of K symbols described by a total of N bits.

For example, consider a VLC having the alphabet $\mathscr{C} = \{1, 01, 00\}$, and a sequence of $K = 4$ source symbols $\mathbf{u} = (0, 2, 0, 1)$. This sequence is mapped to a sequence of the VLC codewords $\mathbf{c} = (1, 00, 1, 01)$. Its length expressed in terms of bits is $N = 6$. All sequences associated with $K = 4$ and $N = 6$ can be represented in the trellis diagram shown in Fig. 2.2.

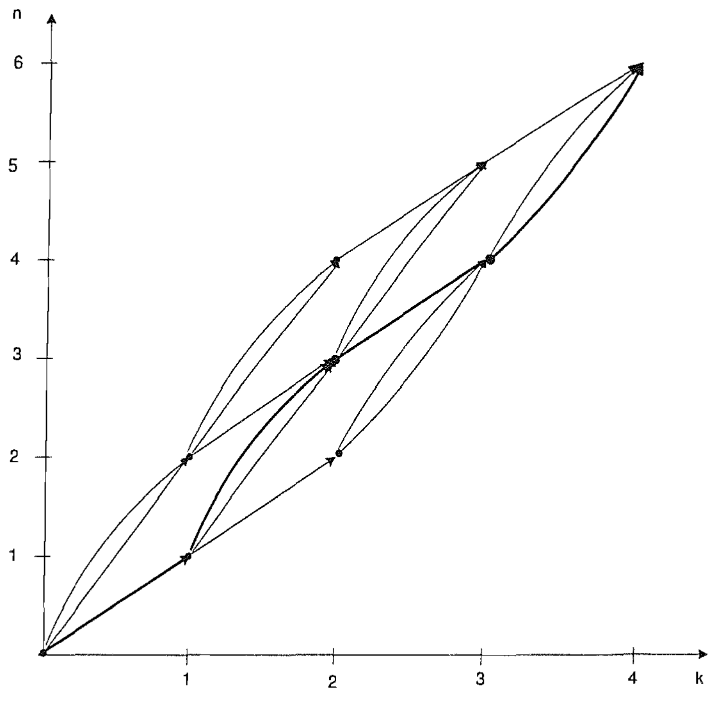


Figure 2.2: Symbol-level trellis for $K = 4$ and $N = 6$ using the VLC alphabet $\mathscr{C} = \{1, 01, 00\}$ [44].

In this diagram k denotes the symbol index and n identifies a particular trellis state at symbol index k. The value of n is equivalent to the number of bits of a subsequence ending in the corresponding

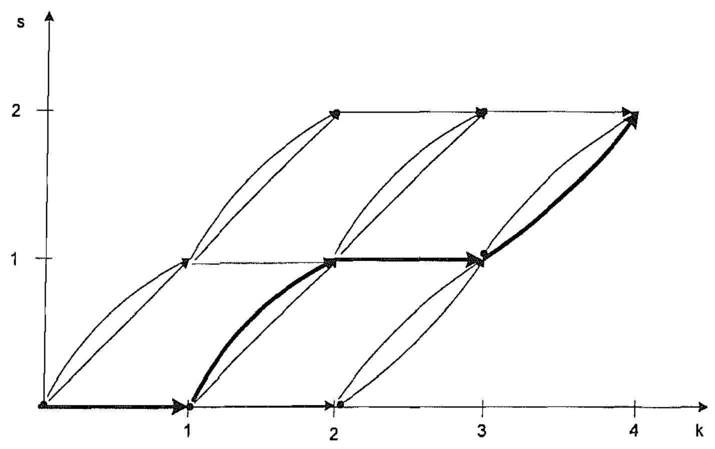


Figure 2.3: Transformed symbol-level trellis for $K = 4$ and $N = 6$ using the VLC alphabet $\mathscr{C} = \{1, 01, 00\}$ [44].

state. At each node of the trellis, the appropriate match (synchronization) between the received bits and received symbols is guaranteed. Let us now introduce the transformation

$$s = n - kl_{min}$$

for the sake of btaining the trellis seen in Fig. 2.3, where l_{min} is the minimum VLC codeword length. The maximal number of states at any symbol index k is

$$s_{max} = N - Kl_{min} + 1,$$

which is less than the original trellis of Fig. 2.2. Hence the transformed trellis of Fig. 2.3 is more convenient for implementation.

2.4.1.2 Bit-level Trellis

The idea of using a bit-level trellis representation for a VLC was proposed by Balakirsky [45], which is similar to that of a binary convolutional code.

The trellis is obtained by assigning its states to the nodes of the VLC tree. The nodes in the tree can be subdivided into a root-node (R), as well as so-called internal nodes (I) and terminal nodes (T). The root node and all the terminal nodes are assumed to represent the same state, since they all show the start of a new symbol. The internal nodes are assigned one-by-one to the other states of the trellis. As an example, Figure 2.4 shows the trellis corresponding to the RVLC-2 scheme of {00, 11, 010, 101, 0110}. Figure 2.5 shows two other examples of RVLC trellises.

The bit-level trellis of a VLC differs from that of a binary convolutional code in many ways. In

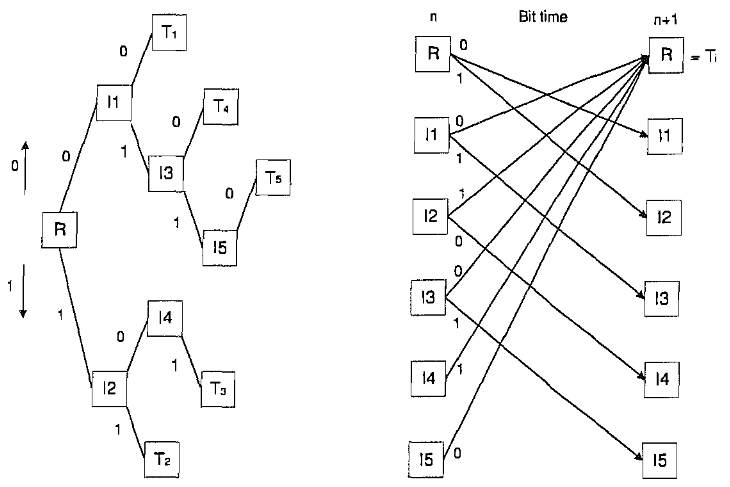


Figure 2.4: Code-tree and trellis for the RVLC-2 scheme of $\{00, 11, 010, 101, 0110\}$ [46].

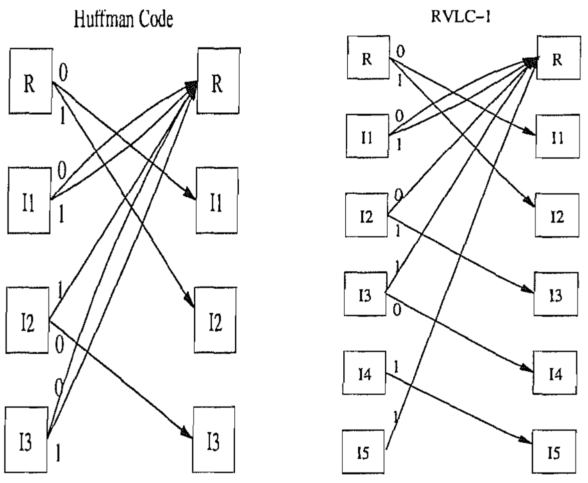


Figure 2.5: Bit-level trellises for the Huffman code $\{00, 01, 11, 100, 101\}$, and the RVLC-1 $\{00\ 01\ 10\ 111\ 11011\}$.

the bit-level trellis of a VLC the number of trellis states is equal to the number of internal nodes plus one (the root node), which depends on the specific code tree structure. The maximum number of branches leaving a trellis state is two, corresponding to a separate branch for each possible bit value. Merging of several paths only occurs in the root node, where the number of paths converging at the root node corresponds to the total number of codewords in the VLC. If the variable-length coded sequence consists of N bits, the trellis can be terminated after N sections. Table 2.11 sumarises our comparison of VLCs and convolutional codes. Furthermore, in comparison to the symbol-level trellis, the bit-level trellis is relatively simpler and time invariant.

Similar to convolutional codes, we may define the constraint length of a VLC. Since a constraint

Table 2.11: Comparison of the trellis of a VLC and that of a binary convolutional code

	VLCs	Convolutional Codes
Number of states	depends on code tree structure	depends on the size of the code memory
Number of branches emitting from a state	1 or 2	2
Number of branches emerging at a state	equal to the number of VLC codewords for the root state; 1 for other states	2
Parallel transitions	may exist	none

length-K convolutional code has 2^{K-1} states in the trellis, the constraint length of a VLC may be defined as

$$K \triangleq \log_2 N + 1, \tag{2.10}$$

where N is the number of states in the bit-level trellis.

2.4.2 Trellis Based VLC Decoding

Based on the above trellis representations, either ML/MAP sequence estimation employing the VA [51,154] or MAP decoding using the BCJR algorithm [52] can be performed. We are going to describe all these algorithms in the following sections.

2.4.2.1 Sequence Estimation Using the Viterbi Agorithm

Sequence estimation can be carried out using either the MAP or the ML criterion. The latter is actually a special case of the former. The VA [51, 154] is invoked in order to search through the trellis for the optimal sequence that meets the MAP/ML criterion. The Euclidean distance is used as the decoding metric for the case of soft-decision, while the Hamming distance is used as the decoding metric for the hard-decision scenario.

2.4.2.1.1 Symbol-level Trellis Based Sequence Estimation Consider the simple transmission scheme shown in Figure 2.6, where the K-symbol output sequence $\mathbf{u}$ of a discrete memoryless source is first VLC encoded into a N-bit sequence $\mathbf{b}$ and then modulated before transmitted over a channel.

At the receiver side, the MAP decoder selects as its estimate the particular sequence that maximises the *a posteriori* probability:

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} P(\mathbf{u}|\mathbf{y}). \tag{2.11}$$

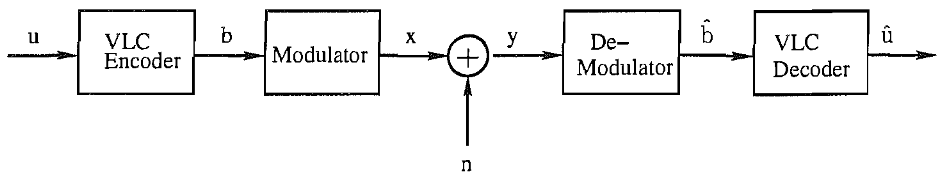


Figure 2.6: A simple transmission scheme of VLC encoded source information communicating over AWGN channels.

Since the source symbol sequence $\mathbf{u}$ is uniquely mapped to the bit sequence $\mathbf{b}$ by the VLC encoder, maximisation of the *a posteriori* probability of $P(\mathbf{u}|\mathbf{y})$ is equivalent to the maximisation of the *a posteriori* probability of $P(\mathbf{b}|\mathbf{y})$, which is fomulated as,

$$\hat{\mathbf{b}} = \arg\max_{\mathbf{b}} P(\mathbf{b}|\mathbf{y}). \tag{2.12}$$

Using Bayes' rule, we have

$$P(\mathbf{b}|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{b})P(\mathbf{b})}{P(\mathbf{y})}. \tag{2.13}$$

Note that the denominator of (2.13) is constant for all $\mathbf{b}$. Hence, to maximise (2.12), we only have to maximise the numerator of (2.13), and it might be more convenient to maximise its logarithm. Therefore, we have to choose a codeword sequence $\mathbf{b}$ which maximises

$$\log P(\mathbf{y}|\mathbf{b}) + \log P(\mathbf{b}). \tag{2.14}$$

If we assume that the channel is memoryless, the first term of (2.14) can be re-written as

$$\log P(\mathbf{y}|\mathbf{b}) = \sum_{i=1}^{N} \log P(y_i|b_i), \tag{2.15}$$

where $\mathbf{b} = b_1 b_2 \cdots b_N$, N is the length of the sequence $\mathbf{b}$ expressed in terms of the number of bits. Furthermore, assume that the source is memoryless, then the probability of occurence $P(\mathbf{b})$, for the sequence $\mathbf{b}$, is given by

$$P(\mathbf{b}) = P(\mathbf{c}_1)P(\mathbf{c}_2)\cdots P(\mathbf{c}_K), \tag{2.16}$$

where $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K$ are the codewords which constitute the sequence $\mathbf{b}$, i.e. we have $\mathbf{b} = \mathbf{c}_1\mathbf{c}_2\cdots\mathbf{c}_K$, and $P(\mathbf{c}_j)$ is determined by the corresponding source symbol probability. Therefore, based on the channel output and source information, the path metric (PM) given by (2.14) is transformed to

$$PM_{MAP} = \sum_{i=1}^{N} \log P(y_i|b_i) + \sum_{j=1}^{K} \log P(\mathbf{c}_j). \tag{2.17}$$

Let $l_j \triangleq \ell(\mathbf{c}_j)$ be the length of the codeword $\mathbf{c}_j$ in terms of bits, then the branch metric (BM) calculated for the jth branch of the path is formulated as

$$BM_{MAP} = \sum_{m=1}^{l_j} \underbrace{\log P(y_{j_m}|c_{j_m})}_{channel\ info.} + \underbrace{\log P(\mathbf{c}_j)}_{source\ info.}, \tag{2.18}$$

where we have $\mathbf{c}_j = (c_{j_1}c_{j_2}\cdots c_{j_{l_j}})$, and $(y_{j_1}y_{j_2}\cdots y_{j_{l_j}})$ represents the corresponding received bits.

Let us detail this algorithm further with the aid of an example. Assume that the source described in Fig. 2.6 is a DMS emitting three possible symbols, namely $A = \{a_0,\ a_1,\ a_2\}$, which have different probabilities of occurrence. Furthermore, the source symbol sequence $\mathbf{u}$ is encoded by the VLC encoder of Fig. 2.6, which employs the VLC table of $C = \{1, 01, 00\}$. Assume that a sequence of $K = 4$ source symbols $\mathbf{u} = (0, 2, 0, 1)$ is generated and encoded to a sequence of VLC codewords $\mathbf{c} = (1, 00, 1, 01)$, which has a length of $N = 6$ in terms of bits. This bit sequence is then modulated and transmitted over the channel. At the VLC decoder of Fig. 2.6, a symbol-level trellis may be constructed using the method described in Section 2.4.1.1, which is shown in Fig. 2.7. Essentially, the trellis seen in Fig. 2.7 is the same as that of Fig. 2.2, since we are using the same VLC and source symbol sequence for the sake of comparability. According to Eq. (2.18), the branch metric of the MAP SE algorithm can be readily exemplified as seen in Fig. 2.7 for a specific path corresponding to the VLC codeword sequence $\mathbf{c} = (1, 00, 1, 01)$.

For binary phase-shift keying (BPSK) transmission over a memoryless Additive White Gaussian Noise (AWGN) channel, the channel-induced transition probability is [13]

$$P(y_i|b_i) = \frac{1}{\sqrt{\pi N_0}} \exp\left\{-\frac{[y_i - \sqrt{E_s}(2b_i - 1)]^2}{N_0}\right\}, \tag{2.19}$$

where $N_0 = 2\sigma^2$ is the single-sided power spectral density of the AWGN, σ^2 is the noise variance, E_s is the transmitted signal energy per channel symbol. Consequently, upon substituting Eq. (2.19) into Eq. (2.17), the path metric to be maximised is

$$PM_{MAP} = -\sum_{i=1}^{N} \frac{[y_i - \sqrt{E_s}(2b_i - 1)]^2}{N_0} + \sum_{j=1}^{K} \log\ P(\mathbf{c}_j). \tag{2.20}$$

Upon removing the common components in (2.20), which may be neglected during the optimisation, (2.20) can be further simplified to

$$PM_{MAP} = \sum_{i=1}^{N} \frac{L_c}{2} y_i(2b_i - 1) + \sum_{j=1}^{K} \log\ P(\mathbf{c}_j), \tag{2.21}$$

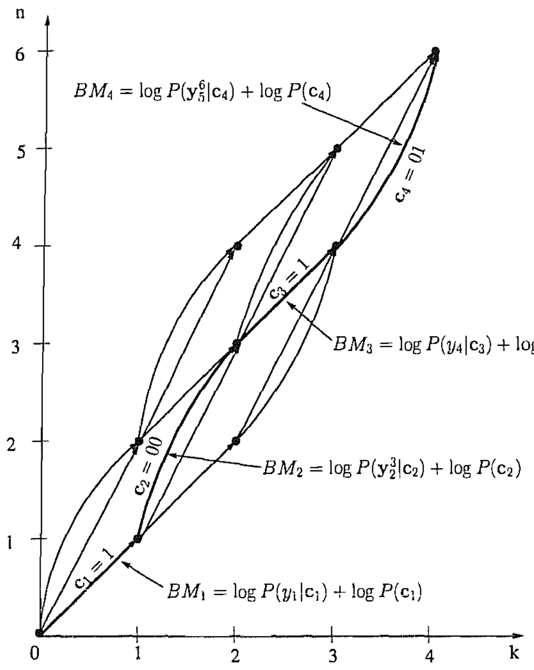

n
6
5
4
3
2
1
0
1
2
3
4
k
$BM_4 = \log P(\mathbf{y}_5^6|\mathbf{c}_4) + \log P(\mathbf{c}_4)$
$\mathbf{c}_4 = 01$
$\mathbf{c}_3 = 1$
$BM_3 = \log P(y_4|\mathbf{c}_3) + \log P(\mathbf{c}_3)$
$\mathbf{c}_2 = 00$
$BM_2 = \log P(\mathbf{y}_2^3|\mathbf{c}_2) + \log P(\mathbf{c}_2)$
$\mathbf{c}_1 = 1$
$BM_1 = \log P(y_1|\mathbf{c}_1) + \log P(\mathbf{c}_1)$

Figure 2.7: An example of the branch metric calculation for the MAP SE algorithm employed in the VLC decoder of Fig. 2.6. The VLC table employed is $C = \{1, 01, 00\}$. The transmitted VLC codeword sequence has a length of $N = 6$ bits, which represents $K = 4$ source symbols. The received channel observation is $\mathbf{y} = y_1 \cdots y_6$.

where

$$L_c = \frac{4\sqrt{E_s}}{N_0} \tag{2.22}$$

is the channel reliability value.

Correspondingly, the branch metric of Eq. (2.18) can be expressed as

$$BM_{MAP} = \sum_{m=1}^{l_j} \frac{L_c}{2} y_{j_m}(2c_{j_m} - 1) + \log\ P(\mathbf{c}_j). \tag{2.23}$$

If we assume that all paths in the trellis are equiprobable, i.e. the probability of occurrence $P(\mathbf{b})$ of sequence $\mathbf{b}$, is equiprobable, then the MAP decoder becomes a ML decoder, and the path metric to be maximised is reduced from the expression seen in (2.14) to

$$PM_{ML} = \log\ P(\mathbf{y}|\mathbf{b}). \tag{2.24}$$

Particularly, for BPSK transmission over a memoryless AWGN channel, the path metric formulated for the ML decoder is

$$PM_{ML} = \sum_{i=1}^{N} \frac{L_c}{2} y_i(2b_i - 1), \tag{2.25}$$

and the corresponding branch metric is given by

$$BM_{ML} = \sum_{m=1}^{l_j} \frac{L_c}{2} y_{j_m}(2c_{j_m} - 1). \tag{2.26}$$

As to hard-decision decoding, the input of the MAP/ML VLC decoder is a bit sequence $\hat{\mathbf{b}}$. The path metric of the MAP decoder is

$$PM_{MAP} = \log\ P(\hat{\mathbf{b}}|\mathbf{b}) + \log\ P(\mathbf{b}). \tag{2.27}$$

In the case of hard decisions and BPSK modulation, a memoryless AWGN channel is reduced to a Binary Symmetric Channel (BSC) having a crossover probability p. Therefore, we have

$$P(\hat{\mathbf{b}}|\mathbf{b}) = p^h(1-p)^{N-h}, \tag{2.28}$$

where $h = H(\mathbf{b}, \hat{\mathbf{b}})$ is the Hamming distance between the transmitted bit sequence $\mathbf{b}$ and the received bit sequence $\hat{\mathbf{b}}$.

Hence, the path metric is formulated as

$$PM_{MAP} = h(\log\ p - \log(1-p)) + \sum_{j=1}^{K} \log\ P(\mathbf{c}_j), \tag{2.29}$$

while the branch metric is given by

$$BM_{MAP} = (\log\ p - \log(1-p))H(\mathbf{c}_j, \hat{b}_{j_1}\hat{b}_{j_2}\cdots\ \hat{b}_{j_{l_j}}) + \log\ P(\mathbf{c}_j). \tag{2.30}$$

As for the ML decoding, assuming that we have $0 \leq\ p < 0.5$ and all transmitted sequences are equiprobable, the branch metric is simply the Hamming distance between the transmitted codeword and the received codeword, namely $H(\mathbf{c}_j, \hat{b}_{j_1}\hat{b}_{j_2}\cdots\ \hat{b}_{j_{l_j}})$.

2.4.2.1.2 Bit-level Trellis Based Sequence Estimation

Consider the same transmission scheme as in the symbol-level trellis scenario of Fig. 2.6. It is readily seen that the path metric of MAP/ML decoding in the bit-level trellis scenario is the same as that in the symbol-level trellis case, e.g. the path metrics derived in Eq. (2.17), (2.21) and (2.25). By contrast, since there is only a single bit associated with each transition in the bit-level trellis, the branch metric is different from that derived for the symbol-level trellis such as Eq. (2.18), (2.23) and (2.26). Generally, there are two convenient ways of calculating the branch metric of a bit-level VLC trellis.

For a bit-level trellis each transition ending in the root node represents the termination or completion of a legitimate source codeword and the merging only occurs at the root node. Hence for transitions that are not ending at the root node, the source symbol probabilities $P(\mathbf{c}_j)$ need not be included in the branch metric. Instead, they are added only when calculating branch metrics for the transitions that do end at the root node.

To be specific, for the soft MAP decoding, the branch metrics of the transitions that do not end at the root node of Fig. 2.4 are

$$BM_{MAP}^{non-root} = \log P(y_i|b_i), \tag{2.31}$$

which may be expressed as

$$BM_{MAP}^{non-root} = \frac{L_c}{2} y_i(2b_i - 1), \tag{2.32}$$

in the case of BPSK transmission over a memoryless AWGN channel. By contrast, the branch metrics of the transitions ending at the root node are

$$BM_{MAP}^{root} = \log P(y_i|b_i) + \log P(\mathbf{c}_j), \tag{2.33}$$

or

$$BM_{MAP}^{root} = \frac{L_c}{2} y_i(2b_i - 1) + \log P(\mathbf{c}_j), \tag{2.34}$$

in the case of BPSK transmission over a memoryless AWGN channel, where $\mathbf{c}_j$ is the source symbol corresponding to the transition that ends at the root node.

The second way of calculating the branch metric of a bit-level trellis is to align the source information with each of the trellis transitions. For example, consider the VLC $\mathscr{C} = \{00, 11, 101, 010, 0110\}$ used in Fig. 2.4 and assume that the corresponding codeword probabilities are $\mathscr{P} = \{p_1, p_2, p_3, p_4, p_5\}$. The probability of the codeword '010' being transmitted can be expressed as

$$P('010') = P('0')P('1'/'0')P('0'/'01'). \tag{2.35}$$

Furthermore, $P('0'), P('1'/'0'), P('0'/'01')$ can be calculated as the state transition probabilities (STP) of the trellis as defined in [155]. In this example, as seen from the code tree of Fig. 2.4, we have

$$P('0') = P(s_{k+1} = \mathrm{I1}, b_k = 0|s_k = \mathrm{R}) = p_1 + p_4 + p_5, \tag{2.36}$$

$$P('1'/'0') = P(s_{k+1} = \mathrm{I3}, b_k = 1|s_k = \mathrm{I1}) = \frac{p_4 + p_5}{p_1 + p_4 + p_5}, \tag{2.37}$$

$$P('0'/'01') = P(s_{k+1} = \mathrm{R(T4)}, b_k = 0|s_k = \mathrm{I3}) = \frac{p_4}{p_4 + p_5}. \tag{2.38}$$

Hence, we arrive at

$$P('010') = (p_1 + p_4 + p_5) \cdot \frac{p_4 + p_5}{p_1 + p_4 + p_5} \cdot \frac{p_4}{p_4 + p_5} = p_4. \tag{2.39}$$

For a generic VLC trellis, the state transition probabilities between two adjacent states associated with an input bit i can be formulated as

$$P(s_{k+1} = n, b_k = i | s_k = m) = \frac{\sum_{\alpha \in g(s_{k+1})} p_\alpha}{\sum_{\beta \in g(s_k)} p_\beta}, \tag{2.40}$$

where $g(s_k)$ represents all the codeword indices associated with node s_k in the code tree.

Hence, the branch metric of soft MAP decoding may be expressed as

$$BM_{MAP} = \log P(y_i|b_i) + \log P(s_{i+1}, b_i|s_i), \tag{2.41}$$

or

$$BM_{MAP} = \frac{L_c}{2} y_i(2b_i - 1) + \log P(s_{i+1}, b_i|s_i), \tag{2.42}$$

when considering BPSK transmission over a memoryless AWGN channel.

More explicitly, we exemplify this algorithm in Fig. 2.8. Consider again the transmission scheme seen in Fig. 2.7, except that the VLC decoder employs a bit-level trellis constructed by the method described in Section 2.4.1.2. As seen from Fig. 2.8, the branch metrics corresponding to the different state transtitions can be readily computed accordig to Eq. (2.41).

For ML soft decoding, source information is not considered for the sake of reducing complexity. Hence the branch metric is simply expressed as

$$BM_{ML} = \log P(y_i|b_i), \tag{2.43}$$

and we have

$$BM_{ML} = \frac{L_c}{2} y_i(2b_i - 1), \tag{2.44}$$

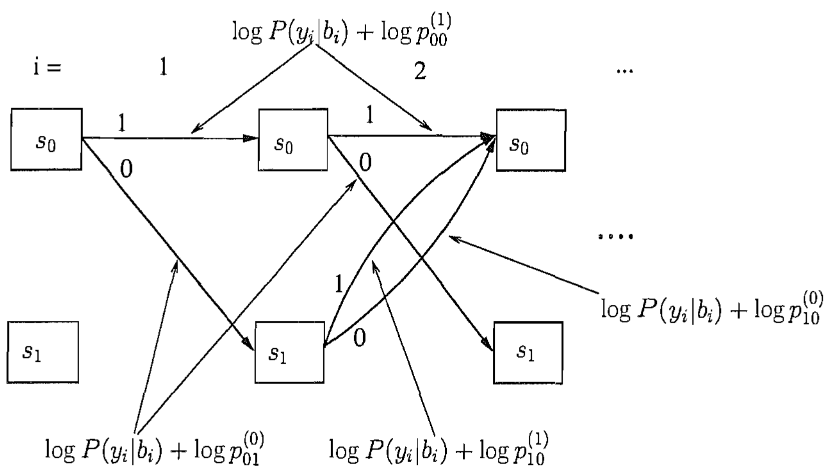
for BPSK transmission over a memoryless AWGN channel.

Similarly, for hard-decision decoding, given the crossover probability p, the branch metric of MAP decoding is expressed as

$$BM_{MAP} = (\log p - \log(1-p) H(b_i, \hat{b}_i) + \log P(s_{i+1}, b_i|s_i), \tag{2.45}$$

while it is

$$BM_{ML} = -H(b_i, \hat{b}_i) \tag{2.46}$$



i =
1
2
...
$\log P(y_i|b_i) + \log p_{00}^{(1)}$
s_0
s_0
s_0
s_1
s_1
s_1
1
0
1
0
1
0
....
$\log P(y_i|b_i) + \log p_{10}^{(0)}$
$\log P(y_i|b_i) + \log p_{01}^{(0)}$
$\log P(y_i|b_i) + \log p_{10}^{(1)}$

Figure 2.8: An example of the branch metric calculation for the bit-level trellis based MAP SE algorithm employed in the VLC decoder of Fig. 2.6. The VLC table employed is $C = \{1, 01, 00\}$. The probability $P(y_i|b_i)$ is the channel-induced transition probability for the i-th transmitted bit and the state transtion probabilities are defined as $p_{00}^{(1)} \triangleq P(s_0, b_i = 1|s_0), p_{01}^{(0)} \triangleq P(s_1, b_i = 0|s_0), p_{10}^{(1)} \triangleq P(s_0, b_i = 1|s_1), p_{10}^{(0)} \triangleq P(s_0, b_i = 0|s_1)$, which can be calculated according to Eq. (2.40), given a specific source symbol distribution.

for ML decoding.

2.4.2.2 Symbol-by-Symbol MAP Decoding

2.4.2.2.1 Symbol-level Trellis Based MAP Decoding

In [44] Bauer and Hagenauer applied the BCJR alogrithm [52] to the symbol-level trellis in order to perform symbol-by-symbol MAP decoding and hence minimisze the SER. The objective of the decoder then is to determine the APPs of the transmitted symbols u_k, $(1 \leq k \leq K)$ from the observed sequence y according to

$$u_k = \arg\max_{u_k} P(u_k|\mathrm{y}). \tag{2.47}$$

Using Bayes' rule, we can write

$$\begin{aligned} P(u_k = i|\mathrm{y}) &= \frac{P(u_k = i, \mathrm{y})}{P(\mathrm{y})} \\ &= \frac{P(u_k = i, \mathrm{y})}{\sum\limits_{i=0}^{M-1} P(u_k = i, \mathrm{y})}, \end{aligned} \tag{2.48}$$

where M is the cardinality of the source alphabet. The numerator of (2.48) can be re-written as

$$P(u_k = i, \mathrm{y}) = \sum_{n' \in \mathcal{N}_{k-1}} \sum_{n \in \mathcal{N}_k} P(s_{k-1} = n', s_k = n, u_k = i, \mathrm{y}), \tag{2.49}$$

where s_{k-1} and s_k are the states of the source encoder trellis at symbol index $k-1$ and k, respectively, $\mathcal{N}_{k-1}$ and $\mathcal{N}_k$ are the sets of possible states at symbol index $(k-1)$ and k, respectively. The state transition of n' to n is encountered, when the input symbol is $u_k = i$. Furthermore, the received sequence y can be split into three sections, namely the received codeword $y_{n'+1}^{n}$ associated with the present transition, the received sequence $y_1^{n'}$ prior to the present transition, and the received sequence y_{n+1}^{N} after the present transition. The received sequence after the split can be seen in Fig. 2.9.

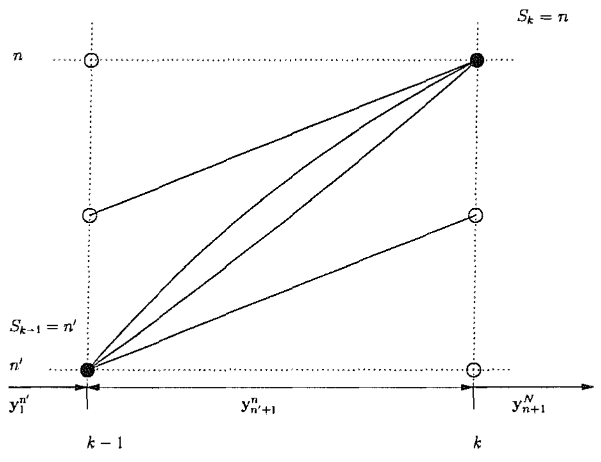


Figure 2.9: A section of transitions from the trellis in Fig. 2.2

Consequently, the probability $P(s_{k-1} = n', s_k = n, u_k = i, \mathbf{y})$ in (2.49) can be written as:

$$P(s_{k-1} = n', s_k = n, u_k = i, \mathbf{y}) = P(s_{k-1} = n', s_k = n, u_k = i, \mathbf{y}_1^{n'}, \mathbf{y}_{n'+1}^{n}, \mathbf{y}_{n+1}^{N}). \quad (2.50)$$

Again, using Bayes' rule and the assumption that the channel is memoryless, we can expand Equation (2.50) as follows:

$$\begin{aligned} P(s_{k-1} = n', s_k = n, u_k = i, \mathbf{y}) &= P(\mathbf{y}_{n+1}^{N} | s_k = n) \cdot P(s_{k-1} = n', s_k = n, u_k = i, \mathbf{y}_1^{n'}, \mathbf{y}_{n'+1}^{n}) \\ &= P(\mathbf{y}_{n+1}^{N} | s_k = n) \cdot \\ &\quad P(s_k = n, u_k = i, \mathbf{y}_{n'+1}^{n} | s_{k-1} = n') \cdot \\ &\quad P(s_{k-1} = n', \mathbf{y}_1^{n'}) \\ &= \beta_k(n) \gamma_k(i, n', n) \alpha_{k-1}(n'). \end{aligned} \quad (2.51)$$

The basic operations of the decoder are the classic forward recursion [74] required for determining the values

$$\begin{aligned}
\alpha_k(n) &= P(s_k = n, \mathbf{y}_1^n) \\
&= \sum_{n' \in \mathcal{N}_{k-1}} \sum_{i=0}^{M-1} P(s_k = n, \mathbf{y}_1^n, u_k = i, s_{k-1} = n') \\
&= \sum_{n' \in \mathcal{N}_{k-1}} \sum_{i=0}^{M-1} P(s_k = n, u_k = i, \mathbf{y}_{n'+1}^n | s_{k-1} = n') P(s_{k-1} = n', \mathbf{y}_1^{n'}) \\
&= \sum_{n' \in \mathcal{N}_{k-1}} \sum_{i=0}^{M-1} \gamma_k(i, n', n) \cdot \alpha_{k-1}(n'), \qquad (2.52)
\end{aligned}$$

and the backward recursion [74] invoked for obtaining the values

$$\begin{aligned}
\beta_k(n) &= P(\mathbf{y}_{n+1}^N | s_k = n) \\
&= \sum_{n'' \in \mathcal{N}_{k+1}} \sum_{i=0}^{M-1} P(\mathbf{y}_{n+1}^N, u_{k+1} = i, s_{k+1} = n'' | s_k = n) \\
&= \sum_{n'' \in \mathcal{N}_{k+1}} \sum_{i=0}^{M-1} P(\mathbf{y}_{n''+1}^N | s_{k+1} = n'') P(s_{k+1} = n'', u_{k+1} = i, \mathbf{y}_{n+1}^{n''} | s_k = n) \\
&= \sum_{n'' \in \mathcal{N}_{k+1}} \sum_{i=0}^{M-1} \gamma_{k+1}(i, n, n'') \cdot \beta_{k+1}(n''). \qquad (2.53)
\end{aligned}$$

Both quantities have to be calculated for all symbol indices k and for all possible states $n \in \mathcal{N}_k$ at symbol index k. For carrying out the recursions of Eq. (2.52) and (2.53) we also need the probability

$$\begin{aligned}
\gamma_k(i, n', n) &= P(u_k = i, s_k = n, \mathbf{y}_{n'+1}^n | s_{k-1} = n') \\
&= P(\mathbf{y}_{n'+1}^n | u_k = i) \cdot P(u_k = i, s_k = n | s_{k-1} = n'), \qquad (2.54)
\end{aligned}$$

which includes the source *a priori* information in terms of the state transition probability $P(u_k = i, s_k = n | s_{k-1} = n')$ and the channel characteristics quantified in terms of the channel transition probability $P(\mathbf{y}_{n'+1}^n | u_k = i)$.

Furthermore, assuming that the source is memoryless, we have

$$P(u_k = i, s_k = n | s_{k-1} = n') = \begin{cases} P(u_k = i) & n' \rightarrow n \text{ is a valid transition,} \\ 0 & \text{otherwise.} \end{cases} \qquad (2.55)$$

Additionally, for a memoryless channel we can express the channel-induced transition probability seen in Eq. (2.54) as the product of the bitwise transition probabilities:

$$P(\mathbf{y}_{n'+1}^{n}|u_k = i) = \prod_{j=1}^{|\mathbf{c}_i|} P(y_{n'+j}|c_{i_j}). \tag{2.56}$$

For BPSK transmission over an AWGN channel, the bitwise transition probability is given by [13]:

$$P(y|x = \pm 1) = \frac{1}{\sqrt{\pi N_0}} \exp\left(- \frac{(y - \sqrt{E_s}x)^2}{N_0} \right), \tag{2.57}$$

where E_s is the transmitted energy per channel symbol and N_0 is the single-sided power spectral density of the AWGN.

So far we have obtained all the quantities that we need for computing the APPs of the information symbols u_k, which are summarized as follows:

$$P(u_k = m|\mathbf{y}) = \frac{\sum\limits_{n\in\mathcal{N}_k} \sum\limits_{n'\in\mathcal{N}_{k-1}} \gamma_k(m, n', n) \cdot \tilde{\alpha}_{k-1}(n') \cdot \tilde{\beta}_k(n)}{\sum\limits_{n\in\mathcal{N}_k} \sum\limits_{n'\in\mathcal{N}_{k-1}} \sum\limits_{i=0}^{M-1} \gamma_k(i, n', n) \cdot \tilde{\alpha}_{k-1}(n') \cdot \tilde{\beta}_k(n)}, \tag{2.58}$$

where, instead of using $\alpha_k(n)$ and $\beta_k(n)$, we used their normalized version $\tilde{\alpha}_k(n)$ and $\tilde{\beta}_k(n)$ to avoid numerical problems such as overflows and underflows. This is because both $\alpha_k(n)$ and $\beta_k(n)$ drop toward zero exponentially. For a sufficiently large value of K, the dynamic range of these quantities will exceed the number-represenation range of conventional processors. To obtain a numerically stable algorithm, these quantities must be scaled as their computation proceeds. Consequently, the forward recursion $\tilde{\alpha}_k(n)$ can be written as:

$$\tilde{\alpha}_k(n) = \frac{\sum\limits_{n'\in\mathcal{N}_{k-1}} \sum\limits_{i=0}^{M-1} \gamma_k(i, n', n) \cdot \tilde{\alpha}_{k-1}(n')}{\sum\limits_{n\in\mathcal{N}_k} \sum\limits_{n'\in\mathcal{N}_{k-1}} \sum\limits_{i=0}^{M-1} \gamma_k(i, n', n) \cdot \tilde{\alpha}_{k-1}(n')}, \tag{2.59}$$

with $\tilde{\alpha}_0(0) = 1$ and $k = 1, \ldots, K$, while the backward recursion $\tilde{\beta}_k(n)$ can be expressed as:

$$\tilde{\beta}_k(n) = \frac{\sum\limits_{n''\in\mathcal{N}_{k+1}} \sum\limits_{i=0}^{M-1} \gamma_k(i, n, n'') \cdot \tilde{\beta}_{k+1}(n'')}{\sum\limits_{n\in\mathcal{N}_k} \sum\limits_{n''\in\mathcal{N}_{k+1}} \sum\limits_{i=0}^{M-1} \gamma_k(i, n, n'') \cdot \tilde{\alpha}_k(n)} \tag{2.60}$$

with $\tilde{\beta}_K(N) = 1$ and $k = K - 1, \ldots, 1$.

Furthermore, the term in the form of $\gamma(i,n,n')$ in (2.58), (2.59), and (2.60) can be computed using (2.54). Note that the scaling or normalization operations used in (2.59) and (2.60) do not affect the result of the APP computation, since they are cancelled out in Equation (2.58).

Further simplification of (2.58) can be achieved by using the logarithms of the above quantities instead of the quantities themselves [74]. Let $A_k(n)$, $B_k(n)$ and $\Gamma_k(i,n',n)$ be defined as follows:

$$A_k(n) \triangleq \ln(\tilde{\alpha}_k(n)), \tag{2.61}$$

$$B_k(n) \triangleq \ln(\tilde{\beta}_k(n)), \tag{2.62}$$

and

$$\Gamma_k(i,n',n) \triangleq \ln(\gamma_k(i,n',n)). \tag{2.63}$$

Then, we have

$$\begin{aligned} A_k(n) &= \ln\left(\frac{\sum_{n'\in\mathcal{N}_{k-1}}\sum_{i=0}^{M-1}\gamma_k(i,n',n)\cdot\tilde{\alpha}_{k-1}(n')}{\sum_{n\in\mathcal{N}_k}\sum_{n'\in\mathcal{N}_{k-1}}\sum_{i=0}^{M-1}\gamma_k(i,n',n)\cdot\tilde{\alpha}_{k-1}(n')}\right) \\ &= \ln\left(\sum_{n'\in\mathcal{N}_{k-1}}\sum_{i=0}^{M-1}\exp[\Gamma_k(i,n',n)+A_{k-1}(n')]\right) \\ &\quad -\ln\left(\sum_{n\in\mathcal{N}_k}\sum_{n'\in\mathcal{N}_{k-1}}\sum_{i=0}^{M-1}\exp[\Gamma_k(i,n',n)+A_{k-1}(n')]\right), \end{aligned} \tag{2.64}$$

$$\begin{aligned} B_k(n) &= \ln\left(\frac{\sum_{n''\in\mathcal{N}_{k+1}}\sum_{i=0}^{M-1}\gamma_{k+1}(i,n,n'')\cdot\tilde{\beta}_{k+1}(n'')}{\sum_{n\in\mathcal{N}_k}\sum_{n''\in\mathcal{N}_{k+1}}\sum_{i=0}^{M-1}\gamma_{k+1}(i,n,n'')\cdot\tilde{\beta}_{k+1}(n'')}\right) \\ &= \ln\left(\sum_{n''\in\mathcal{N}_{k+1}}\sum_{i=0}^{M-1}\exp[\Gamma_{k+1}(i,n,n'')+B_{k+1}(n'')]\right) \\ &\quad -\ln\left(\sum_{n\in\mathcal{N}_k}\sum_{n''\in\mathcal{N}_{k+1}}\sum_{i=0}^{M-1}\exp[\Gamma_{k+1}(i,n,n'')+B_{k+1}(n'')]\right), \end{aligned} \tag{2.65}$$

and

$$\begin{aligned}\Gamma_k(i,n',n) &= \ln\left(P(\mathbf{y}_{n'+1}^{n}|u_k=i)\cdot P(u_k=i)\right)\\ &= \ln P(\mathbf{y}_{n'+1}^{n}|u_k=i)+\ln P(u_k=i)\\ &= -\sum_{j=1}^{l(\mathbf{c}_i)}\left(\frac{1}{2}\ln\pi N_0+\frac{(y_{n'+j}-\sqrt{E_s}(2c_{i_j}-1))^2}{N_0}\right)+\ln P(u_k=i).\end{aligned} \tag{2.66}$$

Finally, the logarithm of the *a posteriori* probability $P(u_k=m|\mathbf{y})$ can be written as:

$$\begin{aligned}\ln P(u_k=m|\mathbf{y}) &= \ln\left(\frac{\sum_{n\in\mathcal{N}_k}\sum_{n'\in\mathcal{N}_{k-1}}\gamma_k(m,n',n)\cdot\tilde{\alpha}_{k-1}(n')\cdot\tilde{\beta}_k(n)}{\sum_{n\in\mathcal{N}_k}\sum_{n'\in\mathcal{N}_{k-1}}\sum_{i=0}^{M-1}\gamma_k(i,n',n)\cdot\tilde{\alpha}_{k-1}(n')\cdot\tilde{\beta}_k(n)}\right) \qquad (2.67)\\ &= \ln\left(\sum_{n\in\mathcal{N}_k}\sum_{n'\in\mathcal{N}_{k-1}}\exp[\Gamma_k(m,n',n)+A_{k-1}(n')+B_k(n)]\right)\\ &\quad-\ln\left(\sum_{n\in\mathcal{N}_k}\sum_{n'\in\mathcal{N}_{k-1}}\sum_{i=0}^{M-1}\exp[\Gamma_k(i,n',n)+A_{k-1}(n')+B_k(n)]\right).\end{aligned}$$

The logarithms in these quantities can be calculated using either the Max-Log approximation [156]:

$$\ln\left(\sum_i e^{x_i}\right)\approx\max_i(x_i), \tag{2.68}$$

or the Jacobian logarithm [156]:

$$\begin{aligned}\ln(e^{x_1}+e^{x_2}) &= \max(x_1,x_2)+\ln\left(1+e^{-|x_1-x_2|}\right)\\ &= \max(x_1,x_2)+f_c(|x_1-x_2|)\\ &= g(x_1,x_2),\end{aligned} \tag{2.69}$$

where the correction function $f_c(x)$ can be approximated by [156]

$$f_c(x) = \begin{cases} 0.65 & 0 < x \leq 0.2 \\ 0.55 & 0.2 < x \leq 0.43 \\ 0.45 & 0.43 < x \leq 0.7 \\ 0.35 & 0.7 < x \leq 1.05 \\ 0.25 & 1.05 < x \leq 1.5 \\ 0.15 & 1.5 < x \leq 2.25 \\ 0.05 & 2.25 < x \leq 3.7 \\ 0 & 3.7 < x, \end{cases} \tag{2.70}$$

and upon recursively using $g(x_1, x_2)$, we have

$$\ln\left(\sum_{i=1}^{N} e^{x_i}\right) = g(x_N, g(x_{N-1}, \cdots, g(x_3, g(x_2, x_1)))). \tag{2.71}$$

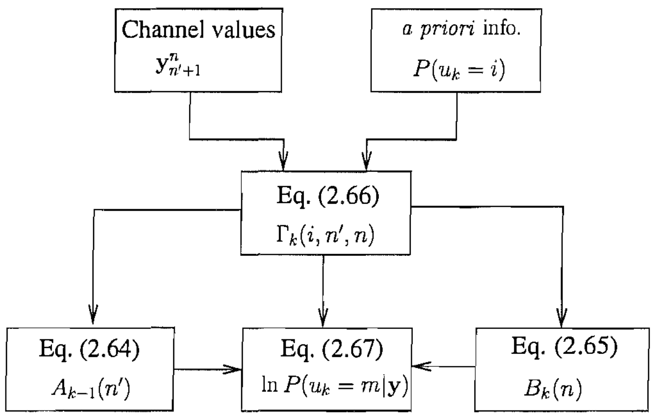


Figure 2.10: Summary of the symbol-level trellised based MAP decoding operations.

Let us summarise the operations of the MAP decoding algorithm using Fig. 2.10. The MAP decoder processes the channel observation sequence y and the source symbol probabilities $P(u_k)$ as its input and calculates the transition probability $\Gamma_k(i, n', n)$ for each of the trellis branches using Eq. (2.66). Then both forward- and backward-recursions are carried out in order to calculate the quantities of $A_k(n)$ and $B_k(n)$ for each of the trellis states, using Eq. (2.64) and (2.65), respectively. Finally, the logarithm of the *a posteriori* probability $P(u_k = m|\mathrm{y})$ is computed using Eq. (2.68).

2.4.2.2.2 Bit-level Trellis Based MAP Decoding

Similar to the MAP decoding algorithm based on the symbol-level trellis e.g. Fig. 2.2, the MAP decoding algorithm based on the bit-level trellis such as Fig. 2.4 can also be formulated. The forward and backward recursions are defined as follows [46]:

$$\begin{aligned}\alpha_n(s) &= P(s_n = s, \mathbf{y}_1^n) \\ &= \sum_{all\ s'} \sum_{i=0}^{1} \gamma_n(i, s', s) \cdot \alpha_{n-1}(s'),\end{aligned} \tag{2.72}$$

with $\alpha_0(0) = 1$ and $\alpha(s \neq 0) = 0$, as well as

$$\begin{aligned}\beta_n(s) &= P(\mathbf{y}_{n+1}^N | s_n = s) \\ &= \sum_{all\ s'} \sum_{i=0}^{1} \gamma_{n+1}(i, s, s') \cdot \beta_{n+1}(s'),\end{aligned} \tag{2.73}$$

with $\beta_N(0) = 1$ and $\beta_N(s \neq 0) = 0$. Furthermore, we have

$$\begin{aligned}\gamma_n(i, s', s) &= P(y_n, b_n = i, s_n = s | s_{n-1} = s') \\ &= P(y_n | b_n = i) \cdot P(b_n = i, s_n = s | s_{n-1} = s'),\end{aligned} \tag{2.74}$$

where b_n is the input bit necessary to trigger the transition from state $s_{n-1} = s'$ to state $s_n = s$. The first term in (2.74) is the channel-induced transition probability. For BPSK transmission over a memoryless AWGN channel, this term can be expressed as [74]:

$$\begin{aligned}P(y_n|b_n) &= \frac{1}{\sqrt{\pi N_0}} \exp\left(- \frac{(y_n - \sqrt{E_s}(2b_n - 1))^2}{N_0} \right) \\ &= \frac{1}{\sqrt{\pi N_0}} \cdot \exp\left(- \frac{E_s}{N_0} \right) \cdot \exp\left(- \frac{y_n^2}{N_0} \right) \cdot \exp\left(\frac{2\sqrt{E_s}}{N_0} y_n (2b_n - 1) \right) \\ &= C \cdot \exp\left(\frac{L_c}{2} y_n (2b_n - 1) \right),\end{aligned} \tag{2.75}$$

where

$$C = \frac{1}{\sqrt{\pi N_0}} \cdot \exp\left(- \frac{E_s}{N_0} \right) \cdot \exp\left(- \frac{y_n^2}{N_0} \right) \tag{2.76}$$

is the same for both $b_n = 0$ and $b_n = 1$.

The second term in (2.74) is the *a priori* source information, which can be calculated using Equation (2.40) of Section 2.4.2.1.2.

According to (2.74) and (2.75), we have

$$\gamma_k(i, s', s) = C \cdot \exp\left(\frac{L_c}{2} y_n (2b_n - 1) \right) \cdot P(b_n = i, s_n = s | s_{n-1} = s'). \tag{2.77}$$

Finally, from Eq. (2.72) to (2.77) the conditional Log-Likelihood-Ratio (LLR) of b_n for a given received sequence y can be expressed as:

$$\begin{aligned} L(b_n|\mathrm{y}) &= \ln\left(\frac{\sum\limits_{(s',s)\Rightarrow b_n=0} \gamma_n(0,s',s)\alpha_{n-1}(s')\beta_n(s)}{\sum\limits_{(s',s)\Rightarrow b_n=1} \gamma_n(1,s',s)\alpha_{n-1}(s')\beta_n(s)}\right) \\ &= L_c y_k + \ln\left(\frac{\sum\limits_{(s',s)\Rightarrow b_n=0} \alpha_{n-1}(s')\beta_n(s)P(b_n=0,s|s')}{\sum\limits_{(s',s)\Rightarrow b_n=1} \alpha_{n-1}(s')\beta_n(s)P(b_n=1,s|s')}\right). \end{aligned} \tag{2.78}$$

It is worth noting that the source *a priori* information and the extrinsic information in (2.78) cannot be separated. A further simplification of (2.78) can be achieved by using the logarithms of the above quantities, as done in Section 2.4.2.2.1.

2.4.2.3 Concatenation of MAP Decoding and Sequence Estimation

There are several reasons for concatenating MAP decoding with sequence estimation:

- For bit-by-bit MAP decoding, the decoder maximises the *a posteriori* probability of each bit, which results in achieving the minimum BER. However, the bit-by-bit MAP decoding is unable to guarantee that the output bit sequence will form a valid symbol-based path through the trellis, which implies that there may be invalid codewords in the decoded sequence. Hence MLSE may be performed after the bit-by-bit MAP decoding so as to find a valid VLC codeword sequence. Specifically, given the APPs $P(b_n|\mathrm{y})$ provided by the MAP decoder, the sequence estimator selects the specific sequence $\hat{\mathrm{b}}$ that maximises the probability $P(\mathrm{b}|\mathrm{y})$, which is formulated as

$$\hat{\mathrm{b}} = \arg\max_{\mathrm{b}}\{P(\mathrm{b}|\mathrm{y}) \approx \prod_n P(b_n|\mathrm{y})\}. \tag{2.79}$$

- For symbol-by-symbol based MAP decoding, the same problem as outlined above exists. The MAP decoder is capable of guaranteeing that the number of decoded symbols is equal to the number of transmitted symbols and that the corresponding SER is minimised. However, the number of decoded bits may still deviate from that of the transmitted bits. This might be a problem, when converting the symbol-by-symbol reliability information to bit-by-bit reliability information. In this case, the MLSE may be performed in order to find a valid path in the trellis. Specifically, given the APPs $P(u_k|\mathrm{y})$ generated by the MAP decoder, the sequence estimator selects the specific sequence $\hat{\mathbf{u}}$ that maximises the probability $P(\mathbf{u}|\mathbf{y})$, which is formulated as

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}}\{P(\mathbf{u}|\mathbf{y}) = \prod_k P(u_k|\mathbf{y})\}. \tag{2.80}$$

2.4.3 Simulation Results

In this section we characterise the achievable performance of the decoding schemes described in the previous sections.

For all simulations BPSK is used as the baseband modulation scheme and an AWGN channel is assumed. Furthermore, assume that the source is memoryless and it is encoded by the various VLCs [17] as shown in Table 2.12. To be more specific, HUFF represents a Huffman code, RVLC-1 is a asymmetric RVLC having a free distance of 1 and RVLC-2 is a symmetric RVLC having a free distance of 2.

Table 2.12: VLC codes used in the simulations

Symbol	Probability	HUFF	RVLC-1	RVLC-2
0	0.33	00	00	00
1	0.30	01	01	11
2	0.18	11	10	010
3	0.10	100	111	101
4	0.09	101	11011	0110
Average Length		2.19	2.37	2.46
Code Rate/Efficiency		0.98	0.90	0.87
Constraint Length		3	3.58	3.58
Free Distance		1	1	2

As we can see, the different source codes have different average codeword lengths, hence for encoding the same source message, the resultant bit sequences will have different lengths, which implies that they require different total transmitted signal power. For the sake of fair comparison, we introduce the source code rate R_s, which is defined as

$$R_s \triangleq \frac{H(u)}{\bar{L}}, \tag{2.81}$$

where $H(u)$ is the entropy of the symbol source U, and $\bar{L}$ is the the average codeword length. Consequently, in our simulations E_b/N_0 is calculated as:

$$E_b/N_0 = \frac{E_s}{R_s \cdot \log_2 M \cdot N_0}, \tag{2.82}$$

where E_s is the transmitted energy per channel symbol, while M is the number of modulation levels in the signal space of M-ary modulation. For BPSK modulation we have $M = 2$.

As for our performance measure, we use the so-called Levenshtein SER [44], since owing to the self-synchronization properties of VLCs, a simple symbol-by-symbol comparison of the decoded sequence with the original sequence is not fair in the context of numerous applications. If a decoding error occurs, there might be VLC symbol insertions or deletions in the decoded sequence. After resynchronization the remaining sequence therefore might be a shifted version of the original one, although a simple symbol comparison would identify the entire remaining part of the sequence as corrupted. A detection metric that is more appropriate in this case is the Levenshtein distance [157], which is defined as the minimal number of symbol insertions, deletions or substitutions that transform one symbol sequence into another. In our simulations we evaluate the SER in terms of the Levenshtein distance.

2.4.3.1 Performance of Symbol-level Trellis Based Decoding

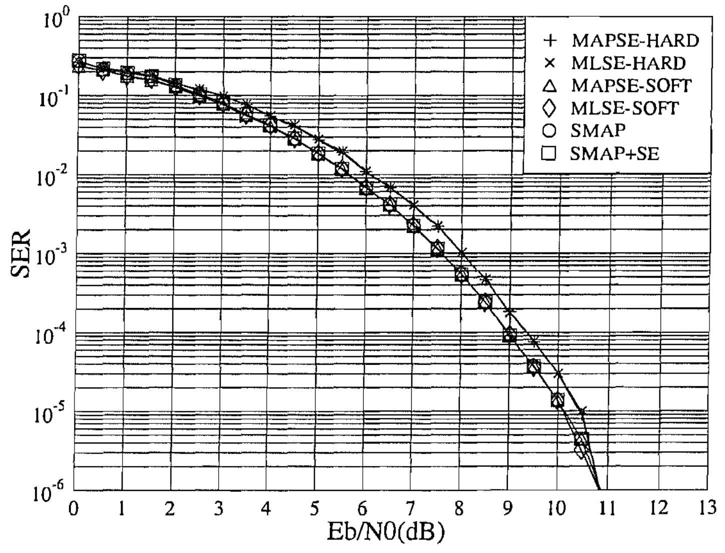


Figure 2.11: SER versus E_b/N_0 results for the **Huffman code HUFF** of Table 2.12, using **symbol-level trellis** decoding and BPSK transmission over AWGN channels.

In our simulations, six different decoding methods are used, namely MAP/ML sequence estimation using hard-decisions (MAP/MLSE-HARD), MAP/ML sequence estimation using soft-decisions (MAP/MLSE-SOFT), symbol-by-symbol MAP decoding (SMAP) and symbol-by-symbol MAP decoding concatenated with sequence estimation (SMAP+SE). For every decoding scheme, we use source symbol packets of length $K = 100$ and transmit 1000 different realizations of the source packets. Furthermore, each of these 1000 packets is transmitted over 50 channel realizations. The simulations are curtailed when either all the packets have been transmitted or the number of source symbol errors reaches 5000. At SER=10^{-5}, an average of 50 errors is accumulated, which makes the measurement of SER sufficiently accurate down to a SER of 10^{-5}.

Our results for the Huffman code HUFF are shown in Fig. 2.11. As we can see, for the Huffman code the performance of soft-decoding schemes is slightly better than that of the corresponding hard-decoding schemes (gain$<$ 0.5 dB). Furthermore, the performance of the MAP decoding scheme is similar to that of the ML decoding scheme, both in the case of soft-decoding and for hard-decoding. Moreover, the performance of sequence estimation using soft-decision is similar to that of the symbol-by symbol MAP decoding, although the former has a significantly lower complexity. The results of the concatenated MAP decoding and sequence estimation are also shown in Fig. 2.11. The concatenation of these two schemes does not affect the achievable SER performance, but it is necessary for solving the previously mentioned length inconsistency problem, namely that the number of decoded bits may be different from the number of transmitted bits..

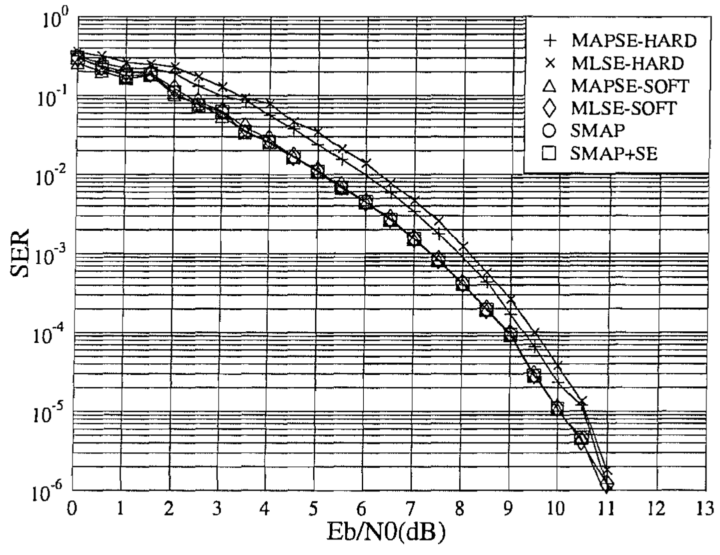


Figure 2.12: SER versus E_b/N_0 results for the **RVLC-1** of Table 2.12, using **symbol-level trellis** decoding and BPSK transmission over AWGN channels.

Our results recorded for the RVLCs RVLC-1 and RVLC-2 scheme of Table 2.12 are shown in Fig. 2.12 and Fig. 2.13, respectively. In this case, it can be seen that the soft-decoding schemes significantly outperform the hard-decoding schemes. More explicitly, RVLC-2 achieves a gain up to 3 dB as shown in Fig. 2.13. By contrast, observe in Fig. 2.13 that in case of hard-decision, the MAP decoding scheme perform slightly better than ML decoding, while they perform similarly in the case of soft-decsisions. Similar to Huffman coded scenario, the performance of soft sequence estimation is almost the same as that of symbol-by-symbol MAP decoding, as evidenced by Fig. 2.12 and Fig. 2.13.

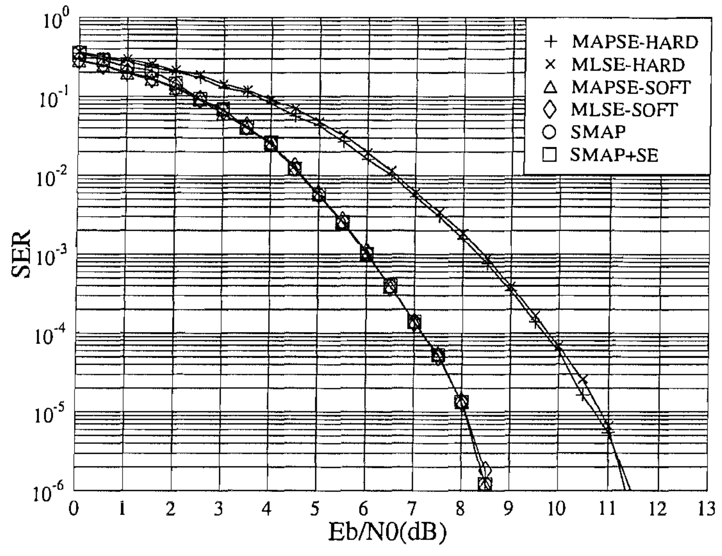


Figure 2.13: SER versus E_b/N_0 results for the **RVLC-2** of Table 2.12, using **symbol-level trellis** decoding and BPSK transmission over AWGN channels.

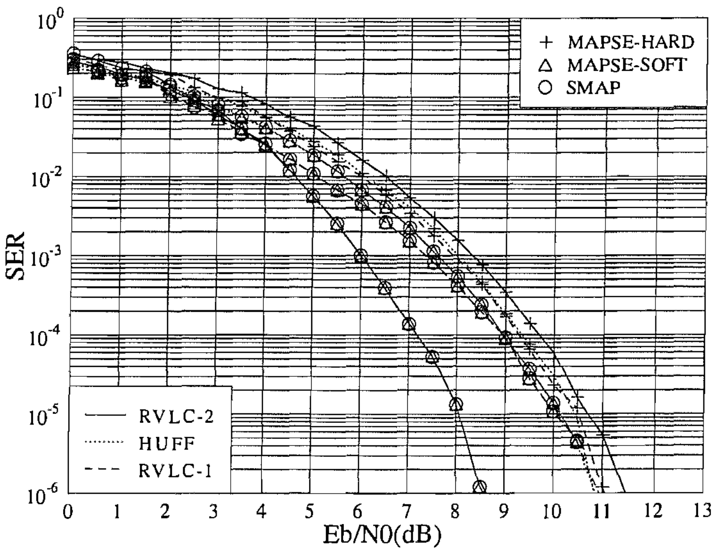


Figure 2.14: SER versus E_b/N_0 **performance comparison** for HUFF, RVLC-1 and RVLC-2 of Table 2.12, using **symbol-level trellis** decoding and BPSK transmission over AWGN channels.

The performances of the HUFF, RVLC-1 and RVLC-2 schemes are compared in Fig. 2.14. In the case of hard-decoding, the RVLC-1 arrangement performs similar to Huffman coding and slightly better than the RVLC-2 scheme. In the case of soft-decoding, the RVLC-2 significantly outperforms both the Huffman code and the RVLC-1 at medium to high SNRs in excess of $E_b/N_0 > 4dB$. The achievable E_b/N_0 gain is as high as about 2 dB at SER of 10^{-5}.

2.4.3.2 Performance of Bit-level Trellis Based Decoding

In these investigations, five different decoding methods are used, namely MAP/ML sequence estimation using hard-decisions (MAP/MLSE-HARD), MAP/ML sequence estimation using soft-decisions (MAP/MLSE-SOFT), and bit-by-bit MAP decoding concatenated with sequence estimation (BMAP+SE). For each of the decoding schemes, we use the same simulation parameters as those setup for the symbol-level trellis based decoding described in Section 2.4.3.1.

The results recorded for the Huffman code of Table 2.12 are shown in Fig. 2.15. As we can see,

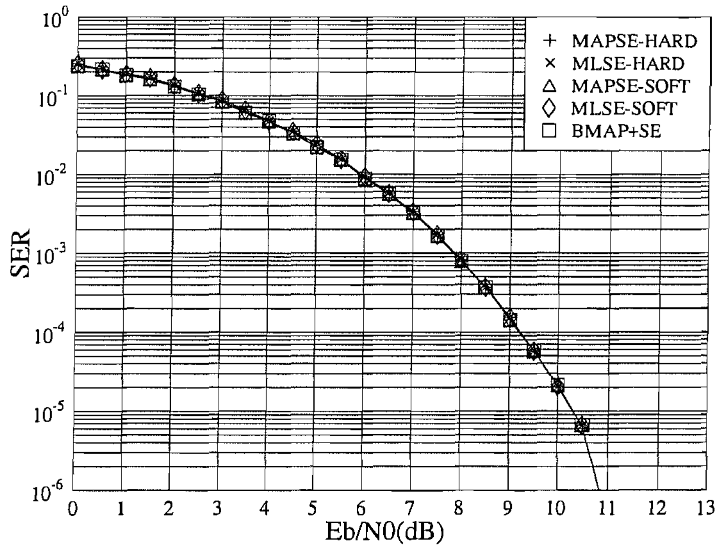


Figure 2.15: SER versus E_b/N_0 results for the **Huffman code HUFF** of Table 2.12, using **bit-level trellis** decoding and BPSK transmission over AWGN channels.

for the Huffman code, the achievable performance of all decoding schemes is similar.

The corresponding results recorded for RVLC-1 and RVLC-2 of Table 2.12 are shown in Fig. 2.16 and Fig. 2.17, respectively. This time, the soft-decoding schemes significantly outperform the hard-decoding aided schemes, especially for the code RVLC-2. The achievable E_b/N_0 gain is approximately 2.5 dB at SER= 10^{-5}. In the case of hard-decisions, the MAP decoding scheme performs slightly better than the ML decoding scheme, while they perform similarly in the case of soft-decisions.

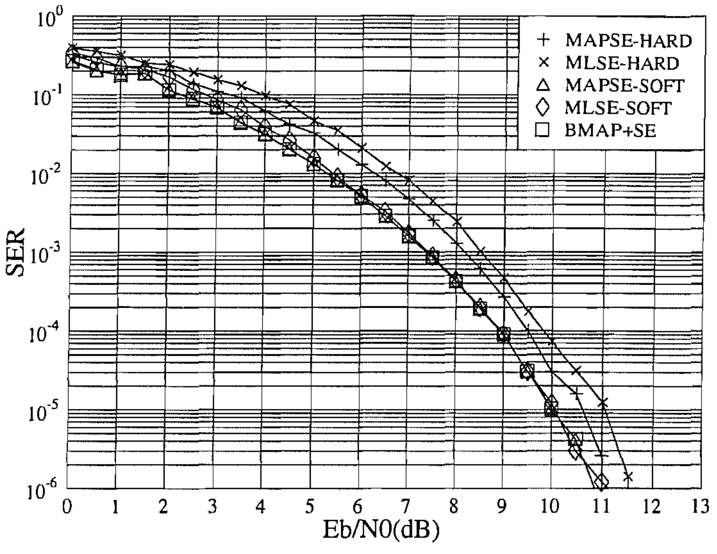


Figure 2.16: SER versus E_b/N_0 results for the **RVLC-1** of Table 2.12, using **bit-level trellis** decoding and BPSK transmission over AWGN channels.

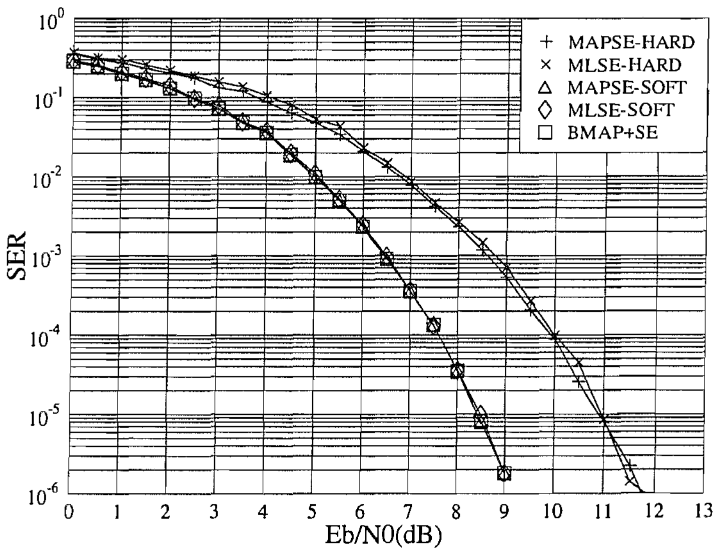


Figure 2.17: SER versus E_b/N_0 results for the **RVLC-2** of Table 2.12, using **bit-level trellis** decoding and BPSK transmission over AWGN channels.

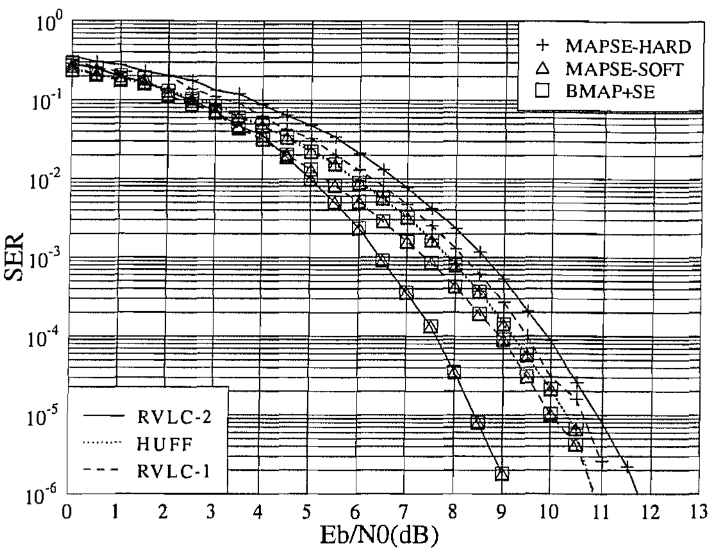


Figure 2.18: SER versus E_b/N_0 **performance comparison** for HUFF, RVLC-1 and RVLC-2 of Table 2.12, using **bit-level trellis** decoding and BPSK transmission over AWGN channels.

The performances of the Huffman code, RVLC-1 and RVLC-2 are compared in Fig. 2.18. In the case of hard-decoding, the Huffman code performs the best, while in the case of soft-decoding, the RVLC-1 and RVLC-2 schemes outperform the Huffman code at medium to high channel SNRs in excess of $E_b/N_0 > 2dB$ and RVLC-2 performs the best. Its gain is as high as about 2 dB at SER= 10^{-5}.

2.4.3.3 Comparison of Symbol-level Trellis and Bit-level Trellis Based Decoding

The achievable performance of various symbol-level trellis based decoding schemes and those of their bit-level counterparts are compared in Fig. 2.19, Fig. 2.20, and Fig. 2.21 for the Huffman code, RVLC RVLC-1 and RVLC-2 schemes, respectively.

In all the decoding schemes considered, we assume that the length of the transmitted packets expressed in terms of bits or symbols as applicable is known at the decoders. More specifically, the length information includes the number of transmitted bits for the bit-level trellis based decoding schemes, while both the number of transmitted bits and source symbols has to be known for the symbol-level trellis based decoding schemes. As shown in Fig. 2.19, for the Huffman code, the symbol-level trellis based decoding schemes perform slightly better than the bit-level trellis based decoding schemes, when using soft-decoding. However the above observations are reversed in Fig. 2.19 in the case of hard-decoding. For the RVLC-1, the symbol-levle trellis based decoding schemes performs slightly better than the corresponding bit-level trellis based decoding schemes when using hard-decoding, while they perform similarly in the case of soft-decoding, as seen in Fig. 2.20. Finally,

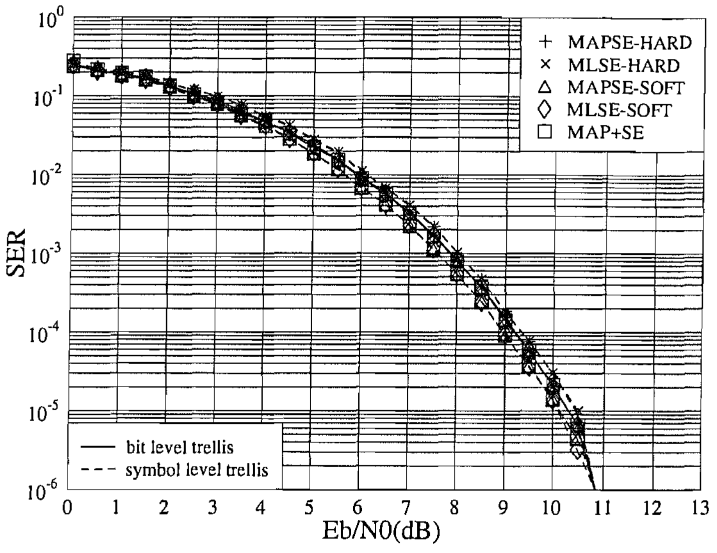


Figure 2.19: SER versus E_b/N_0 **performance comparison** of symbol-level trellis based decoding and bit-level trellis based decoding for the **Huffman code HUFF** of Table 2.12, using BPSK transmission over AWGN channels.

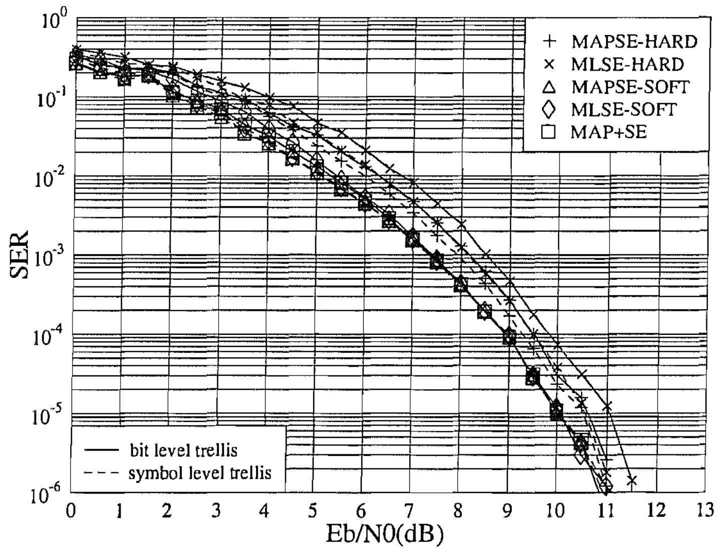


Figure 2.20: SER versus E_b/N_0 **performance comparison** of symbol-level trellis based decoding and bit-level trellis based decoding for **RVLC-1** of Table 2.12, using BPSK transmission over AWGN channels.

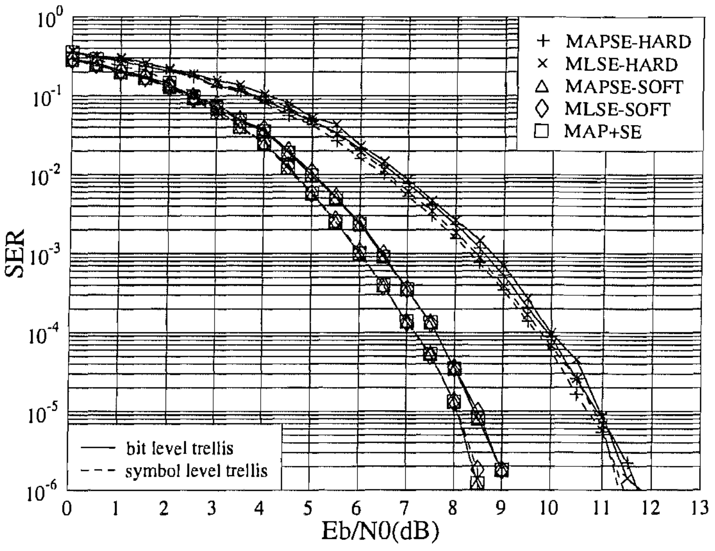


Figure 2.21: SER versus E_b/N_0 **performance comparison** of symbol-level trellis based decoding and bit-level trellis based decoding for **RVLC-2** of Table 2.12, using BPSK transmission over AWGN channels.

for RVLC-2, the symbol-level trellis based decoding schemes perform consistently better (gain$<$ 0.5 dB) than the corresponding bit-level trellis based decoding schemes, for both hard-decoding and soft-decoding.

2.5 Summary and Conclusions

In this chapter, we investigated the design of efficient VLC source codes and their soft-decoding methods. After reviewing their construction methods in Section 2.3, we proposed a novel algorithm for the design of efficient RVLCs and VLEC codes in Section 2.3.4 and Section 2.3.5, respectively, which optimises the VLC codeword length distribution on a length-by-length basis. For RVLCs it results in codes of higher efficiency and/or shorter maximum codeword length than the algorithms previously disseminated in the literature, as shown in Table 2.3, 2.4, 2.5 and 2.9. For VLEC codes, the techniques proposed in Section 2.3.5 significantly reduce the associated construction complexity, as shown in Table 2.10. The optimisation procedure is independent of the codeword selection mechanism, therefore it can be combined with all existing algorithms aiming for efficient codeword design. Our main results are summarised in Table 2.13.

In Section 2.4, we considered the soft decoding of VLCs. Both symbol-level trellises and bit-level trellises were described. Based on these trellis representations, two different types of decoding methods - namely sequence estimation using the VA and MAP decoding employing the BCJR algorithm -

Table 2.13: Comparison of various VLC construction algorithms when using the English alphabet as the source

$d_f = 1$ RVLC					
	Takishima's Alg. [17]	Tsai's Alg. [26]	Lakovic' Alg. [28]	Alg. C of Table 2.3	Alg. E of Table 2.9
Avg. Length	4.36068	4.30678	4.25145	4.18732	4.19959
Max. Length	12	10	10	11	10
Efficiency	0.95	0.96	0.97	0.98	0.98

$d_f = 2$ RVLC			
	Lakovic' Alg. [27]	Alg. C of Table 2.5	Alg. E of Table 2.9
Avg. Length	4.34534	4.33957	4.23656
Max. Length	10	8	9
Efficiency	0.95	0.95	0.97

were discussed with the aid of a host of simulation results, as shown in Section 2.4.3. Table 2.14 lists some of the ML VLC decoding results.

Table 2.14: Summary of ML VLC decoding performance. The numerical values in the table are the minimum E_b/N_0 values required for achiving a SER of 10^{-5} for the different decoding algorithms, except that the numerical values in the "Gain" columns are the E_b/N_0 gains of soft decision decoding over the corresponding hard decision decoding.

	Symbol-level Trellis			Bit-level Trellis		
	Soft Decisions	Hard Decisions	Gain	Soft Decisions	Hard Decisions	Gain
HUFF	10.1 dB	10.5 dB	0.4 dB	10.3 dB	10.3 dB	0.0 dB
RVLC-1	10.0 dB	10.5 dB	0.5 dB	10.0 dB	11.0 dB	1.0 dB
RVLC-2	8.0 dB	10.9 dB	2.9 dB	8.5 dB	11.0 dB	2.5 dB

Based on the simulation results in Table 2.14 and those in Section 2.4.3, we can reach the following conclusions:

1) Soft Decoding versus Hard Decoding

At the beginning of Section 2.4, we conjectured that soft decoding may outperform hard decoding according to one of Viterbi's three lessons [153]. Whether this conjecture is true, depends on both the source code and the bit versus symbol-based trellis structure used. For the Huffman code, soft decoding performs similar to hard decoding in the case of using a bit-level trellis, while soft decoding performs better in the case of symbol-level trellis as evidenced in Fig. 2.19. Furthermore, observe in Fig. 2.20 and Fig. 2.21 for RVLC-1 and RVLC-2 schemes, soft decoding outperforms hard decoding in the case of both trellises. It can be shown that the larger the free distance and the higher the constraint length of the source code, the larger the gain that soft decoding achieves over hard decoding.

2) MAPSE versus MLSE

Our simulation results in Section 2.4.3 show that we cannot attain any performance gain for MAP sequence estimation over ML sequence estimation in the case of soft decoding, although the former exploits the source symbol-probability information. However, in the case of hard decoding, MAP sequence estimation achieves some gain over the ML sequence estimation for both the RVLC-1 and RVLC-2 schemes, as shown in Fig. 2.12, 2.13, 2.16 and 2.17.

In addition to the effects of different source codes, Buttigieg [35] investigated the effects of different source statistics on the achievable performance and proposed the so-called MAP factor performance metric. It has been shown that the larger the MAP factor, the larger the attainable E_b/N_0 gain [35].

3) Bit-level Trellis versus Symbol-level Trellis

Since the symbol-level trellis is capable of exploiting more source information than the bit-level trellis, the decoding schemes based on a symbol-level trellis are typically capable of performing slightly better than their bit-level based counterparts, which can be seen from Fig. 2.19, 2.20, 2.21 as well as Table 2.14. However, according to the construction methods described in Section 2.4.1 the bit-level trellis structure only depends on the specific VLC source code, which is time-invariant. By contrast, the symbol-level trellis construction depends on both the specific VLC source code as well as the specific length of the frame to be decoded, hence it is data-dependent. Therefore, the decoding scheme based on the bit-level trellis is typically of lower complexity than that based on the symbol-level trellis.

4) SE versus Symbol-by-Symbo MAP Decoding

We observed in our investigations that the symbol-by-symbol or bit-by-bit MAP decoding schemes perform similarly to the corresponding SE schemes using soft-decisions, as shown in Fig. 2.11-2.18. However, the MAP decoding schemes are capable of exploiting both the soft inputs generated by the previous decoding stage as well as of providing soft outputs, which can be used in a successive decoding stage. These issues will be explored in more detail in the next chapter.

5) Effect of Source Codes

In the investigations of this chapter, various source codes have been used, including a $d_f = 1$ Huffman code, a $d_f = 1$ RVLC and a $d_f = 2$ RVLC scheme. It is clear that the free distance of the VLC plays an important role in determining the MAP/ML decoding performance, when using soft decision based decoding. Our results seen in Fig. 2.14 and 2.18 demonstrated that as expected, the $d_f = 2$ RVLC significantly outperforms the $d_f = 1$ RVLC and Huffman code.

Chapter 3

Iterative Source/Channel Decoding

Code concatenation constitutes a convenient technique of constructing powerful codes capable of achieving significant coding gains, while keeping the decoding complexity manageable. The concept of serially concatenated codes (SCC)s having an "outer" and an "inner" code used in cascade was proposed in [129]. The discovery of parallel concatenated "turbo" codes (PCC)s [73] considerably improved the achievable performance gains by separating component codes through interleavers and using iterative decoding in order to further reduce the BER. The intensive research efforts of the ensuing era demonstrated [74] that the employment of iterative processing techniques is not limited to traditional concatenated coding schemes. In other words, the "turbo principle" [57] is applicable to numerous other algorithms that can be found in digital communications, for example in turbo equalisation [86], spectrally efficient modulation [87, 88], turbo multiuser detection [91, 92] and channel coded code-division multiple-access [94].

Previously, substantial research efforts have been focused on optimising concatenated coding schemes in order to improve the asymptotic slopes of their error probability curves especially at moderate to high SNRs. Recently, researchers have focussed their efforts on investigating the convergence behaviour of iterative decoding. In [81] the authors proposed a so-called density evolution algorithm for calculating the convergence thresholds of randomly constructed irregular LDPC codes designed for the AWGN channel. This method was used to study the convergence of iterative decoding for turbo-like codes in [108], where an SNR measure was proposed for characterising the inputs and outputs of the component decoders. The mutual information between the data bits at the transmitter and the soft values at the receiver was used to describe the flow of extrinsic information between two soft-in/soft-out decoders in [110]; the exchange of extrinsic information between constituent codes may be visualised as a staircase-shaped gradually increasing decoding trajectory within the EXIT chart of both parallel [102, 115] and serially concatenated codes [116]. A comprehensive study of the convergence behaviour of iterative decoding schemes for bit-interleaved coded modulation, equalisation and serially concatenated binary codes was presented in [119].

In this chapter we will investigate an intriguing application of the turbo principle [57], namely, ISCD [46, 49, 158], where the channel decoder and the source decoder can be viewed as a pair of serial concatenated codes. By applying the turbo principle, the residual redundancy in the source after source encoding may be efficiently utilised for improving the system's performance in comparison to separate source/channel decoding. Furthermore, the convergence behaviour of various ISCD schemes will be analysed with the aid of EXIT charts.

This chapter is organised as follows. Section 3.1 provides a brief introduction of various concatenated schemes and that of the iterative decoding principle - the turbo principle. The kernel of the turbo principle - the SISO APP decoding algorithm as well as the semi-analytical tool of EXIT charts are described in Section 3.2. The achievable performance of the ISCD schemes is investigated in Section 3.3 and Section 3.4 when communicating over both AWGN channels and dispersive channels, respectively. Section 3.5 concludes the chapter.

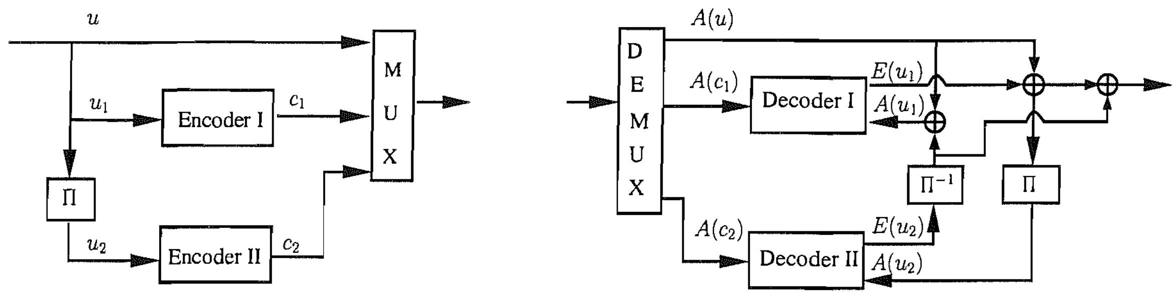
3.1 Concatenated Coding and the Turbo Principle

In this section, we will briefly describe several commonly used concatenated schemes, namely parallel concatenation [73, 130] (PC), serial concatenation [75, 134] (SC) and hybrid concatenation [159–161] (HC) of various consituent codes. Furthermore, we demonstrate that the turbo principle may be applied for detecting any of the above schemes.

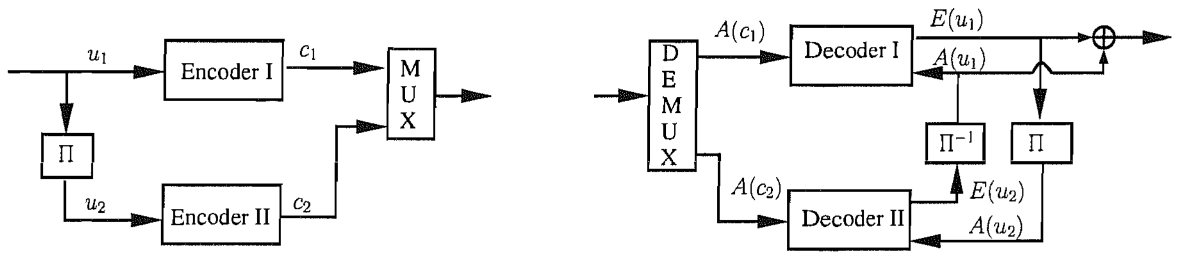
3.1.1 Classification of Concatenated Schemes

3.1.1.1 Parallel Concatenated Schemes

The classic turbo codes [73] employ a parallel concatenated structure. Fig. 3.1 shows two examples of the family of turbo codes. A so-called systematic code is shown in Fig. 3.1(a), where the input information bits are simply copied to the output interspersed with the parity bits. By contrast a non-systematic code is shown in Fig. 3.1(b), where the original information bits do not explicitly appear at the output. The component encoders are often convolutional encoders but binary Bose-Chaudhuri-Hocquenghem (BCH) codes [74] have also been used. At the encoder, the input information bits are encoded by the first constituent encoder (Encoder I) and an interleaved version of the input information bits is encoded by the second constituent encoder (Encoder II). The encoded output bits may be punctured, hence arbitrary coding rates may be obtained. Furthermore, this structure can be extended to a parallel concatenation of more than two component codes, leading to multiple-stage turbo codes [130, 162].



(a) A systematic parallel concatenated code: the encoder and the decoder



(b) A non-systematic parallel concatenated code: the encoder and the decoder

Figure 3.1: Two examples of parallel concatenated codes. The constituent codes are non-systematic codes having arbitrary code rates.

3.1.1.2 Serially Concatenated Schemes

The basic structure of a SCC [75] is shown in Fig. 3.2. The SCC encoder consists of an outer encoder (Encoder I) and an inner encoder (Encoder II), interconnected by an interleaver. The introduction of the interleaver scrambles the bits before they are passed to the other constituent encoder, which ensures that even if a specific bit has been gravely contaminated by the channel, the chances are that the other constituent decoder is capable of providing more reliable information concerning this bit. This is a practical manifestation of time-diversity. Iterative processing is used in the SCC decoder and a performance similar to that of a classic PCC may be achieved [75]. In fact, the serial concatenation constitutes quite a general structure and many transceiver schemes can be described as serially concatenated structures [57, 85], such as those used in turbo equalisation [86], coded modulation [87, 88], turbo multiuser detection [91, 92], joint source/channel coding/decoding [46, 49, 158], LDPC [81]. Similarly, a serially concatenated scheme may contain more than two components, as in [124–126, 134]. Fig. 3.3 shows the schematic of a three-stage serially concatenated code [126, 134].

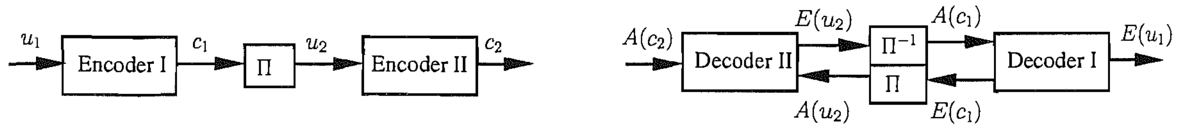


Figure 3.2: The schematic of a serially concatenated code: the encoder and the decoder.

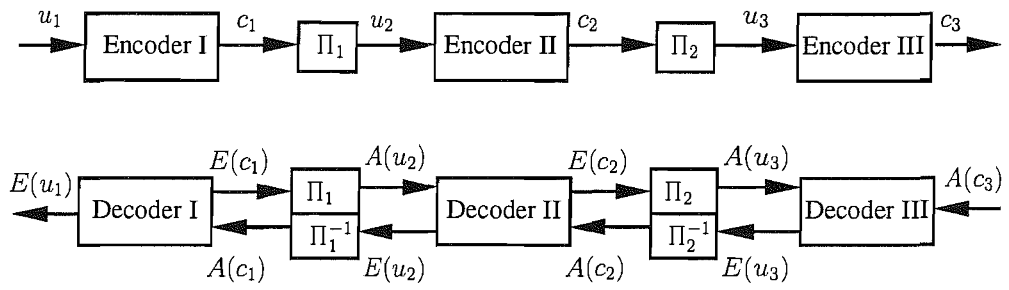


Figure 3.3: The schematic of a three-stage serially concatenated code: the encoder and the decoder.

3.1.1.3 Hybridly Concatenated Schemes

A natural extension of PCCs and SCCs is constituted by the family of hybridly concatenated codes [159–161], which combines the two principles. More specifically, some of the components are concatenated in parallel and some are serially concatenated. Again, the employment of this scheme is not restricted to channel coding. For example, combined turbo decoding and turbo equalisation [163,164] constitutes a popular application.

3.1.2 Iterative Decoding and Scheduling

Since the invention of turbo codes [73], iterative (turbo) decoding has been successfully applied to diverse detection/decoding problems which may inherit any of the three concatenated structures described above. Hence this technique was referred to as the "Turbo Principle" [57]. The crucial point at the receiver is that the component detectors/decoders have to be soft-in/soft-out decoders that are capable of accepting and delivering probabilities or soft values and the *extrinsic* part of the soft-output of one of the decoders is passed on to the other decoder to be used as *a priori* input. If there are only two component decoders, the decoding alternates between them. When there are more component decoders, the activation order of the decoders is an important issue, which is sometimes also referred to as scheduling by borrowing the terminology from the field of resource allocation. The specific activation order may substantially affect the decoding complexity, delay and performance of the system [125, 126].

3.2 SISO APP Decoders and Their EXIT Characteristics

The key point of iterative decoding of various concatenated schemes is that each component decoder employs a SISO algorithm. Benedetto *et al.* [160] proposed a SISO APP module, which implements the APP algorithm [52] in its basic form for any trellis-based decoder/detector. This SISO APP module is employed in our ISCD schemes in the rest of this treatise unless stated otherwise. Moreover, the capability of processing *a priori* information and generating *extrinsic* information is an inherent characteristic of a SISO APP module, which may be visualised by EXIT charts. Let us first introduce the SISO APP module.

3.2.1 A Soft-Input Soft-Output APP Module

3.2.1.1 The Encoder

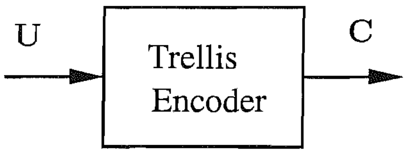


Figure 3.4: A trellis encoder which processes the input information symbol sequence U and outputs encoded symbol sequence C. The encoder's state transitions may be described by certain trellis diagrams.

Fig. 3.4 shows a trellis encoder, which processes the input information symbol sequence U and outputs encoded symbol sequence C. The code trellis can be time-invariant or time-varying. For simplicity of exposition, we will refer to the case of binary time-invariant convolutional codes having code rate of $R = k_0/n_0$.

The input symbol sequence $\mathbf{U} = (U_k)_{k\in\mathcal{K}}$ is defined over a symbol index set $\mathcal{K}$ and drawn from the alphabet $\mathcal{U} = \{u_1, \cdots, u_{N_I}\}$. Each input symbol U_k consists of k_0 bits $U_k^j, j = 1, 2, \cdots, k_0$ with realization $u^j \in \{0, 1\}$. The encoded output sequence $\mathbf{C} = (C_k)_{k\in\mathcal{K}}$ of Fig. 3.4 is defined over the same symbol index set $\mathcal{K}$, and drawn from the alphabet $\mathcal{C} = \{c_1, \cdots, c_{N_o}\}$. Each output symbol C_k consists of n_0 bits $C_k^j, j = 1, 2, \cdots, n_0$ with realization $c^j \in \{0, 1\}$.

The legitimate state transitions of a time-invariant convolutional code are completely specified by a single trellis section, which describes the transitions ("edges") between the states of the trellis at time instants k and $k + 1$. A trellis section is characterised by:

- a set of N states $\mathcal{S} = \{s_1, \cdots, s_N\}$. The state of the trellis at time k is $S_k = s$, with $s \in \mathcal{S}$;
- a set of $N \cdot N_I$ edges obtained by the Cartesian product $\mathcal{E} = \mathcal{S} \times \mathcal{U} = \{e_1, \cdots, e_{N \cdot N_I}\}$, which represent all legitimate transitions between the trellis states.

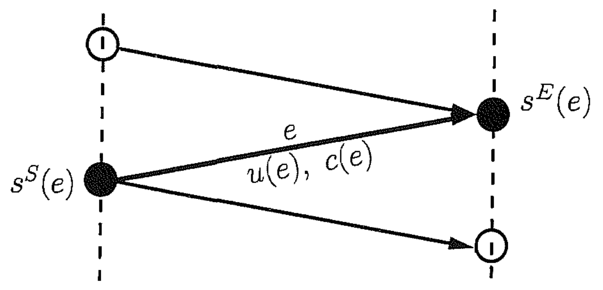


Figure 3.5: An edge of the trellis section.

As seen in Fig. 3.5, the following functions are associated with each edge $e \in \mathcal{E}$:

- the starting state $s^S(e)$;
- the ending state $s^E(e)$;
- the input symbol $u(e)$;
- and the output symbol $c(e)$.

The relationship between these functions depends on the particular encoder, while the assumption that the pair $[s^S(e), u(e)]$ uniquely identifies the ending state $s^E(e)$ should always be fulfilled, since it is equivalent to stating that given the initial trellis state, there is a one-to-one correspondence between the input sequences and state transitions.

3.2.1.2 The Decoder

The APP algorithm designed for the SISO decoder operates on the basis of the code's trellis representation. It accepts as inputs the probability of each of the original information symbols and of the encoded dsymbols, labelling the edges of the code trellis and generates as its outputs an updated version of these probabilities based upon the code constraints. Hence the SISO module may be viewed as a four-terminal processing block, as shown in Figure 3.6.

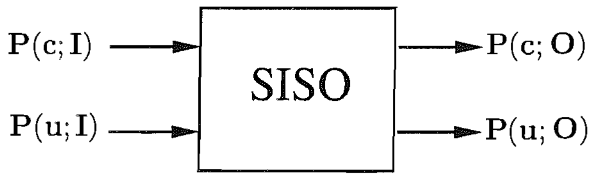


Figure 3.6: A SISO module which processes soft information input of P(u; I) as well as P(c; I) and generates soft information output of P(u; O) as well as P(c; O).

More specifically, we associate the sequence of information symbols $\mathbf{u}$ with the sequence of *a priori* probabilities $\mathbf{P}(\mathbf{u}; I) = (P_k(u; I))_{k \in \mathcal{K}}$. Furthermore, we associate the sequence of encoded symbols $\mathbf{c}$ with the sequence of *a priori* probabilities $\mathbf{P}(\mathbf{c}; I) = (P_k(c; I))_{k \in K}$, where we have $P_k(c; I) = \prod_{j=1}^{n_0} P_k(c^j; I)$. The assumption that the *a priori* probabilities of specific symbols can be represented as the product of the probabilities of its constituent bits may be deemed valid, when sufficiently long independent bit interleavers rather than symbols interleavers are used in the iterative decoding scheme of a concatenated code. The two outputs of the SISO module are the updated probabilities $\mathbf{P}(\mathbf{u}; O)$ and $\mathbf{P}(\mathbf{c}; O)$. Here, the letters I and O refer to the input and output of the SISO module of Fig. 3.6, respectively.

Let the symbol index set $\mathcal{K} = \{1, \cdots, n\}$ be finite. The SISO module operates by evaluating the output probabilities is as follows.

1) The *a posteriori* probabilities $\tilde{P}_k(u^j : O)$ and $\tilde{P}_k(c^j : O)$ associated with the jth bit within each symbol at time k are computed as [160]

$$\tilde{P}_k(u : O) = \tilde{H}_{u^j} \sum_{e:U_k^j(e)=u^j} \alpha_{k-1}(s^S(e))\beta_k(s^E(e))\gamma_k(e), \tag{3.1}$$

and

$$\tilde{P}_k(c : O) = \tilde{H}_{c^j} \sum_{e:C_k^j(e)=c^j} \alpha_{k-1}(s^S(e))\beta_k(s^E(e))\gamma_k(e), \tag{3.2}$$

respectively, where $\tilde{H}_{u^j}$ and $\tilde{H}_{c^j}$ are normalisation factors ensuring that $\sum_{u^j} \tilde{P}_k(u : O) = 1$ and $\sum_{c^j} \tilde{P}_k(c : O) = 1$, respectively.

2) $\gamma_k(e) \triangleq P(s^E(e), e | s^S(e))$ is the state transition probability associated with each edge at the trellis section k of Fig. 3.5, which is computed as

$$\gamma_k(e) = P_k[u(e); I] \cdot P_k[c(e); I]. \tag{3.3}$$

3) The quantity $\alpha_k(s)$ is associated with each state s, which can be obtained with the aid of the *forward* recursions originally proposed in [52] and detailed in [74] as

$$\alpha_k(s) = \sum_{e:s^E(e)=s} \alpha_{k-1}[s^S(e)]\gamma_k(e), \quad k = 1, \cdots, n, \tag{3.4}$$

with the initial values of $\alpha_0(s) = 1$, if $s = S_0$ and $\alpha_0(s) = 0$, otherwise.

4) The quantity $\beta_k(s)$ is also associated with each state s, which can be obtained through the

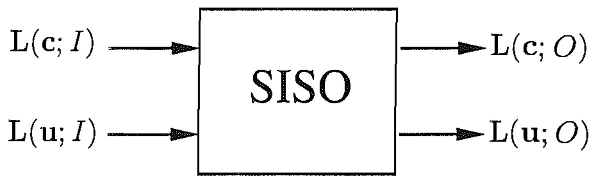


Figure 3.7: A SISO module operating in the log-domain.

backward recursions as [160]

$$\beta_k(s) = \sum_{e:s^S(e)=s} \beta_{k+1}[s^S(e)]\gamma_{k+1}(e), \quad k = n-1, \cdots, 0, \tag{3.5}$$

with the initial values of $\beta_n(s) = 1$, if $s = S_n$ and $\beta_n(s) = 0$, otherwise.

The *extrinsic* probabilities $P_k(u^j; O)$ and $P_k(c^j; O)$ are obtained by excluding the *a priori* probabilities $P_k(u^j; I)$ and $P_k(c^j; I)$ of the individual bits concerned from the *a posteriori* probabilities, yielding [160],

$$P_k(u^j; O) = H_{u^j} \frac{\tilde{P}_k(u : O)}{P_k(u^j; I)}, \tag{3.6}$$

and [160]

$$P_k(c^j; O) = H_{c^j} \frac{\tilde{P}_k(c : O)}{P_k(c^j; I)}, \tag{3.7}$$

where, again, H_{u^j} and H_{c^j} are normalisation factors ensuring that $\sum_{u^j} P_k(u : O) = 1$ and $\sum_{c^j} P_k(c : O) = 1$, respectively.

For the sake of avoiding any numerical instability and reducing the computational complexity, the above algorithm relying on multiplications can be converted to an additive form in the log-domain. More details on the released operations can be found in Appendix A. Furthermore, we may use log-likelihood ratios (LLRs) instead of probabilities, where we define

$$L(U_k^j; I) \triangleq \ln[P_k(U_k^j = 0; I)] - \ln[P_k(U_k^j = 1; I)], \tag{3.8}$$

$$L(C_k^j; I) \triangleq \ln[P_k(C_k^j = 0; I)] - \ln[P_k(C_k^j = 1; I)], \tag{3.9}$$

$$L(U_k^j; O) \triangleq \ln[P_k(U_k^j = 0; O)] - \ln[P_k(U_k^j = 1; O)], \tag{3.10}$$

$$L(C_k^j; O) \triangleq \ln[P_k(C_k^j = 0; O)] - \ln[P_k(C_k^j = 1; O)]. \tag{3.11}$$

Consequently, in the log-domain, the inputs of the four-terminal SISO module are $L(\mathrm{u}; \mathrm{I})$ and $L(\mathrm{c}; \mathrm{I})$, while its outputs are $L(\mathrm{u}; \mathrm{O})$ and $L(\mathrm{c}; \mathrm{O})$, as shown in Figure 3.7.

3.2.2 EXIT Chart

3.2.2.1 Mutual Information

For convenience, we redraw the previously introduced trellis encoder of Fig. 3.4 and the corresponding SISO APP module of Fig. 3.6 as seen in Fig. 3.8. We will use the following notations:

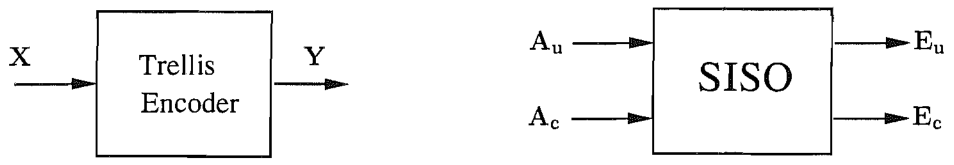


Figure 3.8: A trellis encoder and the corresponding SISO LogAPP decoder.

- ► $\mathrm{X} = (X_n)_{n\in\{0,\cdots,N_u-1\}}$ represents the sequence of information bits of length N_u, which is input to the trellis encoder, while X_n is a binary random variable, denoting the information bits having realizations $x_n \in \{\pm 1\}$;
- ► $\mathrm{Y} = (Y_n)_{n\in\{0,\cdots,N_c-1\}}$ is the sequence of encoded bits of length N_c output by the trellis encoder, while Y_n is a binary random variable, denoting the encoded bits with assuming values of $y_n \in \{\pm 1\}$;
- ► $\mathrm{A_u} = (A_{u,n})_{n\in\{0,\cdots,N_u-1\}}$ is the sequence of the *a priori* LLR input $\mathbf{L}(\mathbf{u}; I)$ and $A_{u,n} \in \mathbb{R}$ denotes the *a priori* LLR of the information bit X_n;
- ► $\mathrm{A_c} = (A_{c,n})_{n\in\{0,\cdots,N_c-1\}}$ is the sequence of the *a priori* LLR inputs $\mathbf{L}(\mathbf{c}; I)$, where $A_{c,n} \in \mathbb{R}$ denotes the *a priori* LLR of the code bit Y_n;
- ► $\mathrm{E_u} = (E_{u,n})_{n\in\{0,\cdots,N_u-1\}}$ is the sequence of *extrinsic* LLR outputs $\mathbf{L}(\mathbf{u}; O)$ and $E_{u,n} \in \mathbb{R}$ denotes the *extrinsic* LLR of the information bit X_n;
- ► $\mathrm{E_c} = (E_{c,n})_{n\in\{0,\cdots,N_c-1\}}$ is the sequence of *extrinsic* LLR outputs $\mathbf{L}(\mathbf{c}; O)$, while $E_{c,n} \in \mathbb{R}$ denotes the *extrinsic* LLR for the code bit Y_n.

We assume that the statistical properties of $X_n, Y_n, A_{u,n}, A_{c,n}, E_{u,n}$ and $E_{c,n}$ are independent of the index n, hence the subscript n is omitted in our forthcoming derivations. Then the Shannonian mutual information between the *a priori* LLR A_u and the information bit X is defined as [102]

$$I(X; A_u) = \frac{1}{2}\sum_{x=\pm 1}\int_{-\infty}^{+\infty} f_{A_u}(\xi|X=x)\log_2\frac{2f_{A_u}(\xi|X=x)}{f_{A_u}(\xi|X=-1)+f_{A_u}(\xi|X=+1)}\mathrm{d}\xi, \qquad (3.12)$$

where $f_{A_u}(\xi|X=x)$ is the conditional probability density function associated with the *a priori* LLR A_u, and the information bits X are assumed to be equiprobable, i.e. we have $P(X=+1)=P(X=-1)=1/2$.

In order to compute the mutual information $I(X;A_u)$ of Eq. (3.12), the conditional PDF $f_{A_u}(\xi|X=x)$ of the LLR A_u has to be known. If we model the *a priori* LLR A_u as an independent Gaussian random variable n_{A_u} having a variance of $\sigma^2_{A_u}$ and a mean of zero in conjunction with the information bit X, then we have

$$A_u = \mu_{A_u}X + n_{A_u}, \tag{3.13}$$

where μ_{A_u} must fulfill

$$\mu_{A_u} = \sigma^2_{A_u}/2, \tag{3.14}$$

according to the consistency condition [81] of LLR distribution, $f_{A_u}(\xi|X=x) = f_{A_u}(-\xi|X=x)\cdot e^{x\xi}$ and the conditional pdf $f_{A_u}(\xi|X=x)$ becomes

$$f_{A_u}(\xi|X=x) = \frac{1}{\sqrt{2\pi}\sigma_{A_u}}\exp\left\{\frac{-(\xi-(\sigma^2_{A_u}/2)x)^2}{2\sigma^2_{A_u}}\right\}. \tag{3.15}$$

Consequently, the mutual information $I_{A_u} \triangleq I(X;A_u)$ defined in Eq. (3.12) can be expressed as

$$I_{A_u} = 1 - \int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}\sigma_{A_u}}\exp\left\{\frac{-(\xi-\sigma^2_{A_u}/2)^2}{2\sigma^2_{A_u}}\right\}\log_2(1+e^{-\xi})\mathrm{d}\xi. \tag{3.16}$$

For the sake of notational compactness we define

$$J(\sigma) \triangleq I_{A_u}(\sigma_{A_u}=\sigma), \qquad \sigma > 0. \tag{3.17}$$

Similarly, the same Gaussian model can be applied to the *a priori* LLR A_c, yielding

$$A_c = \mu_{A_c}Y + n_{A_c}, \tag{3.18}$$

where $\mu_{A_c} = \sigma^2_{A_c}/2$ [102], and n_{A_c} is a Gaussian random variable with a variance of $\sigma^2_{A_c}$ and a mean of zero. Then, the mutual information between the encoded bit Y and the *a priori* LLR A_c can be expressed as

$$I_{A_c} = 1 - \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma_{A_c}} \exp\left\{ \frac{-(\xi - \sigma_{A_c}^2/2)^2}{2\sigma_{A_c}^2} \right\} \log_2(1 + e^{-\xi}) \mathrm{d}\xi, \tag{3.19}$$

or in terms of the function $J(\cdot)$ as

$$I_{A_c} = J(\sigma_{A_c}). \tag{3.20}$$

The Mutual information is also used to quantify the relationship of the extrinsic output $I_{E_u} \triangleq I(X; E_u)$ and $I_{E_c} \triangleq I(Y; E_c)$ with the orignal information bits and the encoded bits as follows

$$I_{E_u} = \frac{1}{2} \sum_{x=\pm 1} \int_{-\infty}^{+\infty} f_{E_u}(\xi | Y = x) \log_2 \frac{2 f_{E_u}(\xi | X = x)}{f_{E_u}(\xi | X = -1) + f_{E_u}(\xi | X = +1)} \mathrm{d}\xi, \tag{3.21}$$

$$I_{E_c} = \frac{1}{2} \sum_{y=\pm 1} \int_{-\infty}^{+\infty} f_{E_c}(\xi | Y = y) \log_2 \frac{2 f_{E_c}(\xi | Y = y)}{f_{E_c}(\xi | Y = -1) + f_{E_c}(\xi | Y = +1)} \mathrm{d}\xi. \tag{3.22}$$

3.2.2.2 Properties of the $J(\cdot)$ Function

The function $J(\cdot)$ defined in Eq. (3.17) is useful for determining the mutual information, hence below we list some of its properties.

- ▶ It is monotonically increasing and thus it has a unique and unambiguous inverse function [102].

$$\sigma = J^{-1}(I). \tag{3.23}$$

- ▶ Its range is $[0 \ldots 1]$ and it satisfies [102]

$$\lim_{\sigma \to 0} J(\sigma) = 0, \qquad \lim_{\sigma \to \infty} J(\sigma) = 1. \tag{3.24}$$

- ▶ The capacity C_G of a binary input/continuous output AWGN channel is given by [102]

$$C_G = J(2/\sigma_n), \tag{3.25}$$

where σ_n^2 is the variance of the Gaussian noise.

- ▶ It is infeasible to express $J(\cdot)$ or its inverse in closed form. However, they can be closely approximated by [126]

$$J(\sigma) \approx (1 - 2^{-H_1\sigma^{2H_2}})^{H_3} \tag{3.26}$$

$$J^{-1}(I) \approx (-\frac{1}{H_1}\log_2(1 - I^{\frac{1}{H_3}}))^{\frac{1}{2H_2}}. \tag{3.27}$$

Numerical optimisation to minimise the total squared difference between Eq. (3.17) and Eq. (3.26) results in the parameter values of $H_1 = 0.3073, H_2 = 0.8935$ and $H_3 = 1.1064$. The solid curve in Fig. 3.9 shows the values of Eq. (3.17) and its visibly indistinguishable approximation in Eq. (3.26).

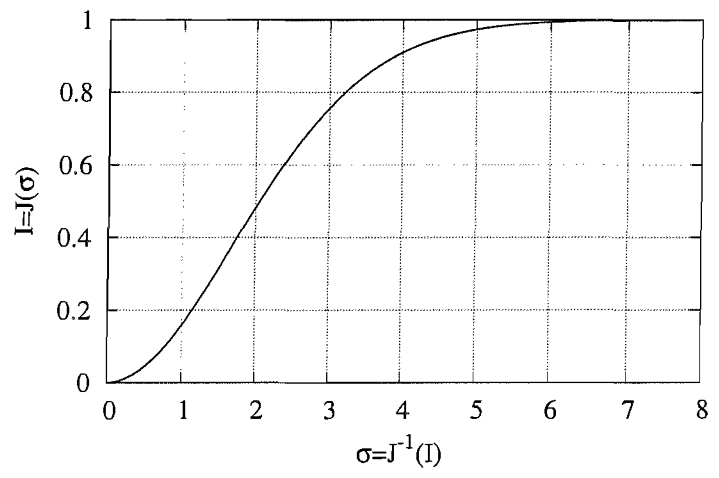


Figure 3.9: The $J(\cdot)$ function of Eq. (3.17) and its visibly indistinguishable approximation by (3.26).

3.2.2.3 Evaluation of the EXIT Characteristics of a SISO APP Module

According to the SISO APP algorithm outlined in Section 3.2.1, the *extrinsic* outputs E_u and E_c of a SISO APP module depend entirely on the *a priori* input A_u and A_c. Hence the mutual information I_{E_u} or I_{E_c} quantifying the *extrinsic* output is a function of the mutual information I_{A_u} and I_{A_c} quantifying the *a priori* input, which can be defined as

$$I_{E_u} = T_u(I_{A_u}, I_{A_c}), \tag{3.28}$$

$$I_{E_c} = T_c(I_{A_u}, I_{A_c}). \tag{3.29}$$

These two functions are often referred to as the EXIT characteristics of a SISO module, since they characterise the SISO module's capability of enhancing our confidence in its input information. Consequently, we can define an EXIT module, which is the counterpart of the corresponding SISO APP module in terms of mutual information transfer. Therefore the EXIT module has two inputs, namely, I_{A_u} and I_{A_c} as well as two outputs, namely, I_{E_u} and I_{E_c}, as depicted in Fig. 3.10. The relationships between the inputs and the outputs are determined by the EXIT functions of Eq. (3.28) and Eq. (3.29).

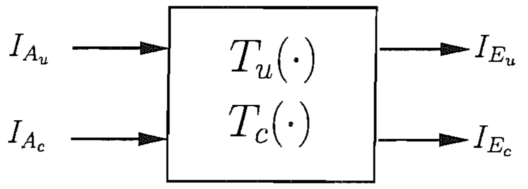


Figure 3.10: An EXIT module defined by two EXIT functions (3.28) and (3.29) of the corresponding SISO APP module.

For simple codes such as repetition codes or single parity check codes, analytical forms of the associated EXIT characteristics may be derived. However, for the SISO modules of sophisticated codecs, their EXIT characteristics have to be obtained by simulation. In other words, for given values of I_{A_u} and I_{A_c} we artificially generate the *a priori* inputs A_u and A_c, which are fed to the SISO module. Then the SISO APP algorithm of Section 3.2.1 is invoked, resulting in the *extrinsic* outputs E_u and E_c. Finally, we evaluate the mutual information I_{E_u} and I_{E_c} of Eq. (3.21) and Eq. (3.22). In this way, the EXIT characteristics can be determined once a sufficiently high number of data points are accumulated. The detailed procedure is as follows:

1) For given *a priori* input (I_{A_u}, I_{A_c}), compute the noise variance $\sigma^2_{A_u}$ and $\sigma^2_{A_c}$ of the Gaussian model of Eq. (3.13) using the inverse of the J-function, i.e. that of $\sigma_{A_u} = J^{-1}(I_{A_u})$ and $\sigma_{A_c} = J^{-1}(I_{A_c})$;

2) Generate a random binary sequence $\mathbf{X}$ with equally distributed values of $\{+1, -1\}$ and encode it, resulting in the encoded bit sequence $\mathbf{Y}$. Then generate the *a priori* inputs $\mathbf{A}_u$ and $\mathbf{A}_c$ using the Gaussian model of Eq. (3.13), i.e. by scaling the input and adding Gaussian noise, where the noise variances are obtained in step 1;

3) Invoke the SISO APP algorithm to compute the *extrinsic* output $\mathbf{E}_u$ and $\mathbf{E}_c$;

4) Use a histogram-based estimate of the conditional pdfs of $f_{E_u}(\xi|Y = y)$ and $f_{E_c}(\xi|Y = y)$ in Eq. (3.21) and Eq. (3.22), respectively. Then compute the mutual information I_{E_u} and I_{E_c} according to Eq. (3.21) and Eq. (3.22), respectively.

5) Set $I_{A_u} = I_{A_u} + \triangle I_{A_u}$, $0 \leq I_{A_u} \leq 1$ and $I_{A_c} = I_{A_c} + \triangle I_{A_c}$, $0 \leq I_{A_c} \leq 1$. Repeat steps 1-4 until a sufficiently high number of data points are accumulated.

This is the original method proposed in [102] for the evaluation of the EXIT characteristics. Actually, Step 4 in the above procedure may be significantly simplified [136, 165], as long as the outputs $\mathrm{E_u}$ and $\mathrm{E_c}$ of the SISO module are true *a posteriori* LLRs. In fact, the mutual information can be expressed as the expectation of a function of solely the absolute values of the *a posteriori* LLRs [136]. This result provides a simple method for computing the mutual information by simulation. As opposed to the original method [102], explicit measurements of histograms of the soft-outputs are not necessary. More details are given in the section below.

3.2.2.4 Simplified Computation of Mutual Information

Let us first consider the mutual information $I(X_n; E_{u,n})$ between the *a posteriori* LLR $E_{u,n}$ and the transmitted bit X_n, which can be expressed as

$$I(X_n; E_{u,n}) = H(X_n) - H(X_n|E_{u,n}), \tag{3.30}$$

$$= H(X_n) - \mathrm{E}\{H(X_n|E_{u,n} = \lambda_n)\}, \tag{3.31}$$

where $E\{x\}$ denotes the expected value x. In tangible physical terms Eq. (3.30) quantifies the extra information gleaned with advent of the knowledge of Eu, n. Note that according to the definition of LLRs, the *a posteriori* probability $P(X_n = \pm 1|\lambda_n)$ can be computed from the *a posteriori* LLR $E_{u,n} = \lambda_n$ as $P(X_n = \pm 1|\lambda_n) = \frac{1}{1+e^{\mp\lambda_n}}$ [74, 101]. Hence the conditional entropy $H(X_n|E_{u,n} = \lambda_n)$ can be obtained as

$$H(X_n|E_{u,n} = \lambda_n) = -\frac{1}{1+e^{\lambda_n}}\log_2\frac{1}{1+e^{\lambda_n}} - \frac{1}{1+e^{-\lambda_n}}\log_2\frac{1}{1+e^{-\lambda_n}}, \tag{3.32}$$

$$\triangleq h(\lambda_n). \tag{3.33}$$

Therefore, the symbol-wise mutual information $I(X_n; E_{u,n})$ of Eq. (3.30) may be expressed as

$$I(X_n; E_{u,n}) = H(X_n) - \mathrm{E}\{h(\lambda_n)\}. \tag{3.34}$$

Similarly, we obtain the symbol-wise mutual information $I(Y_n; E_{c,n})$ as

$$I(Y_n; E_{c,n}) = H(Y_n) - \mathrm{E}\{h(\lambda_n)\}. \tag{3.35}$$

Consequently, the average mutual information I_{E_u} and I_{E_c} can be expressed as

$$I_{E_u} = \frac{1}{N_u}\sum_{n=0}^{N_u-1}(H(X_n) - \mathrm{E}\{h(\lambda_n)), \tag{3.36}$$

$$I_{E_c} = \frac{1}{N_c}\sum_{n=0}^{N_c-1}(H(Y_n) - \mathrm{E}\{h(\lambda_n)). \tag{3.37}$$

In general, both the input and output bit values X_n and Y_n are equiprobable, hence $H(X_n) = H(Y_n) = 1$. Then Eq. (3.36) and (3.37) may be simplified to

$$\begin{aligned} I_{E_u} &= \frac{1}{N_u} \sum_{n=0}^{N_u-1} (1 - \mathrm{E}\{h(E_{u,n} = \lambda_n)) \\ &= \frac{1}{N_u} \sum_{n=0}^{N_u-1} \mathrm{E}\{f_I(E_{u,n} = \lambda_n)\}, \end{aligned} \tag{3.38}$$

$$I_{E_c} = \frac{1}{N_c} \sum_{n=0}^{N_c-1} \mathrm{E}\{f_I(E_{c,n} = \lambda_n), \tag{3.39}$$

where $f_I(\cdot)$ is

$$\begin{aligned} f_I(\lambda) &\triangleq 1 - h(\lambda) \\ &= \frac{1}{1+e^{\lambda}} \log_2 \frac{2}{1+e^{\lambda}} + \frac{1}{1+e^{-\lambda}} \log_2 \frac{2}{1+e^{-\lambda}}. \end{aligned} \tag{3.40}$$

Assume furthermore that the *a posteriori* LLRs are ergodic and the block length is sufficiently high. Then the expectation values in Eq. (3.38) and (3.39) can be replaced by the time averages [136, 165]

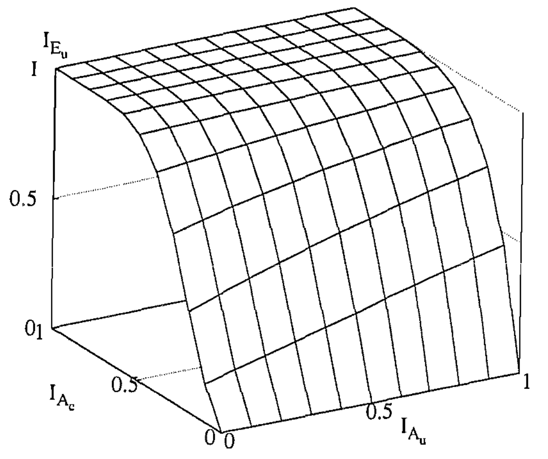
$$I_{E_u} \approx \frac{1}{N_u} \sum_{n=0}^{N_u-1} f_I(E_{u,n} = \lambda_n), \tag{3.41}$$

$$I_{E_c} \approx \frac{1}{N_c} \sum_{n=0}^{N_c-1} f_I(E_{c,n} = \lambda_n). \tag{3.42}$$

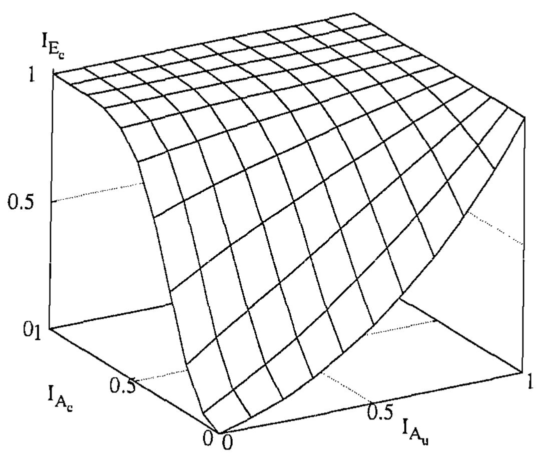
The method outlined in Eqs (3.41) and (3.42) has the following advantages: 1) No LLR histograms have to be recorded, hence the computational complexity is significantly reduced. This method can operate "on-line", because as soon as a new *a posteriori* LLR becomes available, it can be used to update the current estimate of the mutual information. 2) The results are unbiased even for codes with time-varying trellises such as punctured convolutional codes, and bits having different statistical properties do not have to be treated differently [136].

3.2.2.5 Examples

In this section, we will show examples of various EXIT functions, which belong to different SISO modules, and are evaluated using the procedure outlined in Section 3.2.2.3. In general, a SISO module needs two EXIT functions (Eq. (3.28) and Eq. (3.29) in order to characterise its information transfer behaviour in iterative processing, for example, when it is used as the intermediate decoder in multiple serial concatenation of component decoders. However, some of the input/output terminals of a SISO module may not be actively exploited in the iterative processing, hence the employment of a single EXIT function may be sufficient on some occasions. We will discuss these scenarios separately in our forthcoming discourse.



(a) $I_{E_u} = T_u(I_{A_u}, I_{A_c})$



(b) $I_{E_c} = T_c(I_{A_u}, I_{A_c})$

Figure 3.11: EXIT functions for CC(7,5) used as Decoder II in Fig. 3.3.

3.2.2.5.1 The Intermediate Code This is the general case. As seen from Fig. 3.3, the intermediate SISO decoder receives its *a priori* input of I_{A_c} from the preceding SISO decoder, while its *a priori* input of I_{A_u} from the successive decoder. Correspondingly, it feeds back the *extrinsic* outputs of I_{E_c} and I_{E_u}, respectively to the other two constituent decoders, as seen Fig. 3.3. Therefore, the pari of 2D EXIT functions described by Eq. (3.28) and Eq. (3.29) are needed, which may be depicted in two 3D charts. For example, we show the two 2D EXIT functions of a half-rate convolutional code having the octal generator polynomials of (7,5) in Fig. 3.11.

3.2.2.5.2 The Inner Code Some SISO modules, such as soft demappers used for example in [88] and SISO equalisers operate directly on channel observations. These SISO modules receive the *a priori* input I_{A_u} from the successive SISO module and feed back the *extrinsic* output I_{E_u} as seen in Fig. 3.2. The input terminal I_{A_c} and the output terminal I_{E_c} are not used. Hence only a single EXIT function is needed. Moreover, the *extrinsic* output I_{E_u} also depends on the channel input, hence I_{E_u} is usually expressed as a function of I_{A_u} combined with E_s/N_0 as a parameter, i.e. we have

$$I_{E_u} = T_u(I_{A_u}, E_s/N_0). \tag{3.43}$$

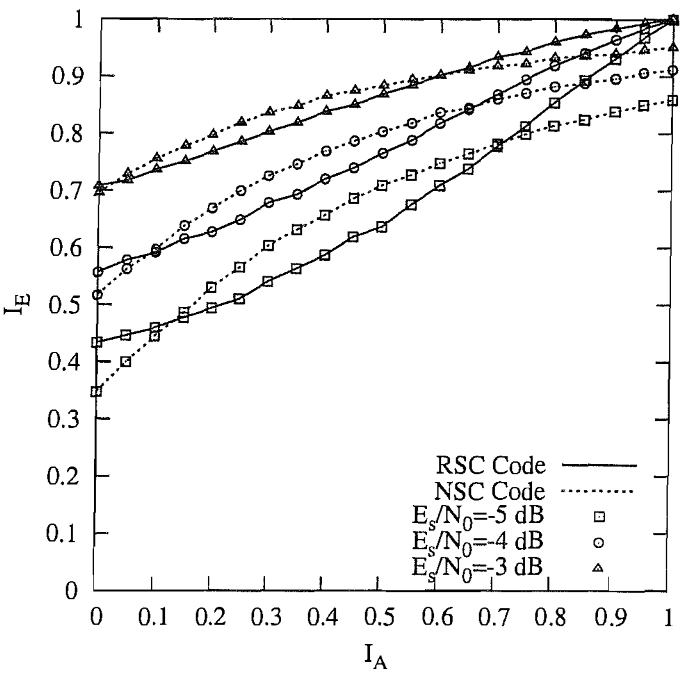


Figure 3.12: The EXIT functions of inner codes exemplified in Fig. 3.2. The half-rate RSC code having octal generator polynomials of (5/7) and the half-rate NSC code having the octal generator polynomials of $(7, 5)$ are used.

When using BPSK modulation, the EXIT function of the SISO decoder which is connected to the soft demapper as exemplified in Fig. 3.2 may also be characterised by Eq. (3.43), since no iterative processing between the demapper and the decoder is needed. For example, we depict the EXIT functions of a half-rate RSC code having octal generator polynomials of (5/7) and those of a half-rate Non-Systematic Convolutional (NSC) code using the same generator polynomials at several E_s/N_0 values in Fig. 3.12. It can be seen that the EXIT functions of the RSC code always satisfy the condition of $T_u(I_{A_u} = 1, E_s/N_0) = 1$, while the EXIT functions of the NSC code do not.

3.2.2.5.3 The Outer Code For SISO modules which are in the "outer" positions of the concatenated scheme exemplified in Fig. 3.2, the resultant EXIT functions may also be simplified to 1D functions. Since the input terminal I_{A_u} and the output terminal I_{E_u} are not used in the associated iterative processing, the EXIT function of an outer SISO module can be expressed as:

$$I_{E_c} = T_c(I_{A_c}). \tag{3.44}$$

Fig. 3.13 shows the EXIT functions of various non-recursive NSC codes used as outer codes. According to our informal observations, the RSC codes using the same generator polynomials have the

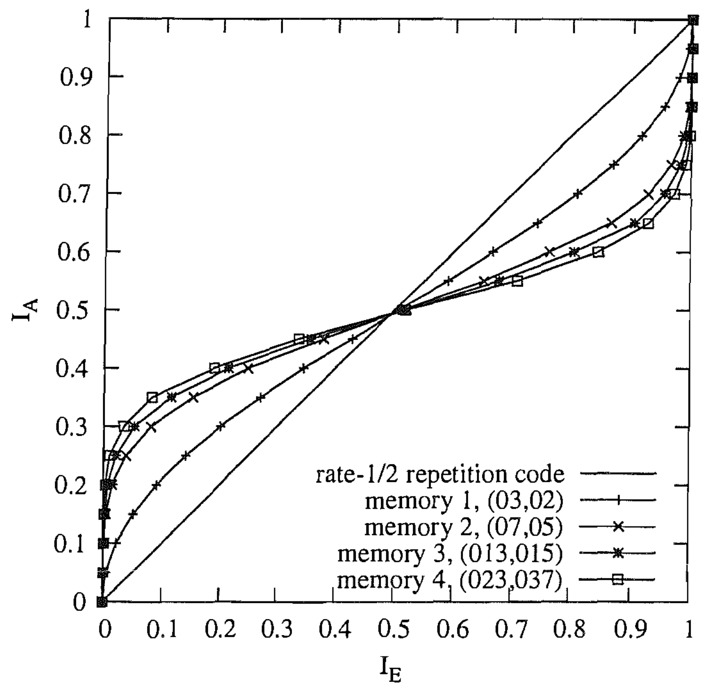


Figure 3.13: The EXIT functions of various outer rate 1/2 non-recursive NSC decoders, where (g_0, g_1) represent the octal generator polynomials.

same EXIT functions as the corresponding NSC codes.

3.3 Iterative Source/Channel Decoding Over AWGN Channels

This section investigates the performance of iterative source/channel decoding techniques. In general, channel coding is invoked after source coding for the sake of protecting the source information against the noise and other sources of impairments imposed by the transmission channel. Inspired by the iterative decoding philosophy of turbo codes [73], the source code and channel code together may be viewed as a serially concatenated code [75], which can be decoded jointly and iteratively, provided that both the source decoder and the channel decoder are soft-in and soft-out components [46]. The performance of the iterative receiver is evaluated for transmission over AWGN channels, and the EXIT chart technique [102] is employed for analysing the performance of iterative decoding and for assisting in the design of efficient systems. Let us first introduce our system model.

3.3.1 System Model

The schematic diagram of the transmission system considered is shown in Fig. 3.14. At the transmitter side of Fig. 3.14, the non-binary source $\{u_k\}$ represents a typical source in a video, image or

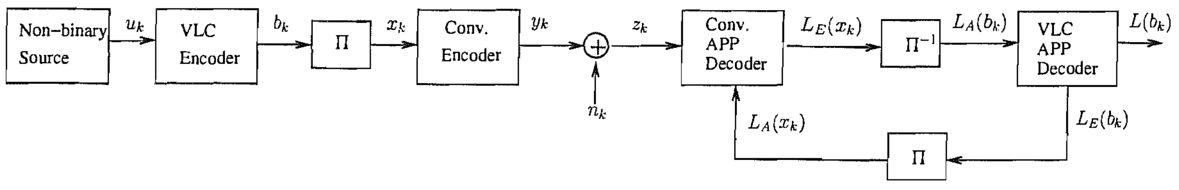


Figure 3.14: Iteratively detected source and channel coding scheme

text compression system. It is modelled as a discrete memoryless source using a finite alphabet of $\{a_1, a_2, \cdots, a_s\}$. The output symbols of the source are encoded by a source encoder, which outputs a binary sequence $\{b_k\}$. We assume that the source encoder is a simple entropy encoder, which employs a Huffman code [14], RVLC [17] or VLEC [34] code. As shown in Fig. 3.14, after interleaving, the source encoded bit stream is protected by an inner channel code against channel impariments. To be more specific, the inner code we employ is a convolutional code. Note that, the interleaver seen in Fig. 3.14 is used for reducing the correlation between the outer source encoded and inner channel encoded bit streams. At the receiver side of Fig. 3.14, an iterative source/channel decoding structure is invoked, as in [46]. The iterative receiver consists of an inner APP channel decoder and an outer APP source decoder. Both of them employ the SISO APP module described in Section 3.2.1 and exchange LLRs as their reliability information.

The first stage of an iteration is constitued by the APP channel decoder. The input of the channel decoder is the channel output z_k and the a *priori* information $L_A(x_k)$ provided by the outer VLC decoder, which is $L_A(x_k) = 0$ for the first iteration. Based on these two inputs, the channel decoder computes the a *posteriori* information $L(x_k)$ and only the *extrinsic* information $L_E(x_k) = L(x_k) - L_a(x_k)$ is forwarded to the outer VLC decoder.

The second stage of an iteration is constitued by the bit-level trellis based APP VLC decoder, which accepts as its input the *a priori* information $L_A(b_k)$ from the inner channel decoder and generates the *a posteriori* information $L(b_k)$. Similarly, only the *extrinsic* information $L_E(b_k) = L(b_k) - L_A(b_k)$ is fed back to the inner channel decoder. After the last iteration, the source sequence estimation based on the same trellis, namely on that of the APP VLC decoder is invoked, in order to obtain the decoded source symbol sequence. Generally, the symbols of the non-binary source have a non-uniform distribution, which may be interpreted as *a priori* information. In order to be able to exploit this *a priori* source information, the APP algorithm of the SISO module has to be appropriately adapted, as detailed in Appendix B.

Table 3.1: VLCs used in the simulations

Symbol	Probability	HUFF	RVLC-1	RVLC-2	VLEC-3
0	0.33	00	00	00	000
1	0.30	01	01	11	0110
2	0.18	11	10	010	1011
3	0.10	100	111	101	11010
4	0.09	101	11011	0110	110010
Average Length		2.19	2.37	2.46	3.95
Code Rate		0.98	0.90	0.87	0.54
Constraint Length		3	3.58	3.58	4.70
Free Distance		1	1	2	3

3.3.2 EXIT Characteristics of VLCs

We evaluated the associated EXIT characteristics for the various VLCs listed in Table 3.1. The corresponding results are depicted in Fig. 3.15.

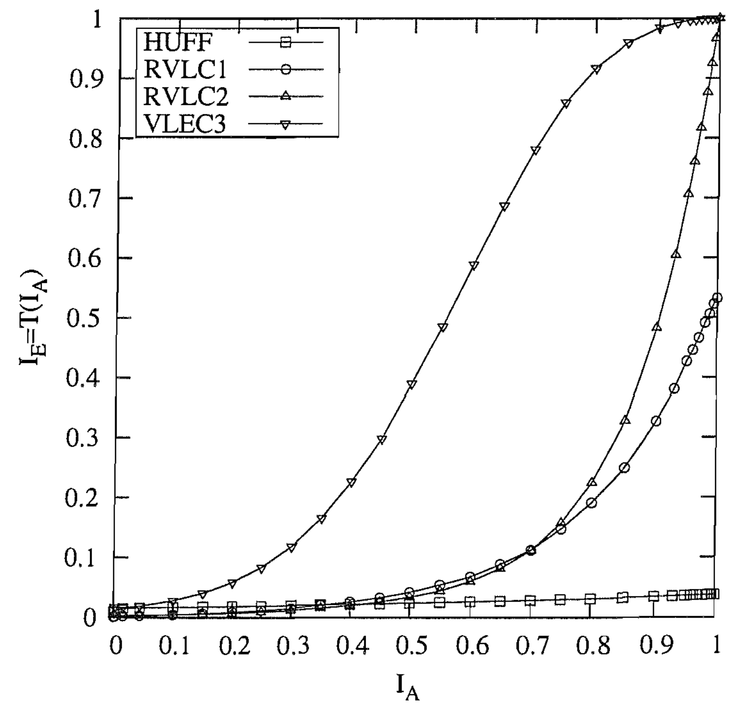


Figure 3.15: EXIT characteristics of VLCs listed in Table 3.1

It can be seen in the figure that for the Huffman code, the SISO VLC decoder only marginally benefits from having extrinsic information, even if exact *a priori* information is provided in the form of $I_A = 1$. Upon increasing the code's redundancy, i.e. decreasing the code rate, the SISO VLC decoder benefits from more extrinsic information, as observed for the RVLC-1, RVLC-2 and VLEC-3 schemes of Table 3.1, which indicates that higher iteration gain will be obtained, when using iterative decoding. It is also worth noting that for VLCs having a free distance of $d_f \geq 2$, such as the RVLC-2 and VLEC-3 schemes, the condition of $T(I_A = 1) = 1$ is satisfied. By contrast, for those having $d_f = 1$, this condition is not satisfied, which indicates that a considerable number of residual errors will persist at in the decoder's output even in the high SNR areas.

3.3.3 Simulation Results for AWGN Channels

In this section we present some simulation results for the proposed transmission scheme. All simulations were performed for BPSK modulation when communicating over an AWGN channel. The channel code used is a terminated memory-2, rate-$1/2$ recursive systematic convolutional (RSC) code having the generator polynomial of $(g_1/g_0)_8 = (5/7)_8$, where g_0 is the feedback polynomial. For every simulation, 1000 different frames of length $K = 1000$ source symbols were encoded and transmitted. For the iterative receiver, six decoding iterations were executed, except for the Huffman code, where only 3 iterations were performed. The simulation were terminated, when either all the frames were transmitted or the number of detected source symbol errors reaches 1000.

For the source encoder, we use the VLCs as listed in Table 3.1. Since these codes have different rates, for the sake of fair comparison, the associated E_b/N_0 values have to be scaled by both the source code rate R_s and the channel code rate R_c as follows.

$$E_b/N_0 = (E_s/(R_s \cdot R_c \cdot \log_2 M))/N_0, \tag{3.45}$$

or in terms of decibels

$$E_b/N_0\ [dB] = E_s/N_0\ [dB] - 10\log_{10}(R_s \cdot R_c \cdot \log_2 M), \tag{3.46}$$

where E_s is the transmitted energy per modulated symbol, M is the number of signals in the signal space of M-ary modulation. For example, $M = 2$, for BPSK modulation. The effect of the tail bits of the channel code is ignored here.

For every VLC, the transfer functions of the outer VLC decoder and inner channel decoder are plotted in the form of EXIT charts in order to predict the performances of the iterative receivers.

The EXIT chart of the iterative receiver using the Huffman code of Table 3.1 and the corresponding SER performance are depicted in Fig. 3.16 and 3.17, respectively. In the EXIT chart, the transfer functions of the inner RSC decoder recorded at $E_b/N_0 = 0$ dB, 1 dB and 2 dB are plotted. Moreover, the *extrinsic* information outputs, I_E^{VLC} and I_E^{RSC}, of both decoders were recorded during our simulations and the average values at each iteration were plotted in the EXIT chart, which form the so-called *decoding trajectories*. The iterative decoding algorithm may possibly converge to a near-zero BER, when the decoding trajectory approaches the point $(I_A = 1, I_E = 1)$. However, it can be readily seen that since the outer Huffman code has a flat transfer characteristic, it provides only limited extrinsic information for the inner channel decoder, hence the overall performance can hardly be improved by iterative decoding. This is confirmed by the SER performance seen in Fig. 3.17, where only a modest iteration gain is attained.

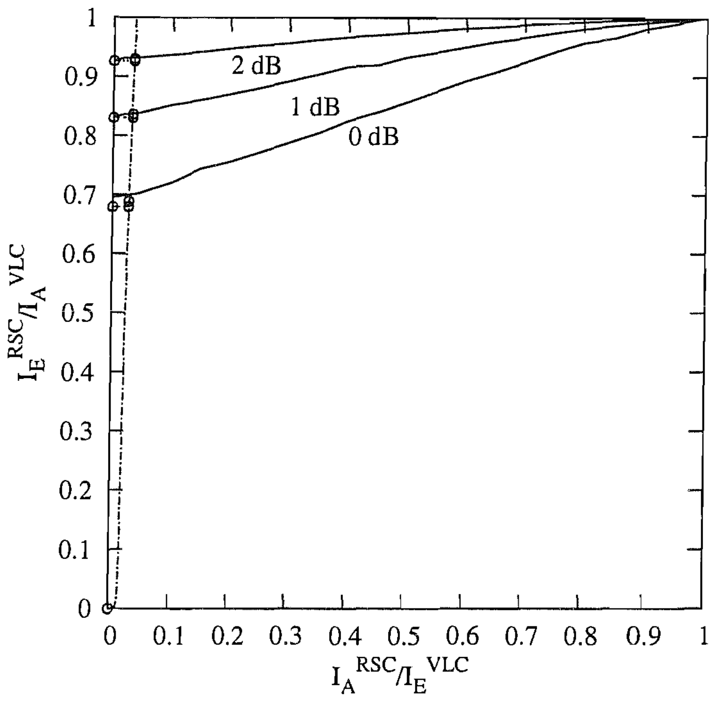


Figure 3.16: EXIT chart for the iterative receiver of Fig. 3.14 using a **Huffman code** having $d_f = 1$ of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

For the transmission scheme employing the RVLC-1 arrangement of Table 3.1, the iterative receiver did exhibit some iteration gain, as shown in Fig. 3.19. However, the achievable improvements saturated after $I = 2$ or 3 iterations. This is because the two EXIT curves intersect, before reaching the $(I_A = 1, I_E = 1)$ point, as shown in the EXIT chart of Fig. 3.18, and the trajectories get trapped at these intersect points, hence no more iteration gain can be obtained.

As stated before, the VLCs having $d_f \geq 2$ exhibit better transfer characteristics, hence also better convergence properties. This is confirmed in Fig. 3.21, where the SER of the system employing the RVLC-2 scheme of Table 3.1 was continuously reduced for up to six iterations. At SER $= 10^{-4}$, the iteration gain is as high as about 2.5 dB. As seen from the EXIT chart of Fig. 3.20, for $E_b/N_0 > 1$ dB there will be an open EXIT "tunnel" between the curves of the two transfer functions, which implies that the iterative decoding process may be carried on without getting curtailed, provided that

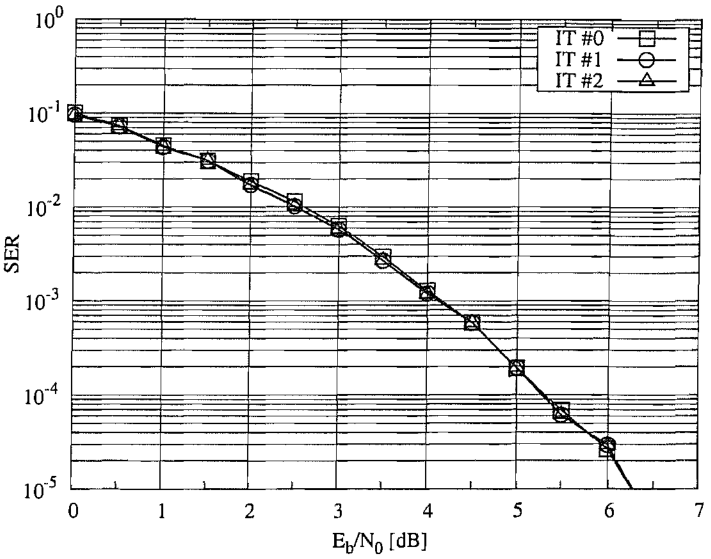


Figure 3.17: SER performance of the iterative receiver of Fig. 3.14 using a **Huffman code** of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

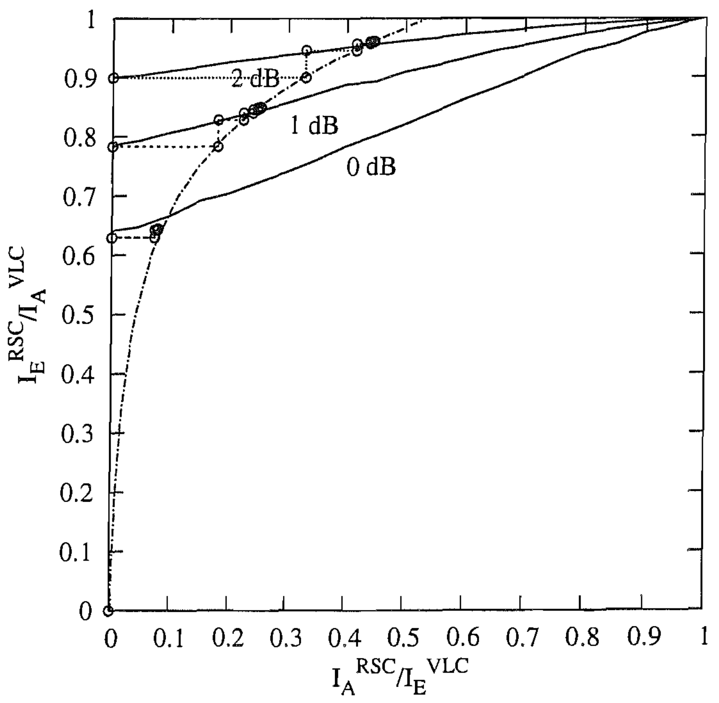


Figure 3.18: EXIT chart for the iterative receiver of Fig. 3.14 using a **RVLC** having $d_f = 1$ of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

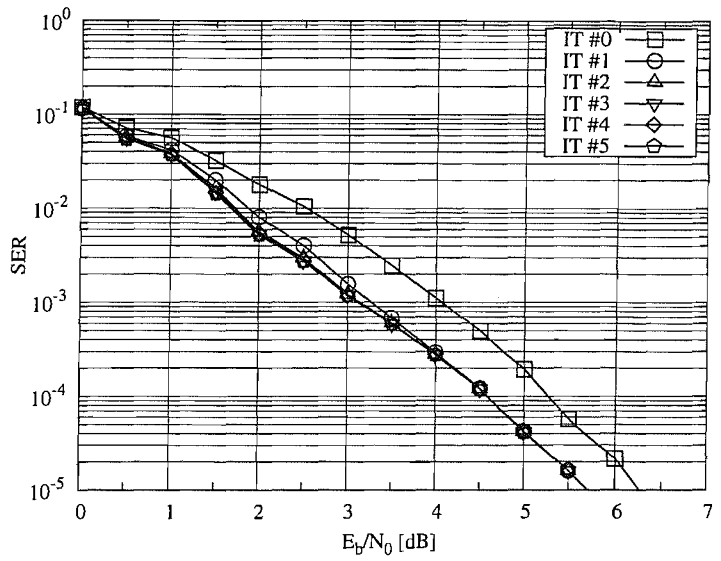


Figure 3.19: SER performance of the iterative receiver of Fig. 3.14 using a **RVLC** having $\mathbf{d_f = 1}$ of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

the interleaver length is sufficiently high. In the corresponding SER chart of Fig. 3.21, the SER curve approaches the so-called "waterfall" region or "turbo cliff" region beyond this point. Hence, this E_b/N_0 value is also referred to as the *convergence threshold*. Also seen from the EXIT chart of Fig. 3.20 that beyond the convergence threshold, the EXIT "tunnel" between the two curves becomes narrow towards the top right corner near the point $I_A = 1, I_E = 1$. This indicates that the iteration gains will become lower during the course of iterative decoding. Particularly, in the case of a limited interleaver size and for a low number of iterations, a considerable number of residual errors will persist at the decoder's output and an error floor will appear even in the high channel SNR region.

Fig. 3.22 and Fig. 3.23 show the EXIT chart and SER performance of the system respectively when employing the VLEC-3 scheme of Table 3.1. The convergence threshold of this system is about $E_b/N_0 = 0$ dB. Hence beyond this point, a significant iteration gain was obtained, as a benefit of the excellent transfer characteristics of the VLEC-3 code. At SER $= 10^{-4}$, the achievable E_b/N_0 gain is about 4 dB after six iterations and no error floor is observed. A performance comparison of the systems employing different VLCs is depicted in Fig. 3.24. Clearly, in the context of iterative decoding, when the inner code is fixed, the achievable system performance is determined by the free distance of the outer code. The VLEC code having a free distance of $d_f = 3$ performs best after $I = 6$ iterations, while it has the lowest code rate and hence results in the lowest achievable system throughput. Similarly, the RVLC-2 scheme having $d_f = 2$ also outperforms the RVLC-1 having $d_f = 1$ and the Huffman code.

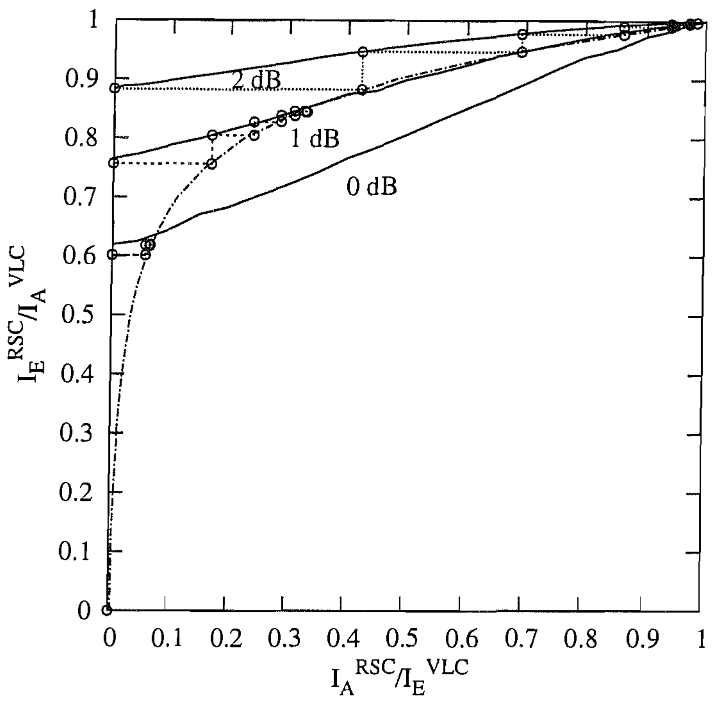


Figure 3.20: EXIT chart for the iterative receiver of Fig. 3.14 using a **RVLC** having $\mathbf{d_f = 2}$ of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

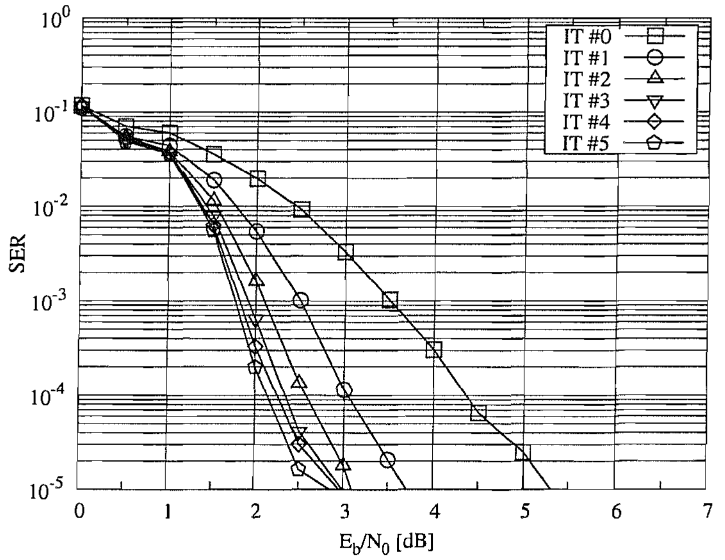


Figure 3.21: SER performance of the iterative receiver of Fig. 3.14 using a **RVLC** having $\mathbf{d_f = 2}$ of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

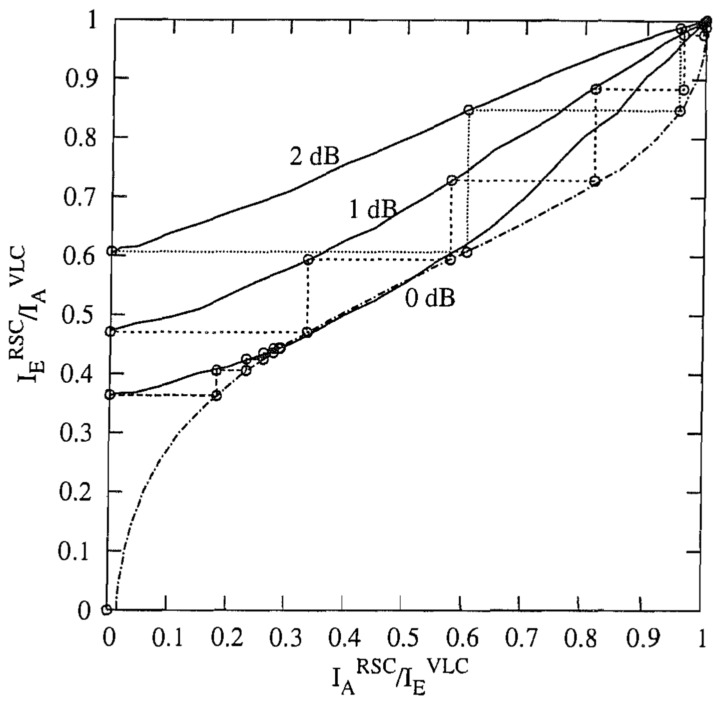


Figure 3.22: EXIT chart for the iterative receiver of Fig. 3.14 using a **VLEC code** having $\mathbf{d_f = 3}$ of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

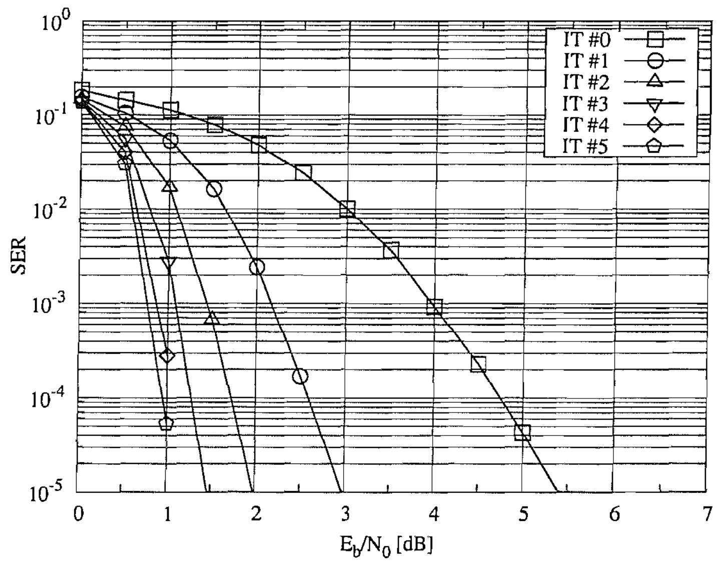


Figure 3.23: SER performance of the iterative receiver of Fig. 3.14 using a **VLEC code** having $\mathbf{d_f = 3}$ of Table 3.1 as the outer code and the RSC(2,1,2) code as the inner code for transmission over AWGN channels.

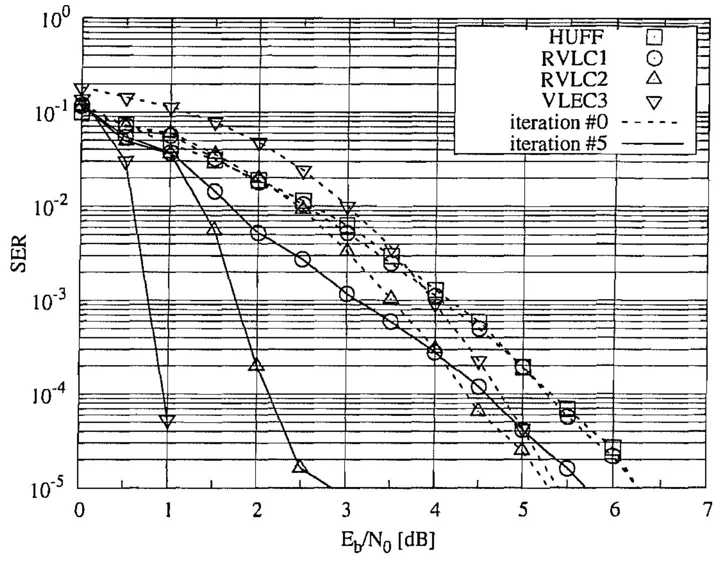


Figure 3.24: Performance Comparison of the schemes employing different VLCs including Huffman code, RVLC-1 having $d_f = 1$, RVLC-2 having $d_f = 2$ and VLEC-3 having $d_f = 3$ of Table 3.1, after the first and the sixth iterations.

3.4 Iterative Channel Equalisation, Channel Decoding and Source Decoding

In practical communications environments, the same signal may reach the receiver via multiple paths, as a result of scattering, diffraction and reflections. In simple terms, if all the significant multipath signals reach the receiver within a fraction of the duration of a single transmitted symbol, they result in non-dispersive fading, whereby occasionally they add destructively resulting in a low received signal amplitude. However, if the arrival time difference among the multipath signals is higher than the symbol duration, then the received signal inflicts dispersive time-domain fading, yielding Inter-Symbol-Interference (ISI).

The fading channel hence typically inflicts bursts of transmission errors and the effects of these errors may be mitigated by adding redundancy to the transmitted signal in the form of error correction coding, such as convolutional coding as discussed in the previous section. However, adding redundancy reduces the effective throughput perceived by the end-user. Furthermore, if the multipath signals are spread over multiple symbols, they tend to further increase the error rate experienced, unless a channel equaliser is used. The channel equaliser collects all the delayed and faded multipath replicas, and linearly or non-linearly combines them [166]. The equalisers typically achieve a diversity gain, because even if one of the signal paths becomes severely faded, the receiver may still be able to extract sufficient energy from the other independently faded paths.

Capitalising on the impressive performance gains of turbo codes, turbo equalisation [86, 117, 167] has been proposed as a combined iterative channel equalisation and channel decoding technique that has the potential of achieving equally impressive performance gains when communicating over ISI-contaminated channels. The turbo-equalisation scheme introduced by Douillard *et al.* [86] may be viewed as an extension of the turbo decoding algorithm [73], which interprets the information spreading effects of the ISI as another form of error protection, reminiscent of that imposed by a rate-1 convolutional code. To elaborate a little further, the impulse response of the dispersive channel has the effect of smearing each transmitted symbol over a number of consecutive symbols, similarly to the action of a convolutional encoder.

By applying the turbo detection principle of exchanging extrinsic information between channel equalisation and source decoding, the receiver becomes capable of efficiently combatting the effects of ISI. In the following sections, we will describe such a scheme, where the redundancy inherent in the source is exploited for the sake of eliminating the channel-induced ISI. Transmission schemes both with and without channel coding are considered.

3.4.1 Channel Model

We assume having a coherent symbol-spaced receiver benefiting from the perfect knowledge of the signal's phase and symbol timing, so that the transmit filter, the channel and the receiver filter can be jointly represented by a discrete-time linear filter having a finite-length impulse response expressed as

$$h_k = \sum_{m=0}^{M} h_m \delta(k-m), \tag{3.47}$$

where the real-valued channel coefficients $\{h_m\}$ are assumed to be time-invariant and known to the receiver. Fig. 3.25 shows two examples of ISI-contaminated dispersive channels and their frequency domain responses introduced in [13]. The channel characterised in the upper two figures is a three-path channel having moderate ISI, while the channel depicted in the lower two figures is a five-path channel suffering from severe ISI.

Given the Channel Impulse Response (CIR) of Eq. (3.47) and a BPSK modulator, the channel output y_k, which is also the equaliser's input, is given by

$$y_k = \sum_{m=0}^{M} h_m d_{k-m} + n_k, \; k = 1, \ldots, N_c, \tag{3.48}$$

where n_k is the zero-mean AWGN having a variance of σ^2. We assume that we have $N_c = N_b + N_h$,

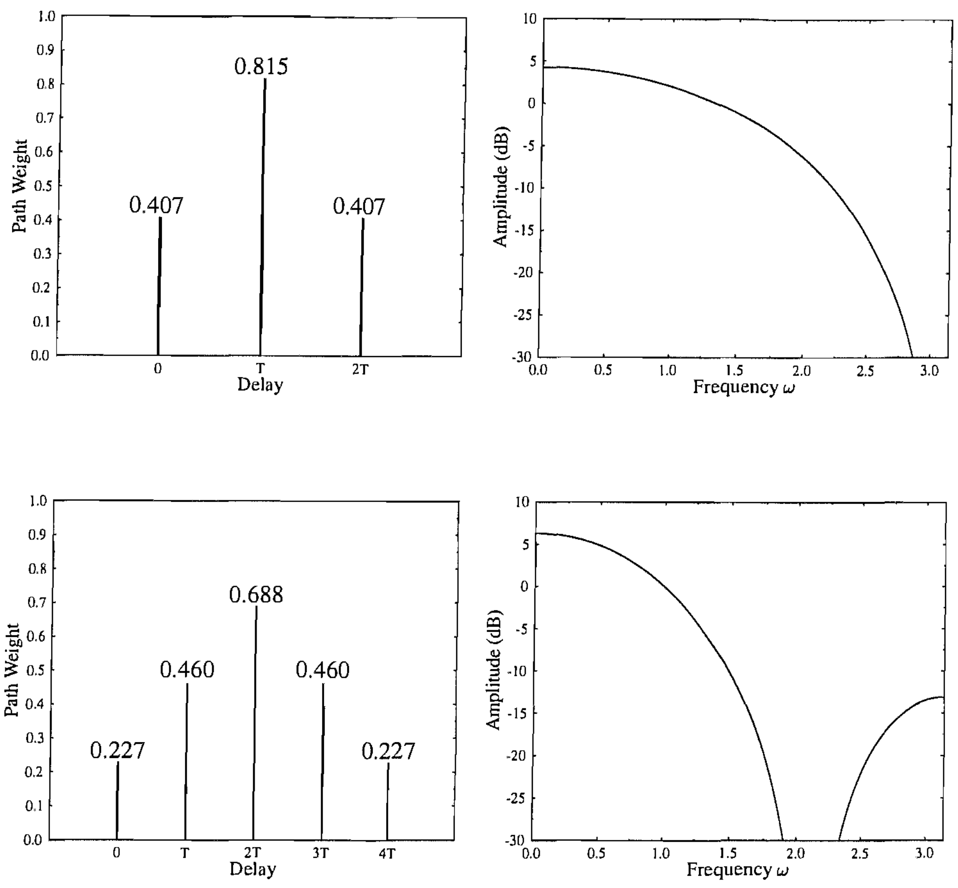


Figure 3.25: Two dispersive channel impulse responses (CIRs) and their frequency-domain representations

where N_h represents the so-called tailing bits concatenated by the transmitter, in order to take into account the effect of ISI induced delay. More specifically, $N_h = M$ number of binary zeros may be appended at the tail of an N_b-bit message, so that the discrete-time linear filter's buffer content converges to the all-zero state. Explicitly, the channel's output samples y_k obey a conditional Gaussian distribution having a PDF expressed as

$$p_{y_k}(y|d_k, d_{k-1}, \ldots, d_{k-M}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{\left[y - \sum_{m=0}^{M} h_m d_{k-m} \right]^2}{2\sigma^2} \right). \tag{3.49}$$

Fig. 3.26 shows a tapped delay line model of the equivalent discrete-time channel of Eq. (3.48) for $M = 2$, associated with the channel coefficients of $h_0 = 0.407$, $h_1 = 0.815$ and $h_2 = 0.407$. Assuming an impulse response length of $(M + 1)$, the tapped delay line model contains M delay elements. Hence, given a binary input alphabet of $\{+1, -1\}$, the channel can be in one of 2^M states

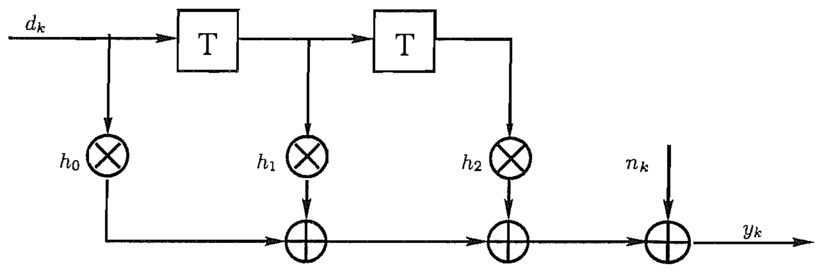


Figure 3.26: Tapped delay line circuit of the channel model Eq. (3.48) for $M = 2$.

q_i, $i = 1, 2, \cdots, 2^M$, corresponding to the 2^M different possible binary contents in the delay elements. Let us denote the set of possible states by $\mathcal{S} = \{q_1, q_2, \cdots, q_{2^M}\}$. At each time instance of $k = 1, 2, \cdots, N_c$ the state of the channel is a random variable $s_k \in \mathcal{S}$. More explicitly, we define $s_k = (d_k, d_{k-1}, \cdots, d_{k-M+1})$. Note that given s_k, the next state s_{k+1} can only assume one of the two values determined by a $+1$ or -1 being fed into the tapped delay line model at time $k+1$. The possible evolution of states can thus be described in the form of trellis diagram. Any path through the trellis corresponds to a sequence of input and output symbols read from the trellis branch labels, where the channel's output symbol ν_k at time instance k is the noise-free output of the channel model of Eq. (3.48) given by:

$$\nu_k = \sum_{m=0}^{M} h_m d_{k-m}. \tag{3.50}$$

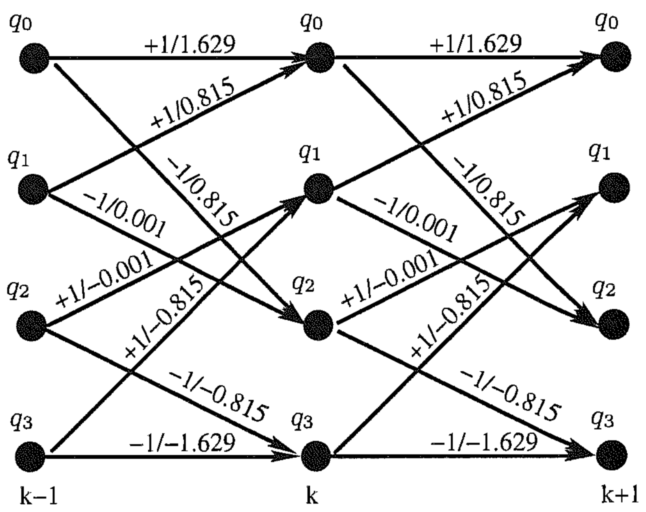


Figure 3.27: Trellis representation of the channel seen in Fig. 3.26. The states $q_0 = (1, 1), q_1 = (-1, 1), q_2 = (1, -1), q_3 = (-1, -1)$ are the possible contents of the delay elements in Figure 3.26. The transitions from a state $s_k = q_i$ at time k to a state $s_{k+1} = q_j$ at time $k+1$ are labeled with the input/output pair $d_{i,j}/\nu_{i,j}$.

The trellis constructed for the channel of Fig. 3.26 is depicted in Fig. 3.27. A branch of the trellis is a four-tuple $(i, j, d_{i,j}, \nu_{i,j})$, where the state $s_{k+1} = q_j$ at time $k+1$ can be reached from state $s_k = q_i$ at time k, when encounters the input $d_k = d_{i,j}$ and output $\nu_k = \nu_{i,j}$, where $d_{i,j}$ and $\nu_{i,j}$ are uniquely identified by the index pair (i, j). The set of all index pairs (i, j) corresponding to valid branches is denoted as $\mathcal{B}$. For the trellis seen in Fig. 3.27, the set $\mathcal{B}$ is as follows:

$$\mathcal{B} = \{(0,0), (0,1), (1,2)(1,3), (2,0), (2,1), (3,3), (3,2)\}. \tag{3.51}$$

3.4.2 Iterative Channel Equalisation and Source Decoding

In this section we investigate the principles of iterative equalisation as well as source decoding and derive the *extrinsic* information exchanged between the equaliser and the source decoder. We assume that the channel is an ISI-contaminated Gaussian channel, as expressed in Eq. (3.48).

3.4.2.1 System Model

At the transmitter side, the output symbols of the non-binary source $\{x_n\}$ seen in Fig. 3.28 are forwarded to an entropy encoder, which outputs a binary sequence $\{b_k\}$, $k = 1, 2, \cdots, N_b$. The source code used in the entropy encoder may be a Huffman code, RVLC [17] or VLEC code [34]. Then, the resultant binary sequence is interleaved and transmitted over an ISI-contaminated channel also imposing AWGN, as shown in Fig. 3.28.

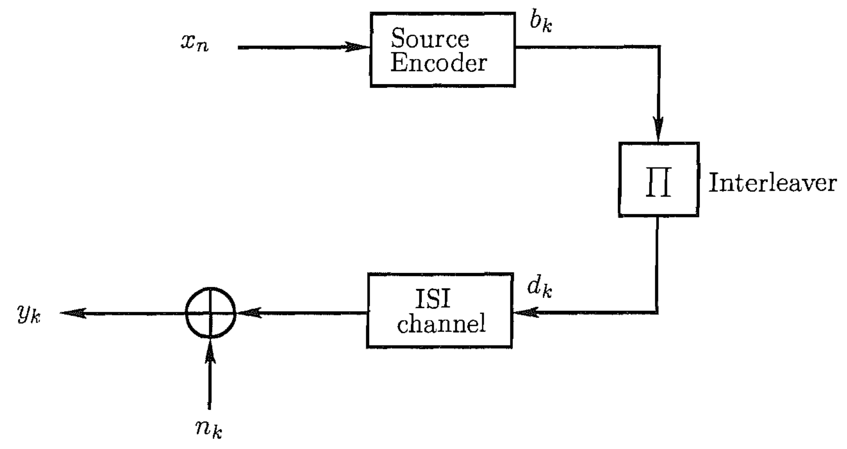


Figure 3.28: Representation of a data transmission system including source coding and a turbo interleaver communicating over an ISI-channel inflicting additive white Gaussian noise.

The receiver structure performing iterative equalisation and source decoding is shown in Fig.3.29. The ultimate objective of the receiver is to provide estimates of the source symbols $\{x_n\}$, given the channel output samples $\mathbf{y} \triangleq y_1^{N_c}$, where we have $y_1^{N_c} = \{y_1, y_2, \cdots, y_{N_c}\}$. More specifically, in the receiver structure of Fig. 3.29, the channel equaliser's feed-forward information generated in the context of each received data bit is applied to the source decoder through a deinterleaver described by the turbo deinterleaver function Π^{-1}. The information conveyed along the feed-forward path is described by $L_f(\ )$. By contrast, the source decoder feeds back the *a priori* information associated with each data bit to the channel equaliser through a turbo interleaver having the interleaving function described by Π. The information conveyed along the feedback path is described by $L_b(\ )$. This process is repeated for a number of times, until finally, the source decoder computes the estimates $\{\hat{x}_n\}$ for the source symbols $\{x_n\}$, with the aid of the information provided by $\{L_f(b_m)\}$. Let us first investigate the equalisation process in a little more detail.

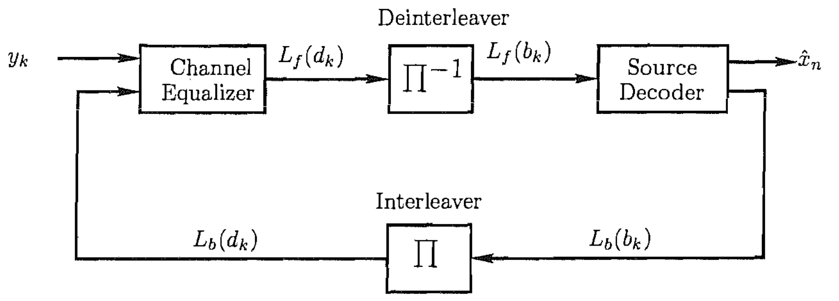


Figure 3.29: Receiver schematic performing iterative channel equalisation and source decoding.

3.4.2.2 EXIT Characteristics of Channel Equaliser

The channel equalisation considered here is based on the APP algorithm, which renders the equaliser another manifestation of the SISO APP module described in Section 3.2.1. More details of the APP channel equalisation can be found in Appendix C. Let us now study the transfer characteristics of the channel equaliser. Consider two ISI channels, namely the three-path and the five-path CIRs depicted in Fig. 3.25, which are represented by their CIR taps as

$$\mathbf{H}_1 = [0.407\ 0.815\ 0.407]^T; \tag{3.52}$$

$$\mathbf{H}_2 = [0.227\ 0.46\ 0.688\ 0.46\ 0.227]^T. \tag{3.53}$$

Fig. 3.30 shows the EXIT functions of the channel equaliser for the dispersive AWGN channels of Eq. (3.52) and Eq. (3.53) at different channel SNRs. It can be seen that the channel equaliser has the following transfer characteristics:

1) The EXIT curves are almost straight lines and those recorded for the same channel at different

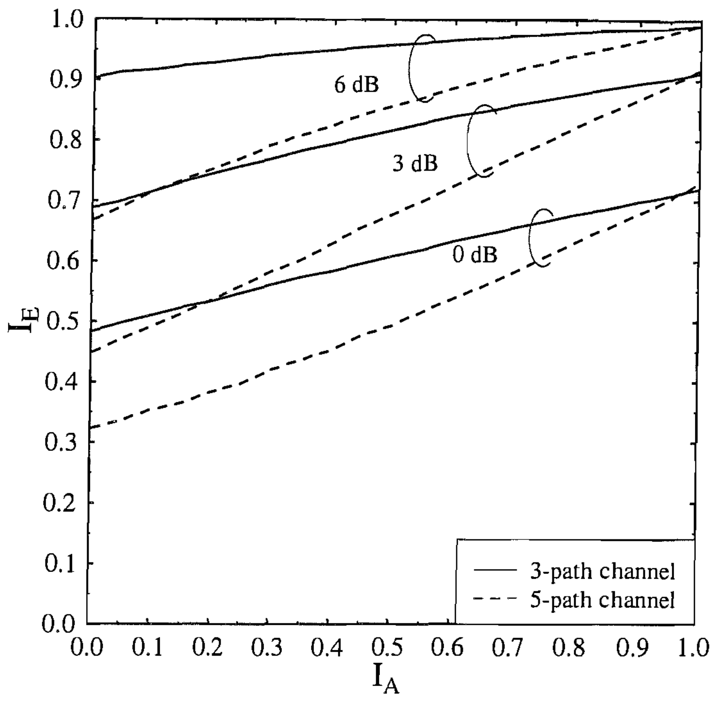


Figure 3.30: EXIT functions of the channel equaliser for transmission over the dispersive AWGN channels of Eq. (3.52) and Eq. (3.52)

SNRs are almost parallel to each other;

2) At low SNRs, $I_E = T(I_A = 1)$ can be significantly less than 1and $I_E = T(I_A = 1)$ will be close to 1 only at high SNRs. This indicates that considerable number of residual errors will persist at the output of the channel equaliser at low to medium SNRs;

3) Observe in Fig. 3.30 that the channel equaliser generally provides less extrinsic information I_E for the 5-path channel H_2 than for the 3-path channel H_1 at the same input *a priori* information I_A, which predicts potentially worse BER performance. Interestingly, the transfer functions of the two equalisers almost merge at $I_A = 1$, i.e. they have a similar value of $T(I_A = 1)$. This indicates that they shall have the same BER performance bound provided that all ISI was eliminated.

3.4.2.3 Simulation Results

In this section, we evaluate the SER performance of the iterative receiver. In the simulations, we used the three-path channel of Eq. (3.52) and the five-path channel of Eq. (3.53). For the source encoder, we employed two different VLCs, namely the RVLC-2 and VLEC-3 schemes listed in Table 3.1. The encoded data is permutated by a random bit interleaver of size $L = 4096$ bits.

The SER was calculated by using Levenshtein distance [157] for the combined transceiver using an RVLC, which was depicted in Fig. 3.31, when transmitting over the 3-path ISI channel $\mathbf{H}_1$ of Eq. (3.52) and over the 5-path ISI channel $\mathbf{H}_2$ of Eq. (3.53). The performance of the same system over a non-dispersive AWGN channel is also depicted as a best-case "bound". It can be seen in Fig. 3.31 that the iterative equalisation and source decoding procedure converges, as the number of iterations increases. This behaviour is similar to that of a turbo equaliser [86] and effectively reduces the detrimental effects of ISI after $I = 4$ iterations, potentially approaching the non-dispersive AWGN channel's performance

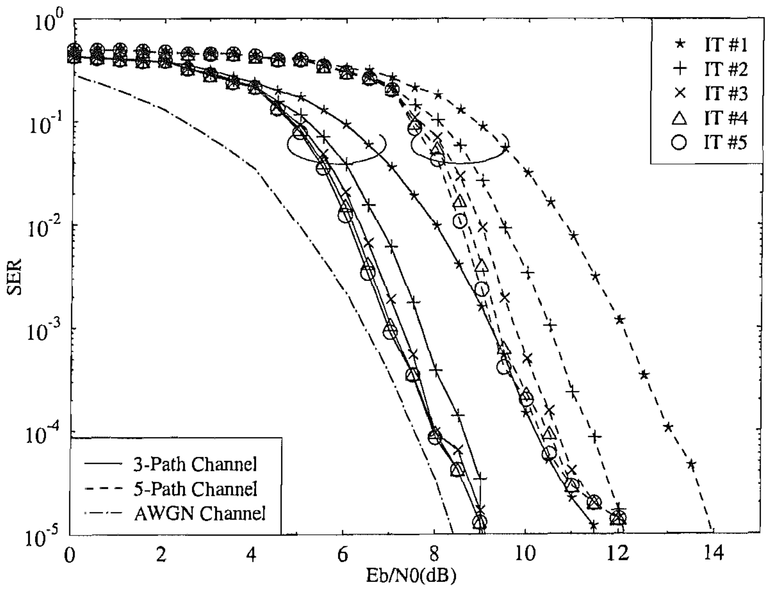


Figure 3.31: SER performance of the RVLC-2 of Table 3.1 for transmission over the dispersive AWGN channels of Eq. (3.52) and (3.53)

The SER performance of the system using a VLEC code of Table 3.1 is depicted in Fig. 3.32, when transmitting over the 3-path ISI channel $\mathbf{H}_1$ of Eq. (3.52) and over the 5-path ISI channel $\mathbf{H}_2$ of Eq. (3.53). Since the VLEC code has a larger free distance, hence a stronger error correction capability, the iterative equalisation and source decoding procedure succeeds in effectively eliminating the ISI and approaches the non-dispersive AWGN performance bound after $I = 4$ iterations.

3.4.2.4 Performance Analysis Using EXIT Charts

In order to analyse the simulation results obtained, we measured the EXIT characteristics of both the channel equaliser as well as those of the VLC decoder, and plotted their EXIT charts in Fig. 3.33.

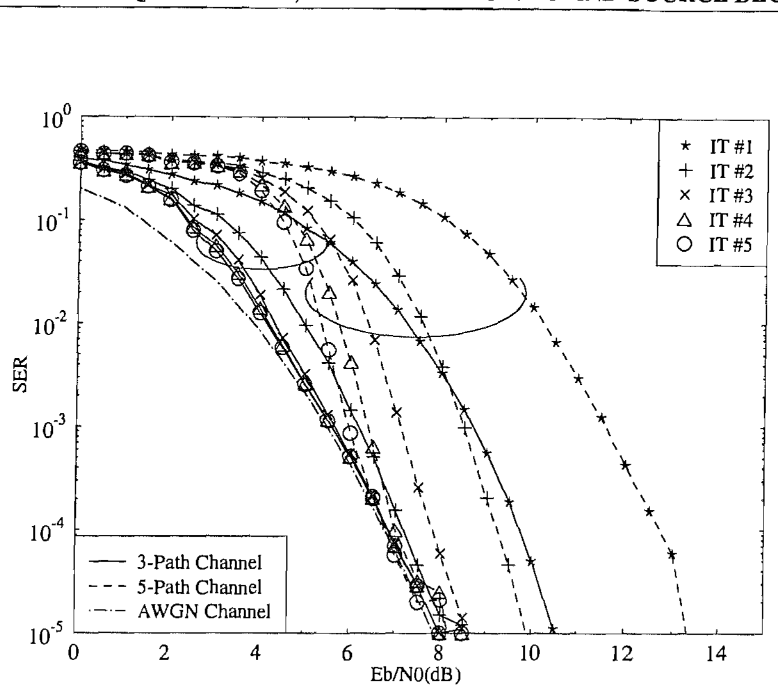
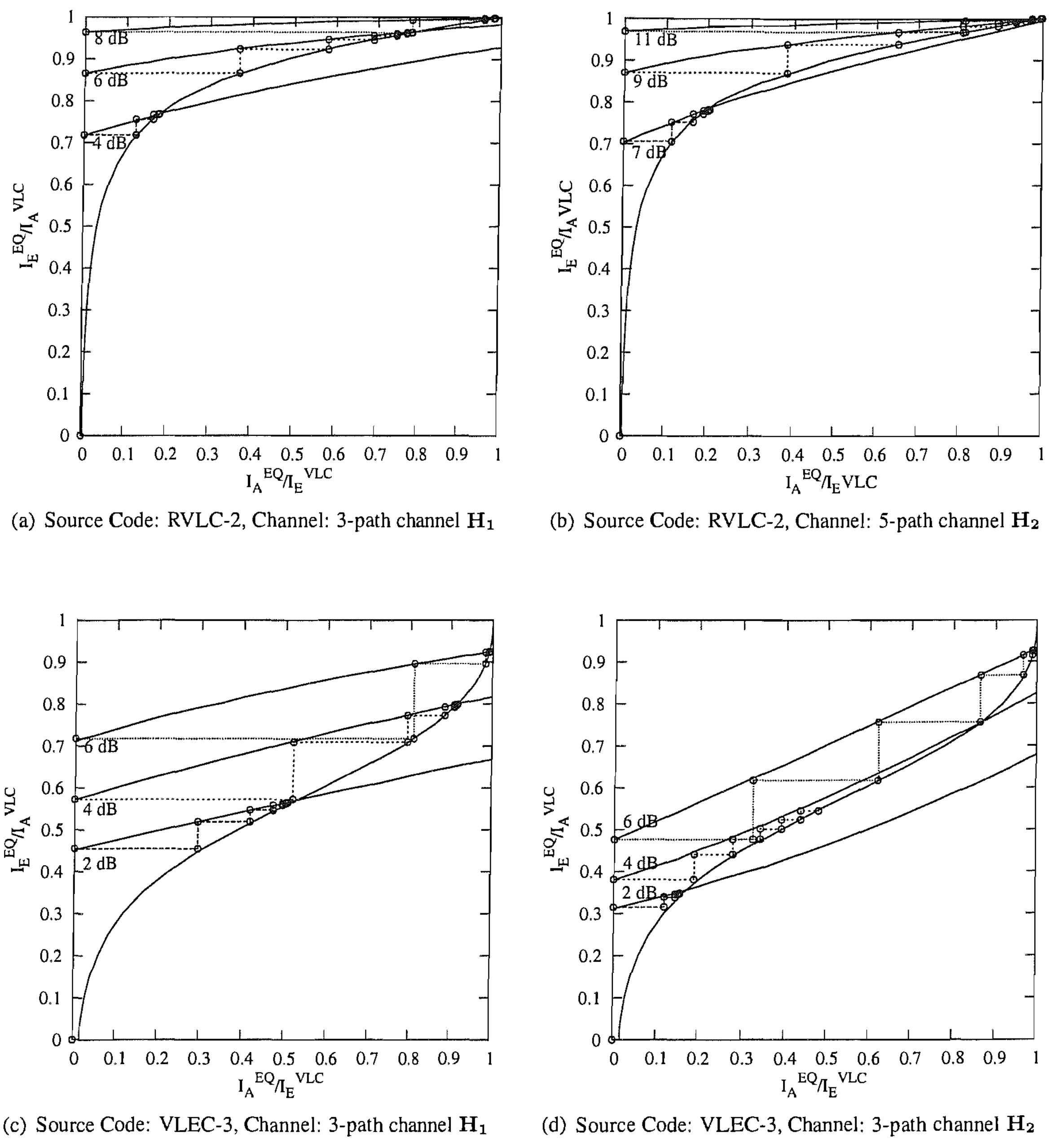


Figure 3.32: SER performance of the VLEC code of Table 3.1 for transmission over the dispersive AWGN channels of Eq. (3.52) and (3.53)

Specifically, both the EXIT chart and the simulated decoding trajectory of the system using RVLC-2 of Table 3.1 for communicating over the 3-path channel $\mathbf{H}_1$ are shown in Fig. 3.33(a). At $E_b/N_0 = 4$ dB, the transfer function of the channel equaliser and that of the VLC decoder intersects in the middle and after three iterations the trajectory gets trapped, which implies that little iteration gain can be obtained, even if more iterations are executed. Upon increasing E_b/N_0, the two transfer functions intersect at higher (I_A, I_E) points and the tunnel between the two EXIT curves becomes wider, which now indicates that higher iteration gains can be achieved and hence the resultant SER becomes lower. Similar observations can be made from the EXIT chart of the same system communicating over the 5-path channel $\mathbf{H}_2$ of Fig. 3.33(b), except that the channel equaliser needs an almost 3 dB higher E_b/N_0 to obtain a similar transfer characteristic. Hence the SER curves are shifted to the right by almost 3 dB, as shown in Fig. 3.31.

Fig. 3.33(c) shows the EXIT chart and the simulated decoding trajectory of the system using VLEC-3 of Table 3.1 for communicating over the 3-path channel $\mathbf{H}_1$. When compared to the system using RVLC-2 (Fig. 3.33(a)), there is a significantly wider tunnel between the two EXIT curves at the same E_b/N_0, hence higher iteration gains can be obtained at a lower number of iterations. This is mainly the benefit of the significantly lower coding rate of the VLEC code used ($R_{VLEC-3} = 0.54 < R_{RVLC-2} = 0.87$). However, the system still suffers from a significant number of residual



(a) Source Code: RVLC-2, Channel: 3-path channel $\mathbf{H_1}$

(b) Source Code: RVLC-2, Channel: 5-path channel $\mathbf{H_2}$

(c) Source Code: VLEC-3, Channel: 3-path channel $\mathbf{H_1}$

(d) Source Code: VLEC-3, Channel: 3-path channel $\mathbf{H_2}$

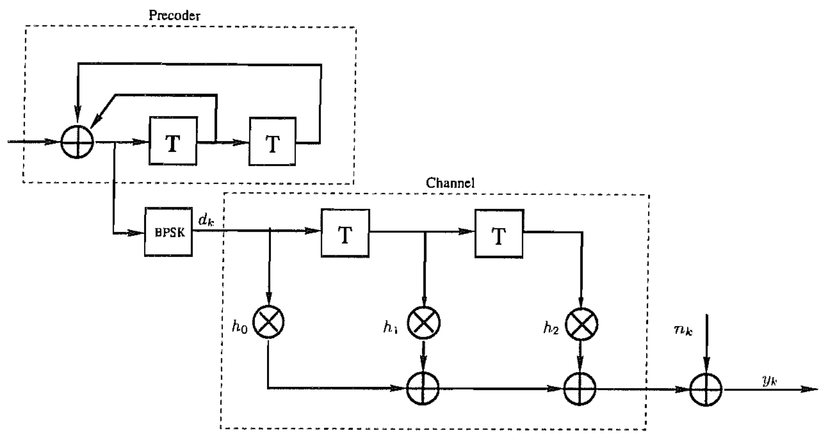
Figure 3.33: EXIT charts of the channel equaliser as well as the VLC decoder and the actual decoding trajectories at various E_b/N_0 values.

errors, since the two EXIT curves intersect in the middle and the resultant decoding trajectory gets trapped before reaching the top right corner of $(I_A = 1, I_E = 1)$. Fig. 3.33(d) shows the EXIT chart and the actual decoding trajectory of the same system when communicating over the 5-path channel $\mathbf{H}_2$. For the same E_b/N_0, the equaliser communicating over the 5-path channel has a steeper EXIT curve than for transmission over the 3-path channel, and its initial point at $(I_E^{EQ} = T(I_A^{EQ} = 0))$ is significantly lower. After the first iteration, the extrinsic output of the VLEC decoder remains smaller. Hence the SER performance recorded after the first iteration is always worse than that of the system communicating over the 3-path channel. At low E_b/N_0 values (e.g.<4 dB) the EXIT tunnel between the inner code's curve and the outer code's curve remains rather narrow. By contrast, at high E_b/N_0 values (e.g. $\geq$6 dB), the tunnel becomes significantly wide and has almost the same extrinsic output as that recorded for the 3-path channel (Fig. 3.33(c)) after 5 iterations. Hence the SER performances of the system recorded over the 3-path channel and 5-path channel converge at high E_b/N_0 values as shown in Fig. 3.32.

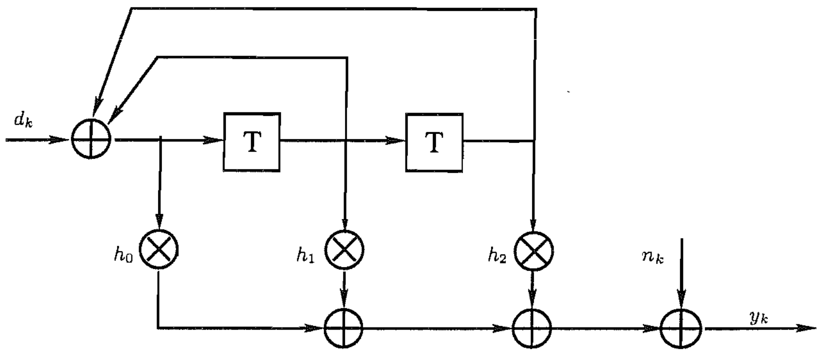
3.4.3 Precoding for Dispersive Channels

When the channel is non-recursive, which is the case for all wireless channels, then the gain of the iterative equalisation remains limited. More explicitly, the performance of the outer channel code for transmission over AWGN channels represents an upper-bound on the achievable receiver performance. In order to achieve a performance better than this, either the outer code can be replaced with a turbo code, as suggested in [168], or the channel can be made to appear recursive to the receiver, thus enabling us to achieve an higher interleaver gain. In the family of the latter techniques, recursive rate-one precoders, that effectively render the channel recursive, have been shown to yield significant performance gains in iterative equalisation systems [169]. The concept was first presented in [109, 170, 171] for magnetic recording channels and has subsequently been investigated in [169, 172] using BPSK modulations and in [138] using higher order modulations for general dispersive channels. A further advantage of the precoder is that there is no substantial increase in receiver complexity associated with its inclusion, as long as the memory of the precoder is no higher than that of the dispersive channel.

The precoder is generally described in terms of a shift register having feedback connections, described by a feedback polynomial. The structure is essentially the same as that of a recursive convolutional encoder. Fig. 3.34 shows an example of a precoded channel.



(a) Shift register representation of the precoder $1 + \mathrm{D} + \mathrm{D}^2$ concatenated with a BPSK modulator and the 3-path channel of Eq. (3.52)



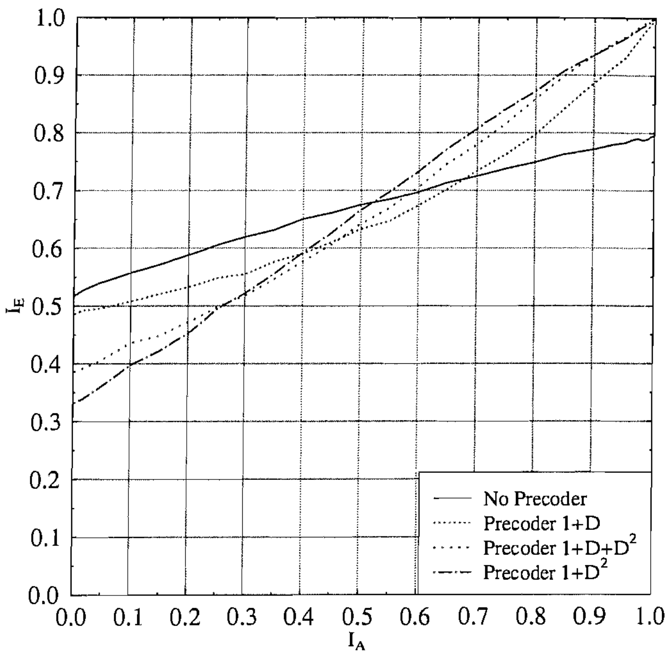
(b) Equivalent recursive channel

Figure 3.34: An example of precoding for BPSK modulation

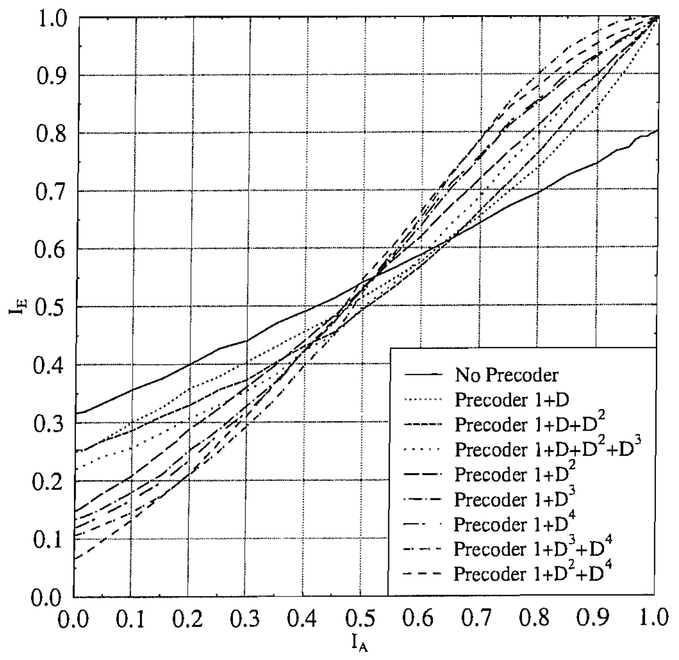
3.4.3.1 EXIT Characteristics of Precoded Channel Equalisers

In this section we investigate the EXIT characteristics of precoded channel equalisers. Again, the 3-path channel of Eq. (3.52) and the 5-path channel of Eq. (3.53) are used as examples. For the 3-path channel, the generator polynomials of the precoders having a memory of $m \leq 2$ are $1 + \mathrm{D}$, $1+\mathrm{D}+\mathrm{D}^2$ and $1+\mathrm{D}^2$. The EXIT functions of the operating channel equaliser both with and without these precoders for transmission over the 3-path channel are shown in Fig. 3.35(a). It can be seen that when using a precoder, the EXIT function can reach the $(I_A = 1,\ I_E = 1)$ point, while it is not the case for the non-precoded system. Hence the channel equaliser is capable of achieving near error-free detection for transmission over precoded channels, while it encounters an error rate floor when communicating over non-precoded channels. Furthermore, all the EXIT functions recorded for precoded channels have lower initial values at $(T(I_A = 0))$ in comparison to the non-precoded case, and this initial value decreases upon increasing the polynomial's order.

Similarly, we study the EXIT function of the channel equaliser for transmission over the 5-path



(a) 3-path channel of Eq. (3.52) at SNR=1 dB



(b) 5-path channel of Eq. (3.53) at SNR=1 dB

Figure 3.35: EXIT functions of the APP equaliser for transmission over precoded dispersive AWGN channels

channel. As the 5-path channel has a memory of 4, there are a total of 15 different precoders having a memory of $m \leq 4$. Only eight of them were selected for quantitative characterisation in Fig. 3.35(b). Similar observations are obtained as in the case of the 3-path channel, such as the transfer functions arrive at the $(I_A = 1, I_E = 1)$ point when using precoders as well as having lower start values.

3.4.3.2 Performance Analysis

The performance analysis of the joint channel equalisation and source decoding system communicating over precoded channels is presented in this section. As observed in Section 3.4.3.1, precoders have similar effects to the channel equaliser's EXIT characteristics in the case of both the 3-path and 5-path channel. Hence, we only consider the 3-path channel here and the precoders used are $(1 + \mathrm{D})$ and $(1 + \mathrm{D}^2)$. For source coding we used the RVLC-2 and VLEC-3 schemes of Table 3.1. In all simulations, $L = 1000$ source symbols were encoded and transmitted in each frame.

First, the EXIT charts of the system using the RVLC-2 of Table 3.1 and precoder $(1 + \mathrm{D})$ as well as $(1 + \mathrm{D}^2)$ are depicted in Fig. 3.36(a) and Fig. 3.36(b), respectively. As seen from Fig. 3.36(a), the transfer function of the inner channel equaliser and that of the outer source decoder intersect before reaching the $(I_A = 1, I_E = 1)$ point. The coordinates of the intersection point increase upon increasing the channel SNR, until the $(I_A = 1, I_E = 1)$ point is reached. Hence we can predict that for the system using the precoder $(1 + \mathrm{D})$, the iteration gains will saturate after a certain number of iterations and the achievable error rate is limited, which is similar to the case of non-precoded channels. By contrast, for the case of the precoder $(1 + \mathrm{D}^2)$, different observations may be made. As seen from Fig. 3.36(b), $E_b/N_0 = 5.2$ dB is the system's convergence threshold. Above this E_b/N_0 value, the EXIT functions of the inner channel equaliser and the outer source decoder intersect only at the $(I_A = 1, I_E = 1)$ point. Hence it is possible to achieve error-free detection provided that the interleaver length and the number of iterations are sufficiently high. Also shown in Fig. 3.36 are the simulated decoding trajectories, which follow the EXIT chart predictions quite well. Further confirmations accrue from the simulated SER performances shown in Fig. 3.37. For the system using the precoder $(1 + \mathrm{D})$, the achievable SER is bounded by the performance when transmitting over an AWGN channel, as seen from Fig. 3.37(a). By contrast, for the system using the precoder $(1 + \mathrm{D}^2)$, the SER can in fact outperform the bound and attain a lower SER value at high channel SNRs, as seen from Fig. 3.37(b), although it performs slightly worse in the low SNR region in comparison to the system using the precoder $(1 + \mathrm{D})$ due to the lower extrinsic output of the equaliser previously observed in Fig. 3.36. Note that although the unity-rate precoder does not introduce any redundancy, a serially concatenated turbo-like code is formed when it is combined with the source VLC. Therefore the resultant scheme is capable of outperforming the AWGN bound, which is the lower SER bound

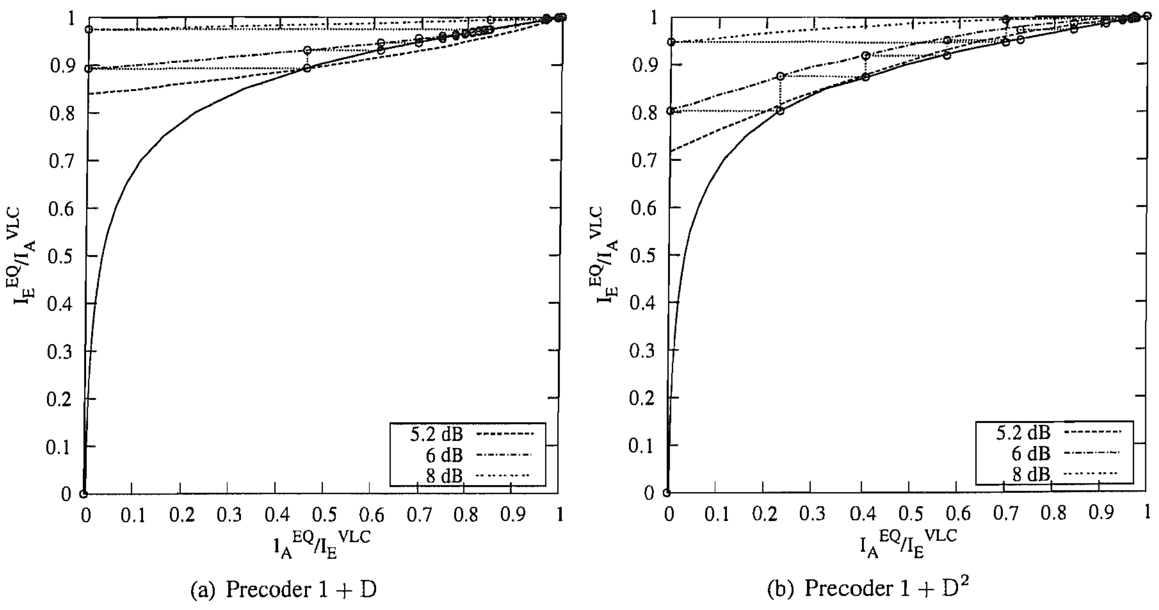


Figure 3.36: EXIT charts of the channel equaliser and the VLC decoder using the RVLC-2 scheme of Table 3.1 over the precoded 3-path channel of Eq. (3.52)

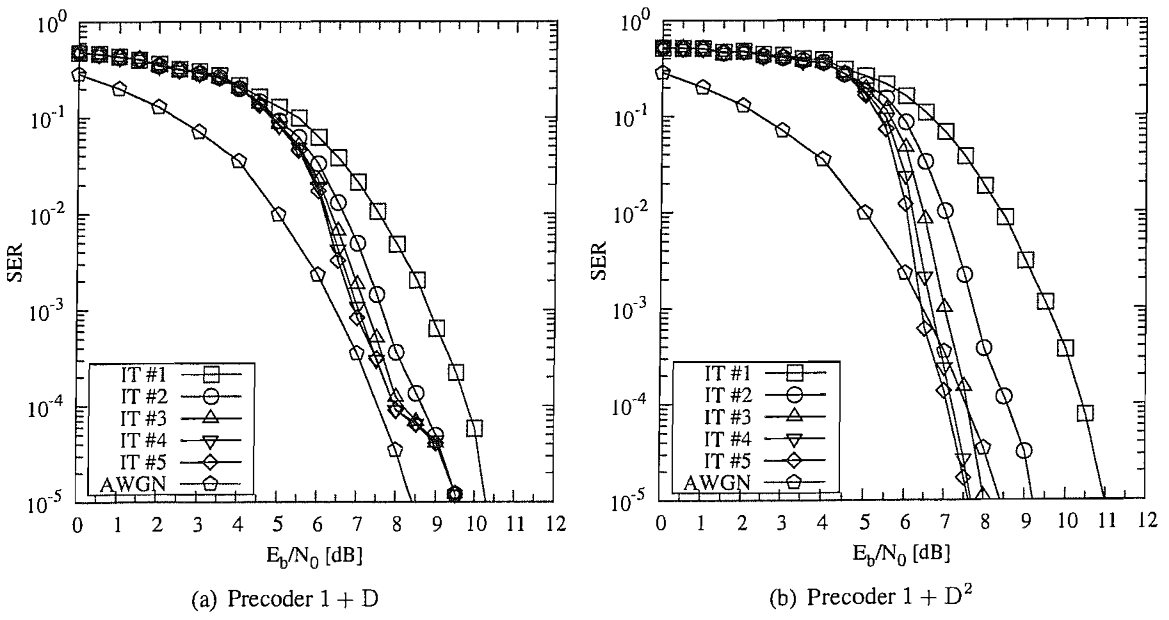
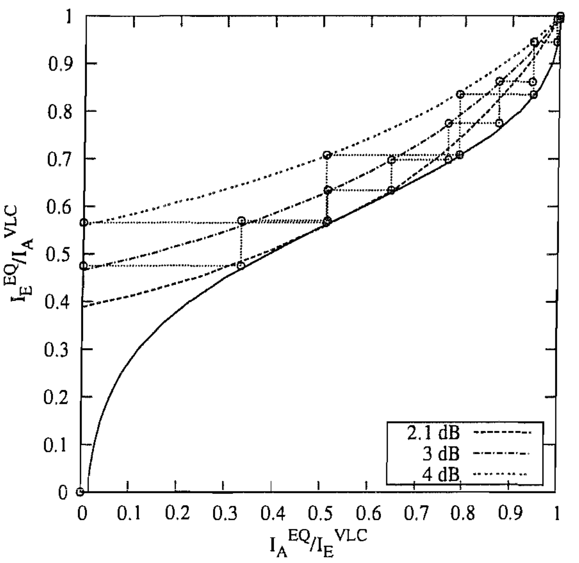
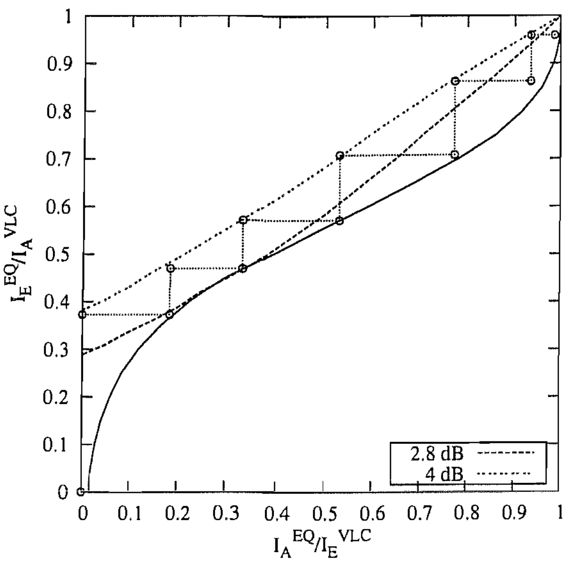


Figure 3.37: SER performance of the iterative receiver using the RVLC-2 scheme of Table 3.1 for transmission over the precoded dispersive AWGN channel of Eq. (3.52)

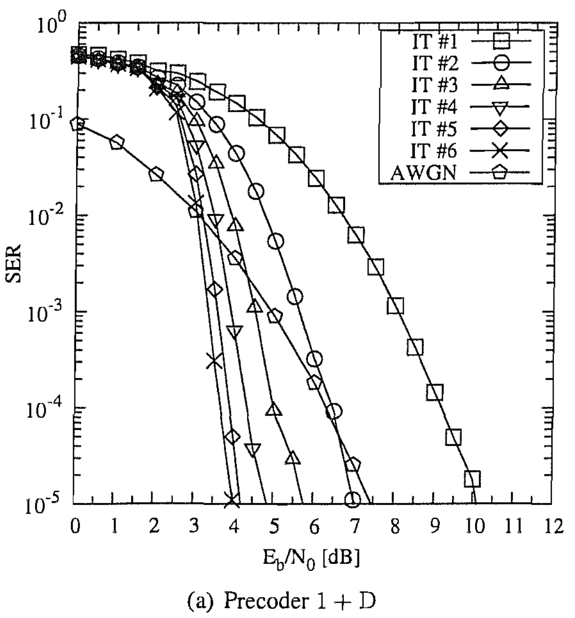


(a) Precoder 1 + D

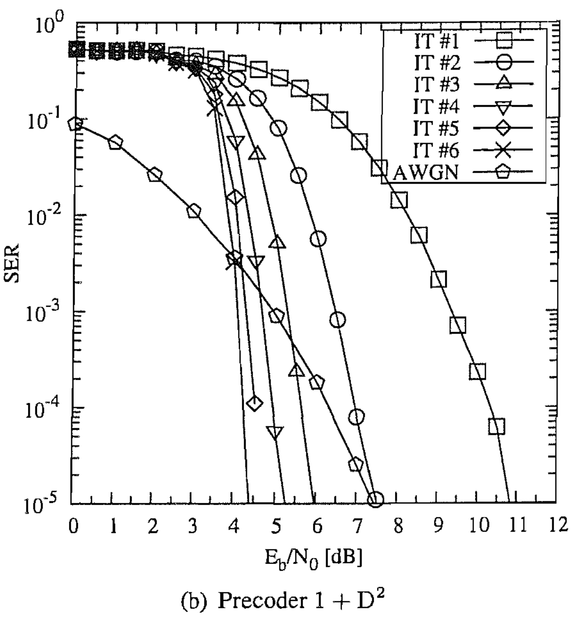


(b) Precoder 1 + D^2

Figure 3.38: EXIT charts of the channel equaliser and the VLC decoder using the VLEC-3 scheme of Table 3.1 for transmission over the precoded 3-path channel of Eq. (3.52)



(a) Precoder 1 + D



(b) Precoder 1 + D^2

Figure 3.39: SER performance of the iterative receiver using the VLEC-3 scheme of Table 3.1 for transmission over the precoded 3-path channel of Eq. (3.52)

for the scheme using VLC encoding without any precoding.

Similarly, we analyse the achievable performances of the various systems using VLEC-3 of Table 3.1. The corresponding EXIT charts are shown in Fig. 3.38. Both systems using the precoder $(1 + \mathrm{D})$ and the precoder $(1 + \mathrm{D}^2)$ show meritorious EXIT characteristics, where an open tunnel can be obtained between the two curves until the $(I_A = 1, I_E = 1)$ point is reached. Furthermore, the system using the precoder $(1 + \mathrm{D})$ has a lower convergence threshold of $E_b/N_0 = 2.1$ dB than that using the precoder $(1 + \mathrm{D}^2)$, which attains convergence at $E_b/N_0 = 2.8$ dB. The corresponding SER performances are shown in Fig. 3.39. Both systems can outperform non-precoded systems performance bound. Particularly, the system using the precoder $(1 + \mathrm{D})$ benefits from earlier convergence and attains a SER of 10^{-4} at $E_b/N_0 = 3.6$ dB, while the system using the precoder $(1 + \mathrm{D}^2)$ needs $E_b/N_0 = 4.2$ dB to achieve the same SER.

3.4.4 Joint Turbo Equalisation and Source Decoding

In this section, we will investigate the achievable performance of iterative equalisation, when both channel decoding and source decoding are considered, which constitutes a three-stage serially concatenated scheme. Let us first introduce our system model.

3.4.4.1 System Model

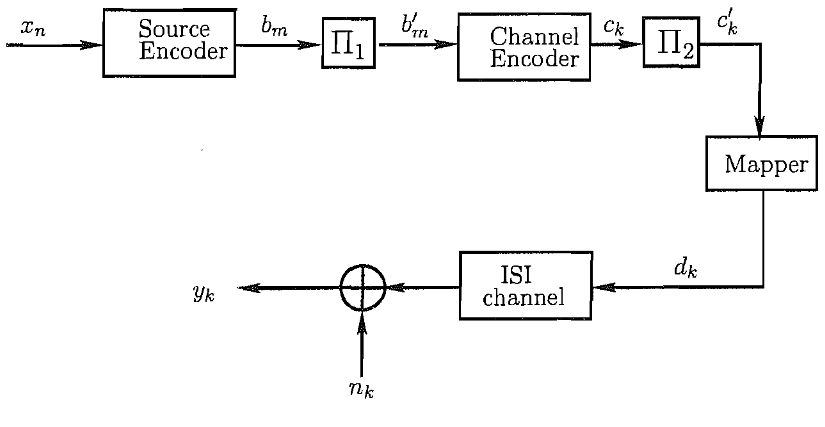


Figure 3.40: Transmitter schematic of the joint source/channel coding scheme for transmission over dispersive channels.

We assume that the source-encoded data is protected by a channel code before its transmission over an ISI-contaminated channel, as shown in Figure 3.40. At the receiver both channel equalisation channel decoding and source decoding are performed iteratively. Generally, the iterative channel equalisation and channel decoding processes carried out in a two-stage structure is referred to as turbo equalisation. However, when additionally using SISO source decoding, the joint channel equalisation, channel decoding and source decoding scheme of Figure 3.41 additionally becomes capable of exploiting the residual redundancy inherent in the source.

More explicitly, the corresponding receiver consists of three SISO modules, the channel equaliser, the channel decoder and the source decoder, as shown in Figure 3.41. All of them employ similar APP decoding algorithms and generate soft output for the benefit of the next processing stage. The channel equalisation and channel decoding form the inner iterations. After a certain number of inner iterations, the source decoder is invoked and the extrinsic information of the source is fed back for the next inner iteration. During our experiments, we found that the scheduling of one outer iteration after two inner iterations offered a good trade-off between the complexity imposed and the achievable performance. At the last outer iteration the source decoder generates the estimate $\{\hat{x}_n\}$ of the source symbol sequence by employing a sequence detection on the basis of the classic Viterbi algorithm using the same bit-level trellis.

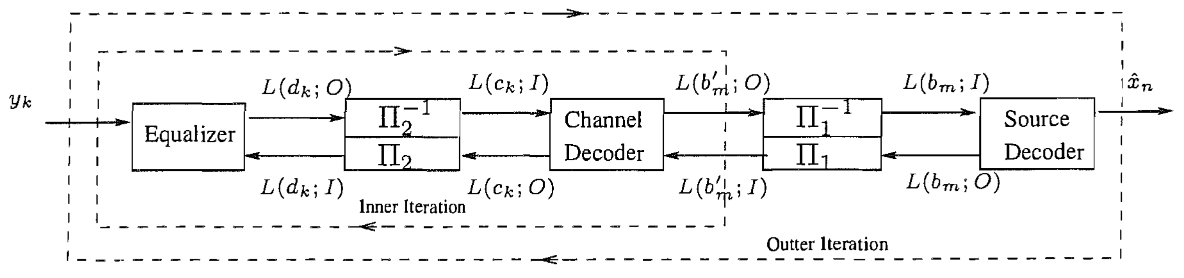


Figure 3.41: Receiver schematic of iterative channel equalisation , channel decoding and source decoding.

3.4.4.2 Simulation Results

The attainable performance of the amalgamated turbo receiver depicted in Figure 3.41 has been evaluated, when communicating over the 3-path ISI channel $\mathbf{H}_1$ of Equation (3.52). The source code used is the RVLC-2 scheme of Table 3.1. Since the error-correction capability of the RVLC-2 code is relatively weak, we use a rather strong channel code here, which is a half-rate, constraint-length $K = 5$, RSC code, using the octally-represented generator polynomial of $G = (035/023)$. The two interleavers are random bit-interleavers having a memory of 2048 bits and 4096 bits, respectively. These system parameters are also listed in Table 3.2.

The corresponding simulation results are depicted in Fig. 3.42. The performances of both the joint and separate source decoding and turbo equalisation schemes are depicted. No outer iterations were executed in the separate source decoding based scheme. The performance of the same system for

Table 3.2: System parameters used in the simulations

Channel Impulse Response (CIR)	$\mathbf{H}_1 = [0.407\ 0.815\ 0.407]^T$
Channel Code	RSC Code $K = 5,\ R = 0.5$ $G = (023, 035)$
Source Code	RVLC-2
Interleaver #1	Random Sequence Interleaver size = 2048 bits
Interleaver #2	Random Sequence Interleaver size = 4096 bits

transmission over the non-dispersive AWGN channel after $I = 6$ iterations is also depicted as the best-case performance "bound". As we can observe in Figure 3.42, the joint three-stage turbo-detection scheme outperforms the separate source decoding based scheme by about 2 dB at SER=10^{-4} after $I = 6$ iterations. The corresponding EXIT chart analysis for this scheme is presented in Section 4.7 after an EXIT-module based convergence analysis technique is introduced in Chapter 4.

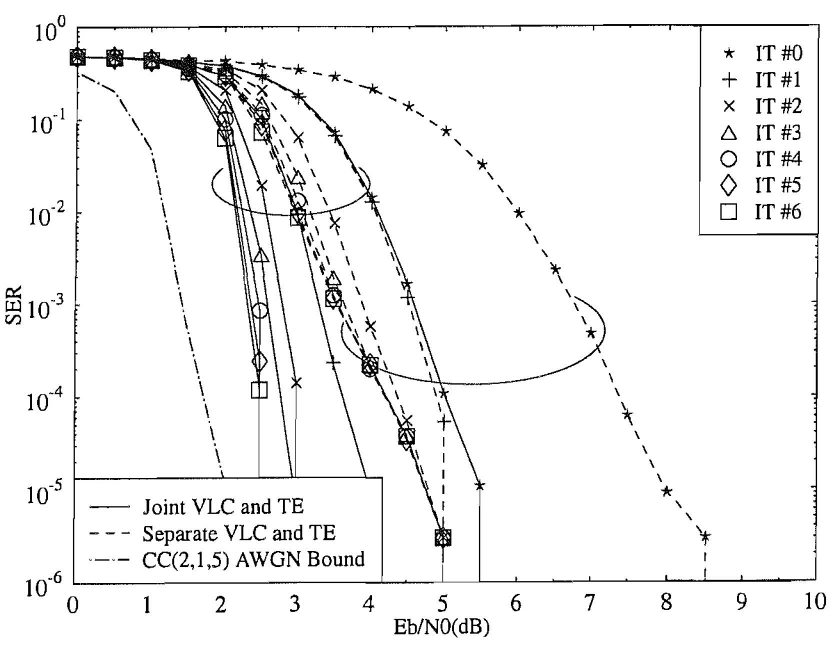


Figure 3.42: SER performance of joint RVLC decoding, channel decoding and channel equalisation over the three-path dispersive AWGN channel of Equation (3.52)

3.5 Summary and Conclusions

In this chapter, the achievable performances of iterative source/channel decoding schemes were investigated for both non-dispersive and dispersive AWGN channels. Since a source decoder combined with a channel decoder or a channel equaliser may be viewed as a serially concatenated system, the design principle of serial concatenated codes [75] applies, where the free distance of the outer code predetermines the achievable interleaver gain.

To be more specific, in the case of non-dispersive AWGN channels our simulation results show that the RVLC-2 of Table 3.1 having $d_f = 2$ outperforms the Huffman code and RVLC-1 having $d_f = 1$, while VLEC-3 having $d_f = 3$ outperforms the RVLC-2 arrangement. When the source codes have the same free distance, such as the Huffman code and RVLC-1 of Table 3.1, the constraint-length determines the achievable performance. The larger the constraint length, the better the attainable performance. Our EXIT chart analysis seen in Fig. 3.33 confirms the simulations results of Fig. 3.37 and Fig. 3.39. It was found that VLCs having $d_f \geq 2$, such as RVLC-2 and VLEC-3, have better convergence properties than their lower-free-distance counterparts..

Unlike convolutional codes, the free distance and the code rate of a source code cannot be adjusted separately. Increasing the free distance usually requires decreasing the code rate. For example, as the free distance increases from 1 to 3, the code rate of the Huffman code, RVLC-1, RVLC-2, and VLEC-3 schemes decreases from 0.98 to 0.54. The increasing of the free distance generally improves the attainable decoding performance. However, decreasing the code rate reduces the system's throughput. For the design of the entire system, we have to find an attractive balance between the code rate and the free distance. Our main results are summarised in Table 3.3.

Table 3.3: Summary of the simulation results for the iterative source/channel decoding scheme investigated in Secion 3.3. Various source codes listed in Table 3.1 are evaluated. The system throughput is measured in bits per channel use (bpc) and the E_b/N_0 gain is based on the SNR value required for achieving a SER of 10^{-4} after six iterations and the benchmarker is the scheme using the Huffman code.

	HUFF	RVLC-1	RVLC-2	VLEC-3
VLC d_f	1	1	2	3
Throughput (bpc)	0.48	0.45	0.44	0.27
E_b/N_0 Gain (dB)	0	0.8	3.1	4.4

Furthermore, in the case of dispersive AWGN channels, the performance of iterative channel equalisation and source decoding is evaluated for transmission over a 3-path channel having mild ISI and a 5-path channel having severe ISI. For both channels, the system using either the RVLC-2 or

VLEC-3 code of Table 3.1 exhibits significant iteration gains. Indeed, the EXIT chart analysis of Fig. 3.30 shows that the EXIT function of a channel equaliser cannot reach the ($I_A = 1$, $I_E = 1$) point, except for high SNRs, which implies that residual errors persist at the channel equaliser's output, regardless how high the interleaver length is and how many iterations are executed. This is further confirmed by the simulations results of Fig. 3.31 and Fig. 3.32, where the performance of the iterative receiver is bounded by the performance of the same system transmitting over an AWGN channel exhibiting no ISI, resulting in "error shoulders" in the SER curves. The system using the VLEC-3 code performs closer to the non-dispersive AWGN bound than that using the RVLC-2 due to the better EXIT characteristics of the VLEC-3 code.

Table 3.4: Summary of the simulation results of the iterative source decoding and channel equalisation scheme for the transmission over the 3-path channel H_1 investigated in Secion 3.4. The RVLC-2 and the VLEC-3 code listed in Table 3.1 are employed, resulting in a system throughput of 0.87 bpc and 0.54 bpc, respectively. The E_b/N_0 value is the SNR value required for achieving a SER of 10^{-4} after five iterations. The iteration gain is measured after five iterations.

	No precoding	Using Precoder $1+D$	Using Precoder $1+D^2$
RVLC-2			
Iteration Gain (dB)	2.0	1.7	3.5
E_b/N_0 (dB)	8.0	8.0	7.0
VLEC-3			
Iteration Gain (dB)	3.0	5.6	6.2
E_b/N_0 (dB)	6.8	3.6	4.2

The effect of precoding on the iterative channel equalisation and source decoding was also investigated. A rate-one precoder renders the channel recursive for the receiver, thus enables the attainment of higher interleaver gains. This effect is more obvious in the EXIT chart analysis of Fig. 3.35, where the EXIT function of a channel equaliser recorded for a precoded channel can reach the ($I_A = 1$, $I_E = 1$) point. It was found in Section 3.4.3 that for different source codes different precoders should be used. For example, when using RVLC-2, the system employing the precoder (1 + D) inflicts an error floor, while the system employing the precoder ($1 + D^2$) does not. By contrast, when using VLEC-3, the system employing the precoder (1 + D) achieves earlier convergence than that employing the precoder ($1 + D^2$), while both systems avoid having error floors. Therefore it is concluded that the source code and the precoder should be designed jointly. Our main findings are summarised in Table 3.4.

Finally, when a channel code is employed at the transmitter, a three-stage iterative receiver is formed consisting of a channel equaliser, a channel decoder and a source decoder. It was found in

Fig. 3.42 that up to 2 dB gain in terms of E_b/N_0 values required for achieving a SER of 10^{-5} can be obtained, when compared to separate turbo equalisation and source decoding.

Chapter 4

Three-Stage Serially Concatenated Turbo Equalisation

4.1 Introduction

Turbo equalisation [86] is an effective means of eliminating the channel-induced ISI imposed on the received signal, hence the achievable performance may approach that recorded over the non-dispersive AWGN channel. When a simple rate-1 precoder is applied before the modulator, which renders the channel to appear recursive to the receiver, the attainable performance may be further improved [109, 169].

The SISO Minimum Mean Square Error (MMSE) equaliser [135], which is capable of utilising *a priori* information from other SISO modules such as a SISO channel decoder and generating *extrinsic* information, forms an attractive design alternative to the MAP equaliser owing to its lower computational complexity. This is particularly so for channels having long CIRs [117, 135]. The precoder can be readily integrated into the shift register model of the ISI channel [169] and may be modelled by combining its trellis with the trellis of a MAP/Soft-Output Viterbi Algorithm (SOVA) based equaliser. However, the precoder's trellis-description cannot be directly combined with the model of an MMSE equaliser. Hence, the achievable performance of MMSE turbo equalisation is potentially limited [117, 135].

EXIT [102] charts have been proposed for analysing the convergence behaviour of iterative decoding schemes, which indicate that an infinitesimally low BER may only be achieved by an iterative receiver, if an open tunnel exists between the EXIT curves of the two SISO components. Recently, both the convergence analysis and the best activation order of the component codes has been studied in the context of multiple-stage concatenated codes [124, 126], which generally require the employment of three-dimensional (3D) EXIT charts. For the sake of simplifying the associated analysis, a 3D to 2D EXIT chart projection technique was proposed in [124, 126]. It has been shown [117] that the EXIT curve of an MMSE equaliser intersects with that of the channel decoder, before reaching the decoding convergence point of (1,1), hence residual errors persist after turbo equalisation. However, it is natural to conjecture that there might exist an open tunnel leading to the convergence point in the 3-D EXIT chart of a well-designed three-stage SISO system.

Against this backdrop, we propose a combined serially concatenated channel coding and MMSE equalisation scheme, which is capable of achieving a precoding-aided convergence-acceleration effect for a MAP/SOVA equaliser. Moreover, the convergence behaviour of the proposed scheme is investigated with the aid of the 3D to 2D EXIT chart projection technique developed in [124, 126], and further design guidelines are derived from an EXIT-chart perspective. For illustration and comparison, let us commence with the family of traditional two-stage turbo equalisation schemes.

4.2 Soft-in/Soft-out MMSE Equalisation

We assume a coherent, symbol-spaced receiver front-end as well as pefect knowledge of the signal phase and symbol timing, where the transmitter filter, the channel and the receiver filter are represented by a discrete-time linear filter, having the Finite-length Impulse Response (FIR)

$$h[n] = \sum_{k=0}^{M-1} h_k \delta[n-k] \tag{4.1}$$

of length M. The coefficients h_k are assumed to be time-invariant and known to the receiver. For simplicity, we assume binary phase shift keying (BPSK) and that the channel impulse response coefficients h_k and the noise samples w_n are real valued. For higher order constellations and complex-valued h_k and w_n, please refer to [135] for the details.

Let $x_n, n = 1, \ldots, K_c, x_n \in \{+1, -1\}$ denote the transmitted symbols and w_n represent the AWGN samples, which are independent and identically distributed (i.i.d.). Given (4.1), the receiver's input z_n is given by

$$z_n = \sum_{k=0}^{M-1} h_k x_{n-k} + w_n. \tag{4.2}$$

Then the MMSE equaliser computes the estimates $\hat{x}_n$ of the transmitted symbols x_n from the received symbols z_n by minimising the cost function $E[|x_n - \hat{x}_n|^2]$.

In contrast to conventional MMSE-based equalisation methods, in the SISO equaliser advocated the MSE is averaged over both the distribution of the noise as well as the distribution of the transmitted symbols. This is because in contrast to classic MMSE-based non-iterative equalisation, in the context of turbo equalisation the symbol distribution is no longer i.i.d., as is typically assumed for , due to the information fed back to the equaliser from the error correction decoder. Let $L_A(x_n) = \ln \frac{P(x_n=+1)}{P(x_n=-1)}$ denote the *a priori* LLR provied by the channel decoder. Then the SISO equaliser's output $L_E(x_n)$ is obtained using the estimate $\hat{x}_n$

$$\begin{aligned} L_E(x_n) &\triangleq \ln \frac{P(x_n = +1|\hat{x}_n)}{P(x_n = -1|\hat{x}_n)} - \ln \frac{P(x_n = +1)}{P(x_n = -1)} \\ &= \ln \frac{p(\hat{x}_n|x_n = +1)}{p(\hat{x}_n|x_n = -1)}, \end{aligned} \tag{4.3}$$

$n = 1, \ldots, K_c$, which requires the knowledge of the distribution $p(\hat{x}_n|x_n = x)$.

In order to perform MMSE estimation, the statistics $\bar{x}_n \triangleq \mathrm{E}[x_n]$ and $v_n \triangleq \mathrm{Cov}(x_n, x_n)$ of the symbols x_n are required. Usually, the symbols x_n are assumed to be equiprobable and i.i.d., which corresponds to $L_A(x_n) = 0, \forall n$, and yields $\bar{x}_n = 0$ as well as $v_n = 1$. For the general case of $L_A(x_n) \in \mathbb{R}$, when the symbols x_n are not equiprobable, $\bar{x}_n$ and v_n are obtained as

$$\begin{aligned} \bar{x}_n &= \sum_{x=\pm 1} x \cdot P(x_n = x) \\ &= \frac{e^{L_A(x_n)}}{1 + e^{L_A(x_n)}} - \frac{1}{1 + e^{L_A(x_n)}} \\ &= \tanh\left(\frac{1}{2} L_A(x_n)\right), \end{aligned} \tag{4.4}$$

$$\begin{aligned} v_n &= \sum_{x=\pm 1} |x - \mathrm{E}(x_n)|^2 \cdot P(x_n = x) \\ &= 1 - |\bar{x}_n|^2. \end{aligned} \tag{4.5}$$

After MMSE estimation, we assume that the pdfs $p(\hat{x}_n|x_n = x)$ are Gaussian, having the parameters of $\mu_{n,x} \triangleq \mathrm{E}[\hat{x}_n|x_n = x]$ and $\sigma^2_{n,x} \triangleq \mathrm{Cov}(\hat{x}_n, \hat{x}_n|x_n = x)$ [94]

$$p(\hat{x}_n|x_n = x) \approx \frac{1}{\sqrt{2\pi}\sigma_{n,x}} \exp\left\{-\frac{(\hat{x}_n - \mu_{n,x})^2}{2\sigma^2_{n,x}}\right\} \tag{4.6}$$

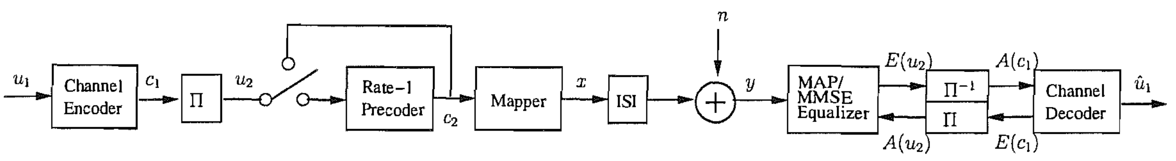


Figure 4.1: Turbo equalisation system using MAP/MMSE equaliser both with and without precoding. The system parameters are summarised in Table 4.1.

and the corresponding output LLRs $L_E(x_n)$ are fomulated as

$$\begin{aligned} L_E(x_n) &= \ln \frac{\frac{1}{\sqrt{2\pi}\sigma_{n,+1}} \exp\left\{-\frac{(\hat{x}_n-\mu_{n,+1})^2}{2\sigma^2_{n,+1}}\right\}}{\frac{1}{\sqrt{2\pi}\sigma_{n,-1}} \exp\left\{-\frac{(\hat{x}_n-\mu_{n,-1})^2}{2\sigma^2_{n,-1}}\right\}} \\ &= \frac{2\hat{x}_n\mu_{n,+1}}{\sigma^2_{n,+1}}. \end{aligned} \tag{4.7}$$

The empoyment of the Gaussian assumption tremendously simplifies the computation of the SISO equaliser output LLRs $L_E(x_n)$. We emphasise that $L_E(x_n)$ should not depend on the particular *a priori* LLR $L_A(x_n)$. Therefore, we require that $\hat{x}_n$ does not depend on $L_A(x_n)$, which affects the derivation of the MMSE equalisation algorithms. For more details we refer to [117, 135].

4.3 Turbo Equalisation Using MAP/MMSE Equalisers

4.3.1 System model

Fig. 4.1 shows the system model of a classic turbo equalisation scheme. At the transmitter, a block of length L information data bits u_1 is encoded by a channel encoder first. After channel coding, the coded bits c_1 are interleaved yielding the data bits u_2 and are either directly fed to the bit-to-modulated-symbol mapper or they are first fed through a rate-1 precoder and encoded for producing the coded bits c_2, as seen in Fig. 4.1. After mapping, the modulated signal x is transmitted over a dispersive channel contaminated by AWGN n. At the receiver of Fig. 4.1, an iterative detection/decoding structure is employed, where extrinsic information is exchanged between the channel equaliser and the channel decoder in a number of consecutive iterations. To be specific, the channel equaliser processes two inputs, namely the received signal y and the *a priori* information $A(u_2)$ fed back by the channel decoder. Then the channel equaliser of Fig. 4.1 generates the *extrinsic* information $E(u_2)$, which is deinterleaved and forwarded as the *a priori* information to the channel decoder. Furthermore, the channel decoder capitalises on the *a priori* information $A(c_1)$ provided by the channel equaliser and generates the extrinsic information $E(c_1)$, which is interleaved and fed back to the channel equaliser as the *a priori* information. Following the last iteration, the estimates $\hat{u}_1$ of the original bits are generated by the channel decoder, as seen in Fig. 4.1.

Table 4.1: System parameters

Channel Encoder	RSC(2,1,3) Generator Polynomials (5/7)
Precoder	Generator Polynomials 1/(1+D)
Modulation	BPSK
CIR	$[0.407\ 0.815\ 0.407]^T$
Block Length	L = 4096 bits

In our forthcoming EXIT chart analysis and Monte Carlo simulations, we assume that the channel is time-invariant and that the CIR is known at the receiver. To be specific, the three-path CIR of [13] described by

$$h[n] = 0.407\delta[n] + 0.815\delta[n-1] + 0.407\delta[n-2] \tag{4.8}$$

is used. We employ a constraint-length 3, half-rate RSC code RSC(2,1,3) having the octally represented generator polynomials of (5/7), where 7 is the feedback polynomial and 5 is the feed forward polynomial. Then we use a simple rate-1 precoder described by the generator polynomials of $1/(1+D)$. Either MAP or MMSE equalisation is invoked, but precoding is only combined with the MAP equaliser. For the sake of simplicity, BPSK modulation is used. Our system parameters are summarised in Table 4.1.

4.3.2 EXIT chart analysis

Fig. 4.2 depicts the EXIT functions of both the MAP/MMSE equalisers and the outer convolutional decoder. It is clear that the EXIT curves of both the MMSE equaliser and the MAP equaliser (without precoding) intersect with that of the outer RSC(2,1,3) decoder, before reaching the convergence point of $(I_A^{EQ} = 1, I_E^{EQ} = 1)$. Hence residual errors may persist, regardless of both the number of iterations used and the size of the interleaver. Furthermore, the MMSE equaliser generally outputs less extrinsic information than the MAP equaliser, resulting in a poorer performance. On the other hand, with the advent of precoding, the EXIT curve of the MAP equaliser becomes capable of reaching the convergence point, as seen in Fig. 4.2. We note however that there is a crossover between the EXIT curves of the precoded and non-precoded MAP equaliser, which implies that the non-precoded MAP equaliser would perform better in the low-SNR region, while the precoded MAP equaliser is capable of achieving an near error-free performance, provided that a sufficiently high number of iterations is

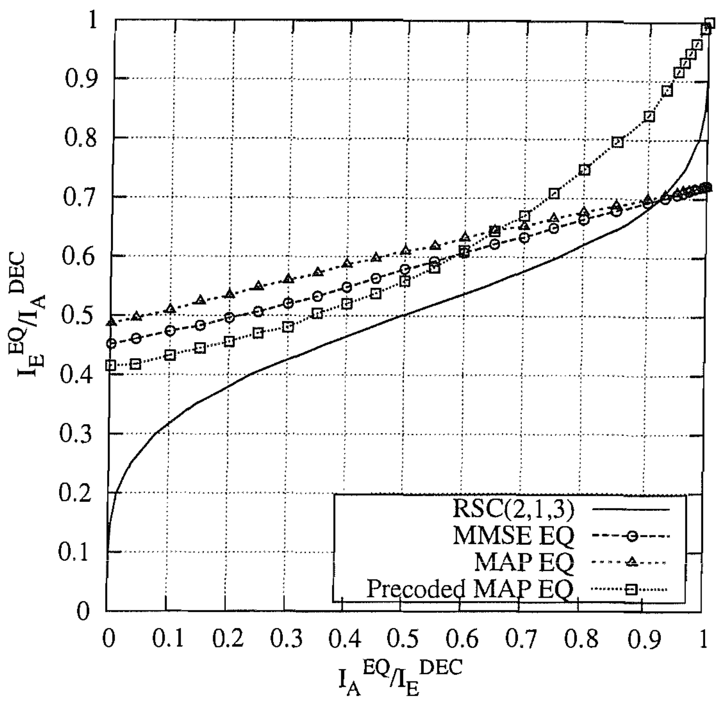


Figure 4.2: EXIT charts for the iterative receiver using either an MMSE equaliser or a MAP equaliser, where the latter is investigated for both a non-precoded and a precoded channel at $E_b/N_0 = 3$ dB

performed. The convergence threshold of the precoded MAP equaliser is about 2.3 dB.

4.3.3 Simulation results

In order to verify the convergence prediction of the EXIT chart analysis outlined in Section 4.3.2, Monte Carlo simulations were also performed and the corresponding BER results are depicted in Fig. 4.3. It can be seen that the BER performance of the precoded system becomes better than that of the non-precoded system at an E_b/N_0 of about 2.7 dB.

4.4 Three-stage serially concatenated coding and MMSE equalisation

4.4.1 System model

Simply incorporating an interleaver between the precoder and the signal mapper in the transmitter of Fig. 4.1 enables the receiver to perform iterative equalisation/decoding by exchanging extrinsic information between three SISO modules, namely the MMSE equaliser, Decoder II and Decoder I of Fig. 4.4. Here, we would prefer not to refer to Encoder II as a precoder, since it cannot be directly combined with the equaliser at the receiver as in Fig. 4.1. The same three-path channel of Eq. (4.8) is used as in Section 4.3. The length of the noncausal and the causal part of the MMSE filter are $N_1 = 5$

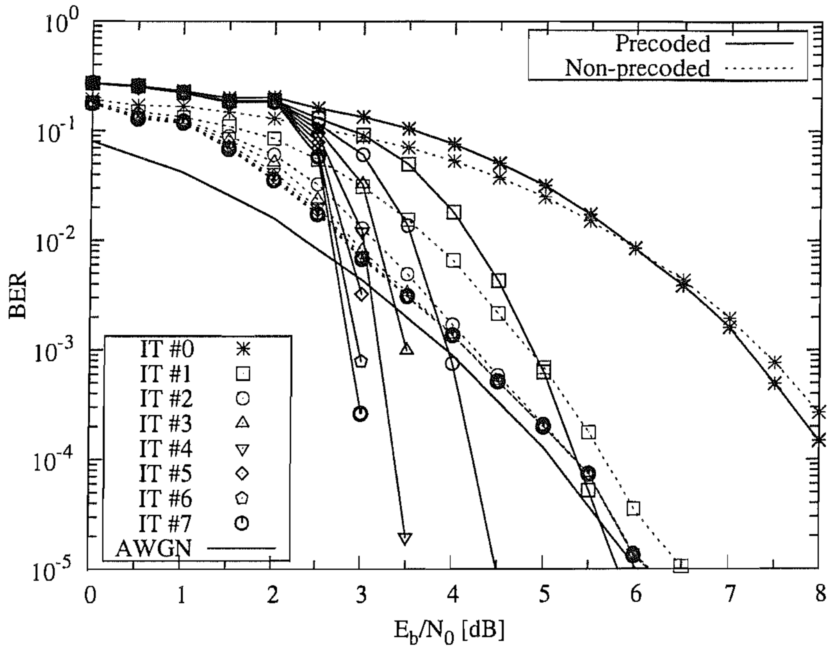


Figure 4.3: BER performance of the iterative receiver using the MAP equaliser both with and without precoding for transmission over the dispersive AWGN channel having a CIR of Eq. (4.8). The system parameters are outlined in Table 4.1.

and $N_2 = 3$, respectively, resulting in an overall filter length of $N = N_1 + N_2 + 1 = 9$.

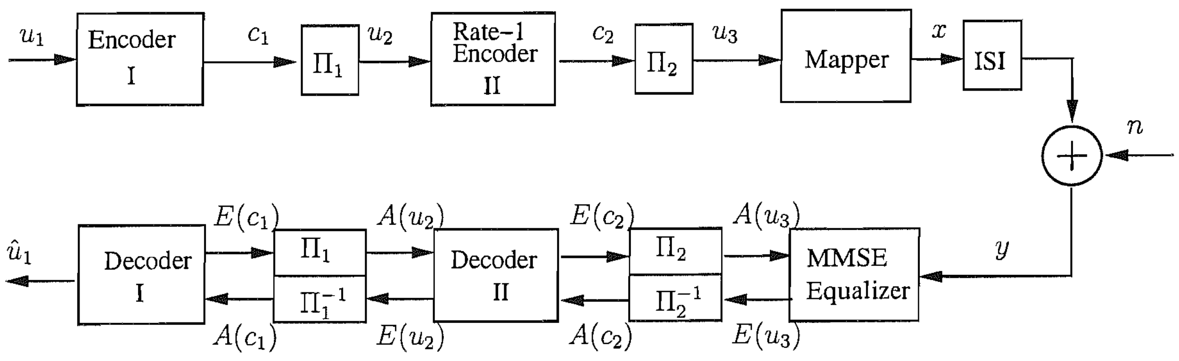


Figure 4.4: System diagram of two serially concatenated codes and MMSE equalisation

4.4.2 EXIT chart analysis

In the following, we will carry out the EXIT chart analysis of the three-stage system of Fig. 4.4. Similar to the two-stage system, the convergence SNR threshold of the three-stage system can be determined. At the same time, the outer code is optimised to give the lowest convergence SNR threshold. Finally, the activation order of the three SISO modules is optimised.

4.4.2.1 Determination of the Convergence Threshold

As seen in Fig. 4.4, Decoder II exploits two a *priori* inputs, namely, $A(c_2)$ and $A(u_2)$. At the same time, it generates two *extrinsic* outputs, i.e., $E(c_2)$ and $E(u_2)$. Hence, in order to describe the EXIT characteristics of Decoder II, we need the following two 2D EXIT functions [102, 126]:

$$I_{E(u_2)} = T_{u_2}[I_{A(u_2)}, I_{A(c_2)}], \tag{4.9}$$

$$I_{E(c_2)} = T_{c_2}[I_{A(u_2)}, I_{A(c_2)}]. \tag{4.10}$$

By contrast, for the MMSE equaliser and Decoder I, only one a *priori* input is available in Fig. 4.4 and the corresponding EXIT functions are:

$$I_{E(u_3)} = T_{u_3}[I_{A(u_3)}, E_b/N_0] \tag{4.11}$$

for the equaliser and

$$I_{E(c_1)} = T_{c_1}[I_{A(c_1)}] \tag{4.12}$$

for Decoder I, where the second parameter of E_b/N_0 in Eq. (4.11) indicates that the extrinsic information also depends on the channel SNR. Therefore, with the aid of the EXIT module concept as described in Section 3.2.2, the three-stage system of Fig. 4.4 may be viewed as a serial concatenation of three EXIT modules, which is shown in Fig. 4.5.

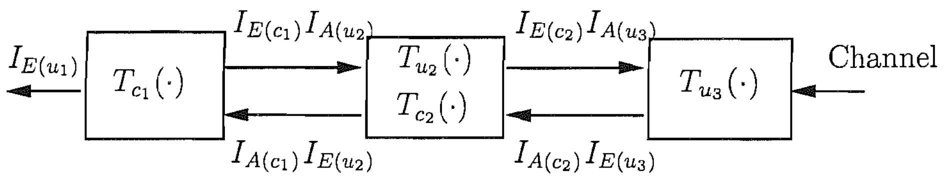
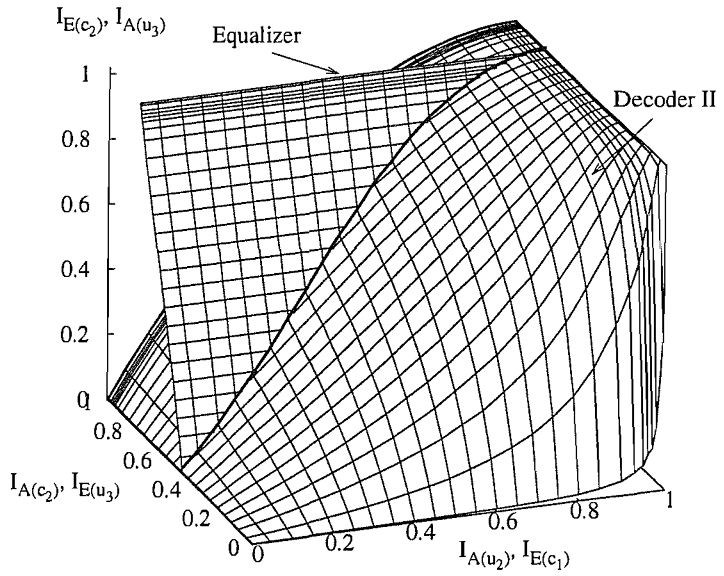


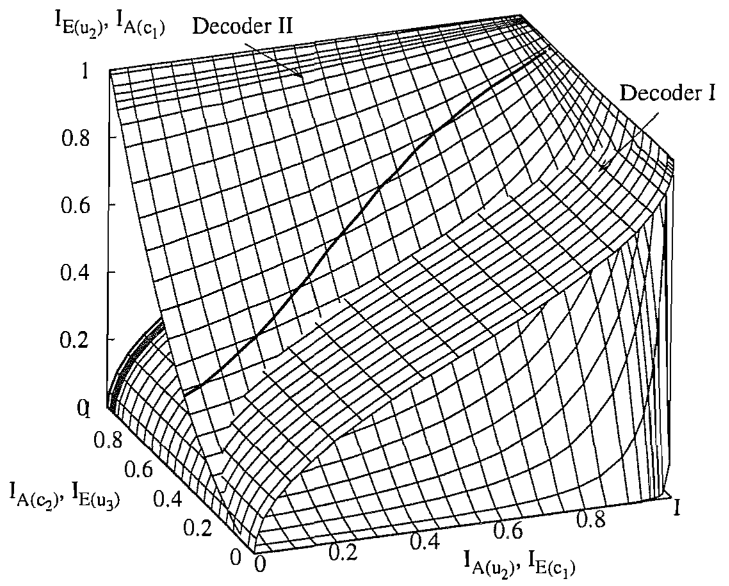
Figure 4.5: Concatenation of the EXIT modules corresponding to the concatenation of the SISO modules in Fig. 4.4

For the sake of plotting all the EXIT functions two 3D EXIT charts are required, namely one for the EXIT functions of both Eq. (4.10) and Eq. (4.11) as shown in Fig. 4.6(a), and another for the EXIT functions of both Eq. (4.9) and Eq. (4.12) as shown in Fig. 4.6(b). Note that $I_{E(u_3)}$ of Eq. (4.11) is independent of $I_{A(u_2)}$, hence the MMSE equaliser's EXIT surface seen in Fig. 4.6(a) is generated by sliding its EXIT curve in the 2D EXIT chart of Fig. 4.2 along the $I_{A(u_2)}$ axis. The EXIT surface of Decoder I was generated similarly, as shown in Fig. 4.6(b), where $I_{E(c_1)}$ of Eq. (4.12) is independent of $I_{A(c_2)}$.

Let us first consider the extrinsic information exchange between the MMSE equaliser and Decoder



(a) Decoder II and the equaliser at $E_b/N_0 = 4$dB



(b) Decoder II and Decoder I

Figure 4.6: 3D EXIT charts for the three-stage SISO system.

II. Let l be the time index. Only one SISO module is invoked each time. Note that we have $I^{(l)}_{A(c_2)} = I^{(l-1)}_{E(u_3)}$ and $I^{(l)}_{A(u_3)} = I^{(l-1)}_{E(c_2)}$. Considering Eq. (4.9), (4.10) and (4.11), for a given a *priori* information $I_{A(u_2)}$, we have

$$I^{(l)}_{A(c_2)} = T_{u3}[T_{c2}[I_{A(u_2)}, I^{(l-2)}_{A(c_2)}], E_b/N_0], \tag{4.13}$$

with $I^{(0)}_{A(c_2)} = 0$ and

$$I^{(l)}_{E(u_2)} = T_{u2}[I_{A(u_2)}, I^{(l)}_{A(c_2)}]. \tag{4.14}$$

The recursive equation of (4.13) actually represents an iteration, including the activation of both Decoder II and the MMSE equaliser. After a sufficiently high number of iterations, $I^{(l)}_{A(c_2)}$ and $I^{(l)}_{E(u_2)}$ will converge to a value between 0 and 1, which depends on the channel SNR and on the a *priori* input $I_{A(u_2)}$ only, i.e., we have

$$I_{A(c_2)} = \lim_{l\to\infty} I^{(l)}_{A(c_2)}, \tag{4.15}$$

$$\begin{aligned} I_{E(u_2)} &= \lim_{l\to\infty} I^{(l)}_{E(u_2)} \\ &= T_{u2}[I_{A(u_2)}, \lim_{l\to\infty} I^{(l)}_{A(c_2)}]. \end{aligned} \tag{4.16}$$

Hence the overall EXIT function of the combined module of the MMSE equaliser and Decoder II is a 1D function of $I_{A(u_2)}$ with E_b/N_0 as a parameter, which can be expressed as

$$I_{E(u_2)} = T'_{u2}[I_{A(u_2)}, E_b/N_0]. \tag{4.17}$$

Fig. 4.7 shows the EXIT function of (4.14) for the combined module of the equaliser and Decoder II. It can be seen that the gain obtained by the second iteration ($l = 4$) is considerable, while the gain obtained by the third iteration is very limited ($l = 6$). Moreover, the extreme values of $I_{A(c_2)}$ in Eq. (4.15), which correspond to different $I_{A(u_2)}$ abscissa values can be visualised as the intersection of the two EXIT surfaces seen in Fig. 4.6(a), which is shown as a thick solid line. Furthermore, the EXIT function of Eq. (4.16) corresponding to the extreme values of $I_{A(c_2)}$ is shown as a solid line in Fig. 4.6(b).

The EXIT function of Eq. (4.17) plotted for the combined module is shown in a 2D EXIT chart in Fig. 4.8. From the 2D EXIT chart of Fig. 4.8, the convergence threshold of the three-stage system can be readily determined. When using a RSC(2,1,3) code having octal generator polynomials of

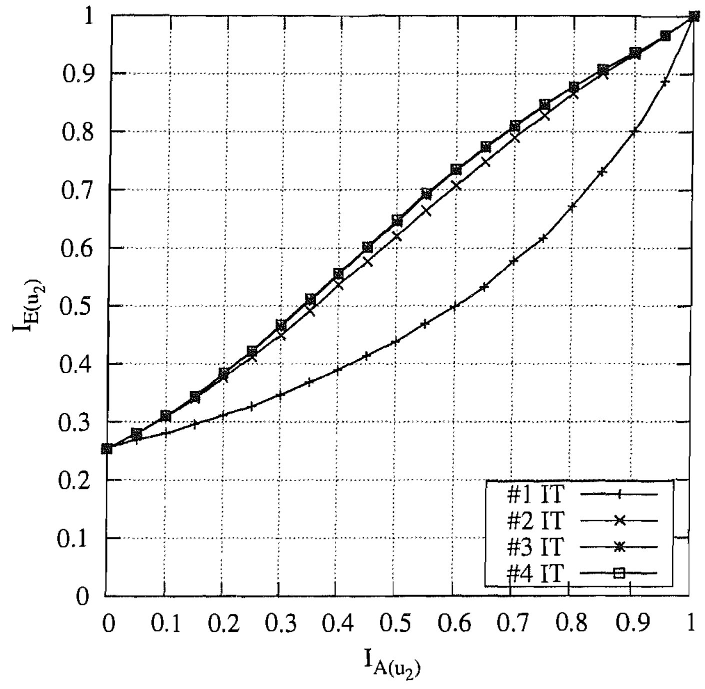


Figure 4.7: EXIT chart for the combined module of the equaliser and Decoder II at $E_b/N_0 = 4$ dB.

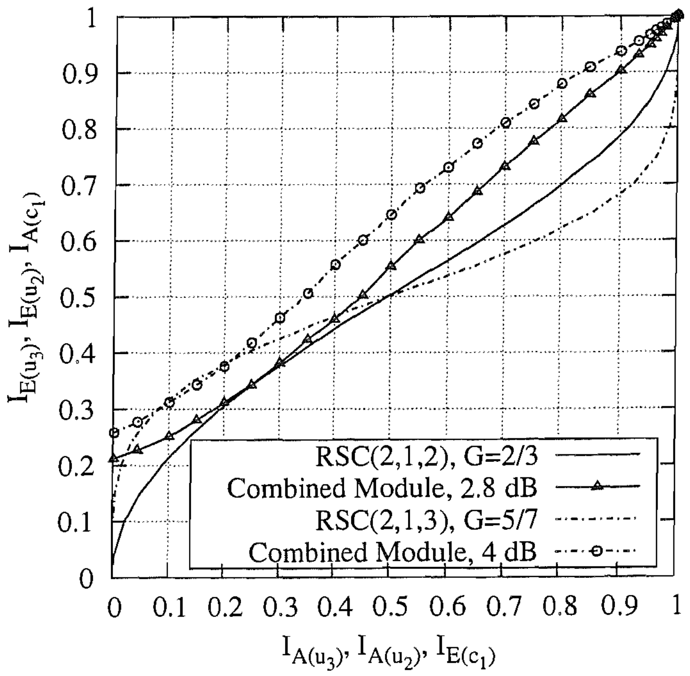


Figure 4.8: 2D EXIT chart for Decoder I and the combined module of the equaliser and Decoder II.

5/7 as the outer code, the EXIT curve of the outer code intersects with the combined EXIT curve of the equaliser and Decoder II at $E_b/N_0 = 4$dB. Hence the convergence threshold of this system is around 4.1 dB. Note that it has been shown in [126] that the convergence point of a multiple-stage concatenated system is independent of the activation order of the component decoders. Hence, the convergence threshold determined in this way is the true convergence threshold, regardless of the activation order.

4.4.2.2 Optimisation of the Outer Code

After obtaining the EXIT function of the combined module of the equaliser and Decoder II, we can optimise the outer code to provide an open tunnel between the EXIT curve of the outer code and that of the combined module at the lowest possible SNR, and hence approach the channel capacity. We carried out a code search for different generator polynomials having constraint lengths up to $K = 5$. Interestingly, we found that the relatively weak code RSC(2,1,2) having generator polynomials of 2/3 yields the lowest convergence threshold of about 2.8 dB. The EXIT function of this code and that of the combined module are also shown in Fig. 4.8 at $E_b/N_0 = 2.8$dB.

4.4.2.3 Optimisation of the Activation Order

Unlike in the two-stage system of Section 4.3, the activation order of the decoders in the three-stage system is an important issue. Although different activation orders will result in the same final convergence point [126], they incur different decoding complexities and delays. A trellis-based search algorithm was proposed in [126] in order to find the optimal activation order of multiple concatenated codes according to certain criteria, such as for example minimising the decoding complexity. However, for the relatively simple case of the three-stage system, we resorted to a heuristic search by invoking the EXIT functions of the three SISO modules according to different activation orders. In these investigations no actual decoding/detection is invoked, hence this design procedure is of low complexity.

In our initial investigations the associated decoding complexity is not considered. Our sole target is that of determining the minimum number of decoder activations required for reaching the convergence point of $I_{E(u_1)} \approx 1$. We tested a number of different activation orders at several SNRs. For example, the natural decoder activation order is [3 2 1 3 2 1], where the integers represent the Index (I) of the various SISO decoder modules. Specifically, $I = 3$ denotes the MMSE equaliser, $I = 2$ represents Decoder II and $I = 1$ corresponds to Decoder I. Hence, according to this activation order, the MMSE equaliser is invoked first, then Decoder II, then Decoder I, and this pattern is repeated for a number of iterations.

Table 4.2: Convergence test results (A/I) for different activation orders. The first number A denotes the number of activations required for reaching the convergence point of $I_{E(u_1)} \approx 0.9999$. The second number I is the corresponding number of iterations. Bold numbers highlight the minimum number of activations/iterations by using different activation orders for each E_b/N_0 value.

Activation Order	2.8 dB	3 dB	3.5 dB	4 dB	5 dB
[3,2,1]	108/36	66/22	39/13	30/10	21/7
[3,2,3,2,1]	130/26	80/16	50/10	40/8	25/5
[3,2,3,2,3,2,1]	175/25	112/16	70/10	49/7	35/5
[3,2,1,2]	101/25	61/15	37/9	29/7	19/5
[3,2,1,2,1]	100/20	**60**/12	35/7	28/6	20/4
[3,2,1,2,1,2]	**99**/17	61/**10**	**35/6**	**27**/5	**19/3**
[3,2,1,2,1,2,1]	108/**16**	66/10	38/6	28/**4**	19/3
[3,2,3,2,1,2,1]	112/16	70/10	42/6	33/5	26/4

Additionally, we can increase the number of the iterations either between the equaliser and Decoder II or between Decoder II and Decoder I. The former scenario includes the activation orders of [3,2,3,2,1] and [3,2,3,2,3,2,1]. By constrast, the activation orders of [3,2,1,2], [3,2,1,2,1], [3,2,1,2,1,2] and [3,2,1,2,1,2,1] correspond to the latter scenario. Finally, the activation order of [3,2,3,2,1,2,1] increases both. The associated convergence test results are listed in Table 4.2. Note that for those activation orders, which do not invoke Decoder I in the end, an additional activation of Decoder I is arranged at the end of the final iteration so that $I_{E(u_1)}$ is updated.

We can observe in Table 4.2 that upon increasing the number of iterations between the equaliser and Decoder II, the total number of activations required for convergence is increased. For example, at $E_b/N_0 = 3$ dB, the number of activations needed for the natural activation order of [3,2,1] to fully converge is $A = 66$, while that necessitated for the activation order of [3,2,3,2,1] is $A = 80$, and $A = 112$ for the activation order of [3,2,3,2,3,2,1]. By contrast, when increasing the number of iterations between Decoder II and Decoder I, the number of activations required for full convergence is decreased. For example, again at $E_b/N_0 = 3$ dB, the numbers of activations needed for the activation orders of [3,2,1,2], [3,2,1,2,1] and [3,2,1,2,1,2] to reach convergence are $A = 61, 60$ and 61, respectively. Further increasing the number of iterations between Decoder II and Decoder I, as in the scenario of [3,2,1,2,1,2,1] will also increase the total number of activations imposed. Among the three best activation orders, [3,2,1,2,1,2] invokes the equaliser the lowest number of times, namely $A_{EQ} = 10$ times. Note that the MMSE equaliser is of the highest computational complexity among the three SISO modules, hence the less frequently the MMSE equaliser is activated, the lower the total decoding complexity. On the whole, the activation order of [3,2,1,2,1,2] constitutes an attractive choice in terms of both fast convergence and low decoding complexity.

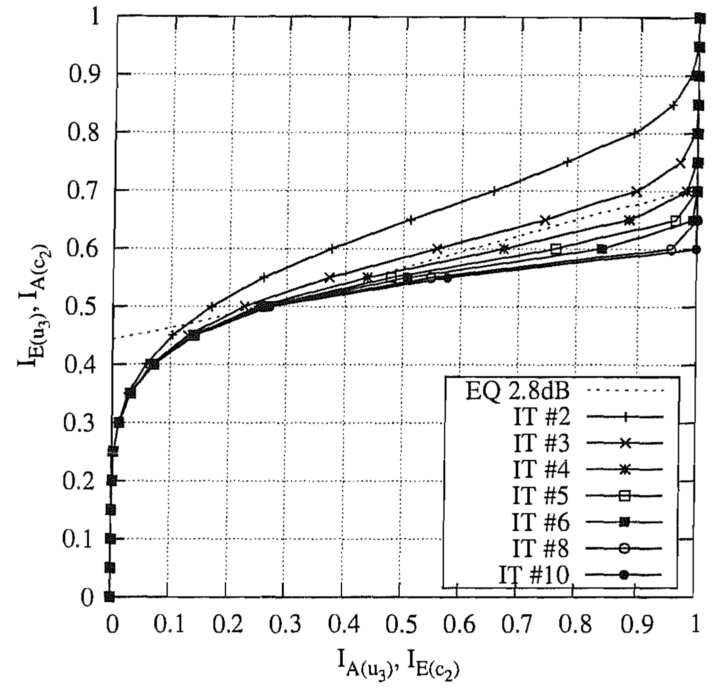


Figure 4.9: EXIT chart for the combined module of Decoder II and Decoder I.

In general, we conclude that by invoking more iterations between Decoder II and Decoder I, while activating the MMSE equaliser from time to time, the three-stage system converges at a lower total number of activations. This is not unexpected, since the error-correction capability is mainly provided by the serial concatenation of Decoder II and Decoder I. Fig. 4.9 shows the EXIT function recorded for the combined module of Decoder II and Decoder I. It can be seen that significant gains can be obtained by iteratively activating Decoder II and Decoder I for up to 10 iterations, especially in the medium a *priori* input information range of $0.5 < I_{A(c_2)} < 0.75$. By contrast, the EXIT function remains similar after three iterations, when iterating between the equaliser and Decoder II, as seen from Fig. 4.7. Hence no further iterations are necessary.

4.4.3 BER performance

In our BER investigations, the RSC(2,1,2) code having the octal generator polynomials of (2/3) was invoked in Encoder/Decoder I. For the rate-1 Encoder II, again, the generator polynomial of 1/(1+D) was used. Fig. 4.10 shows a more specific manifestation of the generic transmitter schematic seen in Fig. 4.4. The block length is $L = 10^5$ bits. The activation order of the three SISO modules is [3 2 1

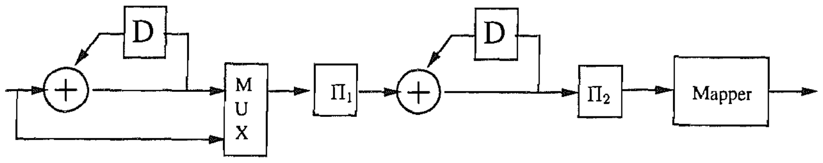


Figure 4.10: A specific manifestation of the generic transmitter schematic seen in Fig. 4.4

2 1 2] and this pattern is repeated twelve times.

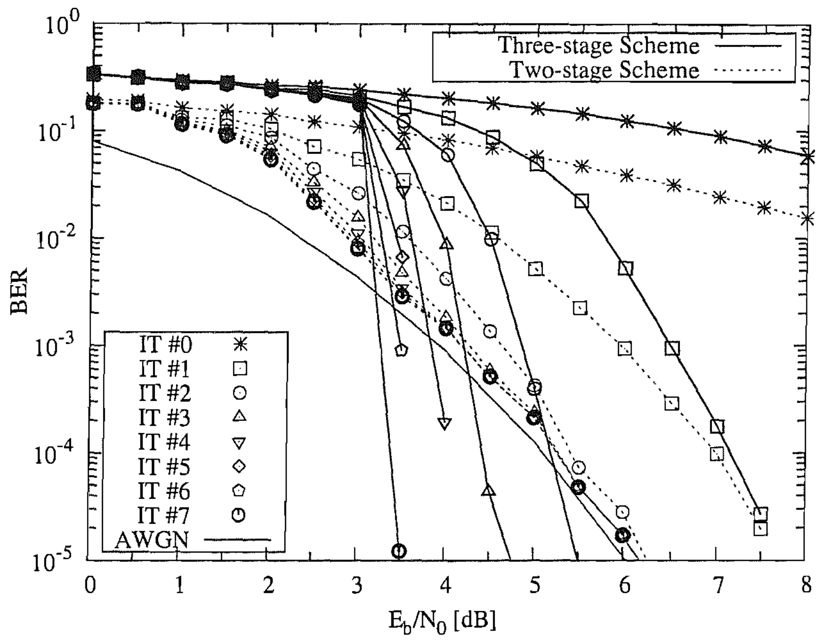


Figure 4.11: BER performance of the three-stage concatenation scheme of Fig. 4.10 and the two-stage concatenation scheme of Fig. 4.1 using an MMSE equaliser for transmission over the dispersive AWGN channel having a CIR of Eq. (4.8). Note that for the three-stage concatenation scheme we used a specially selected activation order of the three SISO modules, namely one iteration is equivalent to an activation period of [3 2 1 2 1 2], where the numbers denote the indices of the SISO modules.

Our BER results are depicted in Fig. 4.11. In addition to the three-stage SISO system, the BER performance of the two-stage SISO system seen in Fig. 4.1 and using an MMSE equaliser is also shown. Observe in Fig. 4.11 that the two-stage SISO system performs better in the low-SNR region, while the three-stage SISO system has the edge in the medium to high SNR region, achieving an infinitesimally low BER for E_b/N_0 values above 3 dB and hence closely matching the convergence threshold of $E_b/N_0 = 2.8$ dB predicted by the EXIT chart analysis of Fig. 4.8. Although the two-stage scheme is of lower decoding complexity, it cannot outperform the AWGN BER bound, regardless of the number of iterations. To achieve a near-zero BER in the medium SNR range, the three-stage scheme has to be used. Additionally, as seen from Fig. 4.11, the actual number of iterations required for achieving near-zero BER at $E_b/N_0 = 3$ dB, 3.5 dB, 4 dB and 5 dB are $I = 12, 7, 5$ and 4, respectively, which match the predictions of $I = 10, 6, 5$ and 3 seen in Table 4.2 quite well.

4.4.4 Decoding trajectories

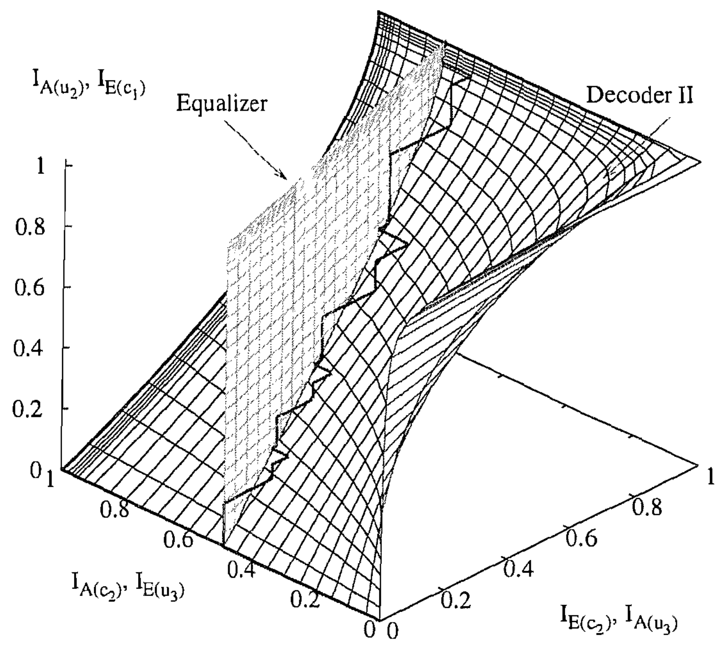


Figure 4.12: The decoding trajectories recorded at Decoder II's output $I_{E(c_2)}$, the equaliser's output $I_{E(u_3)}$ and Decoder I's output $I_{E(c_1)}$ for $E_b/N_0 = 4$ dB. The EXIT function of the equaliser and that of Decoder II are also shown. Each time a SISO module is invoked, the decoding trajectory evolves in the dimension associated with the corresponding *extrinsic* information output.

The mutual information vector of $[I_{E(u_3)}, I_{E(c_2)}, I_{E(u_2)}, I_{E(c_1)}]$ recorded during the simulations may be used to describe the evolution of the system status in the course of iterative decoding. They can be visualised in two 3D EXIT charts, resulting in the so-called decoding trajectories. More explicitly, Fig. 4.12 shows the decoding trajectory associated with the evolution of $I_{E(c_1)}, I_{E(u_3)}$ and $I_{E(c_2)}$. Similarly Fig. 4.13 depicts the decoding trajectory associated with the evolution of $I_{E(c_1)}, I_{E(u_3)}$ and $I_{E(u_2)}$. Note that each graph depicts one and only one *extrinsic* output of each SISO module. Hence, each time a SISO module is invoked, the decoding trajectory evolves in one dimension. For example, in Fig. 4.12, the activation of the equaliser allows the trajectory to evolve along the $I_{A(c_2)}, I_{E(u_3)}$ axis, while the activation of Decoder II evolves the trajectory along the $I_{E(c_2)}, I_{A(u_3)}$ axis. Finally the activation of Decoder I triggers the evolution of the trajectory along the $I_{A(u_2)}, I_{E(c_1)}$ axis.

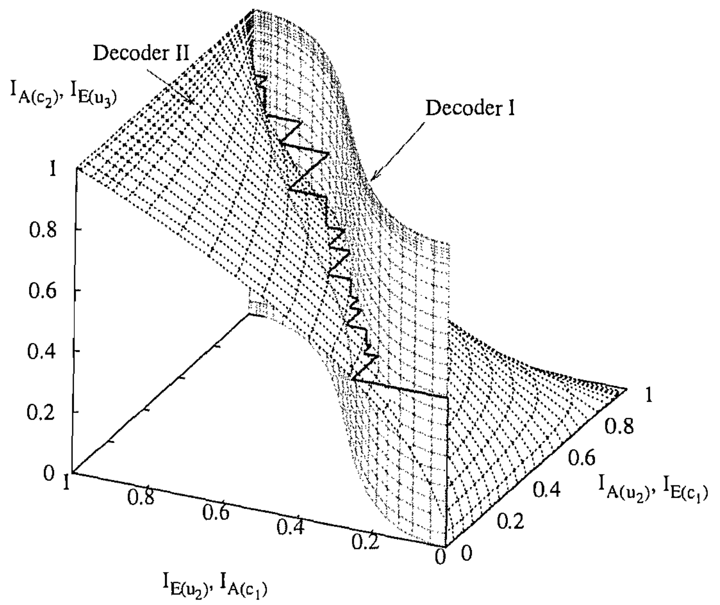


Figure 4.13: The decoding trajectories recorded at Decoder I's output $I_{E(c_1)}$, Decoder II's output $I_{E(u_2)}$ and the equaliser's output $I_{E(u_3)}$ for $E_b/N_0 = 4$ dB. The EXIT function of Decoder II and that of Decoder I are also shown. Each time a SISO module is invoked, the decoding trajectory evolves in the dimension associated with the corresponding *extrinsic* information output.

Alternatively, the decoding process can also be visualised using two 2D graphs, namely, one trajectory associated with the *extrinsic* outputs $I_{E(u_2)}$ and $I_{E(c_1)}$ of Fig. 4.4, while the other associated with the *extrinsic* outputs $I_{E(u_3)}$ and $I_{E(c_2)}$, which characterise the constituent decoder pairs of Decoder I and Decoder II as well as Decoder II and the equaliser of Fig. Fig. 4.4, respectively. The former is depicted in Fig. 4.14, which again characterises the exchange of mutual information between Decoder I and the combined module of the equaliser and Decoder II. The vertical segments of the trajectory represent the activation of the combined equaliser and Decoder II module exchanging extrinsic information a certain number of times, while the horizontal segments represent a single activation of Decoder I. For example, in the simulations, we used an activation order of [3 2 1 2 1 2]. At the beginning of iterations, the equaliser was activated, but given the architecture of Fig. Fig. 4.4 and the activation regime assumed, neither the value of $I_{E(u_2)}$ nor the value of $I_{A(c_1)}$ was changed, hence the trajectory stayed at the point A0 in Fig. 4.14. Then Decoder II was activated, resulting in an increased value of $I_{E(u_2)}$, hence the trajectory reaches the point A1. Subsequently, Decoder I was activated, which increased the value of $I_{E(c_1)}$, hence the trajectory traverses to the point A2. Similarly, the segment between A2 and A3 represents the activation of Decoder II, the segment between A3 and A4 denotes the activation of Decoder I, and the segment between A4 and A5 corresponds to the activation of Decoder II. The segment between A5 and A6 represents the beginning of a new

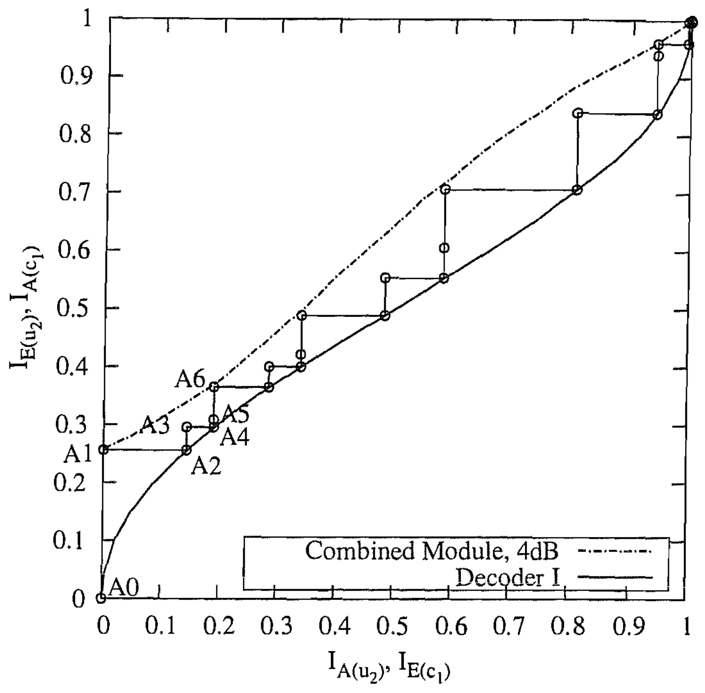


Figure 4.14: The decoding trajectories measured at Decoder II's output $I_{E(u_2)}$ and Decoder I's output $I_{E(c_1)}$ for $E_b/N_0 = 4$ dB. The activations of Decoder II cause the trajectory to grow along the Y-dimension, while the activations of Decoder I cause the trajectory to grow along the X-dimension. The activation of the equaliser is not visualised. More explicitly, the trajectories can be interpreted as [EQ: A0, DEC2: A0→A1, DEC1: A1→A2, DEC2: A2→A3, DEC1: A3→A4, DEC2: A4→A5], where EQ denote the activation of the equaliser and DEC1/DEC2 represent the activation of Decoder I/II, respectively.

combined iteration cycle associated with similar decoding activations.

Fig. 4.15 shows the trajectory associated with the *extrinsic* outputs $I_{E(u_3)}$ and $I_{E(c_2)}$, which characterises the exchange of mutual information between the equaliser and the combined module of Decoder II and Decoder I. The vertical segments of the trajectory represent a single activation of the equaliser, while the horizontal segments represent the activation of Decoder II and Decoder I at a certain number of times. Specifically, the activation schedule of [3 2 1 2 1 2] is visualised as follows. After an initial activation of the equaliser, the trajectory traverses from B0 to B1. Then it remains at B1 after the activation of Decoder II and Decoder I. This is because the output $I_{E(c_2)}$ of Decoder II remained zero after the first activation. However, these two activations are visualised in Fig. 4.14 by the segment traversing from A0 to A1 while representing the activation of Decoder II. Similarly the segment spanning from A1 to A2 denotes the activation of Decoder I. Then the second activation of Decoder II triggers the evolution of the trajectory to the point B2 in Fig. 4.15. The following activation of Decoder I doesnot change the trajectory in Fig. 4.15, but is visualised by the segment

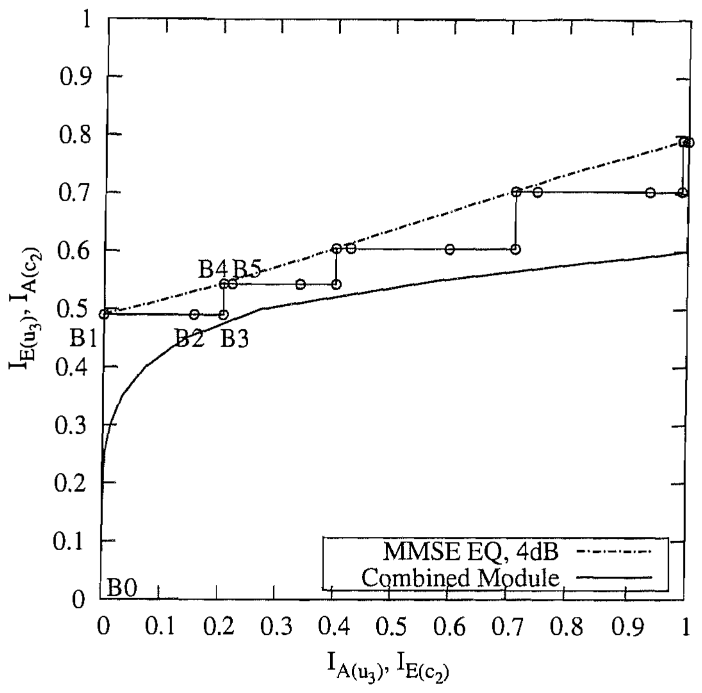


Figure 4.15: The decoding trajectories measured at the equaliser's output $I_{E(u_3)}$ and Decoder II's output $I_{E(c_2)}$ for $E_b/N_0 = 4$ dB. The activations of the equaliser cause the trajectory to grow along the Y-dimension, while the activations of Decoder II cause the trajectory to grow along the X-dimension. The activation of Decoder I is not visualised. More explicitly, the trajectories can be interpreted as [EQ: B0→B1, DEC2: B1, DEC1: B1, DEC2: B1→B2, DEC1: B2, DEC2: B2→B3].

spanning from A3 to A4 in Fig. 4.14. Then the activation of Decoder II triggers a trajectory transition to B3. A new iteration cycle is indicated by the transition from B3 to B4. In conclusion, we surmise that the convergence behaviour of a three-stage SISO system can be adequately visualised using two 2D EXIT charts.

4.4.5 Effects of Interleaver Block Length

As described in Section 3.2.2.3, EXIT functions are usually obtained by recording the *extrinsic* outputs of the SISO decoders, given certain *a priori* inputs. Since the *a priori* LLRs are artificially generated using the Gaussian model, they are independent of each other. However, this assumption is not entirely valid in real systems, where the *a priori* inputs are generated by the interleaved *extrinsic* outputs of the adjacent SISO decoders. The independence assumption only holds for sufficiently large interleaver block lengths and for the first a few iterations. Hence it transpires that the analysis using EXIT functions becomes more and more inaccurate upon increasing the number of iterations and further aggravated, when decreasing the interleaver's block size.

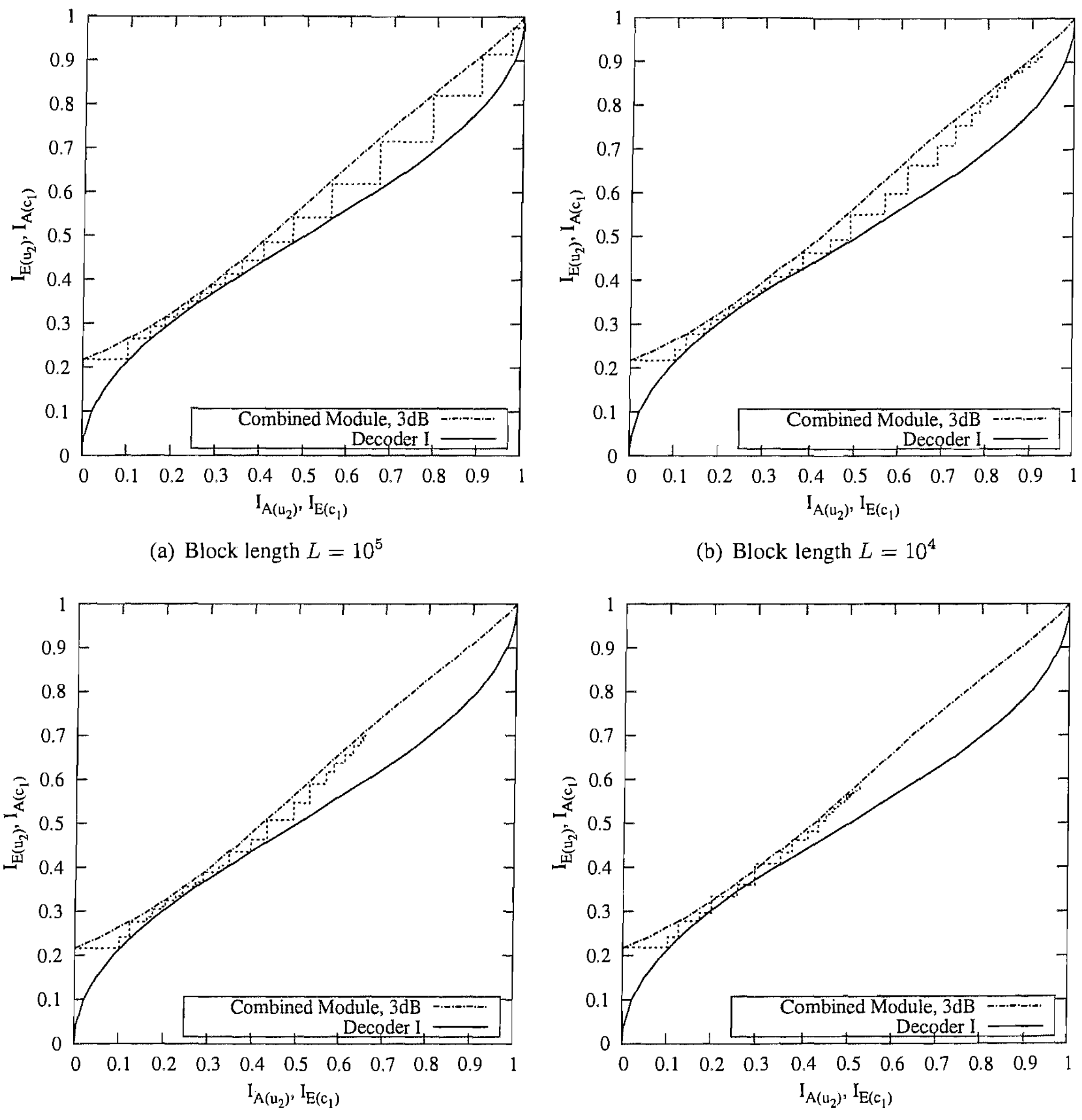
Table 4.3: BER performances for different block lengths after 12 iterations

E_b/N_0	$L = 10^5$	$L = 10^4$	$L = 10^3$	$L = 10^2$
3 dB	2.0×10^{-6}	3.2×10^{-2}	9.4×10^{-2}	1.6×10^{-1}
3.5 dB	0	0	2.0×10^{-2}	1.2×10^{-1}
4 dB	0	0	1.8×10^{-3}	7.2×10^{-2}
4.5 dB	0	0	2.4×10^{-4}	4.8×10^{-2}
5 dB	0	0	0	2.9×10^{-2}

We evaluated the performance of the three-stage system described in Section 4.4.3 for block length $L = 10^5$, 10^4, 10^3 and 10^2 bits using simulations, and the resultant BER performances are listed in Table 4.3. Since the outer code has a rate of $R = 1/2$ and the intermediate code has a unity rate, the length of both of the interleavers seen in Fig. 4.4 is $2L$. According to the EXIT chart analysis in Section 4.4.2.2, the convergence threshold of the three-stage system is 2.8 dB. For a block length of $L = 10^5$ bits, the system achieves convergence at about 3 dB, while the convergence is achieved at around 3.5 dB for $L = 10^4$ bits, at 5 dB for $L = 10^3$ bits and at around 11 dB for $L = 10^2$ bits.

Furthermore, for a given E_b/N_0 value and fixed number of iterations, the BER increases upon decreasing the block length. For example, at $E_b/N_0 = 3$ dB, the BER increases from 2.0×10^{-6} to 1.6×10^{-1} when the block length decreases from 10^5 to 10^2. This effect can also be visualised in the EXIT chart of Fig. 4.16. For instance, the decoding trajectories of the systems using block lengths of $L = 10^5$, 10^4, 10^3 and 10^2 at $E_b/N_0 = 3$ dB are depicted in Fig. 4.16. It can be seen that upon decreasing the block length, the decoding trajectory drifts away from the EXIT curves more and more severely. In other words, the *extrinsic* information outputs of each SISO decoder become less than the EXIT chart predictions, resulting in increasing BERs.

Additionally, we show the achievable coding gain of the three-stage system at BER=10^{-4} in Fig. 4.17. Here, the coding gain is defined as the reduction of the required E_b/N_0, when using the three-stage system as opposed to the uncoded system (i.e. when only MMSE equalisation is employed without channel coding, using a block length of $L = 10^5$ bits). It can be seen that the maximum achievable coding gain decreases upon decreasing the block length. In comparison to the system employing the block length of $L = 10^5$ bits, the maximum coding gain of the system using a block length of $L = 10^4$ bits is decreased by less than 1 dB, while it is decreased by about 2 dB for the system using a block length of $L = 10^3$ bits and around 8 dB for $L = 10^2$ bits. Moreover, for



(a) Block length $L = 10^5$

(b) Block length $L = 10^4$

(c) Block length $L = 10^3$

(d) Block length $L = 10^2$

Figure 4.16: Decoding trajectories of the system of Fig. 4.4 using different block lengths at $E_b/N_0 = 3$ dB.

$L = 10^5$ and $L = 10^4$, less than 0.5 dB iteration gain is achieved after $I = 8$ iterations, while for $L = 10^3$ iteration gain diminishes after $I = 6$ iterations and for $L = 10^2$ only a marginal iteration gain is obtained after $I = 2$ iterations.

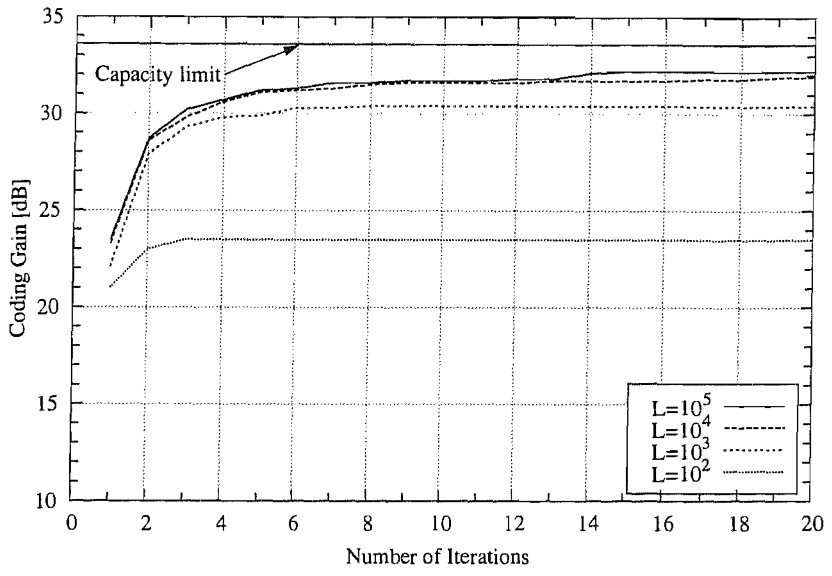


Figure 4.17: Achievable coding gain of the three-stage system of Fig. 4.4 at BER=10^{-4}, when employing various block lengths and different number of iterations.

4.5 Approaching the Channel Capacity Using EXIT-Chart Matching and IRCCs

In Section 4.4.2.2, we optimised the outer code of the three-stage system for achieving the lowest possible convergence threshold. The optimisation was carried out by searching for different generator polynomials for the convolutional codes used. The lowest achievable convergence threshold was about 2.8 dB, when using a RSC(2,1,2) code having octally represented generator polynomials of 2/3. In this section, we will investigate, whether we can reduce this threshold further down using EXIT chart analysis.

For convenience, we first redraw some of the previously used EXIT functions in Fig. 4.18. According to the properties [120] of EXIT charts, the area under the EXIT curve of the inner code is approximately equal to the channel capacity attained, when the channel's input is uniformly distributed. Furthermore, the area under the EXIT curve of the outer code is approximately equal to (1-R_I), where R_I is the outer code's rate. Although these properties were only proven for the family of binary erasure channels (BECs) [120], they have been observed to hold also for AWGN and ISI

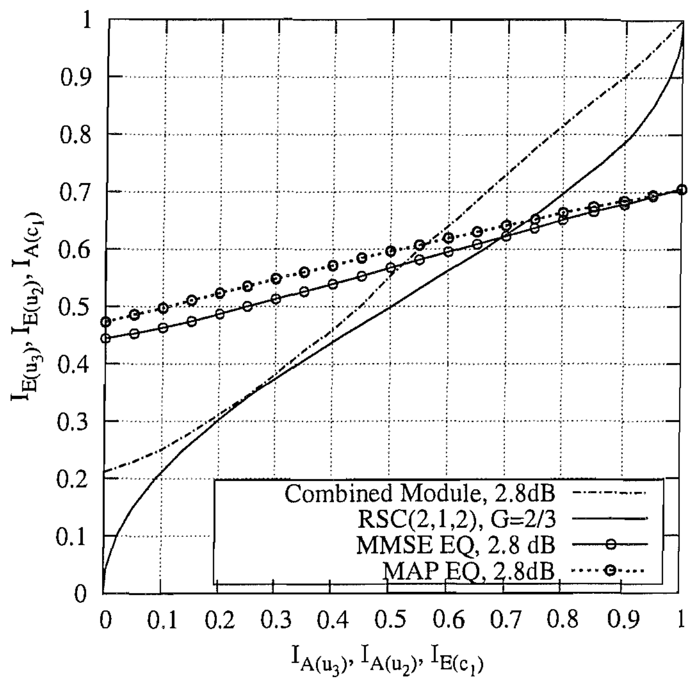


Figure 4.18: EXIT functions of the MMSE equaliser, the MAP equaliser and the combined module of the MMSE equaliser and Decoder II at $E_b/N_0 = 2.8$ dB as well as the EXIT function of the outer RSC(2,1,2) code.

channels [85, 124]. More explicitly, let $\mathcal{A}_I$ and $\bar{\mathcal{A}}_I$ be the areas under the outer code's EXIT curve $T_{c_1}(i)$ and its inverse $T_{c_1}^{-1}(i)$, $i \in [0, 1]$, respectively, which are expressed as:

$$\mathcal{A}_I = \int_0^1 T_{c_1}(i)\mathrm{d}i, \quad \bar{\mathcal{A}}_I = \int_0^1 T_{c_1}^{-1}(i)\mathrm{d}i = 1 - \mathcal{A}_I. \tag{4.18}$$

Similarly, we define the areas $\mathcal{A}_{II}$ and $\bar{\mathcal{A}}_{II}$ for the EXIT curve $T'_{u_2}(i)$ of the combined module of the equaliser and Decoder II, as well as $\mathcal{A}_{III}$ and $\bar{\mathcal{A}}_{III}$ for the EXIT curve $T_{u_3}(i)$ of the inner equaliser. Then we have,

$$\bar{\mathcal{A}}_I \approx R_I, \tag{4.19}$$

and for BPSK modulation and MAP equalisation

$$\mathcal{A}_{III} \approx C_{UI}, \tag{4.20}$$

where C_{UI} is the uniform-input capacity of the communication channel seen in the schematic of Fig. 4.4.

Specifically, in Fig. 4.18 the area under the EXIT curve of the MAP equaliser is $\mathcal{A}_{III} \approx 0.59$ and the area under the EXIT curve of the MMSE equaliser is $\mathcal{A}'_{III} \approx 0.57$. When expressed in terms of bits per channel use, these values approximate the channel's capacity. The slight throughput loss of the latter is due to the employment of the MMSE criterion, which is inferior in comparison to the MAP criterion. Since the intermediate code has a unity rate, the area $\mathcal{A}_{II}$ under the EXIT curve of the combined module is equal to $\mathcal{A}'_{III}$. In order to achieve convergence, there has to be an open tunnel between the EXIT curve of the combined module and that of the outer code. Accordingly we have $\mathcal{A}'_{III} > \bar{\mathcal{A}}_I$ or $\mathcal{A}_{II} > \bar{\mathcal{A}}_I$. Hence the area under the EXIT curve of the MMSE equaliser is equal to the maximum achievable information rate, when using MMSE equalisation, which is achieved when the EXIT curve of the outer code is perfectly fitted to the EXIT curve of the combined module. According to these area properties, the channel capacity derived for a uniformly distributed input and the maximum information rate achieved when using MAP/MMSE equalisation may be readily obtained for the three-path channel of Eq. (4.8), which is depicted as a function of E_s/N_0 in Fig. 4.19. It can be seen that compared with MAP equalisation, there is an inherent information rate loss, when employing the MMSE equalisation, especially at high channel SNRs. The achievable information rates of the systems employing rate-1 intermediate codes are also plotted, which show negligible information rate loss.

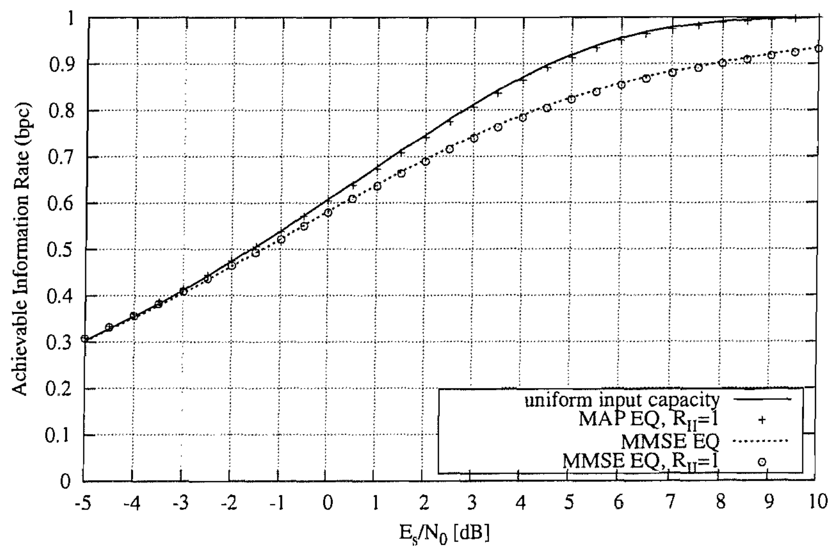


Figure 4.19: The achievable information rate for systems employing MAP/MMSE equalisation for transmission over the three-path channel of Eq. (4.8).

As seen from Fig. 4.18, there is an open tunnel between the EXIT curve of the combined module and that of the outer RSC code at the convergence threshold of $E_b/N_0 = 2.8$ dB, i.e. we have $\mathcal{A}_{II} - \bar{\mathcal{A}}_I > 0$, which implies that the channel capacity is not reached. On the other hand, if we fix the system's throughput to 0.5 bits per channel use, the convergence threshold may be further lowered, provided that a matching outer code can be found. More explicitly, we found that the area under the EXIT curve of the equaliser at $E_b/N_0 = 1.6$ (or equivalently at $E_s/N_0 = -1.4$) dB is about 0.50. Hence our task now is to search for an outer code whose EXIT curve is perfectly matched to that of the combined module of the equaliser and Decoder II, so that the maximum information rate is achieved.

Irregular codes, such as irregular LDPC codes, irregular RA codes [173] and IRCCs [84, 85], are reported to have flexible EXIT characteristics, which can be optimised to match the EXIT curve of the combined equaliser and Decoder II module, in order to create an appropriately shaped EXIT-chart convergence tunnel. For the sake of simplicity, IRCCs are considered in the forthcoming section.

4.5.1 Design of Irregular Convolutional Codes

The family of IRCCs was proposed in [84, 85], which is constituted by a set of convolutional codes having different code rates. They were specifically designed with the aid of EXIT charts [102], for the sake of improving the convergence behaviour of iteratively decoded systems. To be more specific, an IRCC is constructed from a family of P subcodes. First, a rate-r convolutional mother code C_1 is selected and the $(P-1)$ remaining subcodes C_k of rate $r_k > r$ are obtained by puncturing. Let L denote the total number of encoded bits generated from the K uncoded information bits. Each subcode encodes a fraction of $\alpha_k r_k L$ of the original uncoded information bits and generates $\alpha_k L$ encoded bits. Given our overall average code rate target of $R \in [0, 1]$, the weighting coefficient α_k has to satisfy:

$$1 = \sum_{k=1}^{P} \alpha_k, \; R = \sum_{k=1}^{P} \alpha_k r_k, \; \text{and } \alpha_k \in [0, 1], \; \forall k. \tag{4.21}$$

Clearly, the individual code rates r_k and the weighting coefficients α_k play a crucial role in shaping the EXIT function of the resultant IRCC. For example, in [85] a family of $P = 17$ subcodes were constructed from a systematic, rate-1/2, memory-4 mother code defined by the generator polynomial $(1, g_1/g_0)$, where $g_0 = 1 + D + D^4$ is the feedback polynomial and $g_1 = 1 + D^2 + D^3 + D^4$ is the feedforward one. Higher code rates may be obtained by puncturing, while lower rates are created by adding more generators and by puncturing under the constraint of maximising the achievable free distance. The two additional generators used are $g_2 = 1 + D + D^2 + D^4$ and $g_3 = 1 + D + D^3 + D^4$. The resultant 17 subcodes of [85] have coding rates spanning from $0.1, 0.15, 0.2, \cdots$, to 0.9.

The IRCC constructed using this procedure has the advantage that the decoding of all subcodes may be performed using the same mother code trellis, except that at the beginning of each block of $\alpha_k r_k L$ number of trellis transitions/sections corresponding to the subcode C_k, the puncturing pattern has to be restarted. Trellis termination is necessary only after all of the K uncoded information bits have been encoded.

The EXIT function of an IRCC can be obtained from those of its subcodes. Denote the EXIT function of the subcode k as $T_{c_1,k}(I_{A(c_1)})$. Assuming that the trellis segments of the subcodes do not significantly interfere with each other, which might change the associated transfer characteristics, the EXIT function $T_{c_1}(I_{A(c_1)})$ of the target IRCC is the weighted superposition of the EXIT function $T_{c_1,k}(I_{A(c_1)})$ [85], yielding,

$$T_{c_1}(I_{A(c_1)}) = \sum_{k=1}^{P} \alpha_k T_{c_1,k}(I_{A(c_1)}). \tag{4.22}$$

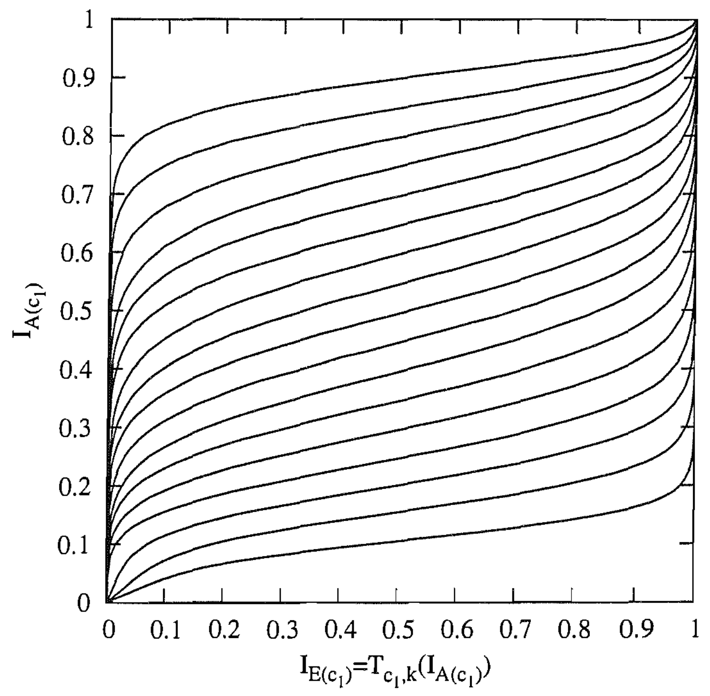


Figure 4.20: EXIT functions of the 17 subcodes in [85].

For example, the EXIT functions of the 17 subcodes used in [85] are shown in Fig. 4.20. We now optimise the weighting coefficients, $\{\alpha_k\}$, so that the IRCC's EXIT curve of Eq. (4.22) matches the combined EXIT curve of Eq. (4.17). The area under the combined EXIT curve of Eq. (4.17) at $E_b/N_0 = 1.8$ dB is $\mathcal{A}_{II} \approx 0.51$, which indicates that this E_b/N_0 value is close to the lowest possible convergence threshold for a system having an outer coding rate of $R_I = 0.5$. We optimise $T_{c_1}(I_{A(c_1)})$ of Eq. (4.22) at $E_b/N_0 = 1.8$ dB by minimising the square of the error function

$$e(i) = [T'_{u_2}(i, E_b/N_0) - T_{c_1}^{-1}(i)] \tag{4.23}$$

subject to the constraints of (4.21) and $e(i) > 0$ over all i:

$$J(\alpha_1, \cdots, \alpha_P) = \int_0^1 e^2(i)\mathrm{d}i, \quad e(i) > 0, \forall i \in [0,1]. \tag{4.24}$$

In more tangible physical terms, we minimise the area between the projected EXIT curve and the

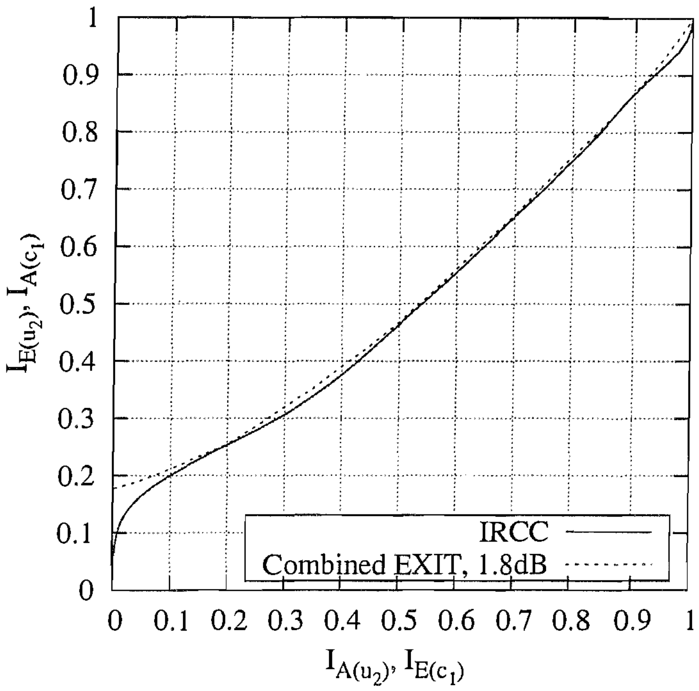


Figure 4.21: The combined EXIT function of Eq (4.17) and the EXIT function of the optimised IRCC at $E_b/N_0 = 1.8$ dB.

EXIT curve of the outer code, as seen in Fig. 4.8. This results in a good match between the two curves, ultimately leading to a narrow EXIT-chart tunnel, which implies a near-capacity operation attained at the cost of a potentially high number of decoding iterations. The associated high complexity is characteristic of schemes operating beyond the cut-off rate. The curve-fitting problem portrayed in Eq. (4.24) is a quadratic programming problem, which can be solved by the gradient descent based iterative solution proposed in [84]. With the aid of this algorithm [84], the optimised weighting coefficients are obtained as:

$$\begin{aligned}[\alpha_1, \cdots, \alpha_{17}] &= [0,\ 0.01374,\ 0,\ 0.28419,\ 0.09306,\ 0, \\ &\quad 0,\ 0.10614,\ 0.09578,\ 0.06183,\ 0,\ 0, \\ &\quad 0.19685,\ 0,\ 0.02059,\ 0,\ 0.12784]. \end{aligned} \tag{4.25}$$

The resultant EXIT curve of the optimised IRCC is shown in Fig. 4.21.

4.5.2 Simulation Results

Monte Carlo simulations were performed for characterising both the IRCC design and the convergence predictions of Section 4.5.1. As our benchmarker scheme, the RSC(2,1,2) code having octal generator polynomials of (2/3) was employed as Encoder I in the schematic of Fig. 4.4. As our proposed scheme, the IRCC designed in Section 4.5.1 was used as Encoder I. The rest of the system components were the same for both schemes and all the system parameters are listed in Table 4.4.

Table 4.4: System parameters used in the simulations of Section 4.5.2.

Encoder I	Half-rate RSC(2,1,2) code, G=2/3 or Half-rate IRCC of Eq. (4.25)
Encoder II	Generator Polynomials 1/3
Modulation	BPSK
CIR	$[0.407\ 0.815\ 0.407]^T$
Block Length	L = 100 000 bits
Overall Coding Rate	0.5

Fig. 4.22 depicts the BER performance of both schemes. It can be seen that after 12 iterations, the three-stage scheme employing an IRCC achieves a BER low than 10^{-5} at about $E_b/N_0 = 2.4$ dB, which outperforms the three-stage scheme using a regular RSC code by about 0.5 dB. Moreover, if more iterations are invoked, for example, $I = 30$ iterations, the three-stage scheme employing an IRCC is capable of achieving convergence at $E_b/N_0 = 2$ dB, which is very close to the convergence threshold of $E_b/N_0 = 1.8$ dB predicted by the EXIT chart analysis. Hence, a total additional gain of about 1 dB is achievable by using the optimised IRCC.

The decoding trajectory recorded during our Monte Carlo simulations using the optimised IRCC is depicted in Fig. 4.23. It can be seen that at $E_b/N_0 = 2$ dB a narrow tunnel exists between the EXIT curve of the outer IRCC code and that of the combined module of the equaliser and Decoder II. Observe furthermore that the simulation-based recorded trajectory matches these EXIT curves quite closely, except for a few iterations come to the point of convergence. Furthermore, since the tunnel between the two EXIT curves is narrow, numerous iterations are required for enabling the iterative receiver to converge to the point of (1,1). When aiming for approaching the channel capacity even more closely, the tunnel becomes even narrower, therefore more iterations would be needed, resulting in a drastically increased complexity. At the same time, even longer interleavers will be required for ensuring that the soft information exchanged between the SISO modules remains uncorrelated,

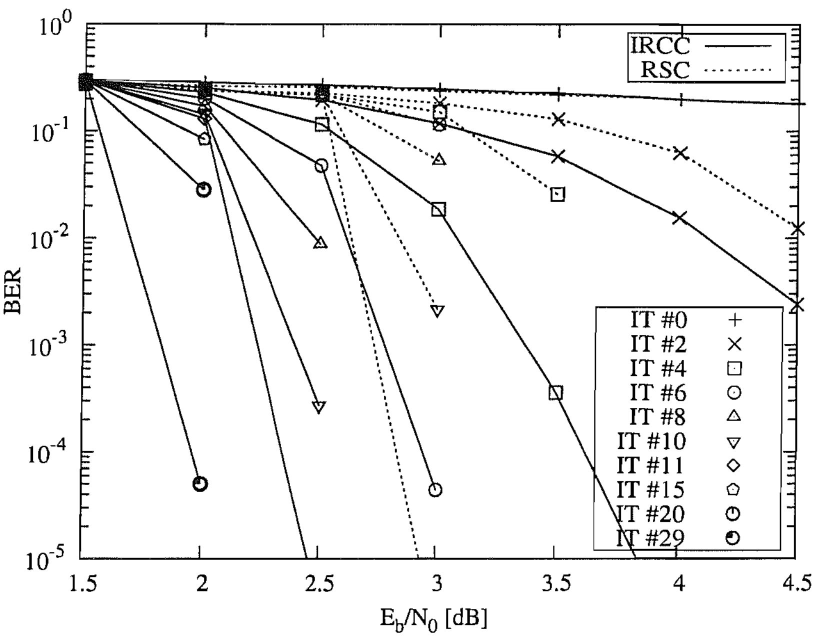


Figure 4.22: BER performances for the three-stage serial concatenation scheme of Fig. 4.4 using both regular RSC code and IRCC for transmission over the dispersive AWGN channel having a CIR of Eq. (4.8). The system parameters are outlined in Table 4.4.

so that the recorded decoding trajectory follows the EXIT curves. This is another manifestation of Shannon's channel capacity theorem [6].

4.6 Rate-Optimisation of Serially Concatenated Codes

In Section 4.4.2, we have shown that by using a unity-rate intermediate code and a half-rate RSC(2,1,2) outer code, the three-stage system of Fig. 4.4 became capable of achieving convergence at $E_b/N_0 = 2.8$ dB, which is 1.4 dB away from the channel capacity achievable in case of uniformly distributed inputs and 1.2 dB away from the maximum achievable information rate of MMSE equalisation. Below, we will investigate the performance of the three-stage system, when employing an intermediate code having a rate of $R_{II} < 1$. In other words, we fix the overall coding rate $R_{tot} = 0.5$ (i.e. the system throughput to be 0.5 bits per channel use assuming BPSK) and optimise the rate allocation between the intermediate code and the outer code as well as their generator polynomials and puncturing matrix, so that convergence is achieved at the lowest possible E_b/N_0 value.

For simplicity, we limit our search space to regular convolutional codes having constraint lengths of $K < 4$. For the rate allocation between the intermediate code and the outer code, we consider two cases: $R_{II}^1 = 3/4$, $R_I^1 = 2/3$ and $R_{II}^2 = 2/3$, $R_I^2 = 3/4$. The rate-2/3 and rate-3/4 codes are

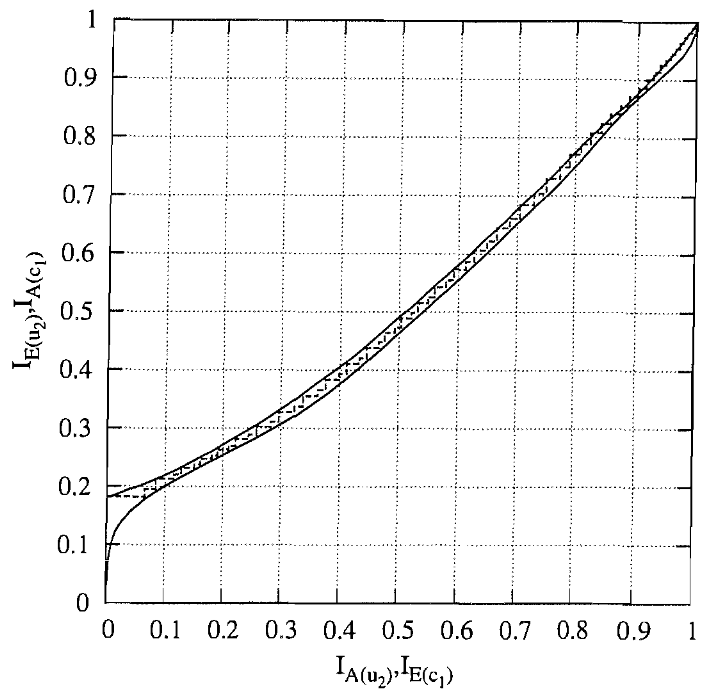


Figure 4.23: The decoding trajectory of the three-stage system using IRCC at $E_b/N_0 = 2$ dB. The EXIT function of the IRCC, $I_{E(c_1)} = T_{c_1}(I_{A(c_1)})$, and that of the combined module of the equaliser and Decoder II, $I_{E(u_2)} = T'_{u_2}(I_{A(u_2)})$, are also shown.

Table 4.5: Puncturing matrice for generating rate-2/3 and rate-3/4 codes from rate-1/2 mother codes

Rate-2/3			
$P_{11} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$		$P_{12} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$	
Rate-3/4			
$P_{21} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$	$P_{22} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$	$P_{23} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$P_{24} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$

generated by puncturing the rate-1/2 mother code and the puncturing matrices used are listed in Table 4.5. An exhaustive search was carried out and the combinations which exhibit good convergence properties are listed in Table 4.6. Furthermore, we depicted the EXIT functions of the SCCs in Fig. 4.24. Note that the EXIT function of each SCC was obtained by using the EXIT module of each component code, rather than by simulations, which incurs a substantiallylower complexity.

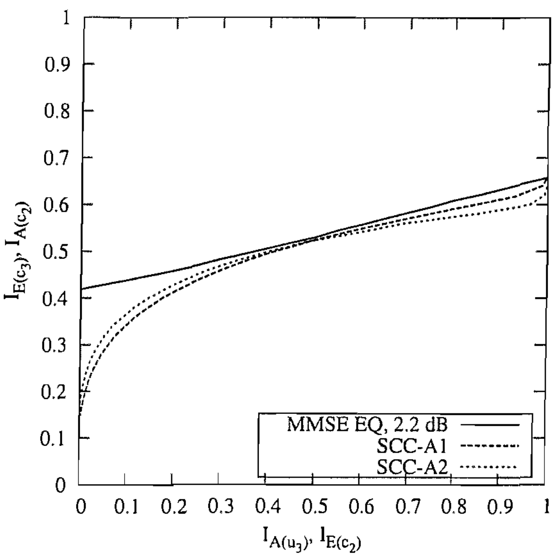
It can be seen from Table 4.6 that the lowest achievable convergence threshold of 2.2 dB is achievable by the SCC-A1/A2 and SCC-B4 schemes. This value is 0.6 dB lower than that of the system

Table 4.6: Optimised serially concatenated codes and the convergence thresholds of the corresponding three-stage systems

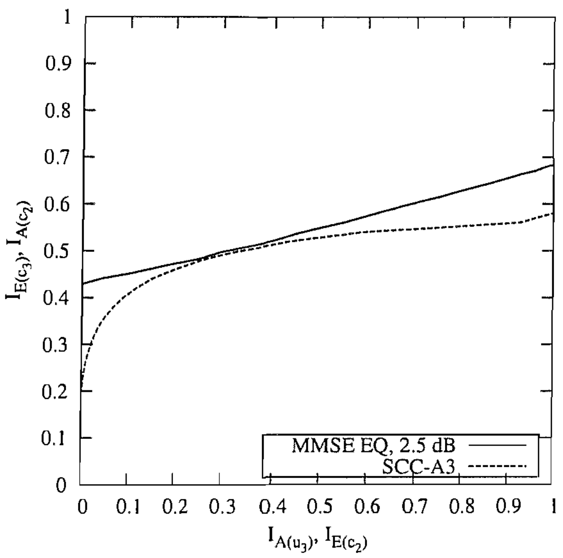
	Inter rate-3/4 code	Outer rate-2/3 code	$E_b/N_0\|_{th}$
SCC-A1	RSC(3,4,2), $G = 2/3$, $P = P_{24}$	RSC(2,3,2), $G = 2/3$, $P = P_{12}$	2.2 dB
SCC-A2	RSC(3,4,2), $G = 2/3$, $P = P_{24}$	RSC(2,3,3), $G = 6/7$, $P = P_{12}$	2.2 dB
SCC-A3	RSC(3,4,2), $G = 2/3$, $P = P_{24}$	RSC(2,3,3), $G = 5/7$, $P = P_{12}$	2.5 dB
SCC-A4	RSC(3,4,3), $G = 7/5$, $P = P_{24}$	RSC(2,3,2), $G = 2/3$, $P = P_{12}$	2.3 dB
SCC-A5	RSC(3,4,3), $G = 7/5$, $P = P_{24}$	RSC(2,3,3), $G = 5/7$, $P = P_{11}$	2.4 dB
	Inter rate-2/3 code	Outer rate-3/4 code	$E_b/N_0\|_{th}$
SCC-B1	RSC(2,3,2), $G = 2/3$, $P = P_{12}$	RSC(3,4,3), $G = 5/7$, $P = P_{21}$	2.5 dB
SCC-B2	RSC(2,3,2), $G = 2/3$, $P = P_{12}$	RSC(3,4,3), $G = 5/7$, $P = P_{22}$	2.5 dB
SCC-B3	RSC(2,3,2), $G = 2/3$, $P = P_{12}$	RSC(3,4,3), $G = 5/7$, $P = P_{23}$	2.4 dB
SCC-B4	RSC(2,3,2), $G = 2/3$, $P = P_{12}$	RSC(3,4,3), $G = 5/7$, $P = P_{24}$	2.2 dB
SCC-B5	RSC(2,3,3), $G = 5/7$, $P = P_{12}$	RSC(3,4,2), $G = 2/3$, $P = P_{21}$	2.3 dB
SCC-B6	RSC(2,3,3), $G = 5/7$, $P = P_{12}$	RSC(3,4,2), $G = 2/3$, $P = P_{24}$	2.5 dB

using rate-1 intermediate codes as discussed in Section 4.4. In order to verify our EXIT chart analysis, Monte Carlo simulations were also conducted and the BER performance of the corresponding three-stage system using, for instance, the SCC-A2 scheme, is depicted in Fig. 4.25. It can be seen that a BER of 10^{-5} is achieved at around $E_b/N_0 = 2.4$ dB, which is close to our EXIT chart prediction.

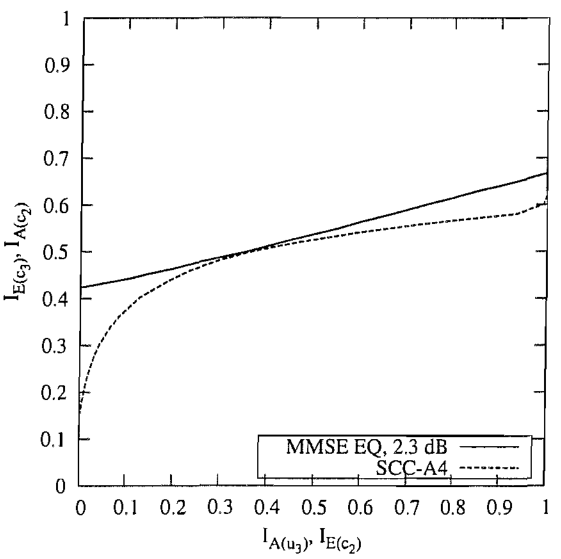
Although the three-stage system using an intermediate code having rate less than unity may achieve a lower convergence threshold than the system using a rate-1 intermediate code, as expected, the maximum achievable information rate is reduced. Let $\mathcal{A}_{II}$ be the area under the EXIT curve of the combined module of the inner equaliser and the intermediate decoder. According to the area properties of EXIT charts [120], the coding rate of the outer code R_I should satisfy $R_I < \mathcal{A}_{II}$ so that convergence may be achieved. Therefore the total coding rate $R_{tot} \triangleq R_I R_{II}$ should satisfy $R_{tot} < R_{II}\mathcal{A}_{II}$. Since $\mathcal{A}_{II} \leq 1$, it is evident that the system's throughput is bounded by R_{II}. For example, the achievable information rates of the systems using intermediate codes of rates of $R_{II} = \{1, 3/4, 2/3\}$ are depicted in Fig. 4.26. It can also be seen that the maximum achievable information rate is affected by the constraint length of the intermediate code. Additional information rate loss is incurred by using a relatively low constraint length code.



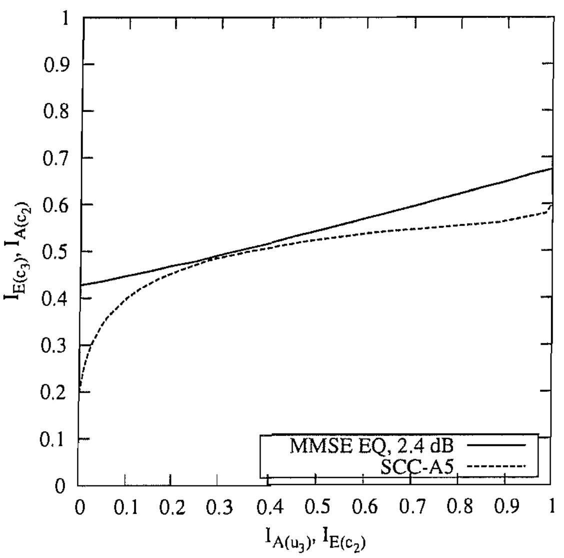
(a) The EXIT functions of SCC-A1 and SCC-A2



(b) The EXIT function of SCC-A3



(c) The EXIT function of SCC-A4



(d) The EXIT function of SCC-A5

Figure 4.24: The EXIT functions of some of the optimised SCCs listed in Table 4.6.

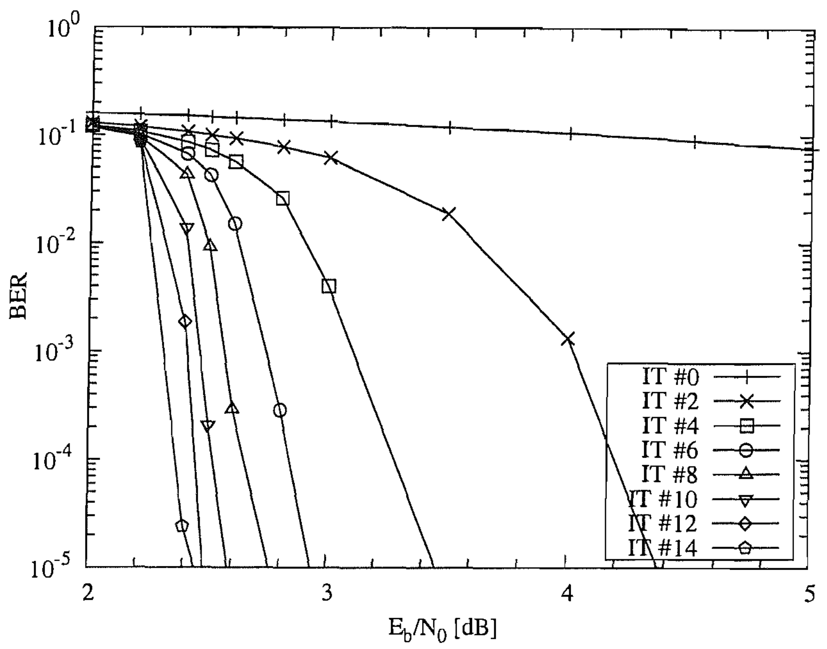


Figure 4.25: BER performance of the three-stage system of Fig. 4.4 when employing the SCC of SCC-A2 listed in Table 4.6 for transmission over the dispersive AWGN channel having a CIR of Eq. (4.8). The block length of the input data is $L = 10^5$ bits.

4.7 Joint Source-Channel Turbo Equalisation Revisited

In Section 3.4.4 we introduced a three-stage joint source and channel decoding as well as turbo equalisation scheme, as depicted in Fig. 3.40 and 3.41. Moreover, it was shown in Fig.3.42 that the iteratively detected amalgamated scheme significantly outperformed the separate source and channel decoding as well as turbo equalisation scheme. In our forthcoming discourse, we will investigate the convergence behaviour of the iterative detection aided amalgamated source-channel decoding and turbo equalisation scheme with the aid of the EXIT-module based convergence analysis technique described in Section 4.4.

For the sake of convenience, we re-draw the iterative receiver structure in Fig. 4.27, which is similar to the three-stage turbo equalisation aided receiver of Fig. 4.4. However, the channel equaliser considered here employs the APP equalisation algorithm described in Section 3.4 instead of the MMSE equalisation algorithm of Section 4.2. Furthermore, the intermediate decoder is constituted by a half-rate memory-4 RSC decoder and the outer decoder is constituted by the SISO VLC decoder characterised in Section 3.3. The rest of the system parameters are the same as in Table 3.2. As shown in Fig. 4.27, the input of the iterative receiver is constituted by the ISI-contaminated channel observations, and after a number of joint channel equalisation, channel decoding and source

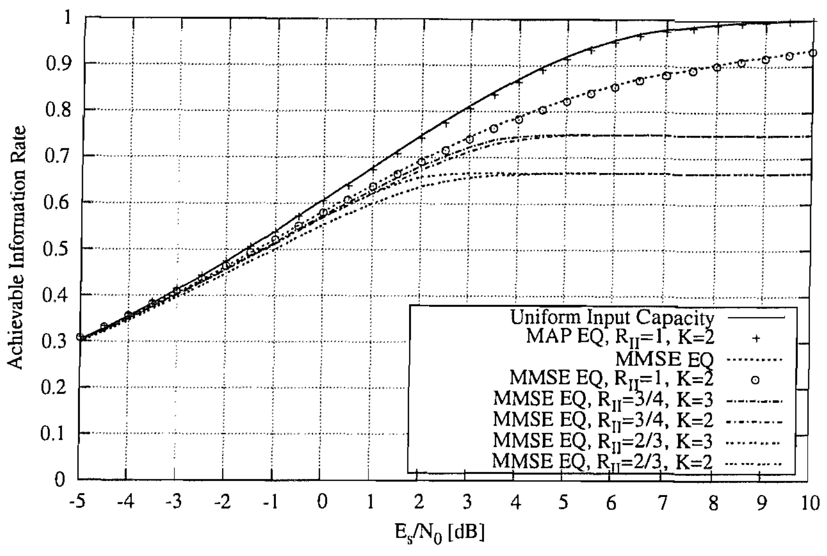


Figure 4.26: Achievable information rates for systems employing MAP/MMSE equalisation and various intermediate codes for transmission over the three-path channel of Eq. (4.8).

decoding iterations, which are performed according to a specific activation order, the source symbol estimates are generated.

Let us first consider the EXIT function of the combined module constituted by the channel equaliser and the channel decoder, which is shown in Fig. 4.28 and indicated by a solid line with plus-symbols. The EXIT function of the outer VLC decoder is also depicted using a solid line with times-symbols. It can be seen that at $E_b/N_0 = 1.5$ dB an open tunnel is formed between the two EXIT functions, which implies that the amalgamated source-channel decoding and turbo equalisation scheme is capable of converging to the point C_2 of $(I^{CH}_{A(u_2)} = I^{VLC}_{E(c_1)} \approx 1, I^{CH}_{E(u_2)} = I^{VLC}_{A(c_1)} \approx 1)$, hence an infinitesimally low SER may be achieved. By contrast, if a separate source-channel decoding and turbo equalisation scheme is employed, which implies that the VLC decoder is invoked once after a certain number of channel equalisation and channel decoding iterations, the receiver can only converge to the relatively low mutual information point of C_1, as indicated in Fig. 4.28.

On the other hand, let us consider the EXIT function of the combined module constituted by

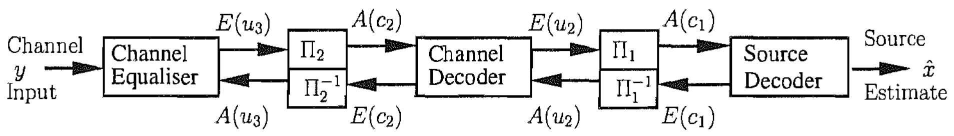


Figure 4.27: The receiver schematic of the joint source-channel turbo equalisation scheme described in Section 3.4.4.

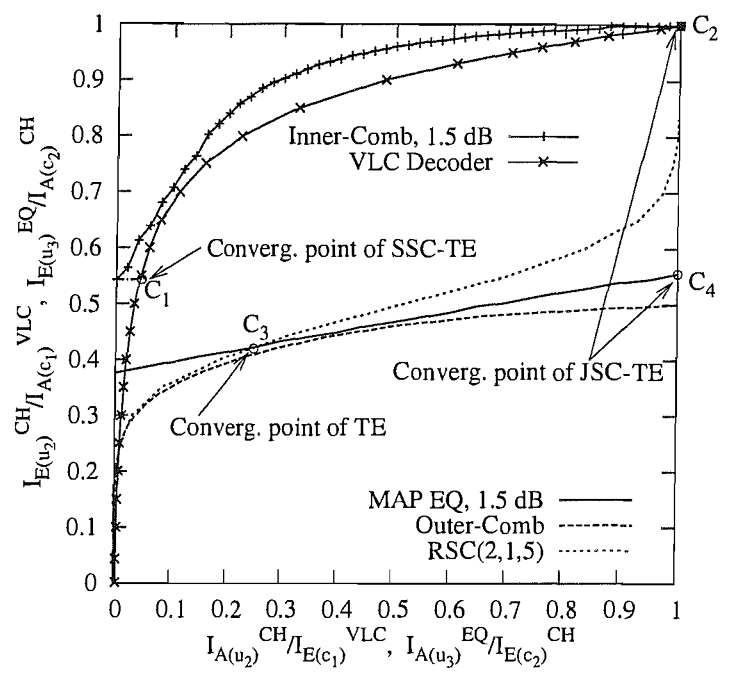


Figure 4.28: EXIT chart of the joint source-channel turbo equalisation receiver of Fig. 4.27, where SSC-TE denotes the separate source-channel turbo equalisation scheme, JSC-TE represents the joint source-channel turbo equalisation scheme and TE represents the two-stage turbo equalisation scheme.

the channel decoder and the VLC decoder, which is depicted in Fig. 4.28 using a dense dashed line. The EXIT function of the outer channel equaliser is also shown as a solid line. As expected, at $E_b/N_0 = 1.5$ dB an open tunnel is formed between the two EXIT functions, and the receiver becomes capable of converging to the point C_4 of $(I_{A(u_3)}^{EQ} = I_{E(c_2)}^{CH} \approx 1, I_{E(u_3)}^{EQ} = I_{A(c_2)}^{CH} \approx 0.55)$, which implies that residual errors may persist at the output of the channel equaliser, but nonetheless, a near-zero probabilitiy of error can be achieved at the output of the channel decoder and the VLC decoder. By contrast, a conventional two-stage turbo equalisation scheme consisting of only channel equalisation and channel decoding without source decoding can only converge to the relatively low mutual information point of C_3, as indicated in Fig. 4.28.

In summary, when incorporating the outer VLC decoder into the iterative decoding process, the three-stage amalgamated source-channel turbo equalisation scheme has a convergence threshold of about $E_b/N_0 = 1.5$ dB, which is significantly lower than that of the separate source-channel turbo equalisation scheme.

4.8 Summary and Conclusions

In this chapter, we investigated several three-stage serially concatenated turbo schemes from an EXIT chart aided perspective. As a result, a number of system design guidelines were obtained, which will be summarized below.

1) The three-stage concatenation principle

   The advocated system design principles were exemplified by an iterative three-stage MMSE equalisation scheme, where the iterative receiver was constituted by three SISO modules, namely the inner MMSE equaliser, the intermediate channel decoder and the outer channel decoder. From the point of view of precoding, the inner equaliser and the intermediate decoder of Fig. 4.4 may be regarded as a combined module and the EXIT function of this combined module was shown in Fig. 4.7. Although the EXIT function of the stand-alone equaliser cannot reach the point of (1,1) (i.e., $T_{c_1}(1) \neq 1$ c. f. Eq. (4.12)), it can be seen that the EXIT function of the combined module can (i.e., $T'_{u_2}(I_{A(u_2)} = 1, E_b/N_0) = 1$ c. f. Eq. (4.17)). Therefore the three-stage system becomes capable of achieving the convergence point of (1,1), which corresponds to an infinitesimally low BER.

   Alternatively, the intermediate decoder and the outer decoder can be viewed as a combined module, which results in a serially concatenated turbo-like code. The EXIT function of this combined module was depicted in Fig. 4.9. It can be seen that the combined module has a steep EXIT characteristics. For example, after $I = 10$ iterations between the intermediate decoder and the outer decoder, the EXIT function of the combined module reaches the upper bound value of 1 for any *a priori* information input larger than 0.6., i.e. we have $T'_{c_2}(I_{A(c_2)}) = 1, \forall I_{A(c_2)} > 0.6$. This characteristic matches that of the inner equaliser quite well and the corresponding three-stage system is capable of achieving $I_{E(c_2)} = 1$, which implies that we have $I_{E(c_1)} = I_{A(u_2)} = 1$ according to the EXIT characteristics of Decoder II, as seen in Fig. 4.6(a). Again, an infinitesimally low BER can be achieved.

2) EXIT chart matching based optimisation and the use of irregular codes

   In the context of three-stage concatenations, the outer code is optimised, so that its EXIT function is matched to that of the combined module of the inner equaliser and the intermediate decoder, yielding the lowest E_b/N_0 convergence threshold. Interestingly, we found in the context of Fig. 4.8 that as far as conventional convolutional codes are concerned, relatively weak codes having a short memory may result in the lowest convergence thresholds in the investigated three-stage context. This is due to the low initial value ($I_{E(u_2)} = T'_{u_2}(0, E_b/N_0)$) of the

EXIT function of the combined module, which results in an undesirable EXIT function intersection with that of a strong code earlier than a weak code, as seen in Fig. 4.18. Furthermore, if a well-designed IRCC is invoked as the outer code, the convergence threshold of the three-stage system may become even lower. For example, in our system of Section 4.5 employing an IRCC, a near-zero BER is achieved at $E_b/N_0 = 2$ dB, which is only about 0.3 dB away from the lowest achievable convergence threshold constitued by the capacity.

3) Rate optimisation of SCCs

In addition to rate-1 codes, lower rate intermediate codes may also be used in three-stage systems. Such systems have been shown in Fig. 4.24 and Fig. 4.25 to exhibit lower convergence thresholds and better BER performances than those using rate-1 intermediate codes and regular outer codes. Even lower convergence thresholds can be achieved, if irregular outer codes are used, however, the decoding complexity typically becomes significantly higher. The disadvantage of using intermediate codes having $R_{II} < 1$ is the reduction of the maximum achievable information rate as seen in Fig. 4.26, however, several SCCs having different intermediate coding rates can be designed and activated in order to render the channel coding scheme adaptive to the time-variant channel conditions.

4) Optimisation of activation orders

For multiple concatenation of more than two SISO modules, the activation order of the component decoders is an important issue, which may affect the computational complexity, delay and performance. With the aid of 1D and 2D EXIT functions, we are able to predict the number of three-stage activations required for achieving converge. It was observed in Table 4.2 that the component module which has the strongest error correction capability, for example, the outer code, should be invoked more frequently, in order that the total number of three-stage activations is minimised. Moreover, considering the computational complexities of the component modules, their activation order should be tuned to achieve the target tradeoff between fast convergence and low complexity. The activation order thus obtained is rather heuristic and might not be the optimal one. However, the optimal activation order would vary as a function of the channel SNR, the interleaver block length, the affordable total complexity, etc. Hence, we surmise that the proposed heuristic techniques are adequate.

Aditionally, similar to the case of two-stage systems, the decoding/detection processes of the three-stage systems may be visualised using EXIT charts as decoding trajectories. We have shown in Section 4.4.4 that either two 3D EXIT charts or two 2D EXIT charts can be used for characterizing the decoding process in terms of the associated mutual information exchange. Since 2D charts are easier to manipulate and interpret, hence they are preferable.

Finally, the analysis and design procedure advocated here may be applied in the context of diverse iterative receivers, employing multiple SISO modules, such as the jointly designed source coding and space-time coded modulation schemes of [174, 175]. The MMSE equaliser used in the three-stage system of Fig. 4.4 discussed here may also be replaced by an M-ary demapper or a MIMO detector and the resultant system can be optimised with the aid of the methods proposed here. The design of further near-capacity systems constitutes our future research.

Chapter 5

Conclusions and Future Work

This chapter summarises the key contents and contributions of each chapter and presents their most salient conclusions. Our suggestions for future research are outlined thereafter.

5.1 Chapter Summaries

5.1.1 Chapter 1

This chapter constitutes the general background of our studies throughout this thesis. More specifically, an overview of the literature of source encoding and soft source decoding was presented in Section 1.1.1. Then the development of iterative decoding techniques and their convergence analysis was described in Section 1.1.3. Furthermore, as a special case of iterative decoding, joint source-channel decoding was introduced and the main contributions to the open literature were summarised in Section 1.1.2. Finally, the organisation of the thesis is described in Section 1.3, while our novel contributions were highlighted in Section 1.4.

5.1.2 Chapter 2

Chapter 2 commenced with the description of general source models, among which a memoryless source model having a known finite alphabet such as that described in Section 2.2 was used throughout the thesis. Then various source codes such as Huffman codes, RVLCs and VLEC codes were introduced in Section 2.3, along with their construction methods. An important contribution of this chapter is that a generic algorithm was presented for the construction of efficient RVLCs and VLEC codes. The philosophy of our proposed algorithm is that we first construct an initial RVLC or VLEC code using existing methods such as those described in [26, 28, 35], then we optimise the codeword length distribution of the resultant code length-by-length. For example, Fig. 5.1 shows the evolution of the codeword length histograms of the RVLC designed for the English Alphabet in Section 2.3.4.2. After 12 iterations of optimisation, the best codeword length distribution is found, resulting in a RVLC having the lowest average codeword length of $AL = 4.18732$.

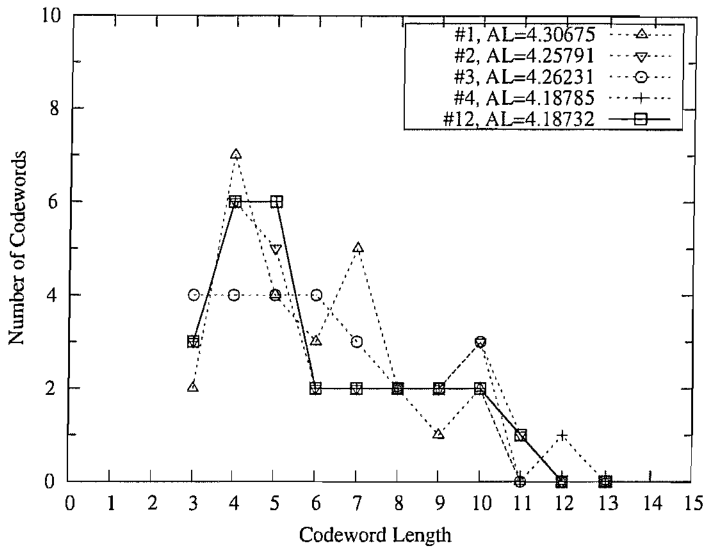
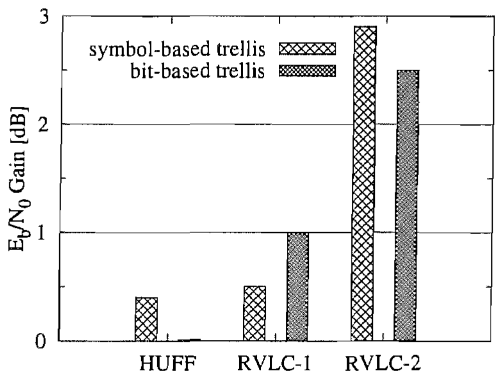


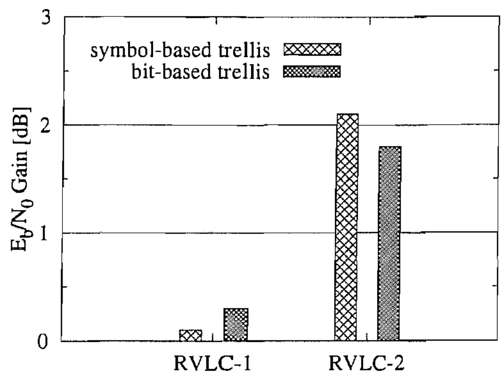
Figure 5.1: Evolution of the RVLC codeword length histograms. The RVLC is designed for the English Alphabet and its detailed construction process is described in Section 2.3.4.2. The codeword length distribution is optimised via a number of iterations for the sake of reducing the average codeword length.

Consequently, as shown in Table 2.3, Table 2.4 and Table 2.5, for a variety of memoryless sources, the proposed algorithm was capable of generating RVLCs of higher code efficiency and/or shorter maximum codeword length than the algorithms previously disseminated in the literature. Furthermore, as seen from Table 2.9 and Table 2.10, the proposed algorithm was also capable of constructing VLEC codes having similar code efficiency as those generated by the existing algorithm [34], but incurring a significantly lower complexity.

In Section 2.4, various VLC decoding methods were presented. First, the source information, such as the number of bits/symbols in the transmitted frames, and the constraints imposed by a source code and formulated in terms of the corresponding codebook were translated into a trellis representation, such as the symbol-based trellis described in Section 2.4.1.1 or the bit-based trellis described in Section 2.4.1.1. Then MAP/ML sequence estimation or MAP decoding may be performed, which were introduced in Section 2.4.2.1 and Section 2.4.2.2, respectively. It has been shown in Section 2.4 that trellis based soft-decoding provides an effective way of capitalising on the available information as much as possible. In general, the more information is utilised, the better the performance. This information can be explicit, such as the transmission frame length information, or implicit, such as the code constraint of a VLC. For example, soft-decoding generally outperforms hard-decoding, and the symbol-level trellis based decoding outperforms the bit-level trellis based decoding. Furthermore, as expected, VLCs having higher free distances outperform VLCs having lower free distances at the price of a reduced system throughput. Fig. 5.2 provides some quantitative results, summarising the conclusions of Section 2.4. It can be seen from Fig. 5.2(a) that soft-decision decoding significantly outperforms hard-decision decoding and the attainable E_b/N_0 gain improves upon increasing the VLC's free distance. Moreover, as seen from Fig. 5.2(b), the performance of soft ML decoding improves upon increasing the free distance of the VLC used.



(a) Soft Decision versus Hard Decisions



(b) Effects of different VLC Constraints

Figure 5.2: Comparison of the various VLC decoding schemes investigated in Section 2.4. Fig. 5.2(a) compares the performance of soft-decision and hard-decision decoding based schemes, where the E_b/N_0 gain is defined as the difference of the minimum E_b/N_0 values required for achieving a SER of 10^{-5} for transmission over AWGN channels, when using ML decoding. Fig. 5.2(b) demonstrates the effects of different VLC free distances, $d_f = 1$ (RVLC-1) and $d_f = 2$ (RVLC-2). The Huffman code (HUFF) based scheme is used as a benchmarker, where the E_b/N_0 gain is defined as the difference of the minimum E_b/N_0 values required for achieving a SER of 10^{-5} for transmission over AWGN channels, when using soft ML decoding.

5.1.3 Chapter 3

Chapter 3 provides an investigation of iterative source/channel decoding techniques. In this chapter, the source code, the channel code and the ISI channel are viewed as a serially concatenated system. Hence, iterative decoding may be performed, provided that the source decoder, channel decoder and the channel equaliser designed for the ISI channel are all SISO modules.

This chapter commenced with an overview of various concatenated schemes, as described in Section 3.1. Then a SISO APP decoding algorithm was introduced in Section 3.2.1. This algorithm provides a general description of any trellis-based APP decoding/detection scheme, which can be applied to source decoding, channel decoding and channel equalisation. Hence it constitutes the core module of iterative decoding schemes.

EXIT charts were introduced in Section 3.2.2. The mutual information between the data bits at the transmitter and the soft values at the receiver was used for characterising the decoding behaviour of a SISO APP module, resulting in the so-called EXIT functions. A histogram-based algorithm and its simplified version were introduced in Section 3.2.2.3 in order to evaluate the EXIT functions of a SISO APP module, followed by several examples of typical EXIT functions of SISO APP modules embedded in different positions of a concatenation scheme.

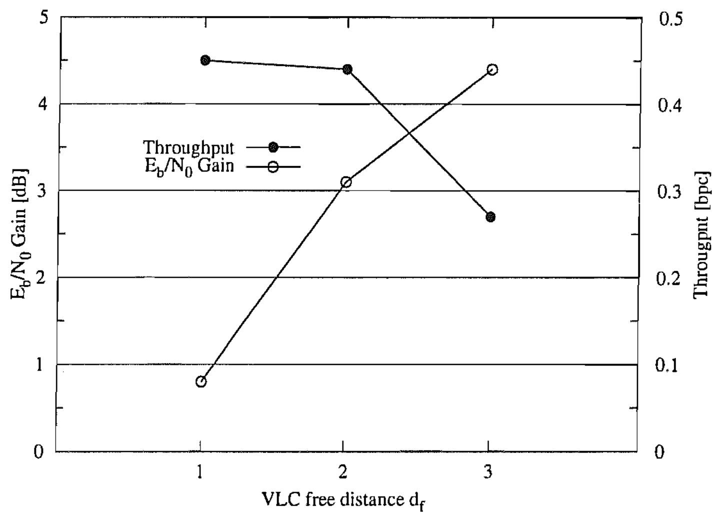


Figure 5.3: Free distance versus E_b/N_0 gain and throughput, where the E_b/N_0 gain is based on the minimum SNR value required for achieving a SER of 10^{-4} for transmission over AWGN channels and the scheme using the Huffman code (HUFF) is used as a benchmarker. The system model is described in Fig. 3.14, where the transmitter is constituted by a VLC encoder and a convolutional encoder, and the receiver is constituted by an APP convolutional decoder as well as an APP VLC decoder, which performs channel decoding and source decoding iteratively.

Given the EXIT characteristics of the constituent modules of a concatenated scheme, we may either predict or explain its convergence behaviour. This is carried out for iterative source/channel decoding for transmission over non-dispersive AWGN channels in Section 3.3 and for transmission over dispersive AWGN channels in Section 3.4. In the scenario of non-dispersive AWGN channels, it was shown in Fig. 3.15 that the free distance of the source code has to be larger than $d_f = 2$ in order that the iterative decoding scheme becomes capable of converging to the perfect mutual-information point of (1,1), which implies attaining infinitesimally low SERs. Furthermore, it was shown in Fig. 3.17-Fig. 3.24 that given a specific channel code, the system's convergence threshold decreases upon increasing the free distance of the source code, resulting in an improved SER performance. Fig. 5.3 serves as a summary of our main results provided in Section 3.3. It is worth noting that when the free distance of the VLC code is increased from $d_f = 1$ to $d_f = 2$, i.e. when using the code RVLC-2 instead of the code HUFF or RVLC-1, the system's throughput is only slightly decreased, but a significant E_b/N_0 gain is attained. Further increasing the free distance will continue to increase the attainable E_b/N_0 gain, while incurring a considerable loss of throughput.

In the scenario of dispersive channels, it was shown by both our EXIT chart analysis and our Monte Carlo simulations provided in Section 3.4.2 that the redundancy in the source codes is capable of effectively eliminating the ISI imposed by the channel, provided that channel equalisation and source decoding are performed jointly and iteratively. Furthermore, the higher the free distance of the source code, the closer the SER performance approaches the SER bound of non-dispersive AWGN channels.

Additionally, in Section 3.4.3 precoding was shown to be an effective way of "modifying" the EXIT characteristic of a channel equaliser. Most importantly, in conjunction with precoding the EXIT function of a channel equaliser becomes capable of reaching the point of $(I_A = 1,\ I_E = 1)$ as shown in Fig. 3.35, which is critical for avoiding potential error floors at the receiver's output. It was demonstrated in Fig. 3.36-Fig. 3.39 that the choice of the precoder depends on both the EXIT characteristics of the channel equaliser and that of the source decoder so that these two are matched to each other, hence achieving the lowest possible E_b/N_0 convergence threshold.

Fig. 5.4 summarises the main results of Section 3.4. It can be seen from Fig. 5.4 that the SER performance of both the scheme using RVLC-2 and that using VLEC-3 can be improved, when using appropriate precoders. However, although the precoder of $1 + \mathrm{D}^2$ is optimal for the scheme using RVLC-2, the precoder of $1 + \mathrm{D}$ constitutes a better choice for the scheme using VLEC-3.

Finally, the performance of a three-stage iterative receiver was evaluated in Section 3.4.4. The receiver of Fig. 3.41 consists of a channel equaliser, a channel decoder and a source decoder, where the extrinsic information is exchanged among all the three SISO modules, which hence constitutes a joint source-channel decoding and equalisation scheme. It was shown in Fig. 3.42 that by exploiting the source redundancy in the iterative decoding process, the system's performance was improved by 2 dB in terms of the E_b/N_0 values required for achieving the same SER, when compared to the separate source/channel decoding scheme. The convergence behaviour of this scheme was analysed using EXIT charts in Section 4.7 after we introduced the convergence analysis technique for multi-stage concatenated schemes in Chapter 4.

5.1.4 Chapter 4

Chapter 4 investigated the design of the three-stage serially concatenated turbo MMSE equalisation scheme seen in Fig. 4.4, which consisted of an inner channel equaliser, a unity-rate recursive in-

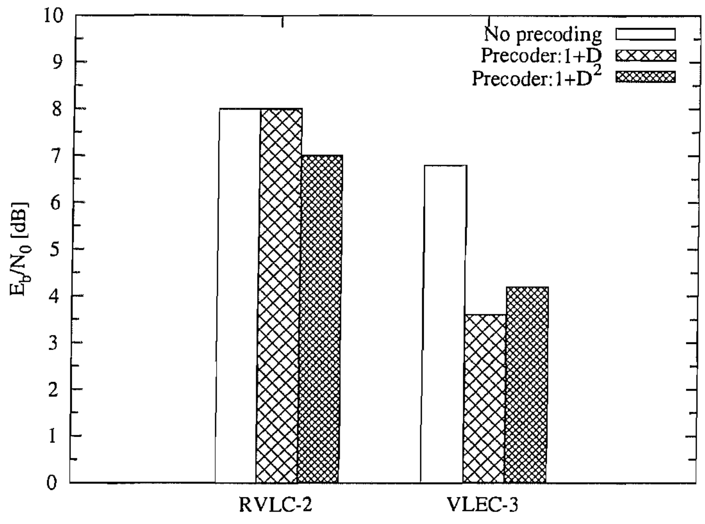


Figure 5.4: The effects of precoding and those of VLC free distances of $d_f = 2$ for the RVLC-2 and $d_f = 3$ for the VLEC-3 schemes on the attainable SER performance, when communicating over dispersive AWGN channels, where the E_b/N_0 value is the minimum SNR value required for achieving a SER of 10^{-4}. The system model is described in Fig. 3.28 and Fig. 3.29, where the transmitter is constituted by a VLC encoder as well as a precoder if precoding is employed, and the receiver is constituted by an APP channel equaliser as well as an APP VLC decoder, which performs channel equalisation and source decoding iteratively.

termediate channel code and an outer channel code. Firstly, a brief introduction to SISO MMSE equalisation was offered in Section 4.2, followed by an example of conventional two-stage turbo equalisation in Section 4.3. The main body of this chapter focused on the optimisation of three-stage turbo equalisation schemes by using EXIT chart analysis.

With the aid of the EXIT modules as proposed in Fig. 3.10 of Section 3.2.2, 3D EXIT chart analysis may be simplified to 2D EXIT analysis as shown in Fig. 4.7, 4.8 and 4.9 of Section 4.4.2. It was also shown in Fig. 4.8 of Section 4.4.2.1 that by employing a unity-rate recursive convolutional code as the intermediate constituent code, the three-stage scheme becomes capable of converging to the perfect mutual information point.

Moreover, the outer constituent code was optimised in Section 4.4.2.2 for achieving the lowest possible E_b/N_0 convergence threshold. Interestingly, it was observed in Fig. 4.8 that relatively weak codes having short memories resulted in a lower convergence threshold than strong codes having long memories.

Additionally, the activation order of the component decoders was optimised in Section 4.4.2.3 for achieving the convergence at the lowest possible E_b/N_0 value, while maintaining a low decoding complexity. It was found in Table 4.2 that by invoking the outer and intermediate decoder of Fig. 4.4

more frequently the total number of decoder activations is reduced, resulting in a decreased decoding complexity.

The BER performance of the optimised scheme was evaluated in Section 4.4.3, which verified the EXIT chart analysis provided in Section 4.4.2. The iterative decoding process was visualised using both 3D and 2D EXIT charts as shown in Fig. 4.12-4.15 of Section 4.4.4. Furthermore, the effects of different interleaver block lengths were discussed in Fig. 4.16 of Section 4.4.5. Generally, the longer the interleaver length, the closer the simulated performance matches the EXIT chart analysis. It was found in Fig. 4.16 that an interleaver length on the order of 10^5 bits is sufficiently high for achieving a good match with the decoding trajectory recorded. Fig. 5.5 provides some quantitative results summarised from Section 4.4.5. It can be seen from Fig. 5.5 that when the interleaver depth is increased from $L = 10^3$ bits to $L = 10^4$ bits, a significant coding gain may be attained. Further increasing the interleaver depth to $L = 10^5$ bits, however, results in a marginal increase of the coding gain. Naturally, the attainable iteration gain is increased upon increasing the interleaver depth.

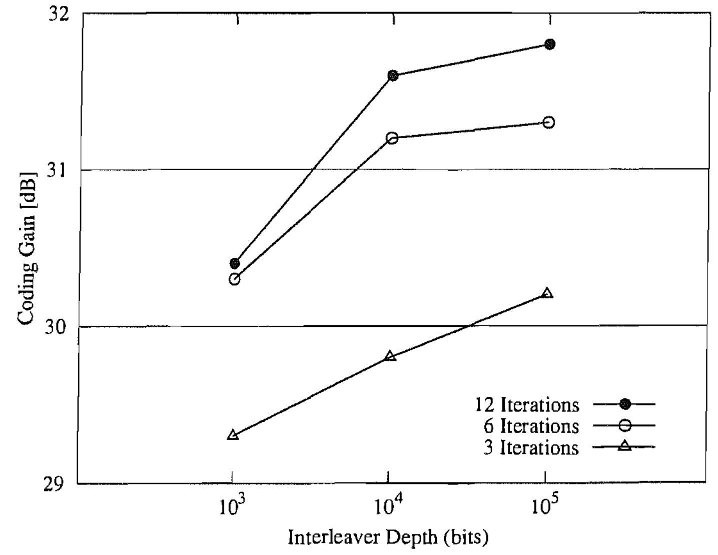


Figure 5.5: Achievable coding gains at a BER of 10^{-4} for the three-stage turbo equalisation scheme of Fig. 4.10 using different interleaver depths. The turbo equalisation scheme is constituted by a RSC(2,1,2) code as the outer code, a unity-rate RSC(1,1,2) code as the intermediate code and an inner MMSE equaliser as described in Section 4.4.

In Section 4.5, the maximum achievable information rate of the three-stage turbo equalisation scheme of Fig. 4.4 was analysed. Then an IRCC was invoked as the outer constituent code, whose EXIT function was optimised for matching that of the combined module of the inner channel equaliser and the intermediate channel decoder, so that the EXIT tunnel-area between these two EXIT functions was minimised. The Monte Carlo simulation results provided in Fig. 4.22 of Section 4.5.2 show that the performance of the resultant scheme is only 0.5 dB away from the channel capacity.

Finally, the employment of non-unity rate intermediate codes was also considered in Section 4.6. It was shown in Fig. 4.26 that as expected, the maximum achievable information rate of such schemes was reduced in comparison to the schemes using unity-rate intermediate codes. By contrast, the E_b/N_0 convergence threshold may be decreased, when only regular convolutional codes are used. A number of optimised serially concatenated codes were obtained and listed in Table 4.6.

As a summary, Fig. 5.6 compares the distance to capacity for the various MMSE turbo equalisation schemes discussed in Chapter 4.

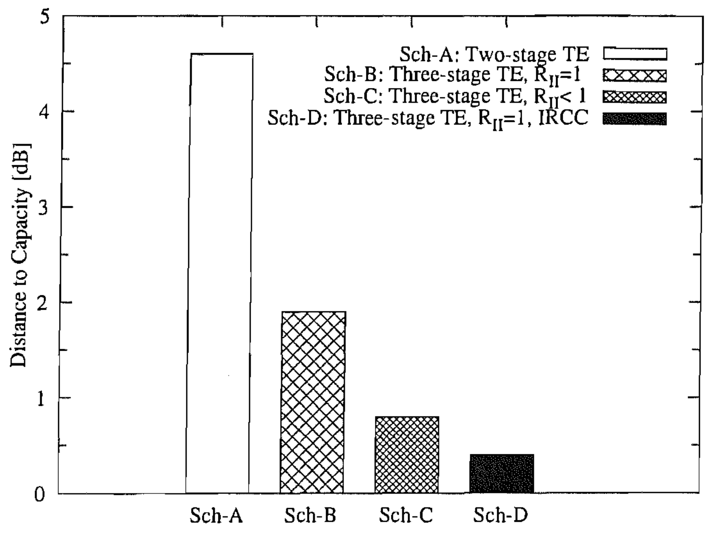


Figure 5.6: Distance to capacity for the various MMSE turbo equalisation schemes of Chapter 4, where the scheme of Sch-A represents the conventional two-stage turbo equalisation scheme of Fig. 4.1. The schemes of Sch-B, Sch-C and Sch-D denotes the same three-stage turbo equalisation scheme of Fig. 4.4, but differ in the channel codes used. The scheme of Sch-B employs a unity-rate RSC(1,1,2) code as the intermediate code and a RSC(2,1,2) code as the outer code. The scheme of Sch-C uses a SCC of SCC-A2 described in Table 4.6, which is constituted by a rate-3/4 RSC(3,4,2) code as the intermediate code and a RSC(2,3,3) code as the outer code. The scheme of Sch-D employs the same unity-rate RSC(1,1,2) code used in the scheme of Sch-B as the intermediate code, while using the IRCC described in Section 4.5.1 as the outer code.

5.2 Future Work

It appears promising to extend the work reported in this thesis in the following directions:

1) In Chapter 2, we proposed a heuristic construction algorithm for RVLCs and VLEC codes. It would be interesting to find the necessary and sufficient conditions for the existence of a RVLC/VLEC code satisfying certain distance constraints. Moreover, it would be interesting

to find the lower bound of the average codeword length of a RVLC/VLEC code, which might facilitate the construction of the corresponding RVLC/VLEC code.

2) In Chapter 2 and Chapter 3, we only considered a memoryless source model having a known alphabet. It would be also interesting to consider a source having memory, i.e. when the consecutive source symbols are correlated rather than correlated. For example, usually the Gaussian Markov source model is used to describe the residual redundancy after source compression [7]. It would be worth investigating the effects of source correlation, especially in the context of the three-stage joint source/channel decoding and equalisation scheme of 3.41.

3) We considered the design of IRCCs in Chapter 4. It would be interesting to investigate other irregular codes, such as bi-regular or irregular LDPCs [82] and irregular repeat accumulate codes [173]. Following the philosophy of IRCCs, it is even possible to design IRregular Variable-Length Codes (IRVLC) [176]. Furthermore, only scalar channels were considered in Chapter 4, but it would be interesting to generalise the three-stage concatenated scheme of Fig. 4.4 to vector channels, such as three-stage MIMO detection schemes.

4) During our investigations, we assume the presence of perfect channel state information (CSI), such as the knowledge of CIR. We also assumed that the AWGN power spectrum density is static and known at the receiver. However, in practical communication systems, the CSI is usually time-variant and has to be regularly estimated. Therefore, more realistic schemes should combine the EXIT chart-based code optimisation with channel estimation. The design of similar adaptive coding schemes constitutes an interesting future research topic.

5) In general, EXIT chart analysis assumes the exchange of independent *extrinsic* information between two concatenated SISO modules. However, the convergence behaviour of an actual system matches that predicted by the EXIT chart only when the interleaver length is sufficiently long. Therefore the analysis and optimisation of a system employing short interleaver lengths required by delay-sensitive interactive voice/video communication systems constitutes a challenging problem for our future research.

6) EXIT chart analysis may also prove beneficial for the joint design of real-time interactive audio/video source encoders and channel encoders as well as VLC MIMO transmit diversity schemes.

Appendix A

VLC Construction Algorithms

A.1 RVLC Construction Algorithm A

We now introduce the MRG-based algorithm proposed in [26].

Definition 12 Minimum Repetition Gap (MRG): A codeword c with k bits is represented as $c_1c_2\dots c_k$. We form a new bit pattern b by concatenating c with a g-bit pattern x denoted as b $=$ cx. Every bit in x can be viewed as 0 or 1 in the following calculation. The minimum repetition gap g of c refers to the minimal interval that all the bits in b are repeated every g bits. That is

$$\min_{1\leq g\leq k}\{b_i = b_j\}, \text{ for all } 1 \leq i, j \leq (g+k)\ s.t.\ i \equiv j \pmod{g}.$$

For example, given a codeword c $= 010$, then we have b $= 010x_1x_2x_3$. We can set x $= 101$ such that $b_1 = b_3 = b_5(i.e. x_2) = 0$ and $b_2 = b_4(x_1) = b_6(x_3) = 1$, therefore, g is equal to 2 in this case. Also, we can set x $= 010$ and obtain $g = 3$ for $b_1 = b_4(x_1) = 0$, $b_2 = b_5(x_2) = 1$, and $b_3 = b_6(x_3) = 0$, therefore, the MRG $= \min\{2, 3\} = 2$.

The RVLC construction method of [26] is based on such a conjecture: For any two candidate codewords c_1 and c_2 at level l, let g_1 and g_2 denote the MRGs of c_1 and c_2, respectively. $V(\mathrm{c}_1, k)$ and $V(\mathrm{c}_2, k)$ each denotes the set of candidate codewords at level k that violate the affix condition for c_1 and c_2. If $g_1 \leq g_2$, then $|V(\mathrm{c}_1, k)| \leq |V(\mathrm{c}_2, k)|$ for $k \geq l$.

Therefore if we always choose codewords with the smallest MRGs while satisfying the affix condition, we can make the usable candidate codewords as many as possible for next levels.

Let $U = \{u_1, \dots, u_M\}$ be a M-ary i.i.d. information source with the probability mass function $P_u = \{p_1, \dots, p_M\}$, $p_1 \geq \dots \geq p_M$. The RVLC construction algorithm of [26] is then sumarized as follows:

Algorithm A:

Step 1: Construct a Huffman code C_H, which maps the source symbols into binary codewords $\mathrm{w}_1, \ldots, \mathrm{w}_M$ of length $l_1, \ldots, l_M$, $L_{min} = l_1, \leq \ldots, \leq l_M = L_{max}$. Denote the bit length vector of C_H by $(n_H(1), \ldots, n_H(L_{max}))$, where $n_H(i)$ represents the number of Huffman codewords having length i. Initialize the bit length vector of the target asymmetrical RVLC, $n(i)$ by $n_H(i)$.

Step 2: Calculate the total number of available candidate codewords of a full binary tree at level i, denoted by $n_a(i)$, and the MRG for each candidate codeword, then arrange these codewords based on the increasing order of MRGs. If $n(i) \leq n_a(i)$, assign $n(i)$ codewords. Otherwise, assign all the available codewords, and the remaining $(n(i) - n_a(i))$ codewords are added to $n(i+1)$ so as to be assigned at the next stage, i.e.

$$n(i+1) := n(i+1) + n(i) - n_a(i), \qquad n(i) := n_a(i).$$

Step 3: Step 2 is repeated until every source symbol has been assigned a codeword.

A.2 RVLC Construction Algorithm B

In [28] Lakovic *et al* improved the above algorithm by establishing the formal relationship between the number of available RVLC codewords of any length and the structure of shorter RVLC codewords that had been selected, and pointed out that the MRG metric was a special case of this relationship. We now re-derive this relationship.

Let a binary codeword w of length i be denoted as $\mathrm{w} = (w_1 w_2 \cdots w_i)$. Define the prefix set $P_j(\mathrm{w})$ as the set of length-j codewords, $j > i$, which are prefixed by w, and define the suffix set $S_j(\mathrm{w})$ as the set of legnth-j codewords, $j > i$, which are suffixed by w. It is obvious that:

$$|P_j(\mathrm{w})| = |S_j(\mathrm{w})| = 2^{(j-i)}$$

Assume that $W = \{\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_m\}$ is a set of RVLC codewords of lengths $l_1 \leq l_2 \leq \cdots \leq l_m$. Denote with $n_a(j)$ the number of codewords of length $j, j > l_m$, which are neither prefixed nor suffixed by any codeword from W. Since the total number of length-j codewords is equal to 2^j, it holds that

$$n_a(j) = 2^j - \Big| \bigcup_{\mathrm{w}_k \in W} P_j(\mathrm{w}_k) \Big| - \Big| \bigcup_{\mathrm{w}_k \in W} S_j(\mathrm{w}_k) \Big| + \Big| \bigcup_{\mathrm{w}_k, \mathrm{w}_h \in W} P_j(\mathrm{w}_k) \cap S_j(\mathrm{w}_h) \Big|.$$

For RVLCs, all the prefix sets, as well as all the suffix sets, are disjoint, therefore

$$\begin{aligned} n_a(j) &= 2^j - \sum_{\mathbf{w}_k \in W} (|P_j(\mathbf{w}_k)| + |S_j(\mathbf{w}_k)|) + \sum_{\mathbf{w}_k \in W} \sum_{\mathbf{w}_h \in W} |P_j(\mathbf{w}_k) \cap S_j(\mathbf{w}_h)| \\ &= 2^j - \sum_{k=1}^{m} 2^{j-l_k+1} + \sum_{\mathbf{w}_k \in W} \sum_{\mathbf{w}_h \in W} |P_j(\mathbf{w}_k) \cap S_j(\mathbf{w}_h)|. \end{aligned}$$

Now define the affix set $A_j(\mathbf{w}_k, \mathbf{w}_h)$ as the set of codewords of length j, $j > \max(l_k, l_h)$, prefixed by $\mathbf{w}_k$ and suffixed by $\mathbf{w}_h$

$$A_j(\mathbf{w}_k, \mathbf{w}_h) = P_j(\mathbf{w}_k) \cap S_j(\mathbf{w}_h).$$

Obviously, $n_a(j)$ depends on the cardinalities of affix sets $A_j(\mathbf{w}_k, \mathbf{w}_h)$, i.e., on the affix index of the set W, defined as

$$a_j(W) = \sum_{\mathbf{w}_k \in W} \sum_{\mathbf{w}_h \in W} |A_j(\mathbf{w}_k, \mathbf{w}_h)|.$$

The cardinalities of affix sets $A_j(\mathbf{w}_k, \mathbf{w}_h)$ are given as follows [32].

For $\max(l_k, l_h) < j < l_k + l_h$

$$|A_j(\mathbf{w}_k, \mathbf{w}_h)| = \begin{cases} 1, & \text{if } (w_{k_{j-l_h+1}} w_{k_{j-l_h+2}} \cdots w_{k_{l_k}}) = (w_{h_1} w_{h_2} \cdots w_{h_{l_k+l_h-j}}) \\ 0, & \text{otherwise} \end{cases}$$

while for $l_k + l_h \leq j$

$$|A_j(\mathbf{w}_k, \mathbf{w}_h)| = 2^{j-l_k-l_h}.$$

With the above formulas, the MRG can be interpreted as follows: The MRG of a length-i codeword $\mathbf{w}$, denoted as $g(\mathbf{w})$, is equal to the minimal positive integer g such that $|A_{i+g}(w, w)| > 0$. Algorithm A [26] selects the codewords with the lowest $g(\mathbf{w})$, which contributes to the increase in $n_a(i+g)$, with the lowest g first, because $n_a(i+g)$ depends on $|A_{i+g}(\mathbf{w}, \mathbf{w})|$. However this algorithm does not consider the elements $|A_{i+g}(\mathbf{w}_k, \mathbf{w}_h)|$, $\mathbf{w}_k \neq \mathbf{w}_h$, which have a significant impact on $n_a(i+g)$, and consequently on the coding efficiency. Therefore, Lajovic *et al.* [28] proposed a new construction algorithm which differed Algorithm A only in the codeword selection criterion:

Algorithm B:

Step 1 and Step 3: Same as Step 1 and Step 3 of Algorithm A.

Step 2: If $n(i) \leq n_a(i)$, $n(i)$ codewords should be assigned. There are $\binom{n_a(i)}{n(i)}$ candidate sets to be selected. Calculate $n_a(i+1)$ for all possible selections, select the set with the maximal $n_a(i+1)$. If there are several candidate sets which yield the same $n_a(i+1)$, choose the set that yields the highest $n_a(i+2)$.

If $n(i) > n_a(i)$, assign all the available codewords, and the remaining $(n(i) - n_a(i))$ codewords are added to $n(i+1)$ so as to be assigned at the next stage, i.e.

$$n(i+1) = n(i+1) + n(i) - n_a(i), \quad n(i) = n_a(i).$$

Algorithm B can generate RVLCs with smaller average code length than Algorithm A, however, the complexity is much higher. At each level i, with $n_a(i) \geq n(i)$, Algorithm A considers $n_a(i)$ MRGs, while Algorithm B considers $\binom{n_a(i)}{n(i)}$ affix indices.

A.3 Greedy Algorithm (GA) and Majority Voting Algorithm (MVA)

Both of the GA and MVA may be applied for constructing a fixed-length code of length n having a given block distance of d from a given set of codewords which may not necessarily contain all possible n-tuples. Given a set of codewords W, the GA initialises the target codeword set C with an arbitrary codeword in W, then select the first available codeword in W that is at a distance of at least d from all the existing codewords in the set C. This procedure is repeated until no more legitimate codewords are available in the set W. This algorithm is easy to implement, however, the number of codewords in the resultant codeword set may be far from the maixmum achievable value, since it ignores the effect that the codewords selected at the current stage will limit the number of legitimate codewords at the next stage. By contrast, the MVA is usually capable of generating larger codeword sets by "looking" a step forward, whose procedure can be described as follows [35]:

Step 1: Initialise the target code C to the null code (i.e. no codewords selected).

Step 2: For each codeword in W determine the number of the rest codewords in W wich are at least distance d from the given codeword.

Step 3: Choose the codeword which has got the maximum number of other codewords which satisfy the minimum distance requirement. If there is more than one such codeword, then choose an arbitrary one. Add this codeword to the target code C.

Step 4: Remove from W all those codewords which do not satisfy the minimum distance requirement to the chosen codeword.

Step 5: Repeat Steps 2-4 with the new set W. Each time the selected codeword is added to the code C. The algorithm ends when W is empty.

The main disadvantage of the above algorithm is that as the number of codewords in W increases it becomes too computationally expensive and the GA will then be preferred.

Appendix B

SISO VLC Decoder

In chapter 2, we have derived the MAP VLC decoder as an inner code, which processes the channel output directly. In the following we derive the APP VLC decoder as an outer code, which processes the extrinsic output of the inner code. We derive the decoding procedure based on the SISO APP module introduced in [160], which was a slightly modified version of the BCJR algorithm [52] and was originally designed for convolutional codes.

The SISO module is a four-terminal device having two inputs and two outputs, as shown in Figure B.1. It accepts as its inputs, $P(\mathbf{u};\mathbf{I})$ and $P(\mathbf{c};\mathbf{I})$, namely the probability distributions of both the information symbols $\mathbf{u}$ and code symbols $\mathbf{c}$ labelling the edges of the code trellis, and forms the resultant outputs, $P(\mathbf{u};\mathbf{O})$ and $P(\mathbf{c};\mathbf{O})$, which constitute an update of these distributions based upon the code constraints, and these outputs represent the extrinsic information.

P(c; I)
P(u; I)
SISO
P(c; O)
P(u; O)

Figure B.1: A SISO moudle.

Following the notation of [160], the extrinsic information is calculated as follows. At time k the output probability distribution is evaluated as

$$\tilde{P}_k(c:O) = \tilde{H}_k \sum_{e:c(e)=c} \alpha_{k-1}(s^S(e))\beta_k(s^E(e))\gamma_k(e), \tag{B.1}$$

where e represents a branch of the trellis, $c(e)$, $s^S(e)$, and $s^E(e)$ are, respectively, the output symbol, the starting state and the terminating state of the branch e, while $\tilde{H}_k$ is a normalizing factor that ensures maintaining $\tilde{P}_k(0;O)+\tilde{P}_k(1;O) = 1$. The quantities $\alpha_k(\cdot)$ and $beta_k(\cdot)$ are obtained through the forward and backward recursions [160], respectively, which can be expressed as

$$\alpha_k(s) = \sum_{e:s^E(e)=s} \alpha_{k-1}[s^S(e)]\gamma_k(e), \tag{B.2}$$

$$\beta_k(s) = \sum_{e:s^S(e)=s} \beta_{k+1}[s^S(e)]\gamma_{k+1}(e), \tag{B.3}$$

in conjunction with the initial values of $\alpha_0(s) = 0$, and $\beta_N(s) = 0$ for all states, except for the root state, since the trellis always starts and ends at the root state.

In order to incorporate the source information, the branch transition probability in [160] is modified as follows:

$$\gamma_k(e) = P_k(c(e);I) \times P(e), \tag{B.4}$$

where $P(e) \triangleq P(c(e), s^E(e)|s^S(e))$ is the source *a priori* information associated with the branch transition probability. It is time invariant and determined by the source distribution, which can be calculated with the aid of the state transition probabilities as [155],

$$P(c(e), s^E(e)|s^S(e)) = \frac{\sum_{i\in g(s^E(e))} p_i}{\sum_{j\in g(s^S(e))} p_j}, \tag{B.5}$$

where $g(n)$ are all the codeword indices associated with node n in the code tree.

Therefore, the extrinsic information can be extracted by excluding the input probability of $P_k(c;I)$ from the output probability, yielding

$$\begin{aligned} P_k(c;O) &= \frac{\tilde{P}_k(c;O)}{P_k(c;I)} \\ &= H_k \sum_{e:c(e)=c} \alpha_{k-1}(s^S(e))\beta_k(s^E(e))P(e), \end{aligned} \tag{B.6}$$

where, again, H_k is a normalization factor. To reduce the computational complexity, the above multiplicative algorithm can be converted to an additive form in the log-domain. The logarithms of the above quantities are defined as follows:

$$\pi_k(c;I) \triangleq \ln[P_k(c;I)] \tag{B.7}$$

$$\pi_k(c;O) \triangleq \ln[P_k(c;O)] \tag{B.8}$$

$$A_k(s) \triangleq \ln[\alpha_k(s)] \tag{B.9}$$

$$B_k(s) \triangleq \ln[\beta_k(s)] \tag{B.10}$$

$$\Gamma_k(e) \triangleq \ln[\gamma_k(e)] \tag{B.11}$$

With the aid of these definitions, the SISO algorithm defined by the equations of (B.2), (B.3), and (B.6) can be correspondingly described as follows:

$$\pi_k(c;O) = \ln\left[\sum_{e:c(e)=c} \exp\{A_{k-1}[s^S(e)] + \Gamma_k(e) + B_k[s^E(e)]\}\right] + h_k \tag{B.12}$$

where h_k is a normalization constant. The quantities $A_k(\cdot)$ and $B_k(\cdot)$ can be obtained through the forward and backward recursions, respecitively, as follows:

$$A_k(s) = \ln\left[\sum_{e:s^E(e)=s} \exp\{A_{k-1}[s^S(e)] + \Gamma_k[e]\}\right] \tag{B.13}$$

$$B_k(s) = \ln\left[\sum_{e:s^S(e)=s} \exp\{B_{k+1}[s^S(e)] + \Gamma_{k+1}[e]\}\right] \tag{B.14}$$

with the initial values:

$$A_0(s) = \begin{cases} 0 & s \text{ is the root node} \\ -\infty & \text{otherwise} \end{cases}$$

$$B_N(s) = \begin{cases} 0 & s \text{ is the root node} \\ -\infty & \text{otherwise.} \end{cases}$$

For simplicity, we use LLRs instead of probabilities themselves. Therefore, the output extrinsic information in the form of LLRs can be computed as:

$$\begin{aligned} L(c_k;O) &= \pi_k(c=0;O) - \pi_k(c=1;O) \\ &= \ln\left[\sum_{e:c(e)=0} \exp\{A_{k-1}[s^S(e)] + \Gamma_k[e] + B_k[s^E(e)]\}\right] \\ &\quad - \ln\left[\sum_{e:c(e)=1} \exp\{A_{k-1}[s^S(e)] + \Gamma_k[e] + B_k[s^E(e)]\}\right] \end{aligned} \tag{B.15}$$

Correspondingly, the *a priori* probability $P_k(c;I)$ may also be calculated from the *a priori* LLR $L(c_k;I)$ as follows:

$$\begin{aligned} P_k(c;I) &= \frac{e^{\frac{c_k}{2}L(c_k;I)}}{e^{-\frac{1}{2}L(c_k;I)} + e^{\frac{1}{2}L(c_k;I)}} \\ &= \eta_k \, e^{\frac{c_k}{2}L(c_k;I)} \end{aligned} \tag{B.16}$$

where η_k is independent of the value of c_k, and can be cancelled out. Hence, the logarithm of the *a priori* probability may be calculated as:

$$\pi_k(c_k; I) = \frac{c_k}{2} L(c_k; I) \tag{B.17}$$

Appendix C

APP Channel Equalisation

The APP equaliser outputs the channel information, which is expressed as $\{L_f(d_k)\}$, associated with all data bits, given N_c received samples hosted by the vector $\mathrm{y} = y_1^{N_c}$ as well as the *a priori* information $\{L_b(d_k)\}$ corresponding to all data bits. The feed-forward information $L_f(d_k)$ corresponding to bit d_k is given by

$$\begin{aligned} L_f(d_k) &\triangleq \ln \frac{P(d_k = +1|\mathrm{y})}{P(d_k = -1|\mathrm{y})} - \ln \frac{P(d_k = +1)}{P(d_k = -1)} \\ &= L(d_k|\mathrm{y}) - L_b(d_k) \end{aligned} \tag{C.1}$$

where $L(d_k|\mathrm{y}) = \ln\left[P(d_k = +1|\mathrm{y})/P(d_k = -1|\mathrm{y})\right]$ represents the *a posteriori* log likelihood ratio (LLR) of the data bit d_k, which, may be computed using the algorithm described below by following the approaches outlined in [73,74,76,103].

The *a posteriori* LLR of the data bit d_k can be expressed in conjunction with the channel states s_{k-1} and s_k as

$$\begin{aligned} L\left(d_k|\mathrm{y}\right) &= \ln \frac{P\left(d_k = +1|\mathrm{y}\right)}{P\left(d_k = -1|\mathrm{y}\right)} \\ &= \ln \frac{\sum\limits_{\forall (i,j)\in\mathcal{B}:\ d_{i,j}=+1} P\left(s_k = q_j; s_{k-1} = q_i|y_1^{N_c}\right)}{\sum\limits_{\forall (i,j)\in\mathcal{B}:\ d_{i,j}=-1} P\left(s_k = q_j; s_{k-1} = q_i|y_1^{N_c}\right)} \\ &= \ln \frac{\sum\limits_{\forall (i,j)\in\mathcal{B}:\ d_{i,j}=+1} P\left(s_k = q_j; s_{k-1} = q_i; y_1^{N_c}\right)}{\sum\limits_{\forall (i,j)\in\mathcal{B}:\ d_{i,j}=-1} P\left(s_k = q_j; s_{k-1} = q_i; y_1^{N_c}\right)}. \end{aligned} \tag{C.2}$$

The sequence $y_1^{N_c}$ involved in the compuation of the probability $P(s_{k-1}, s_k, y_1^{N_c})$ can be written in a more detailed form as $P(s_{k-1}, s_k, y_1^{k-1}, y_k, y_{k+1}^{N_c})$. Applying the chain rule for joint probabilities, i.e., exploiting that we have $P(a,b) = P(a)P(b|a)$, this expression yields the following decomposition of $P(s_{k-1}, s_k, y_1^{N_c})$:

$$P(s_{k-1}, s_k, y_1^{N_c}) = P(s_{k-1}, y_1^{k-1})P(s_k, y_k|s_{k-1})P(y_{k+1}^{N_c}|s_k). \tag{C.3}$$

Let us introduce the following definitions:

$$\alpha_k(s) \triangleq P(s_k = s, y_1^k) \tag{C.4}$$

$$\beta_k(s) \triangleq P(y_{k+1}^{N_c}|s_k) \tag{C.5}$$

$$\gamma_k(s_{k-1}, s_k) \triangleq P(s_k, y_k|s_{k-1}). \tag{C.6}$$

Then, according to [52] the term $\alpha_k(s)$ can be computed via the recursion:

$$\alpha_k(s) = \sum_{\forall\, s' \in \mathcal{S}} \alpha_{k-1}(s')\gamma_k(s', s) \tag{C.7}$$

associated with the initial values of $\alpha_0(q_0) = 1$ and $\alpha_0(q_i) = 0$ for $i \neq 0$. The term $\beta_k(s)$ can be computed via the recursion:

$$\beta_k(s) = \sum_{\forall\, s' \in \mathcal{S}} \beta_{k+1}(s')\gamma_{k+1}(s, s'), \tag{C.8}$$

in conjunction with the initial values of $\beta_{N_c}(q_0) = 1$ and $\beta_{N_c}(q_i) = 0$ for $i \neq 0$. The term $\gamma_k(s_{k-1}, s_k)$ represents the transition probability from state s_{k-1} to state s_k, which can be further decomposed into

$$\gamma_k(s_{k-1}, s_k) = P(s_k|s_{k-1}) \cdot p(y_k|s_{k-1}, s_k). \tag{C.9}$$

The transition probability $\gamma_k(s_{k-1} = q_i, s_k = q_j)$ is zero, if the index pair (i,j) is not in $\mathcal{B}$. For pairs (i,j) within $\mathcal{B}$, the probability $P(s_k = q_j|s_{k-1} = q_i)$ is governed by the corresponding input symbol $d_{i,j}$ and $p(y_k|s_{k-1}, s_k)$ is governed by the corresponding output symbol $\nu_{i,j}$, which can be described as

$$\gamma_k(s_{k-1} = q_i, s_k = q_j) = \begin{cases} P(d_k = d_{i,j}) \cdot p(y_k|\nu_k = \nu_{i,j}), & (i,j) \in \mathcal{B} \\ 0 & (i,j) \notin \mathcal{B}. \end{cases} \tag{C.10}$$

Given the AWGN channel of Eq. (3.49), it follows that $p(y_k|\nu_k)$ is given by

$$p(y_k|v_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(- \frac{(y_k - \nu_k)^2}{2\sigma^2} \right). \tag{C.11}$$

From Eq. (C.2) and from the decomposition of $P(s_{k-1}, s_k, y_1^{N_c})$ given in C.3, it follows that

$$L(d_k|y_1^{N_c}) = \ln \frac{\sum\limits_{\forall (i,j)\in \mathcal{B}:\ d_{i,j}=+1} \alpha_{k-1}(q_i) \cdot \gamma_k(q_i, q_j) \cdot \beta_k(q_j)}{\sum\limits_{\forall (i,j)\in \mathcal{B}:\ d_{i,j}=-1} \alpha_{k-1}(q_i) \cdot \gamma_k(q_i, q_j) \cdot \beta_k(q_j)}. \tag{C.12}$$

As shown in Eq. (C.10), once the channel output samples are available, the corresponding values of $\gamma_k(s', s)$ can be computed and stored. With the aid of these values, both $\alpha_k(s)$ and $\beta_k(s)$ of Eq. (C.7) and (C.8) can be computed, and finally, the *a posteriori* LLR of Eq. (C.12) can be obtained. For the sake of reducing the associated complexity, these computations can be performed in the log-domain, which is described as follows.

Let us introduce the following notations:

$$A_k(s) \triangleq \ln[\alpha_k(s)], \tag{C.13}$$

$$B_k(s) \triangleq \ln[\beta_k(s)], \tag{C.14}$$

$$\Gamma_k(s', s) \triangleq \ln[\gamma_k(s', s)]. \tag{C.15}$$

Consequently, $A_k(s)$ can be formed iteratively as follows:

1) **Initialization**:

$$A_0(0) = 0 \text{ and } A_0(s \neq 0) = -\infty, \tag{C.16}$$

2) **Forward Recursion**:

$$A_k(s) = \ln\left[\sum_{\forall\ s'\in \mathcal{S}} \exp\{A_{k-1}(s') + \Gamma_k(s', s)\}\right], \quad k = 1, \dots, N_c. \tag{C.17}$$

$B_k(s)$ of (C.14) can be solved similarly as follows:

1) **Initialization**:

$$B_{N_c}(0) = 0 \text{ and } A_{N_c}(s \neq 0) = -\infty, \tag{C.18}$$

2) **Backward Recursion**:

$$B_k(s) = \ln\left[\sum_{\forall\ s'\in \mathcal{S}} \exp\{B_{k+1}(s') + \Gamma_{k+1}(s, s')\}\right], \quad k = N_c - 1, \dots, 0. \tag{C.19}$$

Furthermore, given the L-values of the encoded bits d_k, i.e., given the *a priori* information of $L(d_k)$, and given the channel observations, y_k, $\Gamma_k(s', s)$ seen in (C.15) can be calculated as follows:

$$\Gamma_k(q_i, q_j) = \begin{cases} \frac{1}{2} d_{i,j} L(d_k) - \frac{(y_k - \nu_k)^2}{2\sigma^2}, & (i,j) \in \mathcal{B} \\ 0 & (i,j) \notin \mathcal{B} \end{cases} \qquad k = 1, \ldots, N_c. \tag{C.20}$$

Finally, the *a posteriori* LLR of (C.12) can be obtained as follows:

$$\begin{aligned} L(d_k | y_1^{N_c}) &= \ln \sum_{\forall (i,j) \in \mathcal{B}:\ d_{i,j} = +1} \exp\{A_{k-1}(q_i) + \Gamma_k(q_i, q_j) + B_k(q_j)\} \\ &\quad - \ln \sum_{\forall (i,j) \in \mathcal{B}:\ d_{i,j} = -1} \exp\{A_{k-1}(q_i) + \Gamma_k(q_i, q_j) + B_k(q_j)\} \\ &\qquad k = 1, \ldots, N_c. \end{aligned} \tag{C.21}$$

The above calculations consist of the evaluation of the logarithm of a sum of exponentials, which can be done by using the approximation of either the MAX-Log approximation, or the Jacobian logrithm implemented by a single entry look-up table [156].

List of Symbols

General notation

- The superscript * is used to indicate complex conjugation. Therefore, a^* represents the complex conjugate of the variable a.
- The superscript ^{T} is used to denote the matrix transpose operation. Therefore, $\mathbf{a}^T$ represents the transpose of the matrix $\mathbf{a}$.
- The superscript ^{H} is used to indicate the complex conjugate transpose operation. Therefore, $\mathbf{a}^H$ represents the complex conjugate transpose of the matrix $\mathbf{a}$.
- The notation $\hat{x}$ represents the estimate of x.

Special symbols

A:	A random discrete source.
$\mathscr{A}$:	A source alphabet.
$\mathcal{A}$:	The area under an EXIT curve.
$\bar{\mathcal{A}}$:	The area under the inverse curve of an EXIT function.
$\mathbf{A_c}$:	The *a priori* LLRs of the encoded bits.
$\mathbf{A_u}$:	The *a priori* LLRs of the information bits.
A_k:	The forward recursion metric of a MAP decoder in the log-domain.
a_i:	The legitimate symbols of a source alphabet.
B_k:	The backward recursion metric of a MAP decoder in the log-domain.
b_{min}:	The overall minimum block distance of a VLC.
C:	A VLC.
$\mathbf{C}$:	The encoded output sequence of a trellis encoder.
C_k :	The encoded output symbol of a trellis encoder.
C_{UI}:	The channel capacity when using uniformly distributed input.
c_{min}:	The minimum converging distance of the VLC.
$\mathbf{c}_j$:	A VLC codeword.
$c(e)$:	The output symbol associated with an edge e in a trellis.
d_{b_k}:	The minimum block distance for the length L_k of a VLC.
d_c:	The converging distance between two different-length codewords.
d_d:	The diverging distance between two different-length codewords.
d_f:	The free distance of a VLC.
d_{min}:	The minimum diverging distance of a VLC.
d_h:	The Hamming distance between two identical-length VLC codewords.
$\mathbf{E_u}$:	The *a posteriori* LLRs of the information bits.

$\mathbf{E_c}$:	The *a posteriori* LLRs of the encoded bits.
E_b:	Bit energy.
E_b/N_0:	Ratio of the bit energy to the noise power spectral density.
E_s:	Symbol energy.
E_s/N_0:	Ratio of the symbol energy to the noise power spectral density.
e:	The edge of a trellis.
F_N:	The extended code of order N of a VLC.
g_k:	The generator polynomials of a convolutional code.
H:	The entropy of a source.
h_k:	The coefficients of a channel impulse reponse.
I_i:	The information carried by a source symbol.
$\bar{L}$:	The average codeword length of a VLC.
$\ell(\mathbf{c}_j)$:	The length of a VLC codeword $\mathbf{c}_j$ in terms of code symbols or bits for binary codes.
N_0:	Single-sided power spectral density of white noise.
$\mathbf{n}$:	The length distribution vector of a VLC.
P_i:	The state probabilities of a discrete Markov model.
p_i:	The source symbol probabilities.
p_{ij}:	The state transition probabilities of a discrete Markov model.
R:	Coding rate.
R_I:	The source information rate measured in bits/second.
R_s:	The source emission rate measured in symbol/second.
$\mathcal{S}$:	The set of valid states of a trellis encoder.
$s^S(e)$:	The starting state of an edge e in a trellis.
$s^E(e)$:	The ending state of an edge e in a trellis.
$\mathbf{U}$:	The input symbol sequence of a trellis encoder.
U_k:	The input symbol of a trellis encoder.

$u(e)$:	The input symbol associated with an edge e in a trellis.
X_i:	The states of a discrete Markov model.
α:	The forward recursion metric of a MAP decoder.
β:	The backward recursion metric of a MAP decoder.
γ:	The branch transition metric of a MAP decoder.
ρ:	The correlation coefficient of a first-order Markov model.
η:	The efficiency of a VLC.
ν:	The code memory.
Γ_k:	The branch transition metric of a MAP decoder in the log-domain.
Π:	Interleaver.
Π^{-1}:	Deinterleaver.

List of Acronyms

APP	*a posteriori* Probability
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	Bit Error Rate.
BM	Branch Metric
BPSK	Binary Phase-Shift Keying
BSC	Binary Symmetric Channel
BTC	Block Turbo Code
CDMA	Code-Division Multiple-Access
CIR	Channel Impulse Response
CSI	Channel State Information
DMS	Discrete Memoryless Source
EXIT	EXtrinsic Information Transfer
FIR	Finite-length Impulse Response
FLC	Fixed-Length Coding
GSM	Global System of Mobile telecommunication
HC	Hybrid Concatenation

HMM	Hidden Markov Model
I.I.D.	independent and identically distributed
IRA	Irregular Repeat-Accumulate
IRCC	IRregular Convolutional Code
ISCD	Iterative Source-Channel Decoding
ISI	Inter-Symbol-Interference
JSCD	Joint Source-Channel Decoding
LDPC	Low-Density Parity Check
LLR	Log-Likelihood Ratio
MAP	Maximum *a posteriori*
MCE	Minimum Cross-Entropy
MIMO	Multiple-Input Multiple-Output
ML	Maximum Likelihood
MLSE	Maximum Likelihood Sequence Estimation
MMSE	Minimum Mean Square Error
MRG	Minimum Repetition Gap
MSSL	Maximum Symmetric-Suffix Length
NSC	Non-Systematic Convolutional
PC	Parallel Concatenation
PCC	Parallel Concatenated Code
PDF	Probability Density Function
PM	Path Metric
RA	Repeat-Accumulate
RSC	Recursive Systematic Convolutional
RVLC	Reversible Variable-Length Code.
SC	Serial Concatenation

SCC	Serially Concatenated Code
SCCD	Source-Controlled Channel Decoding
SER	Symbol Error Rate.
SISO	Soft-Input Soft-Output
SNR	Signal-to-Noise Ratio
SOVA	Soft-Output Viterbi Algorithm
TE	Turbo Equalisation
TPC	Turbo Product Code
VA	Viterbi Algorithm
VLC	Variable-Length Code.
VLEC	Variable-Length Error-Correcting
VLSI	Very-Large-Scale Integration

Bibliography

[1] J. Wang and L-L. Yang and L. Hanzo, "Iterative Construction of Reversible Variable Length Codes and Variable Length Error Correcting Codes," *IEEE Communications Letters*, pp. 671–673, August 2004.

[2] J. Wang and L-L. Yang and L. Hanzo, "Iterative channel equalization, channel decoding and source decoding," in *Proceedings of the 61st IEEE Vehicular Technology Conference*, (Sweden), May 2005.

[3] L.-L. Yang and J. Wang and R. Maunder and L. Hanzo, "Iterative Channel Equalization and Source Decoding for Vector Quantization Sources," in *Proceedings of the 63rd IEEE Vehicular Technology Conference*, (Melbourne, Austria), 7-10 May 2006.

[4] J. Wang and L.-L. Yang and L. Hanzo, "Combined Serially Concatenated Codes and MMSE Equalisation: an EXIT Chart Aided Perspective," in *Proceedings of the 64th IEEE Vehicular Technology Conference [CDROM]*, (Montréal, Canada), 25-28 September 2006.

[5] J. Wang and S. X. Ng and A. Wolfgang and L.-L. Yang and S. Chen and L. Hanzo, "Near-Capacity Three-Stage MMSE Turbo Equalization Using Irregular Convolutional Codes," in *Proceedings of the 4th International Symposium on Turbo Codes (ISTC'06) in connection with the 6th International ITG-Conference on Source and Channel Coding*, (Munich, Germany), 3-7 April 2006.

[6] C. E. Shannon, "A mathematical theory of communication," *Bell Systems Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[7] L. Hanzo and P. J. Cherriman and J. Streit, *Wireless Video Communications: Second to Third Generation Systems and Beyond.* IEEE Press, 2001.

[8] F. I. Alajaji and N. C. Phamdo and T. E. Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech," *IEEE Transactions on Speech Audio Processing*, vol. 4, pp. 325–336, September 1996.

[9] K. Sayood and J. C. Borkenhagen, "Use of residual redundancy in the design of joint source/channel coders," *IEEE Transactions on Communications*, vol. 39, pp. 838–846, June 1991.

[10] K. Sayood and F. Liu and J. D. Gibson, "A constrained joint source/channel coder design," *IEEE Journal in Selected Areas of Communications*, vol. 12, pp. 1584–1593, December 1994.

[11] R. E. V. Dyck and D. J. Miller, "Transport of wireless video using separate, concatenated, and joint source-channel coding," *Proceedings of IEEE*, vol. 87, pp. 1734–1750, October 1999.

[12] C. Guillemot and P. Siohan, "Joint source-channel decoding of variable-length codes with soft information: A survey," *EURASIP Journal on Applied Signal Processing*, pp. 906–927, June 2005.

[13] J. Proakis, *Digital Communications.* New York: McGraw-Hill, 4th ed., 2001.

[14] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of Institute of Radio Engineering*, vol. 40, pp. 1098–1101, September 1952.

[15] J. J. Rissanen, "Arithmetic codings as number representations," *Acta Polytechnica Scandinavica*, vol. 31, pp. 44–51, 1979.

[16] J. C. Kieffer, "A survey of the theory of source coding with a fidelity criterion," *IEEE Transactions on Information Theory*, pp. 1473–1490, September 1993.

[17] Y. Takishima and M. Wada and H. Murakami, "Reversible Variable Length Codes," *IEEE Transactions on Communications*, vol. 43, pp. 158–162, Feb./Mar./Apr. 1995.

[18] J. Wen and J. D. Villasenor, "A class of reversible variable length codes for robust image and video coding," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, ((Lausanne, Switzerland)), pp. 65–68, October 1997.

[19] J. Wen and J. D. Villasenor, "Reversible variable length codes for efficient and robust image and video coding," in *Data Compression Conference (DCC'98)* , (Snowbird, USA), pp. 471–480, 1998.

[20] S. W. Golomb, "Run-length encodings," *IEEE Transactions on Information Theory*, pp. 1473–1490, July 1966.

[21] R. F. Rice, "Some Practical Universal Noiseless Coding Techniques," *Technical Report JPL-79-22, Jet Propulsion Laboratory, Pasadena, California*, pp. 79–22, March 1979.

[22] J. Teuhola, "A Compression Method for Clustered Bit-Vectors," *Information Processing Letters*, pp. 308–311, October 1978.

[23] "Video Coding for Low Bit Rate Communication," *ITU-T Recommendation H.263*, 1998.

[24] "Generic coding of audio-visual objects," *ISO/IEC 14496-2 MPEG-4*, 1999.

[25] C. Tsai and J. Wu, "Modified symmetrical reversible variable-length code and its theoretical bounds," *IEEE Transactions on Information Theory*, vol. 47, pp. 2543–2548, September 2001.

[26] C. Tsai and J. Wu, "On constructing the Huffman-code-based reversible variable-length codes," *IEEE Transactions on Communications*, vol. 49, pp. 1506–1509, September 2001.

[27] K. Lakovic and J. Villasenor, "On design of error-correcting reversible variable length codes," *IEEE Communications Letters*, vol. 6, pp. 337–339, August 2002.

[28] K. Lakovic and J. Villasenor, "An algorithm for construction of efficient fix-free codes," *IEEE Communications Letters*, vol. 7, pp. 391–393, August 2003.

[29] C. Lin and Y. Chuang and J. Wu, "Generic construction algorithms for symmetric and asymmetric RVLCs," in *Proceedings of IEEE International Conference on Computational Science(ICCS)*, (Amsterdam, The Netherlands), pp. 968–972, April 2002.

[30] Z. Kukorelly and K. Zeger, "New bianry fix-free codes with Kraft sum 3/4," in *IEEE International Symposium on Infomation Theory*, (Lausanne, Switzerland), p. 178, June 2002.

[31] S. Yekhanin, "Sufficient conditions for existence of fix-free codes," in *IEEE International Symposium on Information Theory*, (Washington, USA), p. 284, June 2001.

[32] C. Ye and R. W. Yeung, "Some basic properties of fix-free codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 72–87, January 2001.

[33] V. Buttigieg and P. G. Farrell, "On variable-length error-correcting codes," in *IEEE International Symposium on Information Theory (ISIT'94)*, (Trondheim, Norway), p. 507, June-July 1994.

[34] V. Buttigieg and P. G. Farrell, "Variable-length error-correcting codes," *IEE Proc.-Commun.*, vol. 147, pp. 211–215, August 2000.

[35] V. Buttigieg, *Variable-length error-correcting codes*. PhD thesis, University of Manchester, Department of Electrical Engineering, 1995.

[36] C. Lamy and J. Paccaut, "Optimised constructions for variable-length error correcting codes," in *Proceedings of IEEE Information Theory Workshop (ITW2003)*, (Paris, France), pp. 183–186, March 31-April 4 2003.

[37] M. Schindler, "Practical Huffman Coding. [Online]." Available: http://www.compressconsult.com/huffman/.

[38] C. Fogg, "Survey of software and hardware VLC architectures," in *SPIE Proceedings on Image and Video Compression Conference*, pp. 29–37, May 1994.

[39] P. C. Tseng and Y. C. Chang and Y. W. Huang and H. C. Fang and C. T. Huang and L. G. Chen, "Advances in hardware architectures for image and video coding-A survey," *Proceedings of IEEE*, vol. 93, pp. 184–197, January 2005.

[40] M. Park and D. J. Miller, "Decoding entropy-coded symbols over noisy channels by MAP sequence estimation for asynchronous HMMs," in *Proceedings of Conference on Information Sciences and Systems*, (Princeton, NJ, USA), pp. 477–482, March 1998.

[41] M. Park and D. J. Miller, "Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," in *Proceedings of IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP'99)*, (Phoenix, Arizona, USA), pp. 2451–2454, March 1999.

[42] N. Demir and K. Sayood, "Joint source/channel coding for variable length codes," in *Proceedings of Data Compression Conference (DCC'98)*, (Snowbird, UT, USA), pp. 139–148, March-April 1998.

[43] D. J. Miller and M. Park, "A sequence-based approximate MMSE decoder for source coding over noisy channels using discrete hidden Markov models," *IEEE Transactions on Communications*, vol. 46, no. 2, pp. 222–231, 1998.

[44] R. Bauer and J. Hagenauer, "Symbol by Symbol MAP Decoding of Variable Length Codes," in *Proceedings of 3rd International ITG Conference on Source and Channel Coding*, (Munich, Germany), pp. 111–116, January 2000.

[45] V. B. Balakirsky, "Joint Source-Channel Coding with Variable Length Codes," in *Proceedings of 1997 IEEE International Symposium on Information Theory(ISIT'97)*, (Ulm Germany), p. 419, July 1997.

[46] R. Bauer and J. Hagenauer, "On Variable Length Codes for Iterative Source/Channel Decoding," in *Proceedings of IEEE Data Compression Conference (DCC)*, (Snowbird, USA), pp. 273–282, April 2001.

[47] J. Kliewer and R. Thobaben, "Combining FEC and optimal soft-input source decoding for the reliable transmission of correlated variable-length encoded signals," in *Proceedings of CoIEEE Data Compression Conference*, (Snowbird, UT, USA), pp. 83–91, April 2002.

[48] R. Thobaben and J. Kliewer, "Robust decoding of variable-length encoded Markov sources using a three-dimensional trellis," *IEEE Communications Letters*, vol. 7, pp. 320–322, July 2003.

[49] R. Thobaben and J. Kliewer, "Low complexity iterative joint source-channel decoding for variable-length encoded Markov sources," *IEEE Transactions on Communications*, vol. 53, pp. 2054–2064, December 2005.

[50] A. Guyader and E. Fabre and C. Guillemot and M. Robert, "Joint source-channel turbo decoding of entropy-coded sources," *IEEE Journal of Selected Areas in Communications*, vol. 19, pp. 1680–1696, September 2001.

[51] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.

[52] L. R. Bahl and J. Cocke and F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimal symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, Mar. 1974.

[53] M. Bystrom and S. Kaiser and A. Kopansky, "Soft source decoding with applications," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 11, pp. 1108–1120, October 2001.

[54] L. Perros-Meilhac and C. Lamy, "Huffman tree-based metric derivation for a low-complexity sequential soft VLC decoding," in *Proceedings of IEEE International Conference on Communications (ICC'02)*, (New York,NY,USA), pp. 783–787, April-May 2002.

[55] P. Sweeney, *Error Control Coding: From Theory to Practice.* John Wiley & Sons, 2002.

[56] A. H. Murad and T. E. Fuja, "Joint source-channel decoding of variable-length encoded sources," in *Proceedings of Information Theory Workshop (ITW'98)*, (Killarney, Ireland), pp. 94–95, June 1998.

[57] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proceedings of the 1st International Symposium on Turbo Codes and Related Topics*, (Brest, France), pp. 1–12, 1997.

[58] N. Gortz, "On the iterative approximation of optimal joint source-channel decoding," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 1662–1670, September 2001.

[59] J. Wen and J. D. Villasenor, "Utilizing soft information in decoding of variable length codes," in *Data Compression Conference (DCC'99)* , (Snowbird, USA), pp. 131–139, 1999.

[60] M. Adrat and U. von Agris and P. Vary, "Convergence behavior of iterative source-channel decoding," in *IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP'03)*, (Hong Kong, China), pp. 269–272, April 2003.

[61] J. Kliewer and R. Thobaben, "Parallel concatenated joint source-channel coding," *Electronics Letters*, vol. 39, no. 23, pp. 1664–1666, 2003.

[62] J. Hagenauer, "Source-controlled channel decoding," *IEEE Transactions on Communications*, vol. 43, pp. 2449–2457, September 1995.

[63] R. Steele and L. Hanzo, eds., *Mobile Radio Communications: Second and Third Generation Cellular and WATM Systems.* New York, USA: IEEE Press - John Wiley & Sons, 2nd ed., 1999.

[64] J. Garcia-Frias and J. D. Villasenor, "Joint turbo decoding and estimation of hidden Markov sources," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 1671–1679, September 2001.

[65] G.-C. Zhu and F. Alajaji and J. Bajcsy and P. Mitran, "Non-systematic turbo codes for non-uniform i.i.d. sources over AWGN channels," in *Proceedings of Conference on Information Sciences and Systems (CISS'02)*, (Princeton, NJ, USA), March 2002.

[66] A. Elbaz and R. Pyndiah and B. Solaiman and O. Ait Sab, "Iterative decoding of product codes with a priori information over a Gaussian channel for still image transmission," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'99)*, (Rio de Janeireo, Brazil), December 1999.

[67] Z. Peng and Y.-F. Huang and D. J. Costello, "Turbo codes for image transmission-a joint channel and source decoding approach," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 868–879, 2000.

[68] L. Guivarch and J.-C. Carlach and P. Siohan, "Joint source-channel soft decoding of Huffman codes with turbo-codes," in *Proceedings of IEEE Data Compression Conference (DCC'00)*, (Snowbird, Utah, USA), pp. 8–92, March 2000.

[69] M. Jeanne and J.-C. Carlach and P. Siohan, "Joint source-channel decoding of variable-length codes for convolutional codes and turbo codes," *IEEE Transactions on Communications*, vol. 53, no. 1, pp. 10–15, 2005.

[70] L. Guivarch and J.-C. Carlach and P. Siohan, "Low complexity soft decoding of Huffman encoded Markov sources using turbo-codes," in *Proceedings of IEEE 7th International Conference on Telecommunications (ICT'00)*, (Acapulco, Mexico), pp. 872–876, May 2000.

[71] X. Jaspar and L. Vandendorpe, "Three SISO modules joint source-channel turbo decoding of variable length coded images," in *Proceedings of 5th International ITG Conference on Source and Channel Coding (SCC'04)*, (Erlangen, Germany), pp. 279–286, January 2004.

[72] N. Phamdo and N. Farvardin, "Optimal detection of discrete Markov sources over discrete memoryless channels - applications to combined source-channel coding," *IEEE Transactions on Information Theory*, vol. 40, no. 1, pp. 186–193, 1994.

[73] C. Berrou and A. Glavieux and P. Thitimajshima, "Near-Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proceedings of IEEE International Conference on Communications (ICC'93)*, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[74] L. Hanzo and T. H. Liew and B. L. Yeap, *Turbo coding, turbo equalisation and space-time coding for transmission over fading channels.* IEEE Press, 2002.

[75] S. Benedetto and D. Divsalar and G. Montorsi and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *IEEE Transactions on Information Theory*, vol. 44, pp. 909–926, May 1998.

[76] J. Hagenauer and E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, Mar. 1996.

[77] R. Pyndiah, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE Transactions on Communications*, vol. 46, pp. 1003–1010, August 1998.

[78] P. Sweeney and S. Wesemeyer, "Iterative soft-decision decoding of linear block codes," *IEE Proceedings of Communications*, vol. 147, pp. 133–136, June 2000.

[79] D. Divsalar and H. Jin and R. McEliece, "Coding theorems for 'turbo-like' codes," in *Proceedings of 36th Allerton Conference on Communication, Control and Computing*, (Allerton, Illinois, USA), pp. 201–210, September 1998.

[80] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of Low Density Parity Check Codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[81] T. Richardson and R. Urbanke, "The capacity of LDPC codes under message passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 595–618, February 2001.

[82] T. Richardson and M. A. Shokrollahi and R. Urbanke, "Design of capacity approaching low density parity-check codes," *IEEE Transactions on Information Theory* , vol. 47, pp. 619–637, February 2001.

[83] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[84] M. Tüchler and J. Hagenauer, "EXIT charts of irregular codes," in *Proceedings of Conference on Information Science and Systems [CDROM]*, (Princeton University), 20-22 March 2002.

[85] M. Tüchler, "Design of serially concatenated systems depending on the blocklength," *IEEE Transactions on Communications*, vol. 52, pp. 209–218, February 2004.

[86] C. Douillard and M. Jezequel and C. Berrou and A. Picart and P. Didier and A. Glavieux, "Iterative correction of intersymbol interference: Turbo equalization," *European Transactions on Telecommunications*, vol. 6, pp. 507–511, Sept.-Oct 1995.

[87] X. Li and J. A. Ritcey, "Bit-interleaved coded modulation with iterative decoding," in *Proceedings of International Conference on Communications*, (Vancouver, Canada), pp. 858–863, Juen 1999.

[88] S. ten Brink and J. Speidel and R. Yan, "Iterative demapping and decoding for multilevel modulation," in *Proceedings of IEEE Global Telecommunications Conference*, pp. 858–863, 1998.

[89] S. Benedetto and D. Divsalar and G. Montorsi and F. Pollara, "Serial concatenated trellis coded modulation with iterative decoding," in *Proceedings of IEEE International Symposium on Information Theory*, p. 8, 29 June-4 July 1997.

[90] D. Divsalar and S. Dolinar and F. Pollara, "Serial turbo trellis coded modulation with rate-1 inner code," in *Proceedings of International Symposium on Information Theory*, (Sorrento, Italy), p. 194, June 2000.

[91] M. Moher, "An iterative multiuser decoder for near-capacity communications," *IEEE Transactions on Communications*, vol. 46, pp. 870–880, July 1998.

[92] M. C. Reed and C. B. Schlegel and P. Alexander and J. Asenstorfer, "Iterative multiuser detection for DS-CDMA with FEC: Near single user performance," *IEEE Transactions on Communications*, vol. 46, pp. 1693–1699, July 1998.

[93] A. Nayak, J. Barry, and S. McLaughlin, "Joint timing recovery and turbo equalization for coded partial response channels," *IEEE Transactions on Magnetics*, vol. 38, September 2002.

[94] X. Wang and H. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Transactions on Communications*, vol. 47, pp. 1046–1061, July 1999.

[95] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, Calif, USA: Morgan Kaufmann Publishers, 1988.

[96] R. J. McEliece and D. J. C. MacKay and J.-F. Cheng, "Turbo decoding as an instance of pearl's "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, 1998.

[97] F. R. Kschischang and B. J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[98] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, pp. 28–41, January 2004.

[99] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 219–230, 1998.

[100] E. Biglieri, "Receivers for Coded Signals: A Unified View Based on Factor Graph," in *Proceedings of the Fifth International Conference on Information, Communications and Signal Processing*, (Bangkok, Thailand), p. 97, 06-09 December 2005.

[101] J. Hagenauer, "The EXIT Chart - Introduction To Extrinsic Information Transfer In Iterative Processing," in *Proceedings of 12th European Signal Processing Conference (EUSIPCO)*, pp. 1541–1548, September 2004.

[102] S. ten Brink, "Convergence behaviour of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications* , pp. 1727–1737, Oct. 2001.

[103] 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, Mar. 1996.

[104] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *Electronics Letters*, vol. 32, pp. 445–446, February 1996.

[105] L. C. Perez and J. Seghers and D. J. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1698–1709, November 1996.

[106] T. Duman and R. Salehi, "New performance bounds for turbo codes," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'97)*, (Phoenix, Arizona, USA), pp. 634–638, November 1997.

[107] S. Chung and T. Richardson and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Transactions on Information Theory* , vol. 47, pp. 657–670, February 2001.

[108] D. Divsalar and S. Dolinar and F. Pollara, "Low complexity turbo-like codes," in *Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics*, (Brest, France), pp. 73–80, September 2000.

[109] K. R. Narayanan, "Effect of precoding on the convergence of turbo equalization for partial response channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 686–698, April 2001.

[110] S. ten Brink, "Convergence of iterative decoding," *Electronics Letters*, vol. 35, pp. 806–808, 1999.

[111] M. Tüchler and S. ten Brink and J. Hagenauer, "Measures for tracing convergence of iterative decoding algorithms," in *Proceedings of 4th IEEE/ITG Conference on Source and Channel Coding*, (Berlin, Germany), pp. 53–60, January 2002.

[112] S. Kullback, *Information theory and statistics.* Dover Publications Inc., New York, 1968.

[113] Q. Luo and P. Sweeney, "Method to analyse convergence of turbo codes," *Electronics Letters*, vol. 41, pp. 757- 758, June 2005.

[114] M. Moher and T. A. Gulliver., "Cross-entropy and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, pp. 3097–3104, November 1998.

[115] S. ten Brink, "Iterative decoding trajectories of parallel concatenated codes," in *Proceedings of the 3rd IEEE/ITG Conference on Source and Channel Coding*, (Munich, Germany), pp. 75–80, January 2000.

[116] S. ten Brink, "Design of serially concatenated codes based on iterative decoding convergence," in *Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics*, (Brest, France), pp. 319–322, September 2000.

[117] M. Tüchler and R. Koetter and A. C. Singer, "Turbo equalization: Principles and new results," *IEEE Transactions on Communications*, vol. 50, pp. 754–767, May 2002.

[118] F. Schreckenbach and G. Bauch, "Measures for tracing convergence of iterative decoding algorithms," in *Proceedings of 12th European Signal Processing Conference (EUSIPCO)*, (Vienna, Austria), pp. 1557–1560, September 2004.

[119] S. ten Brink, "Designing iterative decoding schemes with the extrinsic information transfer chart," *AEÜ International Journal of Electronics and Communications*, vol. 54, pp. 389–398, November 2000.

[120] A. Ashikhmin and G. Kramer and S. ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *IEEE Transactions on Information Theory*, vol. 50, pp. 2657– 2673, November 2004.

[121] S. ten Brink and G. Kramer and A. Ashikhmin, "Design of Low-Density Parity-Check Codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, pp. 670–678, April 2004.

[122] S. ten Brink and G. Kramer, "Design of repeat-accumulate codes for iterative detection and decoding," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2764–2772, November 2003.

[123] S. ten Brink, "Convergence of multi-dimensional iterative decoding schemes," in *Proceedings of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, CA), pp. 270–274, October/November 2001.

[124] M. Tüchler, "Convergence prediction for iterative decoding of threefold concatenated systems," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'02)*, (Taipei, China), pp. 1358–1362, 17-21, November 2002.

[125] F. Brännström and L. K. Rasmussen and A. J. Grant, "Optimal scheduling for multiple serially concatenated codes," in *International Symposium on Turbo Codes and Related Topics*, (Brest, France), pp. 383–386, September 2003.

[126] F. Brännström and L. K. Rasmussen and A. J. Grant, "Convergence analysis and optimal scheduling for multiple concatenated codes," *IEEE Transactions on Information Theory*, vol. 51, pp. 3354–3364, September 2005.

[127] A. Grant, "Convergence of non-binary iterative decoding," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'01)*, (San Antonio TX, USA), pp. 668–670, November 2001.

[128] J. Kliewer and S. X. Ng and L. Hanzo, "On the computation of EXIT characteristics for symbol-based iterative decoding," in *Proceedings of the 4th International Symposium on Turbo Codes (ISTC'06) in connection with the 6th International ITG-Conference on Source and Channel Coding*, (Munich, Germany), 3-7 April 2006.

[129] G. D. Forney, *Concatenated codes.* Cambridge (Mass., USA): MIT Press, 1966.

[130] D. Divsalar and F. Pollara, "Multiple turbo codes for deep space communications," TDA Progress Report 42-121, Jet Propulsion Laboratory, May 1995.

[131] S. Benedetto and G. Montorsi, "Serial concatenation of block and convolutional codes," *Electronics Letters*, vol. 32, pp. 887–888, May 1996.

[132] S. Benedetto and G. Montorsi, "Iterative decoding of serially concatenated convolutional codes," *Electronics Letters*, vol. 32, pp. 1186–1187, June 1996.

[133] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Transactions on Information Theory*, vol. IT-28, pp. 55–67, 1982.

[134] S. Benedetto and D. Divsalar and G. Montorsi and F. Pollara, "Analysis, design, and iterative decoding of double serially concatenated codes with interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 231–244, February 1998.

[135] M. Tüchler and A. C. Singer and R. Koetter, "Minimum mean squared error equalization using a priori information," *IEEE Transactions on Signal Processing*, vol. 50, pp. 673–682, March 2002.

[136] I. Land and Peter A. Hoeher and S. Gligorevic, "Computation of symbol-wise mutual information in transmission systems with LogAPP decoders and application to EXIT charts," in *Proceedings of Internatioanl ITG Conference on Source and Channel Coding*, (Erlangen, Germany), pp. 195–202, 2004.

[137] O. Alamri and J. Wang and S. X. Ng and L.-L. Yang and L. Hanzo, "Near-Capacity Three-Stage Turbo Detection of Irregular Convolutional Coded Joint Sphere-Packing Modulation and Space-Time Coding," *submitted to IEEE Transactions on Communications*, 2006.

[138] A. G. Lillie and A. R. Nix and J. P. McGeehan, "Performance and design of a reduced complexity iterative equalizer for precoded ISI channels," in *IEEE Vehicular Technology Conference*, (Orlando, USA), pp. 299–303, October 2003.

[139] B. McMillan, "Two inequalities implied by unique decipherability," *IRE Transactions on Information Theory*, vol. IT-2, pp. 115–116, Dec. 1956.

[140] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 2003.

[141] L. G. Kraft, "A device for quantizing, grouping, and coding of amplitude modulated pulses," Master's thesis, Electrical Engineering Department, M.I.T., Mar. 1949.

[142] E. S. Schwartz and B. Kallick, "Generating a Canonical Prefix Encoding," *Communications of ACM*, vol. 7, pp. 166–169, Mar. 1964.

[143] C. Lamy and F. Bergot, "Lower bounds on the existence of binary error-correcting variable-length codes," in *Proceedings of IEEE Information Theory Workshop*, (Paris, France), pp. 300 – 303, 31 March - 4 April 2003.

[144] W.E. Hartnett, ed., *Foundations of coding theory*. Dordrecht, Holland: D. Reidling Publishing Co., 1974.

[145] E. J. Dunscombe, *Some applications of mathematics to coding theory*. PhD thesis, Royal Holloway and Bedford New College, University of London, 1988.

[146] M. A. Bernard and B. D. Sharma, "Variable length perfect codes," *Journal of Information & Optimization Science*, vol. 13, pp. 143–151, January 1992.

[147] A. Escott, "A new performance measure for a class of two-length error correcting codes," in *Codes and cyphers: Cryptography and coding IV* (P.G.Farrell, ed.), pp. 167–181, Essex, England: Institute of Mathematics and its Applications, 1995.

[148] "Information Technology- Digital Compression and Coding of Continuous-Tone Still Images-Requirements and Guidelines," *ITU-T T.81 (ISO/IEC 10918-1)*, 1992.

[149] "Information Technology - JPEG2000," *ISO/IEC 15444-1 (ITU-T T.800*, 2000.

[150] "Draft Text for Joint Video Specification," *ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC*, 2002.

[151] "Coding of moving pictures and associated audio for digital storage media at up to 1.5 Mbps," *ISO/IEC 11172 MPEG-1*, 1993.

[152] "Generic coding of moving pictures and associated audio information," *ISO/IEC 13818 MPEG-2*, 1994.

[153] A. J. Viterbi, "Wireless Digital Communications:A View Based on Three Lessons Learnt," *IEEE Communications Magazine*, pp. 33–36, September 1991.

[154] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, March 1973.

[155] W. Xiang and S. Pietrobon and S. Barbulescu, "Iterative decoding of JPEG coded images with channel coding," in *IEEE International Conference on Telecommunications*, (Tahiti, French), pp. 1356–1360, Feb. 2003.

[156] P. Robertson and E. Villebrun and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings of IEEE International Conference on Communications*, (Seattle, Washington), pp. 1009–1013, June 1995.

[157] T. Okuda and E. Tanaka and T. Kasai, "A Method for the Correction of Garbled Words Based on the Levenshtein Metric," *IEEE Transactions on Computers*, vol. C-25, pp. 172–176, February 1976.

[158] M. Adrat and P. Vary, "Iterative Source-Channel Decoding: Improved System Design Using EXIT Charts," *EURASIP Journal on Applied Signal Processing (Special Issue: TURBO Processing)*, pp. 928–941, May 2005.

[159] D. Divsalar and F. Pollara, "Hybrid concatenated codes and iterative decoding," TDA Progress Report 42-130, Jet Propulsion Laboratory, August 1997.

[160] S. Benedetto and D. Divsalar and G. Montorsi and F. Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes," *IEEE Communications Letters*, vol. 1, pp. 22–24, January 1997.

[161] S. Benedetto and D. Divsalar and G. Montorsi and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Transactions on Telecommunications*, vol. 9, pp. 155–172, March 1998.

[162] S. Huettinger and J. Huber, "Design of multiple turbo codes with transfer characteristics of component codes," in *Proceedings of Conference on Information Sciences and Systems*, March 2002.

[163] D. Raphaeli and Y. Zarai, "Combined Turbo Equalization and Turbo Decoding," *IEEE Communications Letters*, pp. 107–109, April 1998.

[164] W. E. Ryan and L. L. McPheters and S. W. McLaughlin, "Combined turbo coding and turbo equalization for PR4-equalized Lorentzian channels," in *Proceedings of Conference on Information Sciences and Systems*, pp. 489–493, March 1998.

[165] J. Kliewer and S. X. Ng and L. Hanzo, "Efficient Computation of EXIT Functions for Non-Binary Iterative Decoding," *to appear in IEEE Transactions on Communications (Letter), 2006.*

[166] L. Hanzo and C.H. Wong andM.S. Yee, *Adaptive Wireless Transceivers: Turbo-Coded, Space-Time Coded TDMA, CDMA and OFDM Systems.* Chichester, UK:John Wiley-IEEE Press, 2002.

[167] A. Glavieux and C. Laot and J. Labat, "Turbo equalization over a frequency selective channel," in *Proceedings of International Symposium on Turbo Codes*, (Brest, France), pp. 96–102, 1997.

[168] D. Raphaeli, "Combined turbo equalization and turbo decoding," *IEEE Communications Letters*, vol. 2, pp. 107–109, April 1998.

[169] I. Lee, "The effect of precoder on serially concatenated coding systems with an ISI channel," *IEEE Transactions on Communications*, vol. 49, pp. 1168–1175, July 2001.

[170] W. Ryan, "Performance of high rate turbo codes on a PR4 equalized magnetic recording channel," in *IEEE International Conference on Communications (ICC)*, (Atlanta, GA, USA), pp. 947–951, June 1998.

[171] T. Souvignier and A. Friedman and M. Oberg and P. H. Siegel and R. E. Swanson and J. Wolf, "Turbo decoding for PR4: Parallel versus serial concatenation," in *IEEE International Conference on Communications (ICC)*, (Vancouver, Canada), pp. 1638–1642, June 1999.

[172] D. N. Doan and K. R. Narayanan, "Some new results on the design of codes for inter-symbol interference channels based on convergence of turbo equalization," in *IEEE International Conference on Communications (ICC'02)*, (Anchorage Alaska, USA), pp. 1873–1877, 2002.

[173] H. Jin and A. Khandekar and R. McEliece, "Irregular repeat-accumulate codes," in *Proceedings of 2nd International Symposium on Turbo Codes and Related Topics*, (Brest, France), pp. 1–8, September 2000.

[174] S. X. Ng and J. Wang and M. Tao and L.-L. Yang and L. Hanzo, "Iteratively decoded variable-length space-time coded modulation: code construction and convergence analysis," *to appear in IEEE Transactions on Wireless Communications.*

[175] S. X. Ng and J. Y. Chung and L. Hanzo, "Turbo-Detected Unequal Protection MPEG-4 Wireless Video Telephony using Multi Level Coding, Trellis Coded Modulation and Space-Time Trellis Coding," *IEE Proceedings on Communications*, vol. 152, pp. 1116–1124, December 2005.

[176] R. G. Maunder and J. Wang and S. X. Ng and L.-L. Yang and L. Hanzo, "Iteratively Decoded Irregular Variable Length Coding and Trellis Coded Modulation," in *submitted to IEEE International Conference on Communications (ICC'07)*, (Glasgow, Scotland), June 2007.

Subject Index

Author Index