University of Southampton Research Repository

ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

http://eprints.soton.ac.uk

IRREGULAR VARIABLE LENGTH CODING

By
Robert G. Maunder
B.Eng.(Hons)

A thesis submitted for the degree of
Doctor of Philosophy

School of Electronics and Computer Science,
University of Southampton,
United Kingdom.

December 2007

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS

SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Irregular Variable Length Coding

by Robert G. Maunder

In this thesis, we introduce Irregular Variable Length Coding (IrVLC) and investigate its applications, characteristics and performance in the context of digital multimedia broadcast telecommunications. During IrVLC encoding, the multimedia signal is represented using a sequence of concatenated binary codewords. These are selected from a codebook, comprising a number of codewords, which, in turn, comprise various numbers of bits. However, during IrVLC encoding, the multimedia signal is decomposed into particular fractions, each of which is represented using a different codebook. This is in contrast to regular Variable Length Coding (VLC), in which the entire multimedia signal is encoded using the same codebook.

The application of IrVLCs to joint source and channel coding is investigated in the context of a video transmission scheme. Our novel video codec represents the video signal using tessellations of Variable-Dimension Vector Quantisation (VDVQ) tiles. These are selected from a codebook, comprising a number of tiles having various dimensions. The selected tessellation of VDVQ tiles is signalled using a corresponding sequence of concatenated codewords from a Variable Length Error Correction (VLEC) codebook. This VLEC codebook represents a specific joint source and channel coding case of VLCs, which facilitates both compression and error correction. However, during video encoding, only particular combinations of the VDVQ tiles will perfectly tessellate, owing to their various dimensions. As a result, only particular sub-sets of the VDVQ codebook and, hence, of the VLEC codebook may be employed to convey particular fractions of the video signal. Therefore, our novel video codec can be said to employ IrVLCs.

The employment of IrVLCs to facilitate Unequal Error Protection (UEP) is also demonstrated. This may be applied when various fractions of the source signal have different error sensitivities, as is typical in audio, speech, image and video signals, for example. Here, different VLEC codebooks having appropriately selected error correction capabilities may be employed to encode the particular fractions of the source signal. This approach may be expected to yield a higher reconstruction quality than equal protection in cases where the various fractions of the source signal have different error sensitivities.

Finally, this thesis investigates the application of IrVLCs to near-capacity operation using EXtrinsic Information Transfer (EXIT) chart analysis. Here, a number of component VLEC codebooks having different inverted EXIT functions are employed to encode particular fractions of the source symbol frame. We show that the composite inverted IrVLC EXIT function may be obtained as a weighted average of the inverted component VLC EXIT functions. Additionally, EXIT chart matching is employed to shape the inverted IrVLC EXIT function to match the EXIT function of a serially concatenated inner channel code, creating a narrow but still open EXIT chart tunnel. In this way, iterative decoding convergence to an infinitesimally low probability of error is facilitated at near-capacity channel SNRs.

Contents

Acknowledgements

I wish to thank my supervisor Prof. Lajos Hanzo for his invaluable guidance, support and friendship, without which I could never have overcome disaster. The financial support of the EPSRC, Swindon UK and the EU under the auspices of the NEWCOM, NEXWAY and Phoenix projects is also gratefully acknowledged. This thesis is dedicated to the memory of Simon Wilson, 1981–2006.

Publications

Journal papers

1. **R. G. Maunder** and L. Hanzo, "Near-capacity irregular variable length coding and irregular unity rate coding," *IEEE Transactions on Wireless Communications*, vol. 8, no. 11, pp. 5500–5507, November 2009.
   Available: http://eprints.ecs.soton.ac.uk/14471/
2. **R. G. Maunder** and L. Hanzo, "Genetic algorithm aided design of component codes for irregular variable length coding," *IEEE Transactions on Communications*, vol. 57, no. 5, pp. 1290–1297, May 2009.
   Available: http://eprints.ecs.soton.ac.uk/14470/
3. **R. G. Maunder**, J. Wang, S. X. Ng, L.-L. Yang and L. Hanzo, "On the performance and complexity of irregular variable length codes for near-capacity joint source and channel coding," *IEEE Transactions on Wireless Communications*, vol. 7, no. 4, pp. 1338–1347, April 2008.
   Available: http://eprints.ecs.soton.ac.uk/14467/
4. **R. G. Maunder**, J. Kliewer, S. X. Ng, J. Wang, L.-L. Yang and L. Hanzo, "Joint iterative decoding of trellis-based VQ and TCM," *IEEE Transactions on Wireless Communications*, vol. 6, no. 4 pp. 1327–1336, April 2007.
   Available: http://eprints.ecs.soton.ac.uk/14077/

Conference papers

1. **R. G. Maunder** and L. Hanzo, "Concatenated irregular variable length coding and irregular unity rate coding," in *Proceedings of the IEEE Vehicular Technology Conference*, Barcelona, Spain, April 2009.
   Available: http://eprints.ecs.soton.ac.uk/14472/
2. R. Y. Tee, **R. G. Maunder**, J. Wang and L. Hanzo, "Near-capacity irregular bit-interleaved coded modulation," in *Proceedings of the IEEE Vehicular Technology*

Conference, Marina Bay, Singapore, May 2008, pp. 549–553.
Available: http://eprints.ecs.soton.ac.uk/14745/

3. C. Xu, L.-L. Yang, **R. G. Maunder** and L. Hanzo, "Near-optimum soft-output ant-colony-optimization based multiuser detection for the DS-CDMA uplink," in *Proceedings of the IEEE International Conference on Communications*, Beijing, China, May 2008, pp. 795–799.
Available: http://eprints.ecs.soton.ac.uk/14744/
4. **R. G. Maunder** and L. Hanzo, "Genetic algorithm aided design of near-capacity irregular variable length codes," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, Las Vegas, NV, USA, March 2008, pp. 1256–1260.
Available: http://eprints.ecs.soton.ac.uk/14589/
5. S. Ahmed, **R. G. Maunder**, L.-L. Yang and L. Hanzo, "Iterative detection of three-stage concatenated FFH-MFSK," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, Las Vegas, NV, USA, March 2008, pp. 906–911.
Available: http://eprints.ecs.soton.ac.uk/14743/
6. **R. G. Maunder**, J. Wang, S. X. Ng, L.-L. Yang and L. Hanzo, "Iteratively decoded irregular variable length coding and trellis coded modulation," in *Proceedings of the IEEE Workshop on Signal Processing Systems*, Shanghai, China, October 2007, pp. 222–227.
Available: http://eprints.ecs.soton.ac.uk/13907/
7. M. El-Hajjar, **R. G. Maunder**, O. Alamri, S. X. Ng and L. Hanzo, "Iteratively decoded irregular variable length coding and sphere-packing modulation-aided differential space-time spreading," in *Proceedings of the IEEE Vehicular Technology Conference*, Baltimore, MD, USA, September 2007, pp. 1238–1242.
Available: http://eprints.ecs.soton.ac.uk/14469/
8. S. Ahmed, **R. G. Maunder**, L.-L. Yang, S. X. Ng and L. Hanzo, "Joint source coding, unity rate precoding and FFH-MFSK modulation using iteratively decoded irregular variable length coding," in *Proceedings of the IEEE Vehicular Technology Conference*, Baltimore, MD, USA, September 2007, pp. 1042–1046.
Available: http://eprints.ecs.soton.ac.uk/14468/
9. L.-L. Yang, J. Wang, **R. G. Maunder** and L. Hanzo, "Iterative equalisation and source decoding for vector quantized sources," in *Proceedings of the IEEE*

Vehicular Technology Conference, Melbourne, Australia, May 2006, pp. 2349–2353.
Available: http://eprints.ecs.soton.ac.uk/12633/

10. **R. G. 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 IEEE Vehicular Technology Conference*, vol. 2, Dallas, TX, USA, September 2005, pp. 922–926.
Available: http://eprints.ecs.soton.ac.uk/11669/
11. 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," in *Proceedings of the European Signal Processing Conference*, Vienna, Austria, September 2004, pp. 2231–2234.
Available: http://eprints.ecs.soton.ac.uk/11586/
12. B. L. Yeap, **R. G. Maunder**, S. X. Ng and L. Hanzo, "Turbo detection of space-time trellis-coded constant bit rate vector-quantised videophone system using reversible variable-length codes, convolutional codes and turbo codes," in *Proceedings of the IEEE Vehicular Technology Conference*, vol. 2, Los Angeles, CA, USA, September 2004, pp. 1358–1362.
Available: http://eprints.ecs.soton.ac.uk/11592/

Notation

Schematics

$\mathbf{f}_n$ Current video frame.

$\hat{\mathbf{f}}_n$ Quantised current video frame.

$\tilde{\mathbf{f}}_n$ Reconstructed current video frame.

$\hat{\mathbf{f}}_{n-1}$ Quantised previous video frame.

$\tilde{\mathbf{f}}_{n-1}$ Reconstructed previous video frame.

$\mathbf{e}$ Source sample frame.

$\hat{\mathbf{e}}$ Quantised source sample frame.

$\tilde{\mathbf{e}}$ Reconstructed source sample frame.

$\mathbf{s}$ Source symbol frame.

$\tilde{\mathbf{s}}$ Reconstructed source symbol frame.

$\mathbf{u}$ Transmission frame.

$\hat{\mathbf{u}}$ Received transmission frame.

$\tilde{\mathbf{u}}$ Reconstructed transmission frame.

π Interleaving.

π^{-1} De-interleaving.

$\mathbf{u}'$ Interleaved transmission frame.

$\mathbf{v}$ Encoded frame.

$\mathbf{v}'$ Interleaved encoded frame.

$L_a(\cdot)$ *A priori* Logarithmic Likelihood Ratios (LLRs)/logarithmic *A Posteriori* Probabilities (Log-APPs) pertaining to the specified bits/symbols.

$L_p(\cdot)$ *A posteriori* LLRs/Log-APPs pertaining to the specified bits/symbols.

$L_e(\cdot)$ Extrinsic LLRs pertaining to the specified bits/symbols.

$\mathbf{x}$ Channel's input symbols.

$\mathbf{y}$ Channel's output symbols.

Channel

η Effective throughput.

E_c/N_0 Channel Signal to Noise Ratio (SNR).

E_b/N_0 Channel SNR per bit of source information.

Video blocks (VBs)

J_x^{MB} Number of VB columns in each Macro-Block (MB).

J_y^{MB} Number of VB rows in each MB.

J^{MB} Number of VBs in each MB.

Sub-frames

M Number of sub-frames.

m Sub-frame index.

$\mathbf{e}^m$ Source sample sub-frame.

$\hat{\mathbf{e}}^m$ Quantised source sample sub-frame.

$\tilde{\mathbf{e}}^m$ Reconstructed source sample sub-frame.

$\mathbf{u}^m$ Transmission sub-frame.

$\tilde{\mathbf{u}}^m$ Reconstructed transmission sub-frame.

$\mathbf{s}^m$ Source symbol sub-frame.

$\tilde{\mathbf{s}}^m$ Reconstructed source symbol sub-frame.

Source sample sub-frames

J Number of source samples that are comprised by each source sample sub-frame.

J^{sum} Number of source samples that are comprised by each source sample frame.

j Source sample index.

e_j^m Source sample.

$\hat{e}_j^m$ Quantised source sample.

$\tilde{e}_j^m$ Reconstructed source sample.

Transmission sub-frames

I Number of bits that are comprised by each transmission sub-frame.

I^{sum} Number of bits that are comprised by each transmission frame.

$I_{\min}$ Minimum number of bits that may be comprised by each transmission sub-frame.

$I_{\max}$ Maximum number of bits that may be comprised by each transmission sub-frame.

i Transmission sub-frame bit index.

u_i^m Transmission sub-frame bit.

b Binary value.

Codebooks

K Number of entries in the codebook.

k Codebook entry index.

VQ codebook

$\mathbf{VQ}$ Vector Quantisation (VQ) codebook.

$\mathbf{VQ}^k$ VQ tile.

J^k Number of VBs that are comprised by the VQ tile $\mathbf{VQ}^k$.

j^k VQ tile VB index.

$VQ_{j^k}^k$ VQ tile VB.

VLC codebook

$\mathbf{VLC}$ Variable Length Coding (VLC) codebook.

$\mathbf{VLC}^k$ VLC codeword.

I^k Number of bits that are comprised by the VLC codeword $\mathbf{VLC}^k$.

I_b^k Number of bits in the VLC codeword $\mathbf{VLC}^k$ assuming a value $b \in \{0, 1\}$.

i^k VLC codeword bit index.

$VLC_{i^k}^k$ VLC codeword bit.

VLC codebook parameters

E Entropy.

$L(\mathbf{VLC})$ VLC codebook average codeword length.

$R(\mathbf{VLC})$ VLC coding rate.

$E(\mathbf{VLC})$ VLC-encoded bit entropy.

$T(\mathbf{VLC})$ VLC trellis complexity.

$O^{\text{APP}}(\mathbf{VLC})$ Average number of Add, Compare and Select (ACS) operations performed per source symbol during *A Posteriori* Probability (APP) Soft-In Soft-Out (SISO) VLC decoding.

$O^{\text{MAP}}(\mathbf{VLC})$ Average number of Add, Compare and Select (ACS) operations performed per source symbol during Maximum *A posteriori* Probability (MAP) VLC sequence estimation.

$d_{\text{free}}(\mathbf{VLC})$ VLC codebook free distance.

$d_{b_{\min}}(\mathbf{VLC})$ VLC codebook minimum block distance.

$d_{d_{\min}}(\mathbf{VLC})$ VLC codebook minimum divergence distance.

$d_{c_{\min}}(\mathbf{VLC})$ VLC codebook minimum convergence distance.

$\bar{d}_{\text{free}}(\mathbf{VLC})$ VLC codebook free distance lower bound.

$D(\mathbf{VLC})$ VLC codebook Real-Valued Free Distance Metric (RV-FDM)

Irregular Variable Length Coding (IrVLC)

N Component VLC codebook count.

n Component VLC codebook index.

$\mathbf{u}^n$ Transmission sub-frame.

$\mathbf{s}^n$ Source symbol sub-frame.

C^n Component VLC codebook source symbol frame fraction.

α^n Component VLC codebook transmission frame fraction.

IrVLC codebooks

$\mathbf{VLC}^n$ Component VLC codebook.

$\mathbf{VLC}^{n,k}$ Component VLC codeword.

$I^{n,k}$ Number of bits that are comprised by the component VLC codeword $\mathbf{VLC}^{n,k}$.

$i^{n,k}$ Component VLC codeword bit index.

$VLC^k_{i^{n,k}}$ Component VLC codeword bit.

Irregular Unity Rate Coding (IrURC)

R Component Unity Rate Code (URC) count.

r Component URC index.

$\mathbf{u}'^r$ Interleaved transmission sub-frame.

$\mathbf{v}^r$ Encoded sub-frame.

$\mathbf{URC}^r$ Component URC.

Code parameters

$R_{(\cdot)}$ Coding rate.

$M_{(\cdot)}$ Number of modulation constellation points.

$L_{(\cdot)}$ Coding memory.

EXtrinsic Information Transfer (EXIT) chart

I_a *A priori* mutual information.

I_e Extrinsic mutual information.

y Importance of seeking a reduced computational complexity during EXIT chart matching.

Trellises

$\ddot{i}$ Bit state index.

$\ddot{j}$ Symbol state index.

$\ddot{n}$ Node state index.

$S_{(\ddot{i},\ddot{j})}$ Symbol-based trellis state.

$S_{(\ddot{i},\ddot{n})}$ Bit-based trellis state.

Trellis transitions

T Trellis transition.

k^T Codebook entry index associated with the symbol-based trellis transition T.

b^T Bit value represented by the bit-based trellis transition T.

i^T Index of bit considered by the bit-based trellis transition T.

$\ddot{i}^T$ Bit state index of the trellis state that the symbol-based trellis transition T emerges from.

$\ddot{j}^T$ Symbol state index of the trellis state that the symbol-based trellis transition T emerges from.

$\ddot{n}^T$ Node state index of the trellis state that the bit-based trellis transition T emerges from.

Trellis sets

$\text{en}(u_i^m)$ The set of all trellis transitions that encompasses the transmission sub-frame bit u_i^m.

$\text{en}(u_i^m = b)$ The sub-set of $\text{en}(u_i^m)$ that maps the binary value b to the transmission sub-frame bit u_i^m.

$\text{en}(\hat{e}_j^m)$ The set of all trellis transitions that encompasses the VB $\hat{e}_j^m$.

$\text{en}(\hat{e}_j^m = VQ_{j^k}^k)$ The sub-set of $\text{en}(\hat{e}_j^m)$ that maps the VQ tile $VQ_{j^k}^k$ to the VB $\hat{e}_j^m$.

$\text{fr}(S)$ The set of all transitions that emerge from the trellis state S.

$\text{to}(S)$ The set of all transitions that merge to the trellis state S.

$\text{fr}(T)$ The state that the transition T emerges from.

to(T) The state that the transition T merges to.

nr($\hat{e}_j^m$) The set of all VBs that immediately surround the VB $\hat{e}_j^m$.

Viterbi algorithm

$d(T)$ The distortion of the trellis transition T.

$D(T)$ The minimum cumulative distortion of all trellis paths between the trellis state $S_{(0,0)}$ and the trellis transition T.

$D(S)$ The minimum cumulative distortion of all trellis paths to the state S.

$m(T)$ The Viterbi algorithm metric of the trellis transition T.

$M(T)$ The maximum cumulative Viterbi algorithm metric of all trellis paths between the trellis state $S_{(0,0)}$ and the trellis transition T.

$M(S)$ The maximum cumulative Viterbi algorithm metric of all trellis paths to the state S.

Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm

$P_a(u_i^m = b)$ *A priori* probability of the transmission sub-frame bit u_i^m taking the value b.

$P(k)$ Probability of occurrence of the codebook entry with index k.

$P(S)$ Probability of occurrence of the trellis state S.

$P(T|\text{fr}(T))$ Conditional probability of the occurrence of the trellis transition T given the occurrence of the trellis state that it emerges from.

$P_p(T)$ *A posteriori* trellis transition probability.

C_1 *A posteriori* trellis transition probability normalisation factor.

$\gamma(T)$ *A priori* trellis transition probability.

$\gamma'(T)$ Weighted *a priori* trellis transition probability.

$C_2(S)$ *A priori* trellis transition probability normalisation factor used for all trellis transitions that emerge from the trellis state S.

$\alpha(S)$ Alpha value obtained for the trellis state S.

$\beta(S)$ Beta value obtained for the trellis state S.

C_{L_a} BCJR algorithm LLR pruning threshold.

C_γ BCJR algorithm *a priori* probability pruning threshold.

C_α BCJR algorithm forward recursion pruning threshold.

C_β BCJR algorithm backwards recursion pruning threshold.

Genetic Algorithm (GA) for VLC codebook design

$\mathbf{L}$	List of candidate VLC codebooks.
L^{tar}	Target GA list length.
$M(\mathbf{VLC})$	GA VLC quality metric.
D^{lim}	GA VLC RV-FDM limit.
R^{lim}	GA VLC coding rate limit.
α^D	GA VLC RV-FDM importance.
α^R	GA VLC coding rate importance.
α^E	GA VLC bit entropy importance.
α^T	GA VLC trellis complexity importance.
β^D	GA VLC RV-FDM increase/decrease constant.
β^R	GA VLC coding rate increase/decrease constant.
D^{best}	Most desirable RV-FDM of VLC codebooks admitted to the GA list.
R^{best}	Most desirable coding rate of VLC codebooks admitted to the GA list.
E^{best}	Most desirable bit entropy of VLC codebooks admitted to the GA list.
T^{best}	Most desirable trellis complexity of VLC codebooks admitted to the GA list.
P^{max}	Maximum number of GA mutations.

Chapter 1

Introduction

In this thesis, we introduce the novel concept of Irregular Variable Length Coding (IrVLC) and investigate its applications, characteristics and performance in the context of digital multimedia broadcast telecommunications. More specifically, we consider the employment of IrVLCs to represent or encode multimedia source signals, facilitating their transmission to, and reconstruction by, remote receivers. Explicitly, IrVLCs are defined as encoding schemes, which represent particular components or segments of the source signals with different sets of codewords, which comprise various numbers of bits. This may be contrasted to regular Variable Length Codes (VLCs), which encode all components or segments of the source signals using the same codebook of codewords.

In Section 1.1, we commence by discussing the applications of IrVLCs that are considered in this thesis. Following this, in Sections 1.2, 1.3 and 1.4 we provide detailed background on the topics of VLCs, the turbo principle and irregular coding, respectively. This material will assist our dicussions throughout the thesis, in addition to setting the stage for outlining its novel contributions. These novel contributions are then described in detail in Section 1.5 and are discussed with reference to the corresponding peer-reviewed publications. Furthermore, Section 1.5 also outlines the structure of the thesis.

1.1 Applications of irregular variable length coding

Three particular aspects of IrVLCs designed for digital multimedia telecommunications are investigated in this thesis, namely near-capacity operation, joint source and channel coding as well as Unequal Error Protection (UEP).

1.1.1 Near-capacity operation

In his seminal contribution [1], Shannon considered source signals that contain some redundancy. Per definition, the redundant content of a particular source signal may theoretically be accurately predicted, given the perfect knowledge of its unpredictable information content. Provided that unimpaired knowledge of the source signal's information content can be reliably conveyed to the receiver by the channel, the transmitter is allowed to employ source coding in order to remove all the source's redundancy, without jeopardising the receiver's ability to reconstruct it. When communicating over a perfectly noiseless channel, Shannon [1] showed that the minimum number of bits that are required to reliably convey perfect knowledge of the source signal's information content to the receiver is given by the source's entropy. Note however that the computational complexity and latency imposed by a source codec typically escalates in optimal entropy coding, where all redundancy is removed from the source signal.

Unfortunately, when communicating over noisy channels, which impose uncertainty on the received signal, the reliable transmission of a source signal's information content may never be guaranteed. However, Shannon [1] also showed that if a source signal's information content is conveyed over a noisy channel at a rate (expressed in bits per second) that does not exceed the channel's capacity, then it is theoretically possible to reconstruct it with an infinitesimally low probability of error. This motivates the employment of channel coding which introduces redundancy into the transmitted signal in a specifically designed manner. This redundancy may be exploited in the receiver to mitigate any channel-induced errors within the original non-redundant information content.

Similarly to optimal entropy coding, the computational complexity and latency imposed by a channel codec escalates, when approaching the channel's capacity [2]. Indeed, Jacobs and Berlekamp [3] showed that the decoding trees employed by the then state-of-the-art sequential decoding algorithm [4–7] for Convolutional Codes (CCs) [8] have an escalating computational complexity, when operating at an information rate above the so-called cutoff rate of the channel. This cutoff rate has an upper bound, which is equal to the channel's capacity. Throughout the 1960s and 1970s, the cutoff rate was regarded as a practical design objective, whilst approaching the channel capacity was deemed to be an unrealistic design goal. The escalating complexity of CC decoding trees may be avoided by employing the more convenient trellises [9] to represent CCs and by applying the Bahl-Cocke-Jelinek-Raviv (BCJR) [10] algorithm. This forms the basis of turbo codes designed by Berrou *et al* [11] in 1993, which employ a pair of iteratively decoded CCs and facilitate operation near the channel's capacity without an escalating computational complexity, as will be described in Section 1.3. The advent of turbo coding led to the 'rediscovery' of Low Density Parity Check (LDPC) codes [12–14], which apply iterative decoding techniques to bipartite graphs [15] in order to facilitate operation near the channel's capacity. Recently, irregular coding techniques [16–18] have enabled 'very-near-capacity' operation, as will be described in Section 1.4. Indeed, in Chapters 3 – 5 of this thesis, we demonstrate a novel application of IrVLCs to near-capacity operation in this manner.

Near-capacity operation is of particular interest in digital multimedia broadcast applications. This is because the channel's capacity reduces with the channel's Signal to Noise Ratio (SNR), which depends on the transmit power, on the distance to the remote receiver and on the amount of noise that is imposed upon the signal. Hence, multimedia broadcast schemes having a particular information transmission rate are associated with a minimum threshold SNR at which the channel's capacity drops below the particular information transmission rate to be maintained. If a multimedia broadcast scheme can be designed to support near-capacity operation, then the threshold SNR required to achieve high quality reception is reduced.

1.1.2 Joint source and channel coding

Shannon's source and channel coding separation theorem [1] states that the removal of undesirable redundancy during source coding and the reintroduction of specifically design intentional redundancy during channel coding can be performed separately, without jeopardising our ability to achieve an infinitesimally low probability of transmission errors, while maintaining a near-capacity information transmission rate. However, Shannon's findings are only valid under a number of idealistic assumptions [19], namely that the information is transmitted over an uncorrelated non-dispersive narrowband Additive White Gaussian Noise (AWGN) channel, while potentially imposing an infinite decoding complexity and buffering latency. These assumptions clearly have limited validity for practical finite-delay transmissions over realistic fading wireless channels [20].

Additionally, Shannon assumed that the source is stationary and is losslessly encoded with the aid of entropy-encoded symbols having equal significance and identical error sensitivity. These assumptions have a limited validity in the case of multimedia transmission, since video, image, audio and speech information is typically non-stationary, having characteristics that vary in time and/or space [20, 21]. Furthermore, typically lossy multimedia coding [20, 21] is employed in order to achieve a high level of compression and a concomitant low bandwidth requirement, while exploiting the psycho-visual properties of the human vision or hearing. Finally, the components of the encoded multimedia information typically have varying perceptual significance and a corresponding unequal error sensitivity, since they are often generated using a number of diverse encoding techniques. For example, video coding typically employs the Discrete Cosine Transform (DCT) [22], Motion Compensation (MC) [23] and entropy coding [24], as exemplified in the MPEG-1 [25], MPEG-2 [26], MPEG-4 [27], H.261 [28], H.263 [29] and H.264 [30] video codecs.

Hence, the employment of joint source and channel coding [31, 32] is motivated. This may be achieved using diverse methods, which we now briefly discuss.

'Channel-optimised' source coding [32] may be employed to reduce the reconstruction error that results when the channel decoder is unable to correct all transmission errors. Here, the source encoder is designed with special consideration of the transmission errors that are most likely to occur, namely those causing a particular binary codeword to be confused with another similar codeword. In this way, channel-optimised source encoding allocates pairs of similar codewords to represent similar reconstructed source parameter values. For example, this may be applied to scalar quantisation [33, 34], where real-valued source samples are represented by one of a number of discrete quantisation levels, which are indexed by binary codewords. In this way, the allocation of binary codeword pairs having a low Hamming distance to represent quantisation level pairs having a low Euclidean distance was demonstrated in [35,36]. Similarly, the authors of [37, 38] proposed Channel-Optimised Vector Quantisation (COVQ), which employs the channel-optimised indexing of Vector Quantisation (VQ) tiles [39]. Note that both scalar- and vector-quantisation shall be discussed in greater detail in Section 1.2.

Joint source and channel coding can also be beneficially employed, if some of the source correlation is not removed during source encoding, resulting in the manifestation of residual redundancy [40] within the resultant source encoded signal. In the receiver, a model of the source correlation may be employed to exploit the residual redundancy in order to provide an error correction capability, which may be employed to mitigate any transmission errors that could not be eliminated during channel decoding. The error correction capability constituted by the source-encoded signal's residual redundancy may be invoked during source decoding in order to provide a Minimum Mean Squared Error (MMSE) estimate of the source signal [41–44]. Alternatively, a Maximum *A posteriori* Probability (MAP) estimate [41, 42, 45] of the source coded signal may be obtained prior to source decoding by exploiting the aforementioned residual redundancy. Note that MMSE and MAP decoders have the ability to consider the entire source coded signal sequence at once [44–46], or they may consider the individual symbols of the source coded signal sequence separately [41–43].

In the same way that residual source redundancy can be exploited in the receiver to provide an error correction capability, so can redundancy that is intentionally introduced during source encoding. For example, this may be achieved by imposing correlation on the source signal with the aid of Trellis Coded Quantisation (TCQ) [47], as exemplified in [46]. Alternatively, Variable Length Error Correction (VLEC) coding [48] may be employed to incorporate redundancy within the source encoded signal. This may be exploited to provide an error correction capability in the receiver [49–52], as will be detailed in Section 1.2. In Section 1.3, we shall detail the turbo principle [11], which may be employed for supporting iterative joint source and channel decoding [53–55], where the redundancy introduced by both source- and channel-coding is alternately exploited for providing *a priori* information for each other concerning the source encoded signal. A novel scheme employing this approach is described in Chapter 2 and hence the detailed literature review of this topic is postponed to Section 2.1.

A further approach to joint source and channel coding that we highlight here employs sophisticated rate allocation in order to jointly optimise the amount of redundancy that is retained or introduced during source encoding and the amount that is introduced during channel encoding. In the scenario, where a lossy source codec is employed, an increased degree of lossy compression may be achieved by employing coarser quantisation, for example. While this results in additional quantisation-induced reconstruction distortion, the associated reduction in the amount of resultant source-coded information facilitates the employment of a lower-rate channel codec, without increasing the overall transmission rate. Since lower-rate channel codecs are associated with higher error correction capabilities, they are capable of mitigating more channel-induced reconstruction distortion. Hence, rate allocation may be employed to optimise the trade-off between quantisation- and channel-induced reconstruction distortion [36, 56, 57].

1.1.3 Unequal error protection

As mentioned above, this thesis considers the application of IrVLCs for UEP. In a manner similar to that of [58–60] for example, UEP may be employed to appropriately protect audio-, speech-, image- and video-encoded bit sequences, which are typically generated using diverse encoding techniques and exhibit various error sensitivities. For example, video coding typically employs the DCT and MC, as described above. As noted in [20], typically a higher degree of video reconstruction distortion is imposed by transmission errors that affect the motion vectors of MC than from those inflicted on the DCT-encoded information. Hence, UEP may be employed to protect all MC-related information with a relatively strong error correction capability, whilst employing a relatively weak error correction code to protect the DCT-encoded information. This approach may hence be expected to yield a lower degree of video reconstruction distortion than equal protection, as noted in [58–60], for example. In Chapter 3, we demonstrate the novel application of IrVLCs for UEP by employing different sets of VLEC codewords having various error correction capabilities to appropriately protect particular components of the source signal that have different error sensitivities.

1.2 Variable length coding

Typically, VLCs are employed to encode uncorrelated source symbol values that exhibit unequal probabilities of occurrence. For this reason, we commence by discussing the generation of symbols having these properties using quantisation. Following this, we describe a number of different types of VLCs.

1.2.1 Quantisation

Uncorrelated source symbols with values having unequal probabilities of occurrence may be generated during the scalar quantisation [33, 34] of uncorrelated real-valued source samples, for example. Here, the J number of real-valued source samples in the frame $\mathbf{e} = \{e_j\}_{j=1}^{J}$ are quantised separately. More specifically, an approximation $\hat{e}_j$ of each source sample e_j is provided by one of K number of real-valued quantisation levels $\{\hat{e}^k\}_{k=1}^{K}$. In each case, the selected quantisation level $\hat{e}^k$ is that particular one, which has the smallest Euclidean distance from the source sample, according to

$$\hat{e}_j = \underset{\{\hat{e}^k\}_{k=1}^{K}}{\operatorname{argmin}} (e_j - \hat{e}^k)^2. \tag{1.1}$$

This selection is indicated using a corresponding source symbol $s_j \in [1 \dots K]$ in the J-symbol frame $\mathbf{s} = \{s_j\}_{j=1}^{J}$. During inverse-quantisation, each reconstructed source sample $\hat{e}_j$ in the J-sample frame $\hat{\mathbf{e}} = \{\hat{e}_j\}_{j=1}^{J}$ approximates the corresponding source sample e_j using the quantisation level $\hat{e}^k$ that is indicated by the corresponding source symbol $s_j \in [1 \dots K]$. Owing to this approximation, quantisation noise is imposed upon the reconstructed source samples, which may be reduced by employing a larger number of quantisation levels K.

Furthermore, in Lloyd-Max quantisation [33, 34] typically the K-means algorithm [61] is employed to select the quantisation levels $\{\hat{e}^k\}_{k=1}^{K}$ in order to minimise the quantisation noise imposed that results for a given number of quantisation levels K. This is illustrated in Figure 1.1 for the $K = 4$-level Lloyd-Max quantisation of uncorrelated

Gaussian distributed source samples having a zero mean and unity variance. Observe that the Probability Distribution Function (PDF) of Figure 1.1 is divided into $K = 4$ sections by the decision boundaries, which are located halfway between each pair of adjacent quantisation levels, namely $\hat{e}^{k'}$ and $\hat{e}^{k'+1}$ for $k' \in [1 \ldots K - 1]$. Here, each section of the PDF specifies the range of source sample values that are mapped to the quantisation level $\hat{e}^k$ at its centre of gravity, resulting in the minimum quantisation noise.

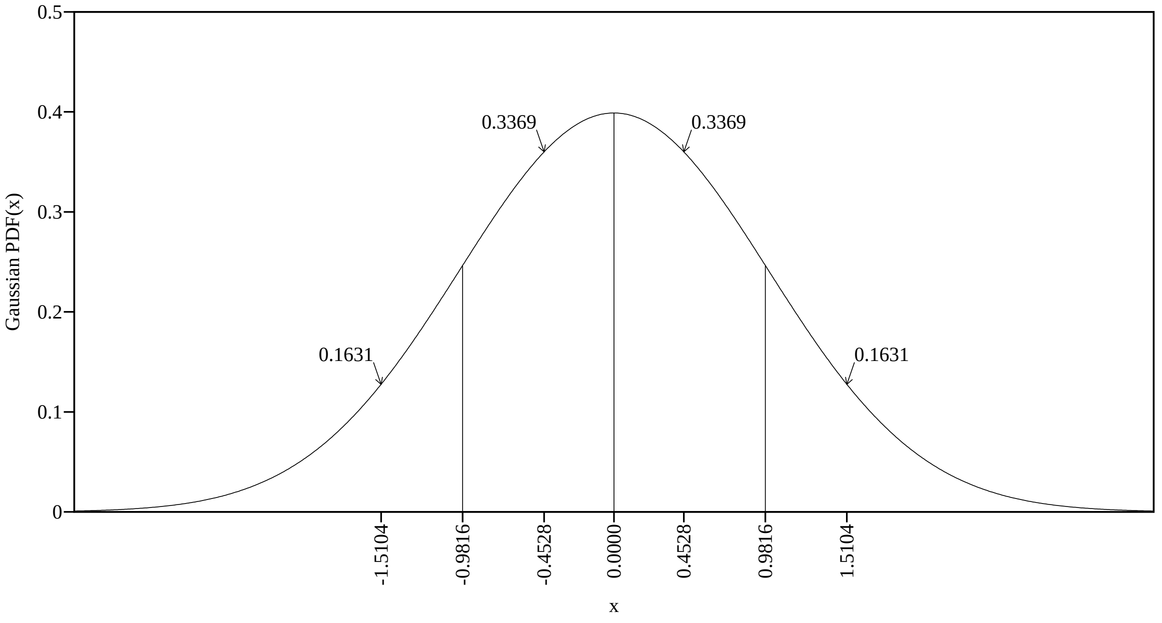


Figure 1.1: Gaussian PDF for zero mean and unity variance. The x axis is labelled with the $K = 4$ Lloyd-Max quantisation levels $\{\hat{e}^k\}_{k=1}^{K}$ and $K - 1 = 3$ decision boundaries as provided in [33]. The decision boundaries are employed to decompose the Gaussian PDF into $K = 4$ sections. The integral $P(k)$ of each PDF section is provided.

The varying probabilities of occurrence $\{P(k)\}_{k=1}^{K}$ of the K number of source symbols values generated during quantisation may be determined by integrating the corresponding sections of the source sample PDF. Figure 1.1 shows the $K = 4$ source symbol value probabilities of occurrence $\{P(k)\}_{k=1}^{K}$ that result for the $K = 4$-level Lloyd-Max quantisation of Gaussian distributed source samples.

In the case where the source samples are correlated, Vector Quantisation (VQ) [39] may be employed to generate source symbols with values having unequal probabilities of occurrence. In contrast to scalar quantisation, where an individual source sample is mapped to each quantisation level, VQ maps a number of correlated source samples to each so-called quantisation tile. Quantisation tiles that impose a minimum quantisation noise may be designed using the Linde-Buzo-Gray (LBG) algorithm [39], which applies the K-means algorithm [61] in multiple dimensions.

1.2.2 Entropy

The amount of information conveyed by a K-ary source symbol having a value $k \in [1 \dots K]$ that occurs with the probability $P(k)$ may be quantified as $-log_2[P(k)]$ bits. This is illustrated in Table 1.1 for the $K = 4$ source symbol value probabilities of occurrence that result for the $K = 4$-level Lloyd-Max quantisation of Gaussian distributed source samples, as depicted in Figure 1.1. For uncorrelated source symbols, the source entropy is given by the average number of bits of information per symbol according to

$$E = -\sum_{k=1}^{K} P(k) \cdot \log_2[P(k)]. \tag{1.2}$$

k	$P(k)$	$-\log_2[P(k)]$	$\mathbf{Huff}^k$	$\mathbf{RVLC}^k$	$\mathbf{VLEC}^k$
1	0.1631	2.62	000	101	1011
2	0.3369	1.57	01	11	0110
3	0.3369	1.57	1	0	000
4	0.1631	2.62	001	1001	11001

Table 1.1: The probabilities of occurrence $P(k)$ and informations $-\log_2[P(k)]$ of the $K = 4$ source symbol values $k \in [1 \dots K]$ that result from the Lloyd-Max quantisation of Gaussian distributed source samples. The corresponding source symbol entropy is $E = 1.91$ bits per source symbol, according to (1.2). Also provided is the composition of the $K = 4$ codewords in the corresponding Huffman $\mathbf{Huff} = \{\mathbf{Huff}^k\}_{k=1}^K$ [24], RVLC $\mathbf{RVLC} = \{\mathbf{RVLC}^k\}_{k=1}^K$ [62] and VLEC $\mathbf{VLEC} = \{\mathbf{VLEC}^k\}_{k=1}^K$ [48] codebooks. According to (1.3), the average Huffman, RVLC and VLEC codeword lengths are $L(\mathbf{Huff}) = 1.99$, $L(\mathbf{RVLC}) = 2.15$ and $L(\mathbf{VLEC}) = 3.83$ bits per source symbol, respectively. The corresponding coding rates are $R(\mathbf{Huff}) = 0.96$, $R(\mathbf{RVLC}) = 0.89$ and $R(\mathbf{VLEC}) = 0.50$, respectively, according to (1.4).

1.2.3 Shannon-Fano coding

Based on his source coding philosophy, Shannon [1], in conjunction with Fano, proposed the employment of VLCs to encode uncorrelated source symbols that assume values having unequal probabilities of occurrence. During Shannon-Fano encoding, the J number of K-ary source symbols in the frame $\mathbf{s} = \{s_j\}_{j=1}^J$ are represented using a K-entry codebook $\mathbf{VLC} = \{\mathbf{VLC}^k\}_{k=1}^K$ of binary VLC codewords, comprising various numbers of bits. More specifically, those source symbols having a particular value of $k \in [1 \dots K]$ are mapped to the corresponding binary VLC codeword $\mathbf{VLC}^k$, which comprises I^k number of bits. Following this, Shannon-Fano encoding is completed by concatenating the different-length VLC codewords that represent the source symbols of the frame $\mathbf{s}$ in order to obtain the VLC-encoded bit frame $\mathbf{u} = \{u_i\}_{i=1}^I$. Note that the number I of bits comprised by the VLC-encoded bit frame $\mathbf{u}$ depends on which particular VLC codewords are invoked.

In order that the source symbols may be uniquely decoded from the concatenated VLC codewords $\mathbf{u}$, it is crucial that no VLC codeword forms a prefix of any other. Shannon [1] showed that this prefix condition can only be satisfied if the average VLC codeword length of

$$L(\mathbf{VLC}) = \sum_{k=1}^{K} P(k) \cdot I^k \tag{1.3}$$

is no lower than the source symbol entropy E. Note that any discrepancy between $L(\mathbf{VLC})$ and E may be quantified by the VLC coding rate of

$$R(\mathbf{VLC}) = \frac{E}{L(\mathbf{VLC})} \in [0, 1] \tag{1.4}$$

and may be attributed to the presence of redundancy within the VLC codewords.

The objective of Shannon-Fano coding is to design a VLC codebook that satisfies the prefix condition, while having a near-unity coding rate $R(\mathbf{VLC})$. This is achieved by initially listing the source symbol values in the order of increasing probability of occurrence. Next, this list is decomposed into a pair of shorter lists comprising source symbol values having roughly equal total probabilities. Source symbol values in one list are allocated a zero-valued bit for their VLC codeword, while a unity-valued bit is allocated to source symbol values in the other list. So long as lists comprising more than one source symbol value exist, they are continually decomposed using the same process and the allocated bits are appended to the end of the source symbol values' VLC codewords. In this way, a tree is constructed from the top down, as illustrated in Figure 1.2 for the source symbol value probabilities of occurrence $\{P(k)\}_{k=1}^{K}$ provided in Table 1.1, resulting in the $K = 4$-entry VLC codebook $\mathbf{Huff} = \{\mathbf{Huff}^k\}_{k=1}^{K}$, which has a coding rate of $R(\mathbf{Huff}) = 0.96$. Note that whilst the described algorithm does ensure that no codeword is a prefix of any other, it does not always achieve the lowest possible average codeword length and hence the highest possible coding rate [24].

1.2.4 Huffman coding

In contrast to the aforementioned Shannon-Fano algorithm, the Huffman coding algorithm [24] facilitates the design of a VLC codebook having a maximal coding rate, while satisfying the above-mentioned prefix condition. This is achieved by constructing an encoding tree from the bottom up, rather than from the top down. More explicitly, rather than continually decomposing lists of source symbol values into pairs of shorter

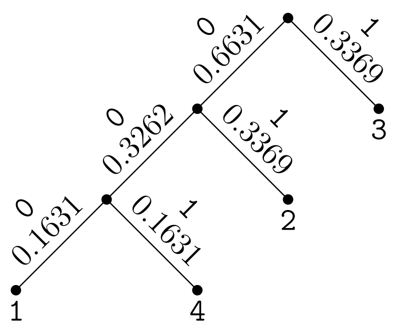


Figure 1.2: Tree representation of the VLC codewords created from the top down by the Shannon-Fano algorithm [1] and from the bottom up by the Huffman algorithm [24] for the $K = 4$ source symbol value probabilities of occurrence $\{P(k)\}_{k=1}^{K}$ provided in Table 1.1. Each branch is labelled with the allocated bit value and the occurrence probability sum of the represented source symbol values. The codewords of the resultant VLC codebook $\mathbf{Huff} = \{\mathbf{Huff}^k\}_{k=1}^{K}$ are provided in Table 1.1.

lists having similar total probabilities of occurrence, the Huffman algorithm continually merges the sets of source symbol values that have the lowest total probabilities of occurrence. Note that in the specific case of using the source symbol value probabilities of occurrence $\{P(k)\}_{k=1}^{K}$ provided in Table 1.1, the Huffman coding algorithm also results in the same tree, as the one shown in Figure 1.2 and in the VLC codebook $\mathbf{Huff} = \{\mathbf{Huff}^k\}_{k=1}^{K}$ of Table 1.1, like the Shannon-Fano algorithm.

A received Huffman-encoded bit frame $\tilde{\mathbf{u}}$ may be Huffman decoded by referring to the corresponding encoding tree, as exemplified in Figure 1.2 for the Huffman codebook **Huff** of Table 1.1. This is achieved by employing the bits of the received frame $\tilde{\mathbf{u}}$ in order to traverse through the tree, in order to generate the original bits. Whenever a leaf node is reached, a corresponding value for a reconstructed source symbol $\tilde{s}_j$ in the frame $\tilde{\mathbf{s}}$ is obtained and the traversal recommences from the root node. Note however that Huffman decoding carried out in this low-complexity manner is sensitive to transmission errors [62].

This problem is illustrated in the example of Figure 1.3a, where a frame $\mathbf{s}$ comprising $J = 11$ source symbols is encoded using the Huffman code **Huff** of Table 1.1 in order to generate the $I = 20$-bit **Huff**-encoded frame $\mathbf{u}$. In this example, the received bit frame $\tilde{\mathbf{u}}$ contains a single bit error and is decoded to obtain the reconstructed symbol frame $\tilde{\mathbf{s}}$. However, the bit error causes the misinterpretation of the three bits that are comprised by a particular Huffman codeword, namely $\mathbf{Huff}^4$, as shown in Figure 1.3a. As a result, these three bits are interpreted as representing two source symbols instead of just one, which is an event we refer to as the loss of ‘synchronisation’. Consequently, all subsequent reconstructed source symbols in the frame $\tilde{\mathbf{s}}$ will be misaligned by a single symbol position, causing numerous future symbol errors. The quality of the reconstructed symbol frame $\tilde{\mathbf{s}}$ may be characterised by its Symbol Error Ratio (SER), which quantifies the fraction of the symbols in the frame $\tilde{\mathbf{s}}$ that are erroneous with respect to those of the encoded symbol frame $\mathbf{s}$. By contrast, if the receiver possesses the *a priori* knowledge that the source symbol frame $\mathbf{s}$ comprises $J = 11$ source symbols, it will become aware of the presence of the transmission error in the example of Figure 1.3a, since 12 source symbols are reconstructed. However, the receiver will be unable to locate this transmission error and may hence nonetheless resort to discarding all reconstructed symbols.

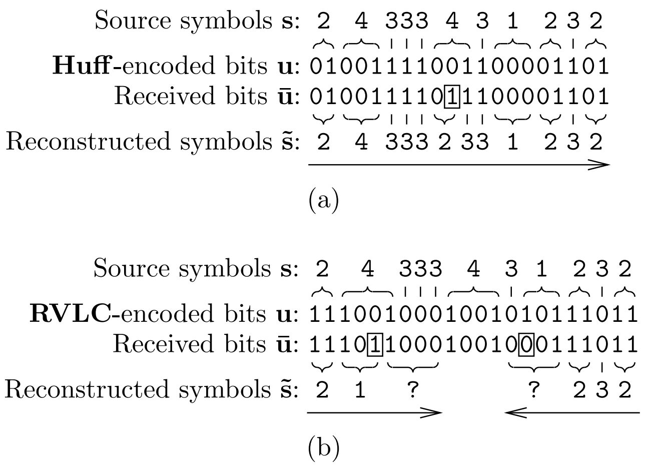

Source symbols s: 2 4 333 4 3 1 2 3 2
Huff-encoded bits u: 01001111001100001101
Received bits ū: 01001111011100001101
Reconstructed symbols s̃: 2 4 333 2 33 1 2 3 2
(a)
Source symbols s: 2 4 333 4 3 1 2 3 2
RVLC-encoded bits u: 1110010001001010111011
Received bits ū: 1110110001001000111011
Reconstructed symbols s̃: 2 1 ? ? 2 3 2
(b)

Figure 1.3: Examples of VLC decoding in the presence of transmission errors using (a) the codebook **Huff** and (b) the codebook **RVLC** from Table 1.1. In both cases, the erroneous bits are boxed.

1.2.5 Reversible variable length coding

The sensitivity of Huffman coding to transmission errors motivated the introduction of Reversible Variable Length Codes (RVLC) [62]. Unlike Huffman codes, RVLCs facilitate decoding both in the classic forward, as well as in the reverse direction, starting from the end of the received RVLC-encoded bit frame $\bar{\mathbf{u}}$. The benefit of this is that in the presence of transmission errors, more RVLC codewords may be recovered without errors. This reversible decoding is facilitated because, in addition to fulfilling the prefix condition described above, RVLCs additionally fulfil a suffix condition, which prevents any RVLC codeword from forming a suffix of any other. This may be observed for the RVLC codebook **RVLC** provided in Table 1.1. By contrast, note that the Huffman codebook **Huff** of Table 1.1 does not fulfil the suffix condition, preventing its decoding in the reverse direction. Note that since RVLC codebooks have to additionally satisfy the suffix condition, they typically cannot achieve coding rates as high as those of the equivalent Huffman codebook. Indeed, the RVLC codebook **RVLC** of Table 1.1

has a coding rate of $R(\mathbf{RVLC}) = 0.89$, which is lower than the $R(\mathbf{Huff}) = 0.96$ coding rate of the Huffman codebook **Huff**. This has motivated the design of methods for constructing RVLC codebooks that strive for maximal coding rates [63–67].

Figure 1.3b illustrates how the additional ability of decoding in the reverse direction can benefit the reconstruction of source symbols that are encoded using the RVLC codebook **RVLC** of Table 1.1. Here, the reconstructed symbols $\tilde{\mathbf{s}}$ can be recovered from both ends, provided that the receiver possesses the *a priori* knowledge that the source symbol frame $\mathbf{s}$ comprises J number of source symbols. This is achieved by decoding the received bit frame $\bar{\mathbf{u}}$ of Figure 1.3b from both ends, following the corresponding decoding tree. Note that if the RVLC codewords are symmetric, like those of the codebook **RVLC** provided in Table 1.1, then the same tree may be employed to assist decoding in both directions. By contrast, if the RVLC codewords are asymmetric [62], then different trees are required for decoding in the opposite directions. Note that owing to the higher degree of design flexibility that is afforded for asymmetric RVLC codebooks, they can typically achieve higher coding rates than the equivalent symmetric RVLC codebooks.

Note that in the case of Huffman codebooks, such as **Huff** of Table 1.1, any continuous bit sequence may be interpreted as a valid codeword sequence. This is because in Huffman coding trees, such as that provided in Figure 1.2, all internal nodes have two branches; one which is followed when a logical zero bit is encountered and one which is followed in the case of a logical one bit. By contrast, RVLC trees typically contain some nodes that have only a single branch, as exemplified in Figure 1.4 for the RVLC codebook **RVLC** of Table 1.1. This induces some redundancy within the RVLC codewords, which may be exploited for detecting erroneous bits. More specifically, for the RVLC codebook **RVLC** of Table 1.1, the bit sequence `1000` does not correspond to a legitimate path between the root node and a leaf node of the tree depicted in Figure 1.4. Hence, if the bit sequence `1000` is encountered during decoding in either the forward or reverse direction, as exemplified in Figure 1.3b, the presence of transmission errors within the previously considered bits of the received bit frame $\bar{\mathbf{u}}$ is detected. Decoding is typically halted in this event, since synchronisation is likely to have been lost and any further reconstructed symbols are likely to be erroneous. It is for this reason that no attempt is made to decode the bits in between the two instances, where the bit sequence `1000` was encountered in Figure 1.3b.

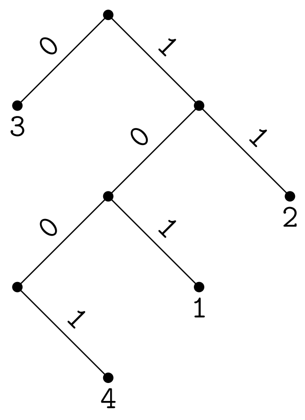


Figure 1.4: Tree representation of the RVLC codebook **RVLC** provided in Table 1.1.

1.2.6 Variable length error correction coding

While RVLC codebooks allow the presence of bit errors to be detected in the manner described above, they typically do not facilitate the precise localisation and, hence, correction of bit errors. This motivated the introduction of VLEC codebooks [48,68,69] for this purpose. Since VLEC codebooks facilitate error correction as well as source coding, they may be viewed as achieving joint source and channel coding, as described in Section 1.1.2. Similarly to RVLC codebooks, VLEC codebooks contain redundancy, having tree representations in which some nodes have only a single emerging branch. However, VLEC codebooks typically contain more redundancy and, hence, have lower coding rates than RVLC codebooks, since this facilitates a higher error correction capability. For example, the VLEC codebook **VLEC** of Table 1.1 has a relatively low coding rate of $R(\mathbf{VLEC}) = 0.50$, yielding the tree of Figure 1.5, which contains eight nodes having only a single emerging branch.

While a tree may be employed to decode the source symbols one at a time, trellises may be constructed that facilitate the consideration of multiple source symbols at once. More specifically, trellises describe the VLEC code constraints, which limit the legitimate sequences of bit values that may appear within a frame $\mathbf{u}$ of concatenated VLEC codewords. We now introduce a number of VLEC trellis designs and describe their employment for error correction. Later, we shall discuss metrics for quantifying the error correction capability of VLEC codes.

1.2.6.1 Bit-based VLEC trellis

Figure 1.5 depicts a bit-based trellis [49, 70, 71] representation of the VLEC codebook **VLEC** provided in Table 1.1. This is constructed by mapping the branches and nodes of the tree to transitions and states within the trellis, respectively. As shown in Figure 1.5, each bit-based trellis state $S_{(\ddot{\imath},\ddot{n})}$ is indexed by a node state index $\ddot{n}$ and a bit state index $\ddot{\imath}$. Note that states having a particular bit state index $\ddot{\imath}$ may be entered *after* the consideration of the first $\ddot{\imath}$ number of bits in the frame $\mathbf{u} = \{u_i\}_{i=1}^{I}$. For this reason, the bit indices i are positioned between the bit state indices $\ddot{\imath}$ in Figure 1.5.

The construction of a bit-based VLEC trellis commences by mapping each node of the tree to a trellis state having a particular bit state index $\ddot{\imath}$. In each case, this bit state index $\ddot{\imath}$ equals the number of branches in (and hence number of bits required to traverse) the path from the tree's root node to the considered node. Furthermore, the tree's root node is mapped to a state having a node state index of $\ddot{n} = 0$, the tree's internal nodes are mapped to states having different node state indices $\ddot{n} > 0$ and the tree's leaf nodes are also mapped to states having a node state index of $\ddot{n} = 0$, like the root node. Finally, the branches between particular nodes of the tree are mapped to transitions between the corresponding states within the bit-based trellis. Note that the tree of Figure 1.5 has been specially arranged to clearly illustrate the described mappings. In the bit-based trellis of Figure 1.5, the consideration of multiple source symbols at once is facilitated by repeatedly performing the described mappings, starting from each instance where a state having a node state index of $\ddot{n} = 0$ is invoked.

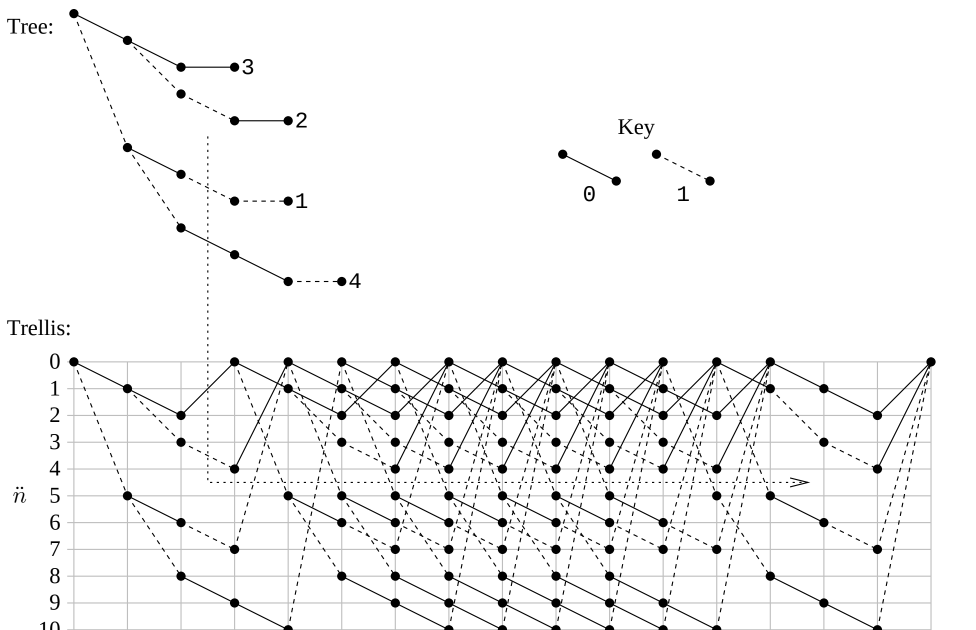

Tree:
3
2
1
4
Key
0
1
Trellis:
0
1
2
3
4
$\ddot{n}$
5
6
7
8
9
10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 i
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 $\ddot{\imath}$

Figure 1.5: Tree and bit-based trellis representations of the VLEC codebook **VLEC** provided in Table 1.1. The trellis is specific to the case where the **VLEC**-encoded bit frame **u** comprises $I = 16$ bits.

Note that bit-based VLEC trellises are often terminated, since any path that represents a sequence of codewords will terminate in a state having a node state index of $\ddot{n} = 0$. This is exemplified by the bit-based trellis of Figure 1.5, which is terminated after $I = 16$ bits in the state $S_{(16,0)}$. In this case, termination was achieved by removing all transitions that result in states other than $S_{(16,0)}$ being entered after all $I = 16$ bits have been considered. Note that for this reason, the bit-based trellis of Figure 1.5 facilitates the consideration of codeword sequences that contain exactly $I = 16$ bits.

1.2.6.2 ML VLEC sequence estimation

Once a trellis has been constructed to describe the code constraints of a VLEC codebook, it may be employed to correct errors inflicted upon a received bit frame $\bar{\mathbf{u}}$ [49, 70, 71]. More specifically, the Viterbi algorithm [72, 73] may be employed for Maximum Likelihood (ML) sequence estimation. This obtains the reconstructed bit frame $\tilde{\mathbf{u}}$ comprising the particular sequence of concatenated codewords from the K-entry VLEC codebook $\mathbf{VLEC} = \{\mathbf{VLEC}^k\}_{k=1}^K$ that agrees with the maximum number of bit values within the received frame $\bar{\mathbf{u}}$. The Viterbi algorithm is exemplified in Figure 1.6, in which the bit-based trellis of Figure 1.5 is employed to correct five transmission errors within a received version $\bar{\mathbf{u}}$ of a VLEC-encoded bit frame $\mathbf{u}$ comprising four codewords from the VLEC codebook $\mathbf{VLEC}$ of Table 1.1.

The Viterbi algorithm commences by allocating a transition metric $m(T)$ to each transition T in the trellis, having a unity-value, if the represented bit value agrees with the corresponding bit in the received bit frame $\bar{\mathbf{u}} = \{\bar{u}_i\}_{i=1}^I$ or a zero-value otherwise. More explicitly,

$$m(T) = \begin{cases} 1 & \text{if } b^T = \bar{u}_{i^T} \\ 0 & \text{otherwise} \end{cases}, \tag{1.5}$$

where the transition T represents the employment of the value $b^T \in \{0, 1\}$ by the bit u_{i^T}.

The Viterbi algorithm proceeds by employing a forward recursion to assign a state metric $M(S_{(\ddot{\imath},\ddot{n})})$ to each state $S_{(\ddot{\imath},\ddot{n})}$ within the trellis according to

$$M(S_{(\ddot{\imath},\ddot{n})}) = \max_{T \in \mathrm{to}(S_{(\ddot{\imath},\ddot{n})})} M(T), \tag{1.6}$$

where $\mathrm{to}(S_{(\ddot{\imath},\ddot{n})})$ is the set of all transitions that merge to the state $S_{(\ddot{\imath},\ddot{n})}$ and $M(T)$ is the cumulative transition metric of the transition T, which is given by

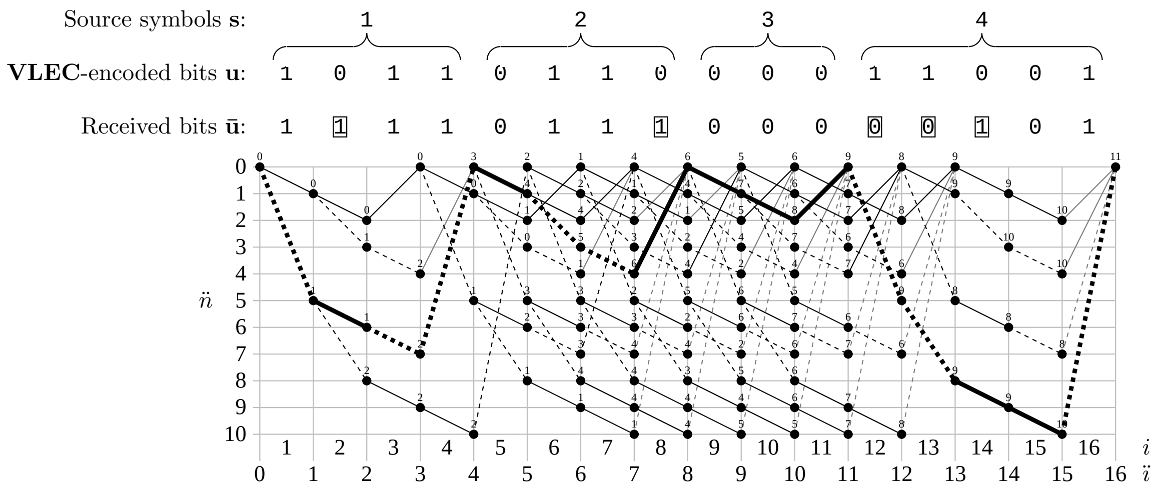
$$M(T) = m(T) + M[\mathrm{fr}(T)], \tag{1.7}$$

where $\mathrm{fr}(T)$ is the state that the transition T emerges from and $M(S_{(0,0)}) = 0$.

Observe in Figure 1.6 that some states having a node state index of $\ddot{n} = 0$ have more than one merging transition. According to (1.6), each corresponding state metric

$M(S_{(\ddot{\imath},\ddot{n})})$ is obtained as the maximum cumulative transition metric of the merging transitions. At this stage, the particular transition T that provides the maximum cumulative transition metric $M(T)$ is selected to form part of the survivor path to the considered state.

Following the completion of the aforementioned forwards recursion, the ML reconstructed bit frame $\tilde{\mathbf{u}}$ is obtained by starting from the state $S_{(I,0)}$ and following the survivor paths in the reverse direction to the state $S_{(0,0)}$ and then outputting the bit values associated with the corresponding transitions in the reverse order. The quality of the resultant reconstructed bit frame $\tilde{\mathbf{u}}$ may be characterised by its Bit Error Ratio (BER), which quantifies the fraction of the bits in the frame $\tilde{\mathbf{u}}$ that are erroneous with respect to those of the transmitted bit frame $\mathbf{u}$. Note that the reconstructed bit frame $\tilde{\mathbf{u}}$ obtained in the example of Figure 1.6 corrects all five bit errors that were present within the received bit frame $\bar{\mathbf{u}}$, giving zero errors. Once the Viterbi algorithm has been completed, the reconstructed bit frame $\tilde{\mathbf{u}}$ may be decoded as usual in order to obtain the reconstructed source symbol frame $\tilde{\mathbf{s}}$.



Source symbols s: 1 2 3 4
VLEC-encoded bits u: 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1
Received bits ū: 1 1 1 1 0 1 1 1 0 0 0 0 0 1 0 1
n̈
i
ï

Figure 1.6: Example of applying the Viterbi algorithm to the bit-based trellis of Figure 1.5 in order to achieve ML sequence estimation and to correct five transmission errors within a sequence of four codewords from the VLEC codebook **VLEC** of Table 1.1. In this figure, the erroneous received bits are boxed, the state metrics $M(S_{(\ddot{\imath},\ddot{n})})$ are provided and the ML sequence path is indicated using bold transitions.

Note that the computational complexity associated with the Viterbi algorithm of Figure 1.6 is commensurate with the number of transitions employed per bit in the associated bit-based VLEC trellis. Since this is equal to the number of branches within the corresponding tree representation, the bit-based trellis complexity depends on the properties of the VLEC codebook that influence its tree. Therefore a low trellis complexity results if the VLEC codebook contains few entries, if it has short codewords (giving a high coding rate) and if the number of bits that are shared by the codewords at their beginning is high.

1.2.6.3 Symbol-based VLEC trellis

Observe that there are some paths between the states $S_{(0,0)}$ and $S_{(I,0)}$ of Figure 1.6 that represent five codewords from the VLEC codebook **VLEC** of Table 1.1. For this reason, if the ML sequence estimation process exemplified in Figure 1.6 was unable to correct all of the transmission errors, the resultant reconstruction may comprise five codewords instead of $J = 4$, resulting in the loss of synchronisation, as discussed above. This motivates the employment of a symbol-based VLEC trellis [50,74]. Similarly to the bit-based VLEC trellis, the symbol-based trellis restricts its consideration to codeword sequences that contain a specific number of bits. However, unlike the bit-based VLEC trellis, the symbol-based trellis additionally restricts its consideration to a specific number of codewords. This is facilitated, because each transition in a symbol-based VLEC trellis represents an entire codeword, rather than just a single bit like in the bit-based trellis.

Figure 1.7 depicts a symbol-based trellis representation of the VLEC codebook **VLEC** provided in Table 1.1 for the case when the source symbol frame $\mathbf{s}$ comprises $J = 4$ source symbols and is encoded using an $I = 16$-bit frame $\mathbf{u}$. Similarly to the bit-based trellis exemplified in Figure 1.5, each state $S_{(\ddot{\imath},\ddot{\jmath})}$ in the symbol-based trellis of Figure 1.7 is indexed by a bit state index $\ddot{\imath}$. As before, the bit indices $i \in [1 \dots I]$ are positioned between the bit state indices $\ddot{\imath} \in [0 \dots I]$ and states having a particular bit state index $\ddot{\imath}$ may be entered into after the consideration of the first $\ddot{\imath}$ number of bits in the frame $\mathbf{u} = \{u_i\}_{i=1}^{I}$. However, in contrast to the bit-based trellis, the states of the symbol-based trellis are indexed by a symbol state index $\ddot{\jmath}$, as shown in Figure 1.7. In analogy with the bit state indices, states having a particular symbol state index $\ddot{\jmath}$ may be entered into after the consideration of the first $\ddot{\jmath}$ number of symbols in the frame $\mathbf{s} = \{s_j\}_{j=1}^{J}$. For this reason, the symbol indices $j \in [1 \dots J]$ are positioned between the symbol state indices $\ddot{\jmath} \in [0 \dots J]$ in Figure 1.7. As shown in the legend of Figure 1.7, each transition T in its trellis represents a particular source symbol value $k^T \in \{1, 2, 3, 4\}$ and the I^{k^T} number of bit values from the corresponding VLEC codeword $\mathbf{VLEC}^{k^T}$. Within the trellis, each transition T encompasses a single symbol index and I^{k^T} number of bit indices.

The trellis of Figure 1.7 was constructed in a recursive manner, emerging from the state $S_{(0,0)}$. During this recursion, a set of the $K = 4$ transitions depicted in the key of Figure 1.7 was positioned to emerge from each state that may be reached from

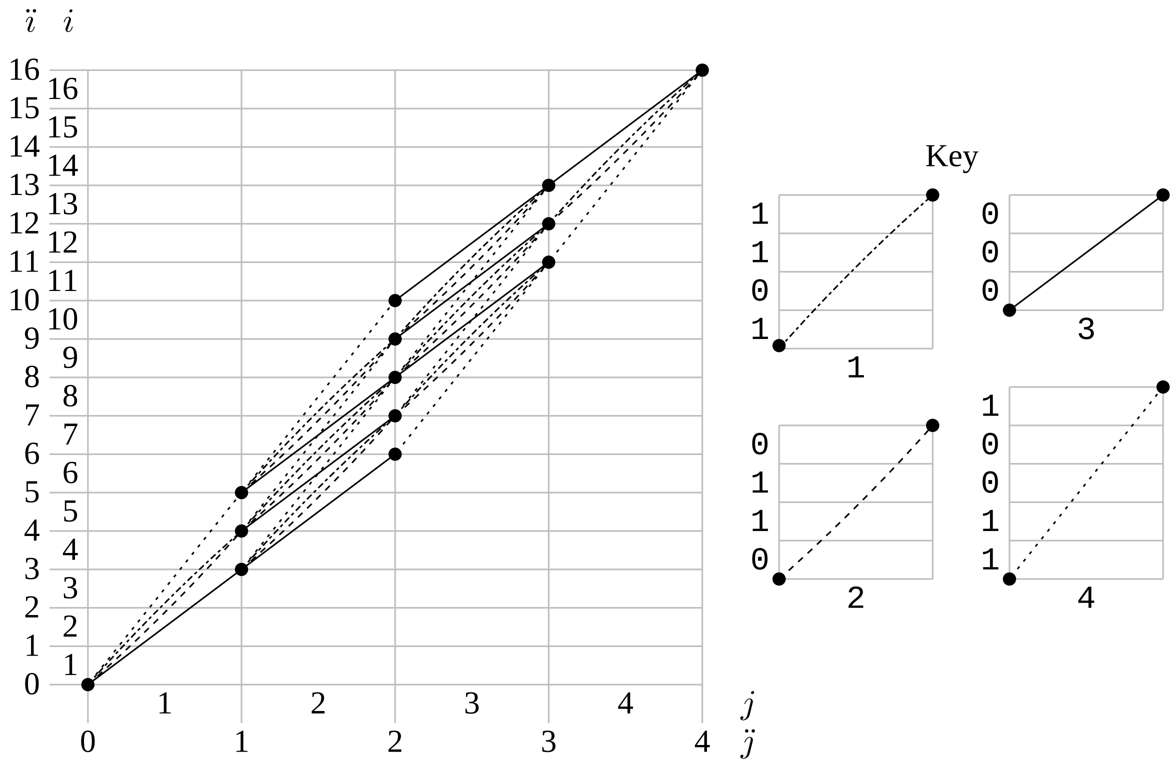


Figure 1.7: Symbol-based trellis representation of the VLEC codebook **VLEC** provided in Table 1.1 for the case when the source symbol frame $\mathbf{s}$ comprises $J = 4$ source symbols and is encoded using an $I = 16$-bit frame $\mathbf{u}$.

previously placed transitions. However, similarly to the bit-based trellis of Figure 1.5, the symbol-based trellis depicted in Figure 1.7 is also terminated. This was achieved by removing all transitions that result in states other than $S_{(I,J)}$ being reached, since $I = 16$ bits and $J = 4$ symbols are considered. It is in this way that the symbol-based trellis restricts its consideration to codeword sequences that represent a specific number of source symbols using a specific number of bits.

Note that since the symbol-based VLEC trellis considers only codeword sequences that represent a specific number of encoded source symbols, its employment as the basis of ML sequence estimation using the Viterbi algorithm avoids the above-mentioned synchronisation problems that are associated with the bit-based VLEC trellis. Here, the Viterbi algorithm proceeds in a similar manner to that described for the bit-based trellis, except for that each transition metric $m(T)$ is quantified as the number of bits within the associated codeword $\mathbf{VLEC}^{k^T}$ that agrees with the corresponding bits in the received frame $\bar{\mathbf{u}}$. Note that the reconstructed symbol frame $\tilde{\mathbf{s}}$ may be directly obtained, if the symbol-based VLEC trellis is employed as the basis of ML sequence estimation. By contrast, when the bit-based trellis is employed, the resultant reconstructed bit frame $\tilde{\mathbf{u}}$ must be decoded in order to obtain the reconstructed symbol frame $\tilde{\mathbf{s}}$, as described above.

Note that as the number J of source symbols considered by a symbol-based VLEC trellis is increased, its parallelogram shape becomes wider and the number of transitions employed exponentially increases. For this reason, the symbol-based trellis complexity per bit depends on the number J of source symbols considered and is typically significantly higher than that of the bit-based trellis described above. Similarly to the bit-based trellis complexity, the symbol-based trellis complexity depends on the number of entries employed within the corresponding VLEC codebook. Furthermore, the difference in length between the longest and shortest VLEC codewords influences the symbol-based trellis complexity, since this dictates the size of the acute angles within the symbol-based trellis' parallelogram shape.

A number of structures other than trellises have been proposed for VLEC decoding. The sequential decoding of VLCs using trees was investigated in [52, 75] and the employment of Bayesian networks was considered in [76]. Graph representations of VLEC codebooks have been demonstrated in [51, 77–79]. Unlike the aforementioned trellis representations, these graphs allow any source symbol correlation to be exploited for assisting error correction. This was also facilitated in [80], where a third dimension was added to the symbol-based VLEC trellis for this purpose. Finally, the video codec of Chapter 2 will employ novel modification of the symbol-based VLEC trellis that replaces the symbol indices on the vertical trellis axis with video block indices. As in the symbol-based trellis, each transition represents a single source symbol, however, in the novel scheme of Chapter 2, different source symbols represent various numbers of video blocks. Hence, in the novel trellis structure of Chapter 2, different transitions encompass various numbers of bit indices *and* video block indices. Chapter 2 will also apply the Viterbi algorithm to the above mentioned trellis structure in a novel manner, facilitating MMSE -based video encoding.

1.2.6.4 VLEC error correction capability

The error correction capability of a VLEC codebook $\mathbf{VLEC}$ is typically characterised by its free distance $d_{\text{free}}(\mathbf{VLEC})$, which is equal to the minimum number of differing bits in any pair of equal-length VLEC codeword sequences [48]. This is because transmission errors that transform the transmitted sequence of VLEC-encoded bits into any other legitimate sequence of VLEC-encoded bits cannot be detected during VLEC decoding [48]. Hence, the free distance characterises the probability of occurrence of the most likely undetectable transmission error scenario, namely that occurring in the presence of the lowest number of corrupted bits.

However, the free distance of a VLEC codebook $d_{\text{free}}(\mathbf{VLEC})$ is typically difficult to determine. Fortunately, Buttigieg and Farrell formulated a simple free distance lower bound $\bar{d}_{\text{free}}(\mathbf{VLEC})$ in [48],

$$d_{\text{free}}(\mathbf{VLEC}) \geq \bar{d}_{\text{free}}(\mathbf{VLEC}) = \min[d_{b_{\min}}(\mathbf{VLEC}), d_{d_{\min}}(\mathbf{VLEC}) + d_{c_{\min}}(\mathbf{VLEC})]. \tag{1.8}$$

Here, the so-called minimum block-distance $d_{b_{\min}}(\mathbf{VLEC})$ [48] is defined as the minimum number of differing bits in any pair of equal-length codewords in the VLEC codebook **VLEC**. Furthermore, the minimum divergence distance $d_{d_{\min}}(\mathbf{VLEC})$ is defined as the minimum number of differing bits in any pair of unequal-length, left-aligned codewords, while the minimum number of differing bits in any pair of unequal-length, right-aligned codewords is termed the minimum convergence distance $d_{c_{\min}}(\mathbf{VLEC})$ [48]. As a benefit of its ease of calculation, typically the free distance lower bound $\bar{d}_{\text{free}}(\mathbf{VLEC})$ of (1.8) is employed instead of the free distance $d_{\text{free}}(\mathbf{VLEC})$ to characterise the error correction capability of a VLEC codebook **VLEC**.

For example, the block-, divergence- and convergence-distances between the $K = 4$ codewords of the VLEC codebook **VLEC** of Table 1.1 are provided in Table 1.2. Furthermore, Table 1.3 provides the minimum block-, divergence- and convergence-distances between the codewords of the codebooks **Huff**, **RVLC** and **VLEC** from Table 1.1, together with their free distance lower bounds. Note that both Huffman and RVLC codebooks can be interpreted as special cases of VLEC codebooks. This is because the prefix-condition of Huffman and RVLC codebooks guarantees a minimum divergence distance of at least one. Similarly, a minimum convergence distance of at least one is guaranteed by the suffix condition of RVLC codebooks.

		k^2			
		1	2	3	4
k^1	1	–	3	2/2	3/1
	2	3	–	2/2	2/4
	3	2/2	2/2	–	2/1
	4	3/1	2/4	2/1	–

Table 1.2: Block-, divergence- and convergence-distances between codewords from the codebook **VLEC**. For codewords $\mathbf{VLEC}^{k^1}$ and $\mathbf{VLEC}^{k^2}$ having the same length $I^{k^1} = I^{k^2}$, the block distance $d_b(\mathbf{VLEC}^{k^1}, \mathbf{VLEC}^{k^2})$ is provided. By contrast, the divergence-distance $d_d(\mathbf{VLEC}^{k^1}, \mathbf{VLEC}^{k^2})$ and the convergence distance $d_c(\mathbf{VLEC}^{k^1}, \mathbf{VLEC}^{k^2})$ are provided using the format $d_d(\mathbf{VLEC}^{k^1}, \mathbf{VLEC}^{k^2})/d_c(\mathbf{VLEC}^{k^1}, \mathbf{VLEC}^{k^2})$ if the codewords have different lengths $I^{k^1} \neq I^{k^2}$.

VLC	$d_{b_{\min}}(\mathbf{VLC})$	$d_{d_{\min}}(\mathbf{VLC})$	$d_{c_{\min}}(\mathbf{VLC})$	$\bar{d}_{\text{free}}(\mathbf{VLC})$
Huff	1	1	0	1
RVLC	2	1	1	2
VLEC	3	2	1	3

Table 1.3: Minimum block-, divergence- and convergence-distances between codewords from the codebooks **Huff**, **RVLC** and **VLEC** from Table 1.1, together with their free distance lower bounds.

Note that VLEC codebooks having different error correction capabilities may have the same Integer-Valued Free Distance (IV-FD) lower bound $\bar{d}_{\text{free}}(\mathbf{VLEC})$. Hence these error correction capabilities cannot be compared by using the IV-FD lower bound $\bar{d}_{\text{free}}(\mathbf{VLEC})$ as a metric owing to its non-unique integer value. This observation motivates the introduction of a novel Real-Valued Free Distance Metric (RV-FDM) $D(\mathbf{VLEC})$ in Chapter 4 in order to characterise the error correction capability of a VLEC codebook **VLEC**. Since this RV-FDM is defined within the real-valued domain, it facilitates the unambiguous comparison of diverse VLEC codebooks' error correction capabilities, even if they happen to have the same IV-FD lower bound.

Methods for designing VLEC codebooks have been proposed in [67, 69, 81]. These methods attempt to maximise the coding rate of VLEC codebooks having particular specified distance properties. Note that high free distance lower bounds and, hence, strong error correction capabilities typically result in codebooks having relatively low coding rates. This is because long VLEC codewords are required in order that the design freedom necessary to achieve high distances may be afforded. In Chapter 4, we shall introduce a novel Genetic Algorithm (GA) for designing VLEC codebooks. Like the methods of [67, 69, 81], this GA facilitates the design of VLEC codebooks having particular specified distance properties. However, unlike the methods of [67,69,81], the VLEC codebooks designed using the GA can have arbitrary coding rates, rather than only maximal coding rates.

1.3 The turbo principle

As described in Section 1.1.1, turbo codes [11], as well as schemes employing the more general 'turbo principle' [82,83], facilitate near-capacity operation without imposing an excessive decoding complexity or latency. This is achieved using an iterative exchange of so-called 'soft' information between a number of decoders. With each iteration, the quality of the exchanged soft information improves, until no further improvement may be obtained.

We begin by defining soft information and by contrasting it to so-called 'hard' information. Following this, the exploitation and generation of soft information is described in the context of VLECs.

1.3.1 Hard and soft information

The bit-based ML sequence estimation algorithm detailed in Section 1.2.6.2 may be considered to be a Hard-In Hard-Out (HIHO) algorithm. This is because both the inputs and outputs of the described algorithm comprise 'hard' information pertaining to the VLEC-encoded bit frame $\mathbf{u} = \{u_i\}_{i=1}^{I}$. More specifically, this hard information is based on the decision, whether the decoder was more confident that the bits were zero- or unity-valued. However, they do not quantify *how* confident the decoder was that the bits should take these values, motivating the employment of 'soft' information for this purpose.

For example, soft information may be employed to express the uncertainty that is associated with a received signal that was subject to Additive White Gaussian Noise (AWGN) during its transmission. More specifically, during Binary Phase Shift Keying (BPSK) modulation [84], for example, zero-valued bits are transmitted by modulating a carrier signal using an amplitude of $+1$, while an amplitude of -1 is employed to signal unity-valued bits. However, Gaussian-distributed noise will be added to these amplitudes during the signal's transmission over the AWGN channel. Since the demodulator is unaware of the specific noise amplitude contaminating a particular bit, it cannot be sure whether the bit should be zero- or unity-valued. However, if the demodulator is aware of the variance of the AWGN, it is capable of determining how *likely* the bit is to be a logical zero or a logical one [84].

Soft information pertaining to bits is typically represented using Logarithmic Likelihood Ratios (LLRs) within the receiver. Here, the particular LLR $L(u_i)$ in the frame $L(\mathbf{u}) = \{L(u_i)\}_{i=1}^{I}$ that pertains to the bit u_i from the frame $\mathbf{u} = \{u_i\}_{i=1}^{I}$ is specified according to

$$L(u_i) = \ln \frac{P(u_i = 0)}{P(u_i = 1)}, \tag{1.9}$$

where $P(u_i = b) \in [0, 1]$ is the probability or confidence that the bit u_i had the value $b \in \{0, 1\}$ within the transmitter. Note that the logarithmic domain is employed since it provides symmetry, resulting in LLRs having a positive or negative sign, when a higher confidence is instilled within a logical zero- or a logical one-valued bit, respectively. Furthermore, the level of this confidence is commensurate with the LLR's magnitude.

1.3.2 Soft VLEC decoders

Let us now discuss a number of trellis-based soft VLEC decoding algorithms [85]. Throughout these discussions we assume that the bit-based VLEC trellis exemplified in Figure 1.5 is employed. In analogy with the bit-based ML sequence estimation algorithm detailed in Section 1.2.6.2, the algorithms discussed in this section have soft inputs and outputs that pertain to the VLEC-encoded bit frame **u**. However, the aforementioned algorithms may also be adapted to operate on the basis of the symbol-based trellis exemplified in Figure 1.7. In this case, the algorithms have soft inputs pertaining to the VLEC-encoded bit frame **u** and soft outputs pertaining to the source symbol frame **s**, in analogy with the discussions of Section 1.2.6.3.

Similar to the ML sequence estimation algorithms described in Sections 1.2.6.2 and 1.2.6.3, the computational complexity of the algorithms introduced in the following discussions depends on the number of trellis transitions employed. In Section 1.3.2.4, we shall quantify and also reduce the computational complexity of these algorithms, as well as highlighting a number of their implementational issues.

1.3.2.1 ML SIHO VLEC sequence estimation

The HIHO ML VLEC sequence estimation algorithm described in Section 1.2.6.2 may be adapted to accept soft inputs [70], hence becoming a Soft-In Hard-Out (SIHO) algorithm. This may be achieved by replacing the transition metric $m(T)$ of (1.5) with

$$m(T) = P(u_{i^T} = b^T), \tag{1.10}$$

where the transition T represents the adoption of the value $b^T \in \{0, 1\}$ by the bit u_{i^T} and $P(u_i = b)$ may be obtained from the corresponding LLR $L(u_i)$ according to (1.9). Similarly, the cumulative transition metric $M(T)$ of (1.7) should be replaced by

$$M(T) = m(T) \times M[\mathrm{fr}(T)], \tag{1.11}$$

where $\mathrm{fr}(T)$ is the state that the transition T emerges from and the state metric $M(S_{(\ddot{\imath},\ddot{n})})$ is given by (1.6), as before. Note that the metric accumulation of (1.7) is replaced by a multiplication in (1.11), because the joint probability of two independent events is given as the product of their individual probabilities. For this reason, the ML SIHO sequence estimation algorithm considered assumes the independence of the LLRs within the frame $L(\mathbf{u})$. Note that this assumption is no longer valid and the optimality of the algorithm is lost, if the LLR frame $L(\mathbf{u})$ is obtained by the demodulation of a signal that was transmitted over a correlated fading channel, for example.

1.3.2.2 SISO VLEC algorithms

The Viterbi algorithm may also be appropriately adapted to provide a soft output, hence becoming a Soft-In Soft-Out (SISO) algorithm. This is achieved using the Soft Output Viterbi Algorithm (SOVA) [86]. Similarly to the soft-input ML sequence estimation algorithm of Section 1.3.2.1, the SOVA algorithm provides the hard ML reconstructed bit frame $\tilde{\mathbf{u}}$, when it is applied to the bit-based VLEC trellis exemplified in Figure 1.5. However, unlike the soft-input ML sequence estimation algorithm, the SOVA algorithm additionally outputs soft information to express the confidence associated with each bit in the reconstructed frame $\tilde{\mathbf{u}}$.

Since the soft output of the SOVA algorithm pertains to the bits in the ML *sequence*, it does not provide so-called *A Posteriori* Probabilities (APPs), which may be obtained by considering the bits *individually*. By contrast, APP SISO decoding may be achieved by applying the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [10] to the bit-based VLEC trellis exemplified in Figure 1.5 [71]. In this way, the BCJR algorithm applies the VLEC code constraints to the input *a priori* soft information, outputting improved so-called *a posteriori* soft information.

The BCJR algorithm commences by considering the *a priori* LLR frame $L_a(\mathbf{u}) = \{L_a(u_i)\}_{i=1}^{I}$ and assigning a so-called gamma value [10] to each transition within the trellis. The gamma value $\gamma(T)$ assigned to a particular transition T is given by considering a single corresponding *a priori* LLR according to

$$\gamma(T) = P(T|\mathrm{fr}(T)) \cdot P_a(u_{i^T} = b^T), \tag{1.12}$$

where the transition T represents the adoption of the value $b^T \in \{0, 1\}$ by the bit u_{i^T} and $P_a(u_i = b)$ may be obtained from the corresponding *a priori* LLR $L_a(u_i)$ with reference to (1.9). Furthermore, $P(T|\mathrm{fr}(T))$ is the conditional probability that the specific trellis transition T is invoked, given that the state $\mathrm{fr}(T)$ from which it emerges was invoked. These values may be obtained by referring to the VLEC encoding tree that is exemplified in Figure 1.8 and to the source symbol value probabilities of occurrence that are exemplified in Table 1.1. More specifically, a particular transition's conditional probability $P[T|\mathrm{fr}(T)]$ may be obtained by considering the occurrence probabilities of the symbol values that can be reached from the corresponding branch in the tree and of the symbol values that can be reached from the node that the particular branch emerges from, as illustrated in Figure 1.8.

Following the assignment of gamma values, a forward recursion from the state $S_{(0,0)}$ is employed to allocate so-called alpha values [10] to the trellis states. The alpha value

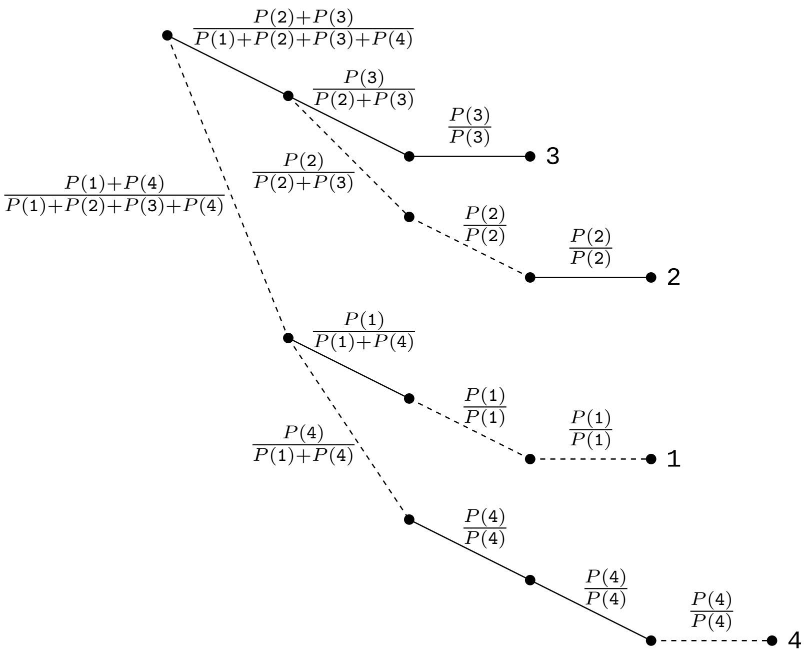


Figure 1.8: Tree representation of the VLEC codebook **VLEC** provided in Table 1.1. Each branch is labelled with the conditional probability $P(T|\mathrm{fr}(T))$ that is assigned to the corresponding transitions within the bit-based VLEC trellis.

$\alpha(S_{(\ddot{\imath},\ddot{n})})$ assigned to a particular state $S_{(\ddot{\imath},\ddot{n})}$ is given by considering all of the preceeding *a priori* LLRs according to

$$\alpha[S_{(\ddot{\imath},\ddot{n})}] = \begin{cases} 1 & \text{if } \ddot{\imath} = 0 \text{ and } \ddot{n} = 0 \\ \sum_{T\in\mathrm{to}(S_{(\ddot{\imath},\ddot{n})})} \gamma(T)\cdot\alpha[\mathrm{fr}(T)] & \text{otherwise} \end{cases}, \quad (1.13)$$

where $\mathrm{to}(S_{(\ddot{\imath},\ddot{n})})$ is the set of all transitions that merge into the state $S_{(\ddot{\imath},\ddot{n})}$ and $\mathrm{fr}(T)$ is the state that the transition T emerges from.

Similarly, a backwards recursion from the state $S_{(I,0)}$ is employed to allocate so-called beta values [10] to the trellis states. The beta value $\beta(S_{(\ddot{\imath},\ddot{n})})$ assigned to a particular state $S_{(\ddot{\imath},\ddot{n})}$ is given by considering all of the forthcoming *a priori* LLRs according to

$$\beta[S_{(\ddot{\imath},\ddot{n})}] = \begin{cases} 1 & \text{if } \ddot{\imath} = I \text{ and } \ddot{n} = 0 \\ \sum_{T\in\mathrm{fr}(S_{(\ddot{\imath},\ddot{n})})} \gamma(T)\cdot\beta[\mathrm{to}(T)] & \text{otherwise} \end{cases}, \quad (1.14)$$

where $\mathrm{fr}(S_{(\ddot{\imath},\ddot{n})})$ is the set of all transitions that emerge from the state $S_{(\ddot{\imath},\ddot{n})}$ and $\mathrm{to}(T)$ is the state that the transition T merges into.

Once the related recursions have been completed, an *a posteriori* probability is calculated for each transition in the trellis. The *a posteriori* probability $P_p(T)$ of a particular transition T is given as

$$P_p(T) = \frac{1}{C_1} \cdot \alpha[\text{fr}(T)] \cdot \gamma(T) \cdot \beta[\text{to}(T)], \tag{1.15}$$

where the transition T emerges from the state $\text{fr}(T)$ and merges into the state $\text{to}(T)$, while C_1 is a constant that is common to all *a posteriori* transition probabilities.

Finally, a so-called *a posteriori* LLR frame $L_p(\mathbf{u}) = \{L_p(u_i)\}_{i=1}^{I}$ is output, where the *a posteriori* probability $P_p(u_i = b)$ of a particular bit u_i assuming a particular value b is given by

$$P_p(u_i = b) = \sum_{\left\{T \middle| {i^T = i, \atop b^T = b}\right\}} P_p(T) \tag{1.16}$$

and the corresponding *a posteriori* LLR $L_p(u_i)$ may be obtained with reference to (1.9).

Note that the BCJR algorithm can only provide optimal *a posteriori* LLRs, if the *a priori* LLRs in the frame $L_a(\mathbf{u})$ are independent. This is because the multiplications within (1.13) – (1.15) assume that the joint *a priori* probability of two bits taking particular values may be obtained as the product of the appropriate separate *a priori* bit value probabilities.

The video decoder of Chapter 2 will apply a novel modification of the BCJR algorithm to the novel symbol-based trellis that was briefly introduced in Section 1.2.6.3. This will guarantee the reconstruction of legitimate – although not necessarily error-free – video encoded information, eliminating the need to discard invalid information that is typical in conventional video decoders. Furthermore, the proposed novel modification to the BCJR algorithm will be shown to generate soft information pertaining to the encoded video blocks, facilitating their MMSE reconstruction.

1.3.2.3 MAP SIHO VLEC sequence estimation

Recall that the ML SIHO VLEC sequence estimation algorithm of Section 1.3.2.1 replaces the transition metrics in the Viterbi algorithm of Section 1.2.6.2 with those of (1.10). Here, the replacement metric $m(T)$ of each transition T is provided by its *a priori* probability $P_a(T) = P(u_{i^T} = b^T)$, which may be obtained directly from the corresponding *a priori* LLR $L_a(u_{i^T})$ in the input frame $L_a(\mathbf{u}) = \{L_a(u_i)\}_{i=1}^{I}$ with reference to (1.9). However, we can expect to improve the BER of the resultant reconstructed bit frame $\tilde{\mathbf{u}}$, if we first apply the VLEC code constraints to the *a priori* transition probabilities. This may be achieved using the BCJR algorithm of Section 1.3.2.2 in order to obtain the *a posteriori* transition probabilities $P_p(T)$ of (1.15). Hence, in the so-called MAP VLEC sequence estimation algorithm [71], each transition metric $m(T)$

employed during the Viterbi algorithm of Section 1.2.6.2 is set to the corresponding *a posteriori* transition probability $P_p(T)$ of (1.15).

As in the ML SIHO VLEC sequence estimation algorithm of Section 1.3.2.1, the revised cumulative transition metric $M(T)$ of (1.11) is employed during MAP VLEC sequence estimation. For this reason the MAP VLEC sequence estimation algorithm assumes the independence of the *a priori* LLRs in the frame $L_a(\mathbf{u})$, as described in Section 1.3.2.1. As is the case for the ML SIHO VLEC sequence estimation algorithm of Section 1.3.2.1 and the APP SISO VLEC algorithm of Section 1.3.2.2, the optimality of the MAP VLEC sequence estimation algorithm is no longer valid, if the *a priori* LLRs are not independent.

1.3.2.4 Computational complexity and other implementational issues

The above-mentioned SISO and sequence estimation algorithms operate on the basis of conventional probabilities. However, these have a high dynamic range, typically having values that vary by many orders of magnitude. For this reason, accurate floating-point processing is required, which nonetheless loses precision when successive operations are performed on small probabilities, as is typical in the algorithms considered [87]. This motivates the transformation of the probabilities into the logarithmic domain. The resultant logarithmic-domain probabilities have a significantly reduced dynamic range, allowing the employment of fixed-point, rather than floating-point processing. This is associated with a complexity saving, since fixed-point Arithmetic and Logic Units (ALUs) are typically significantly simpler than their floating-point equivalents.

Furthermore, operation within the logarithmic domain avoids the exponentials and logarithms that are required to convert between bit value probabilities $P(u_i = b)$ and LLRs $L(u_i)$, according to (1.9). However, the maximum finding, addition and multiplication operations that are employed by the above-mentioned SISO and sequence estimation algorithms must be converted into the logarithmic domain in order to facilitate this. Let us consider operations performed upon two probabilities p_1 and p_2 and the equivalent operations that must be performed upon their logarithmic-domain counterparts $P_1 = \ln(p_1)$ and $P_2 = \ln(p_2)$.

The multiplication of two probabilities may be achieved by the addition of their logarithmic-domain counterparts

$$\ln(p_1 \times p_2) = P_1 + P_2, \tag{1.17}$$

which may be achieved by a fixed-point Arithmetic and Logic Unit (ALU) in a single clock cycle [87]. By contrast, floating-point ALUs typically require a number of clock cycles for the multiplication of the probabilities. Hence, a significant complexity saving can be attained by performing the multiplications of the SISO and sequence estimation algorithms of Sections 1.3.2.1 – 1.3.2.3 in the logarithmic domain.

The maximum-finding operations remain similar when operating in the logarithmic domain, since

$$\ln[\max(p_1, p_2)] = \max(P_1, P_2), \tag{1.18}$$

where p_1 and p_2 are both positive. Note that the related compare and select operation can be typically performed by a fixed-point ALU in a single clock cycle.

However, the addition of two probabilities requires the use of the Jacobian logarithm [88] in the logarithmic domain

$$\ln(p_1 + p_2) = \max(P_1, P_2) + \ln(1 + e^{-|P_1 - P_2|}), \tag{1.19}$$

which cannot be readily performed by a simple ALU, since it includes both logarithms and exponentials. This motivates the employment of the Max-Log-MAP approach [88], in which the Jacobian approximation

$$\ln(p_1 + p_2) \approx \max(P_1, P_2) \tag{1.20}$$

is employed instead of the Jacobian logarithm of (1.19). Since this compare and select operation can be typically performed by a fixed-point ALU in a single clock cycle, its employment mitigates the complexity associated with the Jacobian logarithm of (1.19).

Note that the approximation introduced by employing the Jacobian approximation of (1.20) instead of the Jacobian logarithm of (1.19) during the SISO and sequence estimation algorithms of Sections 1.3.2.1 – 1.3.2.3 inevitably erodes their optimality. The associated performance degradation is typically significant, motivating the employment of the Log-MAP approach [88], which strikes a compromise between the complexity of (1.20) and the sub-optimality of (1.19). This employs the eight-entry lookup Table 1.4 to approximate the term $\ln(1 + e^{-|P_1 - P_2|})$ in the Jacobian logarithm of (1.19). The lookup Table 1.4 is indexed using $|P_1 - P_2|$, which may be obtained by a fixed-point ALU in the same clock cycle that provides $\max(P_1, P_2)$ for (1.19).

Note that the value of $|P_1 - P_2|$ may be compared to each of the ranges provided in the lookup Table 1.4 in the order from top to bottom and curtailing any further operations, once the appropriate entry has been found. Since we may expect small differences between P_1 and P_2 to be relatively rare, this approach may be expected to reduce the average number of comparisons required. Note that the ALU can typically

$\lvert P_1 - P_2 \rvert$	$\ln(1 + e^{-\lvert P_1 - P_2 \rvert})$
≥ 3.70	0.00
$\in [2.25, 3.70)$	0.05
$\in [1.50, 2.25)$	0.15
$\in [1.05, 1.50)$	0.25
$\in [0.70, 1.05)$	0.35
$\in [0.43, 0.70)$	0.45
$\in [0.20, 0.43)$	0.55
$\in [0.00, 0.20)$	0.65

Table 1.4: Lookup table employed to correct the Jacobian approximation in the Log-MAP algorithm.

perform one comparison per clock cycle, requiring between one and eight clock cycles to obtain an approximation for $\ln(1 + e^{-|P_1 - P_2|})$. Once this is obtained, another clock cycle is required to add it to $\max(P_1, P_2)$, completing the approximation of (1.19). Hence, the Jacobian logarithm requires a total of three to ten clock cycles, when the Log-MAP approach is employed. Note that this is significantly higher than the single clock cycle required to perform the addition of (1.17) and the maximum operation of (1.18). However, the Jacobian logarithm is typically invoked relatively rarely, when the SISO and sequence estimation algorithms of Sections 1.3.2.1 – 1.3.2.3 are performed in the logarithmic domain. For this reason, the complexity saving associated with the addition of (1.17) tends to be the dominating factor.

Note that only Add, Compare and Select (ACS) operations are required when the SISO and sequence estimation algorithms are performed in the logarithmic domain and the eight-entry lookup Table 1.4 is employed to correct the Jacobian approximation in the Log-MAP algorithm. Hence, a decoding algorithm's computational complexity may be quantified by the number of ACS operations it performs or, equivalently, by the number of fixed-point ALU clock cycles it requires. In Chapter 3, we shall employ ACS operation counts as the basis of a novel plot for characterising how the decoding complexity imposed varies with the channel SNR.

Further computational complexity reductions can be achieved using the approaches of the T-BCJR and M-BCJR algorithms [89], which restrict the complexity by considering a limited number of trellis transitions. More explicitly, following the calculation of the BCJR algorithm's gamma values using (1.12), transitions are pruned from the trellis if their gamma value is below a particular threshold value T, in the T-BCJR algorithm. Clearly, the higher the threshold value T, the more transitions are pruned from the trellis, simplifying the remaining BCJR calculations and reducing the overall complexity, at the cost of eroding the algorithm's optimality. Similarly, the M-BCJR algorithm prunes all but M number of the transitions that consider each bit in the VLEC encoded frame $\mathbf{u}$.

1.3.3 Concatenated codes

As described above, turbo codes [11], as well as schemes employing the more general 'turbo principle' [82, 83], employ an iterative exchange of soft information between a number of decoders. We now discuss techniques for arranging or 'concatenating' these codes.

1.3.3.1 Parallel concatenation

The parallel concatenation of two codes is exemplified by the classic turbo code [11] of Figure 1.9, which employs two CCs [8]. Here, the upper CC encoder encodes the bit frame $\mathbf{u}$ in order to generate the CC-encoded bit frame $\mathbf{v}^1$. Meanwhile, the lower CC encoder generates the CC-encoded bit frame $\mathbf{v}^2$ by encoding a scrambled version $\mathbf{u}'$ of the bit frame $\mathbf{u}$ that has been interleaved in the block π of Figure 1.9. Note that typically both CC encoders have an identical coding rate $R_{\text{CC}} < 1$ and design, which may be realised using a Linear Feedback Shift Register (LFSR) [8]. Following this, the turbo-encoded bit frame $\mathbf{v}$ is obtained by multiplexing the CC-encoded bit frames $\mathbf{v}^1$ and $\mathbf{v}^2$, while puncturing some bits in order to increase the overall turbo coding rate R_{turbo} from $R_{\text{CC}}/2$.

In analogy to the turbo encoder, an APP SISO turbo decoder may be constructed from a parallel concatenation of two APP SISO CC decoders, as shown in Figure 1.9. Similarly to the APP SISO VLEC decoder of Section 1.3.2.2, these apply the BCJR algorithm to a specific trellis [9], which encompasses the associated code constraints. As shown in Figure 1.9, the upper APP SISO CC decoder considers both an *a priori* LLR frame $L_a^u(\mathbf{u})$ pertaining to the bit frame $\mathbf{u}$ and an *a priori* LLRframe $L_a^u(\mathbf{v}^1)$ pertaining to the bit frame $\mathbf{v}^1$ during the generation of the *a posteriori* LLR frame $L_p^u(\mathbf{u})$. Likewise, the lower APP SISO CC decoder of Figure 1.9 considers both $L_a^l(\mathbf{u}')$ and $L_a^l(\mathbf{v}^2)$ during the generation of $L_p^l(\mathbf{u}')$. Note that these are in contrast to the APP SISO VLEC decoder of Section 1.3.2.2, which has only a single input *a priori* LLR frame.

The APP SISO turbo decoder of Figure 1.9 commences its operation by demultiplexing and depuncturing the *a priori* LLR frame $L_a(\mathbf{v})$. Here, zero values are allocated to any LLRs within the frames $L_a^u(\mathbf{v}^1)$ and $L_a^l(\mathbf{v}^2)$ that correspond to bits that were punctured in the turbo encoder. These zero-valued LLRs indicate the lack of *a priori* knowledge and the total uncertainty concerning the logical values of the pertained bits, with logical zero and one values being equally likely.

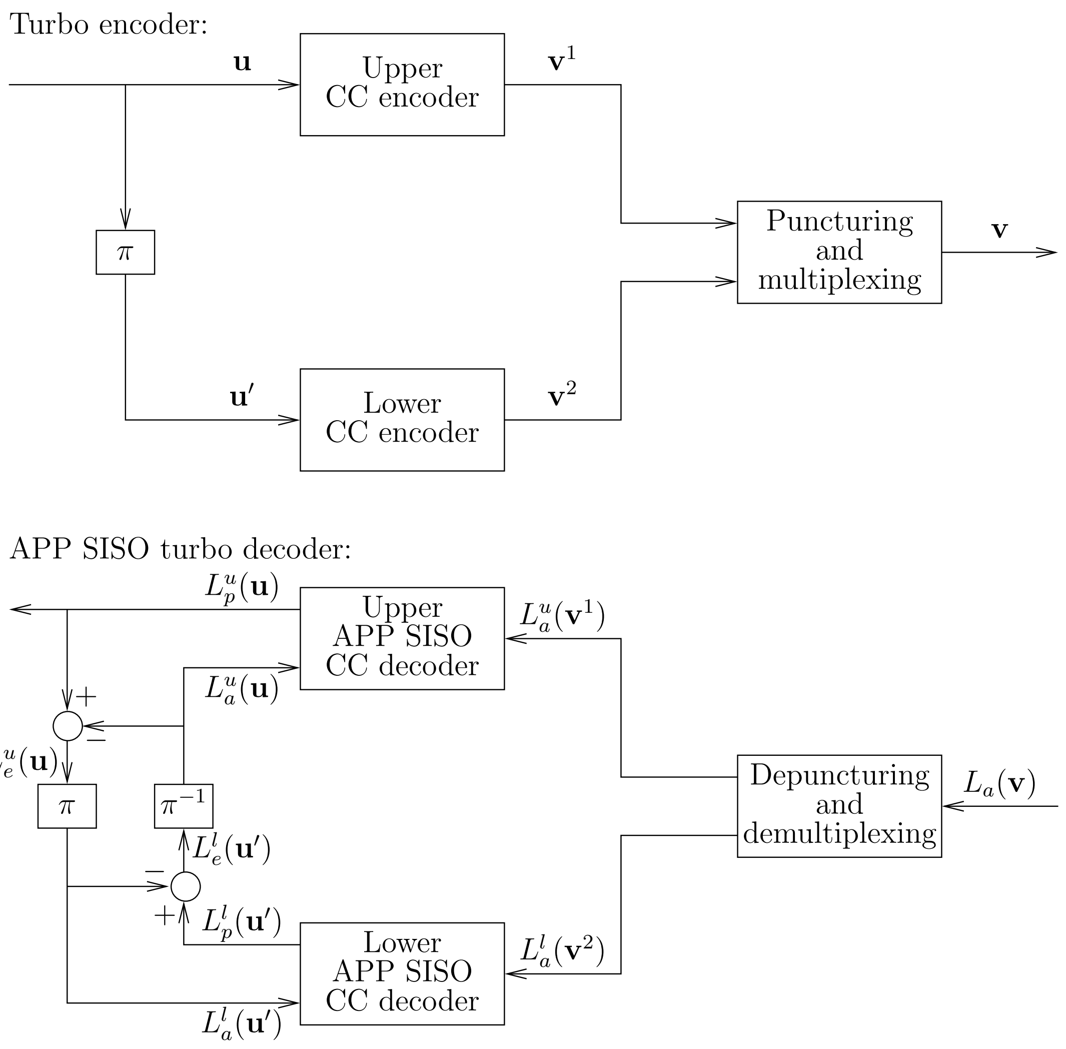


Figure 1.9: Parallel concatenation and iterative decoding of the two CCs of a turbo code.

Following this, APP SISO turbo decoding proceeds in an iterative manner. During this process, the APP SISO CC decoders of Figure 1.9 assist each other's operation, allowing them to dispel more and more uncertainty about the logical values of the bits in the frame $\mathbf{u}$. This is achieved with the alternated activation of the upper and lower APP SISO CC decoders of Figure 1.9 and the exchange of any new, or extrinsic, information that they can infer. More specifically, the upper APP SISO CC decoder provides the *a priori* LLR frame $L_a^l(\mathbf{u}')$ to the lower decoder, which reciprocates with the *a priori* LLR frame $L_a^u(\mathbf{u})$, as shown in Figure 1.9. Once the *a posteriori* LLR frames $L_p^u(\mathbf{u})$ and $L_p^l(\mathbf{u}')$ have been obtained, their new extrinsic content may be extracted with the subtraction of the old *a priori* LLR frames $L_a^u(\mathbf{u})$ and $L_a^l(\mathbf{u}')$, respectively. As shown in Figure 1.9, this results in the extrinsic LLR frames $L_e^u(\mathbf{u})$ and $L_e^l(\mathbf{u}')$, respectively. Following interleaving π or de-interleaving π^{-1} as appropriate, the *a priori* LLR frames $L_a^u(\mathbf{u})$ and $L_a^l(\mathbf{u}')$ generated for the next decoding iteration are obtained.

Note that at the start of the iterative decoding process, the lower APP SISO CC decoder of Figure 1.9 has not yet been invoked and hence the *a priori* LLR frame $L_a^u(\mathbf{u})$ is not available. In this case, an *a priori* LLR frame $L_a^u(\mathbf{u})$ comprising zero-valued LLRs is provided for the upper APP SISO CC decoder, indicating total uncertainty in the logical values of the pertained bits.

After a number of decoding iterations, all the *a priori* information has been exploited and, hence, no further improvements may be obtained by additional decoding iterations. In Section 1.3.4, we shall discuss methods designed for characterising this iterative decoding convergence. Once iterative decoding has been completed, the most recently obtained *a posteriori* LLR frame $L_p^u(\mathbf{u})$ is subjected to a hard decision and output, as shown in Figure 1.9.

Note that the interleaver π has three roles in the turbo code [90], namely

- to provide the lower CC encoder with a bit frame $\mathbf{u}'$ that is different from the one $\mathbf{u}$ encoded by the upper CC encoder,
- to scramble the positions of any strong LLRs that appear in groups within the extrinsic LLR frames $L_e^u(\mathbf{u})$ and $L_e^l(\mathbf{u}')$, hence allowing them to improve the confidence of decoding the entire *a priori* LLR frames $L_a^u(\mathbf{u})$ and $L_a^l(\mathbf{u}')$, and finally
- to mitigate the dependencies between neighbouring LLRs in the *a priori* frames $L_a^u(\mathbf{u})$ and $L_a^l(\mathbf{u}')$, contributing towards maintaining the BCJR algorithm's associated assumption of exploiting independent *a priori* information, as described in Section 1.3.2.2.

Clearly, the interleaver's ability to perform these functions is commensurate with its length. However, long interleavers are associated with long decoding latencies, since the decoding of the *a posteriori* LLR frame $L_p^u(\mathbf{u})$ cannot be completed until the entire *a posteriori* LLR frame $L_a(\mathbf{v})$ has been received.

Turbo codes have found application in the 3rd Generation (3G) wireless standards, satellite communication standards and wireless networking standards, among others [91]. In [92–94], the parallel concatenation and iterative decoding of block codes [95] was proposed. As another concatenated design alternative, the Turbo Trellis Coded Modulation (TTCM) scheme of [96] employs the parallel concatenation of two Trellis Coded Modulation (TCM) [97] schemes. Finally, the parallel concatenation of source and channel codes was considered in [98].

1.3.3.2 Serial concatenation

Following the introduction of turbo codes, relying on the parallel concatenation of codes, their serial concatenation was proposed in [99, 100]. This is exemplified by the serial concatenation of an outer VLEC and an inner CC codec in the joint source and channel coding scheme of Figure 1.10. Here, the VLEC encoder represents the source symbol frame **s** with the VLEC-encoded bit frame **u**. This is interleaved in the block π of Figure 1.10 and the resultant frame $\mathbf{u}'$ is provided to the CC encoder, which generates the CC-encoded bit frame **v**. The overall coding rate is $R_{\text{VLEC}} \cdot R_{\text{CC}}$, where R_{VLEC} is the VLEC coding rate and R_{CC} is the CC coding rate.

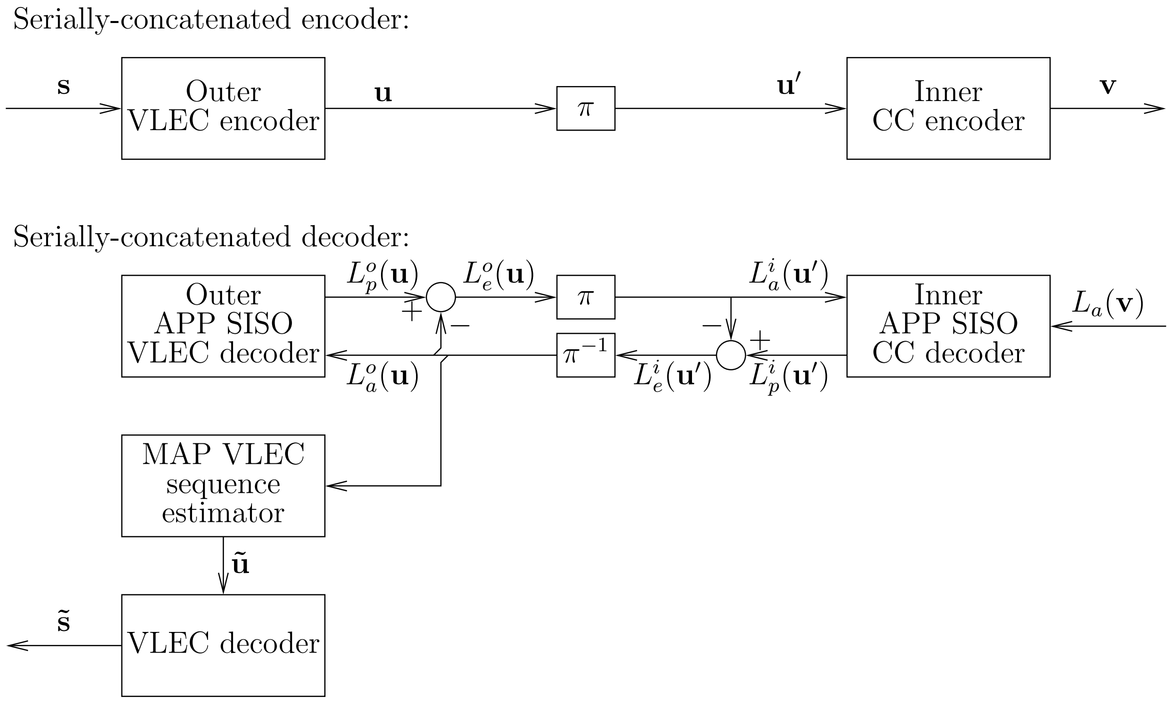


Figure 1.10: Serial concatenation and iterative decoding of a VLEC and a CC.

In the receiver, the APP SISO VLEC decoder is serially concatenated with the APP SISO CC decoder, as shown in Figure 1.10. Similarly to the APP SISO CC decoders employed in the parallel concatenation of Figure 1.9, the inner APP SISO CC decoder generates the *a posteriori* LLR frame $L^i_p(\mathbf{u}')$ by considering the *a priori* LLR frames $L^i_a(\mathbf{u}')$ and $L^i_a(\mathbf{v})$. Similarly, the outer APP SISO VLEC decoder generates the *a posteriori* LLR frame $L^o_p(\mathbf{u})$ by considering the *a priori* LLR frame $L^o_a(\mathbf{u})$ of Figure 1.10, as described in Section 1.3.2.2.

In analogy with the iterative turbo decoding process of Figure 1.9 in Section 1.3.3.1, the APP SISO VLEC and CC decoders assist each other's operation, allowing them to dispel more and more uncertainty about the logical values of the bits in the frame **u** of Figure 1.10. This is achieved with the alternated activation of the APP SISO decoders and the exchange of any new extrinsic information that they can infer. More specifically, the inner APP SISO CC decoder of Figure 1.10 provides the *a priori* LLR frame $L_a^o(\mathbf{u})$ to the APP SISO VLEC decoder, which reciprocates with the *a priori* LLR frame $L_a^i(\mathbf{u}')$, as shown in Figure 1.10. Once the *a posteriori* LLR frames $L_p^i(\mathbf{u}')$ and $L_p^o(\mathbf{u})$ have been obtained, their new extrinsic content may be extracted with the aid of subtracting the old *a priori* LLR frames $L_a^i(\mathbf{u}')$ and $L_a^o(\mathbf{u})$, respectively. As shown in Figure 1.10, this results in the extrinsic LLR frames $L_e^i(\mathbf{u}')$ and $L_e^o(\mathbf{u})$, respectively. Following interleaving π or de-interleaving π^{-1} as appropriate, the *a priori* LLR frames $L_a^i(\mathbf{u}')$ and $L_a^o(\mathbf{u})$ generated for the next decoding iteration are obtained.

Note that at the start of the iterative decoding process, the outer APP SISO VLEC decoder of Figure 1.10 has not yet been invoked and hence the *a priori* LLR frame $L_a^i(\mathbf{u}')$ is not available. In this case, an *a priori* LLR frame $L_a^i(\mathbf{u}')$ comprising zero-valued LLRs is forwarded to the inner APP SISO CC decoder, indicating total uncertainty in the logical values of the pertained bits.

Like the turbo decoding process of Section 1.3.3.1, the iterative decoding process will be unable to glean any further improvements after a particular number of iterations. In Section 1.3.4, we shall discuss methods designed for characterising this iterative decoding convergence. Once iterative decoding has been completed, the most recently obtained *a priori* LLR frame $L_a^o(\mathbf{u})$ is provided to the MAP VLEC sequence estimator, as shown in Figure 1.10. As described in Section 1.3.2.3, this generates the reconstructed bit frame $\tilde{\mathbf{u}}$, which may be VLEC decoded to obtain the reconstructed source symbol frame $\tilde{\mathbf{s}}$, as shown in Figure 1.10.

Note that the interleaver π has two roles in the serial concatenated scheme of Figure 1.10 [90], namely

- to scramble the positions of any strong LLRs that appear in groups within the extrinsic LLR frames $L_e^i(\mathbf{u}')$ and $L_e^o(\mathbf{u})$, hence allowing them to improve the confidence of decoding the entire *a priori* LLR frames $L_a^i(\mathbf{u}')$ and $L_a^o(\mathbf{u})$, and
- to mitigate the dependencies between neighbouring LLRs in the *a priori* frames $L_a^i(\mathbf{u}')$ and $L_a^o(\mathbf{u})$, contributing towards maintaining the BCJR algorithm's associated assumption of exploiting independent *a priori* information, as described in Section 1.3.2.2.

As in parallel concatenated schemes, the interleaver's ability to perform these functions is commensurate with its length. However, long interleavers are associated with long decoding latencies, since the reconstructed source symbol frame $\tilde{\mathbf{s}}$ of Figure 1.10 cannot be obtained until the entire *a posteriori* LLR frame $L_a(\mathbf{v})$ has been received.

In addition to the joint source and channel coding application [53–55, 101–104] outlined above, many other serially concatenated schemes have been considered in the literature. Powerful channel codes can be realised by the serial concatenatation and iterative decoding of simple channel codes. For example, an outer block code [95] may be concatenated with an inner CC [8], as demonstrated in [99]. Furthermore, the serial concatenation and iterative decoding of repeat and accumulate codes was demonstrated in [105]. An LDPC decoder [12–14] may be considered to be a serial concatenation of an inner decoder that considers the variable nodes of the bipartite graph [15] and an outer decoder that considers the check nodes.

The inner decoder in a serially concatenated scheme can also fulfil roles other than channel coding. In [106], an inner equaliser was employed to correct the InterSymbol Interference (ISI) imposed by a dispersive channel. The employment of a demodulator as an inner decoder was demonstrated in an iteratively decoded version [107,108] of Bit-Interleaved Coded Modulation (BICM) [109], as well as for serial concatenation with an outer source codec in [110]. Similarly, TCM provided the inner code in [111, 112]. Other examples of inner-code applications include Multi-User Detection (MUD) [113] and Multiple-In Multiple-Out (MIMO) channel detection [114].

Note that it is possible to serially concatenate more than two codes. This was demonstated in [115], where a half-rate CC, a unity-rate CC (or Unity Rate Code (URC) [112]) and an ISI equaliser were serially concatenated. In this case, each code performed a different role and indeed, the employment of a three-stage concatenation was necessitated. However, in scenarios where all roles can be accomplished using a two-stage concatenation, there is no benefit in including additional codes and creating a serial concatenation of three or more codes, as noted in [116].

1.3.4 Iterative decoding convergence

As described in Section 1.3.3, turbo codes [11], as well as schemes employing the more general ‘turbo principle’ [82, 83], operate on the basis of an iterative exchange of increasingly reliable soft information between a number of decoders, which continues until convergence is achieved. We now consider methods of quantifying the reliability of this soft information and characterising how it evolves, as the iterative decoding process converges.

1.3.4.1 Mutual information

Early methods devised for quantifying the reliability of soft information simply employed the SNRs [117, 118] and SOft BITs (SOBITs) [119]. However, more recently, the mutual information between the soft information and the corresponding logical values of the bits [120] has become the favoured option.

As described in [120], the mutual information between an LLR frame $L(\mathbf{u})$ and the corresponding frame $\mathbf{u}$ of logical values depends on the distribution of the LLR values. More specifically, if the distribution of the LLR values that correspond to zero-valued bits is equal to that of the LLR values pertaining to unity-valued bits, then the mutual information will be zero. In this case, the LLR values are totally unreliable and the selection of a bit's logical value based upon the sign of the corresponding LLR will only give the correct answer 50% of the time. Note that a zero-valued mutual information will also be obtained when all LLRs have a zero value, indicating equal confidences in both the logical bit values of zero and one.

As the reliability of the LLRs increases, the two LLR distributions will move apart and will only partially overlap, giving a mutual information that is greater than zero. Here, the selection of a bit value based upon the sign of the corresponding LLR will give the correct answer more often then not. When the distributions become completely separated, the resultant mutual information will be equal to the bit entropy, which is unity in the typical case where zero- and unity-value bits occur with equal probability. In this case, the correct bit values will always be obtained, when the decision is based upon the sign of the corresponding LLRs.

This is illustrated for an *a priori* LLR frame $L_a(\mathbf{u})$ in Figure 1.11. Here, the LLRs can be seen to have Gaussian distributions, although other distributions may be encountered in practice, depending on the transmission channel employed. In Figure 1.11, the distribution of *a priori* LLRs $L_a(u_i)$ corresponding to zero-valued bits u_i can be seen to move away from the distribution of LLRs pertaining to logical one-valued bits as the mutual information I_a between the LLR frame $L_a(\mathbf{u})$ and the bit frame $\mathbf{u}$ increases.

In practice, the *a priori* distributions may be approximated using histograms of the recorded LLR values, which must be categorised according to the logical bit values. Alternatively, the averaging method of [121] is capable of calculating the mutual information of an LLR frame $L(\mathbf{u})$ without considering the corresponding bit frame $\mathbf{u}$, provided that $L(\mathbf{u})$ is generated by an optimal APP decoder.

1.3.4.2 EXIT function

A particular SISO decoder's operation in an iterative decoding arrangement may be characterised by its EXtrinsic Information Transfer (EXIT) function $I_e(I_a)$ [120], as exemplified in Figure 1.12. A specific point in a SISO decoder's EXIT function quantifies the mutual information $I_e \in [0, 1]$ between the logical bit values and the extrinsic

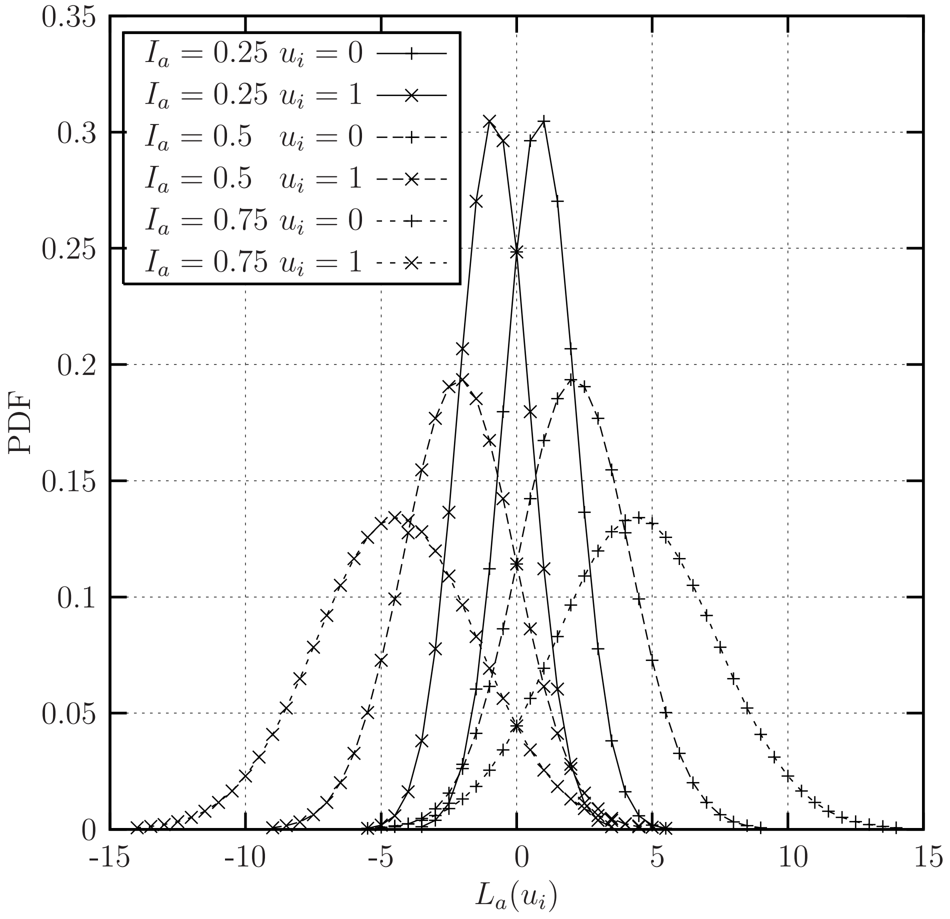


Figure 1.11: Gaussian distributed LLRs $L_a(u_i)$ corresponding to zero- and unity-valued bits u_i for various mutual informations I_a.

LLR frame $L_e(\mathbf{u})$ that is output by the SISO decoder when it is provided with an *a priori* LLR frame $L_a(\mathbf{u})$ having a particular mutual information $I_a \in [0, 1]$. This may be investigated by considering the logical bit values in the frame $\mathbf{u}$ and generating a corresponding synthetic frame of uncorrelated Gaussian distributed *a priori* LLRs $L_a(\mathbf{u})$, having a particular mutual information $I_a \in [0, 1]$ [120], as illustrated in Figure 1.11. This synthetic LLR frame $L_a(\mathbf{u})$ may then be input to the SISO decoder and the mutual information $I_e \in [0, 1]$ of the resultant extrinsic LLR frame can be recorded. By repeating this process for a number of *a priori* mutual information values in the range of $I_a \in [0, 1]$, the SISO decoder's EXIT function can be plotted. To illustrate this, Figure 1.12 provides EXIT functions for each of the codebooks **Huff**, **RVLC** and **VLEC** provided in Table 1.1, when employed as the basis of the APP SISO VLC decoder of Section 1.3.2.2.

Observe in Figure 1.12, that the VLEC codebook **VLEC** generates the most reliable extrinsic information, having the highest mutual information I_e, at all values of *a priori* mutual information I_a. This may be attributed to the VLEC codebook's high level of

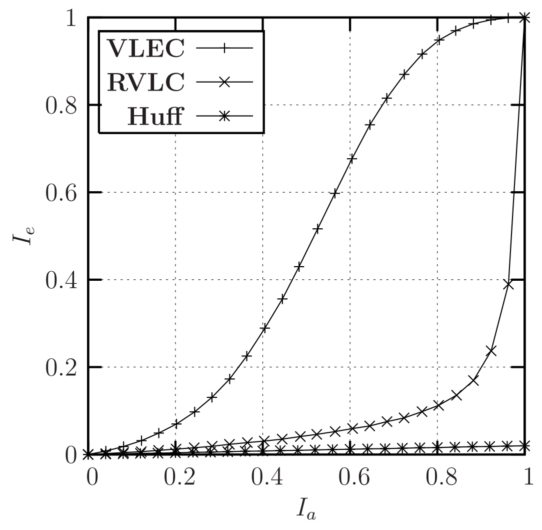


Figure 1.12: EXIT functions characterising the APP SISO VLC decoder of Section 1.3.2.2 when each of the codebooks **Huff**, **RVLC** and **VLEC** provided in Table 1.1 is employed.

redundancy and low coding rate of $R(\textbf{VLEC}) = 0.5$, which equips it with the highest error correction capability, as described in Section 1.2.6.4. By contrast, the Huffman codebook **Huff** introduces very little redundancy and has a near-unity coding rate of $R(\textbf{Huff}) = 0.96$, which renders it unable to glean much extrinsic information, as shown in Figure 1.12.

Also note that the EXIT functions corresponding to the codebooks **VLEC** and **RVLC** reach the $(1, 1)$ point in the top-right hand corner of Figure 1.12. Hence, when provided with an *a priori* LLR frame $L_a(\mathbf{u})$ having a perfect unity mutual information $I_a = 1$, these codebooks can generate unity extrinsic mutual information $I_e = 1$. This indicates that the associated extrinsic LLR frame $L_e(\mathbf{u})$ represents the bit values of the frame $\mathbf{u}$ with absolute certainty and without error, as described in Section 1.3.4.1. By contrast, an APP SISO decoder employing the Huffman codebook **Huff** is never able to generate absolutely certain extrinsic information, even when it is provided with absolutely certain perfect *a priori* information, as shown in Figure 1.12. We shall detail the reason for this discrepancy, as well as its effect on the iterative decoding process in Section 1.3.4.3.

Recall that, in addition to the *a priori* LLR frame $L^i_a(\mathbf{u}')$, the inner APP SISO CC decoder of Figure 1.10 generates the *extrinsic* LLR frame $L^i_e(\mathbf{u}')$ with consideration of a second *a priori* LLR frame $L^i_a(\mathbf{v})$, which pertains to the CC-encoded bit frame $\mathbf{v}$. Note that this is also the case for the upper and lower APP SISO CC decoders in the turbo decoder of Figure 1.9. For this reason, the EXIT function $I_e(I_a)$ of these APP SISO CC decoders is parameterised by the reliability of the *a priori* LLR frame

$L_a^i(\mathbf{v})$ or, in the case where this is provided by a demodulator, by the channel SNR. This is illustrated in Figure 1.13, which depicts the EXIT functions of a unity-rate CC (or URC [112]) that is employed to protect transmissions over a BPSK-modulated uncorrelated narrowband Rayleigh fading channel, having a range of SNRs [84]. Rather than protecting the transmissions by introducing redundancy, the URC scheme has an Infinite Impulse Response (IIR) and hence may be viewed as a scrambler, offering iterative decoding performance improvements in a manner similar to interleavers.

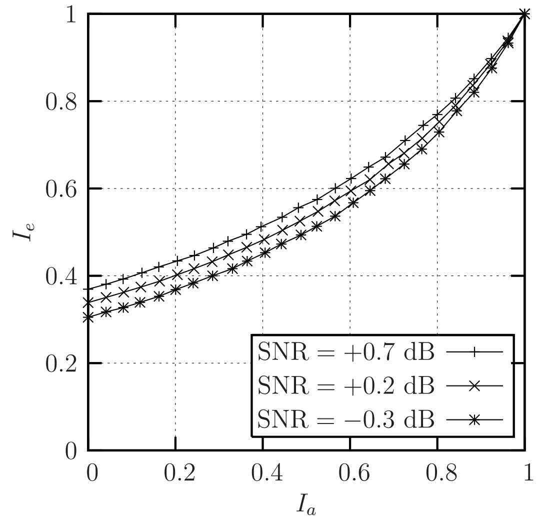


Figure 1.13: EXIT functions characterising an APP SISO URC decoder that is employed to protect transmissions over a BPSK-modulated uncorrelated narrowband Rayleigh fading channel, having a range of SNRs.

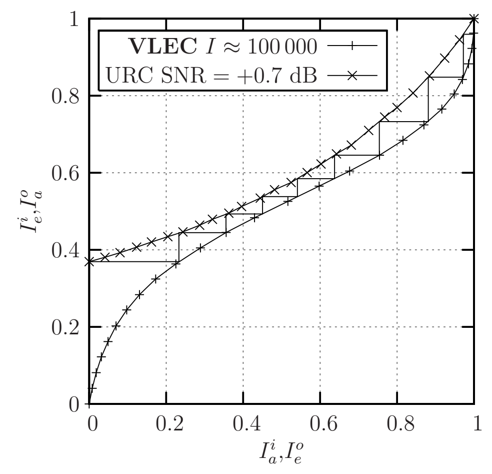
Note that while the VLC EXIT functions start from the $(0, 0)$ point of Figure 1.12, the URC's EXIT functions can be seen to start from a point on the I_e axis at $I_a = 0$ of the plot provided in Figure 1.13. This is because, unlike the APP SISO VLEC decoder, the APP SISO URC decoder is capable of gleaning extrinsic information even in the absence of any *a priori* information. The source of this information is the *a priori* LLR frame $L_a^i(\mathbf{v})$, which is provided by the demodulator. As may be expected, the reliability of this *a priori* information increases with the channel SNR, allowing the URC decoder's EXIT function to emerge from an even higher point on the I_e axis at $I_a = 0$, as shown in Figure 1.13. Also observe that, similarly to the EXIT functions of Figure 1.12 that correspond to the codebooks **VLEC** and **RVLC**, the URC decoder's EXIT function reaches the $(1, 1)$ point in the top-right hand corner of Figure 1.13. Hence, when it is provided with absolutely certain *a priori* information having a unity mutual information $I_a = 1$, the APP SISO URC decoder can generate absolutely certain extrinsic information having a unity mutual information $I_e = 1$.

1.3.4.3 EXIT chart

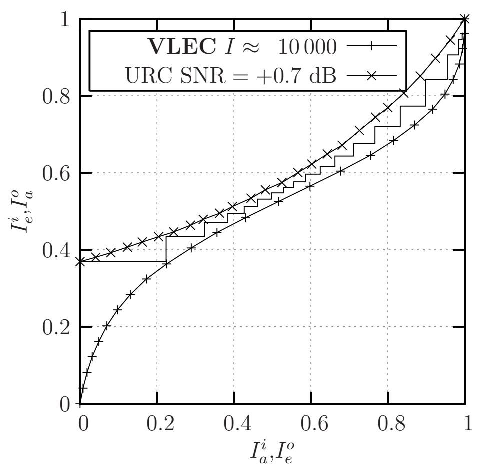
As described in Section 1.3.3, turbo codes [11], as well as other concatenated schemes employing the more general 'turbo principle' [82, 83], operate on the basis of an iterative exchange of increasingly reliable soft information between a number of decoders, which continues until convergence is achieved. An EXIT chart [120] may be employed to predict the convergence behaviour of the iteratively decoded scheme that would be obtained by concatenating two particular APP SISO decoders. This is formed by overlaying the EXIT functions of the two concatenated decoders. However, it is necessary to invert one of the EXIT functions, swapping its I_a and I_e axes. This is because, after interleaving or de-interleaving as appropriate, the extrinsic information generated by each APP SISO decoder becomes the *a priori* information provided for the other concatenated decoder, as shown in Figure 1.9 for a parallel concatenated and in Figure 1.10 for a serial concatenated scheme, respectively.

In the case of a serial concatenation, it is customary to invert the outer EXIT function before overlaying it with the inner EXIT function. Hence, the EXIT chart's x axis represents the inner *a priori* mutual information I_a^i and the outer extrinsic mutual information I_e^o. Meanwhile, the inner extrinsic mutual information I_e^i and the outer *a priori* mutual information I_a^o are represented by the y axis of the EXIT chart. This is exemplified by the EXIT charts of Figure 1.14, in which the URC decoder's EXIT functions of Figure 1.13 are overlaid by the inverted EXIT function corresponding to the VLEC codebook **VLEC** from Figure 1.12.

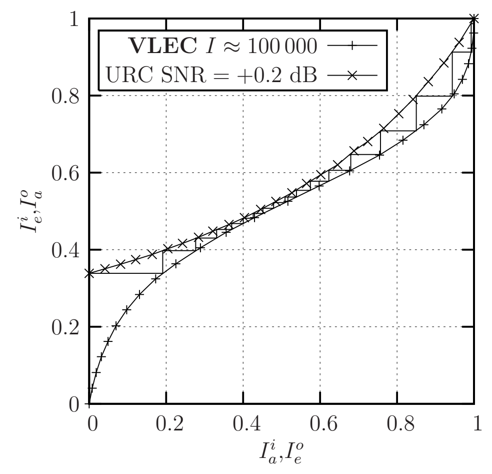
Therefore, the EXIT charts of Figure 1.14 may be employed to predict the iterative decoding convergence behaviour of the serially concatenated scheme shown in Figure 1.10. As described in Section 1.3.3.2, at the commencement of the iterative decoding process, the outer APP SISO VLEC decoder has not yet been invoked and hence the *a priori* LLR frame $L_a^i(\mathbf{u}')$ is unavailable. In this case, an *a priori* LLR frame $L_a^i(\mathbf{u}')$ comprising zero-valued LLRs and having a zero-valued mutual information $I_a^i = 0$ is provided for the inner APP SISO URC decoder. In response, the inner APP SISO URC decoder generates an extrinsic LLR frame $L_e^i(\mathbf{u}')$ having a mutual information of $I_e^i = 0.37$, in the case depicted in Figure 1.14a, where the Rayleigh fading channel's SNR is +0.7 dB. As shown in Figure 1.10, this extrinsic LLR frame is de-interleaved and forwarded to the APP SISO VLEC decoder as the *a priori* LLR frame $L_a^o(\mathbf{u})$, which will also have a mutual information of $I_a^o = 0.37$. *Under the assumption that the EXIT chart of Figure 1.14a accurately models the iterative decoding process*, the APP SISO VLEC decoder will generate an extrinsic LLR frame $L_e^o(\mathbf{u})$ having a mutual information of $I_e^o = 0.23$, since the inverted VLEC EXIT function



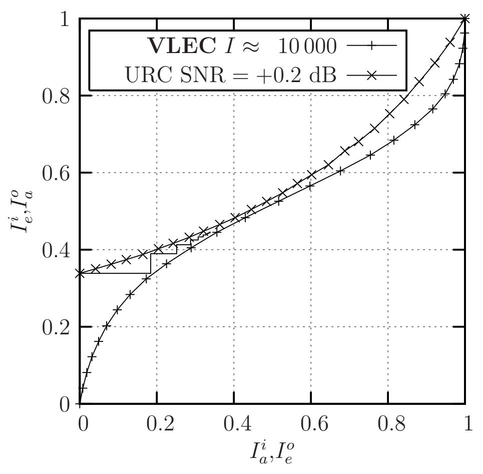
(a)



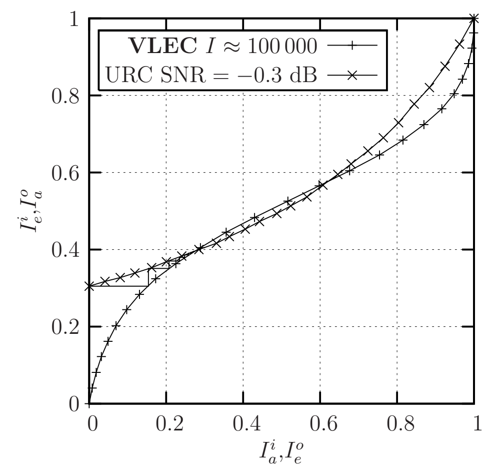
(b)



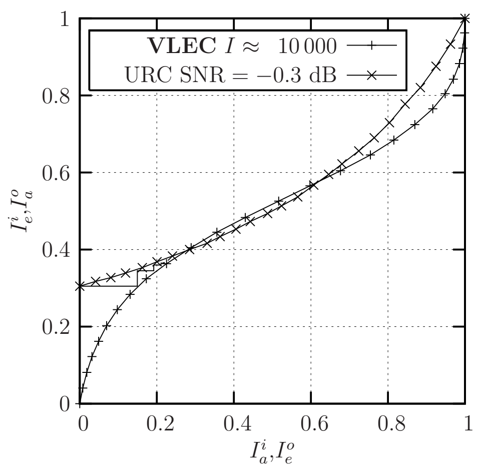
(c)



(d)



(e)



(f)

Figure 1.14: EXIT charts and iterative decoding trajectories for the serial concatenation of a URC with the VLEC code **VLEC** of Table 1.1, as depicted in Figure 1.10. Here, the URC is employed to protect transmissions over a BPSK-modulated Rayleigh fading channel having various SNRs. The iterative decoding trajectories of (a), (c) and (e) correspond to an $I \approx 100\,000$-bit interleaver, while an $I \approx 10\,000$-bit interleaver was employed in (b), (d) and (f).

passes through the $(0.23, 0.37)$ point of the EXIT chart shown in Figure 1.14a. We can therefore represent this operation of the outer APP SISO VLEC decoder using a horizontal line between the $(0, 0.37)$ and the $(0.23, 0.37)$ points of the EXIT chart, as shown in Figure 1.14a. Next, the extrinsic LLR frame $L_e^o(\mathbf{u})$ is interleaved to obtain the *a priori* LLR frame $L_a^i(\mathbf{u}')$ having a mutual information of $I_a^i = 0.23$, which is provided to the inner APP SISO URC decoder. In response, this will generate an extrinsic LLR frame $L_e^i(\mathbf{u}')$ having a mutual information of $I_e^i = 0.44$, *provided that the EXIT chart of Figure 1.14a accurately models the iterative decoding process.* We can therefore represent this operation of the inner APP SISO URC decoder using a vertical line between the $(0.23, 0.37)$ and the $(0.23, 0.44)$ points of the EXIT chart, as shown in Figure 1.14a. Continuing in this fashion, we can complete the iterative decoding trajectory shown in Figure 1.14a.

As alluded to above, an EXIT chart will only accurately model the iterative decoding process, if the step-wise decoding trajectory that is drawn in the manner described above matches that which may be obtained by concatenating the codes, performing iterative decoding and plotting the measured mutual information values. In practice, these trajectories will not match if the BCJR algorithm's assumption of having independent *a priori* LLRs is invalid, as described in Section 1.3.2.2. This is because independent *a priori* LLRs are synthetically generated when drawing the individual EXIT functions of the APP SISO decoders, as described in Section 1.3.4.2. However, when these decoders are concatenated, the *a priori* LLRs of one of the APP SISO decoders are provided by the extrinsic LLRs generated by the other, which are typically not entirely independent. As described in Sections 1.3.3.1 and 1.3.3.2, the concatenated decoders are separated by an interleaver, which attempts to mitigate these dependencies. However, the interleaver's ability to perform this role is commensurate with its length. In the case where a short interleaver is employed, the *a priori* LLRs' residual dependencies will invalidate the assumptions of the BCJR algorithm, degrading the performance of the APP SISO decoders and resulting in a mismatch between the staircase-shaped iterative decoding trajectory and the EXIT functions [122]. This is exemplified by the iterative decoding trajectories of Figures 1.14b, 1.14d and 1.14f, which were obtained using a relatively short interleaver of $I \approx 10\,000$ bits. By contrast, the iterative decoding trajectories obtained using a longer interleaver of $I \approx 100\,000$ bits can be seen to match the EXIT functions seen in Figures 1.14a, 1.14c and 1.14e quite accurately.

Various methods have been proposed for designing interleavers [90] for employment during the iterative decoding of concatenated codes. General pseudo-random interleavers [90], such as those employed to obtain the iterative decoding trajectories of Figure 1.14, rearrange the bits of the frame $\mathbf{u}$ with no particular regard for how far neighbouring bits are separated. These have the advantage of requiring only the storage of the pseudo-random number generator's seed, while the interleaver is not in use. This seed may then be employed to independently construct the interleaver within the transmitter and receiver when required. By contrast, the S-random interleaver [123] is specifically designed to ensure that any pair of bits that are located within a distance of S bits from each other before interleaving, are positioned at a distance of at least S bits from each other after interleaving. This approach is capable of more efficiently mitigating the dependencies within the resultant *a priori* LLRs than the less-sophisticated pseudo-random interleavers [123] and hence S-random interleavers are capable of reducing the performance penalty that is associated with employing short interleavers. However, the interleaver patterns must be stored while the interleaver is not in use, requiring more memory than a general pseudo-random design. Other interleaver designs include various code-matched interleavers [124, 125], which are designed for use with specific concatenated codes, attempting to minimise the correlation of the resultant *a priori* LLRs.

Note that the mutual information of the exchanged soft information will increase, as the iterative decoding trajectory advances. In the serial concatenated scheme of Figure 1.10, the *a priori* LLR frame $L_a^o(\mathbf{u})$ is provided for the MAP VLEC sequence estimator. Clearly the BER of the resultant reconstructed bit frame $\tilde{\mathbf{u}}$ and hence the SER of the reconstructed symbol frame $\tilde{\mathbf{s}}$ depends on the reliability of this LLR frame. Hence, the SER will reduce as iterative decoding proceeds and the iterative decoding trajectory advances.

Observe in the EXIT charts of Figure 1.14 that the step-wise iterative decoding trajectories continue advancing until the EXIT functions intersect. In the EXIT charts shown in Figures 1.14a – 1.14c, the EXIT functions do not intersect before reaching the $(1, 1)$ point, creating an open EXIT chart tunnel [126]. In these cases, the iterative decoding trajectory asymptotically approaches the $(1, 1)$ point of the EXIT chart, where absolutely certain soft information can be obtained. Hence, in these cases, iterative decoding convergence to an infinitesimally low SER may be obtained. However, owing to the reduced channel SNR of -0.3 dB, the URC decoder's EXIT function of Figures 1.14e and 1.14f is low enough to intersect with the inverted VLEC EXIT function at the $(0.27, 0.40)$ point. In this case, the iterative decoding trajectory cannot approach the $(1, 1)$ point of the EXIT chart and hence a relatively high SER that is commensurate with the achieved mutual information will result. Note that while an open EXIT chart tunnel is created in Figure 1.14d, the iterative decoding trajectory does not approach the $(1, 1)$ point of the EXIT chart owing to the relatively short $I \approx 10\,000$-bit interleaver employed. In this case, a relatively high SER that is commensurate with the achieved mutual information will result, as in the case where the EXIT chart tunnel is closed. Figure 1.15 provides a plot showing how the BPSK-modulated Rayleigh fading channel SNR affects the SER that may be obtained following the iterative decoding convergence of a URC decoder and the VLEC code **VLEC** of Table 1.1, as depicted in Figure 1.10.

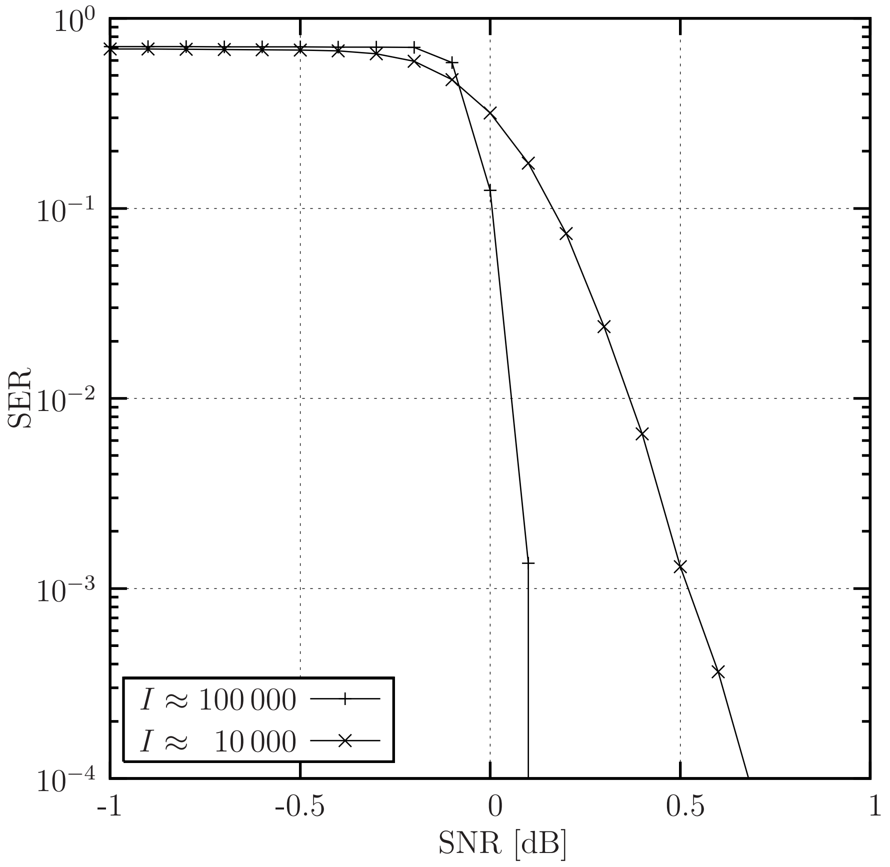


Figure 1.15: Plot showing how the BPSK-modulated Rayleigh fading channel SNR affects the SER that may be obtained following the iterative decoding convergence of a URC and the VLEC code **VLEC** of Table 1.1, as depicted in Figure 1.10. Results are provided for interleaver lengths of both $I \approx 100\,000$ bits and $I \approx 10\,000$ bits.

Observe in Figure 1.15 that when a relatively long interleaver length of $I \approx 100\,000$ bits is employed, the SER rapidly reduces as the channel SNR increases towards +0.2 dB. This so-called 'waterfall' or 'turbo-cliff' region may be explained by the opening of the EXIT chart tunnel at this threshold channel SNR and the resultant convergence of the iterative decoding trajectory towards the $(1, 1)$ point of the corresponding EXIT chart of Figure 1.14c. By contrast, a low SER below 10^{-4} is only achieved for channel SNRs in excess of +0.7 dB, when the shorter $I \approx 10\,000$-bit interleaver is employed, as shown in Figure 1.15. This may be explained by the effective narrowing of the EXIT chart tunnel that is caused by the residual dependencies amongst the *a priori* LLRs, as described above. As a result, the open EXIT chart tunnel is insufficiently wide for the iterative decoding trajectory to convergence to the $(1, 1)$ point of the EXIT chart until the channel SNR is increased to +0.7 dB, as shown in Figure 1.14b.

Note that if either the inner or the outer EXIT function does not reach the $(1, 1)$ point of the EXIT chart, as the EXIT function corresponding to the **Huff** codebook of Figure 1.12 did, then it will always intersect the other EXIT function before reaching the $(1, 1)$ point, regardless of the channel SNR. Hence, the iterative decoding trajectory will never convergence to the $(1, 1)$ point of the EXIT chart, where an infinitesimally low SER may be achieved. Instead, an error floor will result, providing a lower bound to the SER that may be achieved at a particular channel SNR. Therefore, iterative decoding convergence without an error floor is only supported if both the inner and the outer EXIT functions reach the $(1, 1)$ point of the EXIT chart.

An inner CC will support iterative decoding convergence to an infinitesimally low probability of error if it is recursive [127, 128], employing feedback in its shift register representation [8]. Similarly, a necessary and sufficient condition for an outer VLEC **VLEC** to support iterative decoding convergence to an infinitesimally low probability of error is that it has a free distance – as defined in Section 1.2.6.4 – of $d_{\text{free}}(\mathbf{VLEC}) \geq 2$ [129]. By extension, a free distance lower bound of $\bar{d}_{\text{free}}(\mathbf{VLEC}) \geq 2$ is a sufficient condition for an outer VLEC **VLEC** to support iterative decoding convergence to an infinitesimally low probability of error, as described in Section 1.2.6.4. Note that unlike the Huffman codebook **Huff** of Table 1.1, the codebooks **RVLC** and **VLEC** have free distance lower bounds of at least two, as shown in Table 1.3. For this reason, their EXIT functions reach the $(1, 1)$ point in the top-right hand corner of Figure 1.12, unlike the EXIT function corresponding to the Huffman codebook **Huff**. This property has motivated the design of RVLC codebooks **RVLC** that have a minimum block distance – as defined in Section 1.2.6.4 – of $d_{b_{\min}}(\mathbf{RVLC}) = 2$ [66], giving a free distance lower bound of $\bar{d}_{\text{free}}(\mathbf{RVLC}) \geq 2$, as described in Section 1.2.6.4. The related distance property has also motivated the design of Even Weight Variable Length Codes (EWVLCs) [130], which also guarantee a free distance lower bound of at least two.

A number of EXIT chart variations have been proposed in the literature. These include three-dimensional EXIT charts [116] developed for characterising the iterative decoding convergence properties of three-stage serial concatenations. Additionally, non-binary EXIT charts [131] may be employed to characterise concatenated schemes, where the iteratively exchanged soft information pertains to symbol values rather than bit values.

1.3.4.4 Area properties of EXIT charts

Let us now discuss the relevance of the EXIT chart area A_{inner} beneath the inner EXIT function and A_{outer} beneath the inverted outer EXIT function. Various proofs relating to these areas were provided in [126, 132] for the case when the *a priori* LLRs in the frames $L_a^o(\mathbf{u})$ and $L_a^i(\mathbf{u}')$ are provided for the respective APP SISO decoder over a Binary Erasure Channel (BEC), having either zero magnitudes or infinite magnitudes (and the correct sign). However, it has been shown that the shapes of the EXIT functions does not significantly depend on the particular channel considered [133] and hence the results discussed in this section hold approximately for more general channels.

In [126,132], the area A_{outer} beneath the inverted EXIT function of an optimal outer APP SISO decoder having a coding rate of R_{outer} was shown to be given by

$$A_{\text{outer}} = R_{\text{outer}}. \tag{1.21}$$

In the case where this outer code is serially concatenated with a R_{inner}-rate inner code that is employed for protecting M_{mod}-ary modulated transmissions, the effective throughput η in bits of source information per channel symbol is given by

$$\eta = R_{\text{outer}} \cdot R_{\text{inner}} \cdot \log_2(M_{\text{mod}}) = A_{\text{outer}} \cdot R_{\text{inner}} \cdot \log_2(M_{\text{mod}}). \tag{1.22}$$

As described in Section 1.3.4.3, maintaining an open EXIT chart tunnel is a necessary condition for achieving iterative decoding convergence to an infinitesimally low probability of error. Since an open EXIT chart tunnel can only be created if the EXIT chart area beneath the inverted outer EXIT function A_{outer} is less than that beneath the inner code's EXIT function A_{inner}, we have $A_{\text{outer}} < A_{\text{inner}}$ and hence maintaining

$$\eta < A_{\text{inner}} \cdot R_{\text{inner}} \cdot \log_2(M_{\text{mod}}) \tag{1.23}$$

constitutes a necessary condition for iterative decoding convergence to an infinitesimally low probability of error to be supported.

In the case where we have $R_{\text{inner}} = 1$ and an optimal inner APP SISO decoder is employed, [126, 132] showed that

$$A_{\text{inner}} \cdot R_{\text{inner}} \cdot \log_2(M_{\text{mod}}) = C, \tag{1.24}$$

where C is the Discrete-input Continuous-output Modulated Channel's (DCMC) capacity [134] expressed in bits of source information per channel symbol. Hence, in the case where $R_{\text{inner}} = 1$ and an optimal inner APP SISO decoder is employed,

$$\eta < C \tag{1.25}$$

constitutes a necessary condition for iterative decoding convergence to an infinitesimally low probability of error to be supported. Note that this is as Shannon stated in his seminal publication of 1948 [1].

However, in the case where we have $R_{\text{inner}} < 1$ or a sub-optimal inner APP SISO decoder is employed, [126, 132] showed that

$$A_{\text{inner}} \cdot R_{\text{inner}} \cdot \log_2(M_{\text{mod}}) = \bar{C} \leq C, \tag{1.26}$$

where $\bar{C}$ is the *attainable* DCMC capacity. More explicitly, in this case, some capacity loss occurs, since the necessary condition for iterative decoding convergence to an infinitesimally low probability of error to be supported becomes

$$\eta < \bar{C}. \tag{1.27}$$

Note that the property of having an area beneath the inverted EXIT function of a VLEC codebook **VLEC** that is equal to its coding rate $R(\mathbf{VLEC})$ complements the property that the EXIT function will reach the $(1, 1)$ point of the EXIT chart, provided that the VLEC free distance is $d_{\text{free}}(\mathbf{VLEC}) \geq 2$ [129]. Furthermore, we will show in Section 4.2 that the RV-FDM of Chapter 4 dictates how many points of inflection appear within the inverted VLEC EXIT function. By manipulating these properties, the GA of Chapter 4 will be shown to facilitate the design of specific VLEC codebooks having arbitrary EXIT function shapes. This is in contrast to the methods proposed for designing VLEC codebooks in [67, 69, 81], which maximise the coding rate of VLEC codebooks having particular specified distance properties and hence cannot design arbitrary EXIT function shapes.

Note that the EXIT chart area within an open EXIT chart tunnel $A_{\text{inner}} - A_{\text{outer}}$

is proportional to the discrepancy between the effective throughput and the (attainable) DCMC capacity. Hence, we may conclude that near-capacity transmissions are facilitated, when a narrow, marginally open EXIT chart tunnel can be created for facilitating convergence to an infinitesimally low probability of error. This motivates the employment of irregular coding for EXIT chart matching, as will be discussed in the following section.

1.4 Irregular coding

Irregular coding has been proposed for the reliable transmission of information at channel SNRs that are close to the channel's capacity bound [134] without imposing an excessive decoding complexity and latency. The concept was originally introduced [16] in the context of LDPC codes in [13, 14]. These may be represented using bipartite graphs [15], comprising a number of check nodes and a number of variable nodes. In the bipartite graph [15] of an irregular LDPC code, the check nodes are connected to various numbers of variable nodes and vice versa [135]. This is in contrast to regular LDPC codes, where all check nodes are connected to the same number of variable nodes and vice versa. Hence, a higher degree of freedom is facilitated during the design of irregular LDPC codes and this supports operation at channel SNRs that are closer to the channel's capacity bound [135]. Indeed, irregular LDPC operation within 0.13 dB of the channel's SNR capacity bound was demonstrated in [135]. Recently, a number of methods have been proposed for the design of irregular LDPC codes [136–143].

Following the intoduction of irregular LDPC codes, irregular coding was also applied in the context of turbo codes in [17]. As described in Section 1.3.3.1, these employ a parallel concatenation of two iteratively decoded CCs, which consider the original order of the source bits and an interleaved version, respectively. Whilst each source bit is encoded just once by each of the two parallel concatenated CCs in a regular turbo code, some source bits are encoded more than once by the CCs in the irregular scheme of [17]. As a result, it was possible to achieve a coding gain of 0.23 dB and operation within 0.25 dB of the channel's E_b/N_0 capacity bound by employing Monte Carlo simulations in order to find the required number of times that each source bit is encoded.

The serial concatenation [99] and iterative decoding [100] of an irregular outer code with a regular inner code was proposed by Tüchler and Hagenauer in [18]. Here, the irregular outer code is comprised of N number of component codes, having a variety of inverted EXIT function shapes $\{I_a^{o,n}(I_e^o)\}_{n=1}^N$. These component codes are invoked for generating specific fractions $\{\alpha^n\}_{n=1}^N$ of the encoded bit sequence, which may be specifically chosen in order to shape the irregular EXIT function $I_a^o(I_e^o)$ according to

$$I_a^o(I_e^o) = \sum_{n=1}^{N} \alpha^n I_a^{o,n}(I_e^o), \tag{1.28}$$

where we have to obey

$$\sum_{n=1}^{N} \alpha^n = 1. \tag{1.29}$$

Irregular Convolutional Codes (IrCC) were proposed in [144] and further characterised in [115, 145–149]. These typically employ a single mother CC to derive a suite of component CCs $\{\mathbf{CC}^n\}_{n=1}^N$, which have a variety of coding rates $\{R(\mathbf{CC}^n)\}_{n=1}^N$ that are obtained by using various generator polynomials in addition to the mother CC and/or by using puncturing [144]. In this case, the overall IrCC coding rate R_{IrCC} may be obtained according to

$$R_{\text{IrCC}} = \sum_{n=1}^{N} \alpha^n \cdot R(\mathbf{CC}^n). \tag{1.30}$$

An EXIT chart matching algorithm was proposed in [18], which may be employed to select the fractions $\{\alpha^n\}_{n=1}^N$ in order to shape the inverted EXIT function of the irregular outer code so that it matches the EXIT function of the serially concatenated regular inner code. This algorithm employs the method of steepest descent in order to seek the specific set of fractions $\{\alpha^n\}_{n=1}^N$ that minimises the squared error between the inverted EXIT function of the irregular outer code and the EXIT function of the regular inner code, without allowing them to intersect. Note however that the selection of the specific fractions $\{\alpha^n\}_{n=1}^N$ is subject to the constraints of (1.29) and (1.30). In this way, a narrow but nonetheless open EXIT chart tunnel may be created at near-capacity channel SNRs, supporting iterative decoding convergence to an infinitesimally low probability of error, as described in Section 1.3.4.4.

This is illustrated in the EXIT chart of Figure 1.16, in which an inverted IrCC EXIT function has been provided to match the EXIT function of a URC decoder that is employed to protect transmissions over a BPSK-modulated Rayleigh fading channel, having an SNR of +0.2 dB. According to (1.28), the inverted IrCC EXIT function is formed as the weighted average of the inverted EXIT functions that are provided in Figure 1.16 for the $N = 17$ component CCs $\{\mathbf{CC}^n\}_{n=1}^{17}$ of [144]. Figure 1.16 also provides the coding rates $\{R(\mathbf{CC}^n)\}_{n=1}^{17}$ and weights $\{\alpha^n\}_{n=1}^{17}$ of the component CCs $\{\mathbf{CC}^n\}_{n=1}^{17}$, which may be combined using (1.30) to obtain the IrCC coding rate of $R_{\text{IrCC}} = 0.56$.

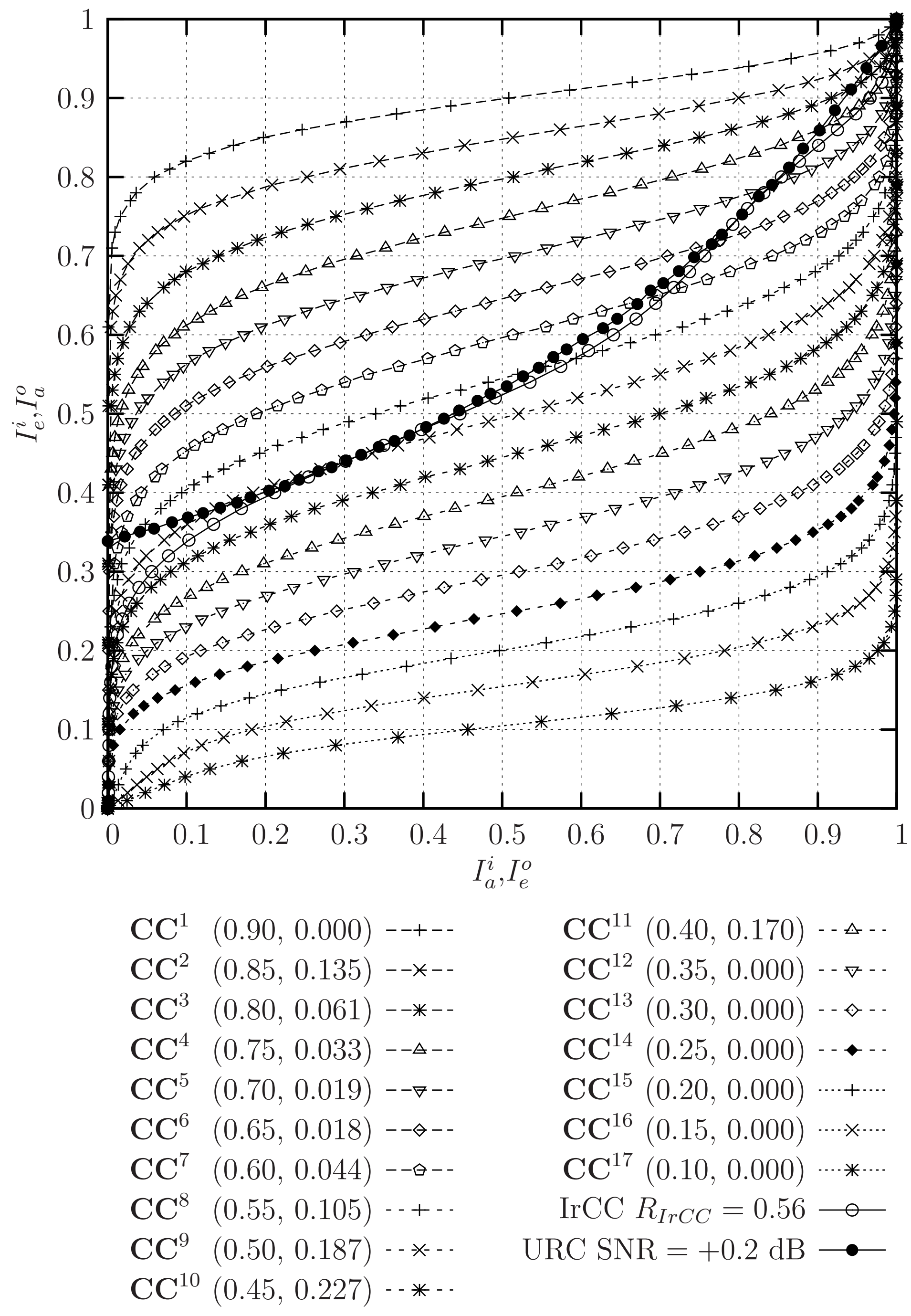


Figure 1.16: Inverted CC EXIT functions. The inverted EXIT function is provided for the corresponding IrCC arrangement, together with the URC EXIT function corresponding to a BPSK-modulated Rayleigh fading channel SNR of +0.2 dB. Inverted CC EXIT functions are labelled using the format $\mathbf{CC}^n$ ($R(\mathbf{CC}^n)$, α^n).

Note that, the ability of EXIT chart matching to create narrow but still open EXIT chart tunnels depends on the availability of a suite of component codes having a wide variety of EXIT function shapes. However, in general it is challenging to design component codes having arbitrary EXIT function shapes, motivating the irregular code design process depicted in Figure 1.17. This advocates the design of diverse candidate component codes, the characterisation of their EXIT functions and the selection of a specific suite having a wide variety of EXIT function shapes, potentially involving a significant amount of 'trial-and-error' based human interaction. Following this, EXIT chart matching may be achieved by selecting the component fractions $\{\alpha^n\}_{n=1}^{N}$, as described above.

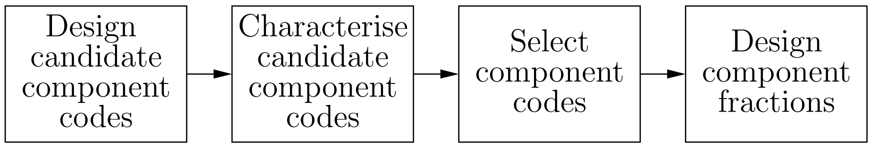

Design candidate component codes
Characterise candidate component codes
Select component codes
Design component fractions

Figure 1.17: Conventional irregular coding design process.

In Chapter 3 of this thesis we shall demonstrate the novel application of IrVLCs to EXIT chart matching, facilitating near-capacity joint source and channel coding. Furthermore, the novel GA of Chapter 4 will be shown to be attractive in the context of EXIT chart matching, since it facilitates the design of component VLEC codebooks having arbitrary EXIT function shapes, unlike the methods of [67, 69, 81]. Hence, the 'trial-and-error' based human interaction in the irregular coding design process of Figure 1.17 is eliminated, when the proposed GA of Chapter 4 is employed to design component VLEC codebooks. In Chapter 5, we propose a novel modification of the EXIT chart matching algorithm of [18] that additionally seeks a reduced APP SISO decoding complexity by considering the complexities associated with each of the component codes. Furthermore, another novel modification of Chapter 5 facilitates the EXIT chart matching of irregular codes that employ a suite of component codes having the same coding rate. This is achieved by removing the EXIT chart matching constraint of (1.30), facilitating the design of a novel Irregular Unity Rate Code (IrURC). Finally, Chapter 5 demonstrates the joint EXIT chart matching of two serially concatenated irregular codecs, namely an outer IrVLC and an inner IrURC. This is achieved by iteratively matching the outer decoder's inverted EXIT function to the inner decoder's EXIT function and vice versa. By employing an irregular inner code, in addition to an irregular outer code, we can afford a higher degree of design freedom than the proposals of [18], which employ a regular inner code. Hence, the described approach facilitates 'very-near-capacity' operation, which is comparable to that of IrLDPCs and irregular turbo codes.

1.5 Structure and novel contributions of the thesis

In this section we provide an overview of the remainder of this thesis and summarise its novel contributions with reference to their original publications.

In Chapter 2 we demonstrate the application of IrVLCs for the joint source and channel coding of video information, as described in Section 1.1.2. The proposed scheme employs the serial concatenation and iterative decoding of a video codec with a channel codec, in the manner detailed in Section 1.3.3.2. Our novel video codec represents the video information using Variable Dimension Vector Quantisation (VDVQ) tiles, which are similar to the VQ tiles described in Section 1.2.1, but having various dimensions. The VDVQ tiles employed are represented using the corresponding RVLC codewords selected from the VDVQ/RVLC codebook, as described in Section 1.2.5. However, the legitimate use of the VDVQ tiles and their corresponding RVLC codewords is limited by a number of code constraints, which ensure that the VDVQ tiles employed perfectly tessellate, among other desirable design objectives. As a result, different sub-sets of the RVLC codewords are available at different points during the encoding of the video information and the proposed approach adopts an IrVLC philosophy.

In the video codec of Chapter 2, the VDVQ/RVLC-induced code constraints are uniquely and unambiguously described by a novel VDVQ/RVLC trellis structure, which resembles the symbol-based VLEC trellis [50, 74] described in Section 1.2.6.3. Hence, the employment of the VDVQ/RVLC trellis structure allows the consideration of all legitimate transmission frame permutations. This fact is exploited in the video encoder in order to perform novel MMSE VDVQ/RVLC encoding, using a variant of the Viterbi algorithm [72] described in Section 1.2.6.2.

Additionally, the employment of the VDVQ/RVLC trellis structure during video decoding guarantees the recovery of legitimate – although not necessarily error-free – video information. As a result, the video decoder never has to discard video information. This is unlike in conventional video decoders, where a single transmission error may render an entire transmission frame invalid. Furthermore, the novel modification of the BCJR algorithm [10] of Section 1.3.2.2 is employed during APP SISO VDVQ/RVLC decoding in order to facilitate the iterative exchange of soft information with the serially concatenated channel decoder and in order to perform the soft MMSE reconstruction of the video sequence. Finally, since the VDVQ/RVLC trellis structure describes the complete set of VDVQ/RVLC-induced code constraints, all of the associated redundancy is beneficially exploited with the aid of the modified BCJR algorithm.

Owing to its aforementioned benefits and its employment of a joint source and channel coding philosophy, the video transmission scheme of Chapter 2 is shown to outperform the corresponding benchmarkers employing a separate source and channel coding philosophy. Our findings were originally published in [150, 151].

In Chapter 3, we investigate the application of IrVLCs to UEP, as described in Section 1.1.3. Here, a number of component VLC codebooks having different error correction capabilities are employed to encode various fractions of the source symbol frame. In the case where the various fractions of the source symbol frame have different error sensitivities, this approach may be expected to yield a higher reconstruction quality than equal protection, as noted in [58–60], for example.

Chapter 3 also investigates the application of IrVLCs to near-capacity operation, as described in Section 1.1.1. Here, a number of component VLC codebooks having different inverted EXIT functions are employed to encode various fractions of the source symbol frame. We show that the inverted IrVLC EXIT function may be obtained as a weighted average of the inverted component VLC EXIT functions, as described in Section 1.4. Additionally, the EXIT chart matching algorithm [18] described in Section 1.4 is employed to shape the inverted IrVLC EXIT function to match the EXIT function of a serially concatenated inner channel code and to create a narrow but still open EXIT chart tunnel. In this way, iterative decoding convergence to an infinitesimally low probability of error is facilitated at near capacity SNRs, as described in Section 1.3.4.4.

Furthermore, in Chapter 3, the UEP and near-capacity operation of the described scheme is assessed using novel plots that characterise the computational complexity of iterative decoding. More specifically, the average number of ACS operations required to reconstruct each source symbol with a high quality is plotted against the channel SNR. These plots are employed to compare the novel IrVLC-based scheme with a suitably designed IrCC and regular VLC based benchmarkers, quantifying the advantages of the IrVLCs Furthermore, these plots demonstrate that the complexity associated with the bit-based VLEC trellis of Section 1.2.6.1 is significantly lower than that of the symbol-based trellis described in Section 1.2.6.3. Our findings were originally published in [152, 153] and we proposed attractive near-capacity IrVLC schemes in [154–159].

In Chapter 4 we introduce a novel RV-FDM as an alternative to the IV-FD lower bound of (1.8) for the characterisation of the error correction capability that is associated with VLEC codebooks. Unlike the IV-FD lower bound, the RV-FDM assumes values from the real-valued domain, hence allowing the comparison of the error correction capability of two VLEC codebooks having equal IV-FD lower bounds, as described in Section 1.2.6.4. Furthermore, we show that a VLEC codebook's RV-FDM affects the number of inflection points that appear in the corresponding inverted EXIT function. This complements the property [132] that the area below an inverted VLEC EXIT function equals the corresponding coding rate, as well as the property that a free distance of at least two yields an inverted VLEC EXIT function that reaches the top right hand corner of the EXIT chart, as described in Section 1.3.4.4.

These properties are exploited by a novel GA in order to design beneficial VLEC codebooks having arbitrary inverted EXIT function shapes. This is in contrast to the methods of [67,69,81], which are incapable of designing codebooks having specific EXIT function shapes without imposing a significant level of 'trial-and-error' based human interaction, as described in Section 1.4. This novel GA is shown to be attractive for the design of IrVLC component codebooks for EXIT chart matching, since Chapter 4 also demonstrates that our ability to create open EXIT chart tunnels at near-capacity channel SNRs depends on the availability of a suite of component codes having a wide variety of EXIT function shapes.

Finally, a suite of component VLEC codebooks designed by the novel GA is found to facilitate higher-accuracy EXIT chart matching than a benchmarker suite designed using the state-of-the-art method of [67]. Our novel RV-FDM and GA were originally published in [156, 157].

In Chapter 5, we propose a novel modification to the EXIT chart matching algorithm of [18] that additionally seeks a reduced APP SISO decoding complexity by considering the complexities associated with each of the component codes. Furthermore, another novel modification of Chapter 5 facilitates the EXIT chart matching of irregular codes that employ a suite of component codes having the same coding rate. This is achieved by removing the EXIT chart matching constraint of (1.30), facilitating the design of a novel IrURC.

Additionally, Chapter 5 demonstrates the joint EXIT chart matching of two serially concatenated irregular codecs, namely an outer IrVLC and an inner IrURC. This is achieved by iteratively matching the inverted outer EXIT function to the inner EXIT function and vice versa. By employing an irregular inner code, in addition to an irregular outer code, we can afford a higher degree of design freedom than the proposals of [18], which employ a regular inner code. Hence, the proposed approach is shown to facilitate even nearer-capacity operation, which is comparable to that of IrLDPC and irregular turbo codes, as described in Section 1.4. Our findings were originally published in [158,159] and we additionally demonstated the joint EXIT chart matching of serially concatenated irregular codecs in [160].

Finally, in Chapter 6, we compare the results and findings of the previous chapters and draw our conclusions.

In summary, the novel contributions of this thesis are:

- a novel VDVQ/RVLC-TCM scheme for the iterative joint source and channel decoding of video information;
- its VDVQ/RVLC trellis structure;
- the adaptation of the Viterbi algorithm for MMSE VDVQ/RVLC encoding;
- the adaptation of the BCJR algorithm for APP SISO VDVQ/RVLC decoding and MMSE video reconstruction;
- IrVLC schemes for near-capacity operation;
- complexity versus channel SNR plots which are parameterised by the reconstruction quality;
- the RV-FDM for characterising the error correction capability of VLECs having the same IV-FD;
- the characterisation of the relationship between a VLEC's RV-FDM and the shape of its inverted EXIT function;
- a GA for designing VLECs having specific EXIT functions;
- a suite of VLECs that are suitable for a wide range of IrVLC applications;
- the adaptation of the EXIT chart matching algorithm to facilitate the use of component codes having the same coding rate;
- the adaptation of the EXIT chart matching algorithm to additionally seek a reduced APP SISO decoding computational complexity;
- the joint EXIT chart matching algorithm for designing schemes employing a serial concatenation of two irregular codecs;
- an IrVLC-IrURC scheme for very near capacity joint source and channel coding.

Bibliography

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

[2] D. J. Costello and G. D. Forney, “Channel coding: The road to channel capacity,” *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, June 2007.

[3] I. Jacobs and E. Berlekamp, “A lower bound to the distribution of computation for sequential decoding,” *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 167–174, April 1967.

[4] J. Wozencraft, “Sequential decoding for reliable communication,” *IRE National Convention Record*, vol. 5, no. 2, pp. 11–25, August 1957.

[5] J. M. Wozencraft and B. Reiffen, *Sequential Decoding.* Cambridge, Mass.: MIT Press, 1961.

[6] J. Massey, *Threshold Decoding.* Cambridge, MA, USA: MIT Press, 1961.

[7] R. Fano, “A heuristic discussion of probabilistic decoding,” *IEEE Transactions on Information Theory*, vol. 9, no. 2, pp. 64–74, April 1963.

[8] P. Elias, “Coding for noisy channels,” *IRE International Convention Record*, vol. 3, no. 4, pp. 37–46, 1955.

[9] G. D. Forney, “Review of random tree codes,” NASA Ames Research Center, Moffett Field, CA, USA, Tech. Rep. NASA CR73176, December 1967.

[10] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate (Corresp.),” *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, March 1974.

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

[12] R. G. Gallager, “Low-density parity-check codes,” *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[13] ——, *Low Density Parity Check Codes.* Cambridge, Mass.: MIT Press, 1963.

[14] 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. 457–458, August 1996.

[15] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, September 1981.

[16] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the ACM Symposium on Theory of Computing*, El Paso, TX, USA, May 1997, pp. 150–159.

[17] B. J. Frey and D. J. C. MacKay, "Irregular turbo-like codes," in *Proceedings of the International Symposium on Turbo Codes*, Brest, France, September 2000, pp. 67–72.

[18] M. Tüchler and J. Hagenauer, "EXIT charts of irregular codes," in *Proceedings of the Conference on Information Sciences and Systems*, Princeton, NJ, USA, March 2002, pp. 748–753.

[19] A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding.* New York, NY, USA: McGraw-Hill, 1979.

[20] L. Hanzo, P. J. Cherriman, and J. Streit, *Video Compression and Communications: H.261, H.263, H.264, MPEG4 and Proprietary Codecs for HSDPA-Style Adaptive Turbo-Transceivers.* John Wiley and IEEE Press, September 2007.

[21] L. Hanzo, F. C. A. Somerville, and J. P. Woodard, *Voice and Audio Compression for Wireless Communications.* John Wiley and IEEE Press, 2007.

[22] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, pp. 90–93, January 1974.

[23] A. N. Netravali and J. D. Robbins, "Motion-compensated television coding. I," *Bell System Technical Journal*, vol. 58, pp. 631–670, March 1979.

[24] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, September 1952.

[25] *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s*, ISO/IEC Std. 11 172, June 1996.

[26] *Information technology - Generic coding of moving pictures and associated audio information*, ISO/IEC Std. 13 818.

[27] *Information technology - Coding of audio-visual objects*, ISO/IEC Std. 14 496.

[28] *Video codec for audiovisual services at* $p \times 64$ *kbit/s*, ITU-T Std. H.261, March 1993.

[29] *Video coding for low bit rate communication*, ITU-T Std. H.263, January 2005.

[30] *Advanced video coding for generic audiovisual services*, ITU-T Std. H.264, March 2005.

[31] J. L. Massey, "Joint source and channel coding," in *Communication Systems and Random Process Theory*, J. K. Skwirzynski, Ed. Amsterdam, The Netherlands: Sijthoff and Noordhoff, December 1978, pp. 279–293.

[32] R. E. van Dyck and D. J. Miller, "Transport of wireless video using separate, concatenated, and joint source-channel coding," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1734–1750, October 1999.

[33] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982.

[34] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, March 1960.

[35] A. Kurtenbach and P. Wintz, "Quantizing for noisy channels," *IEEE Transactions on Communications*, vol. 17, no. 2, pp. 291–302, April 1969.

[36] N. Farvardin and V. Vaishampayan, "Optimal quantizer design for noisy channels: An approach to combined source-channel coding," *IEEE Transactions on Information Theory*, vol. 33, no. 6, pp. 827–838, November 1987.

[37] H. Kumazawa, M. Kasahara, and T. Namekawa, "A construction of vector quantizers for noisy channels," *Electronics and Engineering in Japan*, vol. 67-B, no. 4, pp. 39–47, January 1984.

[38] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Transactions on Information Theory*, vol. 36, no. 4, pp. 799–809, July 1990.

[39] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, January 1980.

[40] K. Sayood and J. C. Borkenhagen, "Use of residual redundancy in the design of joint source/channel coders," *IEEE Transactions on Communications*, vol. 39, no. 6, pp. 838–846, June 1991.

[41] N. Phamdo and N. Farvardin, "Optimal detection of discrete Markov sources over discretememoryless channels - applications to combined source-channel coding," *IEEE Transactions on Information Theory*, vol. 40, no. 1, pp. 186–193, January 1994.

[42] T. Fingscheidt and P. Vary, "Softbit speech decoding: A new approach to error concealment," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 240–251, March 2001.

[43] J. Kliewer and R. Thobaben, "Iterative joint source-channel decoding of variable-length codes using residual source redundancy," *IEEE Transactions on Wireless Communications*, vol. 4, no. 3, pp. 919–929, May 2005.

[44] F. Lahouti and A. K. Khandani, "Soft reconstruction of speech in the presence of noise and packet loss," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 1, pp. 44–56, January 2007.

[45] R. Thobaben and J. Kliewer, "Low-complexity iterative joint source-channel decoding for variable-length encoded Markov sources," *IEEE Transactions on Communications*, vol. 53, no. 12, pp. 2054–2064, December 2005.

[46] 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, February 1998.

[47] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," *IEEE Transactions on Communications*, vol. 38, no. 1, pp. 82–93, January 1990.

[48] V. Buttigieg and P. G. Farrell, "Variable-length error-correcting codes," *IEE Proceedings on Communications*, vol. 147, no. 4, pp. 211–215, August 2000.

[49] V. B. Balakirsky, "Joint source-channel coding with variable length codes," in *Proceedings of the IEEE International Symposium on Information Theory*, Ulm, Germany, June 1997, p. 419.

[50] R. Bauer and J. Hagenauer, "Symbol by symbol MAP decoding of variable length codes," in *Proceedings of the ITG Conference on Source and Channel Coding*, Munich, Germany, January 2000, pp. 111–116.

[51] M. Park and D. J. Miller, "Decoding entropy-coded symbols over noisy channels by MAP sequence estimation for asynchronous HMMs," in *Proceedings of the Conference on Information Sciences and Systems*, Princeton, NJ, USA, March 1998, pp. 477–482.

[52] M. Bystrom, S. Kaiser, and A. Kopansky, "Soft source decoding with applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 10, pp. 1108–1120, October 2001.

[53] N. Görtz, "A generalized framework for iterative source-channel decoding," *Annals of Telecommunications*, pp. 435–446, July/August 2001.

[54] J. Hagenauer and N. Görtz, "The turbo principle in joint source-channel coding," in *Proceedings of the IEEE Information Theory Workshop*, Paris, France, March 2003, pp. 275–278.

[55] X. Jaspar, C. Guillemot, and L. Vandendorpe, "Joint source-channel turbo techniques for discrete-valued sources: From theory to practice," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1345–1361, June 2007.

[56] J. Modestino and D. Daut, "Combined source-channel coding of images," *IEEE Transactions on Communications*, vol. 27, no. 11, pp. 1644–1659, November 1979.

[57] M. Grangetto, B. Scanavino, G. Olmo, and S. Benedetto, "Iterative decoding of serially concatenated arithmetic and channel codes with JPEG 2000 applications," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1557–1567, June 2007.

[58] Q. Qu, Y. Pei, and J. W. Modestino, “An adaptive motion-based unequal error protection approach for real-time video transport over wireless IP networks,” *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 1033–1044, October 2006.

[59] Y. C. Chang, S. W. Lee, and R. Komiya, “A low-complexity unequal error protection of H.264/AVC video using adaptive hierarchical QAM,” *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1153–1158, November 2006.

[60] T. Gan, L. Gan, and K.-K. Ma, “Reducing video-quality fluctuations for streaming scalable video using unequal error protection, retransmission, and interleaving,” *IEEE Transactions on Image Processing*, vol. 15, no. 4, pp. 819–832, April 2006.

[61] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, CA, USA, March 1966, pp. 281–297.

[62] Y. Takishima, M. Wada, and H. Murakami, “Reversible variable length codes,” *IEEE Transactions on Communications*, vol. 43, no. 234, pp. 158–162, February 1995.

[63] J. Wen and J. D. Villasenor, “Reversible variable length codes for efficient and robust image and video coding,” in *Proceedings of the Data Compression Conference*, Snowbird, UT, USA, March 1998, pp. 471–480.

[64] C.-W. Tsai and J.-L. Wu, “On constructing the Huffman-code-based reversible variable-length codes,” *IEEE Transactions on Communications*, vol. 49, no. 9, pp. 1506–1509, September 2001.

[65] C.-W. Lin, Y.-J. Chuang, and J.-L. Wu, “Generic construction algorithms for symmetric and asymmetric RVLCs,” in *Proceedings of the International Conference on Communication Systems*, vol. 2, Singapore, November 2002, pp. 968–972.

[66] K. Lakovic and J. Villasenor, “On design of error-correcting reversible variable length codes,” *IEEE Communications Letters*, vol. 6, no. 8, pp. 337–339, August 2002.

[67] J. Wang, L.-L. Yang, and L. Hanzo, “Iterative construction of reversible variable-length codes and variable-length error-correcting codes,” *IEEE Communications Letters*, vol. 8, no. 11, pp. 671–673, November 2004.

[68] V. Buttigieg and P. G. Farrell, “On variable-length error-correcting codes,” in *Proceedings of the IEEE International Symposium on Information Theory*, Trondheim, Norway, June 1994, p. 507.

[69] V. Buttigieg, “Variable-Length Error-Correcting Codes,” Ph.D. dissertation, Department of Electronic Enginnering, University of Manchester, Manchester, U.K., 1995.

[70] V. B. Balakirsky, “Joint source-channel coding using variable-length codes,” *Problems of Information Transmission*, vol. 37, no. 1, pp. 10–23, January 2001.

[71] R. Bauer and J. Hagenauer, "On variable length codes for iterative source/channel decoding," in *Proceedings of the IEEE Data Compression Conference*, Snowbird, UT, USA, March 2001, pp. 273–282.

[72] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[73] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[74] R. Bauer and J. Hagenauer, "Iterative source/channel-decoding using reversible variable length codes," in *Proceedings of the IEEE Data Compression Conference*, Snowbird, UT, USA, March 2000, pp. 93–102.

[75] L. Perros-Meilhac and C. Lamy, "Huffman tree based metric derivation for a low-complexity sequential soft VLC decoding," in *Proceedings of the IEEE International Conference on Communications*, vol. 2, New York, NY, USA, April 2002, pp. 783–787.

[76] A. Guyader, E. Fabre, C. Guillemot, and M. Robert, "Joint source-channel turbo decoding of entropy-coded sources," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 9, pp. 1680–1696, September 2001.

[77] N. Demir and K. Sayood, "Joint source/channel coding for variable length codes," in *Proceedings of the IEEE Data Compression Conference*, Snowbird, UT, USA, March 1998, pp. 139–148.

[78] M. Park and D. J. Miller, "Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," *IEEE Transactions on Communications*, vol. 48, no. 1, pp. 1–6, January 2000.

[79] C. Lamy and O. Pothier, "Reduced complexity maximum a posteriori decoding of variable-length codes," in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 2, San Antonio, TX, USA, November 2001, pp. 1410–1413.

[80] R. Thobaben and J. Kliewer, "Robust decoding of variable-length encoded Markov sources using a three-dimensional trellis," *IEEE Communications Letters*, vol. 7, no. 7, pp. 320–322, July 2003.

[81] C. Lamy and J. Paccaut, "Optimised constructions for variable-length error correcting codes," in *Proceedings of the IEEE Information Theory Workshop*, Paris, France, March 2003, pp. 183–186.

[82] J. Hagenauer, "The turbo principle - Tutorial introduction and state of the art," in *Proceedings of the International Symposium on Turbo Codes*, Brest, France, September 1997, pp. 1–11.

[83] L. Hanzo, J. P. Woodard, and P. Robertson, "Turbo decoding and detection for wireless applications," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1178–1200, June 2007.

[84] L. Hanzo, S. X. Ng, T. Keller, and W. Webb, *Quadrature Amplitude Modulation.* Chichester, UK: Wiley, 2004.

[85] 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.

[86] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proceedings of the IEEE Global Telecommunications Conference*, Dallas, TX, USA, November 1989, pp. 1680–1686.

[87] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation issues," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1201–1227, June 2007.

[88] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings of the IEEE International Conference on Communications*, vol. 2, Seattle, WA, USA, June 1995, pp. 1009–1013.

[89] V. Franz and J. B. Anderson, "Concatenated decoding with a reduced-search BCJR algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 186–195, February 1998.

[90] B. Vucetic, Y. Li, L. C. Perez, and F. Jiang, "Recent advances in turbo code design and theory," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1323–1344, June 2007.

[91] K. Gracie and M.-H. Hamon, "Turbo and turbo-like codes: Principles and applications in telecommunications," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1228–1254, June 2007.

[92] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, March 1996.

[93] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 409–428, March 1996.

[94] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Transactions on Communications*, vol. 46, no. 8, pp. 1003–1010, August 1998.

[95] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, June 1960.

[96] P. Robertson and T. Wörz, "A novel bandwidth efficient coding scheme employing turbo codes," in *Proceedings of the IEEE International Conference on Communications*, vol. 2, Dallas, TX, USA, June 1996, pp. 962–967.

[97] G. Ungerboeck, “Channel coding with multilevel/phase signals,” *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, January 1982.

[98] J. Kliewer and R. Thobaben, “Parallel concatenated joint source-channel coding,” *Electronics Letters*, vol. 39, pp. 1664–6, November 2003.

[99] S. Benedetto and G. Montorsi, “Serial concatenation of block and convolutional codes,” *Electronics Letters*, vol. 32, no. 10, pp. 887–888, May 1996.

[100] ——, “Iterative decoding of serially concatenated convolutional codes,” *Electronics Letters*, vol. 32, no. 13, pp. 1186–1188, June 1996.

[101] N. Görtz, “On the iterative approximation of optimal joint source-channel decoding,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 9, pp. 1662–1670, September 2001.

[102] M. Adrat, J.-M. Picard, and P. Vary, “Softbit-source decoding based on the turbo-principle,” in *Proceedings of the IEEE Vehicular Technology Conference*, vol. 4, Atlantic City, NJ, USA, October 2001, pp. 2252–2256.

[103] 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 the IEEE Data Compression Conference*, Snowbird, UT, USA, April 2002, pp. 83–91.

[104] 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,” *Electronics Letters*, vol. 39, no. 17, pp. 1259–1261, August 2003.

[105] D. Divsalar, H. Jin, and R. McEliece, “Coding theorems for ‘turbo-like’ codes,” in *Proceedings of the Allerton Conference on Communications*, no. 36, Allerton, IL, USA, September 1998, pp. 201–210.

[106] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, “Iterative correction of intersymbol interference: Turbo equalization,” *European Transactions on Telecommunications*, vol. 6, pp. 507–511, September 1995.

[107] X. Li and J. A. Ritcey, “Bit-interleaved coded modulation with iterative decoding,” in *Proceedings of the IEEE International Conference on Communications*, vol. 2, Vancouver, BC, Canada, June 1999, pp. 858–863.

[108] S. ten Brink, J. Speidel, and R.-H. Yan, “Iterative demapping and decoding for multilevel modulation,” in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 1, Sydney,NSW, Australia, November 1998, pp. 579–584.

[109] G. Caire, G. Taricco, and E. Biglieri, “Bit-interleaved coded modulation,” *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, May 1998.

[110] J. C. Serrato and T. O'Farrell, "Joint demapping and source decoding for multilevel modulation," in *Proceedings of the Wireless Communications and Networking Conference*, vol. 4, Las Vegas, NV, USA, April 2006, pp. 2140–2144.

[111] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenated trellis coded modulation with iterative decoding," in *Proceedings of the IEEE International Symposium on Information Theory*, Ulm, Germany, June/July 1997, p. 8.

[112] D. Divsalar, S. Dolinar, and F. Pollara, "Serial concatenated trellis coded modulation with rate-1 inner code," in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 2, San Francisco, CA, USA, November 2000, pp. 777–782.

[113] M. Moher, "An iterative multiuser decoder for near-capacity communications," *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 870–880, July 1998.

[114] W.-J. Choi, K.-W. Cheong, and J. M. Cioffi, "Iterative soft interference cancellation for multiple antenna systems," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, vol. 1, Chicago, IL, USA, September 2000, pp. 304–309.

[115] 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 *Proceedings of the International Symposium on Turbo Codes*, Munich, Germany, April 2006, electronic publication.

[116] M. Tüchler, "Convergence prediction for iterative decoding of threefold concatenated systems," in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 2, Taipei, Taiwan, November 2002, pp. 1358–1362.

[117] J. Hagenauer and P. Hoeher, "Concatenated Viterbi decoding," in *Proceedings of the Joint Swedish-Soviet International Workshop on Information Theory*, Gotland, Sweden, September 1989, pp. 29–33.

[118] D. Divsalar, S. Dolinar, and F. Pollara, "Low complexity turbo-like codes," in *Proceedings of the International Symposium on Turbo Codes*, Brest, France, September 2000, pp. 78–80.

[119] M. Tüchler, S. ten Brink, and J. Hagenauer, "Measures for tracing convergence of iterative decoding algorithms," in *Proceedings of the ITG Conference on Source and Channel Coding*, Berlin, Germany, January 2002, pp. 53–60.

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

[121] I. Land, P. Hoeher, and S. Gligorević, "Computation of symbol-wise mutual information in transmission systems with log APP decoders and application to EXIT charts," in *Proceedings of the International ITG Conference on Source and Channel Coding*, Erlangen, Germany, January 2004, pp. 195–202.

[122] J. Hokfelt, O. Edfors, and T. Maseng, “A turbo code interleaver design criterion based on the performance of iterative decoding,” *IEEE Communications Letters*, vol. 5, no. 2, pp. 52–54, February 2001.

[123] S. Dolinar and D. Divsalar, “Weight distributions for turbo codes using random and nonrandom permutations,” *Telecommunications and Data Acquisition Progress Report*, vol. 122, pp. 56–65, April 1995.

[124] W. Feng, J. Yuan, and B. S. Vucetic, “A code-matched interleaver design for turbo codes,” *IEEE Transactions on Communications*, vol. 50, no. 6, pp. 926–937, June 2002.

[125] J. Hokfelt, O. Edfors, and T. Maseng, “Turbo codes: correlated extrinsic information and its impact on iterative decoding performance,” in *Proceedings of the IEEE Vehicular Technology Conference*, vol. 3, Houston, TX, USA, July 1999, pp. 1871–1875.

[126] A. Ashikhmin, G. Kramer, and S. ten Brink, “Extrinsic information transfer functions: Model and erasure channel properties,” *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2657–2673, November 2004.

[127] J. Kliewer, A. Huebner, and D. J. Costello, “On the achievable extrinsic information of inner decoders in serial concatenation,” in *Proceedings of the IEEE International Symposium on Information Theory*, Seattle, WA, USA, July 2006, pp. 2680–2684.

[128] R. Thobaben, “EXIT functions for randomly punctured systematic codes,” in *Proceedings of the IEEE Information Theory Workshop*, Lake Tahoe, CA, USA, September 2007, pp. 24–29.

[129] J. Kliewer, N. Görtz, and A. Mertins, “Iterative source-channel decoding with Markov random field source models,” *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3688–3701, October 2006.

[130] R. Thobaben and J. Kliewer, “Design considerations for iteratively-decoded source-channel coding schemes,” in *Proceedings of the Allerton Conference on Communications, Control, and Computing*, Monticello, IL, USA, September 2006.

[131] J. Kliewer, S. X. Ng, and L. Hanzo, “Efficient computation of EXIT functions for nonbinary iterative decoding,” *IEEE Transactions on Communications*, vol. 54, no. 12, pp. 2133–2136, December 2006.

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

[133] M. Adrat, J. Brauers, T. Clevorn, and P. Vary, “The EXIT-characteristic of softbit-source decoders,” *IEEE Communications Letters*, vol. 9, no. 6, pp. 540–542, June 2005.

[134] J. G. Proakis, *Digital Communications.* McGraw-Hill, 1983.

[135] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, February 2001.

[136] G. Yue, X. Wang, and M. Madihian, "Design of rate-compatible irregular repeat accumulate codes," *IEEE Transactions on Communications*, vol. 55, no. 6, pp. 1153–1163, June 2007.

[137] G. Yue, B. Lu, and X. Wang, "Analysis and design of finite-length LDPC codes," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 3, pp. 1321–1332, May 2007.

[138] J. Xu, L. Chen, I. Djurdjevic, S. Lin, and K. Abdel-Ghaffar, "Construction of regular and irregular LDPC codes: Geometry decomposition and masking," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 121–134, January 2007.

[139] S. Tong, S. Zhang, B. Bai, and X. Wang, "Fast encodable and decodable irregular repeat accumulate codes from circulant permutation matrices," *Electronics Letters*, vol. 43, pp. 48–49, January 2007.

[140] A. Mahmood and E. Jaffrot, "Greedy check allocation for irregular LDPC codes optimization in multicarrier systems," in *Proceedings of the Wireless Communications and Networking Conference*, Hong Kong, March 2007, pp. 687–691.

[141] J. Chen, R. M. Tanner, J. Zhang, and M. P. C. Fossorier, "Construction of irregular LDPC codes by quasi-cyclic extension," *IEEE Transactions on Information Theory*, vol. 53, no. 4, pp. 1479–1483, April 2007.

[142] H. Chen and Z. Cao, "A modified PEG algorithm for construction of LDPC codes with strictly concentrated check-node degree distributions," in *Proceedings of the Wireless Communications and Networking Conference*, Hong Kong, March 2007, pp. 564–568.

[143] A. Abbasfar, D. Divsalar, and K. Yao, "Accumulate-repeat-accumulate codes," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 692–702, April 2007.

[144] M. Tüchler, "Design of serially concatenated systems depending on the block length," *IEEE Transactions on Communications*, vol. 52, no. 2, pp. 209–218, February 2004.

[145] 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 IEEE Vehicular Technology Conference*, vol. 5, Melbourne, Australia, May 2006, pp. 2484–2488.

[146] 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," in *Proceedings of the IEEE International Conference on Communications*, Glasgow, UK, June 2007.

[147] J. Wang, N. S. Othman, J. Kliewer, L.-L. Yang, and L. Hanzo, "Turbo-detected unequal error protection irregular convolutional codes designed for the wideband advanced multirate speech codec," in *Proceedings of the IEEE Vehicular Technology Conference*, vol. 2, Dallas, TX, USA, September 2005, pp. 927–931.

[148] S. Tan, J. Wang, S. X. Ng, S. Chen, and L. Hanzo, "Three-stage turbo MBER multiuser beamforming receiver using irregular convolutional codes," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 3, pp. 1657–1663, May 2008.

[149] N. Wu, O. Alamri, S. X. Ng, and L. Hanzo, "Precoded sphere packing aided bit-interleaved differential space-time coded modulation using iterative decoding," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 2, pp. 1311–1316, March 2008.

[150] R. G. Maunder, J. Kliewer, S. X. Ng, J. Wang, L.-L. Yang, and L. Hanzo, "Joint iterative decoding of trellis-based VQ and TCM," *IEEE Transactions on Wireless Communications*, vol. 6, no. 4, pp. 1327–1336, April 2007. [Online]. Available: http://eprints.ecs.soton.ac.uk/14077/

[151] ——, "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 IEEE Vehicular Technology Conference*, Dallas, TX, USA, September 2005, pp. 922–926. [Online]. Available: http://eprints.ecs.soton.ac.uk/11669/

[152] R. G. Maunder, J. Wang, S. X. Ng, L.-L. Yang, and L. Hanzo, "On the performance and complexity of irregular variable length codes for near-capacity joint source and channel coding," *IEEE Transactions on Wireless Communications*, vol. 7, no. 4, pp. 1338–1347, April 2008. [Online]. Available: http://eprints.ecs.soton.ac.uk/14467/

[153] ——, "Iteratively decoded irregular variable length coding and trellis coded modulation," in *Proceedings of the IEEE Workshop on Signal Processing Systems*, Shanghai, China, October 2007, pp. 222–227. [Online]. Available: http://eprints.ecs.soton.ac.uk/13907/

[154] S. Ahmed, R. G. Maunder, L.-L. Yang, S. X. Ng, and L. Hanzo, "Joint source coding, unity rate precoding and FFH-MFSK modulation using iteratively decoded irregular variable length coding," in *Proceedings of the IEEE Vehicular Technology Conference*, Baltimore, MD, USA, September 2007, pp. 1042–1046. [Online]. Available: http://eprints.ecs.soton.ac.uk/14468/

[155] M. El-Hajjar, R. G. Maunder, O. Alamri, S. X. Ng, and L. Hanzo, "Iteratively decoded irregular variable length coding and sphere-packing modulation-aided differential space-time spreading," in *Proceedings of the IEEE Vehicular Technology Conference*, Baltimore, MD, USA, September 2007, pp. 1238–1242. [Online]. Available: http://eprints.ecs.soton.ac.uk/14469/

[156] R. G. Maunder and L. Hanzo, "Genetic algorithm aided design of component codes for irregular variable length coding," *IEEE Transactions on Communications*, vol. 57, no. 5, pp. 1290–1297, May 2009. [Online]. Available: http://eprints.ecs.soton.ac.uk/14470/

[157] ——, "Genetic algorithm aided design of near-capacity irregular variable length codes," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, Las Vegas, NV, USA, April 2008, pp. 1256–1260. [Online]. Available: http://eprints.ecs.soton.ac.uk/14589/

[158] ——, "Near-capacity irregular variable length coding and irregular unity rate coding," *IEEE Transactions on Wireless Communications*, vol. 8, no. 11, pp. 5500–5507, November 2009. [Online]. Available: http://eprints.ecs.soton.ac.uk/14471/

[159] ——, "Concatenated irregular variable length coding and irregular unity rate coding," in *Proceedings of the IEEE Vehicular Technology Conference*, Barcelona, Spain, April 2009. [Online]. Available: http://eprints.ecs.soton.ac.uk/14472/

[160] R. Y. Tee, R. G. Maunder, J. Wang, and L. Hanzo, "Near-capacity irregular bit-interleaved coded modulation," in *Proceedings of the IEEE Vehicular Technology Conference*, Marina Bay, Singapore, May 2008, pp. 549–553. [Online]. Available: http://eprints.ecs.soton.ac.uk/14745/

Author Index

Glossary

ACS	Add, Compare and Select
ALU	Arithmetic and Logic Unit
APP	*A Posteriori* Probability
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem
BCJR	Bahl-Cocke-Jelinek-Raviv
BEC	Binary Erasure Channel
BER	Bit Error Ratio
BICM	Bit-Interleaved Coded Modulation
BPSK	Binary Phase Shift Keying
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CC	Convolutional Coding
COVQ	Channel-Optimised Vector Quantisation
DCMC	Discrete-input Continuous-output Modulated Channel
DCT	Discrete Cosine Transform
EWVLC	Even Weight Variable Length Coding
EXIT	EXtrinsic Information Transfer
FD	Frame Difference
GA	Genetic Algorithm
HA	Heuristic Algorithm
HISO	Hard-In Hard-Out
HMM	Hidden Markov Model

IIR	Infinite Impulse Response
IQ	In-phase Quadrature-phase
IrCC	Irregular Convolutional Coding
IrLDPC	Irregular Low Density Parity Check
IrURC	Irregular Unity Rate Coding
IrVLC	Irregular Variable Length Coding
ISI	InterSymbol Interference
IV-FD	Integer-Valued Free Distance
LBG	Linde-Buzo-Gray
LFSR	Linear Feedback Shift Register
LLR	Logarithmic Likelihood-Ratio
MAP	Maximum *A posteriori* Probability
MB	Macro-Block
MC	Motion Compensation
MIMO	Multiple-In Multiple-Out
ML	Maximum Likelihood
MMSE	Minimum Mean-Squared-Error
MPEG	Motion Picture Experts Group
MSEW	Maximum Squared Euclidean Weight
MSP	Modified Set Partitioning
MUD	Multi-User Detection
PDF	Probability Distribution Function
PSK	Phase Shift Keying
PSNR	Peak Signal to Noise Ratio
QAM	Quadrature Amplitude Modulation
QCIF	Quarter Common Intermediate Format
RV-FDM	Real-Valued Free Distance Metric
RVLC	Reversible Variable Length Coding
SER	Symbol Error Ratio
SIHO	Soft-In Hard-Out
SISO	Soft-In Soft-Out
SNR	Signal to Noise Ratio
SOBIT	SOft BIT
SOVA	Soft Output Viterbi Algorithm
SP	Set Partitioning
SQNR	Signal to Quantisation Noise Ratio

TCM	Trellis Coded Modulation
TCQ	Trellis Coded Quantisation
UEP	Unequal Error Protection
URC	Unity Rate Coding
VB	Video Block
VDVQ	Variable Dimension Vector Quantisation
VLC	Variable Length Coding
VLEC	Variable Length Error Correction
VQ	Vector Quantisation

Index