University of Southampton
Faculty of Engineering
Department of Electronics and Computer Science
Southampton SO17 1BJ

Turbo Equalisation Algorithms for Full and Partial Response Modulation

by

Bee Leong Yeap
B.Eng

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

January 2000

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

University of Southampton

Abstract

Faculty of Engineering

Department of Electronics and Computer Science

Doctor of Philosophy

Turbo Equalisation Algorithms for Full and Partial Response Modulation

by Bee Leong Yeap

This thesis is based on the research of joint channel equalisation and channel decoding algorithms, known as turbo equalisation. Initially, the performance of independent channel equalisation and decoding was studied in the context of Gaussian Minimum Shift Keying (GMSK) modulation. A low-rate 1.9 Kbps GSM-like speech system was constructed with the aim of studying the potential of using convolutional-based turbo codes for low-delay systems. The results confirmed that turbo codes did not outperform conventional convolutional codes in low-delay speech systems, but were advantageous for data transmission systems that can tolerate longer transmission delays.

Subsequently, iterative joint channel equalisation and channel decoding techniques were investigated for the above convolutional-coded GMSK scheme, for convolutional-coding based turbo-coded systems and for block-based Bose-Chaudhuri-Hocquengham (BCH) turbo-coded GMSK schemes. Our results showed that the convolutional-coded GMSK system outperformed the convolutional-coding based turbo-coded scheme and the BCH-coding based turbo-coded system by a margin of 1.0 dB and 0.8 dB, respectively, for the transmission over a five-path Rayleigh fading channel at BER $= 10^{-4}$. This performance trend was confirmed by deriving the theoretical Maximum Likelihood bound upon invoking the union bound approach.

Turbo equalisation for full-response Binary Phase Shift Keying (BPSK) systems was also researched. Three classes of codes, namely convolutional codes, convolutional-coding based turbo codes and BCH-coding based turbo codes were employed. The objective was to compare the turbo equalisation performance of these codes for high code rates of $R = \frac{3}{4}$ and $R = \frac{5}{6}$. It was observed that the turbo-equalised convolutional-coding based turbo-coded system was the most robust scheme. Finally, turbo equalisation was explored in the context of multi-level Quadrature Amplitude Modulation schemes. A novel reduced complexity trellis-based equaliser, known as the In-Phase/Quadrature-phase Equaliser (I/Q EQ), was proposed based on equalising the in-phase and quadrature-phase components of the transmitted signal separately. The I/Q EQ was capable of achieving the same performance as the conventional turbo equaliser, while achieving a complexity reduction by a factor of 2.67 and 16 for 4-QAM and 16-QAM, respectively.

Acknowledgements

I would like to take this opportunity to thank Professor Lajos Hanzo for his guidance and patience. Lajos, through his optimism and enthusiasm always kept my morale high even during the most trying moments. I was also fortunate to have had colleagues who were not only experts in their respective fields but also people with a great sense of humour. Special thanks go out to Jozef Hàmorskỳ, Kuan Ee Lin, Liew Tong Hooi, Wong Choong Hin, Jason Woodard and Yee Mong Suan for the lively discussions and their ideas. Not forgetting also Thomas Keller and Peter Cherriman for lending their time and computing expertise and Denise Harvey for her help with the administrative paper work.

The financial support of the European Union under the auspices of the Pan-European FIRST project and that of Motorola ECID, Swindon UK is also gratefully acknowledged.

Finally, a very special thank you to my family members at home who have always been supportive of my endeavours and to Khar Yee for all her encouragement, understanding and for sharing the 'knowledge of smile'.

Contents

[1]This work was based on a collaboration [1]

List of Publications

1. **F. C. A. Brooks, B. L. Yeap, J. P. Woodard and L. Hanzo** "A Sixth-rate, 3.8 Kbps GSM-like Speech Transceiver", Proceedings of the 3rd ACTS Mobile Communication Summit, held in Rhodes, Greece on 8-11 June 1998, pp 544-548.

2. **B. L. Yeap, T. H. Liew, J. Hàmorskỳ and L. Hanzo** "Block-based Turbo-coded, Turbo-equalised Partial-response Modulation for Dispersive Mobile Channels", Proceedings of Microcoll 1999, held in Budapest, Hungary on 21-24 March 1999, pp 71-74.

3. **B. L. Yeap, T. H. Liew, J. Hàmorskỳ and L. Hanzo** "Comparative Study of Turbo Equalisers using Convolutional Codes and Block-Based Turbo-Codes for GMSK Modulation", Proceedings of the IEEE Vehicular Technology Conference 1999, held in Amsterdam, Netherlands, on 19-22 September 1999, pp 2974-2978.

4. **A. Knickenberg, B. L. Yeap, J. Hàmorskỳ, M. Breiling and L. Hanzo** "Non-iterative Joint Channel Equalisation and Channel Decoding", IEE Electronic Letters Vol.35 No.19, on 16 September 1999, pp 1628-1630.

5. **A. Knickenberg, B. L. Yeap, J. Hàmorskỳ, M. Breiling and L. Hanzo** "Joint Channel Equalisation and Channel Decoding", Proceedings of IEEE Global Telecommunications Conference 1999, held in Rio de Janeiro, Brazil, on 5-9 of December 1999, pp 442-446.

6. **T. H. Liew, B. L. Yeap, J. Woodard, and L. Hanzo** "Modified MAP Algorithm for Extended Turbo BCH Codes and Turbo Equalisers", to appear in Proceedings of the First International Conference on 3G Mobile Communication Technologies 2000, London, United Kingdom, 27-29 March 2000.

7. **M. S. Yee, B. L. Yeap and L. Hanzo** "Iterative Radial Basis Function Assisted Turbo Equalisation", to appear in Proceedings of the IEEE Vehicular Technology Conference 2000, Tokyo, Japan.

8. **C. H. Wong, B. L. Yeap and L. Hanzo** "Wideband Burst-by-burst Adaptive Modulation with Turbo Equalization and Iterative Channel Estimation", to appear in Proceedings of the IEEE Vehicular Technology Conference 2000, Tokyo, Japan, 15-18 May.

9. **B. L. Yeap, T. H. Liew, J. Hàmorskỳ and L. Hanzo** "Comparative Study of Turbo Equalisers using Convolutional Codes, Convolutional-based Turbo Codes and Block-based Turbo Codes", submitted to the IEEE Journal on Selected Areas in Communications 1999.

10. **P. J. Cherriman, B. L. Yeap and L. Hanzo** "Turbo-equalised Interactive Videotelephony over GSM", manuscript in preparation for the IEEE Transactions on Circuits and Systems for Video Technology 2000.

11. **B. L. Yeap, C. H. Wong and L. Hanzo** "Iterative Channel Estimation and Reduced Complexity In-phase/Quadrature-phase Turbo Equalisation", manuscript in preparation for the IEEE Global Telecommunications Conference 2000, San Francisco, United States.

12. **B. L. Yeap and L. Hanzo** "Iterative Equalisation Techniques for Full and Partial Response Modems", manuscript in preparation for the IEEE Proceedings 2000.

13. **B. L. Yeap, C. H. Wong and L. Hanzo** "Reduced Complexity In-phase/Quadrature-phase Turbo Equalisation", manuscript in preparation for the IEEE Journal on Selected Areas in Communications 2000.

14. **B. L. Yeap, C. H. Wong and L. Hanzo** "In-phase/Quadrature-phase Turbo Equalisation in Wideband Burst-by-burst Adaptive Modulation With Iterative Channel Estimation", manuscript in preparation for the IEEE Journal on Selected Areas in Communications 2000.

List of Symbols

General notation

- The circle ($\circ$) superscript is used to indicate complex conjugation. Therefore, a° represents the complex conjugate of the variable a.
- The notation $X(f)$ is the Fourier Transform of $x(t)$.
- The notation $\bar{x}$ represents the estimate of x.
- $x_w(t)$ denotes that the function $x(t)$ has been weighted with a windowing function $w(t)$.

Special symbols

A:	Amplitude of a signal
$A(\kappa)$:	The logarithm domain representation of $\Omega(\kappa)$.
$A^C(W,Z)$:	Input Redundancy Weight Enumerating Function.
$A_w^C(Z)$:	Conditional Weight Enumerating Function.
a_I:	Multiplexed bits into the in-phase arm.
a_Q:	Multiplexed bits into the quadrature arm.
B:	Parameter used in an inter-burst interleaving scheme that specifies the index to a sub-block.
B_c:	Coherence bandwidth of the channel.
$B(\kappa)$:	The logarithm domain representation of $\beta(\kappa)$.
B_b:	Bandwidth of the Gaussian low pass filter. item $[B_L]$ Number of bits at input block of the encoder.
B_n:	Normalised bandwidth.
$C(\bar{\alpha})$:	Represents the correlation between the received signal and the expected received signal over the entire transmission burst.
$C(n,k)$:	Coded frame where n is the coded frame index and k is the index to the encoded bit within the coded frame
C_p:	Notation for serial concatenated encoder
C_s:	Notation for parallel concatenated encoder
$c(t)$:	Modulated sounding sequence.
$c_{i,n}$:	The ith code bit at instant n.
c_o:	Notation for outer encoder
c_i:	Notation for inner encoder
D:	Euclidean distance.
D_m:	Distribution of the codeword Hamming weight

$D^2(s_i(t), s_k(t))$:	Squared Euclidean distance between signals $s_i(t)$ and $s_k(t)$.
$d^2(s_i(t), s_k(t))$:	Normalised squared Euclidean distance between signals $s_i(t)$ and $s_k(t)$.
E_b:	Bit energy
E_b/N_o:	Ratio of bit energy to noise power spectral density.
E_i:	Amount of energy in the estimated channel impulse response within the duration i to $i + N_s \cdot L_w$, where i is the ith sampling position.
E_s:	Symbol energy
f_d:	Doppler frequency.
$f_c(\delta)$:	Correction term used in the Jacobian logarithm relationship $J(x_1, x_2)$, where $\delta = \|x_1 - x_2\|$.
f_c:	Carrier frequency.
$f(\cdot)$:	Conditional probability density function.
G_n:	The nth generator polynomial for convolutional codes.
$g(t)$:	Frequency shaping impulse response.
H:	Dummy variable whose order represents the codeword Hamming weight of the outer encoder
h_f:	Modulation index.
$h(t)$:	Channel impulse response.
$h_I(t)$:	In-phase component of the channel impulse response.
$h_Q(t)$:	Quadrature-phase component of the channel impulse response.
$\bar{h}(t)$:	Estimated channel impulse response.
$h_w(t)$:	Estimated channel impulse response, which has been windowed.
$I(B, j)$:	Notation to represent the interleaved frame, where B is the index to the sub-block while j is bit position in the interleaved frame.
I_t:	Critical number of iterations. It denotes the number of iterations required before the improvement in performance becomes insignificant.
$J(x_1, x_2)$:	Jacobian logarithmic relationship.

K:	Constraint length of a code.
K_b:	The number of bits required to represent a multi-level QAM symbol.
L:	The length or duration of the modulator's impulse response.
L_c:	The duration of the estimated channel impulse response.
L_w:	Duration of the truncated channel impulse response.
$L(u_n\|r)$:	Notation used for Log Likelihood Ratio (LLR).
M:	Indicates the modulation mode or M-ary signalling.
$m(t)$:	Matched filter impulse response.
$\mathsf{max}(x_1, x_2)$:	Function that returns the larger value between x_1 and x_2.
$n(t)$:	AWGN added to the transmitted signal.
N_0:	Single-sided power spectral density of white noise.
N_d:	Number of decoders employed in the turbo equaliser receiver.
N_f:	Number of feed-forward taps in a Decision Feedback Equaliser.
N_s:	Oversampling ratio.
$P_b(e)$:	Bit error probability.
P_{ob}:	Fractional out-of-band power.
P_M:	Path memory of the Viterbi Equaliser (VE). This is the delay before making a decision on the transmitted symbol.
$q(t)$:	Phase shaping impulse response.
R:	Code rate.
$r(t)$:	Received signal.
r_j:	The jth sample of the received signal oversampled at the rate of N_s.
$r_I(t)$:	The in-phase component of a complex received signal.
$r_Q(t)$:	The quadrature-phase component of a complex received signal.
${r'}_I(t)$:	The modified received signal, which is only dependent on the in-phase component of the transmitted signal and the channel impulse response.

${r'}_Q(t)$:	The modified received signal, which is only dependent on the quadrature-phase component of the transmitted signal and the channel impulse response.
$\grave{r}(t)$:	The result of convolving the received signal $r(t)$ with the windowed ambiguity function $\rho_w(t)$.
$S(f)$:	Power spectral density.
S_n:	The trellis state at time instant n.
$s(t)$:	Transmitted modulated signal.
$s_{i/j}$:	The j th sample of the ith transmitted signal oversampled at the rate of N_s.
$s_I(t)$:	The in-phase component of a complex transmitted signal.
$s_Q(t)$:	The quadrature-phase component of a complex transmitted signal.
$\hat{s}_I(t)$:	The estimate of the in-phase component of a complex transmitted signal.
$\hat{s}_Q(t)$:	The estimate of the quadrature-phase component of a complex transmitted signal.
$\tilde{s}_I(t, \alpha_i)$:	In-phase received signal, which is a function of time and data bit α_i.
$\tilde{s}_Q(t, \alpha_i)$:	Quadrature component of the received signal, expressed as a function of time, t and the ith data bit, α_i.
T:	Duration of one information symbol.
$w(t)$:	Windowing function.
w:	Hamming weight — *i.e.* the number of binary 1's in the word — of the input code
W:	Dummy variable whose order represents the input word Hamming weight of the inner encoder
Z:	Dummy variable whose order represents the parity word Hamming weight
$Z_n(\bar{\alpha})$:	Incremental metric calculated based on maximising the correlation between the received signal $r(t)$ and the estimate of the received signal $r(t, \bar{\alpha}_i)$ over the interval nT to $(n+1)T$.

$\cup$:	Statistical notation for union.
ϕ_c:	Carrier phase.
$\phi(t, \alpha_i)$:	Phase of the modulated signal.
$\theta(t, \alpha_i)$:	Correlative State Vector.
$\Upsilon_f(\grave{\kappa}, \kappa)$:	Value of the forward recursion for a single path, before taking into account other possible paths.
$\Upsilon_b(\grave{\kappa}, \kappa)$:	Value of the backward recursion for a single path, before taking into account other possible paths.
$\Gamma(\grave{\kappa}, \kappa)$:	The logarithm domain representation of $\gamma(\grave{\kappa}, \kappa)$.
$(\grave{\kappa}, \kappa) \Rightarrow u_n = +1$:	Represents the set, which contains all the paths from state $\grave{\kappa}$ at time instant $n-1$ to state κ at time instant n resulting in the transmission of a logical bit $+1$.
$(\grave{\kappa}, \kappa) \Rightarrow u_n = -1$:	Represents the set, which contains all the paths from state $\grave{\kappa}$ at time instant $n-1$ to state κ at time instant n resulting in the transmission of a logical bit -1.
$\grave{\kappa}$:	The previous state in the trellis.
κ:	The current trellis state.
$\Omega(\kappa)$:	Forward recursion variable for the MAP algorithm.
$\beta(\kappa)$:	Backward recursion variable for the MAP algorithm.
$\gamma(\grave{\kappa}, \kappa)$:	The product of the *a priori* information $P(u_n)$ with $P(r\|s_i)$, which is the probability that $r(t)$ was received given that $s_i(t)$ was transmitted.
θ_n:	Current phase state of the trellis state.
$\delta(t)$:	Dirac delta function.
$\rho(t)$:	Ambiguity function used for channel estimation.
$\lfloor \cdot \rfloor$:	Denotes the nearest integer floor.
τ_d:	Channel delay spread.
ϕ_{st}:	Reference phase in Frequency Discrimination detection and can have values 0, $\frac{\pi}{2}$, π, $-\frac{\pi}{2}$ and $-\pi$, depending on the previous decision bit.

ϕ_{rx}:	Phase of the received signal.
ϕ_{diff}:	The difference between the reference phase ϕ_{st} and the phase of the received signal, ϕ_{rx}.
$\bar{\alpha}$:	Estimate of the transmitted bit.
ψ_i:	The ith basis function.
ε:	Squared Euclidean distance.
π_c^{-1}:	Channel deinterleaver.
π_c:	Channel interleaver.
π_t^{-1}:	Turbo deinterleaver.
π_t:	Turbo interleaver.

Chapter 1

Introduction

During the last two decades, mobile communications have experienced an enormous growth. Commencing with the first generation mobile systems, speech services were provided through analogue transmissions. This motivated the development of several standards such as the Advanced Mobile Phone Service (AMPS) in the United States, the Total Access Communication System (TACS) in the United Kingdom and the Nippon Telephone and Telegraph (NTT) system in Japan. In the late 1980s second generation systems using digital transmission techniques were introduced. These systems, such as the Global System for Mobile Communication known as GSM [2, 3], offered higher spectrum efficiency and more advanced data services as compared to the first generation systems. This was then followed by intense standardisation activities towards the Third Generation (3G) systems [4] during the recent years. In the International Telecommunication Union's (ITU) activities the third generation networks are known as IMT-2000, while they are termed as Universal Mobile Telecommunication System (UMTS) in Europe. In support of the standardisation activities, several research programmes have developed 3G air interface concepts throughout the world, such as the Advanced Communications Technology and Services (ACTS) initiatives in Europe and the Future Public Land Mobile Telephone System (FPLMTS) as well as the Radio Transmission Special Group in Japan. The 3G proposals submitted to the European Telecommunications Standards Institute (ETSI) can be grouped in the the following main categories [5]: Wideband Code Division Multiple Access (WCDMA), Time Division Multiple Access (TDMA), hybrid CDMA/TDMA, Orthogonal Frequency Division Multiplex (OFDM) and Opportunity Driven Multiple Access (ODMA).

The main objectives of these 3G proposals include [4]:

- Full coverage and mobility for a bit rate of 144 kbps, preferably 384 kbps.

- Limited coverage and mobility for a bit rate of 2 Mbps.
- High spectrum efficiency compared to existing systems.
- High flexibility in supporting new services.

However, there are problems associated with high data rate transmissions and with employing spectrally efficient systems. Systems transmitting at high bit rates, such as 2 Mbps, experience a high grade of channel-induced dispersion and hence suffer from Inter Symbol Interference (ISI). Hence, the equaliser technology employed must be capable of mitigating the effects of ISI. Furthermore, in order to achieve a high bandwidth efficiency, multi-level linear modulation techniques — such as 16-level Quadrature Amplitude Modulation (QAM) and 64-level QAM [6] can be invoked. Partial response modulation techniques — such as Continuous Phase Modulation (CPM) [7] schemes, which include Gaussian Minimum Shift Keying (GMSK) [8] — can also be employed. The above QAM-based systems are susceptible to noise and fading, since the signal constellation points become closer in the constellation space. CPM schemes, which spread the effects of a bit over time in order to decrease the bandwidth of the signals, additionally suffer from intentionally introduced Controlled ISI (CISI). Hence, the equaliser design must also be able to cope with the additional CISI. In conjunction with equalisation, channel decoding can also be employed in order to further improve the performance of the systems. Powerful error correction schemes — such as turbo codes [9] — have been shown to yield performances close to Shannonian performance limits. Instead of performing the equalisation and decoding separately, higher performance gains could be achieved by implementing the equalisation and decoding operations jointly and iteratively. This technique is also known as turbo equalisation [10] and has been shown to combat the effects of channel ISI successfully.

The outline of this chapter is as follows. Section 1.1 presents the motivation of the research, while Section 1.2 describes the layout of the thesis. This is followed by a brief overview of the mobile radio channel in Section 1.3. In Sections 1.4 and 1.5, the basic Continuous Phase Modulation (CPM) theory and the Digital Frequency Modulation (DFM) theory are presented. Subsequently, the state representation of Minimum Shift Keying (MSK) and Gaussian MSK (GMSK) is discussed in Section 1.6. Section 1.7 presents the spectral performance of MSK and GMSK, while Section 1.8 describes the principles of constructing trellis-based equaliser states. Finally, Section 1.9 summarises the main points presented in this chapter.

1.1 Motivation

The scope of this research encompasses a range of equalisation and decoding issues with the aim of enhancing the performance of the GSM system. Since its standardisation in the mid-eighties, the GSM system has been evolving throughout the past decade. This process has been hallmarked by the introduction of the High-Speed Circuit Switched Data (HSCSD) service [11] and the General Packet Radio Service (GPRS) [12]. Further important developments were the introduction of the 5.6 kbps half-rate [13] and the Enhanced Full-Rate (EFR) speech codecs [14]. Turbo-equalised [15] and turbo-coded [16] performance improvements were proposed by Bauch *et al* and Burkert *et al*, respectively. Motivated by these trends, in this treatise we also set out to develop a range of powerful performance enhancement techniques. Specifically, turbo equalisation algorithms — which have been shown to mitigate the effects of ISI [10] — are investigated in the context of partial response CPM schemes, namely GMSK and spectrally efficient multi-level QAM techniques. Furthermore, a novel reduced complexity trellis-based equaliser is developed, in order to mitigate the associated high complexity, which is experienced by trellis-based equalisers when the channel dispersion is severe, since the number of channel equaliser states required increases exponentially with the channel dispersion.

1.2 Organisation of Thesis

Below, we present the layout of the thesis:

- In Chapter 2, the potential of employing turbo codes for low-rate GSM-like speech systems is investigated. An overview of the Soft-In/Soft-Out (SISO) algorithms, namely the Maximum *A Posteriori* (MAP) algorithm [17] and Log-MAP algorithm [18], which are used in the investigated equaliser and turbo decoder, is also presented.

- Chapter 3 introduces the principles of an iterative, joint equalisation and decoding technique known as turbo equalisation [10] invoked in order to overcome the ISI and CISI introduced by the channel and modulator, respectively. Modifications of the SISO algorithms employed in the equaliser and decoder are proposed in order to generate information related to not only the source bits but also to the parity bits of the codewords. The performance of coded systems employing turbo equalisation is analysed. Three classes of encoders are utilised, namely convolutional codes, convolutional-coding based turbo codes and Bose-Chaudhuri-Hocquengham (BCH)-coding based turbo codes.

- Theoretical models are devised for the coded schemes in order to derive the Maximum Likelihood bound of the system in Chapter 4. These models are based on the Serial Concatenated Convolutional Code (SCCC) analysis presented in reference [19]. Essentially, this analysis can be employed, since the modulator could be represented accurately as a rate $R = 1$ convolutional encoder. Apart from convolutional-coded systems, turbo-coded schemes are also considered. Therefore the theoretical concept of Parallel Concatenated Convolutional Codes (PCCC) [20] is utilised in conjunction with the SCCC principles in order to determine the Maximum Likelihood (ML) bound of the turbo-coded systems, which are modelled as hybrid codes consisting of a parallel concatenated convolutional code, serially linked with another convolutional code. An abstract interleaver from reference [20] — termed as the uniform interleaver — is also utilised, in order to reduce the complexity associated with determining all the possible interleaver permutations.

- Chapter 5 presents a comparative study of coded BPSK systems, employing high rate channel encoders. The objective of this study is to investigate the performance of turbo equalisers in systems employing different classes of codes for high code rates of $R = \frac{3}{4}$ and $R = \frac{5}{6}$, since known turbo equalisation results have only been presented for turbo equalisers using convolutional codes and convolutional-based turbo codes for code rates of $R = \frac{1}{3}$ and $R = \frac{1}{2}$ [10, 21]. Specifically, convolutional codes, convolutional-coding based turbo codes, and Bose-Chaudhuri-Hocquengham (BCH)-coding based [22, 23] turbo codes are employed in this study.

- Finally, in Chapter 6, a novel reduced complexity trellis-based equaliser is proposed. In each turbo equalisation iteration the decoder generates information, which reflects the reliability of the source and parity bits of the codeword. With successive iteration, the reliability of this information improves. This decoder information is exploited, in order to decompose the received signal such that each quadrature component consists of the in-phase or quadrature-phase component signals. Therefore, the equaliser only has to consider the possible in-phase or quadrature-phase components, which is a smaller set of signals, than all of their possible combinations.

- Chapter 7 summarises the main findings and conclusions. Suggestions for future work are also presented.

Next, we outline the novel contributions of this thesis:

- The potential of using turbo codes in a low-rate GSM-like speech system was investigated [1] in comparison to the system employing the same speech codecs but protected by convolutional codes.

- The turbo equalisation performance is investigated for convolutional-coding based turbo-coded GMSK systems, BCH-coding based turbo-coded GMSK schemes [24] and convolutional-coded GMSK systems [25].

- The theoretical bound for coded systems employing turbo equalisation is derived. The principles of deriving the maximum likelihood bound for Parallel Concatenated Convolutional Code (PCCC) schemes is used in conjunction with Serial Concatenated Convolutional Code (SCCC) bound theory in order to derive the performance bound of the turbo-coded system.

- The performance of turbo equalisers in BPSK-modulated systems employing different classes of codes — convolutional codes, convolutional-coding based turbo codes, and BCH-coding based turbo codes — for high code rates of $R = \frac{3}{4}$ and $R = \frac{5}{6}$ is studied comparatively. Known turbo equalisation results have only been presented for turbo equalisers using convolutional codes and convolutional-coding based turbo codes for code rates of $R = \frac{1}{3}$ and $R = \frac{1}{2}$ [10, 21].

- A novel reduced complexity equaliser is developed by exploiting the information generated by the decoder in each turbo equalisation iteration. This information is employed, in order to decompose the received signal such that only the in-phase or the quadrature-phase component signals need to be considered, hence reducing the number of states in the trellis-based channel equaliser [26].

In order to explore the practical design constraints of turbo-coded and turbo-equalised interative systems, transceivers were invoked in the context of real-time systems, such as speech and video transceivers. In the context of adaptive modulation, the throughput of the system in terms of bits per symbol was increased by employing turbo equalisation, since the turbo equaliser was capable of mitigating the effects of channel-induced ISI and hence enabling higher-order, higher throughput modulation modes to be employed more frequently, despite experiencing poor channel conditions [27][1]. Furthermore, the benefits of iterative turbo equalisation were quantified in terms of video performance improvements in the context of a GSM-like videophone transceiver [28][1]. With the objective of improving the performance of turbo equalisation schemes and exploring their performance limit, a non-iterative joint channel equalisation and channel decoding algorithm was proposed [29][1], where the equaliser trellis and the decoder trellis were amalgamated into a structure that we referred to as a supertrellis, in order to obtain the ML performance. Research into more powerful turbo equalisation schemes was also conducted, employing the more powerful extended BCH codes instead of conventional BCH codes in conjunction with turbo equalisation [30][1]. Finally, with the aim of reducing the complexity of the turbo equaliser, Radial Basis Function (RBF)-based equalisers were employed for turbo equalisation [31][1].

Having given an overview of the thesis and the novel contributions, we present a model of the mobile radio channel. Subsequently, the basic theory of Continuous Phase Modulation (CPM) is discussed and the fundamental principles for constructing a trellis for equalisation are presented.

1.3 The Mobile Radio Channel

Mobile radio channels have been characterised in depth for example in references [3, 32, 33, 34], hence here we restrict ourselves to a rudimentary introduction. A dispersive mobile radio channel has as an impulse response, which exhibits both time-domain delay-spread and Doppler frequency-domain spreading [6, 3]. The delay-spread introduces time dispersion and, as a consequence of the time-frequency duality, frequency-selective fading as well. Again, due to duality, the frequency-domain Doppler spread results in time-selective fading [3]. Although, all four effects are present in mobile radio channels, their dominance is dependent on the nature of the transmitted signal, i.e. on the modulation technique and Baud rate employed. Hence, the channel can be further categorised based on the dominant effects perceived by the system. We will briefly elaborate on this below.

Frequency dispersion and time-selective fading occurs when the channel is time-variant. In time-selective fading the frequency-domain response of the channel varies, while the signal is being transmitted, hence linearly distorting the signal. Furthermore, due to the time-frequency duality, frequency dispersion occurs and results in the signal bandwidth being stretched. In other words, the signal bandwidth at the receiver is wider than the transmitted signal's bandwidth due to the frequency dispersion over the channel. However, if the signal has a high bit rate and a short transmission frame length, the frame may propagate through the channel before any significant changes in channel characteristics take place. This may improve the system's resistance against time-selective fading. However, due to the reduced bit duration the signal now becomes more susceptible to frequency-selective fading or time dispersion. More explicitly, at high bit rates, the bit period will be shorter than the channel's delay spread in a time dispersive or frequency-selective channel. Therefore, the effect of the previous transmitted bits will be imposed on the present data bit, hence introducing ISI.

[1]This work was conducted in collaboration with the co-authors

In time-dispersive channels, the transmitted signal is stretched in time, since the signal is convolved with the channel's impulse response, hence prolonging the duration of the signal. This occurs when the transmitted signal, which is reflected and refracted, reaches the receiver along a number of different paths. Each of these signal paths arrives at the receiver with different delays and possesses a different power. As a consequence, the impulse response of a wireless channel can be represented as a series of pulses, as illustrated on the left of Figure 1.1. Frequency-selective fading, portrayed on the right in

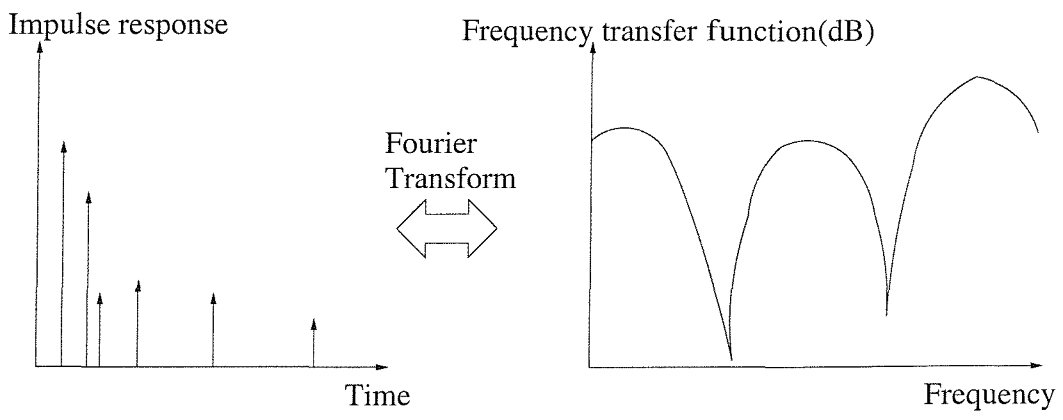


Figure 1.1: Stylised channel impulse response and frequency transfer function of a multipath channel.

Figure 1.1 arises, since the Fourier transform of this impulse response is typically a non-flat frequency-domain transfer function. When the signal bandwidth is narrow compared to the channel's so-called coherence bandwidth, B_c — which is inversely proportional to the delay spread τ_d related to it by $B_c = \frac{1}{2\pi\tau_d}$ [3] — the transmitted frequency components will experience similar attenuations or frequency-flat fading. Conversely, as the delay spread increases, B_c becomes narrower. If the transmission bandwidth exceeds B_c, the transmitted signal's frequency components will begin to encounter different attenuations, unlike in the flat-fading scenario described previously. Subsequently, the channel imposes a filtering effect, and linearly distorts the waveform, hence resulting in frequency-selective fading. For digital systems having large transmission bandwidths, the receiver will experience distinct time-domain echoes of the previous transmitted signals, resulting in ISI.

For example, in the Global System of Mobile Communications known as GSM [3], GMSK [35] modulation is employed at a transmission rate of 270.833 Kbauds, thereby giving a bit duration of 3.69μs. For this system, the bit duration is short with respect to the typical mobile radio channel delay spread, which can be as high as 18μs in the hilly terrain environment and therefore the signal does not experience time-selective fading and frequency dispersion. Instead, it experiences frequency-selective fading and time dispersion due to its large transmission bandwidth for a static Mobile Station (MS). However, when the MS starts to move, Doppler spreading occurs, since the channel characteristics begin to change more significantly with time. Therefore, the mobile radio channel's impulse response can be modelled as a sequence of impulses, where each impulse is attenuated by values satisfying the Rician or Rayleigh distribution statistics [36]. Under these circumstances, the above-mentioned GMSK signals will be exposed to time-dispersion and the receiver will observe contributions from the previous transmitted symbols. Furthermore, the GMSK signal's spectrum will be distorted, since the channel characteristics vary with time as a consequence of Doppler spreading. It is therefore necessary to perform channel equalisation in order to resolve the transmitted signals, which have been degraded by the channel-induced impairments, in addition to the deliberately introduced CISI. Another channel model employed in our simulations is the so-called narrowband fading channel. The corresponding channel impulse response is modelled as a single impulse, which is attenuated by fading values conforming to Rayleigh or Rician statistics.

Having discussed the channel models used, we proceed by describing the basic CPM principles and subsequently present two CPM schemes, namely Minimum Shift Keying (MSK) and Gaussian Minimum Shift Keying (GMSK).

1.4 Continuous Phase Modulation Theory

The classification tree of CPM schemes is shown in Figure 1.2. It is observed that there are two subclasses of CPM, namely Digital Frequency Modulation (DFM) and Digital Phase Modulation (DPM). Examples of DPM are L-Raised Cosine (L-RC) modulation, where L denotes the length of spreading in the modulator filter. Therefore, 1-RC is a full response scheme, while 2-RC and 3-RC are examples of partial response modulation. For an in-depth treatment on the subject of DPM the interested reader is referred to references [7, 3]. In this section, the basic DFM theory, specifically MSK and GMSK, is elaborated.

1.5 Digital Frequency Modulation Systems

In Digital Frequency Modulation (DFM) the transmitted radio signal can be described as:

$$s(t,\alpha) = \sqrt{\frac{2E_s}{T}} \cos\left(2\pi f_c t + \phi(t,\alpha) + \phi_c\right), \tag{1.1}$$

where E_s is the energy within the symbol period T. Note that in a binary system a symbol consists of only one bit. Therefore, the terms *symbol* and *bit* have the same meaning.

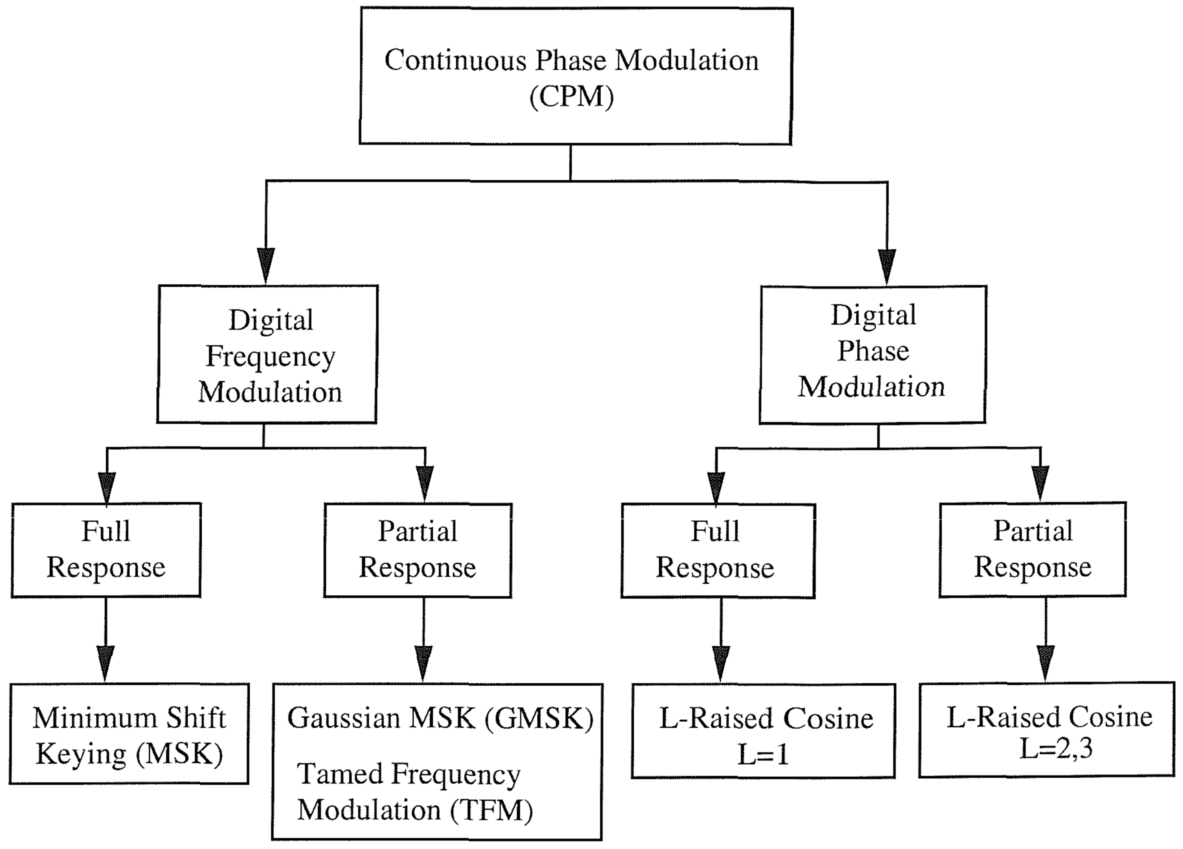


Figure 1.2: Classification tree of Continuous Phase Modulation (CPM) schemes.

The frequency of the carrier is f_c, while $\phi(t, \alpha)$ is the phase contribution dependent on the information bit α. Also, the term ϕ_c is used to denote the phase offset, which can be set to 0 without loss of generality. Here α represents an infinitely long and uncorrelated data sequence, which, for M-ary signalling can be defined as,

$$\alpha \in \left\{ \pm 1, \pm 3 \ldots \pm (M-1) \right\}.$$

Equation 1.1 is also often presented as:

$$s(t,\alpha) = \sqrt{\frac{2E_s}{T}} \cos\left[\phi(t,\alpha)\right] \cdot \cos\left[2\pi f_c t + \phi_c\right] - \sqrt{\frac{2E_s}{T}} \sin\left[\phi(t,\alpha)\right] \cdot \sin\left[2\pi f_c t + \phi_c\right], \quad (1.2)$$

leading to the so-called quadrature-representation, where the in-phase and quadrature-phase carriers are modulated independently by the terms $\cos[\phi(t,\alpha)]$ and $\sin[\phi(t,\alpha)]$. The modulated radio signal, $s(t,\alpha)$, can be produced through two different quadrature-component based implementations, as illustrated in Figure 1.3. The first method applies the input data, α, to a frequency shaping filter, $g(t)$, which constitutes an impulse response that spreads each data bit over a finite number of bit intervals. Subsequently, the output

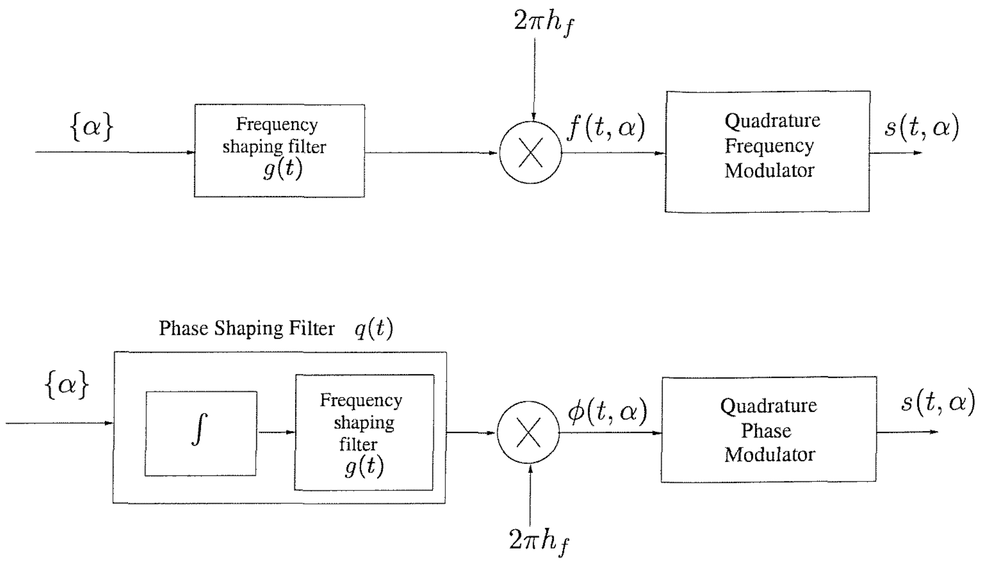

$2\pi h_f$
$\{\alpha\}$
Frequency shaping filter $g(t)$
$f(t,\alpha)$
Quadrature Frequency Modulator
$s(t,\alpha)$
Phase Shaping Filter $q(t)$
$\{\alpha\}$
$\int$
Frequency shaping filter $g(t)$
$\phi(t,\alpha)$
Quadrature Phase Modulator
$s(t,\alpha)$
$2\pi h_f$

Figure 1.3: An illustration of two possible Digital Frequency Modulation (DFM) implementations, where α, $f(t,\alpha)$, $\phi(t,\alpha)$ and $s(t,\alpha)$ are the sequence of uncorrelated bits, the frequency of the DFM signal, the phase of the DFM signal and the modulated DFM signal, respectively.

of the filter is multiplied by $2\pi h_f$, where h_f is the modulation index, in order to give the frequency $f(t,\alpha)$ of the DFM signal. The resultant DFM signal $f(t,\alpha)$ is then directed to a frequency modulator, which generates varying harmonics of frequency, depending on the transmitted data bits, thereby producing the DFM signal.

The alternative approach followed in the lower branch of Figure 1.3, is to integrate the frequency shaping filter's impulse response, $g(t)$, prior to the filtering of the input data stream. Subsequently, the response of the filter is multiplied with the scaled modulation-index term $2\pi h_f$, in order to yield the data dependent phase, $\phi(t,\alpha)$, of the transmitted data bits, which is applied to a phase modulator. Mathematically, this scheme can be described by representing the phase, $\phi(t,\alpha)$, as the integral of the partial response pulse shaped data signal, summed over the signalling interval index $-\infty \leq i \leq \infty$ and scaled by $2\pi h_f$, yielding:

$$\phi(t,\alpha) = 2\pi h_f \sum_{i=-\infty}^{+\infty} \int_{-\infty}^{t} \alpha_i \cdot g(\tau - iT)d\tau, \tag{1.3}$$

where — as mentioned previously — α, is an infinitely long sequence of uncorrelated bits, $g(t)$ is the frequency shaping impulse response and h_f is the modulation index. The modulation index, h_f, is a ratio of relative prime numbers and also a proportionality constant, which determines the magnitude of the phase change upon receiving a data bit. We will elaborate further on the choice of h_f in Section 1.6. The phase, $\phi(t,\alpha)$, of a DFM signal in the nth signalling interval can then be expressed in terms of the phase shaping function, $q(t)$ as:

$$\phi(t,\alpha) = 2\pi h_f \sum_{i=-\infty}^{n} \alpha_i \cdot q(t-iT) \qquad \text{for } nT \leqslant t \leqslant (n+1)T, \tag{1.4}$$

since

$$q(t) = \int_{-\infty}^{t} g(\tau)d\tau. \tag{1.5}$$

In practical implementations the integration of $g(t)$ in Equation 1.5 is performed within the limits of $t = 0$ to $t < LT$. When $t < 0$ and $t \geqslant LT$, we have:

$$q(t) = 0 \qquad \text{for } t < 0 \tag{1.6}$$

and

$$q(t) = 0.5 \qquad \text{for } t \geqslant LT, \tag{1.7}$$

respectively. The frequency shaping function $g(t)$ can be of any arbitrary causal shape having a width of L bit intervals:

$$g(t) = \begin{cases} 0 & \text{for } t < 0 \text{ and } t > LT \\ \text{any arbitrary function} & \text{for } 0 \leqslant t \leqslant LT, \end{cases} \tag{1.8}$$

where $L = 1$ for full response systems, while for partial response schemes we have $L > 1$. Therefore, the phase shaping impulse response, $q(t)$, is also spread over a finite number of bit intervals, L, and the effective output phase $\phi(t,\alpha)$ has contributions from partially overlapping pulses. The intentional introduction of Inter Symbol Interference (ISI) due to partial-response pulse shaping results in a tight spectrum, which reduces the spectral spillage into adjacent channels. However, in order to mitigate the effects of ISI, an equaliser is needed at the receiver, even if the channel is non-dispersive. For the purpose of illustration, in Figure 1.4 we portrayed a pair of stylised impulse responses, namely $q(t)$ and $g(t)$.

1.6 State Representation

The partial response DFM signal can be viewed as a signal exhibiting memory. This is the consequence of spreading the response of the filter, $q(t)$, over a finite number of bit intervals. For modulated signals exhibiting memory the optimal form of detection is Maximum Likelihood Sequence Estimation (MLSE) [37, 3]. MLSE-type detection can be employed by observing the development of the so-called phase tree over an arbitrary number of symbols, before making a decision on the oldest symbol within that time span. The number of

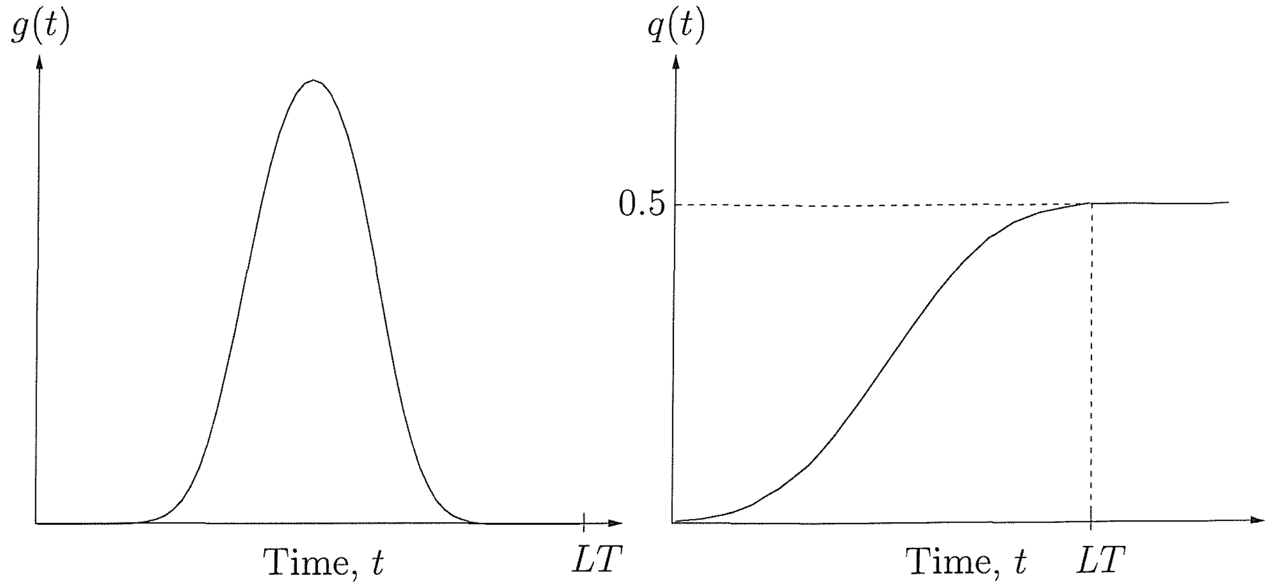


Figure 1.4: Stylised $g(t)$ and $q(t)$ impulse responses. The impulse response $q(t)$ is obtained by integrating $g(t)$, as shown in Equation 1.5.

branches in the phase tree increases exponentially with each data bit received. Hence, the number of correlations required, in order to identify the most likely received sequence will also increase exponentially. However, as it will be shown, upon employing a trellis structure or state representation of the DFM system, this exponential increase in complexity can be avoided. In the following discussion, the underlying principles for constructing a DFM trellis structure are established.

By using the relationship expressed in Equations 1.6 and 1.7 and by rearranging Equation 1.4, the phase of the DFM modulated radio signal at instant n can be written as [7]:

$$\begin{aligned}\phi(t,\alpha) &= 2\pi h_f \sum_{i=-\infty}^{n-L} \alpha_i q(t-iT) + 2\pi h_f \sum_{i=n-L+1}^{n} \alpha_i q(t-iT) \\ &= 2\pi h_f \sum_{i=-\infty}^{n-L} \alpha_i (0.5) + 2\pi h_f \sum_{i=n-L+1}^{n} \alpha_i q(t-iT) \\ &= 2\pi h_f \sum_{i=n-L+1}^{n} \alpha_i q(t-iT) + \pi h_f \sum_{i=-\infty}^{n-L} \alpha_i \\ &= \theta(t,\alpha) + \theta_n,\end{aligned} \tag{1.9}$$

where

$$\begin{aligned}\theta(t,\alpha) &= 2\pi h_f \sum_{i=n-L+1}^{n} \alpha_i q(t-iT) \\ &= \underbrace{2\pi h_f \sum_{i=n-L+1}^{n-1} \alpha_i q(t-iT)}_{\text{Correlative State Vector}} + \underbrace{2\pi h_f \alpha_n q(t-nT)}_{\text{Current bits's effect}},\end{aligned} \tag{1.10}$$

and θ_n

$$\theta_n = \pi h_f \sum_{i=\infty}^{n-L} \alpha_i, \tag{1.11}$$

is the *Phase State*, which is periodic with respect to 2π and represents the accumulated phase due to the previous bits that have passed through the phase shaping filter. Equation 1.10 describes the relationship between three parameters, namely $\theta(t,\alpha)$, the most recent information bit, α_n, and the so-called *Correlative State Vector*, which is the first term on the right.

Let us now establish the relationship between the number of phase states needed and the modulation index, h_f. Since the phase state, θ_n is $\pi h_f \sum_{i=-\infty}^{n-L} \alpha_i$ and recalling that h_f is a ratio of relative prime numbers and θ_n is periodic with respect to 2π, the four phase states for a modulation index $h_f = \frac{1}{2}$ are $0, \frac{\pi}{2}, \pi$ and $\frac{3\pi}{2}$. Similarly, for $h_f = \frac{1}{3}$, there are six phase states, namely $0, \frac{\pi}{3}, \frac{2\pi}{3}, \pi, \frac{4\pi}{3}$ and $\frac{5\pi}{3}$. When the modulation index, h_f, is $\frac{2}{3}$, the three phase states are $0, \frac{2\pi}{3}$ and $\frac{4\pi}{3}$. We observe that for modulation indices, which possess even numerators, the number of phase states is equal to the denominator, as in the example of $h_f = \frac{2}{3}$. By contrast, when the numerator of h_f is odd, as in $h_f = \frac{1}{2}$ and $h_f = \frac{1}{3}$, the number of phase states is double that of the denominator. Therefore, we can express the modulation index, h_f, as [7]:

$$h_f = \frac{2\cdot k}{p}, \tag{1.12}$$

where k and p are two integers with no common factor, such that the denominator, p, determines the number of phase states for the DFM system. For example, for $k = 1$ and $p = 4$ we arrive at $h_f = \frac{1}{2}$ and four phase states. A general expression for the p legitimate phase states , θ_n, is as follows:

$$\theta_n \in \left\{k \cdot \frac{2\pi}{p}\right\} \qquad \text{for } k = 0 \ldots (p-1). \tag{1.13}$$

For a full response DFM system with $L = 1$, *i.e.* when employing no partial-response spreading, Equation 1.9 becomes,

$$\phi(t,\alpha) = 2\pi h_f \alpha_n q(t-nT) + \pi h_f \sum_{i=-\infty}^{n-1} \alpha_i. \tag{1.14}$$

The first term on the right depends on the most recent information bit α_n and the second term is the phase state θ_n.

In a partial response system, there are correlative state vector contributions from overlapping pulses, as we have seen in Equation 1.10. The message of Equation 1.10 will be made more explicit below by introducing the so-called trellis structure. By studying Equations 1.9 and 1.10, it is observed that the overall phase is dependent on the correlative state vector, the phase state, θ_n, and the latest data bit, α_n. Therefore, we define the trellis state, S_n, as a combination of the correlative state vector and phase state, θ_n:

$$S_n = \{\theta_n, \text{Correlative state vector}\},$$

and we will illustrate this relationship in Figure 1.5. However, since the correlative state vector relies on the information bits $\alpha_{n-1}, \alpha_{n-2}, \ldots, \alpha_{n-L+1}$, as seen in Equation 1.10, the trellis state S_n in the partial response DFM system at time nT can be defined as,

$$S_n \triangleq \{\theta_n, \alpha_{n-1}, \alpha_{n-2}, \ldots, \alpha_{n-L+1}\}. \tag{1.15}$$

For a modulation index of $h_f = \frac{2k}{p}$ and M-ary signalling, where $v = \log_2 M$ number of bits are input simultaneously to the modulator, there are p number of phase states and M^{L-1} correlative state vectors for each phase state. Hence, at time nT the total number of states is $2^{v(L-1)}$. Upon receiving an information bit α_n within the interval nT and $(n+1)T$, the state S_n changes to S_{n+1}, depending on the value of α_n. By using the state representation established in Equation 1.15, S_{n+1} becomes,

$$\begin{array}{rcl} S_n & \xrightarrow{\alpha_n} & S_{n+1} \\ (\theta_n, \{\alpha_{n-1}, \alpha_{n-2}, \ldots, \alpha_{n-L+1}\}) & \xrightarrow{\alpha_n} & (\theta_{n+1}, \{\alpha_n, \alpha_{n-1}, \ldots, \alpha_{n-L+2}\}), \end{array} \tag{1.16}$$

where the phase state of Equation 1.13 at time $(n+1)T$ is given by:

$$\theta_{n+1} = [\theta_n + \pi h_f \alpha_{n-L+1}] \qquad \text{modulo } 2\pi, \tag{1.17}$$

since the phase state is accumulative.

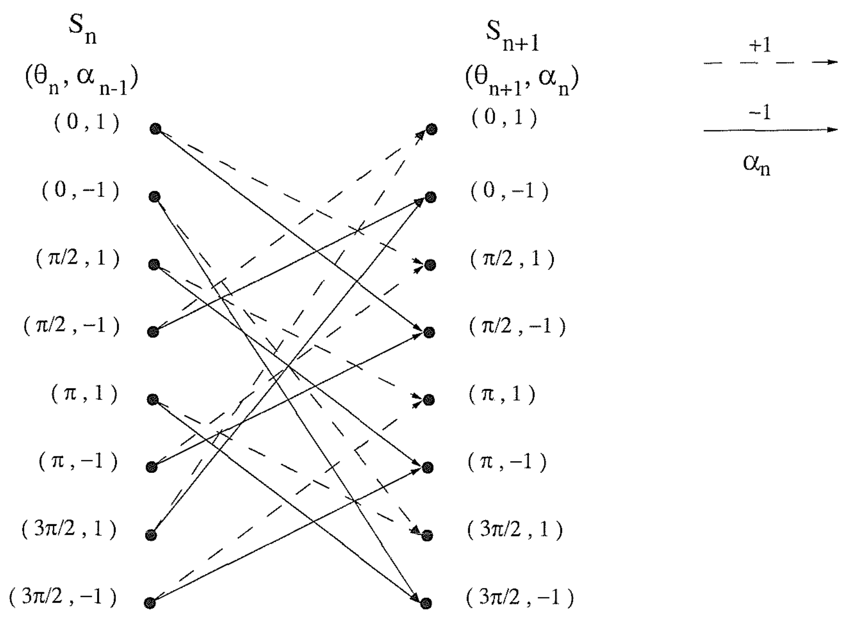


Figure 1.5: The states in a partial response DFM trellis system. The length of spreading is $L = 2$ and the modulation index used is $h_f = \frac{1}{2}$.

Let us consider the example of a binary partial response DFM system spread over $L = 2$ bit intervals and with a modulation index of $h_f = \frac{2 \cdot k}{p} = \frac{1}{2}$, where k and p in Equation 1.12 will be 1 and 4, respectively. Note that the value of the modulation index h_f, in this system is not necessarily an attractive setting. It was only chosen to illustrate the principles discussed previously. In this DFM system the legitimate phase states, θ_n, are as follows:

$$\theta_n \in \left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}.$$

For each phase state θ_n in the trellis, there are 2^{L-1} associated correlative state vectors, where L indicates the extent of the spreading. Hence, for pulses spread over two bit intervals, there will be a total of $4 \cdot 2^{2-1} = 8$ states in the trellis. The transitions from each of these states are illustrated in Figure 1.5 and governed by Equation 1.16. Specifically, for the trellis state $S_n = (\theta_n, \alpha_{n-1}) = (0, 1)$ at the top of Figure 1.5, there are two legitimate trellis transitions, depending on the current incoming bit α_n. Note that the dashed lines indicate transitions due to $\alpha_n = +1$, while the continuous lines represent transitions due to $\alpha_n = -1$. Hence, for each state S_n there are two legitimate trellis transitions, and similarly, there are two transitions merging in each state S_{n+1}. We note at this early stage that the MLSE receiver technique [37, 3] is efficient in partial response systems, since certain transitions are illegitimate, such as a transition from state $(0, 1)$ to $(0, 1)$ in Figure 1.5, hence allowing the receiver to eliminate the associated estimated received bits from its decisions.

In summary, DFM systems can be viewed as a form of Frequency Shift Keying (FSK) with a constant signal envelope and phase continuity at bit intervals and boundaries. They may also be further classified as full response or partial response systems, depending on the extent of spreading, L. For full response systems we have $L = 1$, while partial response schemes have $L > 1$. Spreading the frequency shaping function, $g(t)$, over a higher number of bit intervals improves the spectral efficiency, but at the expense of higher complexity at the demodulator due to the increased contribution of ISI. As a consequence of the spreading in the phase and frequency shaping functions, the transmitted signals exhibit memory, since the bits in the data sequence are interdependent. In this situation, the optimum form of detection is Maximum Likelihood Sequence Estimation (MLSE) [37, 3]. By constructing the DFM modulator as a system of trellis states, the complexity incurred in implementing MLSE is reduced in comparison to a modulator with a phase tree structure. We now proceed by describing two DFM systems, namely Minimum Shift Keying [38, 39] and Gaussian Minimum Shift Keying [8], in more detail.

1.6.1 Minimum Shift Keying

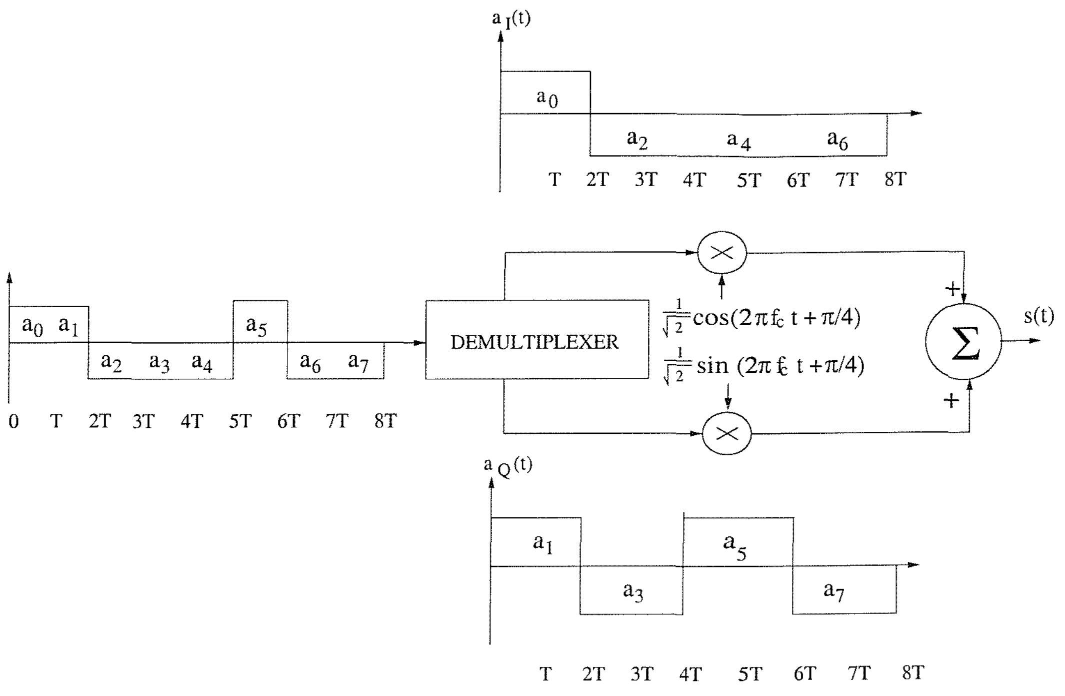
The basic theory behind Minimum Shift Keying (MSK) can be described through a brief discussion of two other modulation schemes, Quadrature Phase Shift Keying (QPSK) and Offset Quadrature Phase Shift Keying (OQPSK) [40].

QPSK is a modulation technique, which imposes a set of patterns on the phase of the carrier signal, depending on the incoming data bits. Figure 1.6(a) is an illustration of a QPSK modulator. The binary stream seen at the left of the figure is demultiplexed into odd and even index data bits, which subsequently modulate a cosine and sine carrier, hence giving,

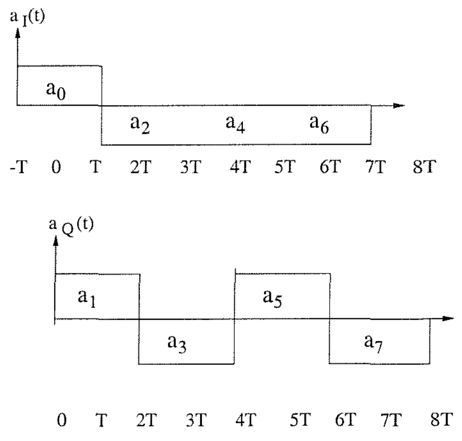
$$s(t) = \frac{1}{\sqrt{2}} a_I \cos\left(2\pi f_c t + \frac{\pi}{4}\right) + \frac{1}{\sqrt{2}} a_Q \sin\left(2\pi f_c t + \frac{\pi}{4}\right), \tag{1.18}$$

where f_c is the carrier frequency, while a_I and a_Q are the demultiplexed bits in the in-phase and quadrature arms, respectively.

In QPSK, the data bits of both quadrature arms are aligned and they are clocked synchronously. Therefore, the phase can only change once at every interval of $2T$. From Equation 1.18, the signal space diagram of Figure 1.7, can be constructed, in order to illustrate the possible phase transitions at every two-bit intervals. The signal space diagram shows a phase transition of $\pm\frac{\pi}{2}$ radians, when only one of the data bits in the quadrature



(a) A QPSK modulator. The data streams in quadrature are time-synchronous.



(b) An offset of T between the quadrature data streams is introduced.

Figure 1.6: The difference in bit stream alignment for QPSK and OQPSK is highlighted. Due to the staggering of data streams in OQPSK, the large phase difference of π radians is avoided.

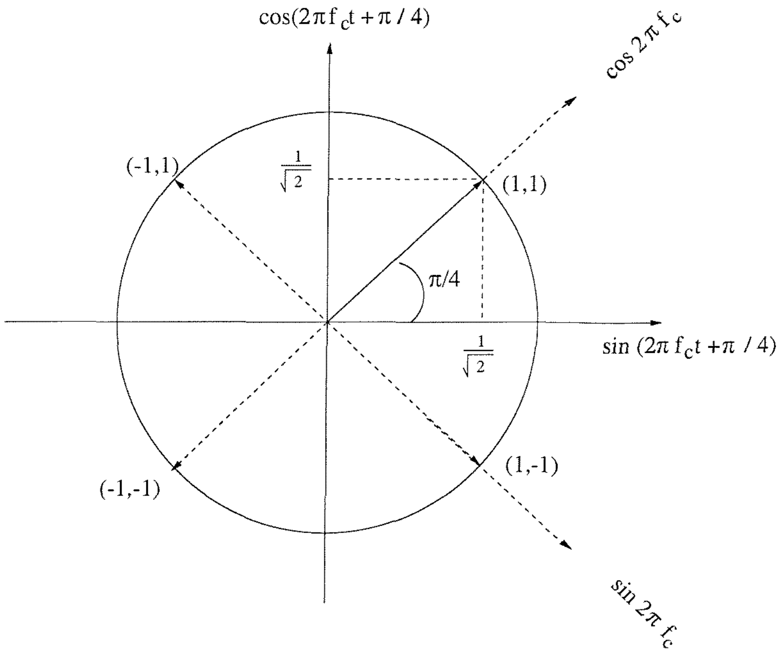


Figure 1.7: Signal space diagram for QPSK and OQPSK. The possible phase transitions for QPSK are 0, $\frac{\pi}{2}$, -$\frac{\pi}{2}$ and π radians, whereas in OQPSK the possible phase changes are 0, $\frac{\pi}{2}$ and -$\frac{\pi}{2}$ radians.

arms changes, while a transition of π radians results when both information bits change simultaneously. In OQPSK, an offset of one bit period, T, is introduced between the two binary quadrature streams. As a consequence, only one of the quadrature bits can change in every bit interval, T, hence preventing the phase difference of π radians between consecutive phase values. Consequently, the large and abrupt signal envelope swing traversing through the origin of the coordinate system of Figure 1.7 is avoided. Hence, the bandwidth required is reduced, as compared to that of QPSK. This large amplitude swing in QPSK would result in substantial out-of-band emission, unless a perfectly linear amplifier is used [6].

The shift in time alignment between the quadrature arms does not produce different power spectra. Therefore, OQPSK and QPSK have the same power spectral density. However, when band-limiting and signal envelope hard-limiting is performed, both systems respond differently. Upon band-limiting, the QPSK signal envelope becomes more distorted and at phase transitions of π radians the amplitude of the signal changes abruptly from unity to zero and back again, during the transition from (1,1) to $(-1,-1)$ in Figure 1.7. Consequently, due to hard-limiting by finite-dynamic amplifiers, for example, large and rapid changes in phase are produced. Hence, the high frequency components, which were removed by band-limiting are regenerated.

For OQPSK, band-limiting also distorts the signal. However — in contrast to QPSK — the signal amplitude no longer falls to zero, since the phase transition of π radians has been prevented. This is because the transitions from any phasor position in Figure 1.7 are now limited to the adjacent phasors, hence limiting the envelope swing. Therefore, when the signal envelope is hard-limited, the rate of signal change is limited, thereby band-limiting the high frequency components. The improvement achieved in spectral efficiency through the minimisation of the phase difference suggests that further reduction in out-of-band emission and adjacent channel interference can be obtained, if phase continuity is preserved at bit intervals. This is the motivation, which led to the development of Minimum Shift Keying (MSK) [38, 39].

MSK can be viewed as OQPSK, but with the in-phase and quadrature-phase components shaped by a half-cycle sinusoid, as highlighted below. The MSK modulated signal can therefore be expressed as:

$$\begin{aligned} s(t) &= \frac{1}{\sqrt{2}} a_I \cos\left(\frac{\pi t}{2T}\right)\cos\left(2\pi f_c t\right) + \frac{1}{\sqrt{2}} a_Q \sin\left(\frac{\pi t}{2T}\right)\sin\left(2\pi f_c t\right) \\ &= \frac{1}{\sqrt{2}}\cos\left(2\pi f_c t + b_k\frac{\pi t}{2T} + \phi_k\right) \\ &= \frac{1}{\sqrt{2}}\cos\left(2\pi t\left[f_c + \frac{b_k}{4T}\right] + \phi_k\right), \end{aligned} \tag{1.19}$$

where

$$\phi_k = \begin{cases} \pi \text{ radians} & \text{for } a_I = -1 \\ 0 \text{ radians} & \text{for } a_I = +1, \end{cases}$$

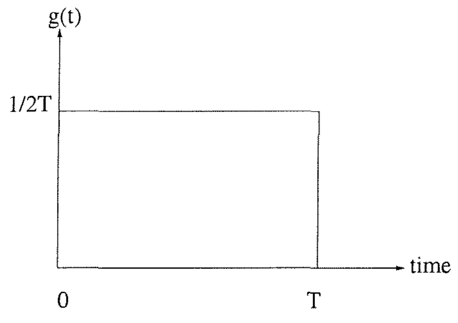
and

$$a_I, a_Q = \pm 1,\ b_k = -a_I \cdot a_Q.$$

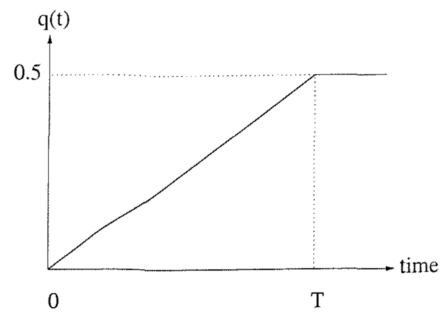
Equation 1.19 shows that MSK has a constant envelope and signalling frequencies of $f_c + \frac{1}{4T}$ and $f_c - \frac{1}{4T}$. The difference between the signalling frequencies is $\frac{1}{2T}$, which is half the bit rate and also equal to the minimum frequency spacing required for two signals to be so-called coherently orthogonal [40].

MSK can also be implemented as a DFM system by setting $g(t)$ in Figure 1.3 according to:

$$g(t) = \begin{cases} \frac{1}{2T} & \text{for } 0 \leqslant t \leqslant T \\ 0 & \text{for } t < 0 \text{ and } t > T. \end{cases} \tag{1.20}$$



(a) The impulse response of the MSK frequency shaping filter g(t).



(b) The corresponding MSK impulse response of the phase shaping filter, q(t).

Figure 1.8: The MSK impulse response q(t) is evaluated by integrating g(t).

The impulse response, $g(t)$ yields $q(t)$ in Figure 1.8(b) by using the relationship in Equation 1.5. MSK is a full response binary system with a modulation index of $h_f = \frac{1}{2}$. A value of $h_f = \frac{1}{2}$ is employed in order to ensure that the minimum frequency spacing of $\frac{1}{2T}$ — which is required for two signals to be coherently orthogonal — can be achieved so that orthogonal detection can be implemented. We note, however that the rectangular $g(t)$ time-domain pulse shaping function of Figure 1.8 implies a sinc-shaped spectrum extending theoretically over an infinite bandwidth and hence resulting in substantial spectral side-lobes, which will be illustrated at a later stage in Figure 1.14. As described in Section 1.6, the states in the trellis structure are represented by the four phase states, $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$. Assuming that the initial phase of the system is 0, a data bit of $+1$ would cause the overall phase, $\phi(t, \alpha)$, to change linearly to $\frac{\pi}{2}$. Meanwhile, a logical -1 will result in a transition from 0 to $-\frac{\pi}{2}$. Although $-\frac{\pi}{2}$ is not one of the four states in the trellis, an equivalent representation would be a transition from state 0 to $\frac{3\pi}{2}$, since the phase state is periodic with respect to 2π. By using the relationships introduced in Equations 1.16 and 1.17, the trellis diagram in Figure 1.9 can be constructed. Let us now consider, how the undesirable spectral side-lobes of MSK can be reduced by invoking a partial response pulse-shaping technique in GMSK.

1.6.2 Gaussian Minimum Shift Keying

Gaussian Minimum Shift Keying (GMSK) [8] is a form of partial response DFM, which incorporates a so-called Gaussian shaped pulse in Figure 1.10(a) as the impulse response for $g(t)$. A so-called Non-Return To Zero (NRZ) data stream of unity amplitude can be

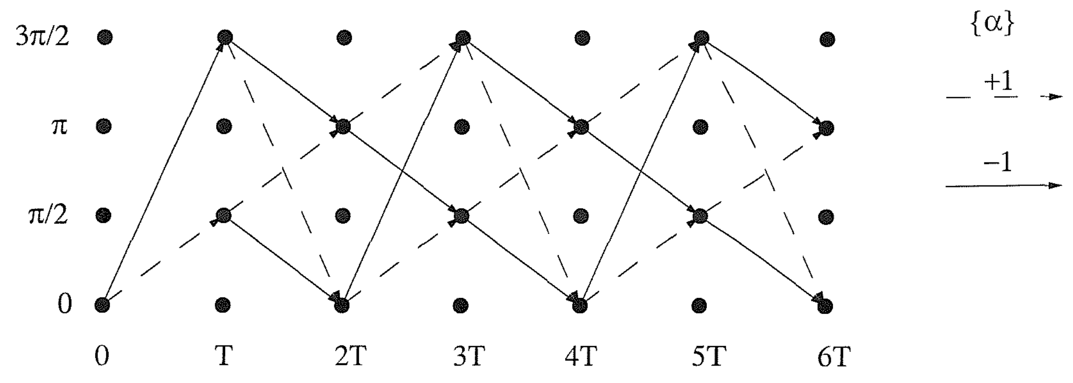


Figure 1.9: Stylised MSK trellis constructed with four phase states, $0, \frac{\pi}{2}, \pi$ and $\frac{3\pi}{2}$.

convolved with a Gaussian low pass filter, whose impulse response is [3]:

$$h_t(t) = \sqrt{\frac{2\pi}{ln2}} B_b \exp\left(-\frac{2\pi^2 B_b^2}{ln2} t^2\right) \tag{1.21}$$

to give [41],

$$g(t) = \frac{1}{2T}\left[Q\left(2\pi B_b \frac{t - T/2}{\sqrt{ln2}}\right) - Q\left(2\pi B_b \frac{t + T/2}{\sqrt{ln2}}\right)\right] \qquad 0 \leqslant B_b T \leqslant \infty, \tag{1.22}$$

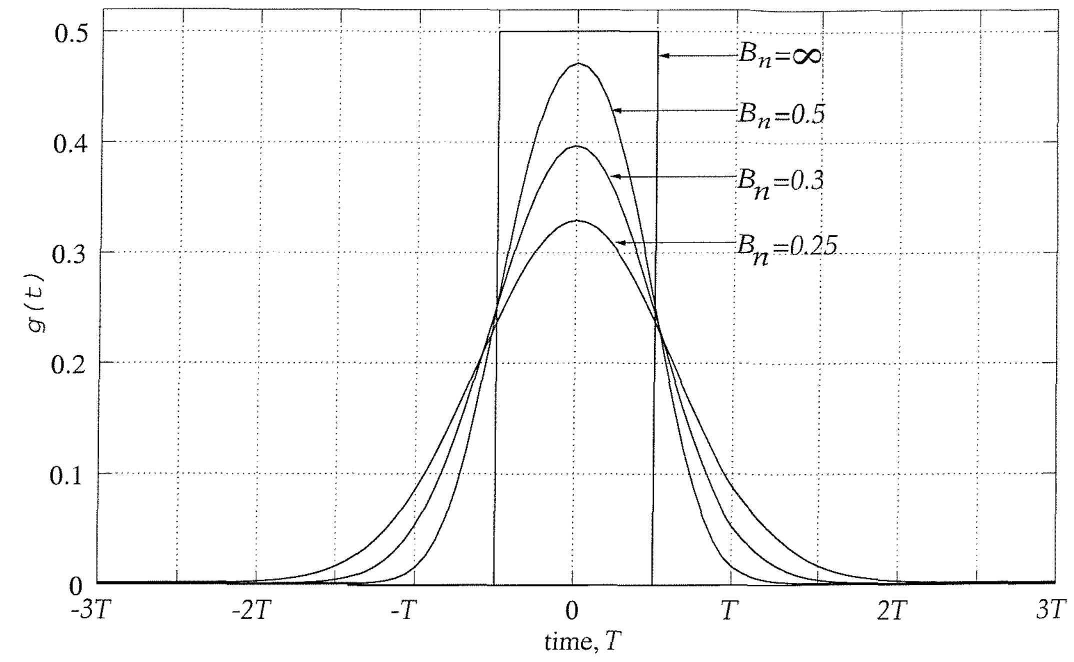
where the parameter B_b is the bandwidth of the Gaussian low pass filter, and the Gaussian Q-function, $Q(x)$ [42], is defined as:

$$Q(x) \triangleq \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{v^2}{2}\right) dv. \tag{1.23}$$

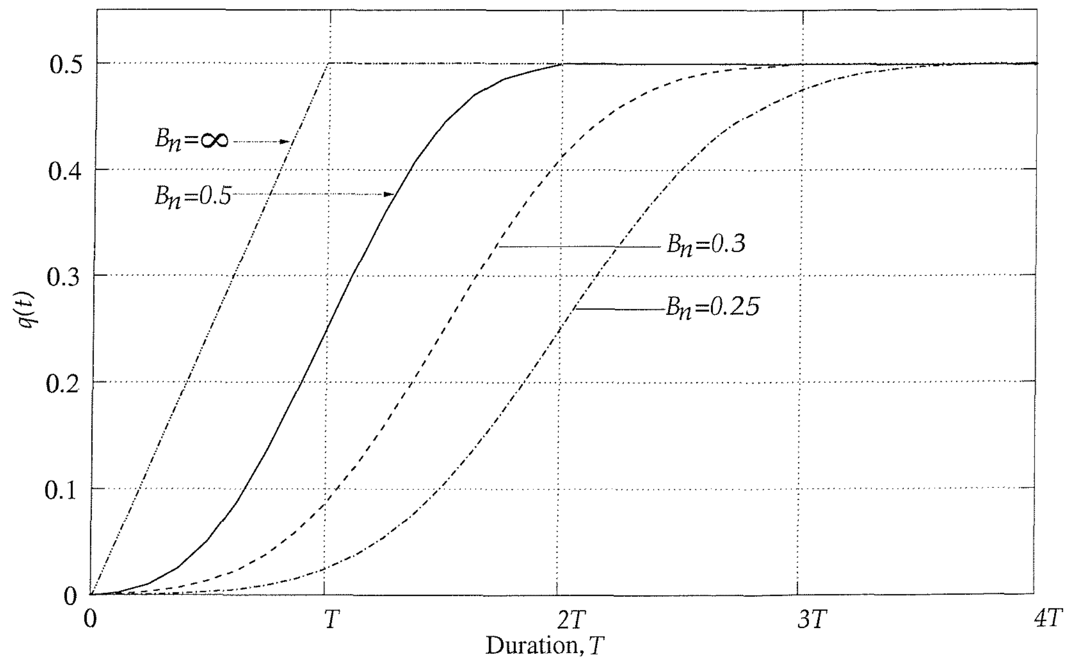
For bit-stream having a period of T, the normalised bandwidth, B_n, can be expressed as,

$$B_n = B_b T.$$

The impulse response, $g(t)$, is illustrated in Figure 1.10(a) for different values of normalised bandwidth, B_n. It is observed that GMSK with $B_n = \infty$ is the full response MSK system. In order to implement a GMSK modulator, the impulse response $g(t)$ must be symmetrically truncated to L bit periods. For B_n of 0.5 the impulse response, $g(t)$, has a pulse width of approximately two bit periods. Therefore, $g(t)$ may be truncated to two bit intervals, i.e. $L = 2$. With different limits of truncation depending on the normalised bandwidth B_n, the corresponding impulse response $q(t)$ in Figure 1.10(b) is obtained using the integrals of Equation 1.5. Typically, for partial response GMSK, the filter impulse response $g(t)$ is truncated by $\lfloor \frac{1}{B_n} \rfloor$ bit periods, where $\lfloor \cdot \rfloor$ represents the nearest integer floor. For example, when $B_n = 0.3$, $g(t)$ will be truncated to $\lfloor \frac{1}{B_n} = 3.33 \rfloor = 3$ bit periods.



(a) Frequency-shaping impulse response, $g(t)$, for different values of B_n.



(b) The corresponding pulse-shaping impulse response, $q(t)$ is obtained by integrating $g(t)$ after symmetrically truncating $g(t)$ to L bit intervals. The origin in the time axis, is used as a reference point and represents the point of truncation.

Figure 1.10: Plots of various $g(t)$ and $q(t)$ GMSK shaping functions. The normalised bandwidth, B_n, determines the extent of the partial-response spreading. In practice, $g(t)$ is symmetrically truncated to $\lfloor \frac{1}{B_n} \rfloor$ bit intervals, where $\lfloor \cdot \rfloor$ denotes the closest integer floor.

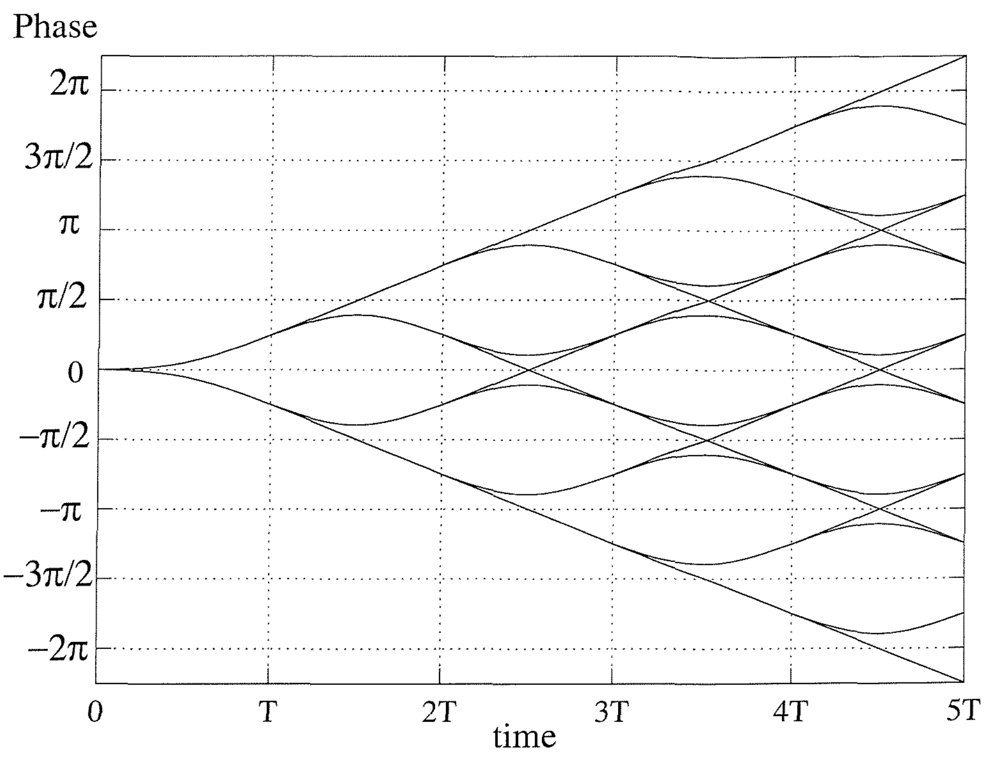


Figure 1.11: GMSK phase tree, where the impulse response $g(t)$ is spread over two bit intervals. The modulation index used is $\frac{1}{2}$.

The objective of introducing a Gaussian filter is to prevent the instantaneous changes in frequency, which exist in MSK, when switching from $f_c + \frac{1}{4T}$ to $f_c - \frac{1}{4T}$. Hence, as depicted in Figure 1.14 in conjunction with Gaussian filtering, the power spectrum obtained is more tight and has lower side lobes than MSK. Further improvement in spectral performance is achieved by spreading $q(t)$ over L bit periods. However, the spreading also introduces Controlled Inter Symbol Interference (CISI), since the resulting phase in each bit interval is now dependent on the previous $L - 1$ bits, as well as on the current bit. The resulting phase tree is shown in Figure 1.11 for a spreading over two bit intervals. We note that the expected $\pm\frac{\pi}{2}$ phase development in this example occurs over 2T. However, these phase changes are not explicitly observable in the figure due to the superposition of consecutive phase trajectory changes.

Partial response DFM signals can be equalised by constructing the GMSK modulator as a system consisting of a finite number of states and subsequently implementing trellis-based MLSE equalisers such as the Viterbi Equaliser [3]. For a GMSK modulator with a modulation index of $h_f = \frac{1}{2}$ and a spreading over L bit intervals, there are four phase states, namely 0, $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$, and 2^{L-1} correlative states per phase state, giving a total of $4{\cdot}2^{L-1}$ states in the trellis structure. The possible transitions are governed by the relationships established in Equations 1.16 and 1.17, and the resulting state representation for the modulator is equivalent to the trellis structure illustrated in Figure 1.5. Each transition corresponds to a particular quadrature-component baseband signal waveform. Therefore, a library of legitimate signal waveforms can be stored in a look-up table for waveform-comparison with the received signal. For a binary DFM signal, the number of signal waveforms that must be stored is twice the total number of states in the trellis.

Both the full response MSK and partial response GMSK system have been discussed. MSK can be viewed as a DFM system with phase continuity but frequency discontinuity at bit boundaries or as an OQPSK system, where consecutive phase-tree transitions are shaped by a half-cycle sinusoid. By preventing phase discontinuities at bit boundaries, the power spectrum obtained has lower side lobes compared to OQPSK and QPSK. However, at bit transitions, there are instantaneous changes in frequency. In GMSK, a Gaussian filter is employed, in order to prevent these instantaneous changes in frequency. Furthermore, by spreading the impulse response of the filter over a finite number of bit intervals, a more tight spectrum than that of MSK is obtained. The improved spectral performance is however achieved at the expense of considerable Controlled ISI (CISI). Hence, equalisation must be performed at the demodulator, even in the absence of channel-induced dispersion. Let us now consider the spectral-domain properties of these signals in the next section in more detail.

1.7 Spectral Performance

With increasing demand for spectrum, the spectral occupancy of the modulated signal becomes an important consideration in the design of modulation schemes. In the following discussion, the spectral characteristics of the modulated signals considered in this treatise are evaluated by computing the Power Spectral Density (PSD) and the fractional out-of-band power, P_{ob}.

1.7.1 Power Spectral Density

The power spectral density of the modulated signal is a measure of the distribution of power with respect to the frequency. Here, the so-called Welch method [43] is employed in order to estimate the spectra of the DFM signals, which are characterised as random processes. In these experiments a pseudo-random data sequence was directed into the MSK and GMSK modulator, which produced the in-phase and quadrature-phase baseband samples. Before transformation in the frequency domain, the time domain quadrature-phase samples of the modulated signals were decomposed into k number of segments, which were overlapped by 50% between successive segments, as shown in Figure 1.12. Each segment of M samples

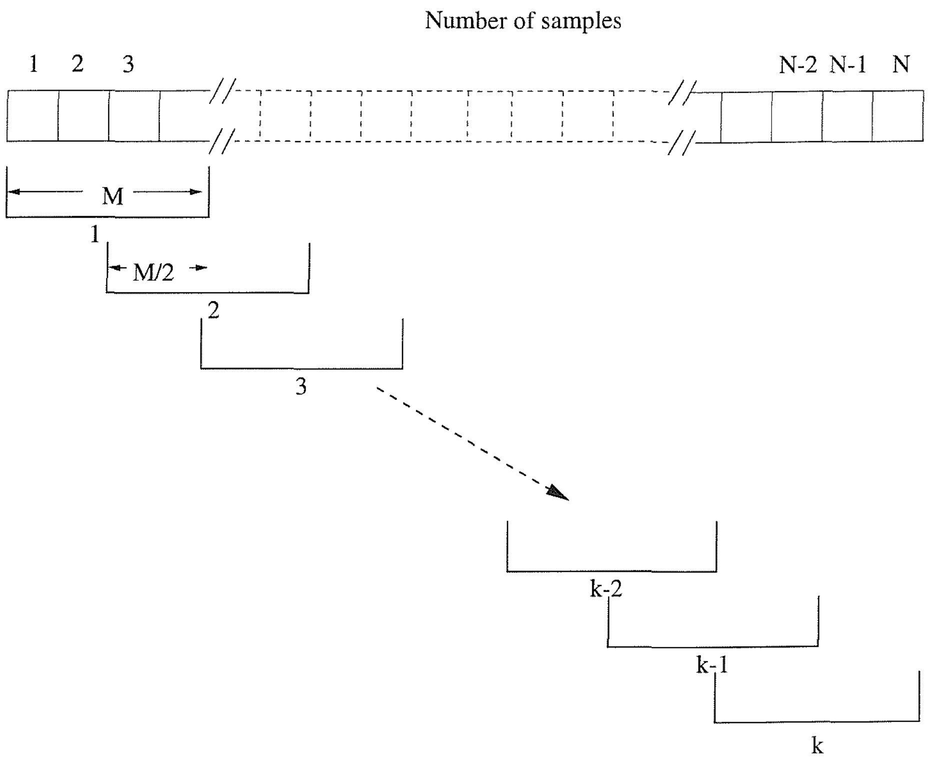


Figure 1.12: The Welch method subdivides the block of N samples into k segments. Each segment has a length M and overlaps with the successive segments by 50%.

was then multiplied with a time-domain window function. Different window functions yield subtle enhancements and degradations in terms of the sharpness of the peaks or narrowness of the computed spectral estimation. Since the purpose of this experiment is to compare the PSD of different modulation schemes, the specific choice of window function is of less significance, if the PSD of these schemes is evaluated with the same window. For our investigations, the Bartlett window [43] was chosen, which is shown in Figure 1.13 along with other commonly used windows. The Bartlett window can be expressed as:

$$w_j = 1 - \left| \frac{j - \frac{1}{2}M}{\frac{1}{2}M} \right|, \qquad \text{for } 0 \leq j \leq \text{M}, \tag{1.24}$$

where j is the sampling index. The so-called periodogram in Equation 1.25, is an estimate of the PSD, which is computed by applying the Fast Fourier Transform (FFT) to the windowed segments having M samples, in order to obtain $X(f)$. Subsequently, the magnitude of $X(f)$

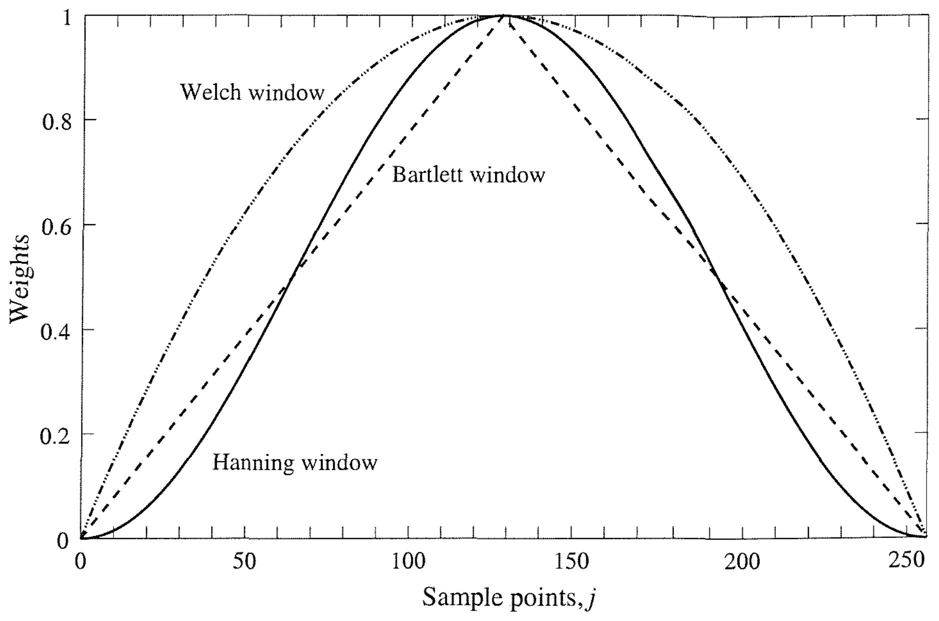


Figure 1.13: Commonly used window functions for computing the power spectral density of the modulated signal.

is squared and normalised.

$$\text{Periodogram} = \frac{1}{M}\left|X(f)\right|^2 . \tag{1.25}$$

All the k periodograms are averaged, in order to give an estimate of the PSD. A detailed mathematical treatment of PSD estimation can be found in reference [43]. The spectral density plot for MSK and GMSK having different normalised bandwidths, B_n, in Figure 1.14, was obtained by subdividing the data samples into $k = 100$ segments, each containing $M = 256$ samples.

From Figure 1.14, it is observed that the power spectrum of MSK contains significant side lobe leakage compared with the GMSK spectra. This is due to the instantaneous changes in frequency at symbol interval boundaries. In GMSK, these rapid frequency transitions are prevented by the use of a Gaussian filter, hence giving a spectrum with lower side lobes. For $B_n = 0.5$ the impulse response of the filter, $g(t)$, is extended approximately over two symbol periods. As a consequence, a more compact spectrum is obtained. Further reduction in the normalised bandwidth, B_n, produces power spectra, which occupy a further reduced bandwidth. However, this is at the expense of considerable CISI.

In Figure 1.14, the power spectrum for the modulated GMSK signal with $B_n = 0.3$ and $B_n = 0.5$ exhibit a bend at a normalised frequency of 1.2 and 1.6, respectively. This is

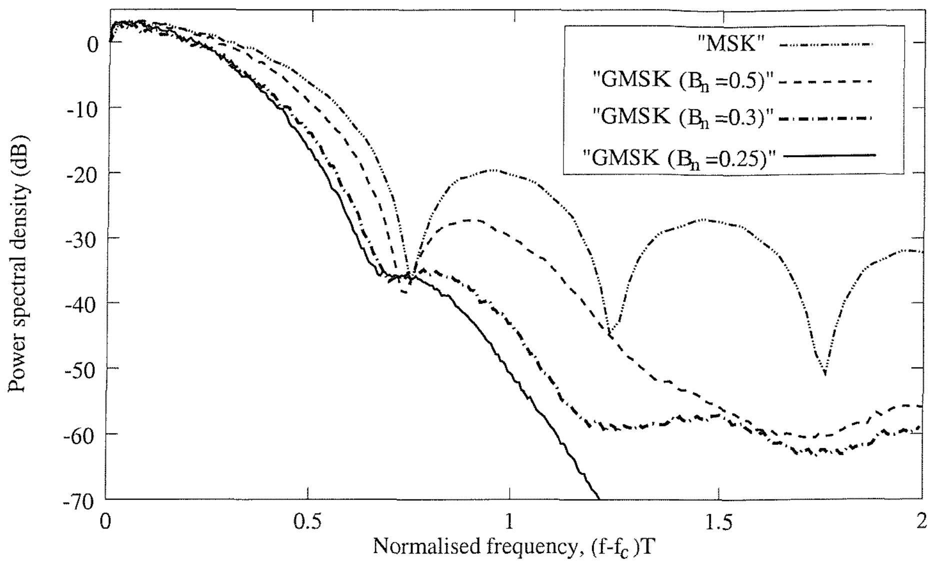


Figure 1.14: Normalised power spectral density for MSK and GMSK for different normalised bandwidths, B_n.

due to the time-domain truncation of the impulse response $g(t)$ [7], which corresponds to multiplying $g(t)$ by a rectangular window, and hence equivalent to convolving the frequency-domain transfer function $G(f)$ with the sinc-shaped window spectrum. Note that $G(f)$ is obtained through the Fourier Transform of the impulse response $g(t)$.

1.7.2 Fractional Out-Of-Band Power

The fractional out-of-band power, P_{ob}, represents the power in the region exceeding a particular bandwidth, B. An expression for P_{ob} is given by:

$$\begin{aligned} P_{ob}(B) &= \frac{\int_{-\infty}^{\infty} S(f)df - \int_{-B}^{B} S(f)df}{\int_{-\infty}^{\infty} S(f)df} \\ &= 1 - \frac{\int_{-B}^{B} S(f)df}{\int_{-\infty}^{\infty} S(f)df}, \end{aligned} \tag{1.26}$$

The function, $P_{ob}(B)$, for MSK and GMSK in Figure 1.15 was obtained by numerically integrating the PSDs in Figure 1.14. It shows the fraction of power — expressed in dB — that falls outside the range $-BT \leqslant fT \leqslant BT$, where B is a variable. As expected, the P_{ob} for MSK is much higher due to the side lobe leakage in the power spectrum. In GMSK at $B_n = 0.25$, the fraction of power leakage is -70 dB for a normalised frequency of 1.0, while MSK has a greater power leakage of -25 dB at the same normalised frequency. Since the power spectra for GMSK associated with $B_n = 0.5$ and $B_n = 0.3$ suffer from an anomalous bend owing to the time-domain truncation of $g(t)$, the function P_{ob} for these systems also

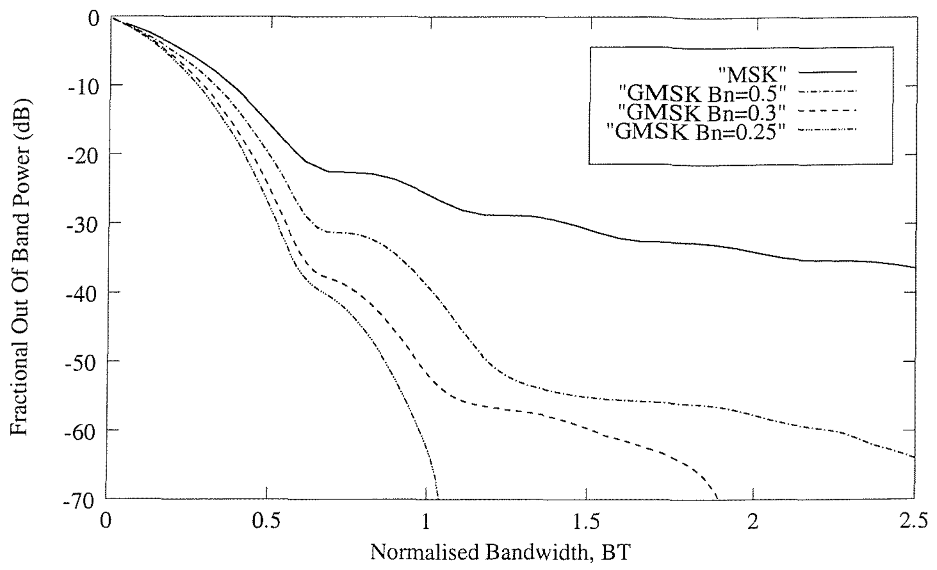


Figure 1.15: The out-of-band power for MSK and GMSK for different normalised bandwidths.

displays an irregularity. This occurs at a normalised frequency of approximately 1.1 and 1.3 for GMSK associated with $B_n = 0.5$ and $B_n = 0.3$, respectively.

Having described the fundamental principles and properties of the DFM schemes, specifically for Minimum Shift Keying (MSK) and Gaussian Minimum Shift Keying (GMSK), we now proceed by describing the basic principles of constructing trellis-based equaliser states.

1.8 Construction of Trellis-based Equaliser States

In order to describe the construction of the associated equaliser trellis states, we must first extend the principles of modulator state representation, which were introduced in Section 1.6. When the signal is transmitted over a Gaussian channel, a trellis-based equaliser has the same state representation as the modulator. However, in dispersive wideband radio channels the trellis-based equaliser must consider a larger set of possible signals, in order to remove the effects of the ISI introduced by the channel's multipath components. Hence, more states will be needed in the trellis-based equaliser in order to perform the equalisation process. We will show later how each of these trellis-based equaliser states, which incorporate information needed to mitigate the CISI introduced by the modulator filter and to remove the channel-induced ISI, can be decomposed into L_w number of modulator states, in order to regenerate the received signal estimates corresponding to a trellis-based equaliser state transition.

CSV_{EQ} in non-dispersive channels	CSV_{EQ} in truncated channel $h_w(t)$ of length L_w
$\alpha_{n-1}, \cdots, \alpha_{n-L+1}$	$\alpha_{n-1}, \cdots, \alpha_{n-L+1}, \cdots, \alpha_{n-L-L_w+1}$

Table 1.1: Correlative State Vector (CSV_{EQ}) used over non-dispersive channels and dispersive radio channels with delay spread L_w at time nT.

Let us introduce the notations CSV_{mod} and CSV_{EQ} to represent the Correlative State Vector (CSV) of the modulator and the trellis-based equaliser, respectively. The states in the trellis-based equaliser are represented by the phase state, θ_n and the Correlative State Vector CSV_{EQ} , as shown in Equation 1.15. Over dispersive multipath channels, the number of phase states remains the same. However, more data bits are required, in order to represent the CSV_{EQ} and it is dependent on $v_w = L + L_w$, which is the sum of the spreading interval L in the modulator and the length of the channel window, L_w. Table 1.1 highlights the number of bits needed in the CSV_{EQ} for non-dispersive channels and dispersive wideband radio channels. At this stage, we have established that the CSV in the demodulator must be represented with the aid of L_w additional data bits, namely bits $\alpha_{n-L}, \cdots, \alpha_{n-L-L_w+1}$, when compared to the trellis-based equaliser designed for non-dispersive channels. The additional information is needed, in order to mitigate the effects of the ISI introduced by the dispersive wideband channel.

We will now show explicitly how the information in the trellis-based equaliser state is used to regenerate the estimated received signal, which is then compared with the received signal in order to give the branch metric. At each trellis state, the phase state θ_n and the correlative state vector CSV_{EQ} are used to reproduce the GMSK modulated signals over $L_w + 1$ bit periods. As illustrated in Figure 1.16, the modulated signal over $L_w + 1$ bit periods is convolved with the estimated channel impulse response of length L_w, hence giving the expected received signal, which extends over a single bit period. There is sufficient information stored in each trellis state in order to regenerate the GMSK modulated signal over $L_w + 1$ bit periods, because each equaliser trellis state describes $L_w + 1$ modulator states, as depicted in Figure 1.17. Since the transition between each of these modulator states releases a modulated signal transition over one bit period, we obtain a resultant modulated signal, which extends over $L_w + 1$ bit periods. As mentioned previously, the estimated modulated signal is then convolved with the measured channel impulse response in order to give an estimate of the received signal.

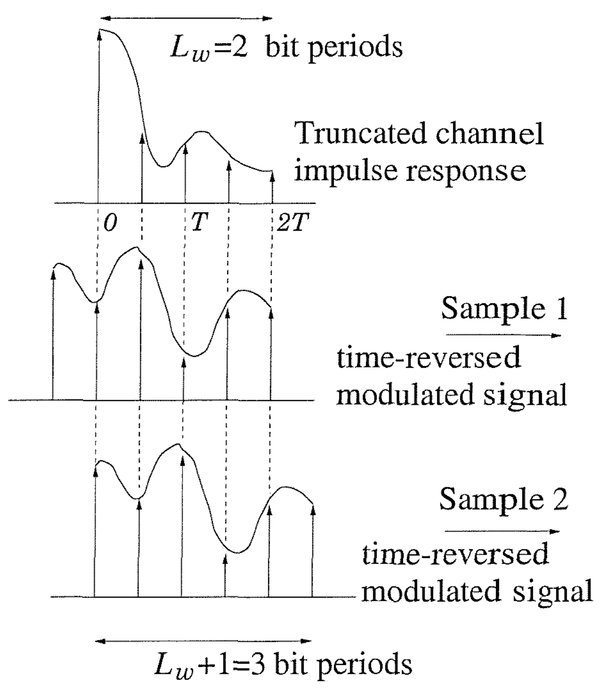

L_w=2 bit periods
Truncated channel impulse response
0
T
2T
Sample 1
time-reversed modulated signal
Sample 2
time-reversed modulated signal
L_w+1=3 bit periods

Figure 1.16: The truncated channel impulse response is over $L_w = 2$ bit periods, while the modulated signal extends over $L_w + 1 = 3$ bit periods. We obtain the regenerated signal by convolving the modulated signal with the truncated channel impulse response, in order to get the estimate of the received signal over one bit period. An oversampling by a factor of two was used in this example.

$$\boxed{\{\theta_{n-L_w} = \theta_n - \tfrac{\pi}{2}\textstyle\sum_{m=0}^{L_w-1} \alpha_{n-L-m},\ \ \alpha_{n-L_w-1}, \dots, \alpha_{n-v_w+1}\}}$$

$$\downarrow\ \alpha_{n-L_w} \longrightarrow \text{ modulated signal at time } (n - L_w)T$$

$$\boxed{\{\theta_{n-L_w+1} = \theta_n - \tfrac{\pi}{2}\textstyle\sum_{m=0}^{L_w-2} \alpha_{n-L-m},\ \ \alpha_{n-L_w}, \dots, \alpha_{n-v_w+2}\}}$$

$$\downarrow\ \alpha_{n-L_w+1} \longrightarrow \text{ modulated signal at time } (n - L_w + 1)T$$

$$\vdots$$

$$\downarrow\ \alpha_{n-2} \longrightarrow \text{ modulated signal at time } (n-2)T$$

$$\boxed{\{\theta_{n-1} = \theta_n - \tfrac{\pi}{2}\cdot\alpha_{n-L},\ \ \alpha_{n-2}, \alpha_{n-3}, \dots, \alpha_{n-L}\}}$$

$$\downarrow\ \alpha_{n-1} \longrightarrow \text{ modulated signal at time } (n-1)T$$

$$\boxed{\{\theta_n,\ \ \alpha_{n-1}, \alpha_{n-2}, \dots, \alpha_{n-L+1}\}}$$

$$\downarrow\ \alpha_n \longrightarrow \text{ current modulated signal at time } nT$$

Figure 1.17: Each of the equaliser trellis state $S_n = \{\theta_n, \alpha_{n-1}, \dots, \alpha_{n-L+1}, \dots, \alpha_{n-L-L_w+1}\}$ can be broken down into $L_w + 1$ modulator states, in order to regenerate the modulated signal $s(t, \bar{\alpha})$, which extends over $L_w + 1$ bit periods.

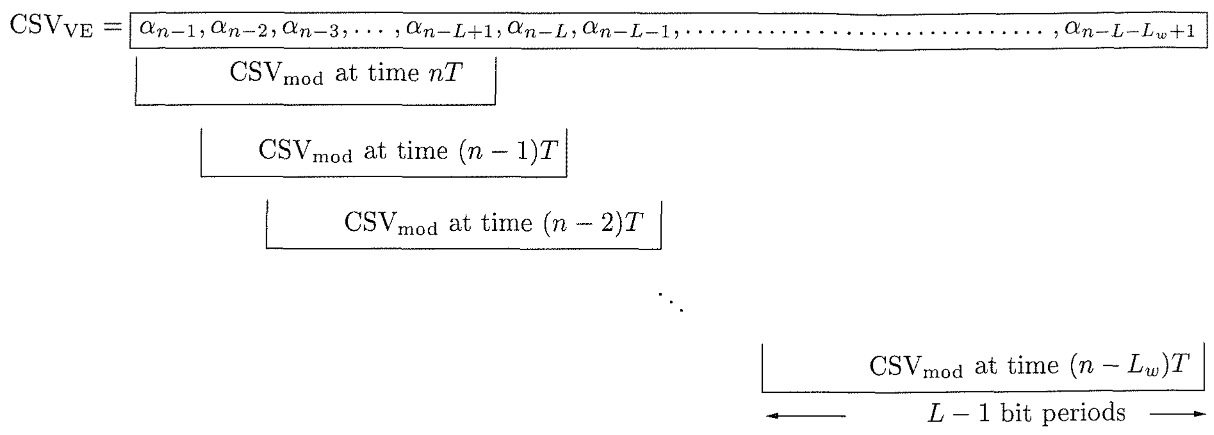

$\mathrm{CSV}_{\mathrm{VE}} = \alpha_{n-1}, \alpha_{n-2}, \alpha_{n-3}, \cdots, \alpha_{n-L+1}, \alpha_{n-L}, \alpha_{n-L-1}, \cdots, \alpha_{n-L-L_w+1}$
$\mathrm{CSV}_{\mathrm{mod}}$ at time nT
$\mathrm{CSV}_{\mathrm{mod}}$ at time $(n-1)T$
$\mathrm{CSV}_{\mathrm{mod}}$ at time $(n-2)T$
$\mathrm{CSV}_{\mathrm{mod}}$ at time $(n-L_w)T$
$L-1$ bit periods

Figure 1.18: Sliding window employed to extract the L_w previous $\mathrm{CSV}_{\mathrm{mod}}$. The current $\mathrm{CSV}_{\mathrm{mod}}$ is $\alpha_{n-1}, \cdots, \alpha_{n-L+1}$. By sliding the window one position to the right, the $(n-1)$th $\mathrm{CSV}_{\mathrm{mod}}$, which is $\alpha_{n-2}, \cdots, \alpha_{n-L}$, is determined. This sliding operation is repeated until $L_w + 1$ $\mathrm{CSV}_{\mathrm{mod}}$ are obtained.

Next, we describe the steps taken to decompose each trellis-based equaliser state into $L_w + 1$ modulator states. First, we determine the $\mathrm{CSV}_{\mathrm{mod}}$ for each modulator state and then the corresponding phase states. The current $\mathrm{CSV}_{\mathrm{mod}}$ can be obtained by choosing the first $L-1$ bits in $\mathrm{CSV}_{\mathrm{EQ}}$. Subsequently, we skip the first bit in $\mathrm{CSV}_{\mathrm{EQ}}$ and select the next $L-1$ bits in order to determine the previous $\mathrm{CSV}_{\mathrm{mod}}$. This procedure is essentially a sliding window process, where the width of the window is $L-1$ bits and the window is shifted by one bit position to the right each time we want to extract the next $\mathrm{CSV}_{\mathrm{mod}}$ for the previous modulator states. This is illustrated in Figure 1.18.

For example, let the equaliser trellis state S_n be $\{\theta_n, \alpha_{n-1}, \cdots, \alpha_{n-L+1}, \cdots, \alpha_{n-L-L_w+1}\}$. The current $\mathrm{CSV}_{\mathrm{mod}}$ at instant nT is $\alpha_{n-1}, \alpha_{n-2}, \cdots, \alpha_{n-L+1}$ and by sliding the window one bit position to the right, it will encompass the $(n-1)$th $\mathrm{CSV}_{\mathrm{mod}}$, which consists of the data bits $\alpha_{n-2}, \alpha_{n-3}, \cdots, \alpha_{n-L}$. By repeating this sliding window operation, we are able to identify the $L_w + 1$ $\mathrm{CSV}_{\mathrm{mod}}$s. At the same time, we are capable of determining the previously transmitted bits from each modulator state since for each $\mathrm{CSV}_{\mathrm{mod}}$, the first bit on the left is the previous bit transmitted. Therefore, if the $\mathrm{CSV}_{\mathrm{mod}}$ is $-1, +1, +1$, the previous data bit was a logical -1.

We have described, how the $\mathrm{CSV}_{\mathrm{mod}}$ can be obtained from $\mathrm{CSV}_{\mathrm{EQ}}$. Now we have to determine the modulator phase states for each of these $\mathrm{CSV}_{\mathrm{mod}}$s, in order to complete the description of the modulator states and hence to regenerate the modulated signal over L_w+1

bit intervals. This estimated modulated signal is then convolved with the channel impulse response, in order to generate the estimated received signal. The phase state θ_n in the trellis-based equaliser state is also the current modulator phase state. Since the current modulator phase state, θ_n, is mathematically expressed as $\theta_{n+1} = [\theta_n + \pi h_f \alpha_{n-L+1}]$ modulo 2π, in Equation 1.17, we can rearrange this equation to give:

$$\theta_n = \theta_{n+1} - \pi h_f \alpha_{n-L+1}. \tag{1.27}$$

In order to evaluate the previous modulator phase state, we substitute n in Equation 1.27 by $n-1$, yielding:

$$\theta_{n-1} = \theta_n - \pi h_f \alpha_{n-L}, \tag{1.28}$$

while for the $(n-2)$th phase state, we get:

$$\theta_{n-2} = \theta_{n-1} - \pi h_f \alpha_{n-L-1}. \tag{1.29}$$

Substituting θ_{n-1} from Equation 1.28 into Equation 1.29, we obtain:

$$\theta_{n-2} = \theta_n - \pi h_f \left(\alpha_{n-L} + \alpha_{n-L-1}\right). \tag{1.30}$$

Repeating this, we can derive a closed form equation for the L_w previous phase states, giving:

$$\theta_{n-1-k} = \theta_n - \pi h_f \sum_{m=0}^{k} \alpha_{n-L-m} \qquad \text{for } k = 0 \ldots L_w - 1. \tag{1.31}$$

In summary, with the aid of the sliding window operation illustrated in Figure 1.18 and Equation 1.31, we can regenerate the modulated signal over $L_w + 1$ bit periods for each of the trellis-based equaliser states, by first subdividing the trellis state $S_n = \{\theta_n, \alpha_{n-1}, \ldots, \alpha_{n-L+1}, \ldots, \alpha_{n-L-L_w+1}\}$ into $L_w + 1$ modulator states and the previous L_w number of bit transitions, as seen in Figure 1.17. The purpose of decomposing each of the trellis states into $L_w + 1$ modulator states is to regenerate the modulated signal over $L_w + 1$ bit periods, which is then convolved with the estimate of the channel impulse response in order to reconstruct the estimated received signal. Together, the estimate of the received signal and the actual received signal is then used to calculate the branch metric by evaluating the correlation or the squared Euclidean distance between these signals.

For example, in a GMSK system with $L = 3$ and $L_w = 2$, the total spreading, v_w, is 5. There will be 4 phase states as before, while the number of CSV$_{\text{EQ}}$ is $2^{v_w - 1}$ or 16 for each phase state, giving a total of $4 \cdot 16 = 64$ states in the trellis-based equaliser. Let us assume that we are currently in the

$$\boxed{\{\theta_{n-2}, \alpha_{n-3}, \alpha_{n-4}\} = \{\tfrac{3\pi}{2}, \ +1, +1\}}$$

$\downarrow \ \alpha_{n-2} = +1 \longrightarrow$ modulated signal at time $(n-2)T$

$$\boxed{\{\theta_{n-1}, \alpha_{n-2}, \alpha_{n-3}\} = \{0, \ +1, +1\}}$$

$\downarrow \ \alpha_{n-1} = -1 \longrightarrow$ modulated signal at time $(n-1)T$

$$\boxed{\{\theta_{n}, \alpha_{n-1}, \alpha_{n-2}\} = \{\tfrac{\pi}{2}, \ -1, +1\}}$$

$\downarrow \ \alpha_{n} = \pm 1 \longrightarrow$ current modulated signal at time nT

Figure 1.19: The trellis-based equaliser state $S_n = \{\frac{\pi}{2}, \ -1, +1, +1, +1\}$ contains information on the $L_w = 2$ previous bit transitions and $L_w + 1 = 3$ modulator states. From the information used to represent state S_n, we can deduce that the current nth modulator state is $\{\frac{\pi}{2}, \ -1, +1\}$, while the $(n-1)$th and $(n-2)$th modulator state, is $\{0, \ +1, +1\}$ and $\{\frac{3\pi}{2}, \ +1, +1)$, respectively.

equaliser trellis state $S_n = \{\theta_n = \frac{\pi}{2}, \ \ \text{CSV}_{\text{EQ}} = -1, \ \ +1, \ \ +1, \ \ +1\}$, where $\theta_n = \frac{\pi}{2}, \alpha_{n-1} = -1, \alpha_{n-2} = +1, \alpha_{n-3} = +1$ and $\alpha_{n-4} = +1$. By substituting these values into the decomposed modulator states in Figure 1.17, we can deduce that the $(n-1)$th modulator state was $\{\theta_{n-1}, \alpha_{n-2}, \alpha_{n-3}\} = \{0, \ +1, +1\}$ and that the $(n-1)$th bit transmitted was $\alpha_{n-1} = -1$, hence reaching the current nth modulator state $\{\theta_n, \alpha_{n-1}, \alpha_{n-2}\} = \{\frac{\pi}{2}, -1, +1\}$, as illustrated in Figure 1.19. Note that the phase state is periodic with respect to 2π. Therefore, the phase state at instant $n-2$ is $\frac{3\pi}{2}$ and not $-\frac{\pi}{2}$. For each trellis-based equaliser state in this system we can determine the previous modulator states and bit transitions. Therefore, being able to extract this information and by considering the latest data bit, α_n, we can regenerate the modulated sequence, which extends over $L_w + 1 = 3$ bit periods. The estimated received signal can be produced by convolving the modulated signal with the truncated channel impulse response.

In summary, over dispersive channels, the number of states in the trellis-based equaliser must be increased to mitigate the effects of the ISI introduced by the multipath channel. Each equaliser trellis state contains the associated information of the L_w previous bit transitions and modulator states. Therefore, with the knowledge of the current data bit α_n leaving an equaliser trellis state, all combinations of the modulated signal can be generated over $L_w + 1$ bit periods. Subsequently, the received signal can be estimated by convolving the modulated signal with the windowed channel impulse response.

1.9 Summary

At this stage, we have provided an overview of the mobile radio channel model and also introduced the fundamental theory and properties of Digital Frequency Modulation (DFM), specifically Minimum Shift Keying (MSK) and Gaussian Minimum Shift Keying (GMSK). The basic principles of constructing the trellis states of the equaliser have also been presented. Let us now commence our discussion of turbo coding in the context of GSM-like systems.

Chapter 2

Turbo Coding in GSM

In 1993, Berrou, Glavieux and Thitimajshima introduced the concept of turbo coding [9, 44], and showed that its performance approximates the Shannonian limit. Turbo encoders encodes the data or information sequence twice, typically using a half rate Recursive Systematic Convolutional (RSC) encoder. As illustrated in Figure 2.1, the input data sequence is directed towards the first RSC encoder, but the same data sequence is also input into an interleaver prior to passing it to the second encoder. This is to ensure that the encoded sequences will be statistically independent of each other. The interleaver introduces time diversity and therefore, when a particular bit is erroneously decoded in the first decoder due to channel impairments, it can still be corrected in the second decoder, since this particular bit has been mapped to a different time instant due to the permutations introduced by the turbo interleaver. Hence, as the size of the interleaver increases, the performance of the turbo codes improves, since the bit sequences, which are passed into the decoders become more independent. Both decoders can then exchange information in the form of the reliability of the bits, hence aiding the error-correction process. In order to achieve the required coding rate, the outputs of both RSC encoders are punctured and multiplexed.

A commonly used structure, which performs turbo decoding through a series of iterations consists of two RSC decoders. The schematic of this turbo decoder is shown in Figure 2.2. Both decoders accept soft channel values and provide soft outputs. These soft outputs express the likelihood or the probability that the data bit was decoded correctly, given the received channel values. They are usually in the form of the so-called Log Likelihood Ratio (LLR) [45], which assists in reducing the computational complexity. As mentioned previously, the turbo decoding process is performed in a series of iterations, where information is passed from one decoder to the other in cycles, until a certain termination criterion is

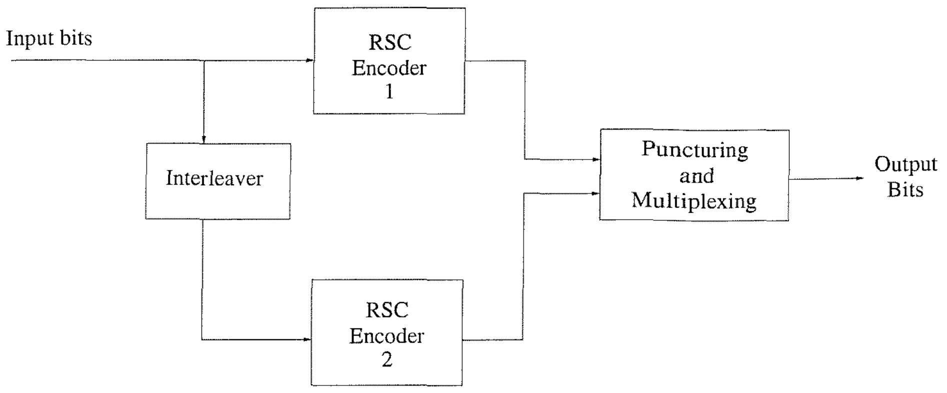


Figure 2.1: Structure of turbo encoder with RSC as component codes.

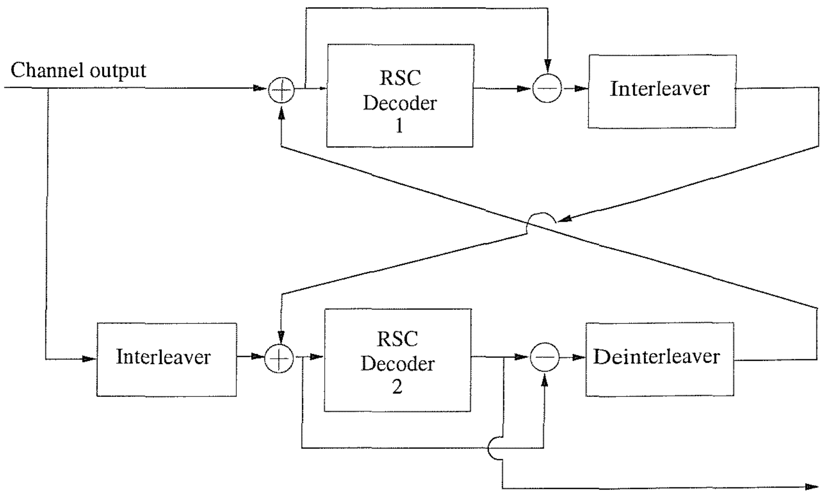


Figure 2.2: Structure of iterative turbo decoder with RSC as component decoders.

satisfied [46, 47]. A more detailed treatment of turbo codes can be found for example in reference [48].

2.1 Motivation

Turbo codes have been shown to exhibit superior performance when long turbo interleavers are employed [44, 49]. However, a delay is incurred, when filling the interleaver matrix. Hence, for delay-sensitive systems such as a speech system, the depth of the turbo interleaver must be reduced, thereby compromising the performance of the turbo code. Currently, the full-rate GSM speech system employs the conventional $\frac{1}{2}$-rate, constraint length of $K = 5$ convolutional code. In this chapter the suitability of turbo codes for the full-rate GSM speech and data systems [3] is assessed, while also considering evolutionary versions of potential low-rate GSM speech systems.

The organisation of this chapter is as follows. Sections 2.2 and 2.3 describe the Maximum *A Posteriori* (MAP) algorithm and the calculation of the Log Likelihood Ratio (LLR). The main operations of the MAP algorithm are summarised in Section 2.4. Subsequently, in Sections 2.5 and 2.6 the Log-MAP algorithm is described and summarised, respectively. This is followed by Section 2.7, which gives an overview of the complexity associated with turbo decoding and convolutional decoding operations. In Section 2.8 the performance of the turbo-coded GSM system and the convolutional-coded GSM scheme is presented and a summary of the main findings is provided in Section 2.9.

Having given a brief introduction to turbo coding, we now describe the soft-output equaliser used in the turbo-coded GSM system.

2.2 Soft Output GMSK Equaliser

In GSM, the most commonly used equaliser is the Viterbi equaliser (VE) [3, 50] although Decision Feedback Equalisers (DFE) have also been used in the past [51]. The conventional Viterbi Equaliser (VE) performs Maximum Likelihood Sequence Estimation (MLSE) by observing the development of the accumulated path metrics in the equaliser's trellis, which are evaluated recursively over several bit intervals. The length of the observation interval depends on the afforded complexity. Hard decisions are then released at the end of the equalisation process. However, since Log Likelihood Ratios (LLRs) [45] are required by the turbo decoders, we must employ specific Soft-In/Soft-Out (SISO) algorithms in place of the conventional VE. A range of SISO algorithms have been proposed in the literature such as the Maximum A Posteriori (MAP) [17] algorithm, the Log-MAP [18], the Max-Log-MAP [52, 53], or the Soft Output Viterbi Algorithm (SOVA) [54, 55, 56]. We opted for employing the Log-MAP algorithm, since it yielded identical performance to the optimal MAP algorithm, but at a significantly lower complexity. Additionally, the Log-MAP algorithm is also numerically more stable. Other schemes — such as the Max-Log-MAP and SOVA — are computationally less intensive, but provide sub-optimal performance.

Although we have chosen to use the Log-MAP algorithm in our simulations, we will describe the MAP algorithm first, in order to demonstrate that the Log-MAP algorithm is a modification of the MAP algorithm, having reduced computational complexity. We commence by showing, how the MAP algorithm provides the LLR $L(u_m|r)$ for each input bit

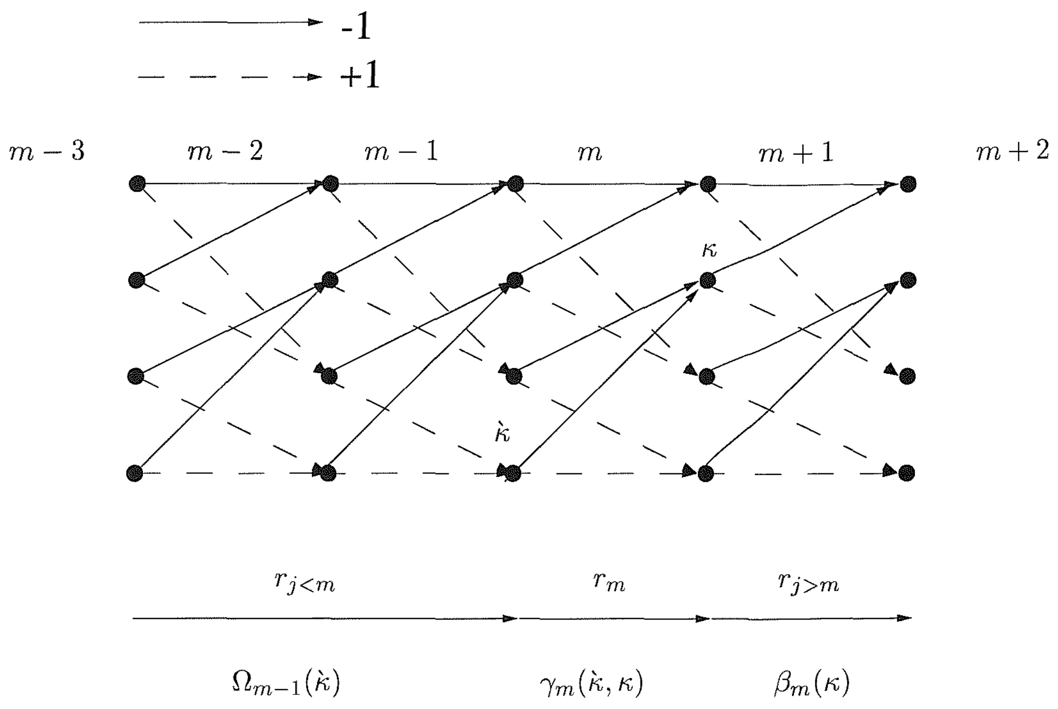


Figure 2.3: Example of binary ($M = 2$) system's trellis structure.

u_m. The LLRs, $L(u_m|r)$ can be defined as the ratio of the probability that the transmitted turbo coded bit, u_m was a logical '+1' to the probability that $u_m = -1$, given that r was the symbol sequence received, expressed as:

$$L(u_m|r) \triangleq \ln\left(\frac{P(u_m = +1|r)}{P(u_m = -1|r)}\right). \tag{2.1}$$

Depending on the implementation of the MAP algorithm, the input bit u_m can represent the channel coded or decoded information bit. When the MAP algorithm is used in an equaliser, u_m is the channel coded bit. However, when it is used as decoder, then u_m represents the source data bit. In this section, we will describe the MAP algorithm in the context of a channel equaliser employed, in order to provide soft values for turbo decoding.

By using Bayes' theorem [57] of $P(a \wedge b) = P(a|b) \cdot P(b)$, where the symbol $\wedge$ means 'and', Equation 2.1 can be rewritten to give the LLR in the form of:

$$L(u_m|r) = \ln\left(\frac{P(u_m = +1 \wedge r)}{P(u_m = -1 \wedge r)}\right). \tag{2.2}$$

For a binary system's trellis with χ states at each trellis stage, there will be χ sets of $u_m = +1$ and $u_m = -1$ bit transitions, which are mutually exclusive. This is because only one transition could have occurred in the modulator, depending on the value of u_m.

Therefore, the probability that u_m was transmitted can also be expressed as the sum of the individual probabilities of the associated χ transitions. We can then express the LLR in Equation 2.2 as:

$$L(u_m|r) = \ln\left(\frac{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=+1} P(\grave{\kappa}\wedge\kappa\wedge r)}{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=-1} P(\grave{\kappa}\wedge\kappa\wedge r)}\right), \tag{2.3}$$

where $(\grave{\kappa},\kappa) \Rightarrow u_m = +1$ denotes the set of transitions from state $\grave{\kappa}$ to κ caused by the bit $u_m = +1$ and similarly, $(\grave{\kappa},\kappa) \Rightarrow u_m = -1$ for the bit $u_m = -1$. In Figure 2.3 we show an example of a binary system, which has χ states at each stage m in the trellis. Since this is an example of a binary system's trellis, at each time interval, there are $M = 2$ transitions leaving each state and $\chi = 4$ transitions due to the bit $u_m = +1$. Each of these transitions belongs to the set $(\grave{\kappa},\kappa) \Rightarrow u_m = +1$. Similarly, there will be $\chi = 4$ number of $u_m = -1$ transitions, which belong to the set $(\grave{\kappa},\kappa) \Rightarrow u_m = -1$. In our forthcoming discussion, we will introduce the other notations in Figure 2.3, with emphasis on $\Omega_{m-1}(\grave{\kappa})$, $\beta_m(\kappa)$, and $\gamma_m(\grave{\kappa},\kappa)$.

Bahl *et al* [17] showed that by using Bayes' theorem and by splitting the received sequence r into $r_{j<m}$, r_m and $r_{j>m}$, where the term r_m represents the present continuous-valued received channel output value, while $r_{j<m}$ and $r_{j>m}$ are the previous and future received channel output value sequences, respectively, Equation 2.3 can be expressed as the product of three terms:

$$\begin{aligned} L(u_m|r) &= \ln\left(\frac{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=+1} P(r_{j<m}\wedge\grave{\kappa})\cdot P(r_m\wedge\kappa|\grave{\kappa})\cdot P(r_{j>m}|\kappa)}{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=-1} P(r_{j<m}\wedge\grave{\kappa})\cdot P(r_m\wedge\kappa|\grave{\kappa})\cdot P(r_{j>m}|\kappa)}\right) \\ &= \ln\left(\frac{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=+1} \Omega_{m-1}(\grave{\kappa})\cdot\gamma_m(\grave{\kappa},\kappa)\cdot\beta_m(\kappa)}{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=+1} \Omega_{m-1}(\grave{\kappa})\cdot\gamma_m(\grave{\kappa},\kappa)\cdot\beta_m(\kappa)}\right). \end{aligned} \tag{2.4}$$

In the derivation of this equation, we assumed that the channel is memoryless. The symbol

$$\Omega_{m-1}(\grave{\kappa}) = P(r_{j<m}\wedge\grave{\kappa}) \tag{2.5}$$

is the probability that we are in state $\grave{\kappa}$ at instant $m-1$ and that the previous continuous-valued received channel output sequence is $r_{j<m}$. The notation

$$\gamma_m(\grave{\kappa},\kappa) = P(r_m\wedge\kappa|\grave{\kappa}), \tag{2.6}$$

indicates the probability that given that we are in state $\acute{\kappa}$ at trellis stage $m-1$, we receive the channel output r_m at instant m, resulting in a transition to state κ. Lastly,

$$\beta_m(\kappa) = P(r_{j>m}|\kappa), \tag{2.7}$$

is the probability that at instant m we will receive the future channel output value sequence $r_{j>m}$ with the condition that the present state at instant m is κ. By using Equations 2.4, 2.5, 2.6 and 2.7 we can calculate the LLR for each input bit, u_m. In the following subsection, we will describe, how each of these three variables, namely $\Omega_{m-1}(\acute{\kappa})$, $\gamma_m(\acute{\kappa},\kappa)$ and $\beta_m(\kappa)$ can be determined recursively, in an effort to maintain as low a complexity, as possible.

2.3 The Calculation of the Log Likelihood Ratio

First, we consider the calculation of $\Omega_m(\kappa)$, which will become $\Omega_m(\acute{\kappa})$ at instant $m+1$. Using Equation 2.5, and being at instant m, we can express $\Omega_m(\kappa)$ as:

$$\begin{aligned}\Omega_m(\kappa) &= P(\kappa\wedge r_{j<m+1}) \\ &= P(\kappa\wedge r_{j<m}\wedge r_m).\end{aligned} \tag{2.8}$$

It is shown in reference [58] that by using Bayes' theorem and writing the probability $P(\kappa\wedge r_{j<m}\wedge r_m)$ as the sum of the joint probabilities $P(\kappa\wedge\acute{\kappa}\wedge r_{j<m}\wedge r_m)$ over all possible states $\acute{\kappa}$, Equation 2.8 becomes:

$$\begin{aligned}\Omega_m(\kappa) &= \sum_{\text{all } \acute{\kappa}} P(\kappa\wedge\acute{\kappa} \wedge r_{j<m}\wedge r_m) \\ &= \sum_{\text{all } \acute{\kappa}} P([\kappa\wedge r_m]|[\acute{\kappa} \wedge r_{j<m}])\cdot P(\acute{\kappa} \wedge r_{j<m}).\end{aligned} \tag{2.9}$$

Now, we can write $P([\kappa\wedge r_m]|[\acute{\kappa} \wedge r_{j<m}])$ as $P([\kappa\wedge r_m]|[\acute{\kappa})$, since we have assumed that the channel is memoryless and therefore the probability that r_m was received and that state κ was reached at instant m, is only dependent on the previous state $\acute{\kappa}$, to give:

$$\Omega_m(\kappa) = \sum_{\text{all } \acute{\kappa}} P([\kappa\wedge r_m]|\acute{\kappa})\cdot P(\acute{\kappa} \wedge r_{j<m}). \tag{2.10}$$

Upon using Equations 2.6 and 2.5, we then have:

$$\Omega_m(\kappa) = \sum_{\text{all } \acute{\kappa}} \Omega_{m-1}(\acute{\kappa})\cdot\gamma_m(\acute{\kappa},\kappa). \tag{2.11}$$

Observing Equation 2.11, we note that $\Omega_m(\kappa)$ can be evaluated recursively, once $\gamma(\acute{\kappa},\kappa)$ is determined. At the beginning, the value of $\Omega_0(\kappa)$ is set to

$$\Omega_0(\kappa) = \begin{cases} 1 & \text{for } \kappa = S_{\text{start}} \\ 0 & \text{for } \kappa \neq S_{\text{start}} \end{cases} \tag{2.12}$$

where S_{start} is the starting state. The forward-oriented recursion of Equation 2.11 is also indicated in Figure 2.3.

Let us now derive a recursive expression for $\beta_m(\kappa)$ using a similar approach as for obtaining $\Omega_m(\kappa)$. From Equation 2.7, we have:

$$\beta_{m-1}(\acute{\kappa}) = P(r_{j>m-1}|\acute{\kappa}), \tag{2.13}$$

which is also depicted in Figure 2.3. By expressing $P(r_{j>m-1}|\acute{\kappa})$ as the sum of joint probabilities over all states κ, we have:

$$\begin{aligned}\beta_{m-1}(\acute{\kappa}) &= P(r_{j>m-1}|\acute{\kappa})\\ &= \sum_{\text{all }\kappa} P([\kappa \wedge r_{j>m-1}]|\acute{\kappa})\\ &= \sum_{\text{all }\kappa} P([\kappa \wedge r_{j>m} \wedge r_m]|\acute{\kappa})\end{aligned} \tag{2.14}$$

and employing Bayes' theorem, we can write $\beta_{m-1}(\acute{\kappa})$ as:

$$\begin{aligned}\beta_{m-1}(\acute{\kappa}) &= \sum_{\text{all }\kappa} \frac{P(\kappa \wedge r_{j>m} \wedge r_m \wedge \acute{\kappa})}{P(\acute{\kappa})}\\ &= \sum_{\text{all }\kappa} P(r_{j>m}|[r_m \wedge \acute{\kappa} \wedge \kappa]) \cdot \left[\frac{P(r_m \wedge \acute{\kappa} \wedge \kappa)}{P(\acute{\kappa})}\right]\\ &= \sum_{\text{all }\kappa} P(r_{j>m}|[r_m \wedge \acute{\kappa} \wedge \kappa]) \cdot P([r_m \wedge \kappa]|\acute{\kappa}).\end{aligned} \tag{2.15}$$

By assuming that the channel is memoryless, the term $P(r_{j>m}|[r_m \wedge \acute{\kappa} \wedge \kappa])$ can be written as $P(r_{j>m}|\kappa)$, because the probability that $r_{j>m}$ occurs depends only on the current state κ, leading to:

$$\beta_{m-1}(\acute{\kappa}) = \sum_{\text{all }\kappa} P(r_{j>m}|\kappa) \cdot P([r_m \wedge \kappa]|\acute{\kappa}). \tag{2.16}$$

Lastly, upon invoking Equations 2.6 and 2.7, we arrive at

$$\beta_{m-1}(\acute{\kappa}) = \sum_{\text{all }\kappa} \beta_m(\kappa) \cdot \gamma_m(\acute{\kappa}, \kappa). \tag{2.17}$$

Similarly to $\Omega_m(\kappa)$, $\beta_{m-1}(\acute{\kappa})$ can also be evaluated recursively, given the value of $\gamma_m(\acute{\kappa}.\kappa)$.

In order to determine the initial conditions for $\beta_N(\kappa)$, where N is the length of the transmitted data burst, we must consider Equation 2.17 and Equation 2.7, as pointed out by Breiling [59]. Rearranging Equation 2.7 yields:

$$\begin{aligned}\beta_{N-1}(\acute{\kappa}) &= P(r_N|\acute{\kappa})\\ &= \sum_{\text{all }\kappa} P([r_N \wedge \kappa]|\acute{\kappa}),\end{aligned} \tag{2.18}$$

and upon using Equation 2.6 we get

$$\beta_{N-1}(\grave{\kappa}) = \sum_{\text{all } \kappa} \gamma_N(\grave{\kappa}, \kappa), \tag{2.19}$$

and Equation 2.17 is re-stated as:

$$\beta_{N-1}(\grave{\kappa}) = \sum_{\text{all } \kappa} \beta_N(\kappa) \cdot \gamma_N(\grave{\kappa}, \kappa), \tag{2.20}$$

to give an expression for $\beta_{N-1}(\grave{\kappa})$. We observe that Equations 2.19 and 2.20 will only be satisfied, when

$$\beta_N(\kappa) = 1, \tag{2.21}$$

for all states κ.

Once $\gamma_m(\kappa)$ has been calculated, both $\Omega_m(\kappa)$ and $\beta_m(\kappa)$ can be determined recursively using Equations 2.11 and 2.17, and the initial conditions in Equations 2.12 and 2.21. Here, we derive a mathematical expression for $\gamma_m(\kappa)$, by invoking Equation 2.6 and using Bayes' theorem,

$$\begin{aligned}
\gamma_m(\grave{\kappa}, \kappa) &= P(r_m \wedge \kappa | \grave{\kappa}) \\
&= \frac{P(r_m \wedge \kappa \wedge \grave{\kappa})}{P(\grave{\kappa})} \\
&= P(r_m | \kappa \wedge \grave{\kappa}) \cdot \left[\frac{P(\kappa \wedge \grave{\kappa})}{P(\grave{\kappa})} \right] \\
&= P(r_m | \kappa \wedge \grave{\kappa}) \cdot P(\kappa | \grave{\kappa}) \\
&= P(r_m | s_i) \cdot P(u_m),
\end{aligned} \tag{2.22}$$

where the notation s_i represents the estimate of the transmitted signal arising from a transition from state $\grave{\kappa}$ to κ, while $P(u_m)$ is the so-called *a priori* probability for bit u_m. At the beginning of the equalisation process we have no prior knowledge concerning bit u_m. Potentially, the *a priori* information $P(u_m)$ can be obtained from independent source(s). Since we assume that the probability of transmitting the binary bit $u_m = 1$ is equal to transmitting $u_m = -1$, we assign $P(u_m)$ to be $\frac{1}{2}$ for a binary system. Essentially, $P(u_m)$ can be ignored in Equation 2.22, since it is independent of bit u_m and can therefore be treated as a constant. However, in an iterative equalisation or feedback scheme [10], the *a priori* information $P(u_m)$ can be obtained from another source — rather than from the equaliser or from the current received channel sequence — which can then be used to calculate $\gamma_m(\grave{\kappa}, \kappa)$ in Equation 2.22. In this case we will then have a non-constant value of $P(u_m)$ in consecutive iterations, which cannot be neglected as before. From Equation 2.22 we observe that in the case of the non-iterative equaliser having no feedback, where $P(u_m) = \frac{1}{2}$, the term $\gamma_m(\acute{\kappa}, \kappa)$, depends solely on $P(r_m|s_i)$. Assuming that the estimate of the transmitted signal s_i was received over a Gaussian channel, we can express $P(r_m|s_i)$ as the noise-contaminated transmitted signal given by:

$$P(r_m|s_i) = \frac{1}{\sqrt{\pi N_o}} \exp\big[-\frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m} - s_{i,k})^2\big], \tag{2.23}$$

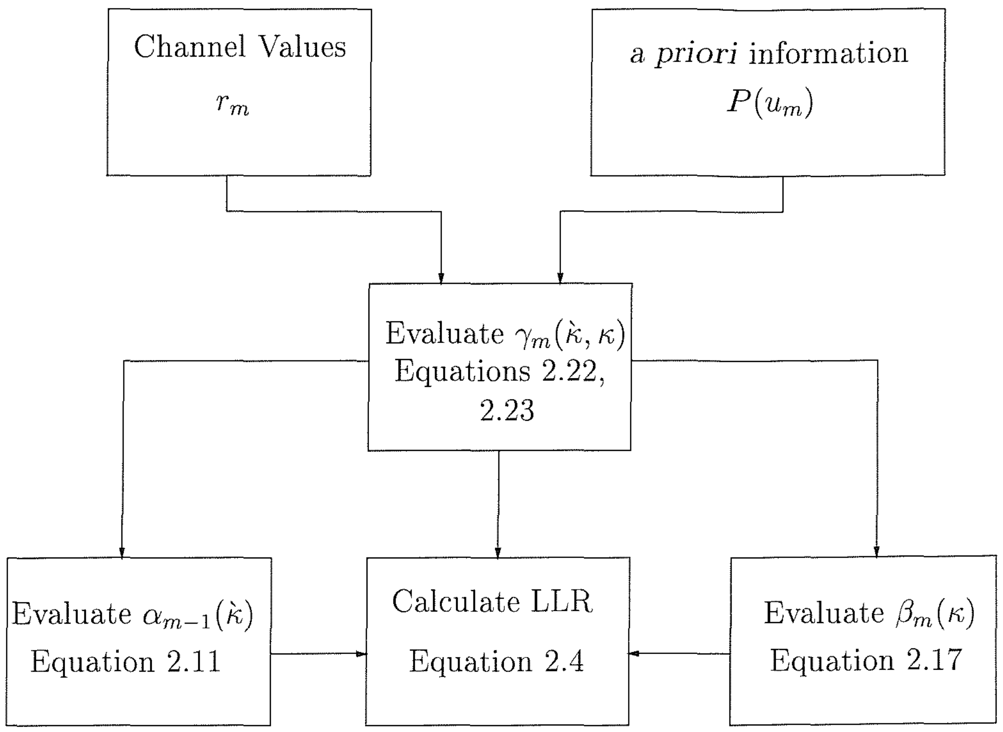
where N_s is the oversampling ratio implemented at the receiver, and $\frac{N_o}{2}$ is the double sided power spectral density of the Gaussian noise. The notations $r_{k,m}$ and $s_{i,k}$ represent the kth sample of the received signal and transmitted signal, respectively. Therefore, $\gamma_m(\acute{\kappa}, \kappa)$ is given by:

$$\begin{aligned}\gamma_m(\acute{\kappa}, \kappa) &= P(r_m|s_i)\cdot P(u_m) \\ &= \frac{1}{\sqrt{\pi N_o}} \exp\big[-\frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m} - s_{i,k})^2\big]\cdot P(u_m),\end{aligned} \tag{2.24}$$

where for a non-iterative binary system equaliser having no feedback, $P(u_m) = \frac{1}{2}$ since it is equiprobable that the bit $u_m = +1$ or $u_m = -1$ was transmitted.

2.4 Summary of the MAP Algorithm

Let us now summarise the operations required in order to calculate the LLR, which is needed by the turbo decoder. Initially, as shown in Figure 2.4, we evaluate $\gamma_m(\acute{\kappa}, \kappa)$ for the entire burst of received symbols r using Equations 2.22 and 2.23. Subsequently, using these values of $\gamma_m(\acute{\kappa}, \kappa)$ we are able to determine $\Omega_{m-1}(\acute{\kappa})$ and $\beta_m(\kappa)$ recursively using Equations 2.11 and 2.17, respectively. Having evaluated these three terms, namely $\Omega_{m-1}(\acute{\kappa})$, $\beta_m(\kappa)$ and $\gamma_m(\acute{\kappa}, \kappa)$, we can calculate the LLR $L(u_m|r)$ for each coded bit u_m, using Equation 2.4, which is passed to the turbo decoder. The complexity involved in implementing the MAP algorithm is high, since we have to consider all possible paths in the trellis, unlike in the Viterbi algorithm, which only considers the survivor paths. Therefore, the MAP algorithm was neglected for many years until the emergence of turbo coding, which needed likelihood values for each input bit. Hence much effort has been devoted to devising soft output algorithms with similar performance as the MAP algorithm, but at lower complexity. Robertson *et al* [18] proposed the Log-MAP algorithm, which gave identical performance to that of the MAP algorithm at a lower complexity. This is the topic of our next discussion.



Channel Values r_m
a priori information $P(u_m)$
Evaluate $\gamma_m(\acute{\kappa},\kappa)$ Equations 2.22, 2.23
Evaluate $\alpha_{m-1}(\acute{\kappa})$ Equation 2.11
Calculate LLR Equation 2.4
Evaluate $\beta_m(\kappa)$ Equation 2.17

Figure 2.4: Summary of key operations in the MAP algorithm, in order to evaluate the LLR for bit u_m at instant m.

2.5 The Log-MAP Algorithm

In the MAP algorithm we calculated the LLR $L(u_m|r)$ at instant m by evaluating $\Omega_{m-1}(\acute{\kappa})$, $\beta_m(\kappa)$ and $\gamma_m(\acute{\kappa},\kappa)$ in Equations 2.11, 2.17 and 2.22. We had to perform computationally exhaustive calculations, involving multiplications and evaluating natural logarithms, $\ln(\cdot)$, in order to determine these three variables. In the Log-MAP algorithm, the complexity of these recursive calculations is reduced by transforming $\Omega_{m-1}(\acute{\kappa})$, $\gamma_m(\acute{\kappa},\kappa)$ and $\beta_m(\kappa)$ into the logarithmic domain and then invoking the so-called Jacobian logarithmic relationship, which is defined as [18]:

$$\begin{aligned}\ln(\mathrm{e}^{x_1}+\mathrm{e}^{x_2}) &= \max(x_1,x_2)+\ln(1+\mathrm{e}^{-|x_1-x_2|})\\ &= \max(x_1,x_2)+f_c(|x_1-x_2|)\\ &= \max(x_1,x_2)+f_c(\delta)\\ &= J(x_1,x_2),\end{aligned} \tag{2.25}$$

where $\delta=|x_1-x_2|$ and $f_c(\delta)$ can be viewed as a correction term, while $\max(x_1,x_2)$ is

$$\max(x_1,x_2)=\begin{cases} x_1 & \text{if } x_1>x_2\\ x_2 & \text{if } x_2>x_1.\end{cases} \tag{2.26}$$

The transformation to the logarithmic domain is done by defining:

$$A_m(\kappa) \triangleq \ln(\Omega_m(\kappa)), \tag{2.27}$$

$$B_m(\kappa) \triangleq \ln(\beta_m(\kappa)), \tag{2.28}$$

$$\Gamma_m(\kappa) \triangleq \ln(\gamma_m(\grave{\kappa}, \kappa)). \tag{2.29}$$

We will show subsequently how the transformation to the logarithmic domain and the use of the Jacobian logarithmic relationship in Equation 2.25 will allow the evaluation of the LLRs through additions and table look-up operations, instead of the more complex multiplication operations, natural logarithm $\ln(\cdot)$ calculations and exponential computations.

Let us consider the forward recursive calculations for $\Omega_m(s)$. By transforming $\Omega_{m-1}(\grave{\kappa})$ to the logarithmic domain, and using Equations 2.29 and 2.11, the term $A_m(\kappa)$ can be expressed as:

$$\begin{aligned} A_m(\kappa) &\triangleq \ln(\Omega_m(\kappa)) \\ &= \ln\left(\sum_{\text{all } \grave{\kappa}} \Omega_{m-1}(\grave{\kappa}) \cdot \gamma_m(\grave{\kappa}, \kappa)\right) \\ &= \ln\left(\sum_{\text{all } \grave{\kappa}} \exp\left[A_{m-1}(\grave{\kappa}) + \Gamma_m(\grave{\kappa}, \kappa)\right]\right) \\ &= \ln\left(\sum_{\text{all } \grave{\kappa}} \exp\left[\Upsilon_f(\grave{\kappa}, \kappa)\right]\right), \end{aligned} \tag{2.30}$$

where $\Upsilon_f(\grave{\kappa}, \kappa)$ is

$$\Upsilon_f(\grave{\kappa}, \kappa) = A_{m-1}(\grave{\kappa}) + \Gamma_m(\grave{\kappa}, \kappa), \tag{2.31}$$

representing the accumulated metric for a forward transition from state $\grave{\kappa}$ to κ in Figure 2.3. Explicitly, for each transition from state $\grave{\kappa}$ to κ, we add the branch metric $\Gamma_m(\grave{\kappa}, \kappa)$ to the previous value $A_{m-1}(\grave{\kappa})$, in order to obtain the current value $\Upsilon_f(\grave{\kappa}, \kappa)$ for that path. In a binary trellis there are two branches leaving each state, and as a consequence, two possible transitions merge to every state, as illustrated in Figure 2.3. Therefore, we will have two values of $\Upsilon_f(\grave{\kappa}, \kappa)$ arising from the two branches reaching state κ, which we will refer to as $\Upsilon_{f1}(\grave{\kappa}, \kappa)$ and $\Upsilon_{f2}(\grave{\kappa}, \kappa)$. Exploiting this property — which is inherent in a binary system — Equation 2.30 can be written as:

$$\begin{aligned} A_m(\kappa) &= \ln\left(\sum_{\text{all } \grave{\kappa}} \exp\left[\Upsilon(\grave{\kappa}, \kappa)\right]\right) \\ &= \ln\left(\exp\left[\Upsilon_{f1}(\grave{\kappa}, \kappa) + \Upsilon_{f2}(\grave{\kappa}, \kappa)\right]\right). \end{aligned} \tag{2.32}$$

Since $A_m(\kappa)$ can be expressed as a natural logarithm of the sum of exponentials, the Jacobian logarithmic relationship in Equation 2.25 can be employed, hence allowing Equation 2.32 to be rewritten as:

$$\begin{aligned} A_m(\kappa) &= \ln\left(\exp\left[\Upsilon_{f1}(\acute{\kappa},\kappa) + \Upsilon_{f2}(\acute{\kappa},\kappa)\right]\right) \\ &= \max(\Upsilon_{f1}(\acute{\kappa},\kappa), \Upsilon_{f2}(\acute{\kappa},\kappa)) + \ln\left(1 + \exp[-|\Upsilon_{f1}(\acute{\kappa},\kappa) - \Upsilon_{f2}(\acute{\kappa},\kappa)|]\right). \end{aligned} \tag{2.33}$$

Although the transformation of $\Omega_m(\kappa)$ to the logarithmic domain term and the use of the Jacobian logarithmic relationship enables us to evaluate the $A_m(\kappa)$ term through recursive additions rather than using the more complex multiplication operations, we still need to compute the natural logarithm term, $\ln(1 + \exp[-|\Upsilon_{f1}(\acute{\kappa},\kappa) - \Upsilon_{f2}(\acute{\kappa},\kappa)|])$ or $f_c(\delta_f)$, as in Equation 2.25, where $\delta_f = \Upsilon_{f1}(\acute{\kappa},\kappa) - \Upsilon_{f2}(\acute{\kappa},\kappa)$. Fortunately, $f_c(\delta_f)$ can be stored as a look-up table to avoid the need for computing it repeatedly. Furthermore, Robertson *et al* [60] have shown that it is sufficient to store only eight values of δ_f, ranging from 0 to 5. Therefore, in the Log-MAP algorithm the evaluation of $A_m(\kappa)$, which is the logarithmic domain representation of $\Omega_m(\kappa)$, is less complex, compared to the MAP algorithm, since it only involves additions and a look-up table search. This look-up table is specified later on in Section 3.5 in conjunction with a numerical example of iterative joint equalisation and decoding, known as turbo equalisation.

Employing the same approach for $B_{m-1}(\kappa)$, and using Equations 2.17 and 2.28, we obtain:

$$\begin{aligned} B_{m-1}(\acute{\kappa}) &= \ln\left(\sum_{\text{all }\kappa} \beta_m(\kappa)\cdot\gamma_m(\acute{\kappa},\kappa)\right) \\ &= \ln\left(\sum_{\text{all }\kappa} \exp[B_m(\kappa) + \Gamma_m(\acute{\kappa},\kappa)]\right) \\ &= \ln\left(\sum_{\text{all }\kappa} \exp[\Upsilon_b(\acute{\kappa},\kappa)]\right), \end{aligned} \tag{2.34}$$

where $\Upsilon_b(\acute{\kappa},\kappa)$, is:

$$\Upsilon_b(\acute{\kappa},\kappa) = B_m(\kappa) + \Gamma_m(\acute{\kappa},\kappa), \tag{2.35}$$

representing the accumulated metric for the backward transition from state κ to $\acute{\kappa}$. We observe in Equation 2.35 that $\Upsilon_b(\acute{\kappa},\kappa)$ for each path from state κ to $\acute{\kappa}$ is evaluated by adding $\Gamma(\acute{\kappa},\kappa)$ to $B_m(\kappa)$. Since we are considering a binary system here, there are only two possible transitions or branches from state κ to states $\acute{\kappa}$. As before, by using this information and representing the accumulated metric from these two branches as $\Upsilon_{b1}(\acute{\kappa},\kappa)$

and $\Upsilon_{b2}(\acute{\kappa},\kappa)$, we can rewrite Equation 2.34 as:

$$\begin{aligned} B_{m-1}(\kappa) &= \ln\left(\sum_{\text{all }\kappa} \exp[\Upsilon_b(\acute{\kappa},\kappa)]\right) \\ &= \ln\left(\exp[\Upsilon_{b1}(\acute{\kappa},\kappa) + \Upsilon_{b2}(\acute{\kappa},\kappa)]\right). \end{aligned} \tag{2.36}$$

Exploiting the Jacobian logarithmic relationship in Equation 2.25, Equation 2.36 becomes:

$$\begin{aligned} B_{m-1}(\kappa) &= \max(\Upsilon_{b1}(\acute{\kappa},\kappa), \Upsilon_{b2}(\acute{\kappa},\kappa)) + \ln(1+\exp[-|\Upsilon_{b1}(\acute{\kappa},\kappa) - \Upsilon_{b2}(\acute{\kappa},\kappa)|]) \\ &= \max(\Upsilon_{b1}(\acute{\kappa},\kappa), \Upsilon_{b2}(\acute{\kappa},\kappa)) + f_c(\delta_b), \end{aligned} \tag{2.37}$$

where $\delta_b = \Upsilon_{b1}(\acute{\kappa},\kappa) - \Upsilon_{b2}(\acute{\kappa},\kappa)$. Similarly to $A_m(\kappa)$, we see that the recursive calculations for $B_{m-1}(\acute{\kappa})$ are also based on additions and the evaluation of $f_c(\delta_b)$, which — as mentioned before — can be stored in a look-up table in order to reduce the computational complexity.

From Equations 2.30 and 2.34, we observe that $A_m(\kappa)$ and $B_{m-1}(\acute{\kappa})$ can be evaluated, once $\Gamma_m(\acute{\kappa},\kappa)$ was obtained. In order to evaluate the branch metric $\Gamma_m(\acute{\kappa},\kappa)$, we use Equations 2.24 and 2.29, to give:

$$\begin{aligned} \Gamma_m(\acute{\kappa},\kappa) &\triangleq \ln(\gamma_m(\acute{\kappa},\kappa)) \\ &= \ln P(u_m)\cdot P(r_m|s_i) \\ &= \ln\left[\left(\frac{1}{2\sqrt{\pi N_o}}\exp\left[-\frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m}-s_{i,k})^2\right]\right)\cdot P(u_m)\right] \\ &= \ln\left(\frac{1}{2\sqrt{\pi N_o}}\right) - \frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m}-s_{i,k})^2 + \ln(P(u_m)). \end{aligned} \tag{2.38}$$

The term $\ln\left(\frac{1}{2\sqrt{\pi N_o}}\right)$ on the right hand side of Equation 2.38 is independent of the bit u_m and therefore can be considered as a constant and neglected. Furthermore, in a non-iterative equaliser we do not have *a priori* information concerning the bit u_m. Therefore, we can make the assumption that all bits have equal probabilities of being transmitted. Hence, $P(u_m)$ is a constant and can be neglected, enabling the branch metric $\Gamma_m(\acute{\kappa},\kappa)$ to be written as:

$$\Gamma_m(\acute{\kappa},\kappa) = -\frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m}-s_{i,k})^2, \tag{2.39}$$

which is equivalent to that used in the Viterbi algorithm.

By transforming $\Omega_m(\kappa)$, $\gamma_m(\acute{\kappa},\kappa)$ and $\beta_m(\kappa)$ into the logarithmic domain in the Log-MAP algorithm, the expression for the LLR, $L(u_m|r)$ in Equation 2.4 is also modified to give:

$$\begin{aligned} L(u_m|r) &= \ln\left(\frac{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=+1} \Omega_{m-1}(\grave{\kappa})\cdot\gamma_m(\grave{\kappa},\kappa)\cdot\beta_m(\kappa)}{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=-1} \Omega_{m-1}(\grave{\kappa})\cdot\gamma_m(\grave{\kappa},\kappa)\cdot\beta_m(\kappa)}\right) \\ &= \ln\left(\frac{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=+1} \exp\left(A_{m-1}(\grave{\kappa})+\Gamma_m(\grave{\kappa},\kappa)+B_m(\kappa)\right)}{\sum\limits_{(\grave{\kappa},\kappa)\Rightarrow u_m=-1} \exp\left(A_{m-1}(\grave{\kappa})+\Gamma_m(\grave{\kappa},\kappa)+B_m(\kappa)\right)}\right) \\ &= \ln\left(\sum_{(\grave{\kappa},\kappa)\Rightarrow u_m=+1} \exp\left(A_{m-1}(\grave{\kappa})+\Gamma_m(\grave{\kappa},\kappa)+B_m(\kappa)\right)\right) \\ &- \ln\left(\sum_{(\grave{\kappa},\kappa)\Rightarrow u_m=-1} \exp\left(A_{m-1}(\grave{\kappa})+\Gamma_m(\grave{\kappa},\kappa)+B_m(\kappa)\right)\right). \end{aligned} \tag{2.40}$$

By considering a trellis with χ number of states at each trellis stage, there will be χ transitions, which belong to the set $(\grave{\kappa},\kappa)\Rightarrow u_m=+1$, and similarly, χ number of $u_m=-1$ transitions, belonging to $(\grave{\kappa},\kappa)\Rightarrow u_m=-1$. Therefore, we must evaluate the natural logarithm $\ln(\cdot)$ of the sum of χ exponential terms. This expression can be simplified by extending and generalising the Jacobian logarithmic relationship of Equation 2.25 in order to cope with higher number of exponential summations. This can be achieved by nesting the $J(x_1,x_2)$ operations as follows:

$$\ln\left(\sum_{k=1}^{V} e^{x_k}\right) = J(x_V, J(x_{V-1},\ldots J(x_3, J(x_2,x_1)))), \tag{2.41}$$

where $V=\chi$ for our χ-state trellis. Hence, by using this relationship and Equation 2.40, the LLR values for each input bit u_m — where x_k is equal to the sum of $A_{m-1}(\grave{\kappa})$ $\Gamma_m(\grave{\kappa},\kappa)$ and $B_m(\kappa)$ for the kth path at instant m — can be evaluated.

2.6 Summary of the Log-MAP Algorithm

The Log-MAP algorithm transforms $\Omega_m(\kappa)$, $\beta_m(\kappa)$ and $\gamma_m(\grave{\kappa},\kappa)$, in Equations 2.5, 2.7, and 2.6, to the corresponding logarithmic domain terms $A_m(\kappa)$, $B_m(\kappa)$ and $\Gamma_m(\grave{\kappa},\kappa)$, using Equations 2.27, 2.28, and 2.29. These logarithmic domain variables are evaluated after receiving the entire burst of discretised received symbols, r_m. Figure 2.5 illustrates the sequence of key operations required for determining the LLR $L(u_m|r)$ of bit u_m in a trellis having χ number of states at each trellis stage. Once $A_{m-1}(\kappa)$, $B_m(\kappa)$ and $\Gamma_m(\grave{\kappa},\kappa)$ have been calculated recursively, we evaluate the sum of these three terms — as seen in Equation 2.40 — for each path or branch caused by the bit $u_m=+1$, *i.e.* all the transitions

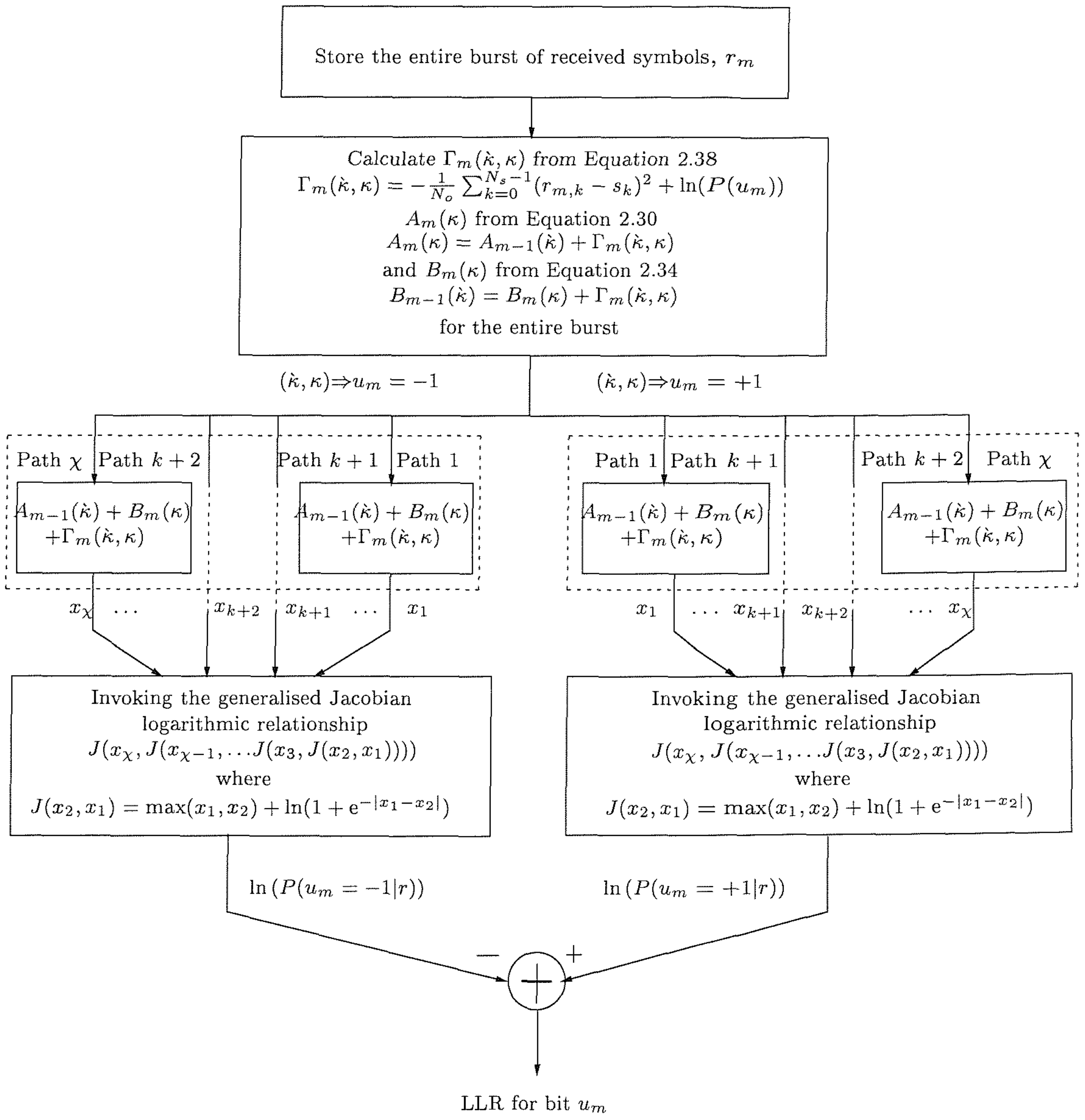

Store the entire burst of received symbols, r_m
Calculate $\Gamma_m(\acute{\kappa},\kappa)$ from Equation 2.38
$\Gamma_m(\acute{\kappa},\kappa) = -\frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{m,k}-s_k)^2 + \ln(P(u_m))$
$A_m(\kappa)$ from Equation 2.30
$A_m(\kappa) = A_{m-1}(\acute{\kappa}) + \Gamma_m(\acute{\kappa},\kappa)$
and $B_m(\kappa)$ from Equation 2.34
$B_{m-1}(\acute{\kappa}) = B_m(\kappa) + \Gamma_m(\acute{\kappa},\kappa)$
for the entire burst
$(\acute{\kappa},\kappa) \Rightarrow u_m = -1$
$(\acute{\kappa},\kappa) \Rightarrow u_m = +1$
Path χ
Path $k+2$
Path $k+1$
Path 1
$A_{m-1}(\acute{\kappa}) + B_m(\kappa) + \Gamma_m(\acute{\kappa},\kappa)$
x_χ ... x_{k+2} x_{k+1} ... x_1
Invoking the generalised Jacobian logarithmic relationship
$J(x_\chi, J(x_{\chi-1}, \ldots J(x_3, J(x_2, x_1))))$
where
$J(x_2,x_1) = \max(x_1,x_2) + \ln(1+e^{-|x_1-x_2|})$
$\ln(P(u_m=-1|r))$
$\ln(P(u_m=+1|r))$
−
+
LLR for bit u_m

Figure 2.5: Summary of key operations in the Log-MAP algorithm, in order to evaluate the LLR for bit u_m in a trellis having χ number of states, at instant m for a non-iterative equaliser.

belonging to the set $(\grave{\kappa}, \kappa) \Rightarrow u_m = +1$ at the trellis stage m between the previous state $\grave{\kappa}$ and the current state κ in Figure 2.3. We repeat the process for all paths $k = 1 \ldots \chi$ in the set $(\grave{\kappa}, \kappa) \Rightarrow u_m = -1$, at the trellis stage m between the previous state $\grave{\kappa}$ and the current state κ in Figure 2.3 as well. In order to evaluate the LLR for bit u_m, we use Equation 2.40 and the generalised expression of the Jacobian logarithmic relationship, as expressed in Equation 2.41, where k denotes the kth path ($k = 1 \ldots \chi$), which belongs to the set $(\grave{\kappa}, \kappa) \Rightarrow u_m = +1$ or $(\grave{\kappa}, \kappa) \Rightarrow u_m = -1$. Both terms on the right hand side of the LLR expression in Equation 2.40 can be determined by substituting the sum $A_{m-1}(\kappa) + B_m(\kappa) + \Gamma_m(\grave{\kappa}, \kappa)$, into the term x_k in the generalised Jacobian logarithmic relationship of Equation 2.41. In comparison with the MAP algorithm, the Log-MAP algorithm is less complex since we do not need to evaluate the LLRs using multiplications, natural logarithm and exponential calculations. Instead, we only need subtractions, additions and a look-up table. This reduction in complexity — while retaining an identical BER performance to the MAP algorithm — renders the Log-MAP algorithm an attractive option for the demodulator or decoder. There are also other algorithms exhibiting a lower complexity, such as the so-called Max-Log-MAP [52, 53] and the Soft Output Viterbi Algorithm (SOVA) [54, 55, 56], which are however suboptimal. In our work, we have used the optimal Log-MAP algorithm in order to obtain the best possible or upper bound performance of the system.

Having described the underlying principles of the MAP and Log-MAP algorithm, which was used in our GMSK equaliser, we now proceed by providing an overview of the complexity associated with turbo decoding and convolutional decoding.

2.7 Complexity of Turbo Decoding and Convolutional Decoding

In this section, the complexity of the turbo decoder (TC) and convolutional decoder (CC) is quantified in terms of the number of trellis states. The convolutional decoder employs the Viterbi algorithm, while the turbo decoder consists of two constituent decoders employing the Log-MAP algorithm. As described previously, the Log-MAP algorithms determines the LLR values, which represent the reliability of the transmitted source bits. In comparison to the Viterbi algorithm, the Log-MAP algorithm is approximately three times more complex. This is because the Log-MAP algorithms has to evaluate the forward and backward recursion values and subsequently employ these values to determine the LLRs of the source bits. By contrast, the Viterbi algorithm only performs an operation similar to the forward recursion, with the exception that only the most reliable trellis path is stored and in contrast to the Jacobian Logarithmic relationship, no correction term has to be computed.

Firstly, the number of trellis states in the convolutional decoder and the turbo decoder is determined. The number of states in the convolutional decoder is dependent on the constraint length K of the convolutional code according to the relationship of 2^{K-1}. For example, a rate $R = \frac{1}{2}$ $K = 5$ convolutional decoder requires $2^{K-1} = 16$ trellis states at each trellis interval. For the turbo decoder, the number of states required is $2 \cdot 2^{K-1} = 2^K$, since the turbo decoder consists of two constituent convolutional decoders. In addition, the complexity evaluation of the turbo decoder must also account for the number of turbo decoding iterations. Therefore, the complexity of the turbo decoder can be expressed as a function of the number of iterations, and the number of states in each trellis interval, yielding:

$$\begin{aligned} &\text{Complexity of TC} = 3 \cdot 2 \cdot \text{Number of iterations} \cdot \text{Number of TC states} \\ &\text{Complexity of CC} = \text{Number of CC states}, \end{aligned} \tag{2.42}$$

where the factor of '3' was introduced in the complexity of the turbo decoder, since the Log-MAP algorithm employed by the constituent decoders is assumed to be three times more complex than the Viterbi algorithm utilised in the convolutional decoder.

For example, turbo codes possessing a constraint length of $K = 3$ in each constituent decoders and performing eight iterations will have a complexity of $3 \cdot 2 \cdot 8 \cdot 2^{3-1} = 192$ states. When compared with the complexity of the $R = \frac{1}{2}$ $K = 5$ convolutional decoder, it is observed that the convolutional decoder has a complexity of 16 and is twelve times less complex than the turbo decoder.

Having given an estimate of the complexity associated with turbo decoding and convolutional decoding, in our following discussion we will investigate the performance of the turbo codec employed in delay-sensitive systems, such as speech systems.

2.8 Turbo Coding Performance Results

One of the key design factors in turbo codes is the choice of interleaver in the channel encoder of Figure 2.1. As described in the introductory section of Chapter 2, the input sequence is encoded twice, where the first encoder operates upon the actual input bit sequence, while the second parallel encoder acts on the interleaved bit sequence. The performance of turbo codes improves, as the interleaved and non-interleaved sequences become more statistically independent. Since upon exchanging information between them, these information sources supply two-near independent 'opinions' concerning the transmitted information. Therefore, as the depth of the turbo interleaver increases, the performance improves as well. It was also shown by Barbulescu and Pietrobon [61] that block interleavers with odd number of rows and columns perform better than a block interleaver having even number of rows and columns. This is also true, when compared with block interleavers having odd number of rows and even number of columns or even number of rows and odd number of columns. Another parameter, which affects the performance of turbo coding, is the number of iterations invoked at the decoder. In our work, the decoder is set to perform eight iterations, since experimental investigations [58] have shown that no significant improvement in BER performance is gained by using higher number of iterations.

We commence by presenting simulation results over Gaussian channels in order to highlight the effect of different interleaver depths. The turbo encoder used consists of two $\frac{1}{2}$-rate,

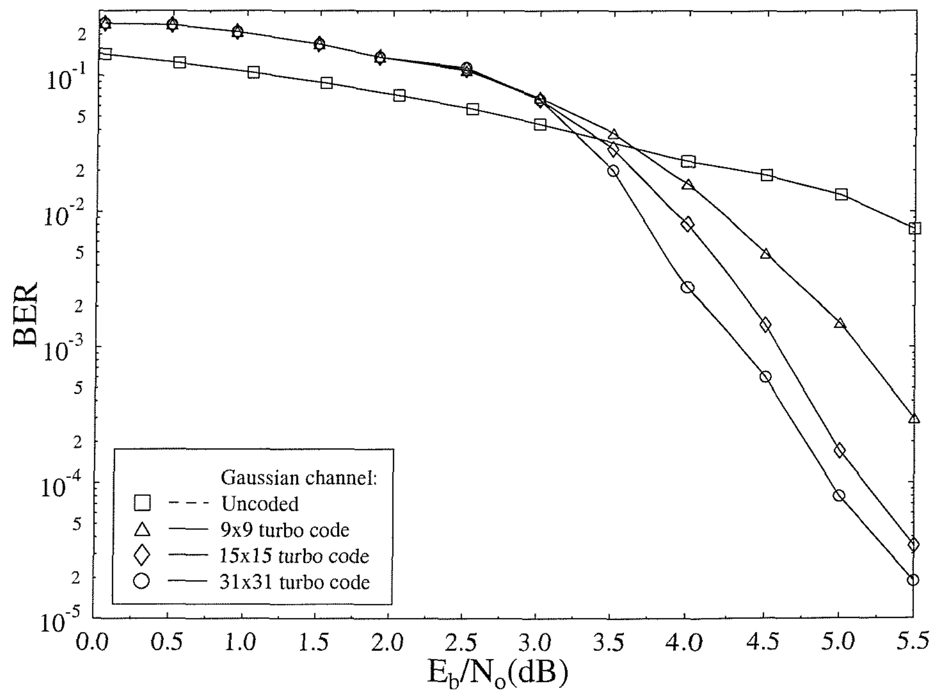


Figure 2.6: BER performance of GMSK without coding and with turbo codes using 9x9 and 31x31 interleavers, half-rate, $K = 3$ RSC encoders using the octal generator polynomial of 7 and 5.

constraint length, $K = 3$, RSC encoders, where the generator polynomials, G_0 and G_1 are $1 + D + D^2$ and $1 + D^2$, or 7 and 5, correspondingly in the octal representation. Note that we have adopted the notation RxC to indicate the dimensions of the block interleaver, which has R rows and C columns. From Figure 2.6, we observe that the BER performance improves by approximately 0.5 dB at a BER of 10^{-3} as the size of the block interleaver is increased from 9x9 to 15x15. Another 0.25 dB improvement is gained by using a 31x31 interleaver, when compared to the performance of the 15x15 turbo encoder at BER = 10^{-3}. This performance is expected, since a longer interleaver will render the two encoded sequences more statistically independent, hence improving the code's performance. However, this will incur long delays, since we must wait for the interleaver matrix to be filled, before the turbo encoding process can begin. Therefore, there must be a compromise between the delay afforded by the system, and the BER performance desired. Next, we investigate the performance of turbo coding in a low rate speech system [1].

2.8.1 Turbo Coding for a 1.9 kbps Speech Codec [1]

In this Subsection, we investigate the potential of using turbo coding in the context of low rate speech codecs, namely a 1.9 kbps speech codec [1] in order to explore whether their performance potential can be exploited in conjunction with low-delay speech systems.

	1.9 kbps GSM-like system
Speech coding rate	38 bits/20 ms = 1.9 kbps[1]
Convolutional codec	Rate=$\frac{1}{2}$, Constraint Length K=5, G_0=23, G_1=33 [3]
(RSC 1 and 2) Turbo codec	Rate=$\frac{1}{2}$, Constraint Length K=3, G_0=7, G_1=5
Turbo interleaver	9x9 block interleaver
Modem	GMSK, using the Log-MAP algorithm for equalisation
	Full-rate GSM-like system
Speech coding rate	260 bits/20 ms = 13.0 kbps[3]
Convolutional codec	Rate=$\frac{1}{2}$, Constraint Length K=5, G_0=23, G_1=33 [3]
(RSC 1 and 2) Turbo codec	Rate=$\frac{1}{2}$, Constraint Length K=3, G_0=7, G_1=5
Turbo interleaver	11x17 block interleaver
Modem	GMSK, using the Log-MAP algorithm for equalisation
	Channel specifications
Doppler frequency	41.7Hz
Impulse Response	Narrowband Rayleigh fading and COST207 TU50 channels

Table 2.1: System specifications for 1.9 kbps GSM-like system [1] and for the full-rate GSM-like system.

Since the speech information is sensitive to delays, which in practical terms must be limited to within a maximum of 100 ms, we cannot use large interleavers in order to obtain a large turbo coding gain in BER performance terms. We compare the turbo coding results with the convolutional code based benchmarks, used in GSM [3]. Our work was based on a GSM-like transceiver, which is illustrated in Figure 2.7, noting that the 1.9 kbps speech

[1]**This work was based on a collaboration [1]**

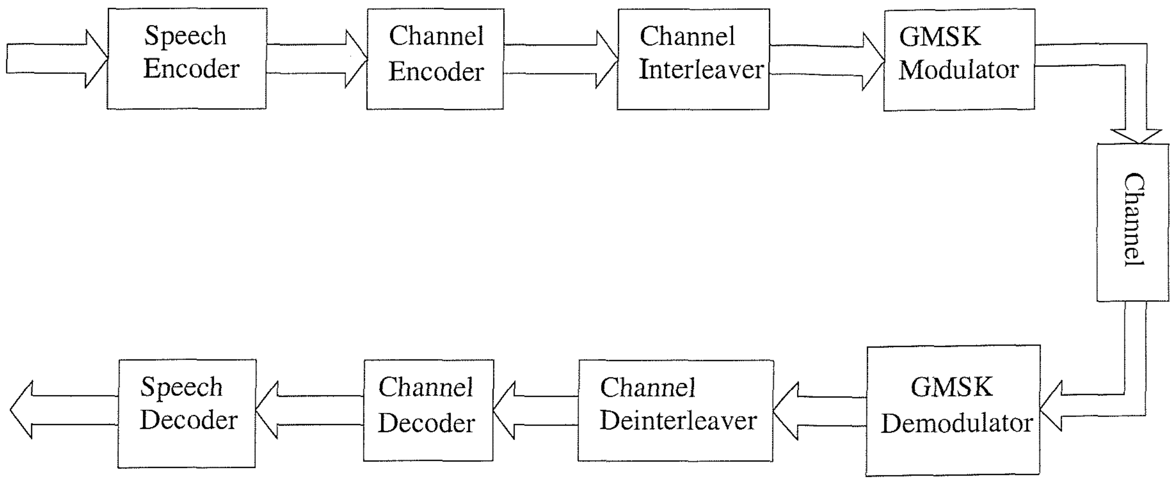


Figure 2.7: GSM-like system block diagram.

codec [1] was a proprietary scheme, generating only 38 bits per 20 ms speech frame.

In this investigation, the 38 encoded speech bits from the speech codec are channel encoded using a $\frac{1}{2}$ rate turbo encoder possessing a 9x9 block interleaver or a GSM convolutional encoder, as shown in Table 2.1. Assuming negligible processing delay, 162 bits will be released every 40 ms upon encoding two 20 ms speech frames, since the 9x9 turbo interleaver matrix employed requires two 20 ms 38-bit speech frames before channel interleaving commences. Hence, we set the transmission burst length to be 162 bits.

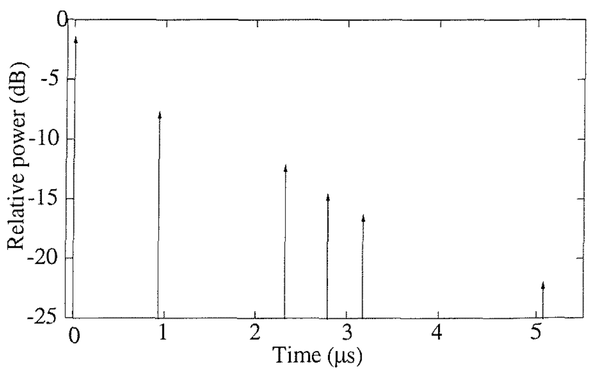


Figure 2.8: The impulse response of the COST207 typical urban channel used.

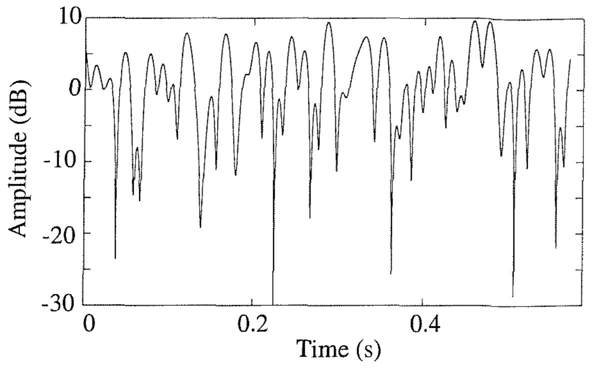
The turbo encoded speech bits are then passed to a channel interleaver in Figure 2.7. Subsequently, the interleaved bits are modulated using Gaussian Minimum Shift Keying (GMSK) [8] with a normalised bandwidth of $B_n = 0.3$ and transmitted at 270.833 Kbit/s

Typical Urban	
Position (μs)	Weights
0.00	0.622
0.92	0.248
2.30	0.062
2.76	0.039
3.22	0.025
5.06	0.004

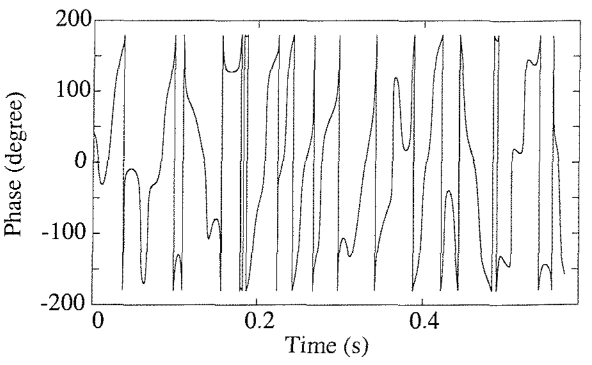
Table 2.2: The weight and delay of each path in the COST207 typical urban channel illustrated in Figure 2.8.

across the COST207 [62] Typical Urban (TU) and Rayleigh fading narrowband channel models. Figure 2.8 portrays the typical urban channel model used, where each path is fading independently with Rayleigh statistics, for a vehicular speed of 50km/h or 13.89 ms^{-1} and transmission frequency of 900 MHz. The GMSK Log-MAP demodulator equalises the received signal, which has been degraded by the wideband or narrowband fading channels with the assumption that perfect channel impulse response information is available. Subsequently, the soft outputs from the demodulator are deinterleaved and passed to the turbo decoder in order to perform channel decoding. Finally, the turbo decoded bits are directed to the speech decoder, in order to extract the original speech information. Let us now describe the channel coder, interleaver/deinterleaver scheme and the GSM-like transceiver used in our investigations.

We compare two channel coding schemes, namely the constraint length $K = 5$ convolutional codec as used in the GSM [3, 63] system, and turbo coding. Again, the turbo codec uses two $K = 3$ so-called RSC component encoders, while the decoder employs the optimal Log-MAP [18] decoding algorithm and performs eight decoding iterations. As mentioned above, turbo codes perform best for long turbo interleavers, however, due to the low bit rate of the speech codec we are constrained to using short interleaving and a short coding frame length. In our investigations, a coding frame length of 81 bits is used, with a 9x9 block interleaver within the turbo codec since we wait for two 38-bit speech frames from the speech codec. The transmission frame also contains three trellis termination bits. Hence the sum of the encoded bits and trellis termination bits gives a total of 79 bits. Therefore, another two dummy bits are included in the turbo interleaver in order to fill it completely, since it has a depth of 9x9=81 bits. The BER performance with a 9x9 turbo interleaver over Gaussian channels is shown in Figure 2.6.



(a) Rayleigh fading envelope.



(b) Rayleigh fading phase.

Figure 2.9: Rayleigh fading amplitude and phase profile examples. Each path in the mobile radio channel is Rayleigh faded independently to create a worst-case mobile radio scenario.

In a Rayleigh fading channel the transmitted burst experiences phase rotation and amplitude fluctuation, as shown in Figure 2.9, due to the time-varying channel impulse response. In a deep fade, the transmitted burst will be severely attenuated, resulting in consecutive errors, also known as burst errors, which cannot be readily combatted by the channel decoder. We can mitigate the effects of burst errors by implementing channel interleaving. The channel interleaver rearranges the positions of the bits into different locations within the transmission burst or even may allocate them to different transmission bursts, depending on the type of interleaving scheme used. At the receiver, when the signal is demodulated and deinterleaved, such that the bits are placed back into their original positions, the errors will no longer be close to each other. They are dispersed, hence improving the channel decoder's chances of decoding the encoded bits. In GSM, the channel interleaving scheme used is known as inter-burst interleaving [3, 2].

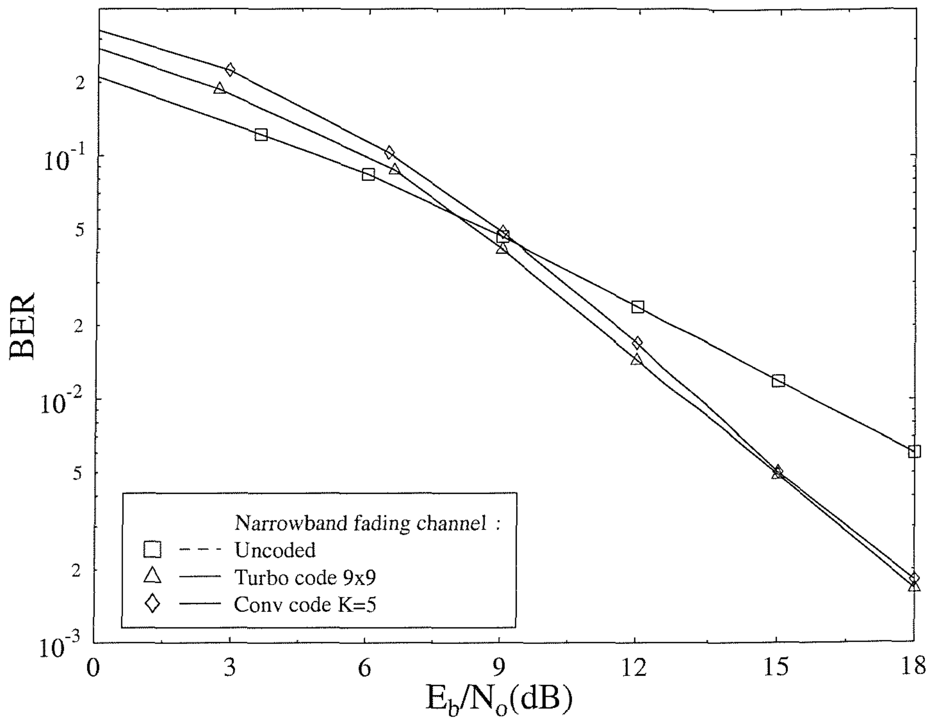


Figure 2.10: BER performance over the narrowband Rayleigh fading channel without channel coding, using rate= $\frac{1}{2}$, $K = 3$ turbo coding and rate= $\frac{1}{2}$ $K = 5$ convolutional coding.

An interburst interleaver is used to introduce time-diversity in the system, such that burst errors are mitigated, hence yielding improved coded BER performance. Briefly, the interburst interleaver in this system spreads the contents of a single 162 coded-bit burst, $C(n, k)$, into 4 sub-blocks, $I(B, j)$, each consisting of 81-bits. The notation k, is the position of the coded input bit in the mth incoming information burst $C(n, k)$, while j indicates the interleaved-bit location in the Bth sub-block. Two consecutive 81-bit sub-blocks are then combined, in order to form a transmitted data burst. This interleaving process adheres to the mapping relationship of:

$$\begin{aligned} B &= 2{\cdot}n + [k \text{ modulo } 4] \qquad k = 0, 1, \ldots, 161 \\ j &= [67{\cdot}k] \text{ modulo } 81. \end{aligned} \tag{2.43}$$

The mapping parameters for the interburst interleaving in Equation 2.43 have been

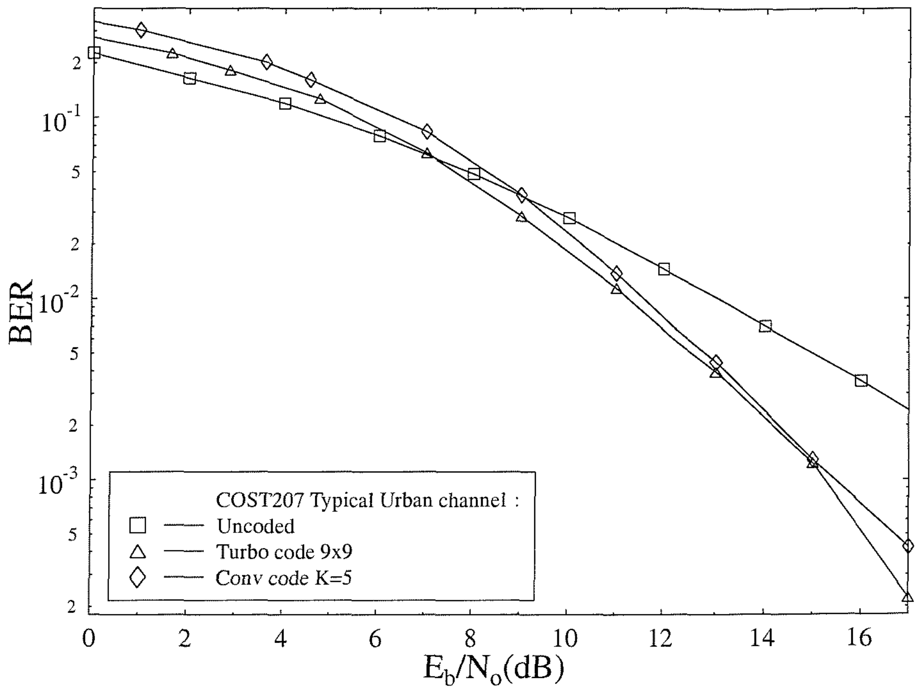


Figure 2.11: BER performance over the COST207 typical urban channel without channel coding, using rate= $\frac{1}{2}$, $K = 3$ turbo coding and rate= $\frac{1}{2}$ $K = 5$ convolutional coding.

chosen to introduce pseudo-random displacements of the coded input bits. Therefore, this scheme has the advantage of randomising the periodic fading-induced errors, since the coded input bursts $C(n, k)$ are mapped into interleaved sub-blocks $I(B, j)$ with irregular offset positions. However, additional delays are introduced, which in this case is equal to one data burst, due to the dispersion of symbols. Having described the turbo codec and channel interleaving scheme used, we proceed with the description of the GMSK transceiver.

A GMSK modulator having $B_n = 0.3$, which is employed in the current GSM mobile radio standard [3], is used in our system. As mentioned before, GMSK belongs to the class of Continuous Phase Modulation (CPM) [7] schemes, and possesses high spectral efficiency as well as constant signal envelope, hence allowing the employment of non-linear power efficient class-C amplifiers. However, this spectral efficiency is achieved at the expense of Controlled Intersymbol Interference (CISI), and therefore a channel equaliser, typically a Viterbi Equaliser is used in conventional implementations. The conventional Viterbi Equaliser (VE) [3, 50] performs Maximum Likelihood Sequence Estimation by observing the development of the accumulated path metrics in the trellis, which are evaluated recursively, over several bit intervals. The length of the observation interval depends on the complexity affordable. The results of the bit-related hard decisions are then released at the end of the equalisation process. However, since Log Likelihood Ratios (LLRs) [45] are required by the turbo decoders, in the proposed system we employ soft output algorithms instead of the VE, such as the Maximum A Posteriori (MAP) [17] or the Log-MAP [18] algorithm, which were described in Subsection 2.2 and Subsection 2.5, respectively. However, other techniques such as the reduced-complexity, but suboptimum Max-Log-MAP [52, 53], and the Soft Output Viterbi Algorithm (SOVA) [54, 55, 56] can also be used. We chose to invoke the Log-MAP algorithm, since it yielded identical performance to the optimal MAP algorithm at a much lower complexity.

2.8.2 Results and Discussion

The performance of our turbo-coded GSM-like system was compared with that of an equivalent system, but using conventional convolutional codes instead, which had the same code specifications as those in the standard GSM [3, 63] system. Figures 2.10 and 2.11 illustrated the BER performance over narrowband Rayleigh fading channel and COST207 typical urban channel. For turbo coding, we observed an E_b/N_o improvement of approximately 0.25 dB at a BER of 10^{-2} over convolutional codes. The marginal improvement of the turbo code over the convolutional code was due to the short interleaver frame length that was used in the turbo encoder imposed by the maximum 40 ms speech codec delay arrived at by concatenating two 20 ms speech frames. Hence, the investment in the higher complexity and longer delay turbo codec was not justifiable, demonstrating the important limitation of short-latency turbo-coded systems. We extended our investigation of turbo-coded GMSK systems by simulating the full rate GSM speech and data system. For the 13 kbps full-rate GSM speech codec, 260 bits were released from the speech coder every 20 ms. As illustrated in Figure 2.12, the speech encoded bits can be categorised into three different classes, namely class 1a, 1b, and 2, depending on their importance [3]. From this figure, it was observed that the class 2 bits were the least important ones and hence were not protected by any form of coding, while the class 1a bits were the most important ones and were encoded using a cyclic (53,50) block encoder, giving 53 bits. Subsequently, the cyclic encoded class 1a bits and the class 1b bits were encoded using a $\frac{1}{2}$-rate constraint length $K = 5$ convolutional coder, specified in GSM [63, 3], hence giving 378 convolutional encoded bits. Since we have used a $K = 5$ convolutional code, we required $K - 1 = 4$ tailing bits, in

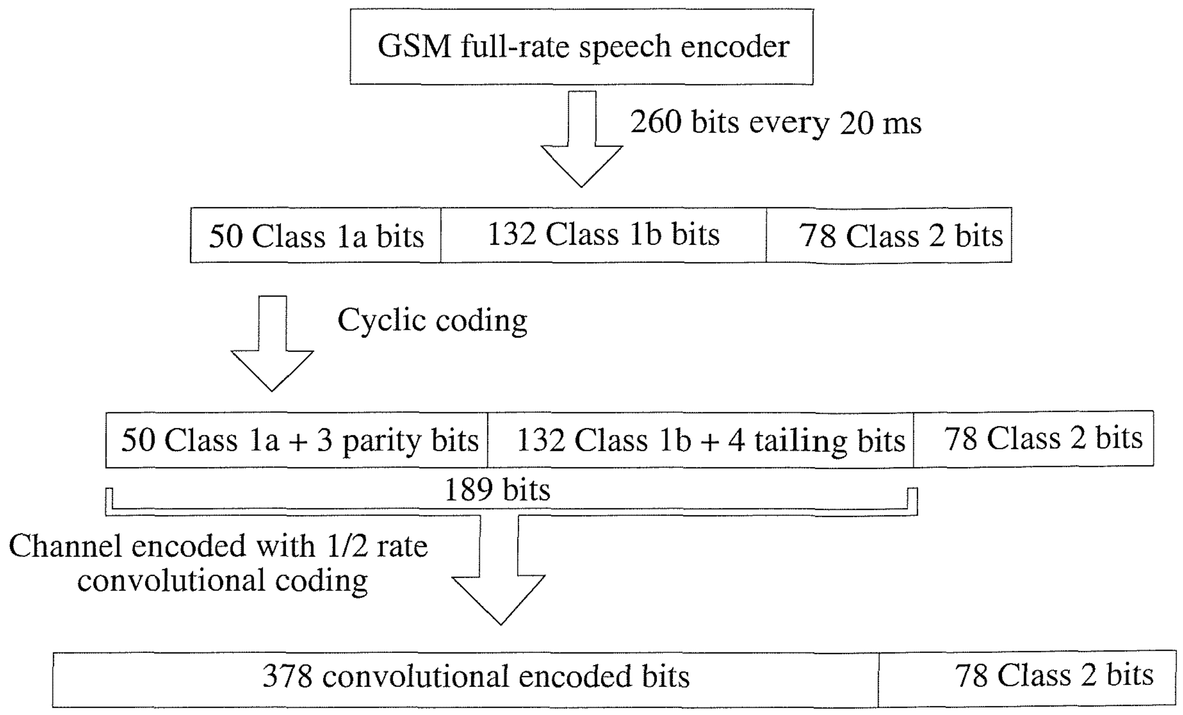
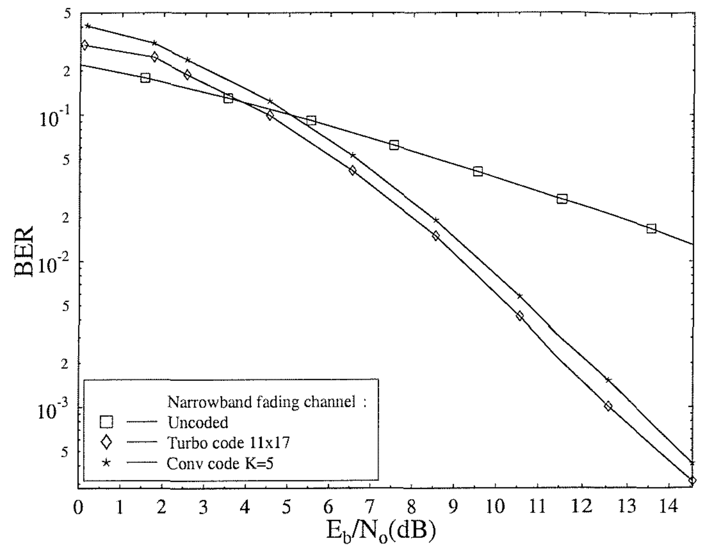

GSM full-rate speech encoder
260 bits every 20 ms
50 Class 1a bits
132 Class 1b bits
78 Class 2 bits
Cyclic coding
50 Class 1a + 3 parity bits
132 Class 1b + 4 tailing bits
78 Class 2 bits
189 bits
Channel encoded with 1/2 rate convolutional coding
378 convolutional encoded bits
78 Class 2 bits

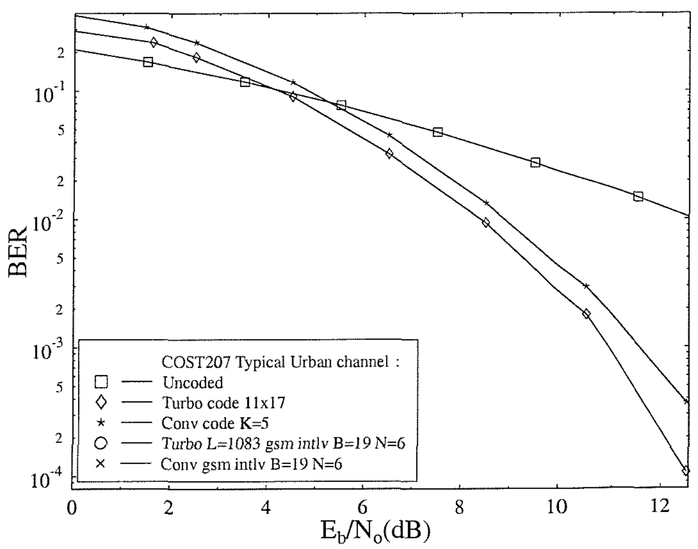
Figure 2.12: Frame structure for full-rate GSM speech coding.

order to terminate the decoder trellis in a known state. The final frame length after cyclic coding and convolutional encoding was 456 bits long, where 378 bits were from the convolutional encoder, while the other 78 bits are the class 2 speech encoded bits. The frame was subjected to interburst interleaving, which used the mapping relationship recommended by GSM [35]. At the receiver, the same Log-MAP based GMSK equaliser as in Subsection 2.2 was employed, which passed soft values to a convolutional decoder.

For the turbo-coded full-rate GSM system, the frame architecture was identical to the convolutional-coded GSM system, with the exception that the convolutional encoder and decoder was substituted by a turbo encoder and decoder, respectively. We used a $\frac{1}{2}$-rate constraint length $K = 3$ turbo decoder — as in the 1.9 kbps turbo-coded GSM-like speech system in Subsection 2.8.1 — however in conjunction with a longer turbo interleaver, namely an 11x17 interleaver, which required 187 input bits, arising from the 53 cyclic coded bits, 132 class 1b bits and the $K - 1 = 2$ tailing bits. Since we were employing a $\frac{1}{2}$-rate encoder, we obtained 374 encoded bits. At the receiver, the received signal was demodulated by the same GMSK demodulator, employing the Log-MAP algorithm. Soft values indicating the LLRs of the coded bit were passed to the iterative turbo decoder, which performed eight iterations. Simulations were conducted over the narrowband Rayleigh fading channel and COST207 typical urban channel using a Doppler frequency of

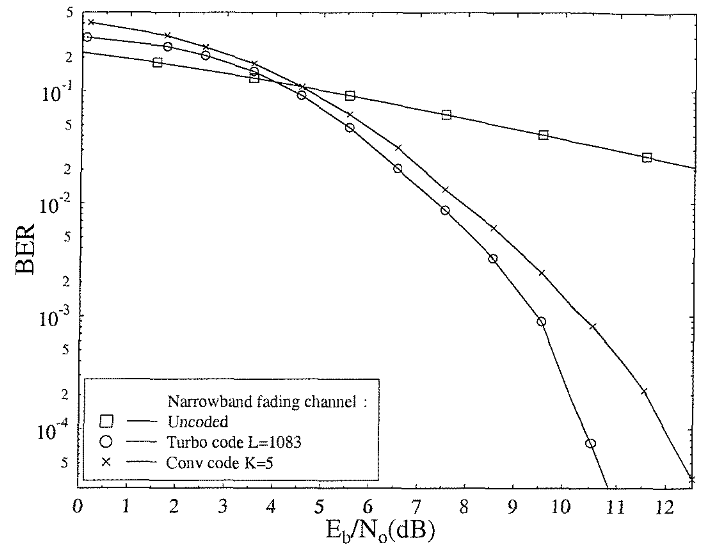


(a) Coded BER versus E_b/N_o over the narrowband Rayleigh fading channel.

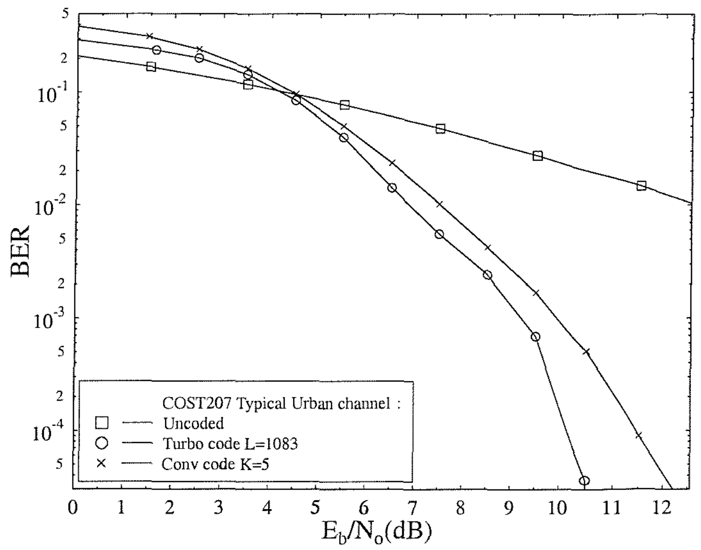


(b) Coded BER versus E_b/N_o over the COST207 typical urban channel.

Figure 2.13: Coded BER performance of the 11x17 turbo-interleaved turbo code, which was approximately 0.5 dB to 0.7 dB better in E_b/N_o terms at BER $= 10^{-3}$ than that of the constraint length $K = 5$ convolutional code over the narrowband Rayleigh fading channel and the COST207 typical urban channel for the full rate GSM speech channel using the parameters of Table 2.1.



(a) Coded BER versus E_b/N_o over the narrowband Rayleigh fading channel.



(b) Coded BER versus E_b/N_o over the COST207 typical urban channel.

Figure 2.14: The coded BER performance of turbo code with random interleaving possessing an interleaving depth of 1083 bits, which was approximately 1.5 dB better in E_b/N_o terms at BER $= 10^{-4}$ than that of the constraint length $K = 5$ convolutional code over the narrowband Rayleigh fading channel and the COST207 typical urban channel for full rate GSM data channel TCH/F9.6 [3]

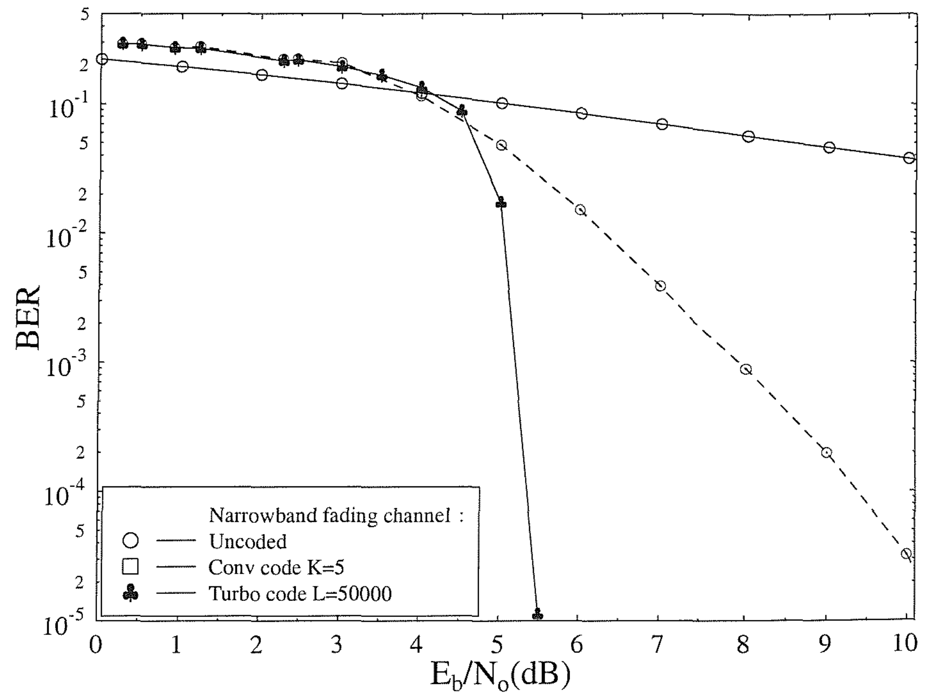


Figure 2.15: Coded BER performance of the $\frac{1}{2}$-rate, $K = 5$ turbo coding in conjunction with a random turbo interleaver of depth 50000 bits, which was approximately 3.7 dB better in E_b/N_o terms at BER $= 10^{-4}$ than convolutional codes over the so-called perfectly interleaved, uncorrelated Rayleigh fading narrowband channel.

f_d =41.7Hz, as seen in Table 2.1.

We observed in Figures 2.13(a) and 2.13(b) that the 11x17 turbo code required an SNR of 0.5 dB lower than the $K = 5$ convolutional code over the narrowband Rayleigh fading channel, and approximately 0.7 dB over the COST207 typical urban channel, at a BER of 10^{-3}. As expected, the turbo codec performed better, than the convolutional code, when the turbo coding frame length was longer, since in conjunction with longer coding frames the depth of the turbo interleaver can be increased. Therefore, there will be less correlation between the original and turbo-interleaved data sequences. In the less delay-sensitive GSM data channel — more specifically the TCH/F9.6 standard [3] — we can afford a delay of 19 114-bit data bursts. Therefore, we used a turbo encoder with an interleaving depth of $19 \cdot 114 = 2166$ bits. A random turbo interleaver was employed as it has been shown in [61] that for long turbo coding frames, the random interleaver has superior performance over block interleavers. However, block interleavers performed better for short frames [64]. The coded BER over the narrowband Rayleigh fading channel and the COST207 typical urban channel were illustrated in Figures 2.14(a) and 2.14(b), respectively. Here, we observed an E_b/N_o improvement of approximately 1.5 dB at BER $= 10^{-4}$ for both channels. By increasing the coding frame length further, to the order of 50000 bits, we obtained an improvement of approximately 3.7 dB at BER $= 10^{-4}$ over the standard convolutional codes, when transmitting over the so-called perfectly interleaved narrowband Rayleigh fading channel — which is modelled as a so-called complex Gaussian channel having a Doppler frequency or vehicular speed of infinity — as illustrated in Figure 2.15. This confirmed the superiority of turbo codes over convolutional codes, when the coding frame length was sufficiently long.

2.9 Summary

In Sections 2.2 and 2.5, the optimal Maximum *A Posteriori* (MAP) algorithm and the Log-MAP algorithm were described, respectively. Subsequently, the performance of the turbo-coded and convolutional-coded potential low-rate GSM speech systems was presented in Section 2.8. Specifically, the speech codec operated on 20 ms speech frames and output 38 bits for each of these 20 ms frames. It was observed in Figures 2.10 and 2.11, that turbo coding obtained a modest improvement of approximately 0.25 dB in comparison to the standard convolutional coding for transmissions over the narrowband Rayleigh fading channel and COST207 typical urban channel of Figure 2.8. This marginal improvement was attributed to the short interleaver depth used in the turbo encoder. Therefore, it was concluded that it was not justified to invest in the higher complexity and longer delay turbo codec, due to the above-mentioned performance limitation of short-latency turbo-coded systems. For longer turbo coding frames, simulations were carried out in conjunction with the full-rate GSM speech codec and for the GSM data channel. The full-rate GSM speech codec utilised a coding frame length of 378 bits and an 11x17-bit turbo interleaver. In the GSM data channel, a turbo encoder with a random turbo interleaver of depth 1083 bits was used. In Figures 2.13(a) and 2.13(b) we observed that the performance of the turbo codec employing an 11x17-bit turbo interleaver was 0.5 dB better, than that of the $K = 5$ convolutional code for the narrowband Rayleigh fading channel and approximately 0.7 dB better over the COST207 typical urban channel at a BER of 10^{-3}. For the GSM data channel, the required E_b/N_o of turbo codes was approximately 1.5 dB lower, than that of the standard convolutional code at BER$=10^{-4}$ for both of these channels. When the turbo coding frame length was increased to the order of 50000 bits, it was observed that a gain of approximately 3.7 dB was achieved in E_b/N_o terms at BER $= 10^{-4}$ as compared to the standard convolutional code over the so-called perfectly interleaved narrowband Rayleigh fading channel, as illustrated in Figure 2.15.

Chapter 3

Turbo Equalisation for Partial Response Systems

Turbo equalisation was first proposed by Douillard, Picart, Jézéquel, Didier, Berrou and Glavieux in 1995 in reference [10] for a serially concatenated convolutional coded BPSK system, as shown in Figure 3.1. In this contribution turbo equalisation employing the structure illustrated in Figure 3.2 was shown to mitigate the effects of ISI, when having perfect channel impulse response information. Instead of performing the equalisation and decoding independently, in order to overcome the channel's frequency selectivity, better performance can be obtained by the turbo equaliser, which considers the discrete channel's memory and performs both the equalisation and decoding iteratively.

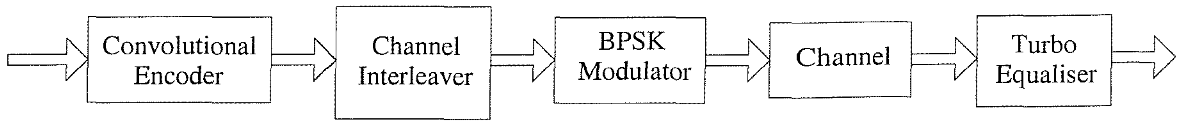


Figure 3.1: Serially concatenated convolutional coded BPSK system using the turbo equaliser, which performs the equalisation, demodulation and channel decoding iteratively.

The basic philosophy of the original turbo equalisation technique stems from the iterative turbo decoding algorithm consisting of two Soft-In/Soft-Out (SISO) decoders, a structure, which was proposed by Berrou *et al* [9, 44]. Before proceeding with our in-depth discussion, let us define below the terms *a priori*, *a posteriori* and extrinsic information, which we employ throughout this treatise.

A priori The *a priori* information associated with a bit v_m is the information known before equalisation or decoding commences, from a source other than the received sequence

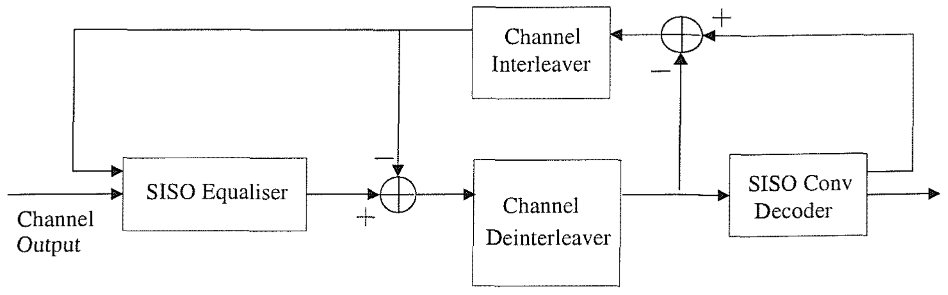


Figure 3.2: Structure of original turbo equaliser introduced by Douillard *et al* [10].

or the code constraints. It is also often referred to as intrinsic information, in order to contrast it with the extrinsic information, which is described next.

Extrinsic The extrinsic information associated with a bit v_m is the information provided by the equaliser or decoder based on the received sequence and on the *a priori* information of all bits with the exception of the received and *a priori* information explicitly related to that particular bit v_m. To elaborate a little further, extrinsic information related to a specific bit exists due to a number of factors. Firstly, when using an error correction encoder exhibiting memory, each input bit influences the encoder's output sequence over a long string of bits. Since a conventional convolutional code has an infinite duration impulse response [3], in theory each input bit influences the output bit sequence over an infinite number of coded bits. In practice, however, this interval is curtailed at about five times the code's constraint. In the context of RSC-coding based turbo codes, typically a long-memory turbo interleaver is employed, which substantially extends the number of coded bits, over which an input bit has an influence, since the turbo code's impulse response is typically prolonged. This justifies their high error correction power and the requirement of a high-depth turbo interleaver. Furthermore, since an input bit influences the encoded sequence over a high number of encoded bits, even when our confidence in a particular bit-decision is low, substantial amount of extrinsic information related to this was 'smeared' across a high number of encoded bits. With the aid of this extrinsic information the turbo decoder can iteratively enhance our confidence in the originally unreliable bit decision. We note however that the above arguments require that the intrinsic and extrinsic information related to a bit are treated separately by the decoder and hence remain uncorrelated, in order to be able to enhance each other's estimates in consecutive turbo decoding iterations.

As mentioned above — apart from the turbo encoder's memory — there are a number of other mechanisms generating extrinsic information due to their inherent memory, such as the channel's memory inducing dispersion over time. The deliberately introduced CISI of the GMSK modulators investigated constitutes another source of extrinsic information. These issues will be treated in more depth throughout our further discourse.

A posteriori The *a posteriori* information associated with a bit is the information that the SISO algorithm provides taking into account all available sources of information about the bit u_k.

As mentioned previously, the original turbo equaliser consists of a SISO equaliser and a SISO decoder. The SISO equaliser in Figure 3.2 generates the *a posteriori* probability upon receiving the corrupted transmitted signal sequence and the *a priori* probability provided by the SISO decoder. However, at the initial iteration stages — *i.e.* at the first turbo equalisation iteration — no *a priori* information is supplied by the channel decoder. Therefore, the *a priori* probability is set to $\frac{1}{2}$, since the transmitted bits are assumed to be equiprobable. Before passing the *a posteriori* information generated by the SISO equaliser to the SISO decoder of Figure 3.2, the contribution of the decoder — in the form of the *a priori* information — accruing from the previous iteration must be removed, in order to yield the combined channel and extrinsic information. This also minimises the correlation between the *a priori* information supplied by the decoder and the *a posteriori* information generated by the equaliser. The term 'combined channel and extrinsic information' indicates that they are inherently linked — in fact they are typically induced by mechanisms, which exhibit memory — and hence they cannot be separated. The removal of the *a priori* information is necessary, in order to prevent the decoder from receiving its own information, which would result in the so-called 'positive feedback' phenomenon, overwhelming the decoder's current reliability-estimation of the coded bits, *i.e.* the extrinsic information.

The combined channel and extrinsic information is channel-deinterleaved and directed to the SISO decoder, as depicted in Figure 3.2. Subsequently, the SISO decoder computes the *a posteriori* probability of the coded bits. Note that the latter steps are different from those in turbo decoding, which only produces the *a posteriori* probability of the source bits rather than those of all channel coded bits. The combined channel and extrinsic information is then removed from the *a posteriori* information provided by the decoder in Figure 3.2 before channel interleaving, in order to yield the extrinsic information. This is to prevent the channel equaliser from receiving information based on its own decisions, which was generated in the previous turbo equalisation iteration. The extrinsic information computed is then employed as the *a priori* input information of the equaliser in the next channel equalisation process. This constitutes the first turbo equalisation iteration. The iterative process is repeated, until the required termination criteria are met [65]. At this stage, the *a posteriori* information of the source bits, which has been generated by the decoder is utilised to estimate the transmitted bits. A more in-depth treatment of the turbo equalisation principles will be presented in Section 3.2.

3.1 Motivation

Following the pioneering turbo equalisation research of Douillard *et al* [10], further work was conducted by Gertsman and Lodge [66], who demonstrated that the iterative process of turbo equalisation can compensate for the performance degradations due to imperfect channel impulse response estimation. In the context of non-coherent detection, Marsland *et al* demonstrated in reference [67] that turbo equalisation offered better performance, than Dai's and Shwedyk's non-coherent, hard-decision based receiver using a bank of Kalman filters [68]. Turbo codes have also been used in conjunction with turbo equalisers by Raphaeli and Zarai [21], giving increased improvement due to the performance gain of turbo coding, as well as due to the ISI mitigation achieved by employing turbo equalisation. Bauch and Franz [15] as well as Jordan and Kammeyer [69] then demonstrated how the turbo equaliser can be applied in the context of GSM — employing GMSK modulation — although having to make modifications due to the interburst interleaving scheme used in GSM. Turbo equalisation was also utilised, in order to improve the performance of a convolutional-coded GMSK system employing a differential phase detector and decoder [70]. Recent work by Narayanan and Stüber [71] demonstrated the advantage of employing turbo equalisation in the context of coded systems invoking recursive modulators, such as Differential Phase Shift Keying (DPSK). Narayanan and Stuber emphasised the importance of a recursive modulator and showed that high iteration gains could be achieved, even when there was no ISI in the channel, *i.e.* for transmission over the non-dispersive Gaussian channel. The above-mentioned advantages of turbo equalisation as well as the importance of a recursive modulator motivated our research on turbo equalisation of coded-GMSK systems, since GMSK is also recursive in its nature. The recursive nature of GMSK modulation will be made explicit during our further discourse in Chapter 4.

The outline of this chapter is as follows. In Section 3.2, we will describe the functionality of the turbo equaliser in the context of convolutional-coded partial response GMSK systems and portray how the original concept of turbo equalisation is extended to multiple encoder assisted systems, such as turbo-coded schemes. Subsequently, Sections 3.3 and 3.4 describe how the Log-MAP algorithm is employed in the context of channel equalisation and channel decoding, respectively. The basic principles of turbo equalisation are then highlighted with the aid of a simple convolutional-coded BPSK system — employing turbo equalisation as an example — in Section 3.5. In Section 3.6 the turbo equalisation operations are summarised. Subsequently, Section 3.7 presents the turbo equalisation performance of the convolutional-coded GMSK system, that of the convolutional-coding based turbo-coded GMSK scheme and the Bose-Chaudhuri-Hocquengham (BCH) [23] coding based turbo-coded GMSK system, followed by a discussion in Section 3.8.

3.2 Principle of Turbo Equalisation using Single/Multiple Decoder(s)

With reference to Figure 3.3, in this section we describe the turbo equalisation principle for a baseband receiver consisting of an equaliser and N_d component decoders. This is an extension of the original turbo equalisation scheme [10], illustrated in Figure 3.2. For simplicity, we have omitted the channel interleaver π_c and turbo interleavers π_t and have only marked their positions. The superscript '-1' is used to represent a deinterleaver. Typically, for turbo codes there are $N_d = 2$ component decoders, whereas for non-iterative convolutional decoding we have $N_d = 1$. A vital rule for such turbo equalisers is that the input information to a particular block in the current iteration must not include the information contributed by that particular block from the previous iteration, since then the information used in consecutive iterations would be dependent on each other [66]. We will elaborate on this issue later on in more depth.

The equaliser and decoder in Figure 3.3 employ a Soft-In/Soft-Out (SISO) algorithm, such as the optimal Maximum *A Posteriori* (MAP) algorithm [17], the Log-MAP algorithm [18] or the Soft Output Viterbi Algorithm (SOVA) [54, 55, 56], which yields the *a posteriori* information. As defined previously, the *a posteriori* information concerning a bit is the information that the SISO block generates taking into account all available sources of information about the bit. When using the MAP algorithm or the Log-MAP algorithm of Chapter 2, we express the *a posteriori* information in terms of its so-called Log Likelihood Ratio (LLR) [45]. Again, the LLR L^{v_m} of a bit v_m, is defined as the natural logarithm of the ratio of the probabilities of the bit taking its two possible values of $+1$ and -1:

$$L^{v_m} \triangleq \ln \frac{P(v_m = +1)}{P(v_m = -1)}. \tag{3.1}$$

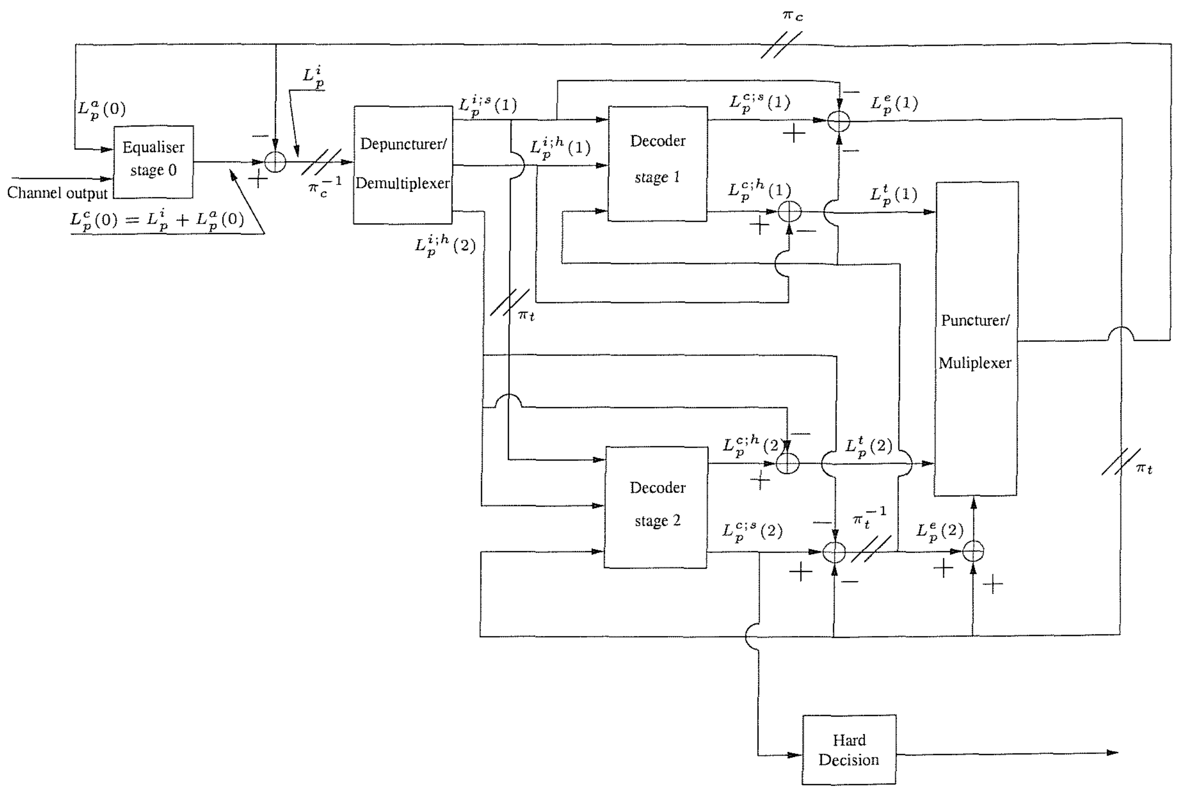


Figure 3.3: Structure of the turbo equaliser using $N_d = 2$ component decoders. For conceptual simplicity, we have omitted the interleavers and only marked the interleaver positions, where π_c and π_t represent the channel interleaver and turbo interleaver, respectively. The superscript '-1' is used to denote a deinterleaver.

For clarity, we have employed the approach used by Gertsman and Lodge [66] and expressed the LLR of the equaliser and decoder using vector notations. The associated superscript denotes the nature of the LLR, namely

L^c: composite *a posteriori* LLR consisting of source or information and parity bit *a posteriori* information.

$L^{c;s}$: *a posteriori* information of the source bit.

$L^{c;h}$: *a posteriori* information of the parity bit.

L^i: combined channel and the so-called extrinsic information, defined previously, for the source and parity bits.

$L^{i;s}$: combined channel and extrinsic information for the source bits.

$L^{i;h}$: combined channel and extrinsic information for the parity bits.

L^e: extrinsic information corresponding to the source bit.

L^t: extrinsic information of the parity bit.

$L^{a;s}$: *a priori* information of the source bit.

$L^{a;h}$: *a priori* information of the parity bit.

Furthermore, the subscript notation is used to represent the iteration index, while the argument within the brackets ( ) is the index of the receiver stage.

At the equaliser — which is denoted by stage 0 in Figure 3.3 — the composite output *a posteriori* LLR $L_p^c(0)$ at iteration p is generated by the sum of the LLR of the *a priori* information $L_p^a(0)$, and the combined LLR of the channel and extrinsic information L_p^i, giving:

$$L_p^c(0) = L_p^i + L_p^a(0). \tag{3.2}$$

As noted before in Chapter 2, at the commencement of the iterative detection there is no *a priori* information about the bit to be decoded and hence $L_p^a(0) = 0$ is used, indicating an equal probability of a binary one and zero. During the forthcoming iterations, however, our estimation concerning this bit can be fed back to the input of the equaliser in Figure 3.3, in order to aid its further iterations. We also have to augment the concept of *a priori* and extrinsic information. Recalling that the former is the information related to a specific bit, while the latter — as suggested by the terminology — indicates the information indirectly related to a bit, for example gained from the parity bits or redundant bits generated by the turbo encoder of Figure 2.1. We are unable to separate the channel's output information and extrinsic information at the output of the equaliser, which is denoted as L_p^i, since the channel impulse response can be viewed as that of a non-systematic code [15] 'smearing' the effects of the channel's input bits over time due to the associated convolution operation. Considering stage 0 of Figure 3.4, which is a detailed illustration of the equaliser, with

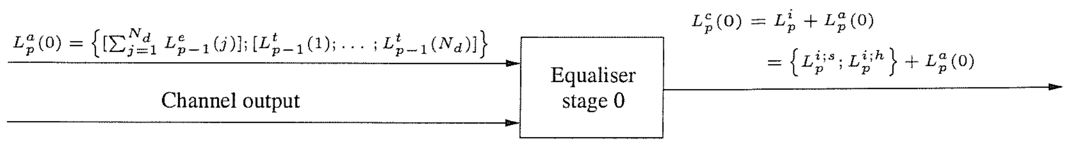

$L_p^a(0) = \left\{[\sum_{j=1}^{N_d} L_{p-1}^e(j)]; [L_{p-1}^t(1); \dots ; L_{p-1}^t(N_d)]\right\}$
Channel output
Equaliser stage 0
$L_p^c(0) = L_p^i + L_p^a(0)$
$= \left\{L_p^{i;s}; L_p^{i;h}\right\} + L_p^a(0)$

Figure 3.4: SISO equaliser illustration with emphasis on the input and output information at the pth iteration.

emphasis on the input and output information. It is noted that the composite *a posteriori* information $L_p^c(0)$, the *a priori* information $L_p^a(0)$ of the equaliser as well as the combined channel and extrinsic LLR L_p^i reflects the reliability of not only the source bits, but also that of the parity bits. The term L_p^i can be written as a set of vectors, given as:

$$L_p^i \in \left\{ \underbrace{L_p^{i;s}}_{\text{source bit}} \; ; \; \underbrace{L_p^{i;h}}_{\text{parity bits}} \right\}. \tag{3.3}$$

As for the stage 0 *a priori* information $L_p^a(0)$, this vector is obtained from the other N_d decoder stages — namely stages 1 and 2 in Figure 3.3 — and contains the *a priori* information of the encoded bits. Therefore, like vector L_p^i, $L_p^a(0)$ is also a set of vectors consisting of the *a priori* information for the source and parity bits, hence giving:

$$\begin{aligned} L_p^a(0) &\in \left\{ \underbrace{L_p^{a;s}(0)}_{\text{source bit}} ; \underbrace{L_p^{a;h}(0)}_{\text{parity bits}} \right\} \\ &\in \left\{ \underbrace{\sum_{j=1}^{N_d} L_{p-1}^e(j)}_{\text{source bit}} ; \underbrace{L_{p-1}^t(1); \ldots ; L_{p-1}^t(N_d)}_{\text{parity bits}} \right\} \end{aligned} \tag{3.4}$$

where $L_{p-1}^e(j)$ and $L_{p-1}^t(j)$ are the source and parity extrinsic LLRs of the jth decoder stage from the previous iteration $p-1$ while N_d is the number of component decoders in Figure 3.3, as before. Using Equations 3.2, 3.3 and 3.4, the composite *a posteriori* LLRs of the encoded bits $L_p^c(0)$ at the output of the equaliser in Figure 3.3 can be expressed as:

$$\begin{aligned} L_p^c(0) &\in \left\{ \underbrace{L_p^{c;s}(0)}_{\text{source bit}} ; \underbrace{L_p^{c;h}(0)}_{\text{parity bits}} \right\} \\ &\in \left\{ \underbrace{L_p^{i;s} + \sum_{j=1}^{N_d} L_{p-1}^e(j)}_{\text{source bit}} ; \underbrace{[L_p^{i;h} + L_{p-1}^t(1); \ldots ; L_p^{i;h} + L_{p-1}^t(N_d)]}_{\text{parity bits}} \right\} \end{aligned} \tag{3.5}$$

The stage b decoder in Figure 3.3 is illustrated here again in more detail in Figure 3.5, with emphasis on its input and output information. It receives the sum of extrinsic information from all the other decoders excluding the stage b decoder — namely $\sum_{j=1}^{b-1} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j)$ — as the *a priori* information and also the combined channel and extrinsic information of the source and parity LLR values, namely $L_p^{i;s}$ and $L_p^{i;h}$,

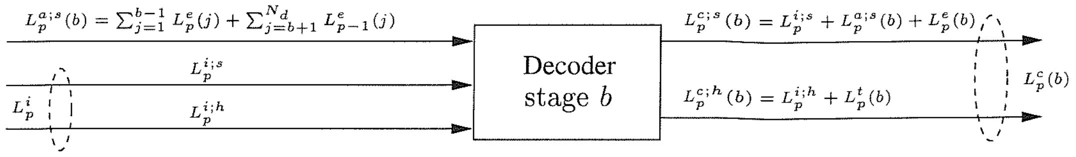

$L_p^{a;s}(b) = \sum_{j=1}^{b-1} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j)$
$L_p^{i;s}$
L_p^i
$L_p^{i;h}$
Decoder stage b
$L_p^{c;s}(b) = L_p^{i;s} + L_p^{a;s}(b) + L_p^e(b)$
$L_p^{c;h}(b) = L_p^{i;h} + L_p^t(b)$
$L_p^c(b)$

Figure 3.5: Schematic of the SISO decoder with emphasis on the input and output information at the pth iteration.

respectively. The augmented source *a priori* information of the stage b decoder at the pth iteration can be expressed as:

$$L_p^{a;s}(b) = \sum_{j=1}^{b-1} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j). \tag{3.6}$$

From Equation 3.6 we observe that the *a priori* information $L_p^{a;s}(b)$ of the source bits consists of the extrinsic information from the decoders of the **previous stages** — namely for $j < b$ in the **current iteration** p — and from the decoders of the **later stages** — namely for $j > b$ from the **previous iteration** $p-1$ — but does not include any extrinsic information from stage b. Note that unlike the equaliser, which receives the *a priori* information of the source and parity bits, the N_d decoders accept *a priori* LLRs of source bits only. This is because the source bit is the only common information to all decoders, whether in the interleaved order or the original sequence, whereas the parity bits are exclusive to a particular decoder, hence they are unable to contribute in the other decoders' decoding operation.

At the stage b systematic decoder — for example at the outputs of the stages 1 and 2 in Figure 3.3 and in greater detail in Figure 3.5 — the composite *a posteriori* LLR $L_p^c(b)$ consists of two LLR vectors, namely the source bit *a posteriori* information $L_p^{c;s}(b)$ and the parity bit *a posteriori* information $L_p^{c;h}(b)$:

$$L_p^c(b) \in \left\{L_p^{c;s}(b); L_p^{c;h}(b)\right\}. \tag{3.7}$$

The composite source *a posteriori* LLR $L_p^{c;s}(b)$ — which is the first component in Equation 3.7 — can be expressed as the sum of the augmented *a priori* information $L_p^{a;s}(b)$, the extrinsic information $L_p^e(b)$ produced by the decoder stage and the combined channel and extrinsic LLR $L_p^{i;s}$ of the source bit from the equaliser:

$$L_p^{c;s}(b) = L_p^{a;s}(b) + L_p^e(b) + L_p^{i;s}, \tag{3.8}$$

which can be rewritten by using Equations 3.6 and 3.8 as:

$$\begin{aligned} L_p^{c;s}(b) &= L_p^{a;s}(b) + L_p^e(b) + L_p^{i;s} \\ &= \sum_{j=1}^{b-1} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j) + L_p^e(b) + L_p^{i;s} \\ &= \sum_{j=1}^{b} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j) + L_p^{i;s}, \end{aligned} \tag{3.9}$$

while the parity *a posteriori* LLR $L_p^{c;h}(b)$ of Equation 3.7 can be expressed as:

$$L_p^{c;h}(b) = L_p^{i;h} + L_p^t(b). \tag{3.10}$$

The term $L_p^{i;h}(b)$ is the combined channel and extrinsic information of the parity bit and $L_p^t(b)$ is the extrinsic LLR corresponding to the parity bit.

In general, only two component encoders are used in practical turbo encoders. Therefore, by substituting $N_d = 2$ into Equations 3.5, 3.9 and 3.10, we can determine the *a posteriori* LLRs of the equaliser and decoders, whereas Equations 3.4 and 3.6 can be used to determine the corresponding *a priori* inputs to the equaliser and decoders. Again, Figure 3.3 illustrates the structure of the turbo equaliser, which employs the principles discussed above.

In the following sections we will concentrate on the main components of Figure 3.3, namely on the equaliser and the decoder(s).

3.3 Soft-In/Soft-Out Equaliser for Turbo Equalisation

In Subsections 2.2 and 2.5 we discussed, how the MAP algorithm and the Log-MAP algorithm can be used as a soft output GMSK equaliser in conjunction with turbo decoding. However, in these subsections, the iterations were only performed in the turbo decoder. Therefore, the equaliser did not receive any *a priori* values from an independent source. However, when implementing turbo equalisation, the SISO equaliser will receive not only channel outputs, but also *a priori* information from the SISO decoder. We can still employ the MAP algorithm or the Log-MAP algorithm described in Subsections 2.2 and 2.5 for the soft output GMSK equaliser, however we must now consider additionally the *a priori* information from the SISO decoder at the turbo equaliser's input. Explicitly, the term $P(u_m)$ — which represents the *a priori* probability of the code bits — required when evaluating $\gamma_m(\acute{\kappa}, \kappa)$ and $\Gamma_m(\acute{\kappa}, \kappa)$ in Equations 2.24 and 2.38 is no longer $\frac{1}{2}$, since the probability of the bit u_m being $+1$ or -1 is no longer equal. Instead, the probability $P(u_m)$ is augmented with the aid of the *a priori* information from the decoder(s), as seen in Equation 3.4.

Typically, the Viterbi algorithm [37, 72] — which is a Maximum Likelihood Sequence Estimation algorithm — is used to decode convolutional codes. However, in turbo equalisation we need soft outputs, which represent the LLRs of the individual coded bits. In the following subsection, we will show how the LLRs of these coded bits is determined.

3.4 Soft-In/Soft-Out Decoder for Turbo Equalisation

In turbo equalisation the SISO decoder block accepts soft inputs from the SISO equaliser and gives the LLR of not only the source bits, but also that of the coded bits, in order to implement turbo equalisation. Therefore, the decoder must employ SISO algorithms, such as the MAP algorithm, Log-Map algorithm or the SOVA. Note that since the Log-MAP algorithm yields identical performance to the optimal MAP algorithm, while incurring lower complexity, we will use this algorithm in our discussions.

Through the Log-MAP algorithm the LLRs of the source and those of the coded bits — which are output by the turbo decoder and turbo equaliser, respectively — can be computed in two key steps. Let us define the notations used before highlighting these steps. In the decoder the trellis transitions are denoted by the index d, whereas the equaliser transition intervals are represented by m. The notation v_d represents the source bit, while $c_{l,d}$ represents the lth coded bit at the dth decoder trellis interval. Relating the equaliser transition intervals to the rate $R = \frac{k}{n}$ decoder trellis intervals, we note that the lapse of n number of equaliser intervals corresponds to a single decoder trellis interval. For example, one decoder transition interval in a rate $R = \frac{1}{2}$ decoder is equivalent to $n = 2$ equaliser intervals. Therefore, the bits u_m and u_{m+1} received at the equaliser correspond to the code bits $c_{1,d}$ and $c_{2,d}$ at decoder interval d. Having defined the notation used, let us now return to the discussion of the two key steps employed to determine the LLR values of the source and coded bits.

Firstly, the forward and backward recursion values, $A_m(\kappa)$ and $B_m(\kappa)$, respectively, as well as the transition metric $\Gamma_m(\acute{\kappa}, \kappa)$ — which are the logarithmic domain counterparts of $\Omega_m(\kappa)$, $\beta_m(\kappa)$ and $\gamma_m(\acute{\kappa}, \kappa)$ in the MAP algorithm — are computed using Equations 2.30, 2.34 and 2.38. However, for the decoder the trellis transition index m is replaced by d. Secondly, we have to modify Equation 2.40 — which was discussed previously in Chapter 2 in the context of channel equalisation for turbo decoding — in order for the

SISO decoder to give the LLR of not only the source bits v_d, but also that of the coded bits, namely $c_{l,d}$. Explicitly, for the rate $\frac{k}{n}$, convolutional encoder there are n number of LLR values generated at each trellis interval, whereas the binary equaliser produces only a single LLR value at each interval. Recalling Equation 2.40, which is repeated here for convenience:

$$L(u_m|r) = \ln\left(\frac{\sum_{(\acute{\kappa},\kappa)\Rightarrow u_m=+1} \exp\left(A_{m-1}(\acute{\kappa}) + \Gamma_m(\acute{\kappa},\kappa) + B_m(\kappa)\right)}{\sum_{(\acute{\kappa},\kappa)\Rightarrow u_m=-1} \exp\left(A_{m-1}(\acute{\kappa}) + \Gamma_m(\acute{\kappa},\kappa) + B_m(\kappa)\right)}\right),$$

it is noted that at the mth equaliser trellis interval, the numerator is the sum of the transition probabilities associated with the paths caused by the coded bit $u_m = +1$, whereas the denominator is the sum of the probabilities corresponding to the transitions generated by $u_m = -1$. As explained in Subsection 2.5 regarding the Log-Map algorithm, these transition probabilities can be written as the a sum of exponential terms, whose arguments are the sum of $A_{m-1}(\acute{\kappa})$, $B_m(\kappa)$, and $\Gamma_m(\acute{\kappa},\kappa)$. Therefore, at each equaliser trellis interval m, the LLR value obtained is for the coded bit u_m. However, for the $R = \frac{k}{n}$ decoder, we require the LLR values for the n number of coded bits. Therefore, in order to determine the LLR for each coded bit $c_{l,d}$ for $l = 1 \ldots n$, we must first consider the branches and the corresponding probability value — which is the summation of the term $\exp((A_{d-1}(\acute{\kappa}) + \Gamma_d(\acute{\kappa},\kappa) + B_d(\kappa)))$ — giving the coded bit $c_{l,d} = +1$ and subsequently the branches resulting in the coded bit of $c_{l,d} = -1$. Consequently, Equation 2.40 is modified to:

$$\begin{aligned} L(c_{l,d}|L_p^i) &= \ln\left(\frac{\sum_{(\acute{\kappa},\kappa)\Rightarrow c_{l,d}=+1} \exp\left(A_{d-1}(\acute{\kappa}) + \Gamma_d(\acute{\kappa},\kappa) + B_d(\kappa)\right)}{\sum_{(\acute{\kappa},\kappa)\Rightarrow c_{l,d}=-1} \exp\left(A_{d-1}(\acute{\kappa}) + \Gamma_d(\acute{\kappa},\kappa) + B_d(\kappa)\right)}\right) \\ &= \ln\left(\sum_{(\acute{\kappa},\kappa)\Rightarrow c_{l,d}=+1} \exp\left(A_{d-1}(\acute{\kappa}) + \Gamma_d(\acute{\kappa},\kappa) + B_d(\kappa)\right)\right) \\ &\quad - \ln\left(\sum_{(\acute{\kappa},\kappa)\Rightarrow c_{l,d}=-1} \exp\left(A_{d-1}(\acute{\kappa}) + \Gamma_d(\acute{\kappa},\kappa) + B_d(\kappa)\right)\right), \end{aligned} \tag{3.11}$$

in order to calculate the LLR of the coded bits $c_{l,d}$, where $L(c_{l,d}|L_p^i)$ is the LLR of the lth coded bit at trellis interval d, given that the combined channel and extrinsic LLR L_p^i was received by the decoder.

Let us now consider the example of a SISO convolutional decoder trellis for a rate $R = \frac{1}{2}$, constraint length $K = 3$ RSC encoder, as illustrated in Figure 3.6. In this example, a RSC code has been chosen instead of a non-systematic convolutional code, since we will be implementing turbo coding, which has been shown to perform better with the aid of

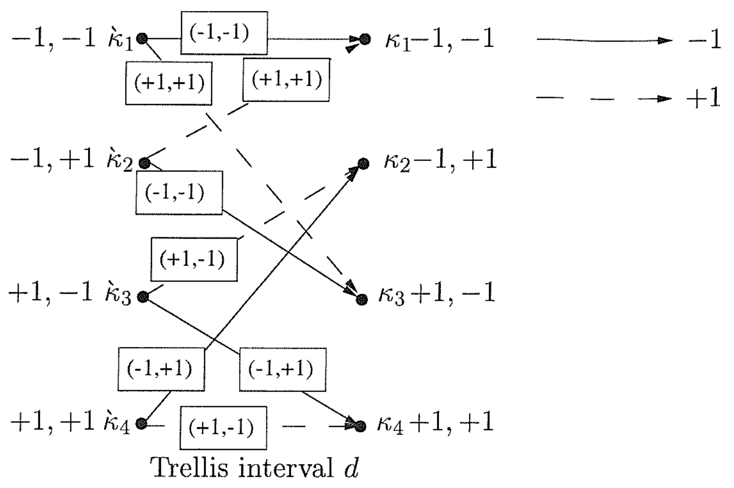


Figure 3.6: Trellis of a $K = 3$ RSC convolutional decoder at the dth decoder trellis interval, where the octal representation of the generator polynomial G_0 and G_1 is 7 and 5, respectively.

RSC component codes [9, 44, 20]. Since the code rate is $R = \frac{1}{2}$, the output codeword consists of $k = 2$ coded bits, $c_{1,d}$ and $c_{2,d}$, at each decoder trellis interval d. The output codeword for each transition is depicted as $(c_{1,d}, c_{2,d})$ in Figure 3.6 and is governed by the generator polynomials G_0 and G_1, which can be represented in the octal form as 7 and 5, respectively. In order to determine the LLR for $c_{1,d}$, we have to consider all paths which give the coded bit $c_{1,d} = +1$. From Figure 3.6 we observe that there are 4 branches, which fulfil this requirement, namely branches from state $\grave{\kappa}_1$ to κ_3, $\grave{\kappa}_2$ to κ_1, $\grave{\kappa}_3$ to κ_2 and $\grave{\kappa}_4$ to κ_4. Observing Equations 2.3 and 2.4 and by substituting Equations 2.27, 2.28 and 2.29 into Equations 2.4, we note that the probability that a transition from state $\grave{\kappa}$ to κ occurs and that the combined channel and extrinsic LLR L_p^i was obtained is given by:

$$P(\grave{\kappa} \wedge \kappa \wedge L_p^i) = \exp\left(A_{d-1}(\grave{\kappa}) + \Gamma_d(\grave{\kappa}, \kappa) + B_d(\kappa)\right). \tag{3.12}$$

Therefore, the individual branch probabilities associated with the coded bit of $c_{1,d} = +1$ will be:

$$\begin{aligned}
P(\grave{\kappa}_1 \wedge \kappa_3 \wedge L_p^i) &= \exp\left(A_{d-1}(\grave{\kappa}_1) + \Gamma_d(\grave{\kappa}_1, \kappa_3) + B_d(\kappa_3)\right) &&= \exp(x(\grave{\kappa}_1, \kappa_3)) \\
P(\grave{\kappa}_2 \wedge \kappa_1 \wedge L_p^i) &= \exp\left(A_{d-1}(\grave{\kappa}_2) + \Gamma_d(\grave{\kappa}_2, \kappa_1) + B_d(\kappa_1)\right) &&= \exp(x(\grave{\kappa}_2, \kappa_1)) \\
P(\grave{\kappa}_3 \wedge \kappa_2 \wedge L_p^i) &= \exp\left(A_{d-1}(\grave{\kappa}_3) + \Gamma_d(\grave{\kappa}_3, \kappa_2) + B_d(\kappa_2)\right) &&= \exp(x(\grave{\kappa}_3, \kappa_2)) \\
P(\grave{\kappa}_4 \wedge \kappa_4 \wedge L_p^i) &= \exp\left(A_{d-1}(\grave{\kappa}_4) + \Gamma_d(\grave{\kappa}_4, \kappa_4) + B_d(\kappa_4)\right) &&= \exp(x(\grave{\kappa}_4, \kappa_4)),
\end{aligned} \tag{3.13}$$

where $\exp(x(\grave{\kappa}_i, \kappa_j)$ represents the sum $A_{d-1}(\grave{\kappa}_i) + \Gamma_d(\grave{\kappa}_i, \kappa_j) + B_d(\kappa_j)$. As before, $A_{d-1}(\grave{\kappa})$ and $B_d(\kappa)$ are evaluated, once $\Gamma_d(\grave{\kappa}, \kappa)$ has been computed. However, the expression of

$\Gamma_d(\acute{\kappa}, \kappa)$ for the decoder can be further simplified from Equation 2.38, which is repeated here for convenience:

$$\Gamma_m(\acute{\kappa}, \kappa) = \ln\left(\frac{1}{2\sqrt{\pi N_o}}\right) - \frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m} - s_{i,k})^2 + \ln(P(u_m)). \tag{3.14}$$

Observe that $\Gamma_m(\acute{\kappa}, \kappa)$ is related to the squared Euclidean distance between the sampled received signal $r_{k,m}$ and the expected signal $s_{i,k}$, which is subsequently normalised by N_o. The above expression can be expanded to:

$$\Gamma_m(\acute{\kappa}, \kappa) = \ln\left(\frac{1}{2\sqrt{\pi N_o}}\right) - \frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m}^2 - 2 \cdot r_{k,m} \cdot s_{i,k} + s_{i,k}^2) + \ln(P(u_m)). \tag{3.15}$$

The expected signals at the decoder are the coded bits of values $+1$ as well as -1. Therefore, the square of the expected signal $s_{i,k}$ is always unity and hence it can be ignored. Furthermore, since the same value of $r_{m,k}$ is considered for each trellis transition, it too can be neglected during the associated pattern-matching process, leaving the decoder metric $\Gamma_d(\acute{\kappa}, \kappa)$ to be dependent on the cross-correlative term $(2 \cdot r_{k,m} \cdot s_{i,k})/N_o$. At the input of the decoder, the combined channel and extrinsic information L_p^i — containing the contribution of N_o [15] — is received, while the locally generated expected signals are associated with the trellis transitions due to the coded bits $c_{l,d}$, for $l = 1 \ldots n$, where n is the number of codeword bits. Hence, the decoder metric $\Gamma_d(\acute{\kappa}, \kappa)$ at decoder trellis interval d can be expressed as [15]:

$$\Gamma_d(\acute{\kappa}, \kappa) = \sum_{l=1}^{n=2}\left(\frac{1}{2}L_p^i \cdot c_{l,d}\right) + \ln(P(v_d)), \tag{3.16}$$

where $P(v_d)$ is the *a priori* probability of the source bit v_d. After evaluating $\Gamma_d(\acute{\kappa}, \kappa)$, the terms $A_{d-1}(\acute{\kappa})$ and $B_d(\kappa)$ can be determined using Equations 2.30 and 2.34, respectively. Therefore, the probability that $c_{1,d} = +1$ was transmitted at instant d and that L_p^i was received is determined by the sum of the $c_{1,d} = +1$-related individual branch probabilities of $P(\acute{\kappa}_1 \wedge \kappa_3 \wedge L_p^i)$, $P(\acute{\kappa}_2 \wedge \kappa_1 \wedge L_p^i)$, $P(\acute{\kappa}_3 \wedge \kappa_2 \wedge L_p^i)$ and $P(\acute{\kappa}_4 \wedge \kappa_4 \wedge L_p^i)$.

Similarly, we must consider all paths in Figure 3.6, which give $c_{1,d} = -1$, namely the branches from state $\acute{\kappa}_1$ to κ_1, $\acute{\kappa}_2$ to κ_3, $\acute{\kappa}_3$ to κ_4 and $\acute{\kappa}_4$ to κ_2. Having observed this, with the aid of Figure 3.6 we proceed to calculate the individual branch probabilities associated with the coded bit of $c_{1,d} = -1$ as follows:

$$\begin{aligned}
P(\acute{\kappa}_1 \wedge \kappa_1 \wedge L_p^i) &= \exp\left(A_{d-1}(\acute{\kappa}_1) + \Gamma_d(\acute{\kappa}_1, \kappa_1) + B_d(\kappa_1)\right) &&= \exp(x(\acute{\kappa}_1, \kappa_1)) \\
P(\acute{\kappa}_2 \wedge \kappa_3 \wedge L_p^i) &= \exp\left(A_{d-1}(\acute{\kappa}_2) + \Gamma_d(\acute{\kappa}_2, \kappa_3) + B_d(\kappa_3)\right) &&= \exp(x(\acute{\kappa}_2, \kappa_3)) \\
P(\acute{\kappa}_3 \wedge \kappa_4 \wedge L_p^i) &= \exp\left(A_{d-1}(\acute{\kappa}_3) + \Gamma_d(\acute{\kappa}_3, \kappa_4) + B_d(\kappa_4)\right) &&= \exp(x(\acute{\kappa}_3, \kappa_4)) \\
P(\acute{\kappa}_4 \wedge \kappa_2 \wedge L_p^i) &= \exp\left(A_{d-1}(\acute{\kappa}_4) + \Gamma_d(\acute{\kappa}_4, \kappa_2) + B_d(\kappa_2)\right) &&= \exp(x(\acute{\kappa}_4, \kappa_2)).
\end{aligned} \tag{3.17}$$

As before, we can evaluate the probability that $c_{1,d} = -1$ was transmitted at trellis interval d given that L_p^i was received by taking the sum of the individual branch probabilities of $P(\grave{\kappa}_1 \wedge \kappa_1 \wedge L_p^i)$, $P(\grave{\kappa}_2 \wedge \kappa_3 \wedge L_p^i)$, $P(\grave{\kappa}_3 \wedge \kappa_4 \wedge L_p^i)$ and $P(\grave{\kappa}_4 \wedge \kappa_2 \wedge L_p^i)$.

With the aid of Equation 3.11 and using Equations 3.12, 3.13 and 3.17, we can express the LLR for the first coded bit $c_{1,d}$ as:

$$\begin{aligned} L(c_{1,d}|L_p^i) &= \ln\left(\frac{\sum_{(\grave{\kappa},\kappa)\Rightarrow c_{1,d}=+1} P(\grave{\kappa}\wedge\kappa\wedge L_p^i)}{\sum_{(\grave{\kappa},\kappa)\Rightarrow c_{1,d}=-1} P(\grave{\kappa}\wedge\kappa\wedge L_p^i)}\right) \\ &= \ln\left(\frac{P(\grave{\kappa}_1\wedge\kappa_3\wedge L_p^i)+P(\grave{\kappa}_2\wedge\kappa_1\wedge L_p^i)+P(\grave{\kappa}_3\wedge\kappa_2\wedge L_p^i)+P(\grave{\kappa}_4\wedge\kappa_4\wedge L_p^i)}{P(\grave{\kappa}_1\wedge\kappa_1\wedge L_p^i)+P(\grave{\kappa}_2\wedge\kappa_3\wedge L_p^i)+P(\grave{\kappa}_3\wedge\kappa_4\wedge L_p^i)+P(\grave{\kappa}_4\wedge\kappa_2\wedge L_p^i)}\right) \\ &= \ln\Big(\exp(x(\grave{\kappa}_1,\kappa_3))+\exp(x(\grave{\kappa}_2,\kappa_1))+\exp(x(\grave{\kappa}_3,\kappa_2))+\exp(x(\grave{\kappa}_4,\kappa_4))\Big) \\ &\quad -\ln\Big(\exp(x(\grave{\kappa}_1,\kappa_1))+\exp(x(\grave{\kappa}_2,\kappa_3))+\exp(x(\grave{\kappa}_3,\kappa_4))+\exp(x(\grave{\kappa}_4,\kappa_2))\Big). \end{aligned} \tag{3.18}$$

Since the term $L(c_{1,d}|L_p^i)$ can be expressed as the natural logarithm of a sum of exponentials, the generalised Jacobian logarithmic relationship of Equation 2.41 can be employed, in order to calculate the LLR of the coded bit $c_{1,d}$ at trellis interval d.

By employing the same approach as above we can demonstrate that the LLR of the coded bit $c_{2,d}$ is given by:

$$\begin{aligned} L(c_{2,d}|L_p^i) &= \ln\left(\frac{\sum_{(\grave{\kappa},\kappa)\Rightarrow c_{2,d}=+1} P(\grave{\kappa}\wedge\kappa\wedge L_p^i)}{\sum_{(\grave{\kappa},\kappa)\Rightarrow c_{2,d}=-1} P(\grave{\kappa}\wedge\kappa\wedge L_p^i)}\right) \\ &= \ln\left(\frac{P(\grave{\kappa}_1\wedge\kappa_3\wedge L_p^i)+P(\grave{\kappa}_2\wedge\kappa_1\wedge L_p^i)+P(\grave{\kappa}_3\wedge\kappa_4\wedge L_p^i)+P(\grave{\kappa}_4\wedge\kappa_2\wedge L_p^i)}{P(\grave{\kappa}_1\wedge\kappa_1\wedge L_p^i)+P(\grave{\kappa}_2\wedge\kappa_3\wedge L_p^i)+P(\grave{\kappa}_3\wedge\kappa_2\wedge L_p^i)+P(\grave{\kappa}_4\wedge\kappa_4\wedge L_p^i)}\right) \\ &= \ln\Big(\exp(x(\grave{\kappa}_1,\kappa_3))+\exp(x(\grave{\kappa}_2,\kappa_1))+\exp(x(\grave{\kappa}_3,\kappa_4))+\exp(x(\grave{\kappa}_4,\kappa_2))\Big) \\ &\quad -\ln\Big(\exp(x(\grave{\kappa}_1,\kappa_1))+\exp(x(\grave{\kappa}_2,\kappa_3))+\exp(x(\grave{\kappa}_3,\kappa_2))+\exp(x(\grave{\kappa}_4,\kappa_4))\Big). \end{aligned} \tag{3.19}$$

The generalised Jacobian relationship in Equation 2.41 is then used to evaluate the sum of the exponential terms in Equations 3.18 and 3.19 in order to yield the LLR values $L(c_{2,d}|L_p^i)$ of the coded bit $c_{2,d}$ at trellis interval d.

3.5 Turbo Equalisation Example

In order to highlight the principle of turbo equalisation, let us now consider a simple convolutional-coded BPSK system transmitting over a three-path, symbol-spaced static

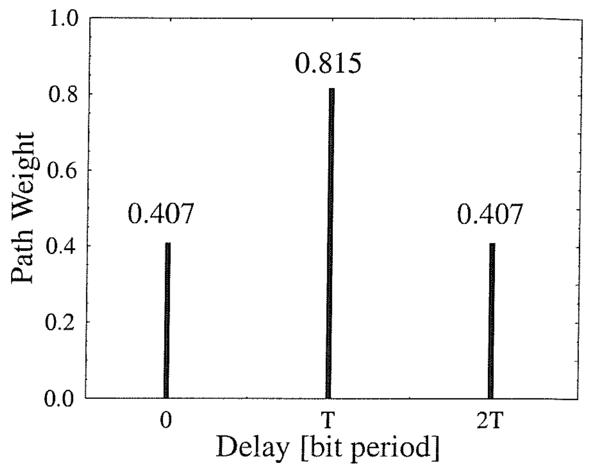


Figure 3.7: Three-path symbol-spaced static channel.

channel detailed in Figure 3.7 and employing a turbo equaliser at the receiver, as shown in Figure 3.8. A rate $R = 0.5$, constraint length $K = 3$ RSC encoder using octal generator polynomials of 7 and 5, respectively, was utilised. The corresponding decoder trellis structure and its legitimate transitions was illustrated in Figure 3.6. Note that this turbo equaliser schematic — consisting of a trellis-based channel equaliser and a channel decoder — was first introduced in Figure 3.2, where its basic operation was highlighted. Hence here we refrain from detailing its philosophy.

Since the maximum dispersion of the channel impulse response is $\tau_d = 2$ bit periods, the trellis-based BPSK equaliser requires $2^{\tau_d=2} = 4$ states, as shown in Figure 3.9. Each of these four states can be viewed as those of a shift-register possessing two memory elements and can therefore be represented by two binary bits. Consider state 1 in Figure 3.9, which corresponds to the binary bits 1 and 0 in the shift register, where bit 0 is the 'older' bit. For an input bit of 1, the bit 0 is shifted out while the input bit 1 is now stored. Therefore, the new state reached is state 3, since the bits in the shift register memory elements are 1 and 1. Applying the same reasoning, all the other legitimate state transitions can be determined, producing the trellis structure of Figure 3.9.

In this example, the transmission burst consists of 10 data bits and 3 tail bits, while the depth of the random channel interleaver is 10 bits. The following discussion will describe the operation of the turbo equaliser, commencing with generating the equaliser's

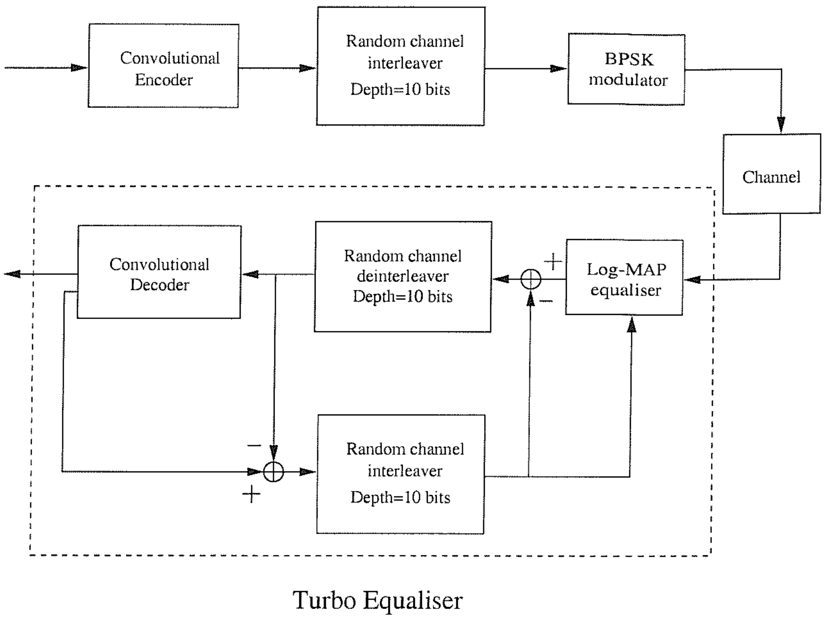

Convolutional Encoder
Random channel interleaver Depth=10 bits
BPSK modulator
Channel
Log-MAP equaliser
Random channel deinterleaver Depth=10 bits
Convolutional Decoder
Random channel interleaver Depth=10 bits
Turbo Equaliser

Figure 3.8: Schematic of the convolutional-coded BPSK system using the turbo equaliser scheme of Figure 3.2.

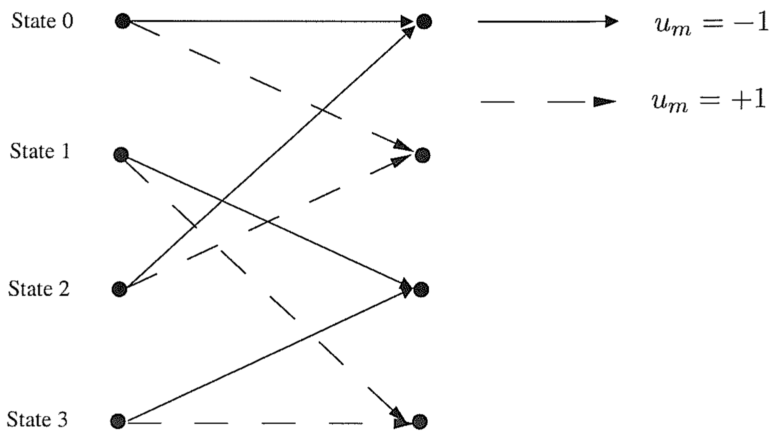


Figure 3.9: Equaliser trellis depicting the legitimate transitions.

a posteriori LLR, which is then processed to extract the combined channel and extrinsic information to be passed to the decoder. Subsequently, it is shown how this combined channel and extrinsic information is utilised, in order to generate the decoder's *a posteriori* LLR, which — as in the equaliser — is processed to obtain the extrinsic information, before passing it back to the equaliser. For simplicity, all floating point values in the following figures have been rounded to one decimal position. The trellis-based channel equaliser employs the Log-MAP algorithm described previously in Subsection 2.2. In order to evaluate the LLR value of a bit u_m, the values of $\Gamma_m(\acute{\kappa},\kappa)$, $A_{m-1}(\acute{\kappa})$ and $B_m(\kappa)$ — which were described in Subsection 2.5 — must be evaluated and used jointly. This will be elaborated on numerically in the following example.

In this example, the 5 source bits and the corresponding 10 convolutional encoded bits are specified in Table 3.1. Subsequently, these encoded bits are rearranged by a 2x5 block

Decoder trellis interval d	Source bits	Encoded bits	Interleaved encoded bits
1	+1	+1, +1	+1, −1
2	+1	+1, −1	+1, −1
3	−1	−1, −1	+1, +1
4	−1	−1, +1	−1, +1
5	+1	+1,−1	−1, −1

Table 3.1: The source bits transmitted and the corresponding encoded bits produced by a rate $R = \frac{1}{2}$ $K = 3$ recursive, systematic convolutional encoder using the octal generator polynomials of 7 and 5.

channel interleaver to give the new sequence detailed in Table 3.1. These interleaved bits are BPSK-modulated before being transmitted through the three-path static channel of Figure 3.7. At the receiver the additive white Gaussian thermal noise corrupts the received signal. The signal is subsequently sampled and stored in a buffer, until all 10 encoded bits and 3 tail bits arrive at the receiver. At this stage, we can begin to evaluate the transition metric $\Gamma_m(\acute{\kappa},\kappa)$ for all trellis intervals. Figure 3.10 illustrates the transitions in the first three trellis intervals. Specifically, starting from state 0 at trellis interval $m = 1$, where the initial value of $A_0(0)$ is 0.0, while the values of $\Gamma_1(0,0)$ and $\Gamma_1(0,1)$ are computed by using Equations 2.29 and 2.24, which are repeated here for convenience:

$$\Gamma_m(\acute{\kappa},\kappa) \triangleq \ln(\gamma_m(\acute{\kappa},\kappa)).$$

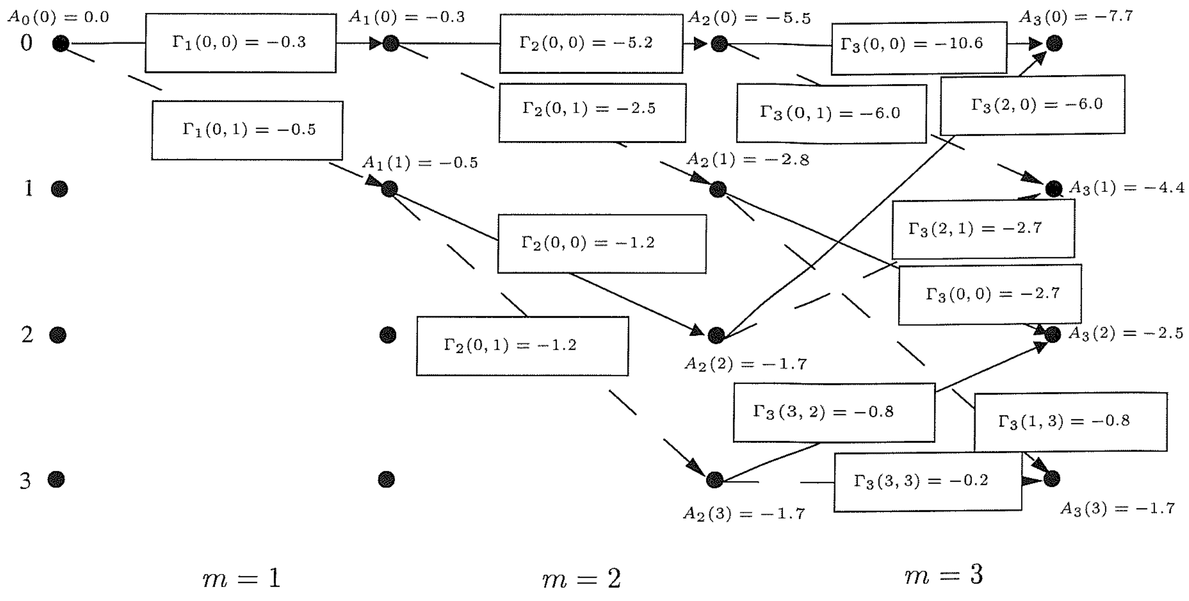


Figure 3.10: Evaluation of $A_m(\kappa)$ in the turbo equaliser of Figure 3.8 through forward recursion using a range of $\Gamma_m(\acute{\kappa}, \kappa)$ values, which were evaluated from Equations 2.29 and 2.24 by employing our simulation program over the channel of Figure 3.7.

and

$$\gamma_m(\acute{\kappa}, \kappa) = \frac{1}{\sqrt{\pi N_o}} \exp\big[-\frac{1}{N_o}\sum_{k=0}^{N_s-1}(r_{k,m} - s_{i,k})^2\big]\cdot P(u_m),$$

where N_s, $r_{k,m}$, $s_{i,k}$, $P(u_m)$, are the oversampling ratio, the sampled channel-impaired received signal, the estimated signal and the *a priori* information of the coded bits, respectively. Note that in the first iteration, no *a priori* information is fedback by the channel decoder to the equaliser. Therefore, the term $P(u_m)$ is $\frac{1}{2}$, indicating that the bits +1 and −1 have equal probability of occurring. Recall that $A_m(\kappa)$ of Equation 2.30, which is repeated here for convenience, is dependent on $A_{m-1}(\acute{\kappa})$ and $\Gamma_m(\acute{\kappa}, \kappa)$:

$$A_m(\kappa) = \ln\left(\sum_{\text{all } \acute{\kappa}} \exp\left[A_{m-1}(\acute{\kappa}) + \Gamma_m(\acute{\kappa}, \kappa)\right]\right).$$

Therefore, the new values of $A_1(0)$ and $A_1(1)$ can be determined. Consider the transition from state 0 to state 0. Since we have $\Gamma_1(0,0) = -0.3$ and $A_0(0) = 0.0$, the term $A_1(0)$ becomes:

$$\begin{aligned} A_{m=1}(\kappa = 0) &= \ln\left(\exp\left[A_{m=1-1}(\acute{\kappa} = 0) + \Gamma_{m=1}(\acute{\kappa} = 0, \kappa = 0)\right]\right) \\ A_1(0) &= 0.0 + (-0.3) = -0.3, \end{aligned} \tag{3.20}$$

while $A_1(1)$ is:

$$\begin{aligned} A_{m=1}(\kappa = 1) &= \ln\left(\exp\left[A_{m=1-1}(\acute{\kappa} = 0) + \Gamma_{m=1}(\acute{\kappa} = 0, \kappa = 1)\right]\right) \\ A_1(1) &= 0.0 + (-0.5) = -0.5. \end{aligned} \tag{3.21}$$

At the next trellis interval of $m = 2$, the same updating process is repeated. However, in the third interval there are more than one transition reaching a particular state. Consider state 2, where there are merging transitions from state 1 and state 3. In this situation the new value of $A_3(2)$ for state 2 is:

$$\begin{aligned} A_{m=3}(\kappa = 2) &= \ln(\exp\left[A_{m=2}(\acute{\kappa} = 1) + \Gamma_{m=3}(\acute{\kappa} = 1, \kappa = 2)\right] \\ &\quad + \exp\left[A_{m=2}(\acute{\kappa} = 3) + \Gamma_{m=3}(\acute{\kappa} = 3, \kappa = 2)\right]) \\ A_3(2) &= \ln(\exp(-2.8 - 2.7) + \exp(-1.7 - 0.8)) \\ &= \ln(\exp(-5.5) + \exp(-2.5)), \end{aligned} \tag{3.22}$$

which can be evaluated using the Jacobian logarithm [18] of Equation 2.25 and a lookup-table, in order to avoid processing natural logarithm and exponential computations, yielding:

$$\begin{aligned} \ln(\exp(-5.5) + \exp(-2.5)) &= \max(-5.5, -2.5) + \underbrace{\ln(1 + \exp(-|-5.5 - (-2.5)|)}_{\text{Lookup table } f_{diff} = |-5.5 - (-2.5)| = 3.0} \\ &= -2.5 + 0.05 = -2.45, \end{aligned} \tag{3.23}$$

where the lookup table of Reference [18] is detailed in Table 3.2. The same operation is

Range of f_{diff}	$\ln(1 + \exp(-f_{diff}))$
$f_{diff} > 3.70$	0.00
$3.70 \geq f_{diff} > 2.25$	0.05
$2.25 \geq f_{diff} > 1.50$	0.15
$1.50 \geq f_{diff} > 1.05$	0.25
$1.05 \geq f_{diff} > 0.70$	0.35
$0.70 \geq f_{diff} > 0.43$	0.45
$0.43 \geq f_{diff} > 0.20$	0.55
$f_{diff} \leqslant 0.20$	0.65

Table 3.2: Lookup table involved, in order to avoid natural logarithm and exponential computations [18]. The value of f_{diff} is the absolute value of the difference between arguments of the exponential function.

performed for all the other states throughout the trellis, until all values of A_m are obtained.

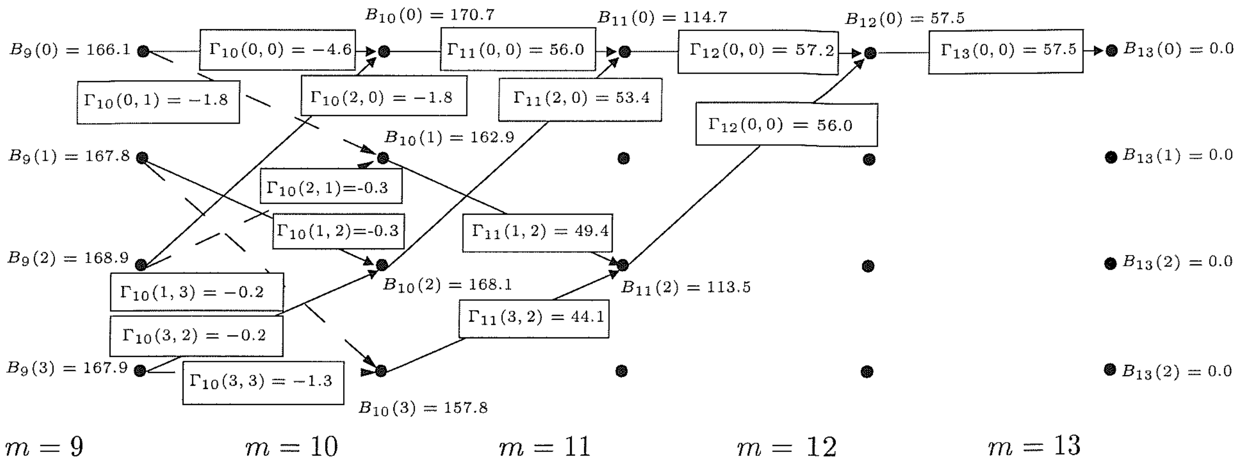


Figure 3.11: Evaluation of $B_m(\acute{\kappa})$ in the turbo equaliser of Figure 3.8 through backward recursion using a range of $\Gamma_m(\acute{\kappa}, \kappa)$ values, which were evaluated from Equation 2.34 by employing our simulation program over the channel of Figure 3.7.

The evaluation of the backward recursion term $B_m(\kappa)$ involves a similar approach. However, the starting point is at the end of the trellis, namely at trellis interval $m = 13$. Initially, the values of $B_{m=13}(\kappa)$ at trellis interval $m = 13$ are set to 0.0, as shown in Figure 3.11. Also observe that since three -1 terminating tailing bits have been inserted in the transmission burst at trellis intervals $m = 11$, $m = 12$ and $m = 13$, the only legitimate transitions at these intervals are due to bit -1. Therefore, the transitions corresponding to a transmission bit of $+1$, are illegitimate and hence are not considered. In order to evaluate the values of $B_{m=12}(\kappa)$, Equation 2.34 is repeated here for convenience:

$$B_{m-1}(\acute{\kappa}) = \ln\left(\sum_{\text{all } \kappa} \exp[B_m(\kappa) + \Gamma_m(\acute{\kappa}, \kappa)]\right).$$

For example, in order to determine the value of $B_{m-1=12}(\acute{\kappa} = 0)$ at $m = 13$, only the values of $B_{m=13}(\acute{\kappa} = 0)$ and $\Gamma_{m=13}(\acute{\kappa} = 0, \kappa = 0)$ are required, yielding:

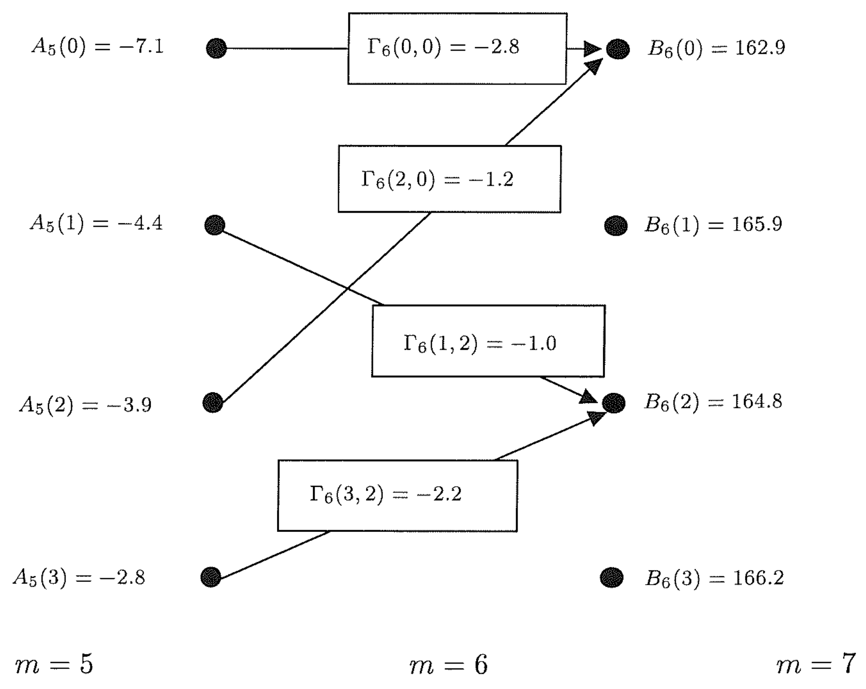
$$\begin{aligned} B_{m-1=12}(\acute{\kappa} = 0) &= \ln(\exp\left[B_{m=13}(\kappa = 0) + \Gamma_{m=13}(\acute{\kappa} = 0, \kappa = 0)\right] \\ &= \ln(\exp(0.0 + 57.5) \\ &= 57.5. \end{aligned} \tag{3.24}$$

This is because there is only one legitimate path, which is caused by the trellis termination bit -1. As seen in Figure 3.11, this backward recursion process is repeated, in order to determine the values of $B_{m-1=11}(\acute{\kappa})$ of the states $\acute{\kappa} = 0 \ldots 3$ at interval $m = 12$, although due to the tailing bits of -1 only $B_{m-1=11}(\acute{\kappa} = 0)$ and $B_{m-1=11}(\acute{\kappa} = 2)$ have to be computed by backtracing. At this stage, the Jacobian logarithmic relationship is not utilised, since there is only one path to be considered at state $\acute{\kappa} = 0$ and state $\acute{\kappa} = 2$, generated by bit -1. Consequently, $B_{m-1}(\acute{\kappa})$ in Equation 2.34 is expressed as the natural logarithm of a single exponential term. However, at trellis interval $m = 10$ the paths associated with the bit $+1$ are also considered in Figure 3.11, since the received signal is no longer due to the -1 tailing bits. Considering state 0, it is observed that there are two transitions leaving this state during trellis interval $m = 10$, where the first is arriving at state 0, while the other at state 1. Therefore, the probabilities of these transitions leaving state 0 during $m = 10$ are summed to give:

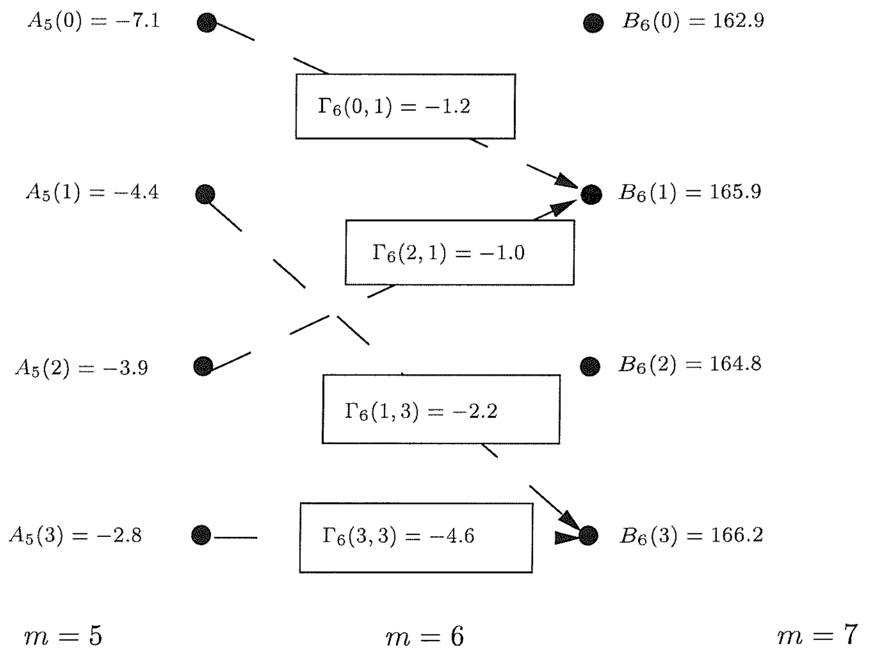
$$\begin{aligned} B_{m-1=9}(\acute{\kappa} = 0) &= \ln(\exp\left[B_{m=10}(\kappa = 0) + \Gamma_{m=10}(\acute{\kappa} = 0, \kappa = 0)\right] \\ &\quad + \exp\left[B_{m=10}(\kappa = 1) + \Gamma_{m=10}(\acute{\kappa} = 0, \kappa = 1)\right]) \\ &= \ln(\exp(170.7 + (-4.6)) + \exp(162.9 + (-1.8))) \\ &= \max(166.1, 161.1) + \underbrace{\ln(1 + \exp(-|166.1 - 161.1|))}_{\text{Lookup table:} f_{diff} = |166.1 - 161.1)| = 5.0} \\ &= 166.1 + 0.00 = 166.1, \end{aligned} \tag{3.25}$$

where the Jacobian logarithmic relationship was employed again, in order to simplify the computation of the natural logarithm of a sum of exponentials to an operation involving a maximisation process and a look-up table operation. By repeating the above backward recursion, all values of $B_m(\kappa)$ for states $\kappa = 0 \ldots 3$ and intervals $m = 1 \ldots 13$ can be determined, some of which were summarised for the sake of illustration in Figure 3.11

Having determined the values of $A_m(\kappa)$, $B_m(\kappa)$ for all states and the $\Gamma(\acute{\kappa}, \kappa)$ values associated with all transitions, the LLR of the bit u_m can be computed by using Equation 2.4. This involves identifying the trellis transitions caused by bit $u_m = -1$ and summing the transition probabilities of each path. Since according to Equation 2.1 the associated probabilities can be expressed as a sum of exponential terms, the LLR is then the natural logarithm of a sum of exponential values. Hence, the Jacobian logarithmic relationship in Equation 2.25 and the lookup table (LUT) of Table 3.2 can again be employed, in order to reduce the complexity associated with computing exponentials and logarithms. Figure 3.12(a) illustrates the set of transitions caused by $u_m = -1$, while Figure 3.12(b) depicts the set of branches corresponding to the bit $u_m = +1$ at trellis interval $m = 6$, which were extracted from the generic trellis seen in Equation 3.9. Considering this trellis interval as an example, the natural logarithm of the transition probability associated with the bit $u_m = -1$ is given



(a) Transitions corresponding to $u_m = -1$ bits



(b) Transitions corresponding to $u_m = +1$ bits

Figure 3.12: Equaliser trellis transitions, which were considered when evaluating the corresponding LLR values of bit -1 and $+1$ at trellis interval $m = 6$.

by the natural logarithm of the denominator in Equation 2.40, yielding:

$$\begin{aligned}\ln[P(u_m = -1|r)] = \ln[&\exp(A_5(0) + \Gamma_6(0,0) + B_6(0)) + \exp(A_5(1) + \Gamma_6(1,2) + B_6(2)) \\ &+ \exp(A_5(2) + \Gamma_6(2,0) + B_6(0)) + \exp(A_5(3) + \Gamma_6(3,2) + B_6(2))].\end{aligned} \tag{3.26}$$

Since Equation 3.26 is the natural logarithm of the sum of four exponentials terms, the generalised Jacobian logarithmic relationship of Equation 2.41 is applied recursively, giving:

$$\begin{aligned}\ln[P(u_m = -1|r)] &= \ln[\exp(-7.1 + (-2.8) + 162.9) + \exp(-4.4 + (-1.0) + 164.8) \\ &\quad + \exp(-3.9 + (-1.2) + 162.9) + \exp(-2.8 + (-2.2) + 164.8)] \\ &\approx \ln[\exp(\overbrace{\max(153.0, 159.4) + \underbrace{0.0}_{\text{LUT}}}^{=159.4}) + \exp(-3.9 + (-1.2) + 162.9) \\ &\quad + \exp(-2.8 + (-2.2) + 164.8)] \\ &\approx \ln[\exp(\overbrace{\max(159.4, 157.8) + \underbrace{0.15}_{\text{LUT}}}^{=159.55}) + \exp(-2.8 + (-2.2) + 164.8)] \\ &\approx \max(159.55, 159.8) + \underbrace{0.55}_{\text{LUT}} \\ &\approx 160.35.\end{aligned} \tag{3.27}$$

Similarly, the paths corresponding to bit $u_m = +1$ are grouped together with the aid of Equation 2.40 and Figure 3.12(b), in order to yield the natural logarithm of the probability of the bit $u_m = +1$ being received given that r was transmitted:

$$\begin{aligned}\ln[P(u_m = +1|r)] &= \ln[\exp(-7.1 + (-1.2) + 165.9) + \exp(-4.4 + (-2.2) + 166.2) \\ &\quad + \exp(-3.9 + (-1.0) + 165.9) + \exp(-2.8 + (-4.6) + 166.2)] \\ &\approx \ln[\exp(\overbrace{\max(157.6, 159.6) + \underbrace{0.15}_{\text{LUT}}}^{=159.75}) + \exp(-3.9 + (-1.0) + 165.9) \\ &\quad + \exp(-2.8 + (-4.6) + 166.2)] \\ &\approx \ln[\exp(\overbrace{\max(159.75, 161.0) + \underbrace{0.25}_{\text{LUT}}}^{161.25}) + \exp(-2.8 + (-4.6) + 166.2)] \\ &\approx \max(161.25, 158.8) + \underbrace{0.05}_{\text{LUT}} \\ &\approx 161.3.\end{aligned} \tag{3.28}$$

Recall from Equation 2.1, repeated here for convenience as Equation 3.29, that the LLR is the natural logarithm of the ratio of $P(u_m = +1|r)$ to $P(u_m = -1|r)$:

$$\begin{aligned} L(u_m|r) &\triangleq \ln\left(\frac{P(u_m = +1|r)}{P(u_m = -1|r)}\right) \\ &= \ln P(u_m = +1|r) - \ln P(u_m = -1|r). \end{aligned} \tag{3.29}$$

Hence, with the aid of the results from Equations 3.27 and 3.28, the *a posteriori* LLR of bit u_m at trellis interval $m = 6$ is $161.3 - 160.35 = 0.95$. By applying the same approach to all trellis stages, the *a posteriori* LLRs of all the 13 bits u_m can be determined. Table 3.3

	Equaliser trellis interval m												
	1	2	3	4	5	6	7	8	9	10	11	12	13
APRI	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
APOS	2.7	2.1	-0.5	1.1	-0.1	1.0	-0.6	0.2	-0.4	-6.5	-123.0	-119.9	-116.2
EXT	2.7	2.1	-0.5	1.1	-0.1	1.0	-0.6	0.2	-0.4	-6.5	-123.0	-119.9	-116.2
TXD-C	1	-1	1	-1	1	1	-1	1	-1	-1	-1	-1	-1
DCSN	1	**1**	**-1**	**1**	**-1**	1	-1	1	-1	-1	-1	-1	-1

Table 3.3: Example of various turbo equaliser probabilities after the first iteration. Notations TXD-C, APRI, APOS, EXT and DCSN represent the transmitted encoded bits, which have been interleaved, the *a priori* LLR, the *a posteriori* LLR, the combined channel and extrinsic LLR, and the decision bits, respectively. In the first iteration the equaliser does not receive any *a priori* information — hence APR= 0.0— and it was observed that four bits — emphasised in bold — were corrupted.

shows the transmitted channel encoded bits, which have been interleaved, in order to yield the sequence (TXD-C). The associated *a priori* LLRs (APRI), the *a posteriori* LLRs (APOS), the combined channel and extrinsic LLRs (EXT) and the decision bits (DCSN) in the first turbo equalisation iteration are also shown in the table. At this stage, no *a priori* information was received by the equaliser, yielding the values of 0.0 in the table. From the *a posteriori* LLRs computed, the detected bit is determined by using a threshold of 0, *i.e.* a decision of $+1$ is made when the LLR value is ≥ 0 and -1 when the LLR value is $<$ 0. The APOS value of 1.0 at $m = 6$ was due to the rounding of the previously calculated value of 0.95. It was observed in Table 3.3 that four errors — shown in bold — were made.

At this stage, we have shown how the LLR of bit u_m is determined by the Log-MAP equaliser using the example of a simple convolutional-coded BPSK system. Let us now consider the combined channel and extrinsic LLR denoted by EXT in Table 3.3, which is the only remaining quantity to be derived. At the next stage of operations, the LLR generated by the equaliser is passed to the channel decoder. However, as mentioned previously, a vital rule for turbo equalisation is that the input information to a particular block in the current iteration must not include the information contributed by that particular block from the previous iteration, since then the information used in consecutive iterations would be dependent on each other hence would fail to achieve iteration gain. Therefore, the *a posteriori* LLR, which consists of the combined channel and extrinsic information plus the *a priori* information provided by the channel decoder, must be processed in order to remove the previous decoder contribution in the form of the *a priori* LLR, before it is passed to the decoder. This is achieved by storing the previous decoder contribution and subtracting it from the *a posteriori* LLR produced by the equaliser, as it was illustrated in Figure 3.8. Since in the first iteration there is no *a priori* information generated by the decoder, the corresponding APRI contributions in Table 3.3 are zero and hence the *a posteriori* LLR constitutes also the combined channel and extrinsic LLR, which can be passed directly to the decoder. Observe furthermore that the equaliser produces LLR values also for the three tail bits. Although these tail bit LLR values are not required by the decoder, they are employed in the equalisation process. This is because the tail bits represent contributions from the data symbols at the edge of the burst, which were spread by the dispersive channel into the adjacent symbols. Should the tail bits be ignored, low-reliability LLR values will be obtained for the bits at the edge of the received burst, since some of the symbol energy and the associated 'smeared' information is lost.

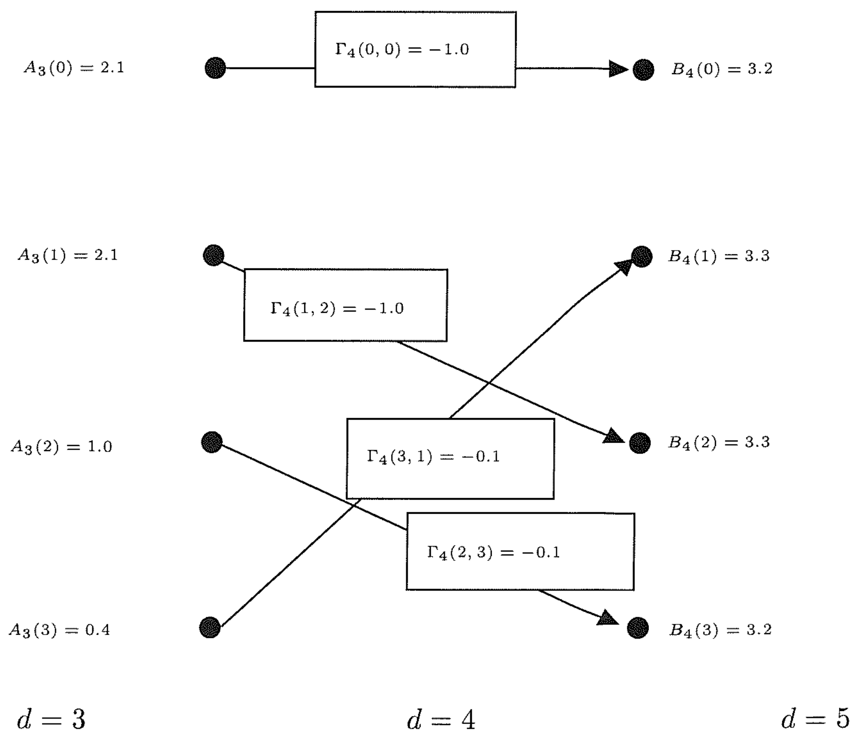
Having highlighted the operation of the Log-MAP equaliser, we now proceed to describe the Log-MAP decoder in the context of turbo equalisation. As depicted in Figure 3.3, the decoder receives the combined channel and extrinsic LLR $L_p^i = \{L_p^{i;s}, L_p^{i;h}\}$, which has been deinterleaved. Here, the purpose of the deinterleaver is to minimise the correlation between the equaliser's input information and the input of the decoder. It also presents the information in the right order to the decoder, since previously the channel interleaver π_c of Figure 3.3 rearranged the bits in the order required by the equaliser. Recall that a rate $R = \frac{1}{2}$ $K = 3$ recursive, systematic convolutional encoder using the octal generator polynomials of 7 and 5 was employed in this example. Since the constraint length K is 3, the decoder's trellis consists of $2^{K-1} = 2^2 = 4$ states, as illustrated in Figure 3.6. As in the Log-MAP equaliser, the Log-MAP decoder also computes the values of $A_m(\kappa)$, $B_m(\kappa)$ and $\Gamma_m(\grave{\kappa}, \kappa)$ using Equations 2.30, 2.34 and 2.38, respectively, for all states and all the trellis stages. However, as mentioned in Subsection 3.4, the calculation of $\Gamma_m(\grave{\kappa}, \kappa)$ — which is evaluated using Equation 3.16 — can be simplified, since the received coded bits and the square of the expected coded bits are always constant for all trellis transitions. Note that in contrast to the equaliser trellis interval index m, here we will use d to denote the index of the decoder trellis intervals. Once these parameters are calculated, the LLRs of both the source and coded bits are determined, in contrast to stand-alone turbo decoding schemes, which only compute the LLRs of the source bits. Therefore, at each rate $R = \frac{1}{2}$ decoder trellis instant the LLR values of two coded bits are produced. In our following discussions, we will demonstrate, how the LLR of each coded bit is determined. Figure 3.13(a) is a portion of the decoder trellis at trellis interval $d = 4$, which shows the set of branches corresponding to the first coded bit $c_{1,4} = -1$, while Figure 3.13(b) illustrates the group of trellis transitions caused by $c_{1,4} = +1$. Note that when a systematic encoder is employed, the LLR of the first coded bit $c_{1,d}$ constitutes the LLR of the source bit. Similarly to our approach in the Log-MAP equaliser, the values of $\ln[P(c_{1,4} = +1|L_p^i)]$ and $\ln[P(c_{1,4} = -1|L_p^i)]$ are determined in order to compute the LLR of $c_{1,4}$. From Figure 3.13(a) and using Equation 3.11, the term $\ln[P(c_{1,4} = -1|L_p^i)]$ is given by:

$$\begin{aligned}
\ln[P(c_{1,4} = -1|L_p^i)] &= \ln[\exp(2.1 + (-1.0) + 3.2) + + \exp(2.1 + (-1.0) + 3.3) \\
&\exp(1.0 + (-0.1) + 3.2) + \exp(0.4 + (-0.1) + 3.3)] \\
&\approx \ln[\exp(\overbrace{\max(4.3, 4.4) + \underbrace{0.65}_{\text{LUT}}}^{5.05}) + \exp(1.0 + (-0.1) + 3.2) \\
&+ \exp(0.4 + (-0.1) + 3.3)] \\
&\approx \ln[\exp(\overbrace{\max(5.05, 4.1) + \underbrace{0.35}_{\text{LUT}}}^{5.4}) + \exp(0.4 + (-0.1) + 3.3)] \\
&\approx \max(5.4, 3.6) + \underbrace{0.15}_{\text{LUT}} \\
&\approx 5.55.
\end{aligned} \tag{3.30}$$

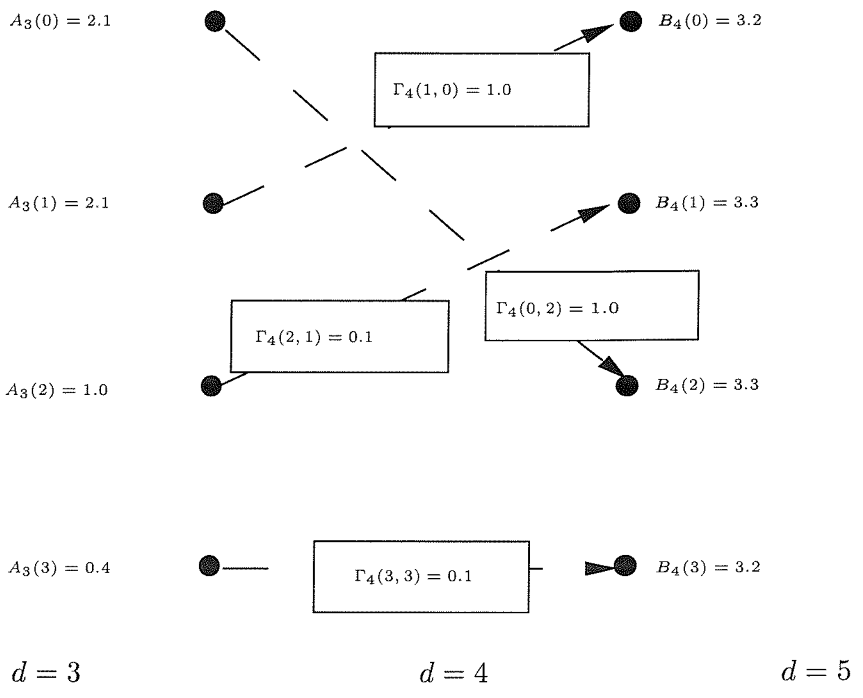
while $\ln[P(c_{1,4} = +1|L_p^i)]$ becomes:

$$\begin{aligned}
\ln[P(c_{1,4} = +1|L_p^i)] &= \ln[\exp(2.1 + 1.0 + 3.3) + \exp(2.1 + 1.0 + 3.2) \\
&+ \exp(1.0 + 0.1 + 3.3) + \exp(0.4 + 0.1 + 3.2)] \\
&\approx \ln[\exp(\overbrace{\max(6.4, 6.3) + \underbrace{0.65}_{\text{LUT}}}^{7.05}) + \exp(1.0 + 0.1 + 3.3) \\
&+ \exp(0.4 + 0.1 + 3.2)] \\
&\approx \ln[\exp(\overbrace{\max(7.05, 4.4) + \underbrace{0.05}_{\text{LUT}} + \exp}^{7.1}(0.4 + 0.1 + 3.2)) \\
&\approx \max(7.1, 3.7) + \underbrace{0.05}_{\text{LUT}} \\
&\approx 7.15.
\end{aligned} \tag{3.31}$$

Therefore, the LLR of bit $c_{1,4}$ at trellis interval $d = 4$ is $\ln[P(c_{1,4} = +1|L_p^i)] - \ln[P(c_{1,4} = -1|L_p^i)] = 7.15 - 5.55 = 1.6$. By following the same approach, the LLR of the second coded

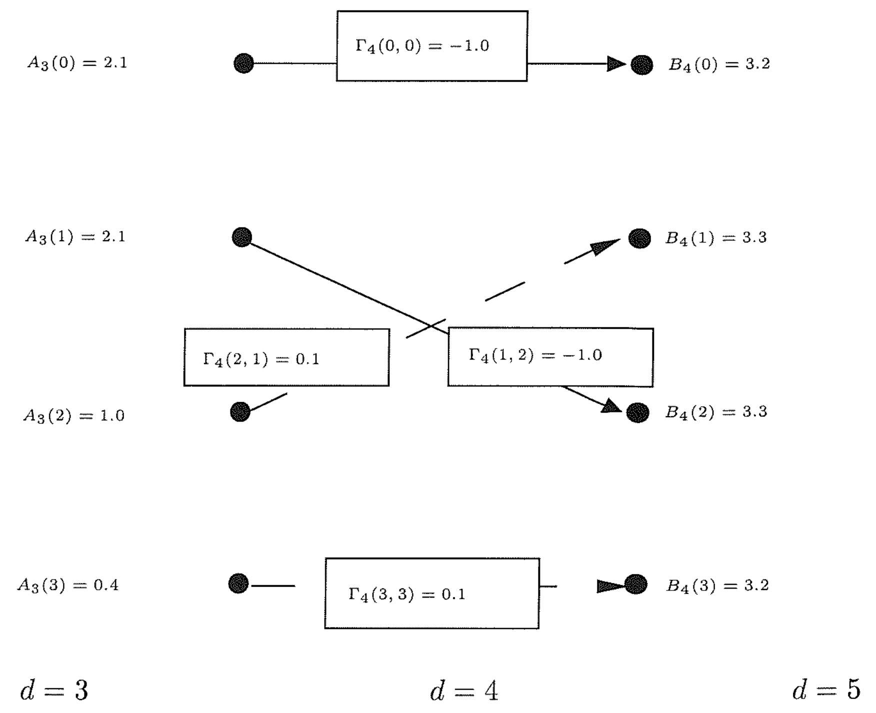


(a) Transitions corresponding to coded bits $c_{1,4} = -1$

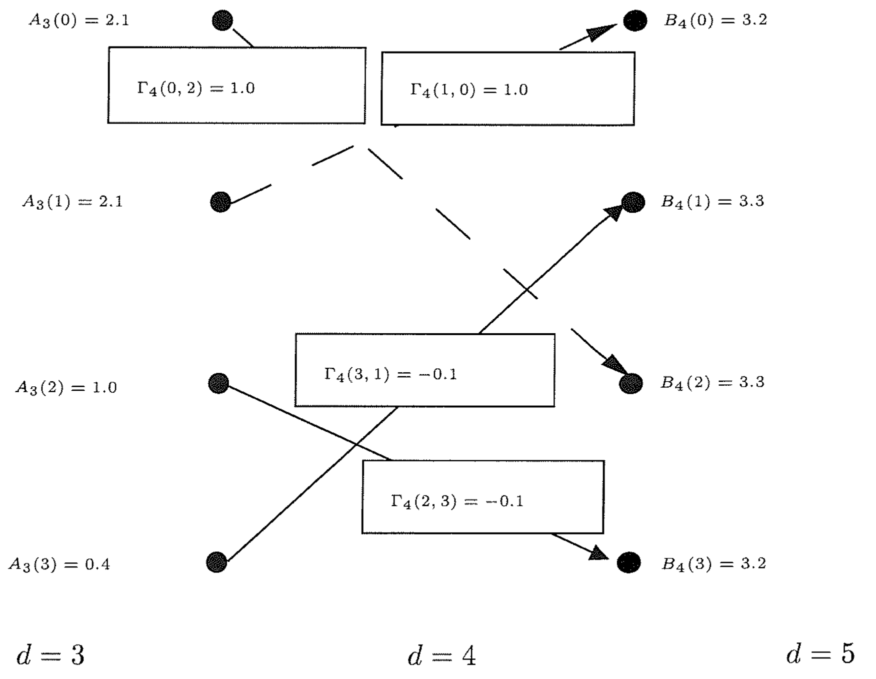


(b) Transitions corresponding to coded bits $c_{1,4} = +1$

Figure 3.13: Half-rate, $K = 3$ RSC channel decoder trellis transitions, which were considered when evaluating the corresponding LLR values of $c_{1,4}$, which is the first coded bit at interval $d = 4$. The trellis is based on Figure 3.6.



(a) Transitions corresponding to coded bits $c_{2,4} = -1$



(b) Transitions corresponding to coded bits $c_{2,4} = +1$

Figure 3.14: Half-rate, $K = 3$ RSC channel decoder trellis transitions, which were considered when evaluating the corresponding LLR values of $c_{2,4}$, which is the second code bit at interval $d = 4$. The trellis is based on Figure 3.6

bit $c_{2,4}$ at interval $d = 4$ can also be evaluated. Explicitly, we must calculate $\ln[P(c_{2,4} = -1|L_p^i)]$ and $\ln[P(c_{2,4} = +1|L_p^i)]$ by considering a different set of transitions obeying the trellis diagram of Figure 3.6, which were caused by the coded bit $c_{2,4} = +1$ and $c_{2,4} = -1$, respectively. Considering the set of transitions corresponding to bit $c_{2,4} = -1$ in Figure 3.14(a), the term $\ln[P(c_{2,4} = -1|L_p^i)]$ is:

$$\begin{aligned}
\ln[P(c_{2,4} = -1|L_p^i)] &= \ln[\exp(2.1 + (-1.0) + 3.2) + \exp(2.1 + (-1.0) + 3.3) \\
&\quad + \exp(1.0 + 0.1 + 3.3) + \exp(0.4 + 0.1 + 3.2)] \\
&\approx \ln[\exp(\overbrace{\max(4.3, 4.4) + \underbrace{0.65}_{\text{LUT}}}^{5.15}) + \exp(1.0 + 0.1 + 3.3) \\
&\quad + \exp(0.4 + 0.1 + 3.2)] \qquad (3.32)\\
&\approx \ln[\exp(\overbrace{\max(5.15, 4.4) + \underbrace{0.25}_{\text{LUT}}}^{5.4}) + \exp(0.4 + 0.1 + 3.2)] \\
&\approx \max(5.4, 3.7) + \underbrace{0.15}_{\text{LUT}} \\
&\approx 5.55,
\end{aligned}$$

while $\ln[P(c_{2,4} = +1|L_p^i)]$ can be written as:

$$\begin{aligned}
\ln[P(c_{2,4} = +1|L_p^i)] &= \ln[\exp(2.1 + 1.0 + 3.3) + \exp(2.1 + 1.0 + 3.2) \\
&\quad + \exp(1.0 + (-0.1) + 3.2) + \exp(0.4 + (-0.1) + 3.3)] \\
&\approx \ln[\exp(\overbrace{\max(6.4, 6.3) + \underbrace{0.65}_{\text{LUT}}}^{7.05}) + \exp(1.0 + (-0.1) + 3.2) \\
&\quad + \exp(0.4 + (-0.1) + 3.3)] \qquad (3.33)\\
&\approx \ln[\exp(\overbrace{\max(7.05, 4.1) + \underbrace{0.05}_{\text{LUT}}}^{7.1}) + \exp(0.4 + (-0.1) + 3.3)] \\
&\approx \max(7.1, 3.6) + \underbrace{0.05}_{\text{LUT}} \\
&\approx 7.15
\end{aligned}$$

upon considering all legitimate trellis transitions corresponding to the bit $c_{2,4} = +1$. Therefore, the *a posteriori* LLR for bit $c_{2,4}$ at interval $m = 4$ is $7.15 - 5.55 = 1.6$. This operation is applied to all trellis transition intervals in the trellis to determine the LLRs of the coded bits. In the last iteration, only the *a posteriori* LLR of the source bit is computed instead of both coded bits, since only the transmitted source bits have to be determined. Note that in our example a decision concerning the source bits — based on their *a posteriori* LLR — was made even after the first iteration, as shown in Table 3.4.

	Decoder trellis interval d				
	1	2	3	4	5
APRI-S	0.0	0.0	0.0	0.0	0.0
RXD-LLR	2.7,-0.5	-0.1,-0.6	-0.4,2.1	1.1,0.8	0.2,-6.5
APOS-C	2.5,2.5	0.2,-0.3	0.1,2.3	1.6,1.6	0.2,-6.5
EXT-C	-0.2,3.0	0.3,0.3	0.5,0.2	0.6,0.8	0.0, 0.0
APOS-S	2.5	0.2	0.1	1.6	0.2
TXD-S	1	1	-1	-1	1
DCSN	1	1	**1**	**1**	1

Table 3.4: Various metrics of the convolutional decoder in Figure 3.8 extracted from our simulations after the first turbo equalisation iteration. The convolutional decoder never receives any source bit *a priori* (APRI-S) information, unless this information can be extracted from another independent source, such as another convolutional decoder. The extrinsic LLR (EXT-C) is computed by subtracting the received combined channel and extrinsic LLR (RXD-LLR) from the decoder's *a posteriori* LLR of the code bits (APOS-C). The decision bits (DCSN) — obtained from the the decoder source *a posteriori* (APOS-S) information — is compared with the transmitted source bits (TXD-S) to determine the number of erroneous bits. It is observed that two errors were obtained in the first iteration.

This decision was invoked for determining the decoding performance of the convolutional code using the Log-MAP algorithm, which is equivalent to the performance of turbo equalisation after one turbo equalisation iteration. In general, however the detected bits are only determined in the final iteration.

Various metrics characterising the convolutional decoder of Figure 3.8 were extracted from our simulations after the first turbo equalisation iteration and summarised in Table 3.4. Let us first consider the information received by the Log-MAP decoder, followed by the output information of the decoder. It is observed in Table 3.4 that the convolutional decoder never receives any source-bit-related *a priori* information (APRI-S), unless this information can be extracted from another independent source, such as another convolutional decoder in turbo-coded systems. Therefore, similarly to the *a priori* LLR of the coded bit received by the equaliser, namely APRI in Table 3.3, in the first iteration the *a priori* LLR of the source bit, namely APRI-S is also initialised to 0, indicating that the bits $+1$ and -1 have equal probability of occurring. However, in contrast to the Log-MAP equaliser, in the absence of additional decoders the values of APRI-S remain at 0 in successive turbo equalisation iterations, whereas the reliability of the *a priori* LLRs at the input of the equaliser APRI improves, as it will be shown next in our discussion of the turbo equaliser operations during the second iteration. As mentioned previously, the decoder also received the LLR associated with the combined channel and extrinsic information, namely EXT of Table 3.3, which had been channel-deinterleaved to give RXD-LLR of Table 3.4. Subsequently, with the aid of these received LLR values *i.e.* APRI-S as well as RXD-LLR and by employing the principles highlighted above, the Log-MAP decoder is now ready to determine the *a posteriori* LLRs of the source bits and the coded bits, namely (APOS-S) and (APOS-C), of Table 3.4, respectively. Recall that the input information to a particular block in the current iteration must not include the information contributed by that particular block from in previous iteration, in order to ensure minimum correlation between these values. Therefore, as seen at the bottom of Figure 3.8, the combined channel and extrinsic LLR RXD-LLR is subtracted from APOS-C, in order to yield the extrinsic LLR (EXT-C), which is subsequently channel-interleaved, before being passed to the Log-MAP equaliser in the next iteration as the *a priori* LLR APRI of Table 3.5. This additional information will be utilised by the equaliser in the second iteration, in order to improve the reliability of the *a posteriori* LLR, *i.e.* to increase our confidence in the associated decision. The *a posteriori* LLR of the source bits, namely APOS-S in Table 3.4 is used to detect the transmitted source bits, which is denoted by DCSN in Table 3.4. . By comparing DCSN with the source bits that were actually transmitted (TXD-S), it was observed that two errors — which were emphasised in bold in Table 3.4 — were obtained in the first iteration. Continuing with our example, we will show next that the iterative exchange of information between the decoder and equaliser will improve the performance further, such that the turbo equaliser outperforms independent equalisation and decoding.

	Equaliser trellis interval m												
	1	2	3	4	5	6	7	8	9	10	11	12	13
APRI	-0.2	0.2	2.9	0.5	0.3	0.8	0.3	0.1	0.5	-0.1	-115.1	-115.1	-115.1
APOS	2.0	1.2	2.2	0.1	0.1	1.3	-0.6	-0.1	-0.1	-6.7	-123.1	-119.9	-116.2
EXT	2.2	1.0	-0.7	-0.4	-0.2	0.5	-0.9	-0.2	-0.6	-6.6	-8.0	-4.8	-1.1
TXD-C	1	-1	1	-1	1	1	-1	1	-1	-1	-1	-1	-1
DCSN	1	**1**	1	**1**	1	1	-1	**-1**	-1	-1	-1	-1	-1

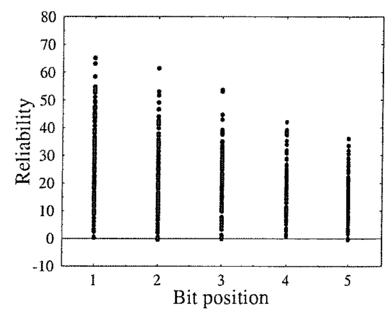
Table 3.5: Various equaliser metrics extracted from our simulations after the second iteration. Notations TXD-C, APRI, APOS, EXT and DCSN represent the transmitted encoded bits, which have been interleaved, the *a priori* LLR, the *a posteriori* LLR, extrinsic LLR and the detected or decision bits. In the second iteration the equaliser now obtains *a priori* information provided by the convolutional decoder, hence improving the reliability of the a *posteriori* LLRs evaluated, generating only three errors, emphasised in bold.

In the second iteration, the equaliser receives *a priori* information of the source bits APRI-S from the decoder output of the first iteration, as shown in Table 3.5 . Recall that in the first turbo equalisation iteration the *a priori* LLR shown in the top line of

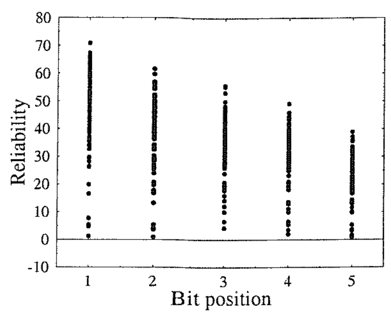
	Decoder trellis interval d				
	1	2	3	4	5
APRI-S	0.0	0.0	0.0	0.0	0.0
RXD-LLR	2.2,-0.7	-0.2,-0.9	-0.5,1.0	-0.4,0.5	-0.1,-6.7
APOS-C	1.4,1.4	0.3,-0.6	-0.2,+0.7	-0.2,0.4	0.1,-6.7
EXT-C	-0.8,2.1	0.5,-0.3	0.3,-0.3	0.2,-0.1	0.2, 0.0
APOS-S	1.4	0.3	-0.2	-0.2	0.1
TXD-S	1	1	-1	-1	1
DCSN	1	1	-1	-1	1

Table 3.6: Channel decoder metrics extracted from our simulations after the second turbo equalisation iteration. No *a priori* APRI-S information is received. As in the first iteration, the extrinsic LLR EXT-C is computed by subtracting the received combined channel and extrinsic LLR RXD-LLR from the decoder's *a posteriori* LLR of the code bits APOS-C. Since the decoder receives higher-reliability LLR values, it produces source *a posteriori* APOS-S, which results in an error-free output.

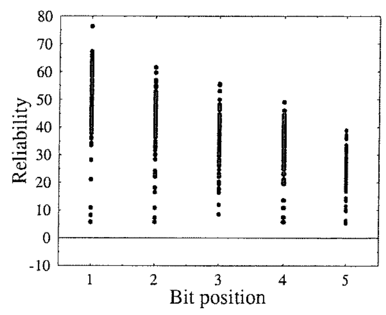
Table 3.3 was initialised to 0, indicating that the bits +1 and −1 had equal probability of being transmitted. Now this additional information assists the equaliser in providing more accurate values of the *a posteriori* LLRs APOS. The improved reliability is reflected in the reduced number of errors, as compared to the results of the first iteration, seen in Table 3.3, indicating a reduction from four to three errors. As before, the *a priori* information is subtracted from the *a posteriori* LLR at the output of the Log-MAP equaliser of Figure 3.8, yielding the combined channel and extrinsic LLR EXT, which is subsequently deinterleaved. At the decoder the received combined channel and extrinsic LLR is processed using the Log-MAP algorithm and — as shown in Table 3.6 — the decoder corrects the two errors incurred in the previous iteration, characterised in Table 3.4. It is observed that by performing the equalisation and decoding jointly, the dispersion induced by the channel can be overcome, as opposed to the scenario characterised in Table 3.4, which is equivalent to situation employing independent equalisation and decoding. In order to further improve the performance of the system, more turbo equalisation iterations can be performed. However, in this example, it was sufficient to perform two turbo equalisation iterations, in order to obtain an error-free performance. Figure 3.15 illustrates the reliability of the LLRs — expressed as the product of the decoder's *a posteriori* LLR and the actual source bits transmitted — versus the source bit index. A large positive reliability value indicates a high-confidence LLR value, whereas a negative value represents an erroneous decision. It is observed that in the first iteration of Figure 3.15(a) there was a large number of low reliability LLR values. The negative reliability values in Figure 3.15(a) indicate that errors have been made. In the second turbo equalisation iteration of Figure 3.15(b) the



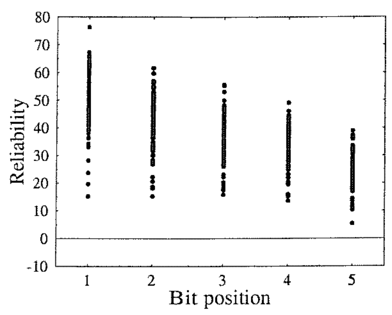
(a) One iteration



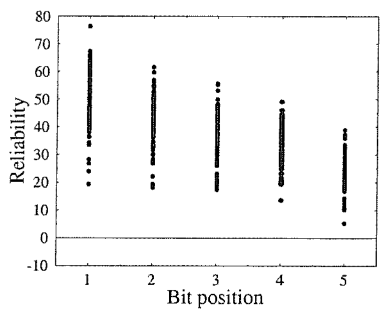
(b) Two iterations



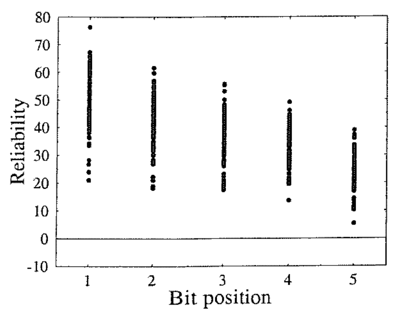
(c) Three iterations



(d) Four iterations



(e) Five iterations



(f) Six iterations

Figure 3.15: Reliability of the decoder *a posteriori* LLR with each turbo equalisation iteration. The reliability is expressed as the product of the source *a posteriori* LLR produced by the decoder and the transmitted source bit.

number of low-confidence LLR values was reduced as compared to the first iteration. With increasing turbo equalisation iterations — for example after six turbo equalisation iterations, as shown in Figure 3.15(f) — the number of low-confidence LLR values was significantly reduced, hence justifying the improved BER performance obtained with an increasing number of turbo equalisation iterations.

However, as it can be seen from the results shown at a later stage, the additional gain decreases with an increasing number of turbo equalisation iterations. Iteration gains are achieved, when the decoder is capable of correcting errors, which were not corrected by the equaliser. When the number of turbo equalisation iterations increases, the deinterleaved output of the equaliser becomes more similar to the decoder output, yielding a high correlation. This diminishes the advantage of having information from both the equaliser and decoder, hence reducing the iteration gain.

3.6 Summary of Turbo Equalisation

In this section we have shown the operation and structure of the original turbo equaliser presented by Douillard *et al* [10] and subsequently extended the principle to turbo equalisation schemes utilising multiple decoders, as in turbo decoders. The SISO equaliser must be able to accept both the *a priori* information provided by the channel decoder as well as channel outputs and to utilise both of these information sources, in order to calculate the LLR of the received encoded bits. Here, the only modification to our MAP algorithm highlighted in Subsection 2.2 and to the Log-MAP algorithm of Subsection 2.5, for employment in our iterative soft output GMSK equaliser was to consider the *a priori* probability $P(u_m)$ in the calculation of $\gamma_m(\acute{\kappa}, \kappa)$ or $\Gamma_m(\acute{\kappa}, \kappa)$, since $P(u_m)$ was no longer constant — in contrast to the non-iterative scenario. The decoder(s) in the turbo equaliser must also be able to provide soft outputs in the form of the source and parity LLR values. Therefore, instead of implementing a hard decision trellis-based algorithm for the decoder — such as the Viterbi Algorithm —, we have used an optimal SISO algorithm. Specifically, the Log-MAP algorithm was employed since it guaranteed a performance identical to that of the optimal MAP algorithm, despite having lower computational complexity. We then showed in Section 3.4 that the *a posteriori* LLR of both the parity and source bits can be calculated by considering all trellis branches, where the parity bit or source bit is -1 or $+1$, as expressed in Equation 3.11. The *a posteriori* LLR of each turbo equalisation stage was subsequently processed, in order to yield the combined channel and extrinsic information of the equaliser or the extrinsic information of the decoder, such that the input information to a particular block in the current iteration did not include the information contributed by that particular block from the previous iteration, in order to avoid any correlation between the information used in consecutive iterations.

In the next section the performance of various coded GMSK systems employing turbo equalisation is characterised.

3.7 Performance of Coded GMSK Systems using Turbo Equalisation

In this section the turbo equalisation performance for three coded systems, namely that of the convolutional-coded GMSK system, the convolutional-coding based turbo-coded GMSK scheme and the BCH-coding based turbo coded system are presented. These systems were evaluated over the non-dispersive Gaussian channel and over the equally-weighted and symbol-spaced five-path Rayleigh fading channel experiencing a normalised Doppler frequency of $f_d = 1.5\times10^{-4}$, which corresponds to a transmission frequency of 900 MHz, vehicular speed of 30 mph and a signalling rate of 270.833 KBaud. For our investigations, we have employed the transmission burst structure specified by GSM [2, 3] consisting of 3 tail bits at both ends of the burst and two 57-bit data segments separated by a 26-bit midamble. We have assumed that the channel impulse response was known. Furthermore, the fading magnitude and phase was kept constant for the duration of a transmission burst, a condition which we refer to as employing burst-invariant fading. In the following subsections the simulation parameters — which are specific to the class of the encoder utilised — are outlined.

Encoder type	Component code parameters
Convolutional code	Rate $R = 0.5$, Constraint length $K = 5$ RSC Octal generator polynomials $G_0 = 35$ $G_1 = 23$
Convolutional-coding based turbo code	Rate $R = 0.5$, Constraint length $K = 5$ RSC Octal generator polynomials $G_0 = 35$ $G_1 = 23$
BCH-coding based turbo code	Rate $R = \frac{11}{19} = 0.58$ BCH (15,11)

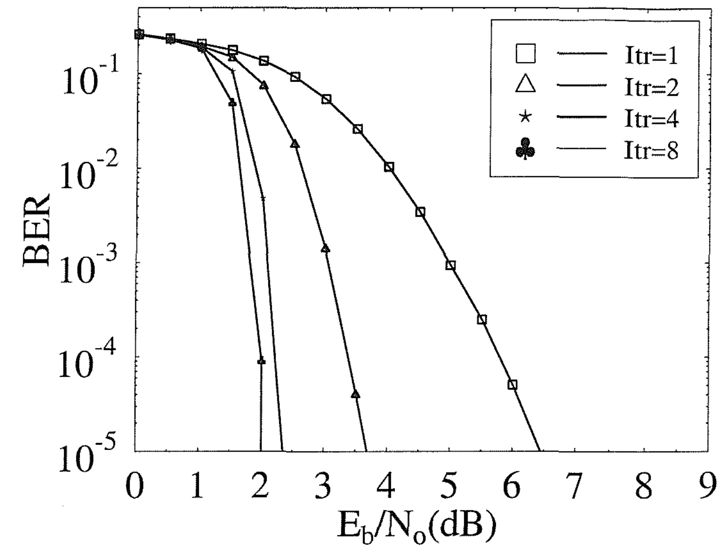
Table 3.7: Parameters of the encoders employed in the rate $R = 0.5$ convolutional-coded GMSK system, the $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme and the rate $R = \frac{11}{19} = 0.58$ BCH-coding based turbo-coded GMSK system.

3.7.1 Convolutional-coded GMSK System

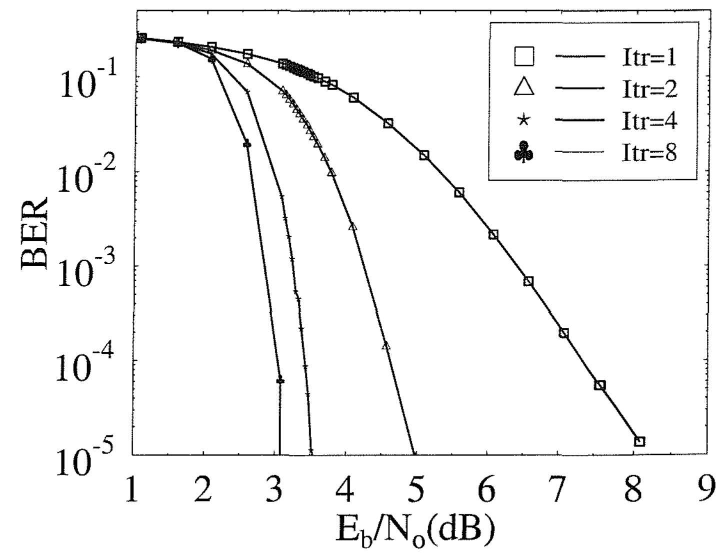
Non-dispersive Gaussian channel		5-path Rayleigh fading channel	
Iteration index	Iteration gain	Iteration index	Iteration gain
2	2.5 dB	2	2.9 dB
4	3.5 dB	4	3.9 dB
8	3.7 dB	8	4.2 dB

Table 3.8: The iteration gains relative to the first iteration at BER $= 10^{-4}$ for the $R = 0.5$ convolutional-coded GMSK system employing turbo equalisation over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading.

A code rate $R = 0.5$, constraint length $K = 5$, recursive, systematic convolutional encoder was employed. The octal generator polynomials of the encoder were $G_0 = 35$ and $G_1 = 23$ as summarised in Table 3.7. No puncturing was applied to the encoded bits, which were passed to a random channel interleaver having a depth of 20000 bits. Figure 3.16(a) showed the turbo equalisation performance of the convolutional-coded GMSK system transmitting over a non-dispersive Gaussian channel after one, two, four and eight turbo equalisation iterations. Note that the convolutional decoding performance was equivalent to the first turbo equalisation iteration. It was observed that the gain obtained from joint equalisation and decoding after eight turbo equalisation iterations was approximately 3.7 dB as compared to independent equalisation and decoding at BER $= 10^{-4}$. As for the turbo equalisation performance over the five-path Rayleigh fading channel, it can be seen from Figure 3.16(b) that the gain achieved by using turbo equalisation after eight turbo equalisation iterations was 4.2 dB, compared to conventional convolutional decoding at BER $= 10^{-4}$. The turbo equalisation performance was observed to improve significantly by employing more turbo equalisation iterations. At BER $=10^{-4}$ and when transmitting over the non-dispersive Gaussian channel, iteration gains — *i.e.* gains in SNR performance with respect to the first iteration — of approximately 2.5 dB, 3.5 dB and 3.7 dB were obtained after two, four and eight turbo equalisation iterations, as displayed in Table 3.8. Over the five-path Rayleigh fading channel using burst-invariant fading the iteration gains, also summarised in this table, achieved were 2.9 dB, 3.9 dB and 4.2 dB at BER $=10^{-4}$ after two, four and eight turbo equalisation iterations. Let us now consider a convolutional-coding based turbo-coded GMSK system.

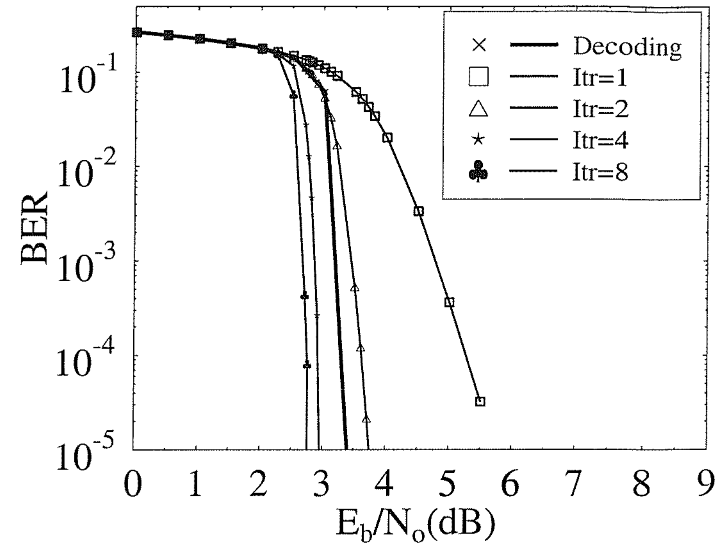


(a) Non-dispersive Gaussian channel.

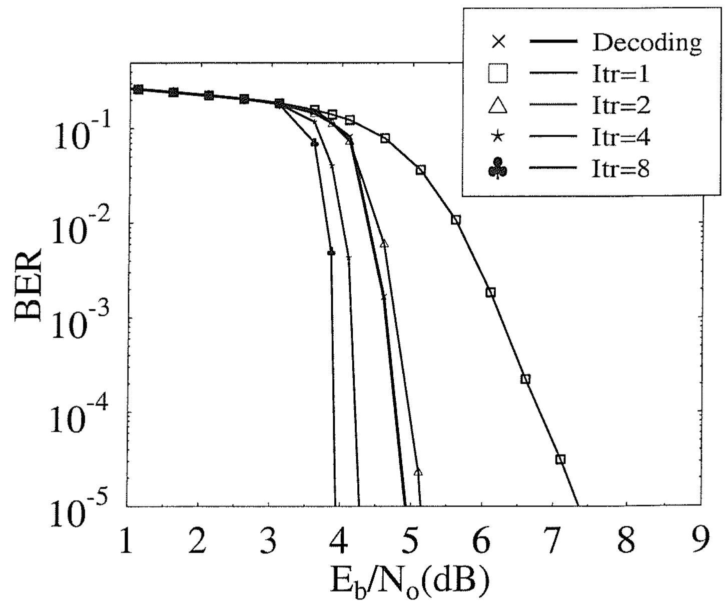


(b) Five-path Rayleigh fading channel.

Figure 3.16: Turbo equalisation performance over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading for a rate $R = 0.5$ **convolutional-coded** GMSK system using a turbo equaliser, which performs a maximum of eight turbo equalisation iterations.



(a) Non-dispersive Gaussian channel.



(b) Five-path Rayleigh fading channel.

Figure 3.17: Turbo equalisation performance over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading for a rate $R = 0.5$ **convolutional-coding based turbo-coded** GMSK system employing a turbo equaliser, which performs eight turbo equalisation iterations. The turbo decoding performance after eight turbo decoding iterations is also presented.

Code Rate $R = 0.5$
C1: 1 0 C2: 0 1
1 = transmitted bit 0 = non transmitted bit

Table 3.9: Regular puncturing pattern used in order to obtain the $R = 0.5$ convolutional-coding based turbo codes. The terms C1 and C2 represent the parity bits of the $R = 0.5$ convolutional codes of the first and second constituent codes, respectively.

Non-dispersive Gaussian channel		5-path Rayleigh fading channel	
Iteration index	Iteration gain	Iteration index	Iteration gain
2	1.6 dB	2	1.7 dB
4	2.2 dB	4	2.5 dB
8	2.4 dB	8	2.7 dB

Table 3.10: The iteration gains relative to the first iteration at BER $= 10^{-4}$ for the $R = 0.5$ convolutional-coding based turbo-coded GMSK system employing turbo equalisation over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading.

3.7.2 Convolutional-coding Based Turbo-coded GMSK System

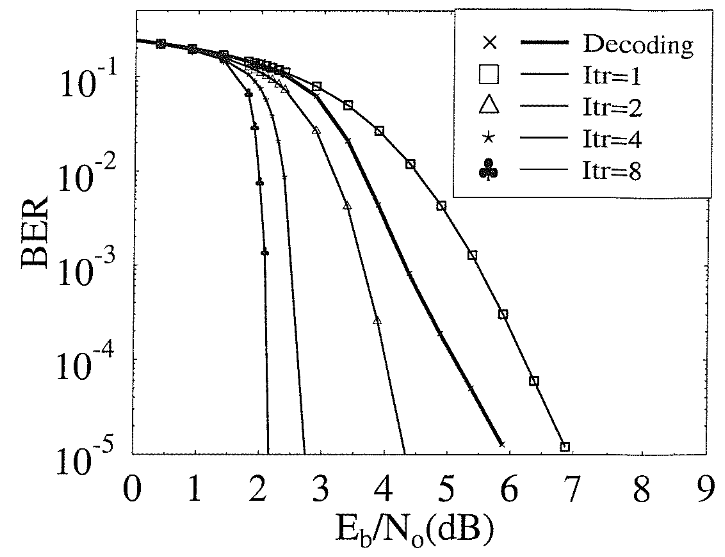
The convolutional-coding based turbo encoder consists of two rate $R = 0.5$, constraint length $K = 5$ recursive, systematic convolutional encoders, using octal generator polynomials of $G_0 = 35$ and $G_1 = 23$ as summarised in Table 3.7. A random turbo interleaver possessing a depth of 10000 bits separated the component encoders. The encoded bits were punctured regularly using the puncturing pattern of Table 3.9 in order to obtain an overall code rate of $R = 0.5$. Subsequently, the punctured encoded bits were passed to a random channel interleaver, which had a depth of 20000 bits. The turbo equalisation performance of the convolutional-coding based turbo-coded GMSK system transmitting over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel were shown in Figures 3.17(a) and 3.17(b), respectively. In the non-dispersive Gaussian channel scenario it was observed that the joint equalisation and decoding arrangement using eight turbo equalisation iterations outperformed the system employing eight turbo decoding iterations by approximately 0.5 dB at BER $= 10^{-4}$. For the five-path Rayleigh fading channel shown in Figure 3.17(b), the gain achieved by performing turbo equalisation after eight turbo equalisation iterations over the independent equalisation and turbo decoding employing eight turbo decoding iterations was 0.8 dB at BER $= 10^{-4}$. As in the convolutional-coded GMSK system, the turbo equalisation performance improved by performing more turbo equalisation iterations. When transmitting over the non-dispersive Gaussian channel, iteration gains of approximately 1.6 dB, 2.2 dB and 2.4 dB were achieved after two, four and eight turbo equalisation iterations, as displayed in Table 3.10. Over the five-path Rayleigh fading channel using burst-invariant fading, the iteration gains — also summarised in this table — achieved were 1.7 dB, 2.5 dB and 2.7 dB at BER $=10^{-4}$ after two, four and eight turbo equalisation iterations. Again — as expected — including the equaliser in the iterative loop was more complex but resulted in further performance gains. Let us now consider the performance of the BCH turbo-coded systems in the context of turbo equalisation.

3.7.3 BCH-coding Based Turbo-coded GMSK System

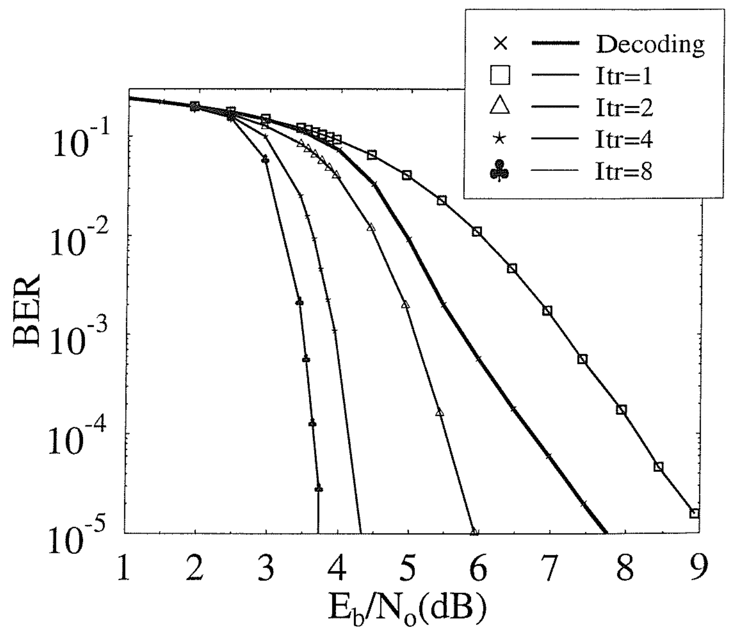
Non-dispersive Gaussian channel		5-path Rayleigh fading channel	
Iteration index	Iteration gain	Iteration index	Iteration gain
2	2.0 dB	2	2.5 dB
4	3.8 dB	4	4.0 dB
8	4.3 dB	8	4.5 dB

Table 3.11: The iteration gains relative to the first iteration at BER $= 10^{-4}$ for the $R = \frac{11}{19} = 0.58$ BCH-coding based turbo-coded GMSK system employing turbo equalisation over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading.

The BCH-coding based turbo encoders invoked consisted of two BCH (15,11) encoders — characterised in Table 3.7 — which were separated by a random turbo interleaver, having a depth of 12100 bits. No puncturing was applied, yielding an overall code rate of $R = \frac{11}{19} = 0.58$. The encoded bits were directed to a random channel interleaver possessing a depth of 20900 bits. When the BCH-coding based turbo-coded GMSK system was transmitted over the non-dispersive Gaussian channel, it was observed from Figure 3.18(a) that by performing the equalisation and decoding jointly — *i.e.* upon invoking turbo equalisation — and using eight turbo equalisation iterations, a gain of 2.8 dB was achieved over independent equalisation and BCH-coding based turbo decoding performing eight turbo decoding iterations at BER $= 10^{-4}$. The advantage of using turbo equalisation over turbo decoding was also highlighted in Figure 3.18(b). Here, it was observed that a gain of 3.0 dB was attained over the turbo decoding scheme employing eight turbo decoding iterations. The BER performance of the system employing turbo equalisation was also improved upon subsequent iterations. Specifically, in Figure 3.18(a) and at BER $= 10^{-4}$, the iteration gains obtained were 2 dB, 3.8 dB and 4.3 dB — as shown in Table 3.11 — when transmitting over the non-dispersive Gaussian channel. For the five-path fading channel scenario using



(a) Non-dispersive Gaussian channel.



(b) Five-path Rayleigh fading channel.

Figure 3.18: Turbo equalisation performance over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading for a rate $R = 0.58$ BCH(15,11)-based turbo-coded GMSK system using a turbo equaliser, which performs eight turbo equalisation iterations. The turbo decoding performance after eight turbo decoding iterations is also plotted.

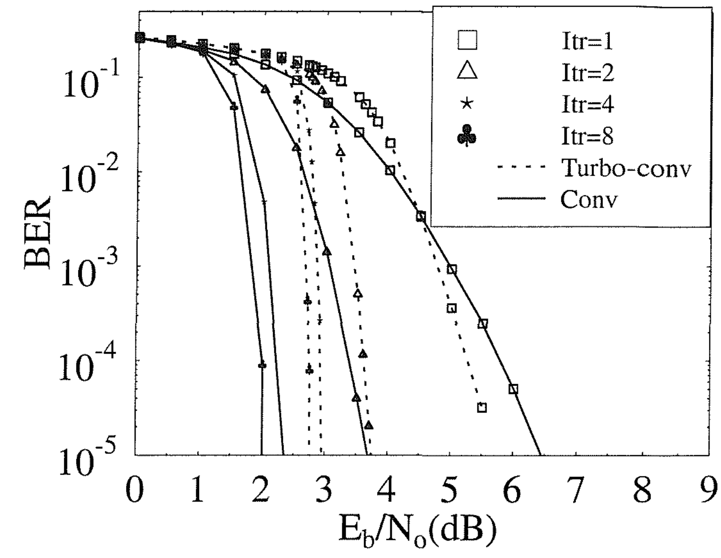
burst-invariant fading, it can be seen from Figure 3.18(b) and Table 3.11 that the iteration gains achieved were 2.5 dB, 4 dB and 4.5 dB at BER $= 10^{-4}$.

3.8 Discussion of Results

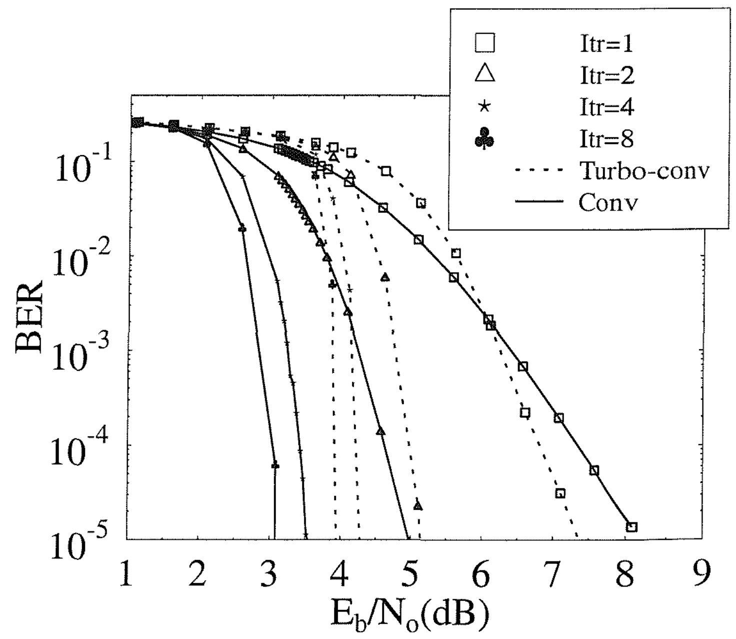
The key observations from the results in Subsections 3.7.1, 3.7.2 and 3.7.3 were:

- The advantage of performing turbo equalisation over turbo decoding.
- Improvement with subsequent turbo equalisation iterations.
- The rate $R = 0.5$ convolutional-coded GMSK system employing turbo equalisation achieved better performance than that of the $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme.
- The $R = 0.58$ BCH-coding based turbo-coded GMSK system employing turbo equalisation achieved better performance than that of the lower rate $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme.

It was observed that significant advantage was gained by performing the equalisation and decoding jointly. Upon each turbo equalisation iteration the equaliser benefits from the *a priori* information of the encoded bits obtained from the decoder(s). Consequently, more reliable LLR values can be generated by the equaliser, hence aiding the decoder in improving the confidence of the LLR values. In contrast, by performing independent equalisation and decoding, the decoder(s) can only operate on the initial soft values from the equaliser. Therefore, the equaliser was unable to exploit the improved decoder estimates for enhancing the reliability of the equaliser's soft values. Hence, the joint equalisation and decoding technique outperformed the independent equalisation and decoding scheme. The disadvantage of turbo equalisation was the increased complexity due to the need of performing equalisation and decoding in each turbo equalisation iteration. For the convolutional-coded GMSK scheme, each additional iteration involved the equalisation and decoding process, hence increasing the complexity significantly, as compared to the conventional convolutional decoding technique. Upon comparing turbo decoding with turbo equalisation, it was observed that turbo decoding iterations involved exchange of information between the decoders, whereas in turbo equalisation information was exchanged not only between the decoders, but also with the equaliser in each turbo equalisation iteration. Hence, the complexity of the turbo equalisation scheme was also higher than that of turbo decoding. However, the number of iterations can be reduced by employing various termination criteria [65], similar to those used in turbo decoding [47]. This will reduce the complexity associated with turbo equalisation, making this technique



(a) Non-dispersive Gaussian channel.



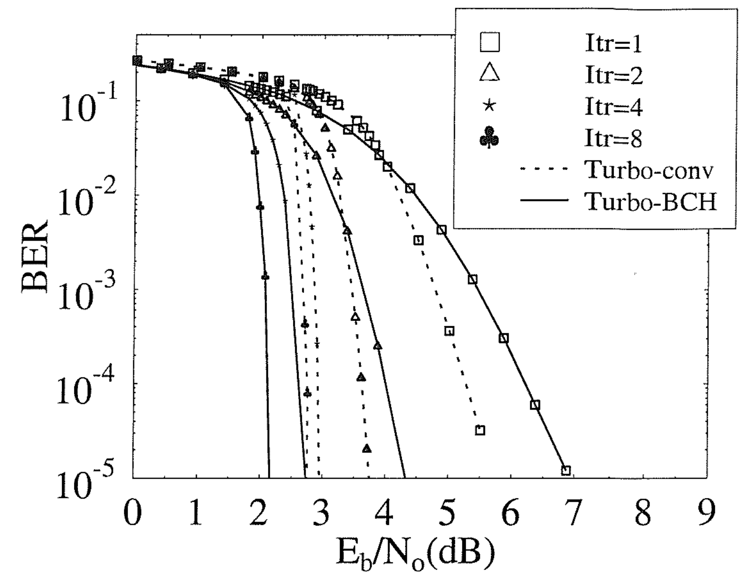
(b) Five-path Rayleigh fading channel.

Figure 3.19: Comparison of the $R = 0.5$ convolutional-coded GMSK system with $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme, transmitting over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading. Both systems employ turbo equalisation, which performs eight turbo equalisation iteration at the receiver.

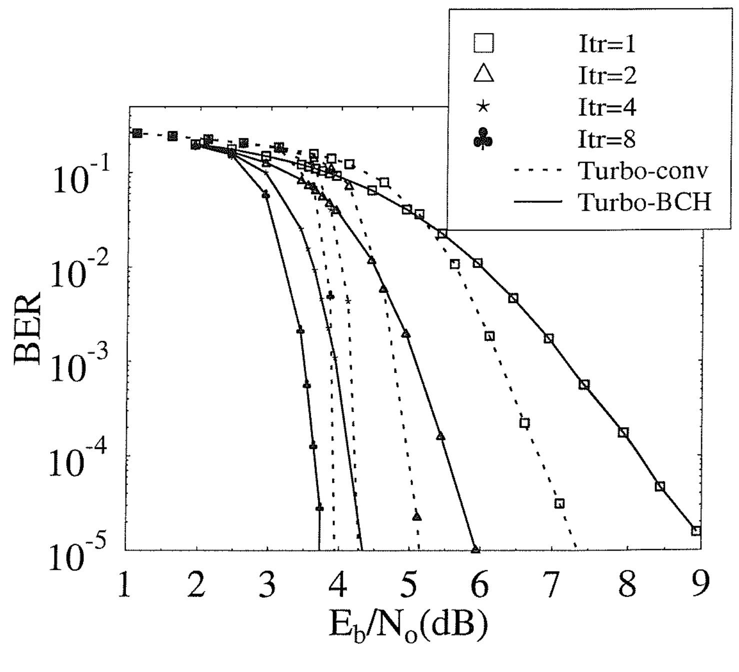
a good alternative for mitigating the channel's frequency selectivity. Furthermore, the Log-MAP algorithm used for the equaliser and decoder can be substituted by lower complexity SISO algorithms [73, 74, 75, 56], in order to practically realise the turbo equaliser. Recent work by Reeve [76] has demonstrated that the Log-MAP algorithm can be readily parallelised in a form suitable for parallel computing. This will speed up the computations associated with the complex Log-MAP algorithm, hence enabling the turbo equaliser — consisting of the Log-MAP equaliser and decoder — to be implemented in real-time systems.

As mentioned previously, by performing more iterations, the equaliser and decoder are capable of enhancing the reliability of their estimates. Therefore, the BER performance was improved as the number of turbo equalisation iterations were increased. However, as observed from the results of Subsections 3.7.1, 3.7.2 and 3.7.3, the rate of increase of the iteration gain, which was again the gain in SNR performance relative to the performance achieved in the first turbo equalisation iteration, decreased with successive turbo equalisation iterations. Iteration gains were obtained, when the decoder was capable of correcting errors previously inflicted by the equaliser. As the number of turbo equalisation iterations increased, the deinterleaved output of the equaliser became more similar to the decoder output. This therefore, reduced the advantage of having information from the equaliser and decoder and consequently failed to achieve greater iteration gains, despite performing further turbo equalisation iterations.

Another interesting observation was the ability of the convolutional-coded GMSK system employing turbo equalisation to outperform the convolutional-coding based turbo-coded GMSK scheme, as demonstrated by Figure 3.19. Over the non-dispersive Gaussian channel, the convolutional-coded GMSK system had a better E_b/N_o performance, than the convolutional-coding based turbo-coded GMSK scheme by 0.8 dB at BER $= 10^{-4}$, while in the five-path Rayleigh fading scenario using burst-invariant fading the advantage of the convolutional-coded system over the convolutional-coding based turbo-coded scheme was approximately 1.0 dB, as illustrated in Figures 3.19(a) and 3.19(b), respectively. These results were surprising since the more complex turbo-coded system was expected to form a more powerful encoded system compared to the convolutional-coded scheme. Below, we offer a possible explanation, which considered the performance of both codes after the first turbo equalisation iteration. Over the non-dispersive Gaussian channel in Figure 3.19(a), the convolutional-coded scheme yielded a lower BER than that of the turbo-coded system at E_b/N_o values less than 4.5 dB, indicating that the *a posteriori* LLR of the source bits — which constitute the first coded bit of the systematic codeword — produced by the convolutional decoder had a higher reliability than that of the turbo-coded scheme.



(a) Non-dispersive Gaussian channel.



(b) Five-path Rayleigh fading channel.

Figure 3.20: Comparison of the rate $R = 0.58$ BCH-coding based turbo-coded GMSK system with the $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme, transmitting over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading. Both systems employ turbo equalisation, which performs eight turbo equalisation iteration at the receiver.

Consequently, upon receiving the higher-confidence LLR values from the decoder, the equaliser in the convolutional-coded scheme was able to produce much more reliable LLR values in the subsequent turbo equalisation iteration, as compared to the turbo-coded system. After receiving these LLR values, the decoder of the convolutional-coded system will also generate more reliable LLR values. Similarly, for the five-path Rayleigh fading channel scenario in Figure 3.19(b), it was also observed that for $E_b/N_o < 6$ dB and after one turbo equalisation iteration, the convolutional-coded system outperformed the turbo-coded scheme. Similarly to the non-dispersive Gaussian channel scenario, the convolutional-coded system transmitting over the dispersive fading channel also produced LLR values of higher reliability as compared to those generated by the turbo-coded system, hence, allowing the convolutional-coded system to outperform the turbo-coded scheme after performing eight turbo equalisation iterations. For E_b/N_o values beyond the previously mentioned values, the performance of the convolutional-coding based turbo-coded scheme after one turbo equalisation iteration was better than that of the convolutional-coded system. Therefore, it was predicted that although the performance of the turbo-coded system was poorer than the convolutional-coded system after eight turbo equalisation iterations for $E_b/N_o < 4.5$ dB for the non-dispersive Gaussian channel and $E_b/N_o < 6.0$ dB for the five-path Rayleigh fading channel, it will outperform the convolutional-coded system beyond these E_b/N_o values, potentially yielding a lower error floor.

Finally, in Figure 3.20 the rate $R = 0.58$ BCH-coding based turbo-coded system was also observed to outperform the $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme transmitting over the non-dispersive Gaussian channel and over the five-path Rayleigh fading channel by a margin of 0.7 dB and 0.2 dB, respectively, at BER $= 10^{-4}$, when using eight turbo equalisation iterations. Here, the performance improvements achieved by the BCH-coding based turbo-coded system over the convolutional-coding based turbo-coded scheme can also be attributed to the fact that the former attained a lower BER for E_b/N_o values less than 4.0 dB and 5.5 dB over the non-dispersive channel and the five-path Rayleigh fading channel using burst-invariant fading, respectively. Therefore, the equaliser in the BCH-coding based turbo-coded system received more reliable LLR information compared to the convolutional-coding based turbo-coded scheme, hence allowing it to achieve better BER performance upon invoking successive turbo equalisation iterations.

In order to further justify these results, the associated Maximum Likelihood (ML) performance bound of coded systems was derived and compared in our forthcoming chapter.

3.9 Summary

In this chapter the principles of turbo equalisation in the context of multiple encoder assisted systems, were described. Here, the advantage of performing the channel equalisation and channel decoding jointly was examined and quantified in the context of GMSK-modulated systems, as compared to implementing the equalisation independently from the decoding. Gains of 3.7 dB and 4.2 dB were achieved for the rate $R = \frac{1}{2}$ convolutional-coded GMSK system transmitting over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading, respectively. Similarly, for the rate $R = \frac{1}{2}$ convolutional-coding based turbo-coded scheme the corresponding gains achieved through turbo equalisation over independent equalisation and turbo decoding were 0.5 dB and 0.8 dB over the non-dispersive Gaussian channel and the above-mentioned dispersive Rayleigh fading channel. Finally, the BCH-coding based turbo-coded systems employing turbo equalisation and transmitting over the non-dispersive Gaussian and the five-path fading channel exhibited gains of 2.8 dB and 3 dB, respectively. It was also observed that the rate $R = 0.5$ convolutional-coded GMSK system employing turbo equalisation obtained a better performance, than that of the $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme. Over the non-dispersive Gaussian channel, the convolutional-coded GMSK system outperformed the convolutional-coding based turbo-coded GMSK scheme by 0.8 dB at BER $= 10^{-4}$, while in the five-path Rayleigh fading scenario the convolutional-coded system attained a gain of approximately 1.0 dB over the convolutional-coding based turbo-coded scheme, as illustrated in Figures 3.19(a) and 3.19(b), respectively. These results were surprising, since the more complex turbo-coded system was expected to perform better generally, as it was deemed to be a more powerful code compared to convolutional codes. However, from Figures 3.19(a) and 3.19(b), which compared the turbo equalisation performance of the convolutional-coding based turbo-coded system and the conventional convolutional-coded scheme over the non-dispersive Gaussian channel and five-path Rayleigh fading channel, it was observed that at E_b/N_o values less than 4.5 dB and 6 dB, respectively, the BER of the rate $R = 0.5$ convolutional-coded GMSK system was lower than that of the $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme after the first turbo equalisation iteration. This indicated that the decoder in the convolutional-coded system mentioned above was providing more reliable LLR values to the equaliser. Consequently, the equaliser in the convolutional-coded scheme — upon receiving the higher confidence LLR values from the decoder — was capable of producing more reliable LLR values, which was subsequently passed to the decoder in the following turbo equalisation iteration. Beyond the E_b/N_o values of 4.5 dB and 6 dB over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel, respectively, the turbo-coded scheme exhibited a better performance after one turbo equalisation iteration. Therefore, it is predicted that the convolutional-coding based turbo-coded GMSK system will potentially yield a lower error floor. In Figure 3.20 it was also observed that the weaker BCH-coding based turbo-coded GMSK system yielded better performance, than that of the convolutional-coding based turbo coded scheme after eight turbo equalisation iterations. The performance of the BCH-coding based turbo-coded system was better, than that of the convolutional-coding based turbo-coded scheme after the first turbo equalisation iteration at E_b/N_o values less than 4.0 dB and 5.5 dB for transmissions over the non-dispersive Gaussian channel and over the five-path Rayleigh fading channel, respectively. Therefore, the equaliser in the BCH-based turbo-coded system received more reliable LLR information compared to the convolutional-based turbo-coded scheme, hence enabling it to achieve a better BER performance after invoking successive turbo equalisation iterations.

Chapter 4

Turbo Equalisation Performance Bound

Since the invention of turbo codes [9, 44], which have been shown to approach Shannonian performance limits, much research has been devoted to improving the overall performance by optimising the turbo code components, such as the constituent codes [77, 78, 79, 80], interleavers [81, 82, 83] and their decoding techniques [9, 84], either independently or jointly. Berrou *et al* proposed an iterative decoding algorithm instead of the Maximum Likelihood (ML) decoder [84], in order to reduce the decoder's complexity. Since the maximum likelihood decoder can only be implemented for specific and rather specific interleavers [84, 85] at the cost of a high complexity, theoretical performance bounds were derived in order to observe the ability of the iterative decoder to approximate the performance of the optimal maximum likelihood decoder. The theoretical bounds of the Parallel Concatenated Convolutional Code (PCCC) [20], namely turbo code, can be derived by employing the union bound technique in conjunction with Benedetto's and Montorsi's uniform interleaver [20]. This uniform interleaver is a probabilistic interleaver model, which maps the input codeword of weight w into all possible distinct permutations with equal probability. Therefore, the associated theoretical bound characterises the upper bound performance of turbo codes in conjunction with maximum-likelihood decoding averaged over all possible interleaver structures. The performance bound of Serial Concatenated Convolutional Codes (SCCC) can also be determined using this approach [19].

4.1 Motivation

In this chapter, the principles employed for deriving the ML performance bound for the SCCC and PCCC schemes are adapted, in order to derive the ML bound for non-punctured convolutional-coded and for turbo-coded systems. Previous work in this area by Narayanan and Stüber in reference [71] employed the analysis by Benedetto *et al* [19], which emphasised the importance of possessing recursive properties for the inner encoder — constituted for example by the partial-response GMSK modem — in SCCC schemes. This technique was utilised for determining the performance bound of the iterative demodulation and decoding of convolutional-coded interleaved systems in conjunction with differential modulation schemes, such as Differential Phase Shift Keying (DPSK) [71, 86] and $\frac{\pi}{4}$-Differential Quadrature Phase Shift Keying (DQPSK) [71]. Differential modulation techniques were employed, in order to show that the inherent recursive nature of these modulation schemes in conjunction with the interleaver significantly improves the distance spectrum of the transmitted signal, yielding good interleaving gains, quantified as the factor by which the bit error probability is decreased upon increasing the interleaver depth. Specifically, DPSK and DQPSK can be modeled as rate $R = 1$ convolutional codes followed by a memoryless mapper. This rate $R = 1$ recursive encoder can be viewed as an inner code in the SCCC scheme. Motivated by these research trends, we set out to employ the principles for PCCC schemes in conjunction with SCCC, in order to evaluate the ML performance bound of non-punctured turbo-coded systems employing recursive modulators. However, the main objective of the theoretical analysis is not to obtain the exact ML performance bound, but to characterise the turbo equalisation performance trends for turbo-coded systems. Explicitly, we aim to explain the turbo equalisation results obtained for turbo-coded GMSK and convolutional-coded GMSK systems, where the convolutional-coded scheme outperformed the turbo-coded system, despite using only a single decoder in each turbo equalisation iteration. Although the union bound diverges at the so-called cut-off rate [71] giving unreliable performance bounds, this bounding technique is still capable of demonstrating the performance trends associated with the turbo-equalised systems. For our investigations we have employed DPSK modulation, since it too is inherently recursive like GMSK, but lends itself to simpler analysis.

The organisation of this chapter is as follows. Sections 4.2 and 4.3 present an overview of the associated parallel concatenated convolutional coding and serial concatenated convolutional coding schemes, respectively. Subsequently, Section 4.4 describes the algorithm employed in determining the key parameter — namely the so-called Input Redundancy Weight Enumerating Function (IRWEF) — of an encoder. Section 4.5 models DPSK, MSK and GMSK as a recursive encoder, while Section 4.6 shows, how the SCCC and PCCC principles are adapted in order to determine the ML performance bound of convolutional-coded and turbo-coded systems. The results of the theoretical analysis and computer simulations are then presented in Section 4.7. This is followed by a discussion and a summary of the observations made in Section 4.8.

4.2 Parallel Concatenated Convolutional Code Analysis

This section describes the fundamental principles involved in determining the performance bound for Parallel Concatenated Convolutional Codes (PCCC). The parallel concatenated

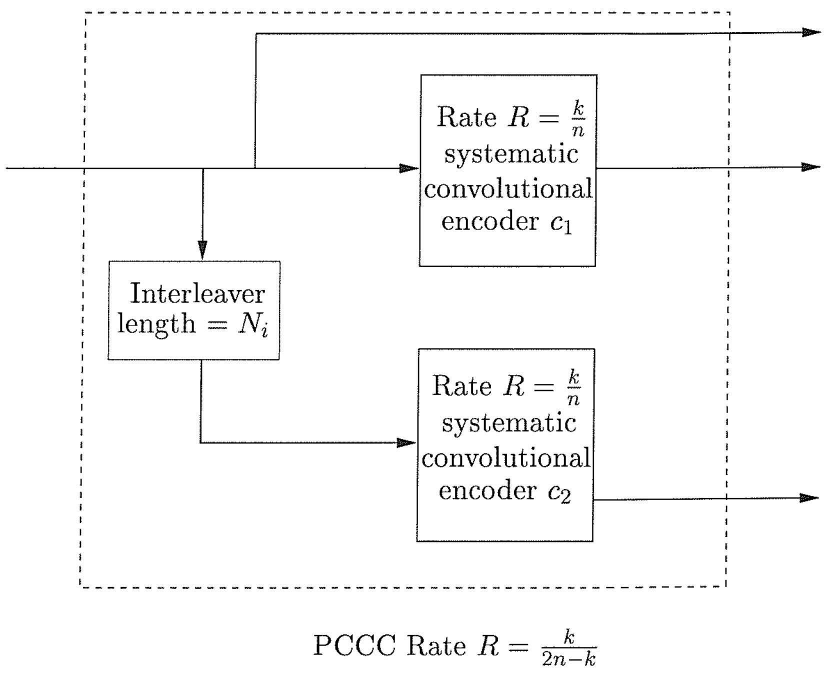

Rate $R = \frac{k}{n}$ systematic convolutional encoder c_1
Interleaver length $= N_i$
Rate $R = \frac{k}{n}$ systematic convolutional encoder c_2
PCCC Rate $R = \frac{k}{2n-k}$

Figure 4.1: Structure of the parallel concatenated code considered, consisting of encoders $c1$ and $c2$ and of the turbo interleaver of depth N_i bits.

convolutional codes illustrated in Figure 4.1 consist of two encoders, namely $c1$ and $c2$, which have a common input information sequence of length N_i bits and which are linked through an interleaver, so that the information sequence entered into the second encoder is the permuted version of the original input information sequence. Therefore, the weights of the input sequences of the first and second encoder are identical, although c1's and c2's parity weight will be different. In order to determine the performance bound of the PCCC, the constituent encoders are characterised by the so-called Input Redundancy Weight Enumerating Function (IRWEF) and the concept of uniform interleaving is introduced.

Essential PCCC notations from reference [20] are quoted and described, before elaborating on the proposed turbo equalisation analysis.

The first term introduced is the IRWEF represented by $A^C(W, Z)$, where W and Z are the dummy variables, whose exponent w and j represents the input weight and parity check weight, respectively. It can be expressed as a polynomial of W and Z:

$$A^C(W, Z) = \sum_{w,j} A_{w,j} W^w Z^j, \tag{4.1}$$

where $A_{w,j}$ is the number of codewords of parity weight j generated by the input sequence of weight w. The IRWEF quantifies explicitly the separate contributions of the input source bit information segment and that of the parity-check segment to the total Hamming weight of the codewords. Hence, the IRWEF characterises the entire encoder, since it depends on both the input information and parity bits. In order to further highlight the significance of the IRWEF, let us consider a BCH(7,4) block encoder, which accepts four input bits and produces a seven-bit codeword. Table 4.1 shows all the possible codewords, which consist

Codeword		Input	Codeword	Parity	Term in	Term in
Input bit	Parity bit	weight w	weight	weight j	Equation 4.2	Equation 4.12
0000	000	0	0	0	1	1
1000	101	1	3	2	2	2
0100	111	1	4	3	3	3
1100	010	2	3	1	4	4
0010	110	1	3	2	2	2
1010	011	2	4	2	5	5
0110	001	2	3	1	4	4
1110	100	3	4	1	7	7
0001	011	1	3	2	2	2
1001	110	2	4	2	5	5
0101	100	2	3	1	4	4
1101	001	3	4	1	7	7
0011	101	2	4	2	5	5
1011	000	3	3	0	6	6
0111	010	3	4	1	7	7
1111	111	4	7	3	8	8

Table 4.1: All possible BCH(7,4) codewords, which consist of the input word and parity word. The corresponding Hamming weights of the input word, parity word and codeword are also presented.

of the input word and the parity word. For example, it is observed that the input word 1100, whose Hamming weight — *i.e.* the number of 1's — is two, produces the parity word

$A_{w,j}$	$A_{0,0}$	$A_{1,2}$	$A_{1,3}$	$A_{2,1}$	$A_{2,2}$	$A_{3,0}$	$A_{3,1}$	$A_{4,3}$
	1	3	1	3	3	1	3	1

Table 4.2: The number of codewords of parity weight j generated by the input sequence of weight w for the BCH(7,4) code. These $A_{w,j}$ coefficients constitute the weighting factors in Equations 4.1 and 4.2.

010, hence giving the codeword 1100010 of Hamming weight three. Considering all input words and Equation 4.1, the IRWEF for the BCH(7,4) code becomes:

$$\begin{aligned} A^C(W,Z) &= 1W^0Z^0 + 3W^1Z^2 + W^1Z^3 + 3W^2Z^1 + 3W^2Z^2 + 1W^3Z^0 \\ &\quad + 3W^3Z^1 + 1W^4Z^3 \\ &= 1 + W(3Z^2 + Z^3) + W^2(3Z + 3Z^2) + W^3(1 + 3Z) + W^4Z^3. \end{aligned} \tag{4.2}$$

As it can be seen from Equation 4.2, each term in the IRWEF polynomial represents the input word weight and parity segment weight of the codewords, as well as the number of such Hamming-weight codes. For example, the second polynomial term on the right hand side of Equation 4.2, namely $W(3Z^2 + Z^3)$, shows that there are three codewords, which were generated by input bit segments having a Hamming weight one and resulting in parity weight of two, while there is only a single codeword caused by a Hamming-weight one input word and yielding a parity word of weight three. The last but one column of Table 4.1 indicates, which BCH(7,4) codeword contributes to which term in the top two lines of Equation 4.2. It is also noted that the sum of the $A_{w,j}$ coefficients is 16 in this example, which is the total number of codewords. The associated $A_{w,j}$ coefficients of Equation 4.2 are summarised in Table 4.2. Although we have considered block codes instead of convolutional codes in this example, we note that convolutional codes can be treated as block codes by adding all-zero tailing bits in order to terminate the encoder in the all-zero state.

Once the IRWEF of the code is determined, the bit error probability $P_b(e)$ can be evaluated by using the upper bound approximation in reference [20], which is given by:

$$P_b(e) \approx \frac{1}{2} \sum_{m=j+w} D_m \operatorname{erfc}\left(\sqrt{\frac{mRE_b}{N_o}}\right), \tag{4.3}$$

where m is the Hamming-weight of the codeword generated by the input word of weight w and the weight distribution D_m is defined as:

$$D_m \triangleq \sum_{m=j+w}^{n} \frac{w}{k} A_{w,j} \tag{4.4}$$

while k is the number of original information bits in the n-bit codeword. We will show at a later stage that Equation 4.3 can also be employed for PCCC and SCCC schemes.

Having defined the IRWEF and described, how the BER can be subsequently evaluated, we now introduce the Conditional Weight Enumerating Function (CWEF) of the parity check bits $A_w^C(Z)$ for the purpose of evaluating the performance bound of the PCCC scheme. The CWEF is the IRWEF of the encoder C corresponding to a particular input weight w. Referring to the previous BCH (7,4) example, the CWEF $A_w^C(Z)$ becomes:

$$\begin{aligned} A_0^C(Z) &= 1 & w &= 0 \\ A_1^C(Z) &= 3Z^2 + Z^3 & w &= 1 \\ A_2^C(Z) &= 3Z + 3Z^2 & w &= 2 \\ A_3^C(Z) &= 1 + 3Z & w &= 3 \\ A_4^C(Z) &= Z^3 & w &= 4. \end{aligned} \tag{4.5}$$

The derivation of the CWEF can be generalised as [20]:

$$A_w^C(Z) = \sum_j A_{w,j} Z^j, \tag{4.6}$$

where the corresponding $A_{w,j}$ coefficients can be extracted from both Table 4.1 or from Table 4.2 for the BCH(7,4) code.

Before showing the contribution of the CWEF to the evaluation of the performance bound, let us introduce a further abstract device known as the uniform interleaver. Benedetto's and Montorsi's [20] uniform interleaver of depth k_i is a conceptual probabilistic device, which maps a given input word of weight w to all possible distinct $\binom{k_i}{w}$ number of permutations of it with equal probability of $1/\binom{k_i}{w}$. As a consequence of the interleaving — between the first encoder $c1$ and the second $c2$ — the conditional weight enumerating function of the first encoder is independent from the second. Hence, the CWEF $A_w^{C_p}(Z)$ of the entire concatenated code C_p can be expressed as the product of two CWEFs functions of the constituent codes, which is given by:

$$A_w^{C_p}(Z) = \frac{A_w^{c1}(Z) \times A_w^{c2}(Z)}{\binom{k_i}{w}}, \tag{4.7}$$

which is normalised by — or averaged over — the number of possible permutations $\binom{k_i}{w}$. In order to highlight the principle of computing the bit error probability of a PCCC scheme, let us consider the parallel concatenated BCH (7,4) scheme employing the above abstract uniform interleaver with a depth of $k_i = 4$ bits, replacing the convolutional codec in Figure 4.1. Using Equation 4.7 and the CWEF of the BCH (7,4) code in Equation 4.5, the

CWEF of the entire code becomes:

$$
\begin{aligned}
A_0^{C_p}(Z) &= \frac{1 \times 1}{\binom{4}{0}} = 1 \\
A_1^{C_p}(Z) &= \frac{(3Z^2 + Z^3) \times (3Z^2 + Z^3)}{\binom{4}{1}} = \frac{9}{4}Z^4 + \frac{3}{2}Z^5 + \frac{1}{4}Z^6 \\
A_2^{C_p}(Z) &= \frac{(3Z + 3Z^2) \times (3Z + 3Z^2)}{\binom{4}{2}} = \frac{3}{2}Z^2 + 3Z^3 + \frac{3}{2}Z^4 \\
A_3^{C_p}(Z) &= \frac{(1 + 3Z) \times (1 + 3Z)}{\binom{4}{3}} = \frac{1}{4} + \frac{3}{2}Z + \frac{9}{4}Z^2 \\
A_4^{C_p}(Z) &= \frac{Z^3 \times Z^3}{\binom{4}{4}} = Z^6.
\end{aligned}
\tag{4.8}
$$

Since $A^{C_p}(W, Z) = \sum_w A_w^{C_p}(Z)W^w$, with the aid of Equation 4.8 the IRWEF of the overall parallel concatenated BCH (7,4) code can be expressed as:

$$
\begin{aligned}
A^{C_p}(W, Z) = 1 &+ W\left(\frac{9}{4}Z^4 + \frac{3}{2}Z^5 + \frac{1}{4}Z^6\right) \\
&+ W^2\left(\frac{3}{2}Z^2 + 3Z^3 + \frac{3}{2}Z^4\right) \\
&+ W^3\left(\frac{1}{4} + \frac{3}{2}Z + \frac{9}{4}Z^2\right) \\
&+ W^4\left(Z^6\right).
\end{aligned}
\tag{4.9}
$$

In possession of Equation 4.9 we can now tabulate the $A_{w,j}$ values of the parallel concatenated BCH(7,4)-based code, as seen in Table 4.3. This then allows us to determine the values

$A_{w,j}$	$A_{0,0}$	$A_{1,4}$	$A_{1,5}$	$A_{1,6}$	$A_{2,2}$	$A_{2,3}$	$A_{2,4}$	$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{4,6}$
	1	$\frac{9}{4}$	$\frac{3}{2}$	$\frac{1}{4}$	$\frac{3}{2}$	3	$\frac{3}{2}$	$\frac{1}{4}$	$\frac{3}{2}$	$\frac{9}{4}$	1

Table 4.3: The number of codewords of parity weight j generated by the input sequence of weight w for the BCH(7,4) turbo code extracted from Equation 4.9.

of the weight-distribution D_m in Equation 4.4, since we can obtain $A_{w,j}$ — *i.e.* the number of codewords having a parity word of weight j generated by the input sequence of weight w — from the IRWEF $A^{C_p}(W, Z)$. Recalling from Equation 4.4 that $D_m = \sum_{m=j+w}^{n} \frac{w}{k}A_{w,j}$ and using Table 4.3 for the BCH(7,4) based turbo code, we obtain:

$$
\begin{aligned}
D_0 &= D_1 = D_2 = 0 \\
D_3 &= \frac{3}{4}\cdot\frac{1}{4} = 0.1875 \\
D_4 &= \frac{2}{4}\cdot\frac{3}{2}+\frac{3}{4}\cdot\frac{3}{2} = 1.875 \\
D_5 &= \frac{1}{4}\cdot\frac{3}{2}+\frac{2}{4}\cdot 3+\frac{3}{4}\cdot\frac{9}{4} = 3.75 \\
D_6 &= \frac{1}{4}\cdot\frac{3}{2}+\frac{2}{4}\cdot\frac{3}{2} = 1.125 \\
D_7 &= \frac{1}{4}\cdot\frac{1}{4} = 0.0625 \\
D_8 &= D_9 = 0 \\
D_{10} &= \frac{4}{4}\cdot 1 = 1.
\end{aligned}
\tag{4.10}
$$

Consequently, with the aid of Equation 4.3 the bit error probability $P_b(e)$ is given by:

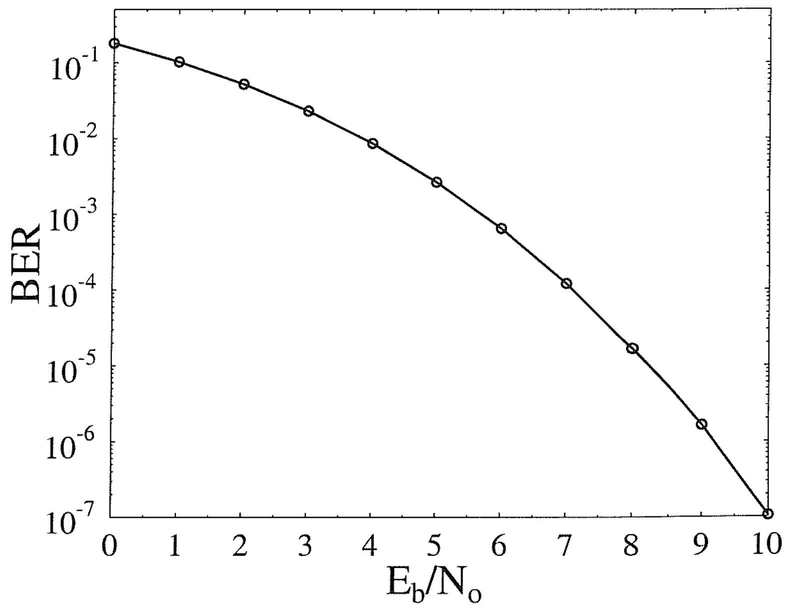


Figure 4.2: Theoretical bound of the parallel concatenated BCH (7,4) code, using a uniform interleaver and possessing an overall code rate of $R = \frac{4}{10} = 0.4$.

$$
\begin{aligned}
P_b(e) &\approx \frac{1}{2}\sum_{m=j+w} D_m \mathrm{erfc}\left(\sqrt{\frac{mRE_b}{N_o}}\right) \\
P_b(e) &\approx \frac{1}{2}\Big[0.1875 \ \mathrm{erfc}\left(\sqrt{\frac{3RE_b}{N_o}}\right) + 1.875 \ \mathrm{erfc}\left(\sqrt{\frac{4RE_b}{N_o}}\right) \\
&+ 3.75 \ \mathrm{erfc}\left(\sqrt{\frac{5RE_b}{N_o}}\right) + 1.125 \ \mathrm{erfc}\left(\sqrt{\frac{6RE_b}{N_o}}\right) \\
&+ 0.0625 \ \mathrm{erfc}\left(\sqrt{\frac{7RE_b}{N_o}}\right) + \ \mathrm{erfc}\left(\sqrt{\frac{10RE_b}{N_o}}\right)\Big],
\end{aligned}
\tag{4.11}
$$

which is plotted in Figure 4.2 for various E_b/N_o values, where the overall rate of the parallel concatenated BCH (7,4) code is $R = \frac{4}{10} = 0.4$.

The PCCC principles described previously can be adapted to evaluate the bit error probability of parallel concatenated convolutional codes, when the convolutional code is terminated in the all-zero state with the aid of tailing bits consisting of logical 0s.

In summary, the key parameters and principles, which were required for the evaluation of the bit error probability bound were the:

- Input Redundancy Weight Enumerating Function (IRWEF) of the constituent codes and entire code.
- Uniform interleaver.
- Conditional Weight Enumerating Function (CWEF) of the constituent codes.
- Union bound approximation [20].

The IRWEF of each constituent code is a function of the input word and the parity word weight. As seen in Equation 4.1, the IRWEF is a polynomial of the dummy variables W and Z, whose exponents are determined by the Hamming weight w of the input word and weight j of the parity word, respectively. The polynomial coefficients $A_{w,j}$ represent the number of codewords with these weights. Consequently, the CWEF, which is constituted by the IRWEF for a particular input weight w, can be evaluated. For block codes the codeword is of finite length, while convolutional coded codewords can be potentially infinitely long. However, when we employ terminated convolutional codes, *i.e.* convolutional codes using all-zero termination bits, these codes can be treated as block codes. In order to evaluate the performance bound of PCCCs for a specific interleaver, the permutation of the input bit sequence by the interleaver must be considered, in order to allow the parity word of the second encoder to be determined. Consequently, an exhaustive enumeration of all possible

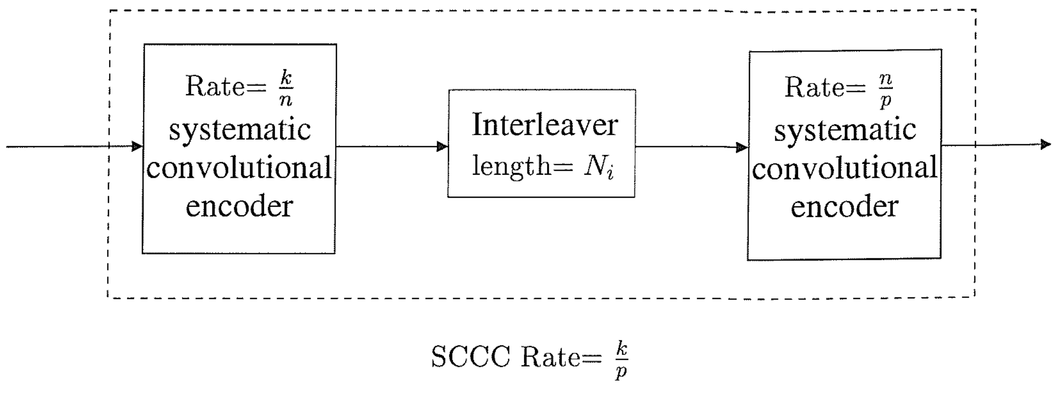


Figure 4.3: Serial concatenated convolutional code scheme consisting of a rate $\frac{k}{n}$ outer encoder serially cascaded with a rate $\frac{n}{p}$ inner encoder, which are separated by an interleaver.

cases must be performed. This high-complexity operation was circumvented by using the abstract concept of the uniform interleaver, which maps a given input into all possible distinct permutations of the input word with equal probability. Therefore, the CWEF of the overall PCCC code using the uniform interleaver can be evaluated as the product of the CWEF of the constituent codes and normalised by the number of possible distinct permutations. Subsequently, the bit error probability bound can be evaluated using the union bound relationship in Equation 4.3.

Having described the key notations and principles of union-bounding for the PCCC scheme, we now proceed to analyse SCCC schemes.

4.3 Serial Concatenated Convolutional Code Analysis

The Serial Concatenated Convolutional Code (SCCC) scheme illustrated in Figure 4.3 consists of a rate $\frac{k}{n}$ convolutional outer encoder serially cascaded with a rate $\frac{n}{p}$ convolutional inner encoder, separated by an interleaver of depth N_i.

Let the notations w, l, h represent the Hamming weights of the input sequence, the outer codeword and the inner codeword. Since the output of the outer encoder becomes the input of the inner encoder, the Hamming weight of the codeword is considered instead of the weight of the parity segment as in the PCCC analysis. Hence, the original concept of IRWEF is modified, in order to include the weight of the codewords, resulting in another function known as the Input Output Weight Enumerating Function (IOWEF) [19]. Using the BCH(7,4) example of Section 4.2 and observing the IRWEF expressed in Equation 4.2, which is repeated here for convenience,

$$A^C(W,Z) = 1 + W(3Z^2 + Z^3) + W^2(3Z + 3Z^2) + W^3(1 + 3Z) + W^4Z^3,$$

and with the aid of the last column in Table 4.1 the IOWEF of the BCH(7,4) code can be written as:

$$\begin{aligned} A^C(W,L) &= \sum_{w,l} A_{w,l}W^wL^l \\ A^C(W,L) &= 1W^0L^0 + 3W^1L^3 + W^1L^4 + 3W^2L^3 + 3W^2L^4 + 1W^3L^3 \\ &\quad + 3W^3L^4 + 1W^4L^7 \\ &= 1 + W(3L^3 + L^4) + W^2(3L^3 + 3L^4) + W^3(L^3 + 3L^4) + W^4L^7, \end{aligned} \tag{4.12}$$

where W and L are dummy variables, whose exponent, namely w and l, are used to represent the Hamming weights of the input word and codeword, respectively. Therefore — in contrast to the PCCC scheme of Section 4.2 — for the purpose of the SCCC analysis, all enumerations will consider the weight of the codeword, rather than that of the parity segment.

Codeword		Input	Codeword	Parity
Input bit	Parity bit	weight w	weight	weight j
000	0	0	0	0
100	1	1	2	1
010	1	1	2	1
110	0	2	2	0
001	1	1	2	1
101	0	2	2	0
011	0	2	2	0
111	1	3	4	1

Table 4.4: All possible (4,3) parity check codewords, which consist of the input word and parity word. The corresponding Hamming weights of the input word, parity word and codeword are also presented.

Having determined the IOWEF, the associated CWEF can be evaluated as in our previous PCCC analysis. Subsequently, since the SCCC scheme assumes that the interleaver permutations are of equal probability through the use of the uniform interleaver, we can obtain an expression for the IOWEF of the entire SCCC using the CWEF of the constituent codes. By using the IOWEF of the overall code, the corresponding bit error probability bound can be determined. As an example, let us consider a simple Serial Concatenated Block Code (SCBC), in order to highlight the key principles in determining the performance bound of the SCCC configuration concerned. Let us assume that the outer constituent encoder c_o is a (4,3) parity check code, while the inner encoder c_i employed is a BCH(7,4) code. The corresponding IOWEF of the outer encoder is derived by using Table 4.4, which details all the possible input words and the corresponding codewords for the (4,3) parity check code, yielding:

$$A^{c_o}(W,L) = 1 + W(3L^2) + W^2(3L^2) + W^3(L^4). \tag{4.13}$$

Furthermore, upon replacing W by L, as well as L by H in Equation 4.12, we arrive at the IOWEF of the BCH(7,4) inner code:

$$A^{c_i}(L,H) = 1 + L(3H^3 + H^4) + L^2(3H^3 + 3H^4) + L^3(H^3 + 3H^4) + L^4H^7, \tag{4.14}$$

where W, L and H are dummy variables, whose exponent represents the Hamming weights of the input word, that of the codeword of the outer encoder constituting the input to the inner encoder and that of the codeword of the inner encoder, respectively. As mentioned in Section 4.2, the CWEF is essentially the IRWEF of the constituent code for a particular weight. However, in the serial concatenated code analysis the CWEF is the IOWEF conditioned upon the Hamming weight of a word, such as the Hamming weight of the outer encoder codeword. For example, $A_l^{c_o}(W)$ is the CWEF of the outer encoder c_o obtained from the IOWEF $A^{c_o}(W,L)$, which has been conditioned upon l — namely upon the Hamming weight of the codeword — whereas $A_l^{c_i}(H)$ is the CWEF of the inner encoder c_i, which is obtained by conditioning the IOWEF $A^{c_o}(L,H)$ upon the weight of the input word l, which constitutes also the outer encoder's codeword. Continuing with the SCBC example described previously, the corresponding values of $A_l^{c_o}(W)$ employing the (4,3) parity check code are extracted from Equation 4.13, yielding:

$$\begin{aligned}
A_{l=0}^{c_o}(W) &= 1\\
A_{l=1}^{c_o}(W) &= 0\\
A_{l=2}^{c_o}(W) &= 3W + 3W^2\\
A_{l=3}^{c_o}(W) &= 0\\
A_{l=4}^{c_o}(W) &= W^3,
\end{aligned} \tag{4.15}$$

while the CWEFs $A_l^{c_i}(H)$ of the BCH(7,4) code are inferred from Equation 4.14, giving:

$$\begin{aligned}
A_{l=0}^{c_i}(H) &= 1\\
A_{l=1}^{c_i}(H) &= 3H^3 + H^4\\
A_{l=2}^{c_i}(H) &= 3H^3 + 3H^4\\
A_{l=3}^{c_i}(H) &= H^3 + 3H^4\\
A_{l=4}^{c_i}(H) &= H^7.
\end{aligned} \tag{4.16}$$

Here, we have assumed uniform interleaving, *i.e.* each codeword of weight l is permuted at the output of the outer encoder into all of its possible distinct $\binom{k_i}{l}$ permutations, where k_i is the depth of the interleaver. Each codeword of the outer code c_o of weight l — after uniform interleaving — is then directed towards the inner encoder generating $\binom{k_i}{l}$ inner encoder codewords. In reference [19] it was shown that the IOWEF of the entire concatenated code $A^{C_s}(W,H)$ — *i.e.* that of the SCCC using uniform interleaver — can be expressed as the product of two CWEFs of the constituent codes, which is normalised by the number of possible permutations $\binom{k_i}{l}$:

$$A^{C_s}(W,H) = \sum_{l=0}^{k_i} \frac{A_l^{c_o}(W) \times A_l^{c_i}(H)}{\binom{k_i}{l}}, \tag{4.17}$$

Therefore, with the aid of Equations 4.15, 4.16 and 4.17 the IOWEF of the SCBC example using the (4,3) parity check code as the outer encoder and the BCH(7,4) as the inner encoder can be written as:

$$\begin{aligned} A^{C_s}(W,H) &= \sum_{l=0}^{k=4} \frac{A_l^{c_o}(W) \times A_l^{c_i}(H)}{\binom{k_i}{l}} \\ A^{C_s}(W,H) &= \frac{1 \times 1}{1} + \frac{0 \times (3H^3 + H^4)}{4} + \frac{(3W + 3W^2) \times (3H^3 + 3H^4)}{6} \\ &+ \frac{0 \times (H^3 + 3H^4)}{4} + \frac{W^3 \times H^7}{1} \\ &= 1 + W(1.5H^3 + 1.5H^4) + W^2(1.5H^3 + 1.5H^4) + W^3H^7, \end{aligned} \tag{4.18}$$

where the depth of the interleaver is $N_i = k_i = 4$. Recalling from Equation 4.4 that the

$A_{w,h}$	$A_{0,0}$	$A_{1,3}$	$A_{1,4}$	$A_{2,3}$	$A_{2,4}$	$A_{3,7}$
	1	1.5	1.5	1.5	1.5	1

Table 4.5: The number of SCBC codewords extracted from Equation 4.18.

weight distribution is $D_m = \sum_{m=j+w}^{n} \frac{w}{k} A_{w,j}$, where m — which is the sum of the input word weight w and the parity segment weight j — also represents the weight of the codeword and is denoted as h for the serial concatenated code configurations. Referring to Equation 4.18 and Table 4.5, it is observed that the possible codeword weights generated are $h = 3$, $h = 4$

and $h = 7$, hence giving $D_{m=h}$ values of:

$$\begin{aligned}
D_{h=0} &= D_{h=1} = D_{h=2} = 0 \\
D_{h=3} &= \frac{w=1}{k_i=4} \cdot A_{1,3} + \frac{w=2}{k_i=4} \cdot A_{2,3} = \frac{1}{4} \cdot 1.5 + \frac{2}{4} \cdot 1.5 = 1.125 \\
D_{h=4} &= \frac{w=1}{k_i=4} \cdot A_{1,4} + \frac{w=2}{k_i=4} \cdot A_{2,4} = \frac{1}{4} \cdot 1.5 + \frac{2}{4} \cdot 1.5 = 1.125 \\
D_{h=5} &= D_6 = 0 \\
D_{h=7} &= \frac{w=3}{k_i=4} \cdot A_{3,7} = \frac{3}{4} \cdot 1 = 0.75,
\end{aligned} \tag{4.19}$$

and with the aid of Equation 4.3, the bit error probability bound can be expressed as [20]:

$$\begin{aligned}
P_b(e) &\approx \frac{1}{2} \sum_{m=h} D_m \text{erfc}\left(\sqrt{\frac{mRE_b}{N_o}}\right) \\
P_b(e) &\approx \frac{1}{2}\left[1.125 \text{ erfc}\left(\sqrt{\frac{3RE_b}{N_o}}\right) + 1.125 \text{ erfc}\left(\sqrt{\frac{4RE_b}{N_o}}\right) + 0.75 \text{ erfc}\left(\sqrt{\frac{7RE_b}{N_o}}\right)\right],
\end{aligned} \tag{4.20}$$

as plotted in Figure 4.4, where R is the overall SCBC code rate, which is equal to $\frac{3}{7}$ in our

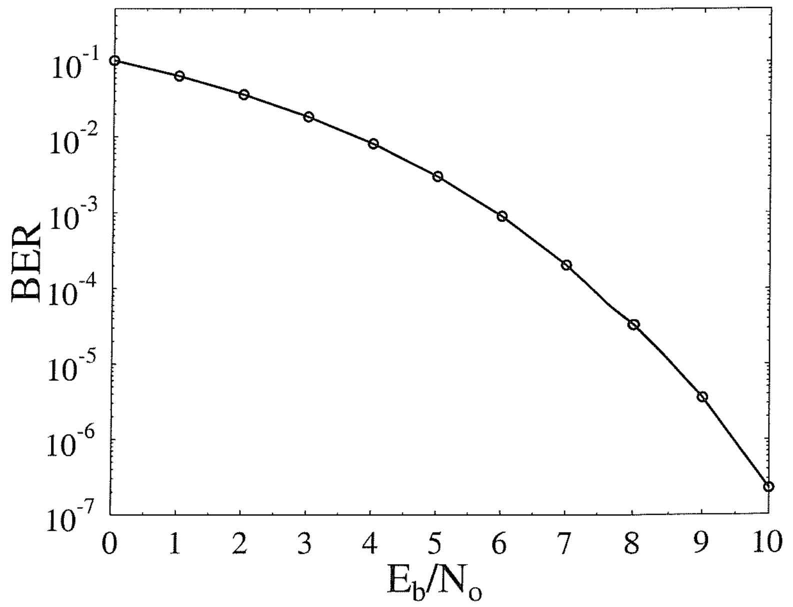


Figure 4.4: Theoretical performance bound of the serial concatenated code consisting of a (4,3) parity check code as the outer encoder and a BCH (7,4) code as the inner encoder.

example. As argued before, the same principles can also be applied for serial concatenated codes using convolutional codes as constituent codes provided that the code is terminated with all-zero tailing bits. More explicitly, this is because convolutional codes, which can potentially have infinitely long codewords, can be treated as block codes when all-zero tailing bits are inserted, such that the trellis terminates in the all-zero state.

In reference [19], Benedetto *et al.* showed that the key design criterion for SCCC schemes is that the inner encoder must possess recursive properties. This is crucial in order to obtain a large interleaving gain, which is defined as the factor by which the bit error probability is decreased with increasing interleaver length.

In summary, we have described how the Maximum Likelihood (ML) performance of the PCCC and the SCCC schemes can be approximated by utilising the constituent codes' IRWEF and IOWEF, respectively. Furthermore, the application of the abstract uniform interleaver also alleviated the computational complexity associated with the evaluation of the ML bound of the PCCC and SCCC schemes for a specific interleaver. In order to evaluate the IRWEF of the code, all possible input words and its corresponding parity words are determined. Subsequently, the individual contribution of the input word and parity word Hamming weight to the overall codeword Hamming weight was identified, in order to yield the IRWEF. Similarly, the IOWEF of a code is determined by considering the individual contribution of the input word and codeword Hamming weight. However, the number of input words increases exponentially as the number of bits forming the input word increases. In a PCCC or SCCC analysis convolutional codes are assumed to be terminated and therefore the length of the input word is equivalent to the depth of the turbo interleaver for PCCC schemes, or to the depth of the interleaver separating the inner encoder and outer encoder for SCCC schemes. Therefore, the longer the interleaver depth, the greater the complexity associated with the evaluation of the IRWEF and IOWEF. In order to overcome this complexity problem, we employed a trellis search algorithm based on the Viterbi algorithm [37, 72], which is the focus of our next discussion.

4.4 Enumerating the Weight Distribution of the Convolutional Code

This section discusses our advocated method employed in order to determine the IRWEF of convolutional codes. Recall that in Section 4.3 the IRWEF of the BCH(7,4) code was evaluated by identifying all the possible input words and the associated parity words. This approach can also be employed for convolutional codes. However, as the length of the input word increases, the number of possible sequences increases exponentially, hence rendering the task in general impractical. Alternatively, the enumeration of each input word's Hamming weight and its corresponding codeword Hamming weight can be performed through a recursive trellis search algorithm, in order to reduce the associated computational complexity. The search algorithm employed is similar to the Viterbi algorithm [37, 72], where a metric is updated recursively. Below, we present a simple example of the algorithm and subsequently we generalise the key points discussed, where possible.

Consider a rate $R = \frac{1}{2}$, constraint length $K = 2$, Recursive Systematic Convolutional (RSC) encoder, using octal generator polynomials of $G_0 = 3$ and $G_1 = 2$, which is depicted in Figure 4.5. Let the length of the input sequence be four bits. As it was shown in

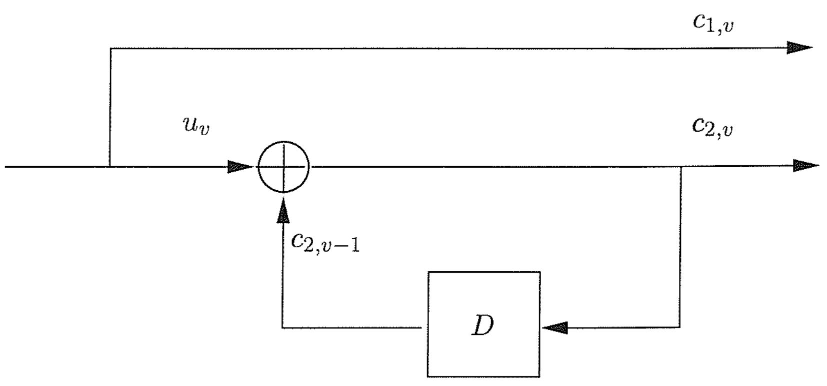


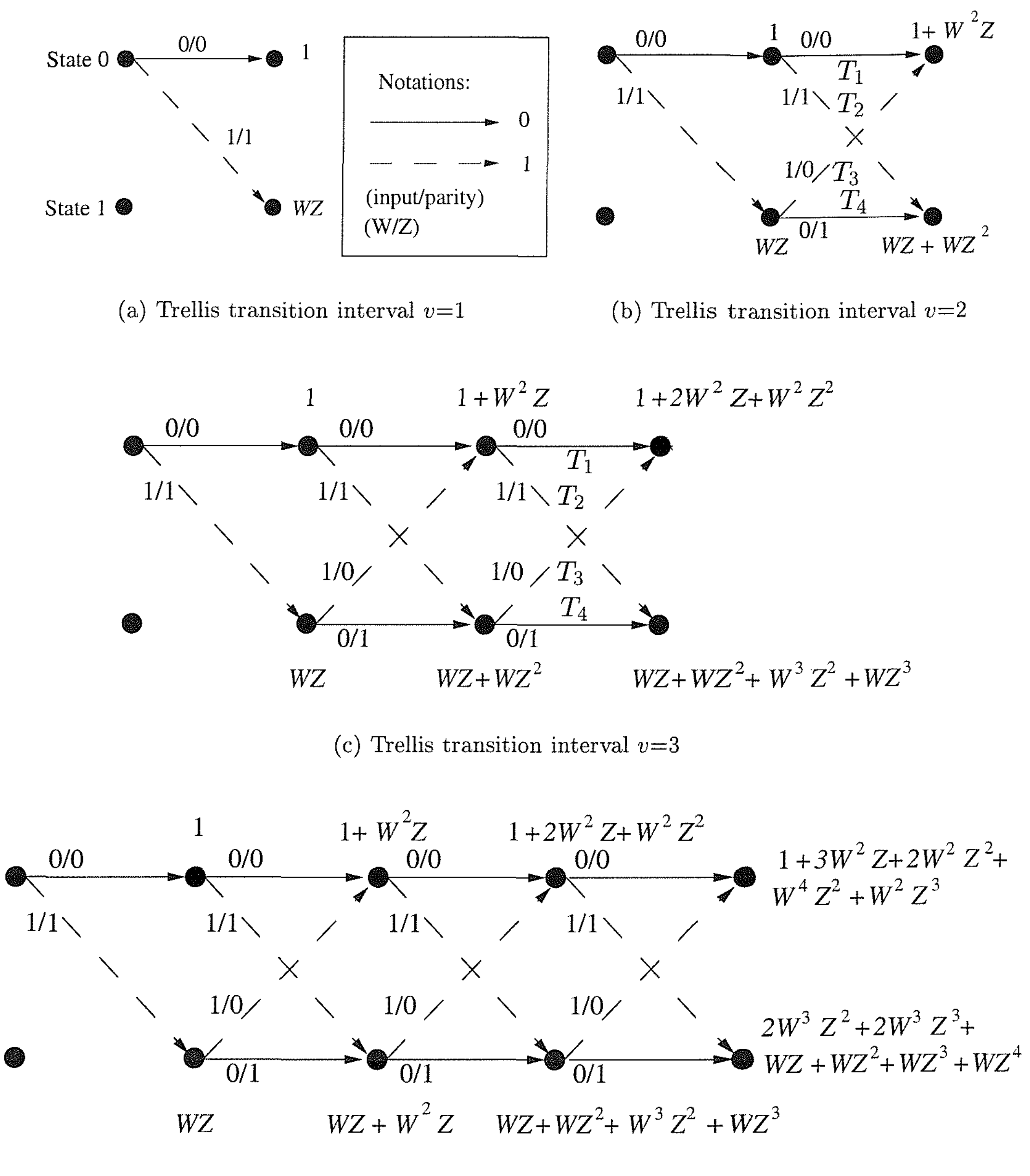
Figure 4.5: Schematic of the rate $R = \frac{1}{2}$, constraint length $K = 2$, recursive systematic convolutional code employing octal generator polynomials of $G_0 = 3$ and $G_1 = 2$. The notations D, u_v, $c_{1,v}$, $c_{2,v}$ and $c_{2,v-1}$ represent the symbol duration, the source bit at time instant v, the systematic bit of the codeword at instant v, parity bit of the codeword at instant v and the parity bit of the codeword at the previous time instant $v-1$, respectively.

Section 4.2, the IRWEF of the code can be determined by listing all possible input words and the corresponding parity words. Therefore, by considering all possible 4-bit input sequences entered into the rate $R = \frac{1}{2}$, $K = 2$ RSC encoder of Figure 4.5 we can determine their corresponding parity segments and codeword Hamming-weights, as summarised in Table 4.6. Using this approach and Table 4.6 it can be readily verified that the IRWEF and IOWEF of the code is:

$$\begin{aligned} A^C(W,Z) &= 1 + W(Z + Z^2 + Z^3 + Z^4) + W^2(3Z + 2Z^2 + Z^3) \\ &+ W^3(2Z^2 + 2Z^3) + W^4(2Z^2), \end{aligned} \tag{4.21}$$

and

$$\begin{aligned} A^C(W,H) &= 1 + W(H^2 + H^3 + H^4 + H^5) + W^2(3H^3 + 2H^4 + H^5) \\ &+ W^3(2H^5 + 2H^6) + W^4(2H^6), \end{aligned} \tag{4.22}$$



(a) Trellis transition interval v=1

(b) Trellis transition interval v=2

(c) Trellis transition interval v=3

(d) Trellis transition interval v=4

Figure 4.6: Trellis development and the accumulated Hamming weight of the input words and the parity words for each state of the $\frac{1}{2}$-rate, constraint length $K = 2$ recursive systematic convolutional encoder, using octal generator polynomials of $G_0 = 3$ and $G_1 = 2$ illustrated in Figure 4.5.

Input word	Input Weight	Parity Bit	Parity Weight	Codeword weight
0000	0	0000	0	0
1000	1	1111	4	5
0100	1	0111	3	4
1100	2	1000	1	3
0010	1	0011	2	3
1010	2	1100	2	4
0110	2	0100	1	3
1110	3	1011	3	6
0001	1	0001	1	2
1001	2	1110	3	5
0101	2	0110	2	4
1101	3	1001	2	5
0011	2	0010	1	3
1011	3	1101	3	6
0111	3	0101	2	5
1111	4	1010	2	6

Table 4.6: Input information, codeword and parity Hamming weight for rate $R = \frac{1}{2}$, constraint length $K = 2$ recursive systematic convolutional encoder, using octal generator polynomials of $G_0 = 3$ and $G_1 = 2$.

respectively. Alternatively, we can employ a trellis search algorithm that updates a weight metric recursively upon traversing through the trellis. Specifically, in this case the metric updated for each state is the input and parity word Hamming weight. From this, we can easily determine the Hamming weights of the codewords. Figure 4.6 illustrates the development of the trellis until the trellis transition interval of $v = 4$, which will be detailed in the next paragraph. At each interval, there are two main trellis operations. Firstly, the Hamming weight of the input and parity word is evaluated for each transition. This is analogous to the transition metric in the Viterbi algorithm. Secondly, the IRWEF associated with the previous state is updated after considering the Hamming weight of the corresponding transition, in order to yield the accumulated IRWEF of the new state. This too resembles the accumulated path-metric computed by the Viterbi algorithm. One major difference between this trellis search algorithm and the Viterbi algorithm is that during the determination of the IRWEF, no paths are discarded. This will be made more explicit in our forthcoming discussion.

At trellis interval $v = 1$ in Figure 4.6(a), the accumulated IRWEF at state 0 is initialised to $W^0Z^0 = 1$. Note that the first bit written on the transition is the input bit — which is the same as the systematic encoded bit —, while the second is the associated parity bit generated by the encoder. There are two branches leaving state 0, giving the corresponding weight contributions of $W^0Z^0 = 1$ and W^1Z^1 for input bits of 0 and 1, respectively. Therefore, at the next trellis stage in state 0 we have an IRWEF of $W^0Z^0 = 1$, while in state 1 the accumulated IRWEF is W^1Z^1. In trellis interval $v = 2$ in Figure 4.6(b) there are two transitions leaving both state 0 and state 1. Consider transitions T_1 and T_3, which merge into state 0. The corresponding Hamming weights of the input word and parity bit for transition T_1 is $w = 0$ and $j = 0$, while for T_3 it is $w = 1$ and $j = 0$. Therefore, we can express the IRWEF of the transition T_1 as $W^0Z^0 = 1$, while for transition T_3 the IRWEF is W^1Z^0. Consequently, the accumulated IRWEF of state 0 at interval $v = 2$ can be expressed as:

$$\begin{aligned} A^C(W,Z)_{\text{ of state 0 at } (v=2)} &= A^C(W,Z)_{\text{ of state 0 at } v=1} \times A^C(W,Z)_{\text{ of } T_1} \\ &+ A^C(W,Z)_{\text{ of state 1 at } (v=1)} \times A^C(W,Z)_{\text{ of } T_3} \\ &= 1 \times 1 + WZ \times W \\ &= 1 + W^2Z, \end{aligned} \tag{4.23}$$

as seen also in Figure 4.6(b). Similarly, the accumulated IRWEF of state 1 at interval $v = 2$ can be written as:

$$\begin{aligned} A^C(W,Z)_{\text{ of state 1 at } (v=2)} &= A^C(W,Z)_{\text{ of state 0 at } v=1} \times A^C(W,Z)_{\text{ of } T_2} \\ &+ A^C(W,Z)_{\text{ of state 1 at } (v=1)} \times A^C(W,Z)_{\text{ of } T_4} \\ &= 1 \times WZ + WZ \times Z \\ &= WZ + WZ^2, \end{aligned} \tag{4.24}$$

as portrayed in Figure 4.6(b). Note that — in contrast to the Viterbi algorithm — there is no selection of the winning path here and hence no paths are discarded. Instead, the Hamming weight contributions from the two merging paths contribute to the accumulated Hamming weight of the specific state reached as we have seen in Equations 4.23 and 4.24. Following the same reasoning, at trellis interval $v = 3$ state 0 accumulates the IRWEF $A^C(W,Z)$ of:

$$\begin{aligned} A^C(W,Z)_{\text{ of state 0 at } (v=3)} &= A^C(W,Z)_{\text{ of state 0 at } (v=2)} \times A^C(W,Z)_{\text{ of } T_1} \\ &+ A^C(W,Z)_{\text{ of state 1 at } (v=2)} \times A^C(W,Z)_{\text{ of } T_3} \\ &= (1 + W^2Z) \times 1 + (WZ + WZ^2) \times W \\ &= 1 + 2W^2Z + W^2Z^2, \end{aligned} \tag{4.25}$$

while $A^C(W,Z)$ for state 1 at $(v=3)$ is:

$$\begin{aligned} A^C(W,Z)_{\text{ of state 0 at } (v\,=\,3)} &= A^C(W,Z)_{\text{ of state 0 at } (v\,=\,2)} \times A^C(W,Z)_{\text{ of } T_2} \\ &+ A^C(W,Z)_{\text{ of state 1 at } (v\,=\,2)} \times A^C(W,Z)_{\text{ of } T_4} \\ &= (1+W^2Z)\times WZ + (WZ+WZ^2)\times Z \\ &= W(Z+Z^2+Z^3)+W^3Z^2. \end{aligned} \tag{4.26}$$

These functions can be also seen in Figure 4.6(c). We can generalise the expression of $A^C(W,Z)$ for state κ at trellis interval v as:

$$\begin{aligned} A^C(W,Z)_{\text{ of state } \kappa \text{ at interval } v} = \sum_{\forall \acute{\kappa}\to\kappa} \Big(& A^C(W,Z)_{\text{ of state } \acute{\kappa} \text{ at } v\,-\,1} \\ & \times A^C(W,Z)_{\text{ of transition } \acute{\kappa}\to\kappa} \Big), \end{aligned} \tag{4.27}$$

where $\forall \acute{\kappa} \to \kappa$ represents all paths from the previous state $\acute{\kappa}$ to the current state κ.

In Figure 4.6(d) the trellis ends and the sum of the accumulated IRWEF at states 0 and 1 represents the IRWEF of the entire code. This is because the IRWEF accumulated at state 0 indicates the weight of all the possible codes, which started from state 0 at trellis interval $v=1$ and reached state 0 at trellis interval $v=4$, while the IRWEF at state 1 encompasses all the codes that emerged from state 0 at $v=1$ and arrived at state 1 at interval $v=4$. Furthermore, since there are only two possible states in the trellis, the sum of the IRWEF in these states represents all the possible input words and codewords. Therefore, we infer from Figure 4.6(d) that the IRWEF of the entire code $A^C(W,Z)$ is:

$$\begin{aligned} A^C(W,Z) = 1 &+ W(Z+Z^2+Z^3+Z^4) + W^2(3Z+2Z^2+Z^3) \\ &+ W^3(2Z^2+2Z^3)+W^4(Z^2), \end{aligned} \tag{4.28}$$

which is identical to the IRWEF in Equation 4.21 that was obtained from Table 4.6 by generating all possible input words. The advantage of employing the trellis search algorithm is that its complexity is not determined by the consideration of exponentially increasing number of possible input words.

Although the trellis search algorithm's complexity is not determined by the exponentially increasing number of codewords, it is constrained by the memory required to store the Hamming weights of the input and parity words. Therefore, input and parity weights exceeding a certain threshold, which we termed as the **IRWEF weight threshold**, are not stored.

In our forthcoming discussion we aim to show that by modelling the modulator as an inner encoder, the PCCC and SCCC principles can be employed for the analysis of turbo equalisation performance for convolutional-coded and turbo-coded GMSK systems. However, since it is not possible to model the GMSK modulator as a binary, recursive encoder in order to exploit the SCCC and PCCC principles for the ML bound analysis, we employ a simpler modulator, namely DPSK, which is also inherently recursive like GMSK.

4.5 Recursive Properties of the MSK, GMSK and DPSK Modulator

As mentioned previously, one of the important design criteria for SCCC schemes is that the inner encoder must be recursive [19]. In this section, CPM schemes such as MSK and GMSK are shown to be recursive. However, due to the difficulty in modelling these modulation techniques as binary recursive convolutional codes, a simpler modulation scheme is considered, namely DPSK modulation. Here, a suitable recursive convolutional code is utilised, in order to represent the DPSK modulator as an inner code, hence allowing the SCCC principles to be adopted in order to derive the ML bound of coded DPSK systems.

Let us proceed by examining the first CPM scheme concerned, namely MSK. As mentioned in Section 1.6.1, the transmitted information is embedded in the phase of the signal. Recalling from Equation 1.14 that the output phase of the MSK modulator at interval v is

$$\begin{aligned}\phi(t,\alpha) &= 2\pi h_f \alpha_v q(t-vT) + \pi h_f \sum_{i=-\infty}^{v-1} \alpha_i \qquad \text{for } vT \leqslant t \leqslant (v+1)T \\ &= \frac{\pi}{2}\alpha_v + \frac{\pi}{2}\sum_{i=-\infty}^{v-1}\alpha_i \qquad \text{where } \alpha_i = \pm 1 \\ &= \frac{\pi}{2}\alpha_v + \theta_v,\end{aligned} \tag{4.29}$$

since the MSK modulation index is $h_f = \frac{1}{2}$ and the phase shaping function $q(t-vT) = 0.5$ for $t > T$. The notation θ_v is the phase state and represents the accumulated phase due to the previous bits α_i, $i = -\infty \ldots v-1$, that have passed through the filter $q(t-vT)$. Equation 4.29 can also be represented graphically by the schematic of Figure 4.7. In the MSK modulator, the interleaved encoder bits are mapped from logical 0s and 1s to -1s and $+1$s, respectively. The mapped bits are multiplied by $\frac{\pi}{2}$ and subsequently added to the phase state θ_v, in order to yield the current phase $\phi(t,\alpha)$. We observe that there is inherent memory in MSK and the modulator is recursive its in nature. However, the MSK modulator is not readily modelled as a recursive convolutional encoder, since the range of phase values involved in the recursive operation is much wider compared to the binary AND operations between bits 0 and 1, which are performed in the case of DPSK,

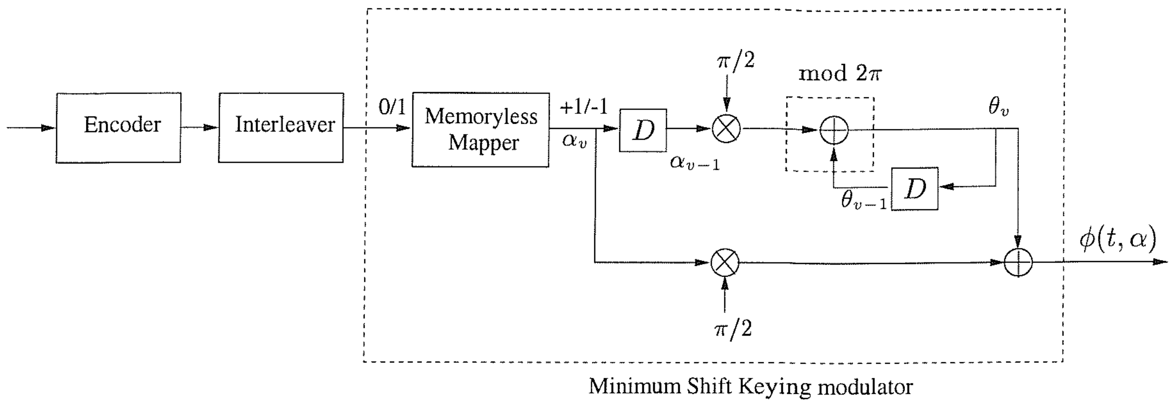


Figure 4.7: Schematic of the MSK modulator based on Figures 1.3 and 1.8, which was modified according to Equation 4.29, where D represents the symbol duration.

as described at a later stage. Furthermore, the model chosen must account for the effect of the recursive summation of the phase values on the minimum distance of the modulation scheme, hence increasing the complexity of the performance analysis of turbo-coded and convolutional-coded MSK systems.

The GMSK modulator can also be represented with the aid of a model similar to that employed for MSK. Here, the additional modifications are due to the partial-response spreading introduced by the phase shaping function $q(t)$ in the time domain. For convenience, the

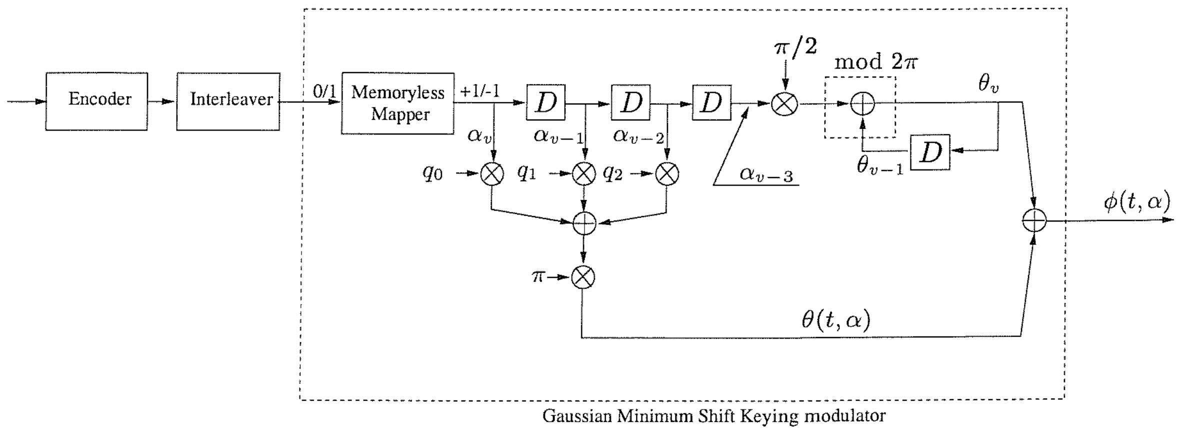


Figure 4.8: GMSK modulator model based on Figures 1.3 and 1.10 and modified according to Equation 4.30, where D represents the symbol duration.

expression for the phase of GMSK is repeated from Equation 1.9:

$$\begin{aligned}\phi(t,\alpha) &= 2\pi h_f \sum_{i=v-L+1}^{v} \alpha_i q(t-iT) + \pi h_f \sum_{i=-\infty}^{v-L} \alpha_i \\ &= \pi \sum_{i=v-2}^{v} \alpha_i q(t-iT) + \frac{\pi}{2} \sum_{i=-\infty}^{v-3} \alpha_i \\ &= \theta(t,\alpha) + \theta_v,\end{aligned} \tag{4.30}$$

where $\phi(t,\alpha)$ is the overall phase of the GMSK signal, while θ_v and $\theta(t,\alpha)$ are the phase state and the phase, which is dependent on the previously transmitted bits, also known as the correlative state vector, respectively. The phase shaping function $q(t)$ is typically spread over $L = 3$ symbol periods, as in the GSM standard [2, 3]. Taking these changes into consideration, the GMSK modulator can be schematically represented in Figure 4.8. As before, it is observed that GMSK is also recursive in its nature and possesses memory. The recursive nature of the MSK and GMSK modulation schemes can therefore be exploited, in order to yield interleaving gains when concatenated with an outer encoder and separated by an interleaver as in the SCCC schemes of Figure 4.3 for example. However, as mentioned above, the analysis of coded GMSK systems cannot readily adopt the principles invoked for SCCC schemes since GMSK — like MSK — cannot be modelled as a binary recursive convolutional code because its recursive nature exhibits itself in terms of the phase, not in terms of binary bits. Let us now describe a simple differential modulation technique, namely DPSK, which is also inherently recursive.

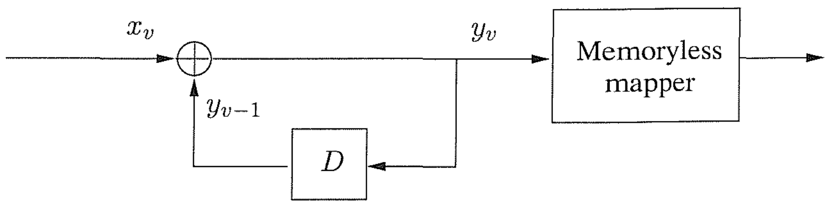


Figure 4.9: Schematic of the DPSK modulator based on Equation 4.31, where D represents the symbol duration.

Figure 4.9 illustrates the schematic of the recursive DPSK modulator, where D represents the symbol period delay. The DPSK modulator can be viewed as a block, which performs a binary AND operation on the current binary input bit x_v and the previous output bit y_{v-1}, in order to give y_v at trellis interval v, which is formulated as:

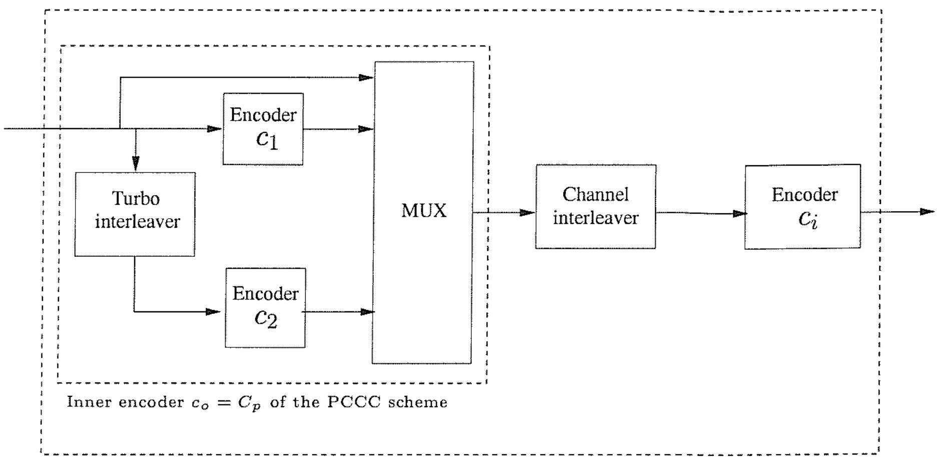
$$y_v = x_v \oplus y_{v-1}. \tag{4.31}$$

Subsequently, the memoryless mapper as shown in Figure 4.9 translates the resultant bit $y_j = 1$ to the phase $\theta = 0$ radians and $\theta = \pi$ radians for $y_j = 0$. Explicitly, the output of the DPSK modulator, is dependent on the previous output bit, hence exhibiting recursive properties. It was observed that the schematic of the DPSK modulator of Figure 4.9 has a structure similar to the rate $R = 0.5$, constraint length $K = 2$, recursive systematic convolutional encoder using octal generator polynomials $G_0 = 3$ and $G_0 = 2$, as illustrated in Figure 4.5. However, the DPSK modulator only generates a single discrete response for every input bit received, unlike the rate $R = 0.5$ convolutional encoder above, which produces two coded bits for every source bit received. Therefore, the DPSK modulator can be modelled as a rate $R = 1$ recursive systematic convolutional encoder with constraint length $K = 2$ and octal generator polynomials $G_0 = 3$ and $G_0 = 2$, coupled with a memoryless mapper. The rate $R = 1$ RSC encoder is obtained by retaining the parity bit of the rate $R = 0.5$ RSC encoder and discarding the systematic bit. Furthermore, since the mapper is memoryless, the encoder state transitions are not affected. Hence, the DPSK modulator can be modelled solely by the encoder without the mapper.

At this stage, we have shown that MSK, GMSK as well as DPSK possess memory and are recursive in their nature, hence they are suitable for SCCC-like schemes and capable of achieving large interleaving gains [19]. Due to the difficulty of representing MSK and GMSK modulation as a binary convolutional code, in our approach DPSK modulation is employed. DPSK can be modelled as a rate $R = 1$ recursive, convolutional code using octal generator polynomials $G_0 = 3$ and $G_1 = 2$. Therefore, in our approach the theoretical analysis of turbo-coded and convolutional-coded GMSK schemes ensues employing DPSK instead of GMSK. In the next section, we discuss the analytical model of the coded DPSK system and how the associated ML bound can be determined.

4.6 Analytical Model of Coded DPSK Systems

Having identified a suitable convolutional encoder for representing the DPSK modulator, the analytical model of the coded DPSK systems can be constructed. The model of the convolutional-coded DPSK scheme is the same as that of a SCCC configuration illustrated in Figure 4.3, where the inner encoder of rate $R = \frac{n}{p}$ represents the DPSK modulator. As mentioned in the previous section, the DPSK model is a rate $R = 1$, constraint length $K = 2$ recursive convolutional code, employing octal generator polynomials of $G_0 = 3$ and $G_1 = 2$. Therefore, the ML bound of the convolutional-coded DPSK system can be determined by applying the principles of the SCCC scheme of Section 4.3 directly.



Overall encoder of the SCCC scheme C_s

Figure 4.10: Modelling the turbo-coded DPSK system as a hybrid code, consisting of a parallel concatenated convolutional code, C_p which is coupled serially via a channel interleaver, to a inner encoder c_i. The DPSK modulator is modelled by a recursive, rate $R = 1$ inner convolutional encoder.

For a non-punctured turbo-coded DPSK system the analytical model consists of two parallel concatenated encoders c_1 and c_2, which are separated by a turbo interleaver. This forms the outer encoder c_o — also equivalent to C_p of Section 4.2 — and is coupled serially with an inner encoder c_i, which models the DPSK modulator. The entire model C_s of the turbo-coded DPSK system is depicted in Figure 4.10. The resultant configuration is that of a hybrid PCCC and SCCC code and therefore the theoretical analysis of the turbo-coded DPSK system considered requires employing both the PCCC and SCCC principles. Initially, the IRWEF of the parallel concatenated code $A^{C_p}(W, Z)$ is evaluated by employing the PCCC principles described in Section 4.2. Once this IRWEF has been determined, the overall system can be viewed as a SCCC scheme, consisting of the outer encoder c_o — which is the parallel concatenated code now characterised by its IRWEF — and the inner encoder c_i describing the DPSK modulator. The IRWEF $A^{C_p}(W, Z)$ of Equation 4.1 is subsequently modified and treated as the IOWEF of the outer encoder $A^{c_o}(W, L)$, which is a function of the dummy variables W and L. This is achieved by substituting Z in Equation 4.1 by the dummy variable L, where the exponent of L is the associated Hamming weight of the outer codeword l obtained by taking the sum of the exponents of the dummy variables W and Z in Equation 4.1, *i.e.* $w + j$. Now, by evaluating the IOWEF of the inner encoder $A^{c_i}(L, H)$ as well, where the orders of L and H are the Hamming weights of the input words of the inner encoder (or the Hamming weight of the output encoder's codewords) and those of its output codeword, the overall

IOWEF $A^{C_s}(W, H)$ of the PCCC/SCCC hybrid scheme can be determined by using the SCCC principles of Section 4.3. Subsequently, with the aid of Equation 4.3 and using the IOWEF $A^{C_s}(W, H)$ determined, the ML bound of the turbo-coded DPSK system can be evaluated.

At this stage we have to remind the reader that there are several limitations associated with the theoretical analysis of the coded-DPSK system concerned. The first limitation is due to the use of the abstract uniform interleaver in the analysis, which only yields the average ML performance instead of the exact performance of the specific interleaver employed. However, since the analysis is utilised as a tool for comparisons, all the coded systems are analysed in the same manner, and hence determining the average performance bound is adequate for these purposes. Secondly, the theoretical bound derived using union bound techniques produces a divergence in the ML bound at E_b/No values below the so-called cut-off rate [71], leading to an inaccurate ML bound. However, as long as the systems compared are analysed using the same technique, the ML bound derived can be used — with due caution — in order to characterise the performance trends of the associated systems. Thirdly, the recursive weight evaluation algorithm of Section 4.4 employed is constrained by the memory requirement for large Hamming weights. In order to circumvent this memory-limitation, a threshold known as the IRWEF weight threshold can be employed, whereby only Hamming weights below this threshold are retained, while the others are discarded. Due to this limitation, we are unable to determine the weight distribution D_m associated with high values of the codeword Hamming weights m, which were identified as the reason for the divergence in the ML bound [20]. In such cases, where the input word is long, the ML bound can be employed in a more limited context, for example in order to determine the error floor of the system.

In the following section we will compare our simulation and theoretical results for non-punctured convolutional-coded DPSK systems and non-punctured turbo-coded DPSK schemes, which employ channel interleavers of depths of 300 bits and 30000 bits between the encoder and modulator.

4.7 Theoretical and Simulation Performance of Coded DPSK Systems

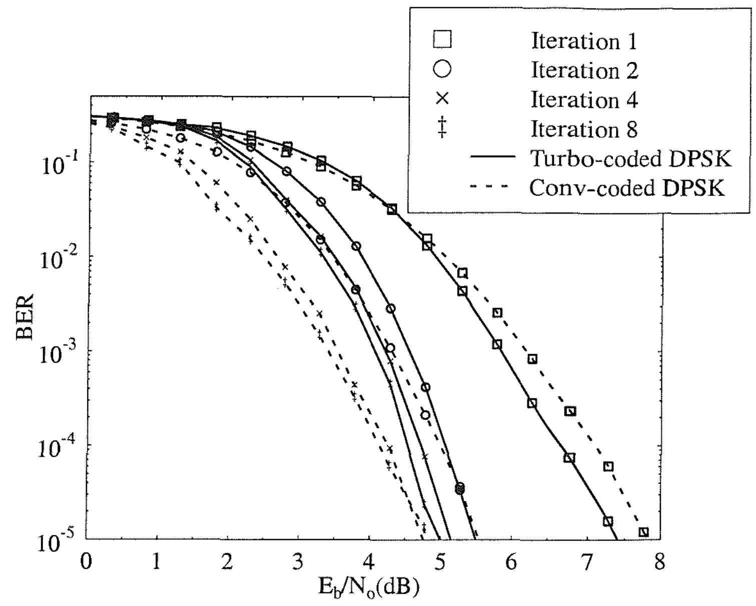
In this section, the results of our comparative study between a code rate $R = \frac{1}{3}$ turbo-coded DPSK scheme and a $R = \frac{1}{3}$ convolutional-coded DPSK system will be presented. The parameters employed are summarised in Table 4.7. Initially, a channel interleaver

Encoder	Constraint length K	Octal Generator polynomials
$R = \frac{1}{3}$ convolutional code	3	$G_0 = 5$ $G_1 = 7$ $G_2 = 7$
	5	$G_0 = 25$ $G_1 = 33$ $G_2 = 37$
$R = \frac{1}{3}$ convolutional-coding	3	$G_0 = 7$ $G_1 = 5$
based turbo code	5	$G_0 = 35$ $G_1 = 23$

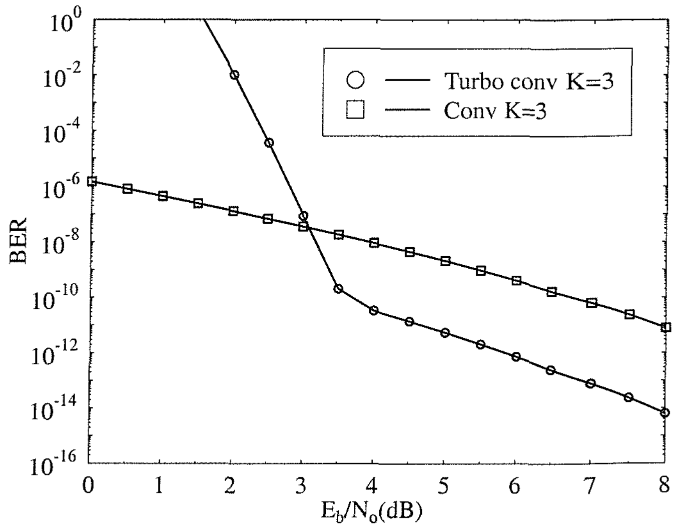
Table 4.7: Parameters of the encoders used in the $R \approx \frac{1}{3}$ rate turbo-coded and convolutional-coded DPSK systems.

possessing a depth of 300 bits was employed. This corresponds to a turbo interleaver depth of 100 bits at the input of the turbo encoder, while introducing the same delay. The reason for employing such a short interleaver is to avoid the previously mentioned memory limitation associated with the theoretical analysis. In Figure 4.11(a) the simulation results of the rate $R = \frac{1}{3}$, constraint length $K = 3$ turbo-coded and convolutional-coded DPSK system employing eight turbo equalisation iterations over the non-dispersive Gaussian channel, were presented. Again, a random channel interleaver with a depth of 300 bits was employed, while the random turbo interleaver had a depth of 100 bits. It was observed that iteration gains — *i.e.* SNR performance gains with respect to the first iteration — can be obtained by both the convolutional-coded and turbo-coded DPSK schemes. Based on our simulation results in Figure 4.11(a) the convolutional-coded DPSK system achieved an iteration gain of 2.0 dB, 2.9 dB and 3.0 dB, after two, four and eight turbo equalisation iterations, respectively, at BER $= 10^{-4}$. By contrast, for BER $= 10^{-4}$, the turbo-coded DPSK system achieved iteration gains of 1.5 dB, 2.1 dB and 2.2 dB after two, four and eight turbo equalisation iterations. These results further justify the importance of the recursive nature in the inner component of a serially concatenated code. After eight turbo equalisation iterations, the convolutional-coded DPSK scheme was observed to outperform the turbo-coded DPSK system by a margin of 0.5 dB at BER $= 10^{-4}$.

Studying the ML bound of Figure 4.11(b) obtained through our theoretical analysis — where the same simulation parameters were employed as in our computer simulations —, it was observed that at $E_b/N_o < 3$ dB, the convolutional-coded DPSK system has a lower BER compared to the turbo-coded DPSK scheme. This result further justifies the ability of the convolutional-coded DPSK system outperforming the turbo-coded DPSK scheme. However, at $E_b/N_o > 3$ dB the convolutional-coded system yielded a higher error floor compared to the turbo-coded scheme. Note that although the theoretical analysis only yields the average performance of the system and does not predict the BER performance accurately due to the previously mentioned divergence, it is an adequate tool for comparing



(a) Computer simulation



(b) Theoretical bound

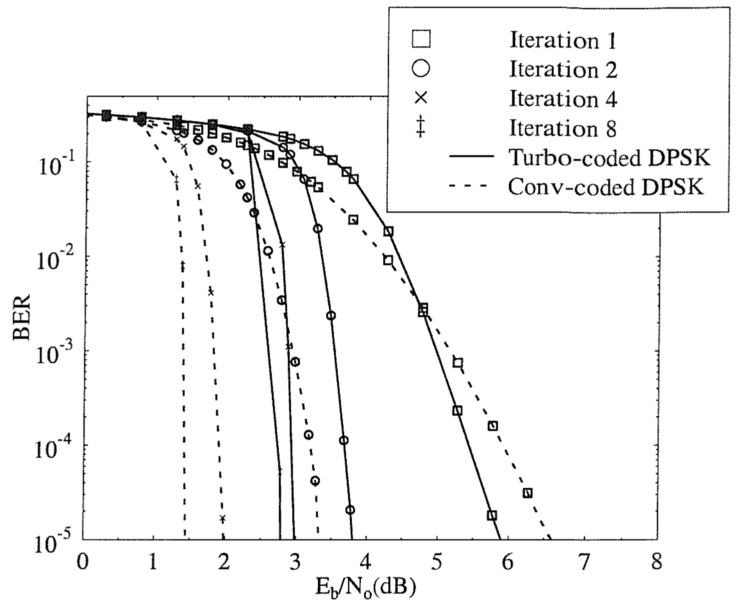
Figure 4.11: Comparing the performance of the rate $R = \frac{1}{3}$ $K = 3$ convolutional-coded DPSK system with that of the $R = \frac{1}{3}$ turbo-coded DPSK scheme over the non-dispersive Gaussian channel using both computer simulations and theoretical analysis. A random channel interleaver with a depth of 300 bits was employed, while the random turbo interleaver had a depth of 100 bits.

the performance of coded schemes in order to identify the more robust encoder. Note that the theoretical analysis for such short channel interleavers is not memory intensive, since the corresponding input word length, and consequently the Hamming weight, is not high. Therefore, no IRWEF weight threshold was set.

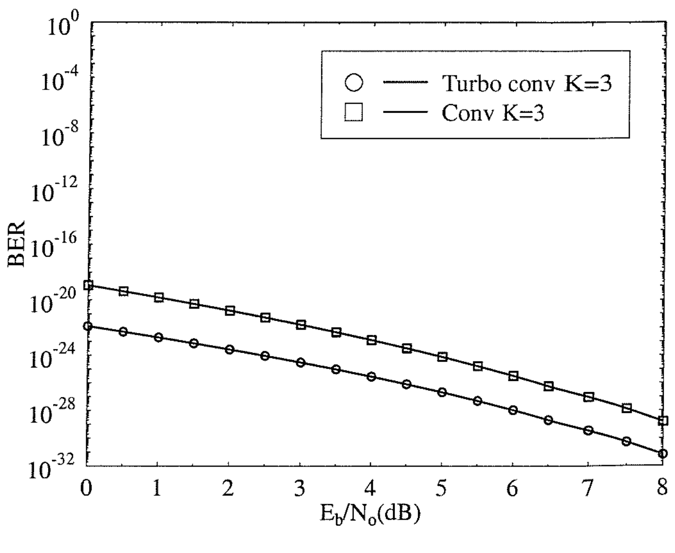
In our next experiment, random channel interleavers having depths of 30000 bits and random turbo interleavers possessing depths of 10000 bits were utilised. Here, the constraint length of the rate $R = \frac{1}{3}$ encoders was set to $K = 5$ and the octal generator polynomials of Table 4.7 were employed. The results of our computer simulations were presented in Figure 4.12(a). It was noted that the iteration gains achieved by the convolutional-coded scheme were 2.7 dB, 4.0 dB and 4.5 dB after two, four and eight turbo equalisation iterations, whereas the turbo-coded system obtained gains of 1.8 dB, 2.5 dB and 2.7 dB, respectively. Compared to the scenario, where a short turbo interleaver with a depth of 100 bits was employed, the iterations gains observed in Figure 4.12(a) were higher, since the longer interleaver minimised the correlation between the input of the equaliser and the decoder(s), hence benefiting from the information provided by the other equaliser or decoder blocks. Another key observation was that after eight turbo equalisation iterations, the convolutional-coded DPSK scheme was observed to outperform the turbo-coded DPSK system by a margin of 1.2 dB at BER $= 10^{-4}$.

For the theoretical analysis of Figure 4.12(b), a IRWEF weight threshold of 100 was set, since it was not feasible to accumulate Hamming weights corresponding to an input word length of 10000 bits. The analysis was based on the same $R = \frac{1}{3}$, $K = 5$ encoder parameters as tabulated in Table 4.7 and employing the abstract uniform interleaver of reference [20]. Since the Hamming weights have been truncated, we were unable to determine the weight distribution values D_m of Equation 4.4, for higher codeword Hamming weights. The distribution of the codeword Hamming weights in this high-weight region determines the divergence of the ML bound, while the D_m values of the lower-weight region are responsible for the error floor performance [20]. Therefore, the ML bound evaluated for the coded systems considered in conjunction with such long channel interleavers — with a depth of 30000 bits — only reflects the error-floor performance of the coded DPSK schemes. As before, the error floor of the turbo-coded DPSK system in Figure 4.12(b) was observed to be lower than that of the convolutional-coded DPSK scheme.

In conclusion, our simulation results in Figures 4.11(a) and 4.12(a) showed that convolutional-coded DPSK systems outperformed turbo-coded DPSK schemes, which employed turbo equalisation. In our simulations employing 300-bit random channel interleavers



(a) Computer simulation



(b) Theoretical bound

Figure 4.12: Comparing the performance of the rate $R = \frac{1}{3}$ $K = 5$ convolutional-coded DPSK system with that of the $R = \frac{1}{3}$ turbo-coded DPSK scheme, over the non-dispersive Gaussian channel using both computer simulations and theoretical analysis. A random channel interleaver possessing a depth of 30000 bits and a 10000-bit random turbo interleaver were implemented.

the convolutional-coded DPSK system outperformed the turbo-coded scheme by a margin of 0.5 dB at BER $= 10^{-4}$, while a gain of 1.2 dB was achieved by the convolutional-coded scheme over the turbo-coded system, when a channel interleaver with a depth of 30000 bits was employed. This performance trend was also observed in Chapter 3 for our coded GMSK schemes. Intuitively, the turbo-coded scheme is expected to perform better since it employs two encoders, which are concatenated in parallel, as opposed to the single encoder in convolutional codes. Therefore, the theoretical analysis of coded DPSK systems was invoked, in order to justify the simulation results obtained. Through the analysis using a 300-bit uniform channel interleaver, it was observed in Figure 4.11(b) that the convolutional-coded system performed better, than the turbo-coded system at $E_b/N_o < 3$ dB. However the former yielded a higher undesirable error floor. For longer channel interleaver depths, specifically for 30000 bits, the ML bound of the convolutional-coded system gave a worse error-floor performance, than that of the turbo-coded DPSK scheme. Note that in the analysis of the systems employing such interleaver depths, the input words of Hamming weights above 100 were not considered. Hence, the region where the divergence occurs could not be determined and therefore the ML bound derived reflects only the error-floor performance and it is incapable of reflecting the BER performance adequately at low E_b/N_o values. From the observations made, it can be concluded that for speech or data systems — requiring BER $= 10^{-3} - 10^{-4}$ — employing recursive modulation techniques such as DPSK or GMSK and utilising turbo equalisation, convolutional coding is the more robust choice as compared to the more complex turbo coding schemes considered.

4.8 Summary

This chapter described how the union bound principles of the SCCC and PCCC schemes can be employed to evaluate the maximum likelihood bound of the convolutional-coded and turbo-coded DPSK systems investigated. Each component encoder was characterised by the Input Redundancy Weight Enumerating Function, which categorised the codes according to the input word and parity word Hamming weight or in terms of the input word and codeword Hamming weight quantified by the Input Output Weight Enumerating Function (IOWEF). By modelling the modulator as a convolutional encoder, the IRWEF and IOWEF was evaluated in conjunction with the abstract uniform interleaver in order to determine the ML performance of an encoder concatenated with a modulator. The aim of this study was to provide a theoretical justification of the results obtained for coded GMSK schemes employing turbo equalisation in Chapter 3. Furthermore, it was intended that the investigations showed the importance of employing inner encoders constituted by the modems in a serial concatenated scheme, which were recursive in their nature. In Chapter 3, it was observed that the convolutional-coded GMSK scheme achieved better performance than the convolutional-coding based turbo-coded GMSK system employing turbo equalisation, despite utilising only one convolutional decoder, while the turbo-coded scheme employed two decoders. In our investigations, we have used DPSK instead of GMSK modulation. This was because DPSK — unlike GMSK — could be readily modelled as a binary recursive convolutional encoder and hence allowing the union bound SCCC principles to be employed in order to evaluate the ML bound of the system. GMSK is also recursive in its nature, but in terms of phase values (radians), whereas DPSK operated recursively on binary values and hence it was modelled similarly to recursive convolutional encoders. Hence we have employed DPSK in our analysis of the associated performance bounds. From our simulation results generated for turbo-coded DPSK schemes and convolutional-coded DPSK systems, it was observed that the convolutional-coded scheme outperformed the turbo-coded system. This was further justified by the similar trends of the theoretical performance bounds. Despite the divergence of the theoretical bound below the cut-off rate, the ML bound derived was still a reliable experimental tool, since the aim of the analysis was to characterise the relative performance of the coded systems analysed with the aid of the same analytical technique. In Figure 4.11(b), for a channel interleaver depth of 300 bits, the theoretical bound of the turbo-coded DPSK system diverged at an E_b/N_o value of approximately 3.0 dB, and showed that the performance of the convolutional-coded system was better than that of the turbo-coded system at low E_b/N_o values. As the E_b/N_o value increased beyond 3 dB, the theoretical bound for the convolutional-coded system began to flatten at approximately $\text{BER} = 10^{-8} - 10^{-12}$, whereas the error-floor for the turbo-coded system for $E_b/N_o > 3$ dB was approximately $\text{BER} = 10^{-10} - 10^{-14}$. For channel interleaver depths of 30000 bits the error-floor of the convolutional-coded scheme was approximately $\text{BER} = 10^{22} - 10^{28}$, while the turbo-coded DPSK system yielded an error-floor of $\text{BER} = 10^{26} - 10^{32}$. Therefore, it can be concluded that the turbo-coded DPSK system exhibited a better error-floor, than the convolutional-coded system but at low E_b/N_o values, its performance was poorer, than that of the convolutional-coded DPSK system. These observations and conclusions were also consistent with the results obtained for convolutional-coded GMSK schemes and turbo-coded GMSK systems. Furthermore, it can be concluded that for speech and data systems — requiring $\text{BER} = 10^{-3}$ and $\text{BER} = 10^{-4}$, respectively — employing recursive modulation techniques such as DPSK or GMSK and utilising turbo equalisation, the convolutional code is the more robust choice, compared to the more complex turbo code.

Chapter 5

Comparative Study of Turbo Equalisers

In Chapter 3, turbo equalisation was investigated in the context of coded partial response GMSK systems. The inherent recursive nature of GMSK modulation was exploited, in order to achieve large interleaver gains. Furthermore, it was observed in Chapter 4 through computer simulations that for recursive modulation systems such as GMSK and DPSK convolutional-coded schemes outperformed convolutional-coding based turbo coded systems at low E_b/N_o values and for BER $> 10^{-5}$. The theoretical ML bounds of the convolutional-coded and turbo-coded DPSK systems in Chapter 4 also showed that the convolutional-coded scheme was more powerful than the investigated turbo-coded systems at these E_b/N_o values. However, at higher E_b/N_o values, it was observed that the turbo-coded scheme yielded lower error-floors compared to the convolutional-coded scheme. It was therefore concluded for recursive modulation systems transmitting data and speech — *i.e.* upon requiring BER $= 10^{-4}$ and BER $= 10^{-3}$, respectively and employing turbo equalisation — that convolutional codes are more robust, compared to the investigated turbo-coded schemes.

5.1 Motivation

In this chapter, BPSK modulation is employed in order to investigate the performance of the turbo equaliser in the context of non-recursive modulation systems. In this case the modulator and dispersive channel is viewed as the inner encoder, while the channel encoder employed is perceived to be the outer encoder, as in the SCCC scheme described in Section 4.3. In addition to convolutional codes and convolutional-coding based turbo codes, block-coding based turbo codes are also researched in conjunction with BPSK systems utilising turbo equalisation. With the ever increasing demand for bandwidth, current systems aim to increase the spectral efficiency by invoking high-rate codes. This has been the motivation for research into block turbo codes, which have been shown by Hagenauer *et al.* [87] to outperform convolutional turbo codes, when the coding rate is higher than $\frac{2}{3}$. It was also observed that a rate $R = 0.981$ block turbo code using BPSK over the non-dispersive Gaussian channel can operate within 0.27 dB of the Shannon limit [88]. In reference [89] Pyndiah presented iterative decoding algorithms for BCH turbo codes. **In this chapter we construct a BPSK turbo equaliser, which employs block-coding based turbo codes with the objective of investigating its performance in comparison to turbo equalisers employing different classes of codes for high code rates of $R = \frac{3}{4}$ and $R = \frac{5}{6}$, since known turbo equalisation results have only been presented for turbo equalisers using convolutional codes and convolutional-coding based turbo codes for code rates of $R = \frac{1}{3}$ and** $R = \frac{1}{2}$ [10, 21]. Specifically, Bose-Chaudhuri-Hocquengham (BCH) codes [22, 23] are used as the component codes of the block-coding based turbo codec. Since BCH codes may be constructed with parameters n and k, which represent the number of coded bits and data bits, respectively, we will use the notation BCH (n,k). The BCH-coding based turbo-coded systems are denoted as **BT**, while the convolutional-coding based turbo-coded schemes and convolutional-coded systems are represented as **CT** and **CC**, respectively.

The organisation of this chapter is as follows. Section 5.2 provides an overview of the systems researched. Subsequently, Section 5.3 summarises the simulation parameters. Finally, Section 5.4 provides results and discussions, while Section 5.5 summarises the the systems' performance.

5.2 System overview

Again, in this comparative study three classes of encoders, namely convolutional codes, convolutional-coding based turbo codes and BCH-coding based turbo codes are employed, which are serially concatenated with the BPSK modulator. The encoder parameters will be specified in the following section. In addition to the channel interleaver, which separates the encoder and the modulator, turbo interleavers are also implemented for the turbo encoders. At the receiver, the equaliser and decoder(s) are configured to perform either independent equalisation and decoding or turbo equalisation, where equalisation and decoding is performed jointly by exchanging information iteratively between the equaliser and decoder(s). Specifically, for the convolutional-coded system, the receiver implements either conventional convolutional decoding [72, 37] or turbo equalisation. The turbo equalisation operation is based on the principles described in Section 3.2 using $N_d = 1$ decoder. For convolutional-coding based turbo codes and BCH-coding based turbo codes, independent equalisation and decoding refers to the scenario, where soft decision is passed from the equaliser to the turbo decoder, which performs its decoding by passing information between the decoders, but never with the equaliser [9, 89]. In these systems turbo equalisation is also based on the principles of Section 3.2, but in this case information is exchanged between the equaliser and the $N_d = 2$ decoders.

In the following section, we specify the parameters of the convolutional-coded BPSK system, convolutional-coding based turbo-coded BPSK scheme and the BCH-coding based turbo-coded BPSK system.

5.3 Simulation Parameters

In this chapter, BPSK modulation is employed in all the examined systems. The first system described is a convolutional-coded scheme, denoted by **CC**. A rate $R = \frac{1}{2}$, constraint length $K = 5$, recursive systematic convolutional code was used with octal generator polynomials of $G_0 = 35$ and $G_1 = 23$ as summarised previously in Table 3.7. In order to obtain $R = \frac{3}{4}$ and $R = \frac{5}{6}$-rate convolutional codes, we have employed the Digital Video Broadcast (DVB) puncturing pattern [90] specified in Table 5.1. For a fair comparative

Code Rate $R = \frac{3}{4}$	Code Rate $R = \frac{5}{6}$
G_0: 1 0 1 G_1: 1 1 0	G_0: 1 0 1 0 1 G_1: 1 1 0 1 0
1 = transmitted bit 0 = non transmitted bit	

Table 5.1: DVB puncturing pattern [90] applied to the coded bits of the $R = \frac{1}{2}$ convolutional code in order to obtain code rates $R = \frac{3}{4}$ and $R = \frac{5}{6}$ convolutional codes.

study, it was adequate for the turbo codes to employ a simple regular puncturing pattern, even though it was recognised that puncturing patterns can be optimised to improve the performance of turbo codes [91]. For the convolutional-coding based turbo-coded system, represented by **CT**, we have used the convolutional constituent codes with the same parameters — *i.e.* $R = \frac{1}{2}$, $K = 5$ — as described previously for example in Table 3.7. When no puncturing is implemented, the overall rate of the turbo code is $R = \frac{1}{3}$. Therefore, we have applied regular puncturing — as detailed in Table 5.2 — to the turbo codes, in order to obtain $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$ rate convolutional-coding based turbo codes.

Code Rate $R = \frac{1}{2}$	Code Rate $R = \frac{3}{4}$	Code Rate $R = \frac{5}{6}$
C1: 1 0	C1: 1 0 0 0 0 0	C1: 1 0 0 0 0 0 0 0 0 0
C2: 0 1	C2: 0 0 1 0 0 0	C2: 0 0 0 0 1 0 0 0 0 0
1 = transmitted bit		
0 = non transmitted bit		

Table 5.2: Regular puncturing pattern used in order to obtain the $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$ convolutional-coding based turbo codes. The terms C1 and C2 represent the parity bits of $R = \frac{1}{2}$ convolutional codes of the first and second constituent codes, respectively.

Finally, for the BCH-coding based turbo-coded system, which we denoted by **BT**, three different constituent BCH codes were used, namely the BCH(15,11) code, the BCH(31,26) code and the BCH(63,57) code, in order to obtain the $R = \frac{11}{19}$, $R = \frac{26}{36}$ and $R = \frac{57}{69}$ code rates, respectively. No puncturing is required for this class of turbo equalisers. A summary of all three classes of encoder parameters is shown in Table 5.3. We used random channel interleavers for all three turbo equalisation systems and the depth was set to approximately 20000 bits. Similarly, random turbo interleavers — which have an odd-even separation [61] — were used in the turbo equalisers employing BCH turbo codes and convolutional-coding based turbo codes. The detailed channel and turbo interleaver depths are specified in Table 5.3.

We have assumed perfect knowledge of the channel impulse response and for the Soft-In/Soft-Out (SISO) equaliser and SISO decoder we have used the Log-Maximum *A Posteriori* (Log-MAP) algorithm [60], since the Log-MAP algorithm achieves identical performance to the original Maximum *A Posteriori* (MAP) algorithm [17], despite having a reduced computational complexity. Furthermore, the term **decoding** refers here to the scenario, where the equaliser passes soft outputs to the decoder and there is no iterative processing between the equaliser and decoder(s). When using **turbo decoding**, there will be decoding iterations, where information is passed iteratively between the component decoders, but not between the decoders and the equaliser. Information is only passed iteratively between the equaliser and decoder(s), when **turbo equalisation** is employed. For our work, we have used eight turbo decoding and turbo equalisation iterations.

The complexity of the turbo equaliser for each system investigated can be characterised by the number of states in the entire decoder trellis for each iterative step. Here, the complexity of the equaliser is not taken into account, since the same equaliser is used in all the turbo-equalised systems. For example, a trellis-based convolutional decoder employing

Encoder	Random π_t depth	Random π_c depth	Puncturing
Conv Rate $R = \frac{1}{2} = 0.5$	None	20736	None
Conv Rate $R = \frac{3}{4} = 0.75$	None	20736	See Table 5.1
Conv Rate $R = \frac{5}{6} = 0.833$	None	20736	See Table 5.1
Turbo Conv Rate $R = \frac{1}{2} = 0.5$	10368	20736	See Table 5.2
Turbo Conv Rate $R = \frac{3}{4} = 0.75$	15552	20736	See Table 5.2
Turbo Conv Rate $R = \frac{5}{6} = 0.833$	17280	20736	See Table 5.2
Turbo BCH (15,11) Rate $R = \frac{11}{19} = 0.579$	12672	21888	None
Turbo BCH (31,26) Rate $R = \frac{26}{36} = 0.722$	14976	20736	None
Turbo BCH (63,57) Rate $R = \frac{57}{69} = 0.826$	16416	19872	None

Table 5.3: Parameters of the encoders used in the $R \approx \frac{1}{2}$, $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$-rate BPSK **CC**, **CT**, **BT** systems. The notations π_t and π_c represent the turbo and channel interleaver, respectively.

the Log-MAP algorithm has 2^{K-1} states at each time instant, where K is the code constraint length. Hence, for an encoder input block length of 10000 bits the total number of states in the entire trellis is $10000 \cdot 2^{K-1}$. Since a turbo equaliser employing convolutional-coding based turbo codes consists of $N_d = 2$ convolutional decoders, its receiver complexity is twice that of the turbo equaliser using conventional convolutional codes. For BCH (n,k) trellis decoders the total number of states in the trellis is approximately :

$$\text{Number of states in decoder trellis} = \frac{\text{Encoder Input Block Length}}{k} \cdot (2k - n + 3) \cdot 2^{n-k}. \tag{5.1}$$

Table 5.4 shows the turbo equaliser complexity of each turbo equalisation iteration, for the three different classes of encoders.

The transmission burst structure used in all systems was the so-called FMA1 non-spread speech burst as specified in the Pan-European FRAMES proposal [92] and shown in Figure 5.1. Our comparative study was conducted over the five-path Gaussian channel and the equally-weighted five-path Rayleigh fading channel using a normalised Doppler frequency

Code rate	Complexity [States]		
	Convolutional code	convolutional-coding based turbo code	BCH-coding based turbo code
$R \approx \frac{1}{2}$	165 888	331 776	368 640
$R \approx \frac{3}{4}$	248 832	497 664	884 736
$R \approx \frac{5}{6}$	276 480	552 960	1 990 656

Table 5.4: The complexity of the BPSK turbo equalisers employing three different classes of codes after one turbo equalisation iteration, as a function of the total number of states in the trellis-based decoder(s).

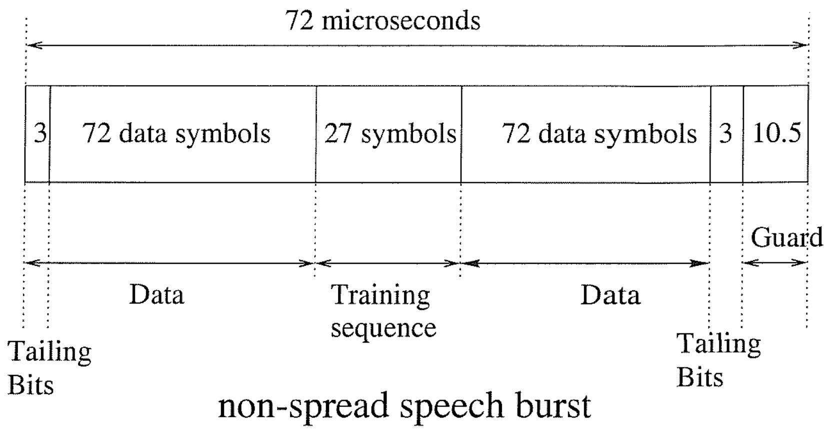
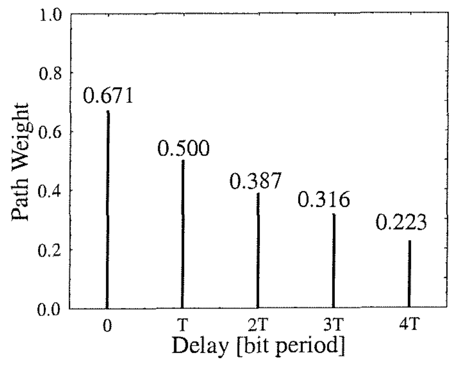


Figure 5.1: Transmission burst structure of the FMA1 non-spread speech burst of the FRAMES proposal [92].

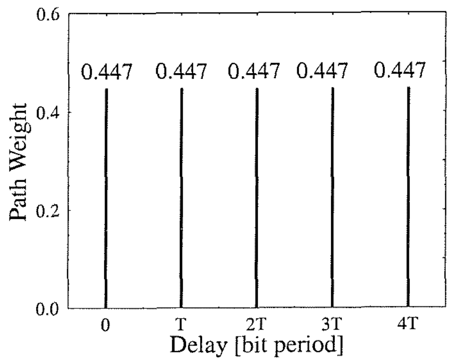
of $f_d = 1.5 \times 10^{-4}$, as illustrated in Figures 5.2(a) and 5.2(b), respectively. Again, the fading magnitude and phase was kept constant for the duration of a transmission burst, a condition which we refer to as employing burst-invariant fading.

5.4 Results and Discussion

In this section we compare the turbo equalisation and decoding performance of the **CC**, **CT** and **BT** systems investigated. We commence by comparing the turbo equalisation performance of the **CT**, **BT** and **CC** systems, followed by a study of the turbo equalisation performance in comparison to the decoding performance of each system for code rates of $R \approx \frac{1}{2}$, $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ over the five-path Gaussian channel and the five-path Rayleigh fading channel using burst-invariant fading of Figures 5.2(a) and 5.2(b), respectively.



(a) Five-path Gaussian channel



(b) Equally-weighted five-path Rayleigh fading channel

Figure 5.2: Channel impulse response of the five-path Gaussian channel and the equally-weighted five-path Rayleigh fading channel.

5.4.1 Five-path Gaussian Channel

Figure 5.3 shows the BPSK turbo equalisation performance of the $R = \frac{11}{19}$ **BT** system, of the $R = \frac{1}{2}$ **CT** system and that of the $R = \frac{1}{2}$ **CC** system after one and eight turbo equalisation iterations over the five-path Gaussian channel illustrated in Figure 5.2(a). We

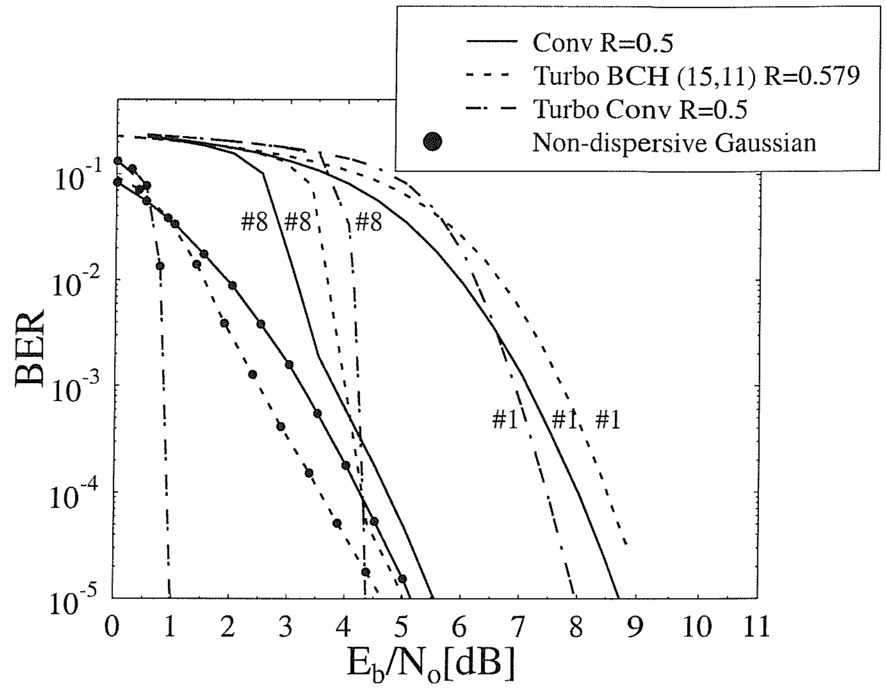


Figure 5.3: Comparing the BPSK turbo equalisation performance of the $R = \frac{1}{2}$ **CC** system, the $R = \frac{1}{2}$ **CT** scheme and the $R = \frac{11}{19}$ **BT** system for one (#1) and eight (#8) turbo equalisation iterations, over the five-path Gaussian channel of Figure 5.2(a). The decoding performance over the non-dispersive Gaussian channel — *i.e* the lower bound performance — is shown as well.

observed that the BER performance of the $R = \frac{1}{2}$ **CT** system and the $R = \frac{11}{19}$ **BT** system was comparable and both were better than that of the $R = \frac{1}{2}$ **CC** system by approximately 0.5 dB after eight turbo equalisation iterations at BER $= 10^{-4}$. The same comparison was performed for $R \approx \frac{3}{4}$ BPSK turbo equalisers in Figure 5.4. We observed that the $R = \frac{3}{4}$ **CT** scheme achieved a gain of 0.8 dB over the $R = \frac{3}{4}$ **CC** system, whereas the $R = \frac{26}{36}$ **BT** system outperformed the $R = \frac{3}{4}$ **CT** arrangement by 0.4 dB at BER $= 10^{-4}$ after eight turbo equalisation iterations. Figure 5.5 shows the turbo equalisation performance of the $R = \frac{57}{69}$ **BT** system, the $R = \frac{5}{6}$ **CT** scheme and that of the $R = \frac{5}{6}$ **CC** system. For this code rate, we observed that the $R = \frac{5}{6}$ **CT** system had a comparable BER performance to that of the $R = \frac{57}{69}$ **BT** system after eight turbo equalisation iterations. At BER $= 10^{-4}$

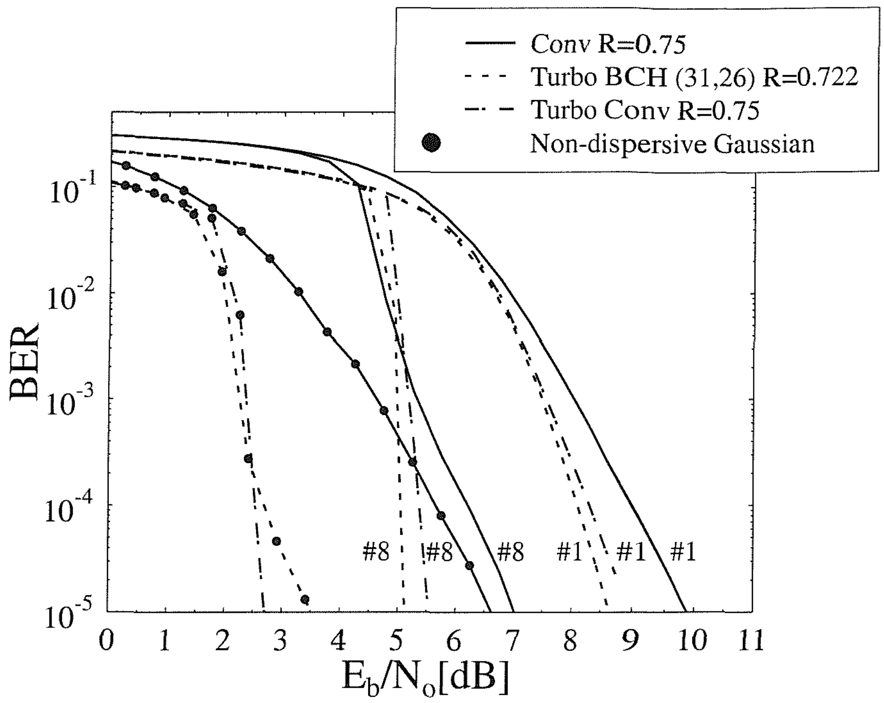


Figure 5.4: Comparing the BPSK turbo equalisation performance of the $R = \frac{3}{4}$ **CC** system, the $R = \frac{3}{4}$ **CT** scheme and the $R = \frac{26}{36}$ **BT** system for one (#1) and eight (#8) turbo equalisation iterations, over the five-path Gaussian channel of Figure 5.2(a). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

both the $R = \frac{5}{6}$ **CT** system and the $R = \frac{57}{69}$ **BT** scheme obtained a 1 dB gain over the $R = \frac{5}{6}$ **CC** system.

The results demonstrated that at high code rates the BPSK turbo equaliser using BCH turbo codes required the lowest E_b/N_o value of the three systems in order to achieve a BER of 10^{-4}, except at $R \approx \frac{5}{6}$, where the performance of the **BT** system and the **CT** system was similar. We summarised the above turbo equalisation performance results for the different **CC**, **CT** and **BT** systems and ranked them according to the E_b/N_o required to achieve a BER of 10^{-4} in Table 5.5, where '1' represents the system that requires the lowest E_b/N_o value and '3' the system that required the highest E_b/N_o value. The value within the brackets ( ) represents the E_b/N_o loss relative to the system in column '1'.

Next we compared the performance of the BPSK turbo equaliser with the decoding performance of each system over the five-path Gaussian channel of Figure 5.2(a). Note that for the concatenated-coded **BT** and **CT** schemes, turbo equalisation and turbo

Code rate	1	2	3
$R \approx \frac{1}{2}$	**BT** $\approx$ **CC** (0.0 dB)		**CC** (0.4 dB)
$R \approx \frac{3}{4}$	**BT** (0.0 dB)	**CT** (0.3 dB)	**CC** (1.1 dB)
$R \approx \frac{5}{6}$	**BT** $\approx$ **CT** (0.0 dB)		**CC** (1.0 dB)

Table 5.5: Ranking of the BPSK turbo equalisation performance for all systems over the five-path Gaussian channel from Figure 5.2(a). The notation '1' represents the system that required the lowest E_b/N_o value and '3' for the system that needed the highest E_b/N_o value to achieve a BER of 10^{-4}. The value within the brackets ( ) represents the E_b/N_o loss relative to the system in column '1'.

decoding have the same processing sequence when only one iteration is implemented, hence giving the same performance. From Figures 5.6(a), 5.6(b) and 5.6(c) we observed that by performing turbo equalisation using convolutional-coding based turbo codes instead of convolutional-coding based turbo decoding, gains of 0.7 dB, 0.8dB and 0.6 dB were achieved for code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$ at BER $= 10^{-4}$, respectively. In Figures 5.7(a), 5.7(b) and 5.7(c), we observed the same trend, where gains of 3.0 dB, 1.4 dB and 0.7 dB were achieved by the turbo equaliser using BCH turbo codes over BCH turbo decoding for code rates of $R = \frac{11}{19}$, $R = \frac{26}{36}$ and $R = \frac{57}{69}$ at BER $= 10^{-4}$. Note that for the **CC** system, the performance of the turbo equaliser after one turbo equalisation iteration is the same as the convolutional decoding performance. Therefore, we have used Figures 5.3, 5.4 and 5.5 for the comparison between the turbo equalisation and the convolutional decoding performance. Here, gains of 3.2 dB, 2.8 dB and 2.5 dB were obtained by using turbo equalisation over convolutional decoding for code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$ at BER $= 10^{-4}$.

In summary, we can conclude from the results observed that by performing equalisation and decoding jointly, a better BER performance can be obtained than by performing these operations in isolation, although for the **BT** system this performance gain begins to erode, as the code rate increases.

5.4.2 Equally-weighted Five-path Rayleigh Fading Channel

We now compare the turbo equalisation performance of the **CC**, **CT** and **BT** systems, for code rates of $R \approx \frac{1}{2}$, $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ over the five-path Rayleigh fading channel using burst-invariant fading depicted in Figure 5.2(b).

As shown in Figure 5.8, the $R = \frac{1}{2}$ **CT** system achieved a significant gain of 2.4 dB and

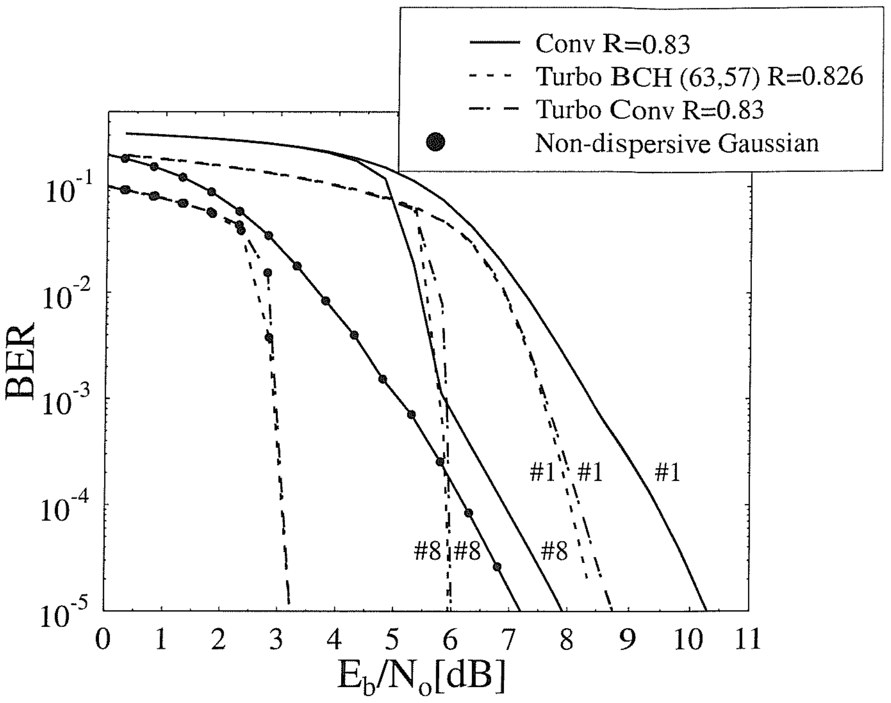
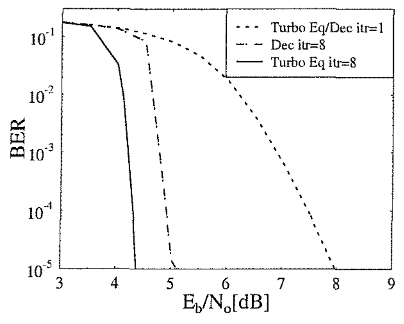


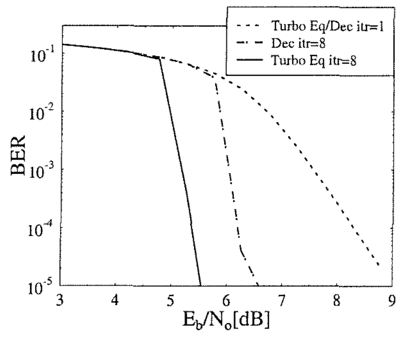
Figure 5.5: Comparing the BPSK turbo equalisation performance of the $R = \frac{5}{6}$ **CC** system, the $R = \frac{5}{6}$ **CT** scheme and the $R = \frac{57}{69}$ **BT** system for one (#1) and eight (#8) turbo equalisation iterations, over the five-path Gaussian channel of Figure 5.2(a). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

3.0 dB, when compared to the $R = \frac{11}{19}$ **BT** system and the $R = \frac{1}{2}$ **CC** system after eight turbo equalisation iterations at BER $= 10^{-4}$. For a code rate of $R \approx \frac{3}{4}$ we observed in Figure 5.9 that from the set of three turbo-equalised systems, the **BT** system required the lowest E_b/N_o value in order to achieve BER $= 10^{-4}$. Relative to the **BT** system, the **CT** system exhibited an E_b/N_o loss of 0.1 dB, while the **CC** system yielded an E_b/N_o loss of 3.6 dB at BER $= 10^{-4}$ after eight turbo equalisation iterations. The same performance trend was observed in Figure 5.10 for the $R \approx \frac{5}{6}$ rate turbo equalisers, where the **BT** system obtained an E_b/N_o gain of 0.1 dB, when compared to the $R = \frac{5}{6}$ rate **CT** system, whereas a significant gain of 3.8 dB was observed, when compared to the **CC** system after eight turbo equalisation iterations at BER $= 10^{-4}$.

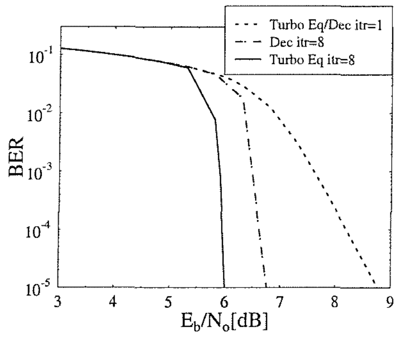
We observed, again, in the five-path Rayleigh fading channel scenario that the turbo equaliser using high rate — *i.e.* $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ — BCH turbo decoders outperformed the high rate **CC** system significantly, while only a marginal improvement over the **CT** system was obtained. In Table 5.6 we ranked the **CC**, **CT** and **BT** systems, according to the



(a) $R = \frac{1}{2}$

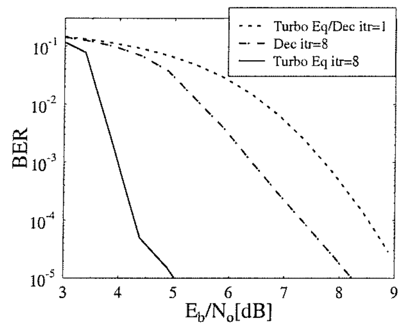


(b) $R = \frac{3}{4}$

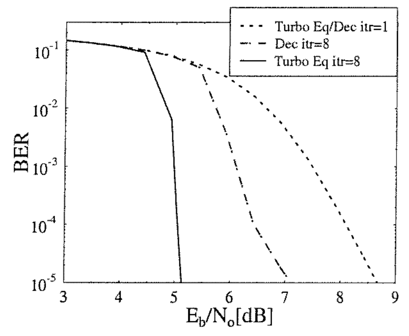


(c) $R = \frac{5}{6}$

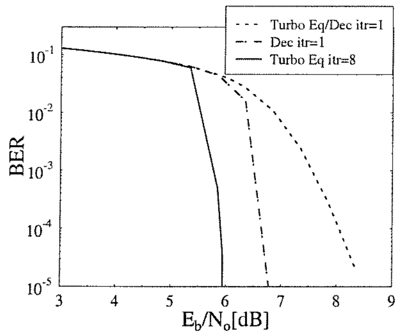
Figure 5.6: Decoding performance of the BPSK **CT** system using isolated turbo decoding compared with the turbo equalisation performance after the first iteration — which is identical for both — and after the eighth iteration for code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$, over the five-path Gaussian channel of Figure 5.2(a).



(a) $R = 0.579$



(b) $R = 0.722$



(c) $R = 0.826$

Figure 5.7: Decoding performance of the BPSK **BT** system using isolated turbo decoding compared with the turbo equalisation performance after the first iteration — which is identical for both — and after the eighth iteration for code rates of $R = \frac{11}{19}$, $R = \frac{26}{36}$ and $R = \frac{57}{69}$, over the five-path Gaussian channel of Figure 5.2(a).

Code rate	1	2	3
$R \approx \frac{1}{2}$	**CT** (0.0 dB)	**BT** (2.4 dB)	**CC** (3.0 dB)
$R \approx \frac{3}{4}$	**BT** (0.0 dB)	**CT** (0.1 dB)	**CC** (3.6 dB)
$R \approx \frac{5}{6}$	**BT** (0.0 dB)	**CT** (0.1 dB)	**CC** (3.8 dB)

Table 5.6: Ranking of the BPSK turbo equalisation performance for all systems for the five-path Rayleigh fading channel using burst-invariant fading in Figure 5.2(b). The notation '1' represents the system that required the lowest E_b/N_o value and '3' for the system that needed the highest E_b/N_o value to achieve a BER of 10^{-4}. The value within the brackets ( ) represents the E_b/N_o loss relative to the system in column '1'.

E_b/N_o value required to achieve a BER of 10^{-4}, where the index '1' is used for the system that required the lowest E_b/N_o value and '3' for the system that needed the highest E_b/N_o value. The value within the brackets ( ) represents the E_b/N_o loss relative to the system in column '1' for the five-path Rayleigh fading channel using burst-invariant fading of Figure 5.2(b).

In our next endeavour a comparison of the turbo equalisation and decoding performance was conducted for the **BT**, **CT** and **CC** system over the five-path Rayleigh fading channel. From Figures 5.11(a), 5.11(b) and 5.11(c) we observed that for all code rates investigated the turbo equaliser using convolutional-coding based turbo codes required approximately 0.4 dB lower E_b/N_o in order to achieve a BER of 10^{-4} when compared to isolated turbo decoding. The same performance trend was observed for the turbo-equalised systems in Figures 5.12(a), 5.12(b) and 5.12(c) using BCH turbo codes. Here, gains between 0.5 dB and 0.6 dB were achieved through turbo equalisation, as compared to BCH turbo decoding at BER $= 10^{-4}$ for all code rates investigated. Finally, Figures 5.8, 5.9 and 5.10 showed gains of 0.7 dB, 0.5 dB and 0.5 dB, which were achieved by employing turbo equalisation instead of convolutional decoding at BER $= 10^{-4}$ for the **CC** system at code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$, respectively.

In summary, we observed over the five-path Rayleigh fading channel that turbo equalisation — *i.e.* joint equalisation and decoding — outperforms isolated equalisation and decoding for all code rates investigated, although for the **BT** system the gain achieved through turbo equalisation was lower than that obtained over the dispersive Gaussian channel scenario.

The turbo equalisation simulations over the five-path Rayleigh fading channel of Figure 5.2(b) also showed that the **CC** system has poor iteration gain — *i.e.* a modest

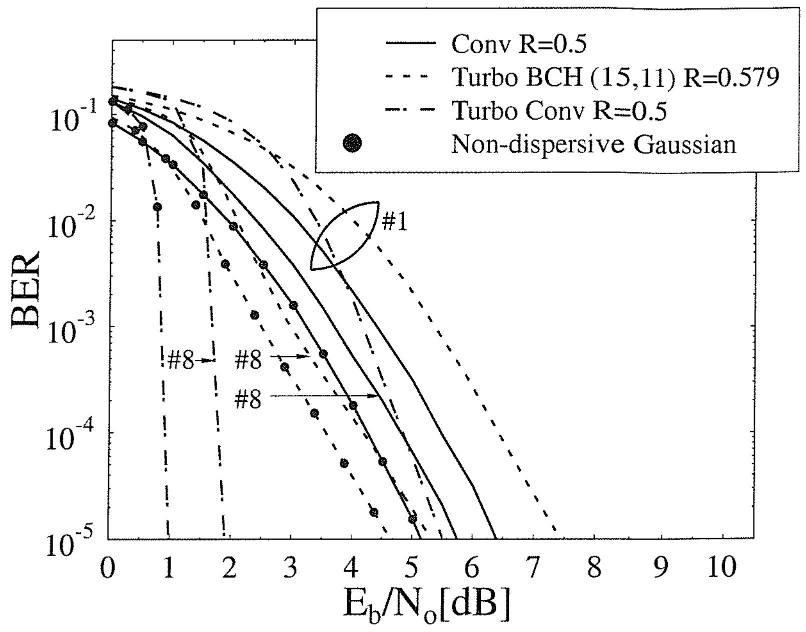


Figure 5.8: Comparing the BPSK turbo equalisation performance of the $R = \frac{1}{2}$ **CC** system, the $R = \frac{1}{2}$ **CT** scheme and of the $R = \frac{11}{19}$ **BT** system for one (#1) and eight (#8) turbo equalisation iterations, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 5.2(b). The decoding performance over the non-dispersive Gaussian channel — *i.e* the lower bound performance — is shown as well.

gain in E_b/N_o — performance with respect to the first iteration (consistent with the results presented for the $R = \frac{1}{2}$ **CC** system in references [10, 65]). For the turbo-equalised **CT** and **BT** systems, the **CT** system obtained slightly higher iteration gains. For example, the $R = \frac{5}{6}$ **CT** system obtained an iteration gain of 2.4 dB, while the $R = \frac{57}{69}$ **BT** system achieved a gain of 2.3 dB after eight turbo equalisation iterations at BER $= 10^{-4}$, as shown in Figure 5.10. At this BER, the $R = \frac{5}{6}$ **CC** system only achieves an iteration gain of 0.5 dB after eight turbo equalisation iterations.

In the five-path Gaussian channel and the five-path Rayleigh fading channel scenario, the performance of the high-code-rate $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ **BT** system is marginally better or comparable to the **CT** system at BER $= 10^{-4}$. However, this was obtained at the cost of higher receiver complexity compared to the **CT** system as seen in Table 5.4. At high code rates the **CC** system performs poorly over the five-path Rayleigh fading channel. For

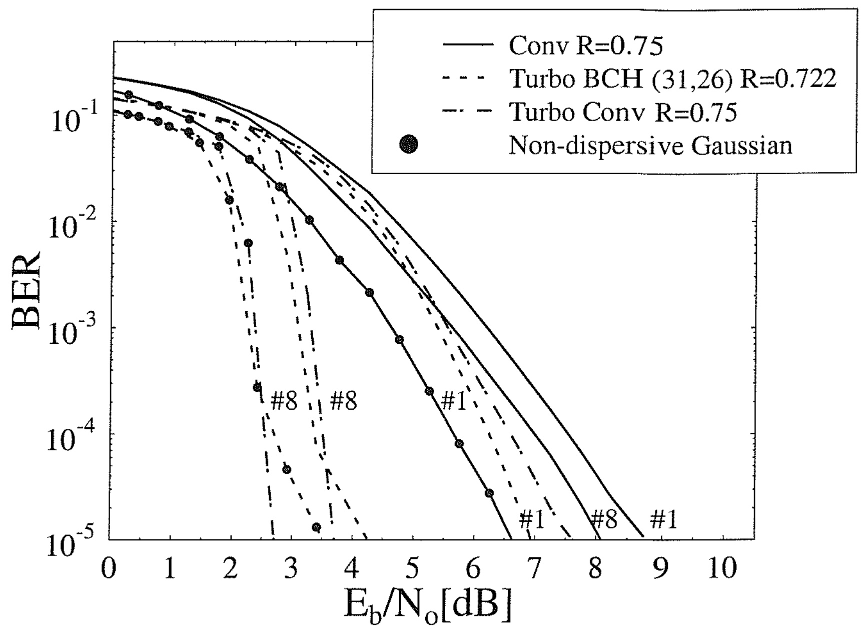


Figure 5.9: Comparing the BPSK turbo equalisation performance of the $R = \frac{3}{4}$ **CC** system, the $R = \frac{3}{4}$ **CT** scheme and the $R = \frac{26}{36}$ **BT** system for one (#1) and eight (#8) turbo equalisation iterations, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 5.2(b). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

example, at BER $= 10^{-4}$ a loss of 3.8 dB was observed, when compared to the turbo-equalised **BT** system after eight turbo equalisation iterations and of 1 dB for the dispersive Gaussian channel, when compared to the **CC** system. This inferior performance is due to the low iteration gain, which does not exceed 0.9 dB. A reason for this marginal improvement through iterative equalisation and decoding is that the **CC** system's performance is already close to the optimum — *i.e.* to the decoding performance over the non-dispersive Gaussian channel — after the first iteration.

5.5 Summary

Different receiver configurations were compared for BPSK modulated transmission systems using BCH turbo codes **BT**, convolutional-coding based turbo codes **CT** and convolutional codes **CC**. Non-iterative and iterative equaliser/decoders operating at code rates $R \approx \frac{1}{2}, R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ were studied. In the iterative cases loops containing only decoders — as in isolated turbo decoding — and loops containing joint equalisation and decoding

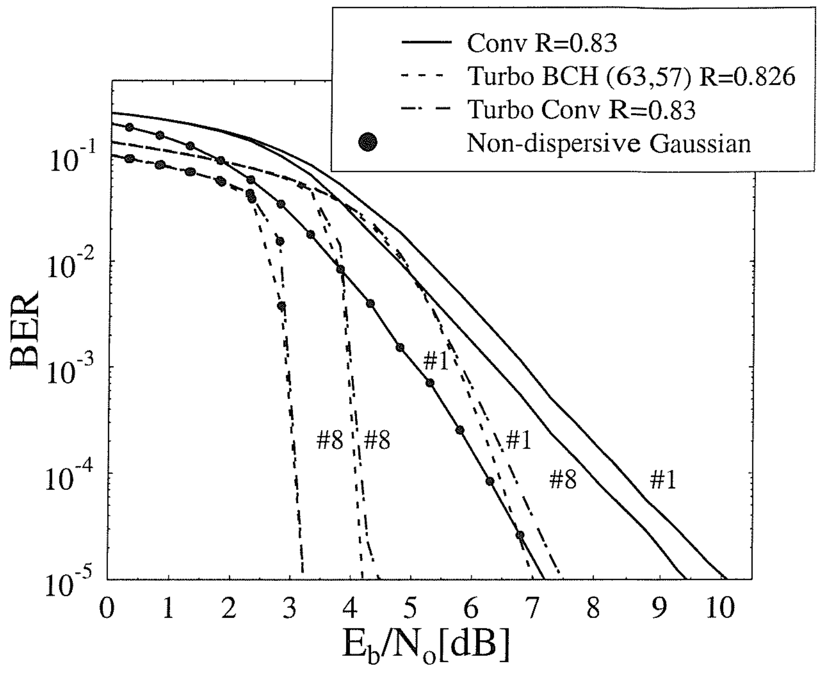
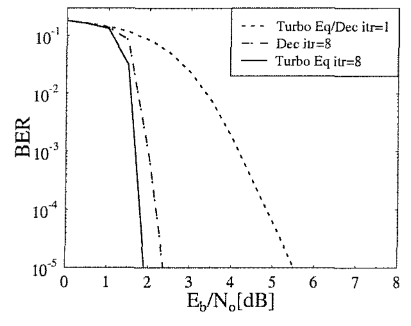
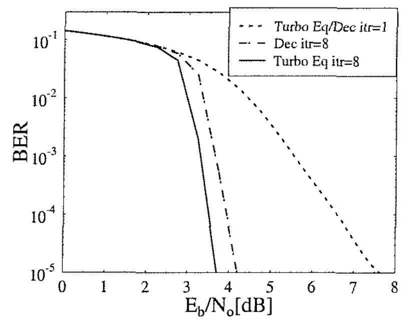


Figure 5.10: Comparing the BPSK turbo equalisation performance of the $R = \frac{5}{6}$ **CC** system, the $R = \frac{5}{6}$ **CT** scheme and the $R = \frac{57}{69}$ **BT** system for one (#1) and eight (#8) turbo equalisation iterations, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 5.2(b). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

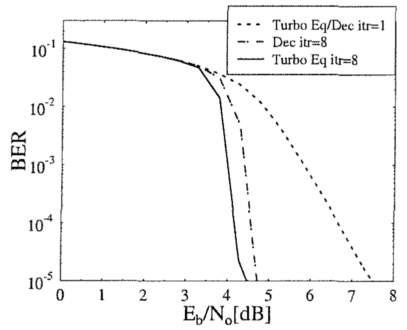
stages — as in turbo equalisation — were implemented. The SISO equaliser and decoders employed the Log-MAP algorithm. Our comparative study of the turbo equalisers for the **BT**, **CT**, **CC** systems showed that at high code rates of $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ the **BT** system is marginally better or comparable to the **CT** system at BER $= 10^{-4}$, at the expense of a higher complexity compared to the **CT** system. At these high code rates and over the equally-weighted symbol-spaced five-path Rayleigh fading channel using burst-invariant fading of Figure 5.2(b), the **CC** system performs poorly since its iteration gain is low. This is because after the first iteration the system's performance is already close to the decoding results over the non-dispersive Gaussian channel. At $R = \frac{5}{6}$ we observed a loss of $E_b/N_o = 1.0$ dB over the five-path Gaussian channel and a loss of $E_b/N_o = 3.8$ dB, when compared to the **BT** system over the five-path Rayleigh fading channel at BER $= 10^{-4}$ after eight turbo equalisation iteration. On the whole, the turbo-equalised **CT** system is the most robust scheme, giving comparable performance within a few tenths of a dB for all code rates investigated, compared to the best system in each scenario. Furthermore,



(a) $R = \frac{1}{2}$

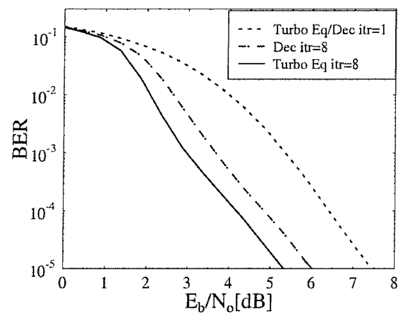


(b) $R = \frac{3}{4}$

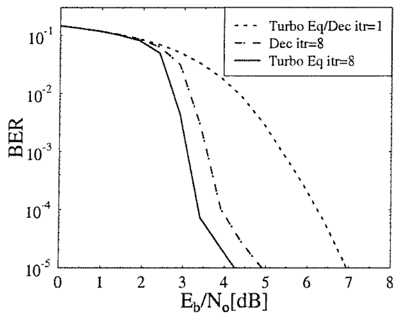


(c) $R = \frac{5}{6}$

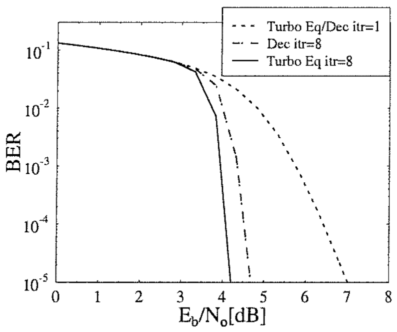
Figure 5.11: Decoding performance of the BPSK **CT** system using isolated turbo decoding compared with the turbo equalisation performance after the first iteration — which is identical for both — and after the eighth iteration for code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 5.2(b).



(a) $R = 0.579$



(b) $R = 0.722$



(c) $R = 0.826$

Figure 5.12: Decoding performance of the BPSK **BT** system using isolated turbo decoding compared with the turbo equalisation performance after the first iteration — which is identical for both — and after the eighth iteration for code rates of $R = \frac{11}{19}$, $R = \frac{26}{36}$ and $R = \frac{57}{69}$, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 5.2(b).

the turbo-equalised **CT** system has a lower receiver complexity, when compared to the **BT** system, hence making it the best choice in most applications.

Chapter 6

Reduced Complexity Trellis-based Equaliser

In a communications system, the received signal is degraded by ISI introduced by the channel. We can mitigate the effects of ISI by employing equalisation and further reduce the bit error rate by using error correction/decoding schemes. When performing the equalisation and decoding independently, we cannot compensate for the loss due to the ISI effectively, even when soft decisions are passed from the equaliser to the decoder. Instead, by performing the equalisation and decoding jointly, as in the iterative turbo equalisation scheme proposed by Douillard *et al* [10], the channel ISI can be mitigated effectively. Knickenberg *et al* [29] subsequently proposed a non-iterative joint equalisation and decoding technique based on a supertrellis structure. This technique yielded an optimum performance but was restricted to simple interleavers due to the high complexity incurred by high-depth interleavers. Therefore, Radial Basis Function (RBF) equalisers, which incur lower computational complexity as compared to conventional trellis-based equalisers, have been researched [31] in the context of turbo equalisation. Other reduced complexity equalisers for turbo equalisation have also been proposed by Glavieux *et al* [93].

6.1 Motivation

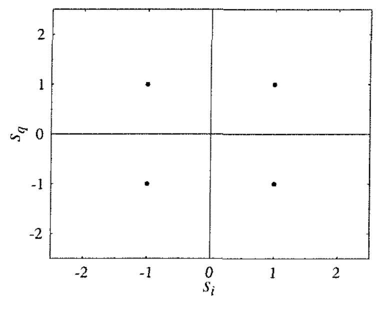
This chapter characterises turbo equalisation implemented in the context of multi-level full response modulation schemes, namely in M-level Quadrature Amplitude Modulation (M-QAM) [6], where M denotes the number of constellation points in a particular modulation mode. However, due to its complexity, turbo equalisation using trellis-based equalisers can only be realistically applied to BPSK and QPSK modulation schemes [66], since the computational complexity incurred by the trellis-based equaliser is dependent on the length of the Channel Impulse Response (CIR) and on the modulation mode utilised. Explicitly, the number of states in the M-QAM trellis-based equaliser is proportional to M^{τ_d}, where τ_d is the maximum CIR duration expressed in terms of symbol periods. Consequently, a high complexity can be incurred for long CIR durations and higher order modulation modes. Hence, turbo equalisation research has been focused on developing reduced complexity equalisers, such as the receiver structure proposed by Glavieux *et al* [93]. Motivated by these trends, we propose a turbo equaliser, which employs a novel reduced complexity trellis-based scheme, in order to allow the implementation of turbo equalisation in the context of higher-order modulation modes, such as 16-QAM and 64-QAM. We refer to this equaliser as the In-Phase/Quadrature-phase Equaliser (I/Q EQ). The basic principle of the reduced complexity equaliser is based on equalising the in-phase and quadrature-phase component of the transmitted signal independently. Therefore, the number of states for the in-phase and quadrature-phase trellis-based equaliser is reduced, when compared to the afore-mentioned trellis-based equaliser. This is made more explicit in our forthcoming discourse.

The outline of this chapter is as follows. Section 6.2 provides a brief introduction to the principles of constructing trellis-based equalisers and to the associated complexity. Subsequently, the operations of the reduced complexity equaliser are described in Section 6.3. Section 6.4 then summarises the system parameters followed by the system performance in Section 6.5. Finally, Section 6.6 provides discussions and concluding remarks concerning the systems' performance.

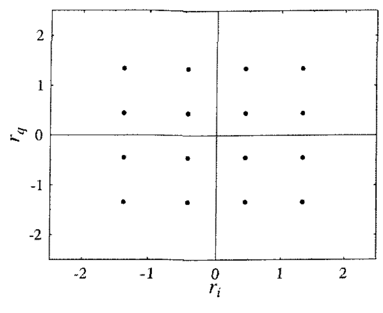
6.2 Complexity of the Multi-level Full Response Turbo Equaliser

The original turbo equaliser proposed by Douillard *et al* [10] for BPSK transmission can be extended to M-QAM transmissions by modifying the trellis-based equaliser. This equaliser accepts soft inputs and produces soft outputs, which reflect the reliability of the equalised encoded bits. Hence, such equalisers are also known as the Soft-In Soft-Out (SISO) trellis-based equalisers. In our forthcoming discussions, we will provide a brief introduction to the principles of constructing a M-QAM equaliser trellis and subsequently highlight the relationship between the equaliser complexity, the modulation mode as well as the channel's memory.

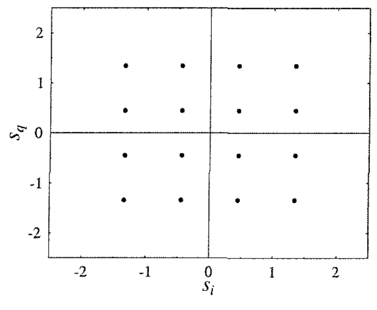
Figure 6.1(a) shows the transmitted constellation points of the 4-QAM signal using Cartesian coordinates, whereas Figure 6.1(b) depicts the corresponding signal constellation



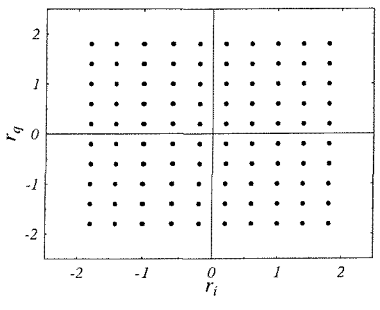
(a) 4-QAM signal constellation



(b) Signal constellation after the 4-QAM symbols are transmitted over the noiseless two-path Gaussian channel



(c) 16-QAM signal constellation



(d) Signal constellation after the 16-QAM symbols are transmitted over the noiseless two-path Gaussian channel.

Figure 6.1: Plotted 4-QAM and 16-QAM signal constellation consisting of in-phase and quadrature-phase components $s_I(t)$ and $s_Q(t)$, respectively, is expanded after transmission over the noiseless two-path Gaussian channel of Figure 6.2, as a consequence of the memory in the channel. The received in-phase and quadrature-phase components at the channel output are denoted by r_I and r_Q, respectively and since due to the two-path channel's memory now two consecutive symbols determine each constellation point. Therefore, the number of received points is now given by all possible combinations of two consecutive symbols, *i.e.* by M^2. Note, however that in Figure 6.1(d) only 100 points — instead of $16^2 = 256$ points — are visible, since some points happen to coincide due to the regularity of the constellation.

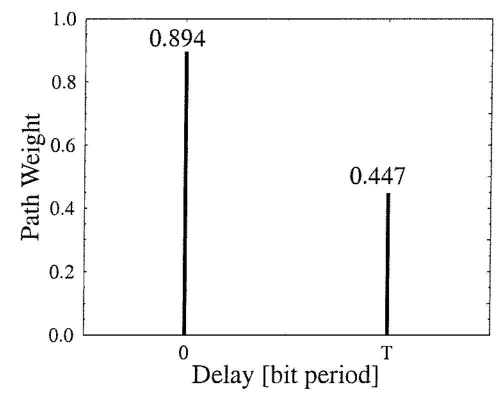


Figure 6.2: The impulse response of a two-path static channel.

received over the two-path static channel illustrated in Figure 6.2. Similarly, when the 16-QAM signal constellation of Figure 6.1(c) is transmitted over the same dispersive Gaussian channel of Figure 6.2, the number of constellation points is determined by the number of all possible combinations of the two consecutive symbols, *i.e.* M^2, although some points happen to coincide as in Figure 6.1(d) due to the regularity of the constellation. This can be explained by observing firstly that when the channel is non-dispersive, the output of the channel is only dependent on the transmitted symbol at that time instant. However, when there is channel dispersion, the output of the channel becomes dependent on not only the current transmitted symbol, but also on the previously transmitted symbol. Hence, the number of constellation points becomes M^2 over a two-path channel as shown in Figure 6.1(b). Again, in Figure 6.1(d) only 100 points are visible since some points coincide for the specific CIR of Figure 6.2. In order to obtain a more quantitative explanation, the design and construction of the trellis-based equaliser is briefly discussed.

Previously in Section 1.8 we described, how the estimated received signal at each time instant can be regenerated, when we have the knowledge of the current and previous τ_d partial response symbols, as well as the estimated channel impulse response. Applying the same principles here, we can construct a trellis, which can regenerate all possible received signals for each time instant. At instant m, information on the previous τ_d number of symbols $s_{m-\tau_d} \dots s_{m-1}$ and the current symbol s_m is embedded in the states of the trellis and the state transitions, respectively. The symbol sequence $s_{m-\tau_d} \dots s_{m-1}, s_m$, which is associated with each state transition, is convolved with the estimated channel impulse response in order to yield the received signal estimate at instant m. Since there are M possible symbols in the M-QAM constellation, M^{τ_d} states are required to represent all combinations of the previous τ_d number of symbols. Furthermore, since there are M transitions leaving each state, the total number of possible received signal constellation points is $\mathrm{M}^{\tau_d} \times \mathrm{M} = \mathrm{M}^{\tau_d+1}$. For 4-QAM symbols transmitted over the dispersive channel of Figure 6.2, the resulting number of possible received constellations points is $\mathrm{M}^{\tau_d+1} = 4^{1+1} = 16$, as shown in Figure 6.1(b), since $\tau_d = 1$ symbol period. Similarly, when transmitting 16-QAM symbols over this channel, the maximum number of signal constellation points is $16^{1+1} = 256$. By studying the trellis construction, we observe that the number of states increases exponentially with the maximum CIR duration τ_d. Consequently, the practical implementation of trellis-based equalisers for higher-order full response modulation over long dispersive channels is not feasible. In order to implement a practical turbo equaliser for M-QAM, the complexity of the SISO equaliser must be reduced. The next section presents a novel turbo equaliser structure, employing two reduced complexity SISO equalisers and a single channel decoder.

6.3 In-phase/Quadrature-phase Equaliser Principle

When a signal $s(t)$, which possesses an in-phase component $s_I(t)$ and quadrature-phase component $s_Q(t)$ is transmitted over a complex channel $h(t)$, the resultant received signal $r(t)$ is given by:

$$\begin{aligned} r(t) &= s(t) * h(t) \\ &= [s_I(t) + js_Q(t)] * [h_I(t) + jh_Q(t)] \\ &= [s_I(t) * h_I(t) - s_Q(t) * h_Q(t)] + j[s_I(t) * h_Q(t) + s_Q(t) * h_I(t)] \\ &= r_I(t) + jr_Q(t), \end{aligned} \tag{6.1}$$

where

$$\begin{aligned} r_I(t) &= s_I(t) * h_I(t) - s_Q(t) * h_Q(t) \\ r_Q(t) &= s_I(t) * h_Q(t) + s_Q(t) * h_I(t). \end{aligned} \tag{6.2}$$

These equations are modelled and illustrated in Figure 6.3, where the in-phase and quadrature-phase components of the received signal, namely $r_I(t)$ and $r_Q(t)$, become dependent on $s_I(t)$ and $s_Q(t)$ after transmission over the complex channel. We refer to the cross-correlation between $s_I(t)$ and $s_Q(t)$ in the received quadrature signals $r_I(t)$ and $r_Q(t)$ as **cross-coupling**. This cross-coupling of the transmitted signal's quadrature components requires the receiver to consider more signal combinations, hence requiring a high number of equaliser trellis states. Therefore, we can reduce the number of states significantly when the cross-coupling is removed such that the quadrature components of the channel

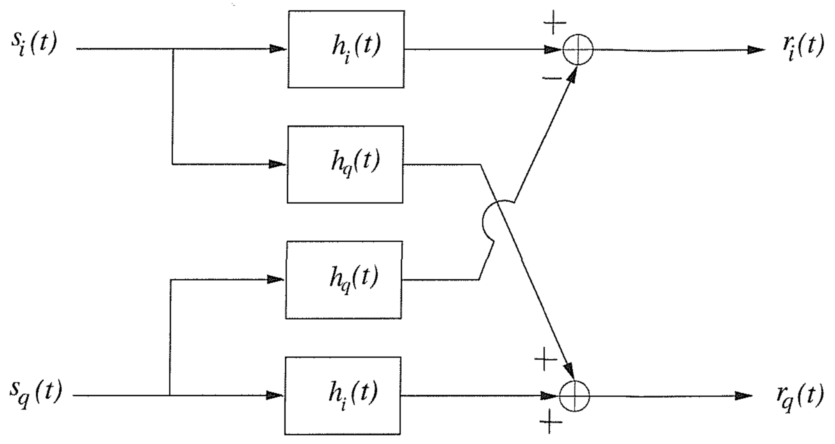


Figure 6.3: Model of the complex, dispersive channel. After transmission over the complex channel, the received signal $r(t)$ becomes dependent on the in-phase component $s_I(t)$ and quadrature-phase component $s_Q(t)$ of the transmitted signal, as expressed in Equations 6.1 and 6.2.

output $r(t)$ are only dependent on $s_I(t)$ or $s_Q(t)$ solely, as illustrated in Figure 6.4. This decoupling operation will be elaborated on in Section 6.3.4. When decoupled, each of the modified complex channel output signals, namely $r'_I(t)$ and $r'_Q(t)$, respectively, can be perceived to be the consequence of convolving a real transmitted signal constituted by one of the quadrature components with a complex channel on each quadrature arm, which is expressed as:

$$\begin{aligned} r'_I(t) &= s_I(t) * h(t) \\ &= s_I(t) * h_I(t) + j(s_I(t) * h_Q(t)) \\ r'_Q(t) &= s_Q(t) * h(t) \\ &= s_Q(t) * h_I(t) + j(s_Q(t) * h_Q(t)). \end{aligned} \tag{6.3}$$

Note that after decoupling the signals $r'_I(t)$ and $r'_Q(t)$ constitute the received signal $r(t)$. Consequently, we can equalise the in-phase signal quadrature-phase signals $s_I(t)$ and $s_Q(t)$ independently, hence reducing significantly the number of states in the trellis. Explicitly, one I/Q EQ is assigned to equalise the $r'_I(t)$ signal, in order to obtain soft decisions for $s_I(t)$ and similarly, another I/Q EQ is employed for providing soft decisions concerning $s_Q(t)$. Instead of considering all M signal constellation points, the I/Q EQs only consider the $\sqrt{M}$ number of points of the in-phase and quadrature-phase components of the M-QAM signal. In our following discussion an overview of the above-mentioned reduced complexity turbo equaliser is given. Subsequently, the main operations are described in greater detail and finally the complexity of the I/Q EQ is quantified as a function of the modulation mode and the maximum CIR duration τ_d.

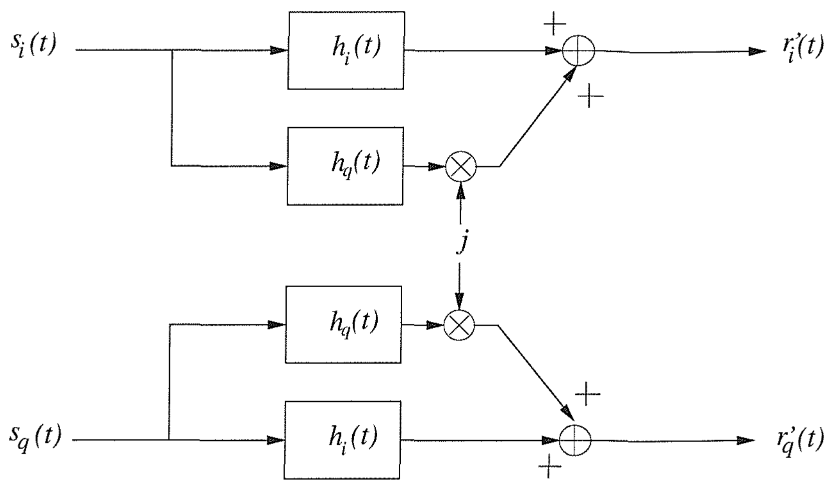


Figure 6.4: Removing the dependency of $r_I(t)$ and $r_Q(t)$ on the quadrature components of the transmitted signal, namely $s_I(t)$ and $s_Q(t)$, to give $r'_I(t)$ and $r'_Q(t)$, respectively. This modified complex channel outputs, namely $r'_I(t)$ and $r'_Q(t)$, respectively, can be viewed as the result of convolving both quadrature components independently with a complex channel on each quadrature arm.

6.3.1 Overview of the Reduced Complexity Turbo Equaliser

Figure 6.5 — which is discussed in detail throughout this section — illustrates the schematic of the turbo equaliser utilising two reduced complexity SISO equalisers. Since the Log-MAP algorithm of Section 2.5 is employed in the SISO equaliser and in the decoder blocks, the soft decisions generated are in the form of LLRs. As before in Section 3.2, we have employed the approach used by Gertsman and Lodge [66] and expressed the LLR of the equaliser and decoder using vector notations. The superscript denotes the nature of the LLR, namely 'c' is used for the composite *a posteriori* information, 'i' for the combined channel and extrinsic information and 'e' for the extrinsic information. Furthermore, the subscript is used to represent the iteration index, while the argument within the brackets ( ) indicates the receiver stage. The equalisers are denoted as stage 0, while the decoder as stage 1. Furthermore, in our discussions related to multi-level QAM, the term **bit** refers to either the +1 or −1 bit of the M-QAM **symbols**. For 4-QAM, there are two bits in a symbol, whereas a 16-QAM symbol consists of four bits.

As seen in Figure 6.5, in the first turbo equalisation iteration a conventional Decision Feedback Equaliser (DFE) is employed, in order to provide soft decisions in the form of the LLR $L_1^c(0)$ to the decoder. The DFE was employed at the first iteration, since it constitutes a low-complexity approach to providing initial estimates of the transmitted

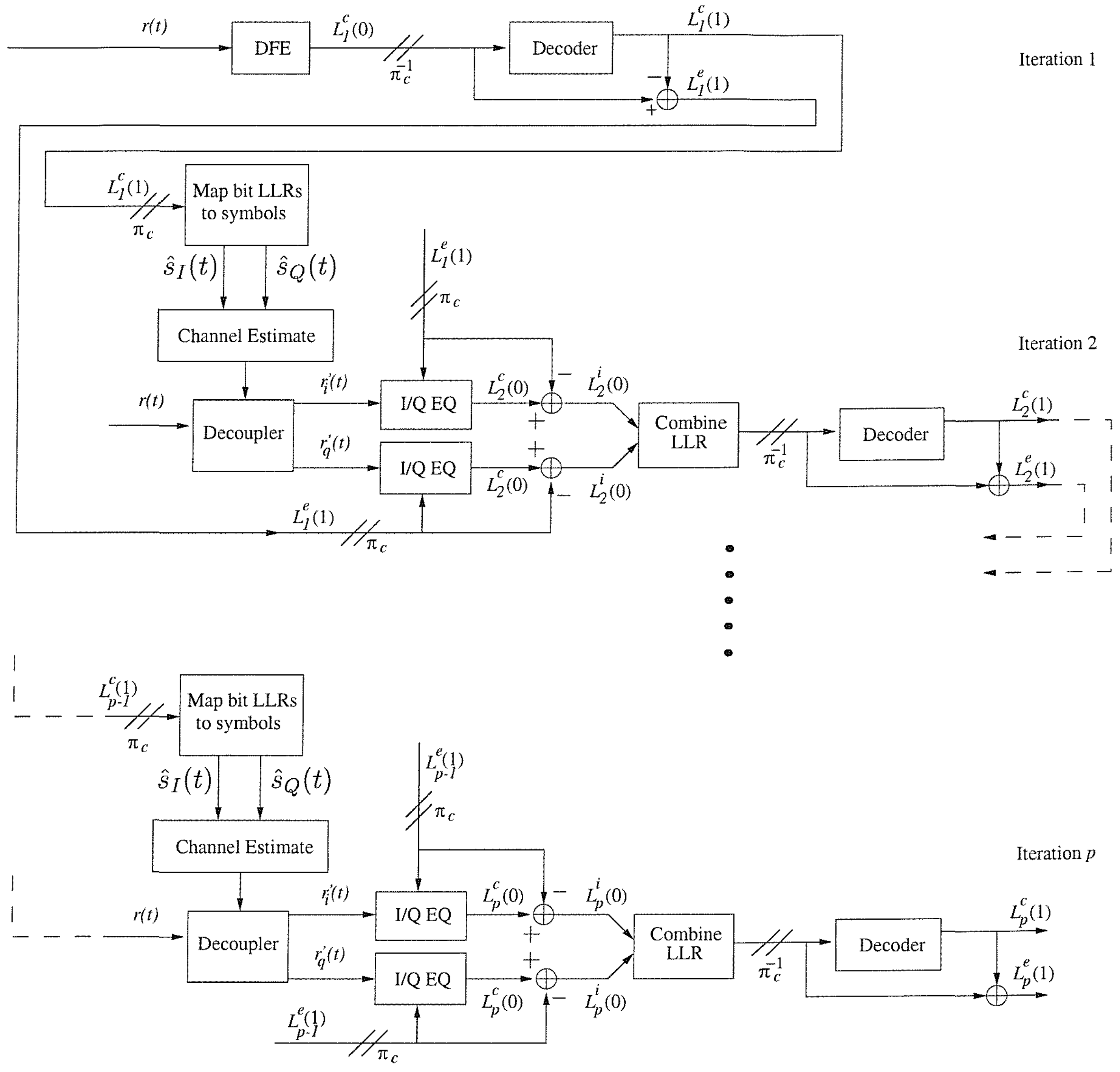


Figure 6.5: Schematic of the turbo equaliser employing a Decision Feedback Equaliser (DFE) and a SISO channel decoder in the first turbo equalisation iteration. In subsequent iterations, two reduced complexity SISO In-phase/Quadrature-phase Equalisers (I/Q EQ) and one SISO channel decoder is employed. The notation π_c represents a channel interleaver, while π_c^{-1} is used to denote a channel deinterleaver.

symbols, which were required in the decoupling operation of Figure 6.4 and Equation 6.3. Conventional trellis-based equalisers could not be employed due to the associated high number of trellis states. The SISO channel decoder of Figure 6.5 then computes the *a posteriori* LLR $L_1^c(1)$ and subsequently the extrinsic information of the encoded bits, namely $L_1^e(1)$ is extracted. In the next iteration, the *a posteriori* LLR $L_1^c(1)$ is used to regenerate estimates of the in-phase and quadrature-phase components of the transmitted signal, namely $\hat{s}_I(t)$ and $\hat{s}_Q(t)$ as seen in the 'MAP bit LLRs to symbols' block of Figure 6.5. The estimated transmitted quadrature components are then convolved with the estimate of the channel impulse response $h(t)$ and the resultant signals are directed to the decoupler block of Figure 6.5. As mentioned previously, the cross-coupling between $s_I(t)$ and $s_Q(t)$ in each quadrature arm of the received signal $r(t)$ results in the need for a large number of equaliser trellis states. Therefore, in order to reduce the number of trellis states, the cross-coupling of $s_I(t)$ and $s_Q(t)$ in each quadrature branch of $r(t)$ is mitigated by the decoupler. This results in $r'_I(t)$ and $r'_Q(t)$ of Equation 6.3, which are formed by the channel impulse response $h(t)$, the in-phase component of the transmitted signal $s_I(t)$ and the quadrature-phase of the transmitted signal $s_Q(t)$. As shown in Figure 6.5, after the decoupling operation, the signals $r'_I(t)$ and $r'_Q(t)$ are passed to the I/Q EQ. In addition to these received quadrature signals, the I/Q EQ also processes the *a priori* information received — which is the extrinsic LLR from the previous iteration *i.e.* $L_1^e(1)$ — and generates the *a posteriori* information $L_2^c(0)$. Subsequently, the combined channel and extrinsic information $L_2^i(0)$ is extracted from both I/Q EQs in Figure 6.5, multiplexed and combined, before being passed to the Log-MAP decoder. As in the first turbo equalisation iteration, the *a posteriori* and extrinsic information of the encoded bits, namely $L_2^c(1)$ and $L_2^e(1)$, respectively, are evaluated. The following turbo equalisation iterations also obey the same operation sequence, until the iteration termination criterion used is met.

Let us now discuss in greater detail the following aspects of the schematic in Figure 6.5:

- Conversion of the DFE symbol estimates to LLR.
- Conversion of the decoder *a posteriori* LLR into symbols.
- Decoupling operation.
- I/Q EQ complexity.

6.3.2 Conversion of the DFE Symbol Estimates to LLR

We commence by describing the DFE symbol-to-LLR conversion for a specific example, namely for 4-QAM, in order to highlight its salient steps, before generalising the concept.

The associated DFE block can be found at the top left corner of Figure 6.5 at iteration 1. The output of the DFE is the estimate $\tilde{s}(t)$ of the transmitted symbol at time instant t. The squared Euclidean distance between the DFE symbol estimate $\tilde{s}(t)$ and the individual

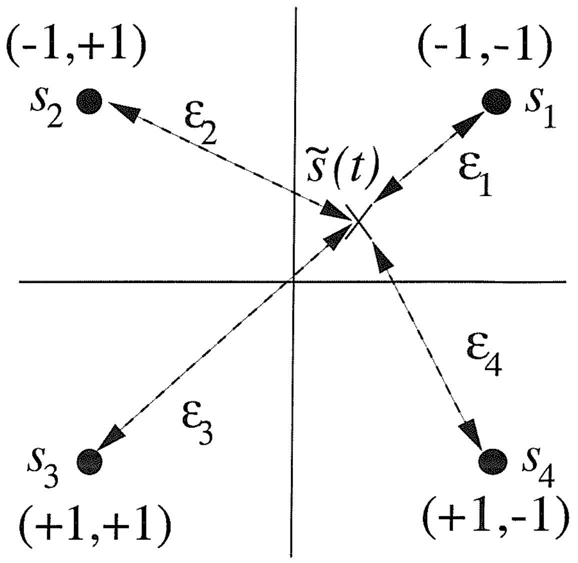


Figure 6.6: The squared Euclidean distance ε_1, ε_2, ε_3 and ε_4 between the DFE symbol estimate $\tilde{s}(t)$ and the 4-QAM signal constellation points s_1, s_2, s_3 and s_4. The values in the brackets ( ) are the bits of the 4-QAM symbol.

4-QAM constellation points, namely s_1, s_2, s_3 and s_4 at instant t can be evaluated as:

$$\varepsilon_j = |\tilde{s}(t) - s_j|^2 \qquad j = 1, 2 \ldots \mathrm{M} = 4, \tag{6.4}$$

where M is the number of constellation points, which is depicted in Figure 6.6. Note that in Figure 6.6 the values in the brackets ( ) represent the bits of the 4-QAM symbol. Subsequently, the probability that symbol s_j, $j = 1, 2 \ldots \mathrm{M} = 4$, was transmitted given that $\tilde{s}(t)$ was received can be evaluated by using [93]:

$$P[s_j|\tilde{s}(t)] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-\varepsilon_j}{2\sigma^2}\right) \qquad j = 1, 2 \ldots \mathrm{M} = 4, \tag{6.5}$$

where M is the number of constellation points. Before we proceed, we define new notations, in order to represent each element of the decoder's *a posteriori* LLR vector $L_1^c(1)$. Information related to the time instant m and to bit b_k of a symbol, where k is the bit index, is introduced to give $L^c(m;k)$ for all bits of all symbols, which is formulated as:.

$$L_1^c(0) = [L^c(m;k=1); L^c(m;k=2)] \qquad m = 1, 2 \ldots B_L, \tag{6.6}$$

where B_L is the length of the coded block, while the maximum value of k in our example is two for 4-QAM. Subsequently, having determined the probability of the transmitted symbols

s_1, s_2, s_3 and s_4, we can now compute the LLR of each transmitted bit $L^c(m;k)$ at instant m, since

$$L^c(m;k) = \ln\left(\frac{P[b_k=+1|\tilde{s}(t)]}{P[b_k=-1|\tilde{s}(t)]}\right) = \ln\left(\frac{\sum_{\forall s_j \text{where } b_k=+1} P[s_j|\tilde{s}(t)]}{\sum_{\forall s_j \text{where } b_k=-1} P[s_j|\tilde{s}(t)]}\right), \tag{6.7}$$

where b_k is the kth bit of the M-QAM symbol. Explicitly, the numerator is the sum of all probabilities $P[s_j|\tilde{s}(t)]$, where the kth bit of the symbol s_j is $+1$. For the 4-QAM scenario we observe in Figure 6.6 that, when $k=1$ this corresponds to signal points s_3 and s_4, while to s_2 and s_3, when $k=2$. For the denominator, we sum all the probabilities $P[s_j|\tilde{s}(t)]$, which correspond to the kth bit of the symbol being -1. From Figure 6.6, we observed that when $k=1$, the summation of probabilities $P(s_1|\tilde{s}(t))$ and $P(s_2|\tilde{s}(t))$ is performed, while for $k=2$ $P(s_1|\tilde{s}(t))$ is added to $P(s_4|\tilde{s}(t))$. Therefore, at instant m the equaliser *a posteriori* LLR $L^c(m;k=1)$ for the $(k=1)$st bit is given by:

$$L^c(m;k=1) = \ln\left(\frac{P[s_3|\tilde{s}(t)] + P[s_4|\tilde{s}(t)]}{P[s_1|\tilde{s}(t)] + P[s_2|\tilde{s}(t)]}\right), \tag{6.8}$$

while for the $(k=2)$nd bit, we obtain:

$$L^c(m;k=2) = \ln\left(\frac{P[s_2|\tilde{s}(t)] + P[s_3|\tilde{s}(t)]}{P[s_1|\tilde{s}(t)] + P[s_4|\tilde{s}(t)]}\right). \tag{6.9}$$

The symbol estimates at the output of the DFE for the other modulation modes can be easily converted into the LLR form by substituting the value of M in Equations 6.4 and 6.5 for the corresponding modulation mode and using Equation 6.7. Having described the conversion of the DFE symbol estimates to LLRs, we now proceed with the discussion of the LLR conversion to modulated symbols.

6.3.3 Conversion of the Decoder *A Posteriori* LLRs into Symbols

Let us now highlight the operation of the top left block of Figure 6.5 at iteration 2. There are two potential approaches, which can be employed to convert information from LLRs into symbols. The first technique is based on the inverse of the conversion of DFE symbol estimates to LLRs — which was described in the previous subsection — while the second approach determines the statistical average of the in-phase and quadrature-phase signals, denoted as $\hat{s}_I(t)$ and $\hat{s}_Q(t)$, respectively, by using [94]:

$$\begin{aligned} E\{s_I(t)\} = \hat{s}_I(t) &= \sum_{i=0}^{\sqrt{\mathrm{M}}-1} s_{I;i} \cdot P[s_{I;i}|\tilde{s}(t)] \\ E\{s_Q(t)\} = \hat{s}_Q(t) &= \sum_{i=0}^{\sqrt{\mathrm{M}}-1} s_{Q;i} \cdot P[s_{Q;i}|\tilde{s}(t)], \end{aligned} \tag{6.10}$$

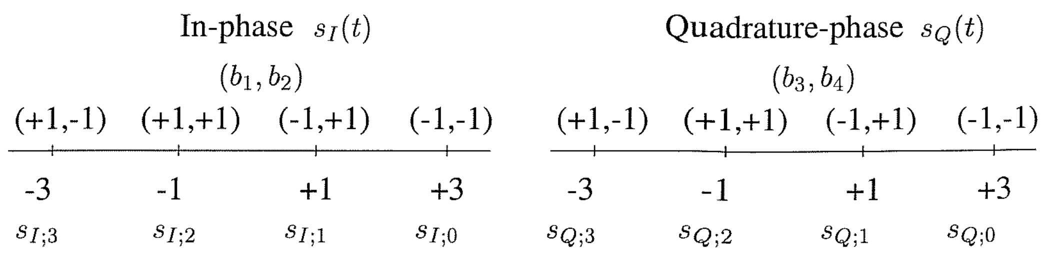


Figure 6.7: The Gray mapping [95] of the 16-QAM mode depicting the in-phase and quadrature-phase components and the corresponding bit assignments.

where $E\{\cdot\}$ denotes the expectation or averaging operation and M is the number of constellation points in a particular modulation mode. The terms $s_{I;i}$ and $s_{Q;i}$ $i = 0 \ldots \sqrt{\mathrm{M}} - 1$ represent the ith in-phase signal and the ith quadrature-phase signal, as illustrated in Figure 6.7 for 16-QAM. In attempting to determine the estimated symbol from the LLR values, the former technique involves solving second-order equations arising from the squared Euclidean distance term in Equation 6.4. Therefore, there are more than one possible symbol estimates at each time instant. The latter approach provides the symbol estimates based on the probabilities of symbols, which can be extracted from the LLR values. Since there is no ambiguity in the latter statistical approach — in the contrast to the inverse process of the symbols to LLR conversion — we will adopt this technique, in order to translate LLR values to symbol estimates.

In order to highlight the associated operations, let us consider a particular example, namely 16-QAM, before generalising the concept. Each 16-QAM symbol $s(t)$ can be represented by four bits, namely by b_1, b_2, b_3 and b_4. Let us assume that the first two bits (b_1, b_2) correspond to a coordinate on the in-phase axis $s_I(t)$ of the signal constellation, while the other two bits (b_3, b_4) represent a point on the quadrature-phase axis $s_Q(t)$, as depicted in Figure 6.7. Hence, the 16-QAM symbol can be expressed as a combination of the coordinates $s_I(t)$ and $s_Q(t)$:

$$s(t) = s_I(t) + j s_Q(t). \tag{6.11}$$

With reference to Figure 6.7, which illustrates the Gray mapping used in each quadrature arm of the 16-QAM signal constellation and by employing Equation 6.10, the regenerated quadrature components of $s(t)$, namely $\hat{s}_I(t)$ and $\hat{s}_Q(t)$ can be expressed as:

$$\begin{aligned}\hat{s}_I(t) &= 3{\cdot}P[s_{I;0}|\tilde{s}(t)] + 1{\cdot}P[s_{I;1}|\tilde{s}(t)] - 1{\cdot}P[s_{I;2}|\tilde{s}(t)] - 3{\cdot}P[s_{I;3}|\tilde{s}(t)] \\ &= 3{\cdot}P[b_1 = -1; b_2 = -1|\tilde{s}(t)] + 1{\cdot}P[b_1 = -1; b_2 = +1|\tilde{s}(t)] - \\ &\quad 1{\cdot}P[b_1 = +1; b_2 = +1|\tilde{s}(t)] - 3{\cdot}P[b_1 = +1; b_2 = -1|\tilde{s}(t)],\end{aligned} \tag{6.12}$$

and

$$\begin{aligned}\hat{s}_Q(t) &= 3\cdot P[s_{Q;0}|\tilde{s}(t)] + 1\cdot P[s_{Q;1}|\tilde{s}(t)] - 1\cdot P[s_{Q;2}|\tilde{s}(t)] - 3\cdot P[s_{Q;3}|\tilde{s}(t)] \\ &= 3\cdot P[b_3 = -1; b_4 = -1|\tilde{s}(t)] + 1\cdot P[b_3 = -1; b_4 = +1|\tilde{s}(t)] - \\ &\quad 1\cdot P[b_3 = +1; b_4 = +1|\tilde{s}(t)] - 3\cdot P[b_3 = +1; b_4 = -1|\tilde{s}(t)],\end{aligned} \tag{6.13}$$

where the probabilities $P[s_{I;i}|\tilde{s}(t)]$ and $P[s_{Q;i}|\tilde{s}(t)]$ for $i = 0 \ldots \sqrt{\mathrm{M}} - 1$ can be written as $P[b_1; b_2|\tilde{s}(t)]$ and $P[b_3; b_4|\tilde{s}(t)]$, respectively, since the symbol $s_{I;i}$ corresponds to bits b_1 and b_2, while the symbol $s_{Q;i}$ is represented by bits b_3 and b_4. Note also that $P[b_1; b_2|\tilde{s}(t)]$ can be expressed as $P[b_1|\tilde{s}(t)]\cdot P[b_2|\tilde{s}(t)]$, since the transmission of bits b_1 and b_2 are statistically independent events. Similarly, the probability $P[b_3; b_4|\tilde{s}(t)]$ can be written as $P[b_3|\tilde{s}(t)] \cdot P[b_4|\tilde{s}(t)]$, yielding

$$\begin{aligned}\hat{s}_I(t) &= 3\cdot P[b_1 = -1|\tilde{s}(t)]\cdot P[b_2 = -1|\tilde{s}(t)] + 1\cdot P[b_1 = -1|\tilde{s}(t)]\cdot P[b_2 = +1|\tilde{s}(t)] - \\ &\quad 1\cdot P[b_1 = +1|\tilde{s}(t)]\cdot P[b_2 = +1|\tilde{s}(t)] - 3\cdot P[b_1 = +1|\tilde{s}(t)]\cdot P[b_2 = -1|\tilde{s}(t)],\end{aligned} \tag{6.14}$$

and

$$\begin{aligned}\hat{s}_Q(t) &= 3\cdot P[b_3 = -1|\tilde{s}(t)]\cdot P[b_4 = -1|\tilde{s}(t)] + 1\cdot P[b_3 = -1|\tilde{s}(t)]\cdot P[b_4 = +1|\tilde{s}(t)] - \\ &\quad 1\cdot P[b_3 = +1|\tilde{s}(t)]\cdot P[b_4 = +1|\tilde{s}(t)] - 3\cdot P[b_3 = +1|\tilde{s}(t)]\cdot P[b_4 = -1|\tilde{s}(t)].\end{aligned} \tag{6.15}$$

Therefore, it is possible to infer the associated 16-QAM symbols, once the probabilities $P[b_k|\tilde{s}(t)]$ of the bits in the symbol are known. This can be extracted from the decoder's *a posteriori* LLR $L_1^c(1)$ in Figure 6.5, since the LLR reflects the confidence associated with the assumption of the transmitted bit being either +1 or −1 in the form of a ratio, as expressed in Equation 6.7. Since $L^c(m; k)$ can be expressed as:

$$L^c(m; k) = \ln\left(\frac{P[b_k = +1|\tilde{s}(t)]}{P[b_k = -1|\tilde{s}(t)]}\right), \tag{6.16}$$

the probability that $b_k = +1$ was transmitted at instant m given that $\tilde{s}(t)$ was estimated by the DFE, namely $P[b_k = +1|\tilde{s}(t)]$ can be expressed after a few steps as:

$$P[b_k = +1|\tilde{s}(t)] = \frac{\exp(L^c(m; k))}{1 + \exp(L^c(m; k))}. \tag{6.17}$$

Subsequently, using the relationship that $P[b_k = +1|\tilde{s}(t)] + P[b_k = -1|\tilde{s}(t)] = 1$, the term $P[b_k = -1|\tilde{s}(t)]$ becomes:

$$P[b_k = -1|\tilde{s}(t)] = \frac{1}{1 + \exp(L^c(m; k))}. \tag{6.18}$$

We now have an expression for the kth bit's probability in the mth symbol, which can be extracted from the decoder's LLRs by substituting Equations 6.17 and 6.18 into Equations 6.14 and 6.15. At this stage, the probabilities in Equations 6.10 and 6.12-6.18 have been conditioned upon the DFE symbol estimates $\tilde{s}(t)$, since the DFE is utilised and LLR values

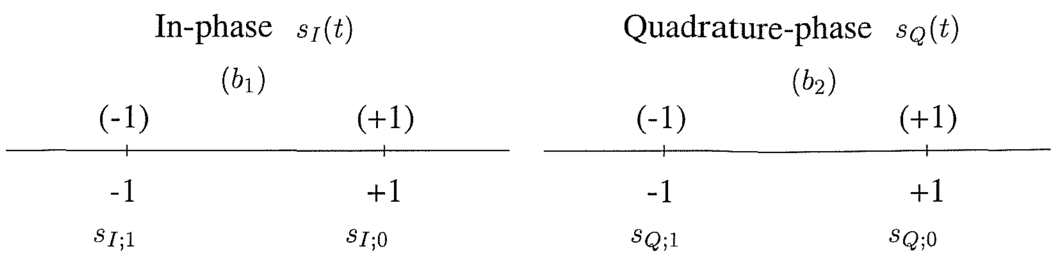


Figure 6.8: The Gray mapping [95] of the 4-QAM mode depicting the in-phase and quadrature-phase components and the corresponding bit assignments.

are generated based on the DFE symbol estimates. However, when I/Q EQs are employed instead of the DFE, the probability terms in these equations will be conditioned upon the received signal $r(t)$. Closer inspection of Equations 6.17 and 6.18 shows that the maximum value of the $\tilde{s_I}(t)$ and $\tilde{s_Q}(t)$ components is $+3$, when $P[b_1 = -1|\tilde{s}(t)]{\cdot}P[b_2 = -1|\tilde{s}(t)] = 1$ and $P[b_3 = -1|\tilde{s}(t)]{\cdot}P[b_4 = -1|\tilde{s}(t)] = 1$, respectively. Conversely, $\tilde{s_I}(t)$ and $\tilde{s_Q}(t)$ has a minimum value of -3, when $P[b_1 = +1|\tilde{s}(t)]{\cdot}P[b_2 = -1|\tilde{s}(t)] = 1$ and $P[b_3 = +1|\tilde{s}(t)]{\cdot}P[b_4 = -1|\tilde{s}(t)] = 1$, respectively. Therefore, by evaluating the average value of the in-phase and quadrature-phase components, we will obtain a constellation of estimated symbols, which are within the boundaries of the maximum and minimum values of each quadrature arm.

Let us highlight this point more explicitly by considering a simple 4-QAM signal constellation as illustrated in Figure 6.8. Again, by employing Equation 6.10 and with reference to Figure 6.8, which illustrates the Gray mapping used in each quadrature arm of the 4-QAM signal constellation, the regenerated quadrature components of $s(t)$, namely $\hat{s}_I(t)$ and $\hat{s}_Q(t)$ can be expressed as:

$$\hat{s}_I(t) = 1 \cdot P[b_1 = +1|\tilde{s}(t)] - 1 \cdot P[b_1 = -1|\tilde{s}(t)] \tag{6.19}$$

and

$$\hat{s}_Q(t) = 1 \cdot P[b_2 = +1|\tilde{s}(t)] - 1 \cdot P[b_2 = -1|\tilde{s}(t)]. \tag{6.20}$$

Since the expression $P[b_1 = +1|\tilde{s}(t)] + P[b_1 = -1|\tilde{s}(t)] = 1$ is valid, the maximum value of 1 is assumed by $\hat{s}_I(t)$ in Equation 6.19, when $P[b_1 = +1|\tilde{s}(t)] = 1$ and $P[b_1 = -1|\tilde{s}(t)] = 0$. Conversely, a minimum value of -1 is assumed by $\hat{s}_I(t)$, when $P[b_1 = -1|\tilde{s}(t)] = 1$ and $P[b_1 = +1|\tilde{s}(t)] = 0$. Employing the same reasoning as above, the maximum and minimum values of $\hat{s}_Q(t)$ in Equation 6.20 are $+1$ and -1, respectively. Hence, the average symbol values of $\hat{s}_I(t)$ and $\hat{s}_Q(t)$ will never be beyond the boundaries formed by the minimum and maximum values of these quadrature signal estimates, as it will be shown in Figures 6.11, 6.13 and 6.15 for 4-QAM, 16-QAM and 64-QAM, respectively.

We summarise the associated operations described previously and generalise the concept, where possible, in order to allow the conversion of decoder's bit LLR values to M-QAM symbols in the top left block of Figure 6.5 at iteration 2, as follows:

- Initially, determine the Gray mapping for the M-QAM symbol, as shown in Figure 6.7. Knowledge of the mapping allows the in-phase and quadrature-phase components of the symbol, namely $s_I(t)$ and $s_Q(t)$, to be determined as a function of the bit probabilities, as seen in Equations 6.12 and 6.13.
- Subsequently, we evaluate the probability that bit $b_k = +1$ was transmitted at time instant m, given that $\tilde{s}(t)$ was estimated by the DFE, namely $P[b_k = +1|\tilde{s}(t)]$, by using Equation 6.17. Similarly, with the aid of Equation 6.18 we compute the probability $P[b_k = -1|\tilde{s}(t)]$ of $b_k = -1$ being transmitted at instant m, given that $\tilde{s}(t)$ was estimated by the DFE. For 4-QAM, 16-QAM and 64-QAM the maximum values of k are two, four and six, respectively.
- Finally, we substitute the above bit probability values computed into Equations 6.14 and 6.15 in order to obtain the M-QAM symbol.

So far, we have described the conversion of DFE symbol estimates $\tilde{s}(t)$ to the LLR $L_1^c(0)$. This LLR information is then passed to the decoder at iteration 1 in Figure 6.5, which subsequently computes the *a posteriori* LLR $L_1^c(1)$ of the encoded bits. Next, this LLR $L_1^c(1)$ is converted into M-QAM symbols at iteration 2 in Figure 6.5 using the principles described previously. These symbols are then used in conjunction with the channel estimates $h(t)$, in order to remove the cross-coupling between $s_I(t)$ and $s_Q(t)$ in each quadrature branch, such that a lower number of states are required in the equaliser. This decoupling operation is seen at the bottom left corner of Figure 6.5 at iteration 2 and will be the focus of our following discussion.

6.3.4 Decoupling Operation

Recall that the term **cross-coupling** refers to the cross-correlation of $s_I(t)$ and $s_Q(t)$ in the quadrature branches of the received signal, as shown in Equation 6.2 and Figure 6.3. This is the consequence of transmitting the M-QAM signal over a complex channel. The aim of removing the cross-coupling from the quadrature branches of the received signal is to reduce the number of states required by the trellis-based M-QAM equaliser. We will quantify the associated reduction in the number of states at a later stage.

Equation 6.2 is used as our starting point to highlight the decoupling operation, which is repeated here for convenience:

$$\begin{aligned} r_I(t) &= s_I(t) * h_I(t) - s_Q(t) * h_Q(t) \\ r_Q(t) &= s_I(t) * h_Q(t) + s_Q(t) * h_I(t). \end{aligned}$$

With the assumption that we have perfect knowledge of $s_Q(t)$, $h_I(t)$ and $h_Q(t)$ at the receiver, we can then generate the signals $s_Q(t) * h_Q(t)$ and $s_Q(t) * h_I(t)$. Subsequently, we can achieve **perfect decoupling** by removing the contribution of these signals from $r_I(t)$ and $r_Q(t)$, leaving $s_I(t) * h_I(t)$ and $s_I(t) * h_Q(t)$, respectively. Both of these resultant signals can be combined, in order to give $r'_I(t)$ in Equation 6.3, which is also repeated here for convenience:

$$\begin{aligned} r'_I(t) &= s_I(t) * h(t) \\ &= s_I(t) * h_I(t) + j(s_I(t) * h_Q(t)) \\ r'_Q(t) &= s_Q(t) * h(t) \\ &= s_Q(t) * h_I(t) + j(s_Q(t) * h_Q(t)). \end{aligned}$$

Similarly, if we had perfect knowledge of $s_I(t)$, $h_I(t)$ and $h_Q(t)$, we can remove the effects of the signal $s_I(t)$ from $r_I(t)$ and $r_Q(t)$, and subsequently augment the resultant signals to yield $r'_Q(t)$ in Equation 6.3. This is the basis of the decoupling operation at the bottom left corner of Figure 6.5 at iteration 2. In a more realistic scenario, we will not have perfect knowledge of the in-phase and quadrature-phase components of the transmitted signal. Instead, we must operate on the basis of the available information at the receiver. Therefore the previous decoder's *a posteriori* information, which reflects our confidence in whether a $+1$ or -1 bit was transmitted, given that $\tilde{s}(t)$ was estimated by the DFE, is used to extract information concerning $s_I(t)$ and $s_Q(t)$. This conversion process from decoder LLRs to M-QAM symbol estimates was described in the previous section. Subsequently, these symbols are convolved with the channel estimates $h(t)$ — which can be obtained from a multitude of midamble-based channel estimation techniques using the the Least Mean Square algorithm [96] or the Kalman algorithm [96] — and is utilised for the decoupling as seen in Figure 6.5. There will be errors introduced in the decoupling operation, when inaccurate M-QAM symbol estimates are generated from low-confidence LLR values. However, as we will show in our simulation results later on, this is compensated through successive turbo equalisation iterations and the performance approaches that of the turbo equaliser utilising the conventional trellis-based equaliser. In our simulations, we have assumed perfect channel knowledge in order to study the upper bound performance of the turbo equalisation scheme employing the I/Q EQs. We note furthermore that for BPSK transmission no reduction in complexity is achieved by employing the I/Q EQ, since BPSK signalling consists of the in-phase component only, resulting in no cross-coupling between the quadrature components.

Having discussed the decoupling operation, we now proceed to assess the complexity incurred by M-QAM I/Q EQs. As mentioned previously, the decoupling operation produces $r'_I(t)$ and $r'_Q(t)$, which when perfectly decoupled are only dependent on $s_I(t) * h(t)$ and $s_Q(t) * h(t)$, respectively, as expressed in Equation 6.3. These signals, namely $r'_I(t)$ and $r'_Q(t)$, can be perceived as real signals, which have been convolved with the complex channel impulse response $h(t)$. For conceptual simplicity, let us examine the trellis-based I/Q EQ, which equalises the perfectly decoupled signal $r'_I = s_I(t) * h(t)$. Initially, we determined all possible values of the in-phase component $s_I(t)$. For the 4-QAM system $s_I(t)$ can be either $+1$ or -1, whereas for 16-QAM, there are four possible values of $s_I(t)$, namely $s_I(t) = \{-3, -1, +1, +3\}$. When utilising 64-QAM, $s_I(t)$ has eight possible values, namely $s_I(t) = \{-7, -5, -3, -1, +1, +3, +5, +7\}$. Hence, there are $\sqrt{M}$ possible values of $s_I(t)$, where M is the number of constellation points for a particular modulation mode. Recalling the principles of constructing M-QAM trellis states from Section 6.2, all possible combinations of $r'_I(t)$ can be reproduced, when the previous τ_d number of symbols and the current symbol are available, yielding a total of M^{τ_d} states and M transitions emerging from each state. In the I/Q EQ case, the required symbols are values of $s_I(t)$. Hence, the total number of states in the I/Q EQ trellis is $\sqrt{\mathrm{M}}^{\,\tau_d}$ and there are $\sqrt{\mathrm{M}}$ number of transitions leaving each state. Once the trellis states and transitions are determined, any trellis-based Soft-In/Soft-Out algorithm, such as the MAP algorithm or the Log-MAP algorithm of Sections 2.2 and 2.5, respectively, can be implemented. As mentioned before, each signal constellation point $s(t) = s_I(t) + js_Q(t)$ can be represented by using a given combination of bits. For example, the 16-QAM signal constellation can be represented by $K_b = 4$ bits, namely b_1, b_2, b_3 and b_4, where $b_k = \pm 1$ for $k = 1, 2 \ldots K_b = 4$. Let us assume that bits b_1 and b_2 are used to represent $s_I(t)$ while bits b_3 and b_4 are mapped to $s_Q(t)$. Therefore, the SISO I/Q EQ, which equalises $r'_I(t)$ will give the *a posteriori* LLRs of bits b_1 and b_2. The other I/Q EQ, which equalises the signal $r'_Q(t)$ is also based on the same principles. However, in this case, combinations of $s_Q(t)$ are considered instead of $s_I(t)$ and the LLRs of bits b_3 and b_4 are produced. In general, for the M-QAM schemes, the number of bits used to represent a symbol is $K_b = \log_2 \mathrm{M}$. Therefore, the I/Q EQ associated with $r'_I(t)$ will compute the *a posteriori* LLRs of the first $\frac{K_b}{2}$ number of bits, while the other I/Q EQ determines the LLRs of the following $\frac{K_b}{2}$ bits. Subsequently, the

LLRs of both I/Q EQs are multiplexed before being passed to the decoder, in order to ensure that they are rearranged in the right order, *i.e.* in the order of $b_1, \ldots b_{\frac{K_b}{2}}, b_{\frac{K_b}{2}+1}, \ldots b_{K_b}$.

Note that it is important to identify the bits which are mapped to the in-phase and quadrature-phase component of the signal. For example, as in the previous 16-QAM example, bits b_1 and b_2 correspond to signal $s_I(t)$, while b_3 and b_4 map to $s_Q(t)$. Therefore, the above-mentioned principle of I/Q EQ is not directly applicable to M-Phase Shift Keying (M-PSK) modulation schemes, which have M values higher than 4, such as M = 8, 16, 32 since the bits of the multi-level symbol do not distinctly map to $s_I(t)$ and $s_Q(t)$ as in the above square-shaped M-QAM constellations. Another exception to the above I/Q EQ principles exists for non-square constellation M-QAM schemes, like 8-QAM and 32-QAM. For these signal constellations the number of bits, which map to $s_I(t)$ is not identical to that mapping to $s_Q(t)$. For example, in 8-QAM schemes, there are three bits, namely b_1, b_2 and b_3 in a symbol. Bits b_1 and b_2 can be used to represent $s_I(t)$, leaving bit b_3 to be mapped to $s_Q(t)$. Hence, the I/Q EQ of $s_I(t)$ will require more states, than the I/Q EQ equalising $s_Q(t)$. However, since non-square constellation M-QAM schemes, such as 8-QAM and 32-QAM, are not widely used due to their poor Euclidean distance, we will confine the following complexity analysis and the rest of our discussions to square-constellation M-QAM systems, such as 4-QAM, 16-QAM and 64-QAM.

6.3.5 Complexity of the In-phase/Quadrature-phase Equaliser

The complexity of the conventional trellis-based equaliser and the I/Q EQ for square-constellation M-QAM systems is expressed here as a function of the number of states and transitions for each trellis interval. Explicitly, the number of transitions considered at a particular trellis interval is our measure of complexity. For the conventional trellis-based equaliser, the complexity associated with equalising M-QAM signals transmitted over a complex channel having a delay spread of τ_d symbols is:

$$\begin{aligned}\text{Complexity} &= \text{Number of states} \cdot \text{Number of transitions}\\ &= \mathrm{M}^{\tau_d} \cdot \mathrm{M}\\ &= \mathrm{M}^{\tau_d+1} \text{ transitions per trellis interval,}\end{aligned} \tag{6.21}$$

whereas for the single I/Q trellis:

$$\begin{aligned}\text{Complexity} &= \text{Number of states} \cdot \text{Number of transitions}\\ &= \sqrt{\mathrm{M}}^{\,\tau_d} \cdot \sqrt{\mathrm{M}}\\ &= \sqrt{\mathrm{M}}^{\,\tau_d+1} \text{ transitions per trellis interval.}\end{aligned} \tag{6.22}$$

Hence, by using Equations 6.21 and 6.22 we can evaluate the reduction in complexity achieved by the I/Q EQ over the conventional trellis-based equaliser. Since two I/Q EQs are required to perform the equalisation, the complexity reduction factor achieved by the I/Q EQ can be expressed as:

$$\begin{aligned} \text{Complexity reduction factor} &= \frac{\mathrm{M}^{\tau_d+1}}{2 \times \sqrt{\mathrm{M}}^{\,\tau_d+1}} \\ &= 0.5 \times \sqrt{\mathrm{M}}^{\,\tau_d+1}. \end{aligned} \tag{6.23}$$

However, in order to evaluate the complexity of the entire turbo equaliser, which performs iterative joint equalisation and decoding, in addition to the equaliser complexity we also have to consider the number of iterations required. From the results obtained, we will show that even though the turbo equaliser using the reduced complexity I/Q EQ required a higher number of turbo equalisation iterations, its complexity was still lower than that of the turbo equaliser using the conventional trellis-based equaliser.

In summary, the principle of constructing the I/Q trellis is the same as that of a trellis-based equaliser equalising real signals, which have been transmitted over a complex channel. When implementing the I/Q equaliser, we treat $s_I(t)$ and $s_Q(t)$ as real-component signals. Once the trellis states and transitions are determined, trellis-based SISO algorithms can be invoked, in order to yield the *a posteriori* LLR of the encoded bits b_k. The novelty of the I/Q EQ is in the removal of the cross-coupling between $s_I(t)$ and $s_Q(t)$ from the received quadrature signals $r_I(t)$ and $r_Q(t)$. After the decoupling operation of Figure 6.5, the resultant decoupled signals $r'_I(t)$ and $r'_Q(t)$ can be equalised by utilising the low complexity I/Q EQs. For M-QAM signals transmitted over a complex channel having a delay spread of τ_d symbols, the complexity of the I/Q EQ is reduced by a factor of $0.5 \times \sqrt{\mathrm{M}}^{\,\tau_d+1}$, when compared to the conventional equaliser scheme. However, for BPSK transmission there is no reduction in complexity, since BPSK signalling employs the in-phase component only, resulting in no cross-coupling. It was also stated that the I/Q EQ required a specific mapping between bits b_k of a symbol and the quadrature signals $s_I(t)$ and $s_Q(t)$, which was shown in Figure 6.5. Since this mapping is not straightforward in M-PSK modulation, the I/Q EQ principle cannot be readily implemented in this context. Furthermore, it was noted that non-square M-QAM constellations result in unidentical I/Q EQ complexity. However, such systems are not commonly used, since they exhibit a poor Euclidean distance amongst constellation points and were therefore not considered in our further discussions.

Having described the salient operations of the I/Q EQ, we now give an overview of the system parameters utilised and subsequently present performance results for our turbo equalisation scheme employing the I/Q EQ.

6.4 System Parameters

In our forthcoming deliberations, the performance of our turbo equaliser employing the proposed reduced-complexity I/Q EQ and a convolutional decoder is investigated in the context of square-constellation M-QAM systems having a fixed system delay. We will elaborate on our considerations related to the system delay after describing the transmission burst structure.

The rate $R = \frac{1}{2}$, constraint length $K = 5$, recursive systematic convolutional code, with octal generator polynomials of $G_0 = 35$ and $G_1 = 23$, was invoked in the turbo-equalised M-QAM systems considered. In the first iteration, the DFE of Figure 6.5 was employed, as mentioned in Subsection 6.3.1. The number of forward and backward taps in the DFE was fifteen and four, respectively. In the subsequent iterations, two SISO I/Q EQs were employed, which utilised the Log-MAP algorithm of Section 2.5. The convolutional decoder used in these turbo-equalised M-QAM systems also employed the Log-MAP algorithm of Section 2.5 — rather than the less complex and more conventional Viterbi MLSE decoder — in order to supply the turbo equaliser with bit confidence values.

The transmission burst structure used in this system was the so-called FMA1 non-spread speech burst as specified in the Pan-European FRAMES proposal [92] and shown in Figure 5.1. In order to decide on the tolerance delay and the depth of the channel interleaver, we considered the maximum affordable delay of a speech system. This system delay is mainly determined by the latency introduced by the channel interleavers, where an entire block of bits must be received in the interleaver's buffer, before their transmission can commence. Here the processing delay attributed to the channel encoding, modulation and turbo equalisation operations has been ignored although practical systems typically have a processing delay, which allows them to complete their operations 'just' before they have to commence processing the incoming information block. Typically, speech systems can tolerate system delays, which are less than 40ms. Here, the acceptable delay is conservatively set to $\approx$ 30ms. For example, for a Time Division Multiple Access/Time Division Duplex (TDMA/TDD) system, which employs eight uplink and eight downlink slots, one transmission slot will be available after every sixteen TDMA slots. Furthermore, since each 72μs burst of Figure 5.1

consists of 144 data symbols, the total number of symbols transmitted within $\approx$ 30ms is:

$$\begin{aligned}
\text{Number of symbols} &= \left\lfloor \frac{\text{Maximum system delay}}{\text{Burst duration} \times \text{Number of slots between transmission}} \right\rfloor \\
&\quad \times \text{Data symbols per burst} \\
&= \left\lfloor \frac{30\text{ms}}{72\mu\text{s} \times 16} \right\rfloor \times 144 \qquad (6.24) \\
&= 26 \times 144 \\
&= 3456,
\end{aligned}$$

corresponding to 3456 symbols, when employing BPSK transmission. The notation $\lfloor \cdot \rfloor$ represents the integer floor. Upon assuming an identical signalling rate, the corresponding number of transmitted encoded bits for 4-QAM, 16-QAM and 64-QAM is 6912, 13824 and 20736, respectively. Hence, random channel interleavers of these depths were invoked in our investigations.

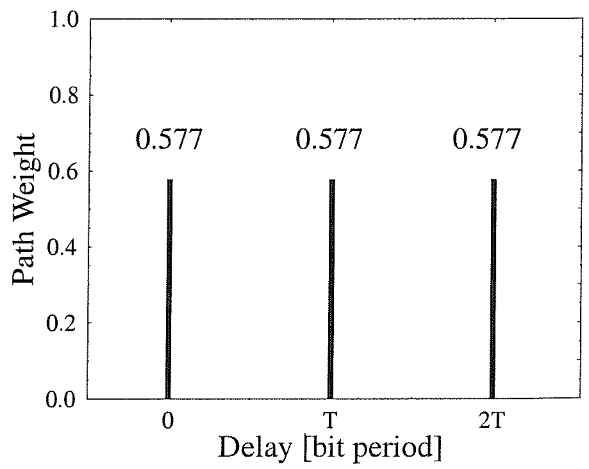


Figure 6.9: Illustration of the equally-weighted, symbol spaced three-path fading channel.

A three-path, symbol-spaced fading channel of equal weights was utilised, where the Rayleigh fading statistics obeyed a normalised Doppler frequency of 3.3×10^{-5}. This corresponded to a transmission frequency of 1900 MHz, signalling rate of 2600 KBaud and a vehicular speed of 30 mph. As before, in our investigations the CIR was assumed to be known and the fading magnitude and phase was kept constant for the duration of a transmission burst, a condition which we refer to as employing burst-invariant fading.

6.5 System Performance

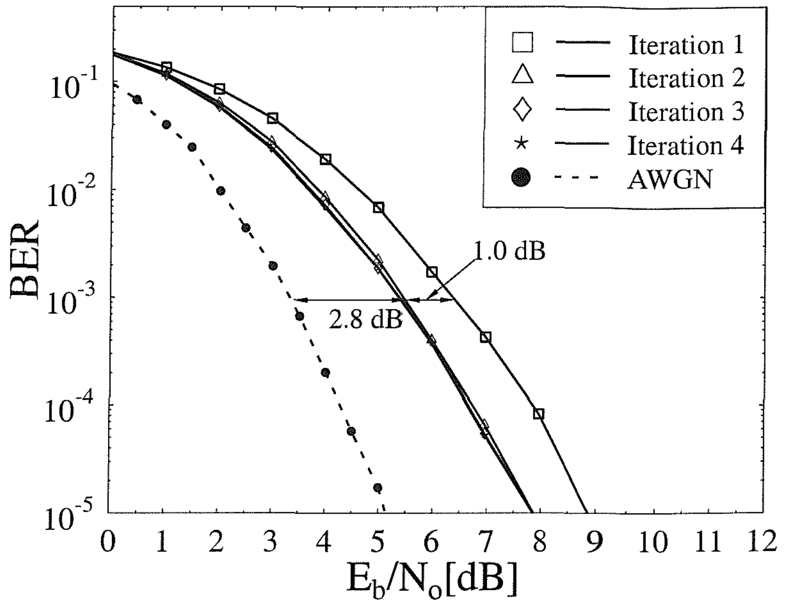
Figures 6.10(a) and 6.10(b) characterise the performance of our turbo equaliser using the reduced complexity I/Q EQs (**TEQ-IQ**) and that utilising the conventional trellis-based equaliser (**TEQ-EQ**) for a 4-QAM system, respectively, after four turbo equalisation iterations. The signals were transmitted over the equally-weighted three-path Rayleigh fading channel of Figure 6.9 using burst-invariant fading. For a transmission delay of $\approx$ 30ms, the depth of the channel interleaver implemented was 6912 bits. In Figure 6.10(a), it was observed that after two turbo equalisation iterations the performance of the TEQ-EQ did not improve significantly despite invoking further iterations. We used the term **critical number of iterations** I_t, in order to denote this iteration number, namely two in this case. When employing the TEQ-IQ receiver, the performance obtained after two and three turbo equalisation iterations was similar, as shown in Figure 6.10(b). Hence, the critical number of iterations performed by the TEQ-IQ receiver was three. The performance achieved by the TEQ-IQ receiver after three iterations was also observed to be similar to that obtained by the TEQ-EQ receiver after two iterations in Figure 6.10(b).

Let us study the complexity of the TEQ-IQ and TEQ-EQ receivers by using Equations 6.21 and 6.22, with the modification that the critical number of iterations required by TEQ-EQ and TEQ-IQ was also considered. The corresponding expressions for the receiver complexity can therefore be formulated as:

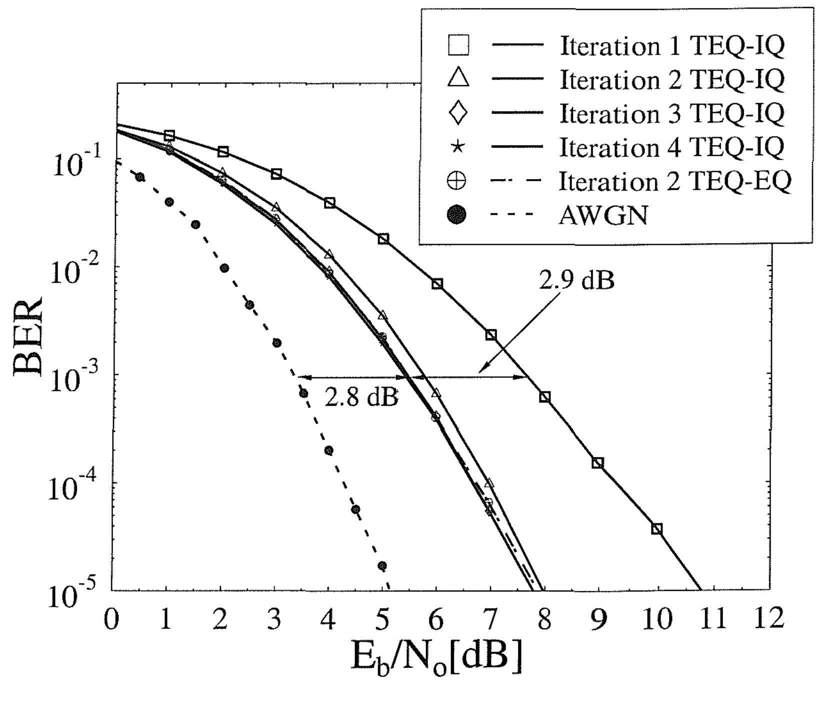
$$\text{TEQ-EQ complexity} = I_t(\text{TEQ-EQ}) \times \; \mathrm{M}^{\tau_d+1} \qquad (6.25)$$

$$\text{TEQ-IQ complexity} = I_t(\text{TEQ-IQ}) \times 2 \times \sqrt{\mathrm{M}}^{\;\tau_d+1}. \qquad (6.26)$$

Note that a factor '2' has been introduced in Equation 6.26, since two I/Q EQs are required in each iteration. Also observe that the channel decoder's complexity is not explicitly considered here, since the TEQ-IQ and TEQ-EQ receivers employ the same channel decoder. With the aim of simplifying the TEQ-IQ complexity expression, we have assumed the complexity of the DFE and I/Q EQ to be identical. In reality, the DFE's complexity is significantly lower than that of the I/Q EQ. In terms of arithmetic operations, the DFE's complexity is approximately proportional to N_f^3 [97], where N_f is the number of feed-forward filter taps. As mentioned previously, we have employed $N_f = 15$ feed-forward filter taps in our investigations, consequently setting the complexity to approximately 3375 arithmetic operations per equalised symbol. In contrast, the I/Q EQ has to evaluate the trellis transition metric as well as the forward and backward recursions of Equations 2.38, 2.30 and 2.34, respectively, for every transition, resulting in a higher number of operations. Hence, — although only an approximation — Equation 6.26 can be viewed as the maximum

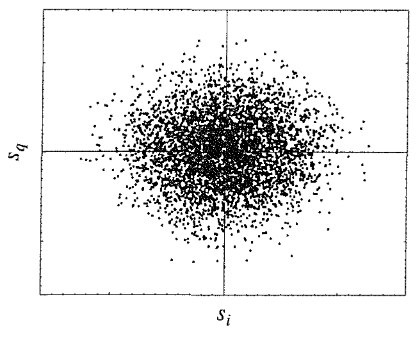


(a) Turbo equaliser using the conventional trellis-based equaliser

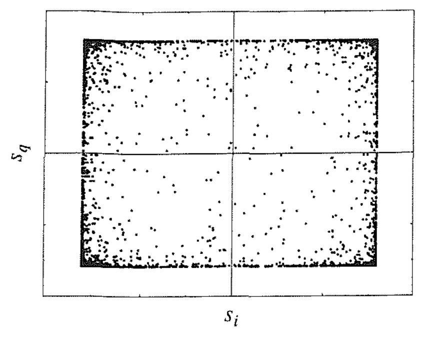


(b) Turbo equaliser using two I/Q EQs

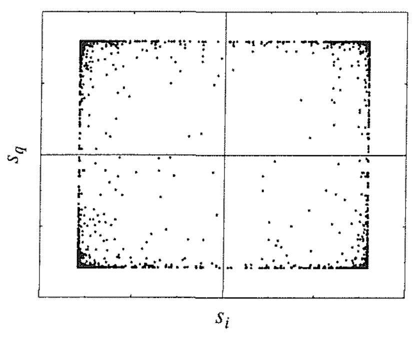
Figure 6.10: Performance of the turbo equaliser employing two I/Q EQs and that utilising the conventional trellis-based equaliser for a convolutional-coded 4-QAM system, possessing a channel interleaving depth of 6912 bits, transmitted over the equally-weighted, three-path Rayleigh fading channel of Figure 6.9 using a normalised Doppler frequency of 3.3×10^{-5} and burst-invariant fading.



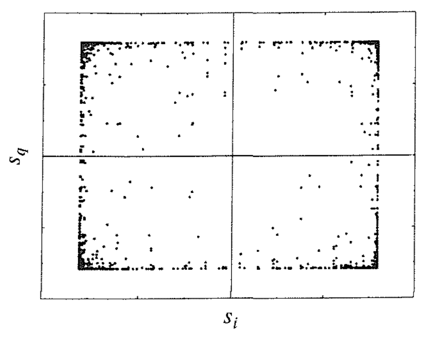
(a) Received signal constellation



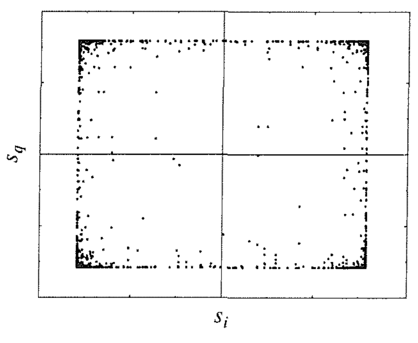
(b) Iteration 1



(c) Iteration 2



(d) Iteration 3



(e) Iteration 4

Figure 6.11: Phasor constellation of the received signal and that of the 4-QAM signal, which was generated from the decoder's LLR after one to four iterations at $E_b/N_o = 6$ dB over the three-path Rayleigh fading channel of Figure 6.9 using a normalised Doppler frequency of 3.3×10^{-5} and burst-invariant fading.

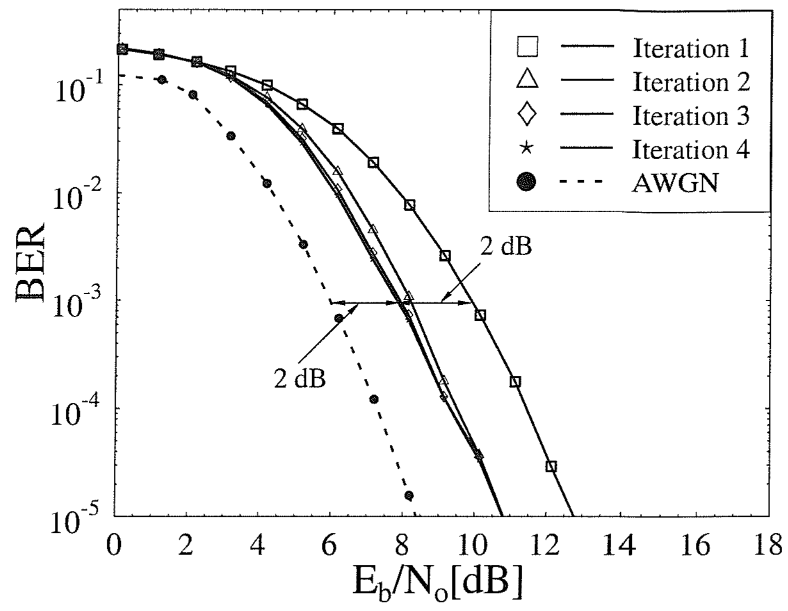
TEQ-IQ complexity. Substituting M = 4 and τ_d = 2 symbol periods in Equation 6.25, the TEQ-IQ and TEQ-EQ complexity becomes $3 \times 2 \times \sqrt{4}^{\,2+1} = 48$ transitions per trellis interval and $2 \times 4^{2+1} = 128$ transitions per trellis interval, respectively, since the critical number of iterations for the TEQ-IQ was three, whereas for TEQ-EQ it was two. Hence, a complexity reduction by a factor of $\frac{128}{48} = 2.67$ was achieved by the TEQ-IQ receiver, while obtaining the same performance as the TEQ-EQ receiver.

The ability of the TEQ-IQ receiver to mitigate the channel's ISI was studied as a function of the E_b/N_o loss evaluated for the turbo equalisation scheme after the critical number of iterations with respect to the decoding performance obtained over the AWGN channel, *i.e.* over the ISI-free channel at BER $= 10^{-3}$. In this respect, a loss of 2.8 dB was observed, as evidenced by Figure 6.10(b). In order to justify the associated E_b/N_o loss, the constellation of the received signal and that of the signal converted from the decoder's *a posteriori* LLR seen in Figure 6.5 was plotted for $E_b/N_o = 6$ dB after one to four iterations in Figure 6.11 over the three-path Rayleigh fading channel of Figure 6.9. After one iteration, the converted signals began to cluster around the 4-QAM constellation points. It was observed that additional iterations did not refine the constellation of the converted signals significantly, which still exhibited substantial residual ISI. Therefore, the turbo equaliser cannot completely overcome the severe frequency selectivity of the three-path fading channel, yielding an E_b/N_o degradation of approximately 2.8 dB, when compared to the decoding performance over the non-dispersive Gaussian channel *i.e.* to the ISI-free channel. It was also observed that the iteration gain after three turbo equalisation iterations — *i.e.* the gain in SNR performance terms with respect to the first iteration — for the 4-QAM system implementing TEQ-EQ was about 1 dB, whereas for the TEQ-IQ, it was 2.9 dB. We note that the lower iteration gain achieved by the TEQ-EQ is due to the fact that the performance of the TEQ-EQ — which employs the conventional trellis-based equaliser — is better than that of the DFE in the TEQ-IQ receiver after the first turbo equalisation iteration and identical after three iterations.

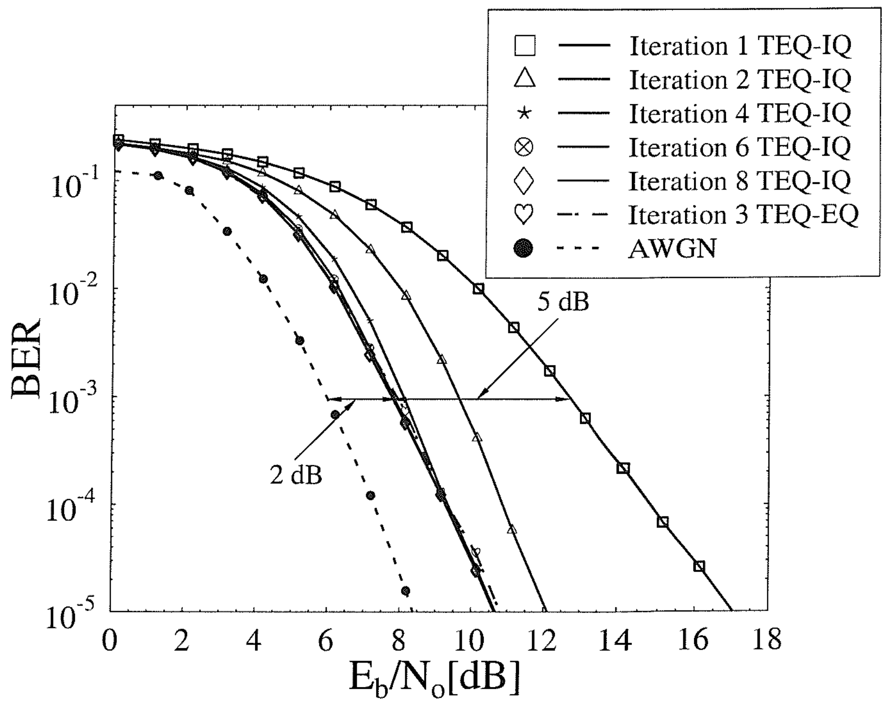
In summary, our findings showed that for 4-QAM transmission over an equally-weighted three-path Rayleigh fading channel, the TEQ-IQ can achieve the same performance as the TEQ-EQ after three turbo equalisation iterations. Using Equations 6.25 and 6.26, the complexity of the TEQ-IQ and TEQ-EQ receivers was found to be 48 and 128 transitions per trellis interval, respectively. The TEQ-IQ receiver achieved a complexity reduction factor of 2.67 compared to the TEQ-EQ receiver, whilst achieving an identical BER performance. It was also observed that the TEQ-IQ receiver was unable to further mitigate the channel ISI after the critical number of iterations, *i.e.* after three turbo equalisation iterations. This was because the signal mapped from the channel decoder's *a posteriori* information to the 4-QAM symbols still retained residual ISI at this stage, as shown in Figure 6.11. It was also observed that the TEQ-IQ receiver yielded an iteration gain of approximately 2.9 dB, while the TEQ-EQ receiver achieved a gain of about 1 dB after performing their respective critical number of iterations. The TEQ-EQ exhibited a lower iteration gain since the performance of the conventional trellis-based equaliser was better, than that of the DFE invoked in the TEQ-IQ receiver in the first iteration.

As a further set of results, Figure 6.12(a) displays the performance of the TEQ-EQ receiver for 16-QAM transmitted over the equally-weighted three-path Rayleigh fading channel of Figure 6.9 using burst-invariant fading and employing a channel interleaving depth of 13824 bits in order to maintain a total system delay of approximately 30 ms. The critical number of iterations was three when employing the 16-QAM TEQ-EQ receiver, since no further significant gain was achieved by performing additional turbo equalisation iterations. In contrast, the critical number of iterations was six when employing the 16-QAM TEQ-IQ receiver over the same channel, as shown in Figure 6.12(b). Comparing the performance obtained by the 16-QAM TEQ-IQ and the TEQ-EQ after their critical number of iterations, it was observed in Figure 6.12(b) that both receivers yielded a similar BER performance. The complexity of the 16-QAM TEQ-EQ and TEQ-IQ receivers was estimated as before, by using Equations 6.25 and 6.26, giving $3 \times 16^3 = 12288$ and $6 \times 2 \times \sqrt{16}^{\ 3} = 768$ transitions per trellis interval, respectively. Here, the 16-QAM TEQ-IQ receiver achieved a complexity reduction by a factor of 16, relative to the TEQ-EQ receiver, while still obtaining the same performance.

It was also observed in Figure 6.12(b) that the performance of the 16-QAM TEQ-IQ receiver after six iterations at BER $= 10^{-3}$ was 2 dB from the decoding performance obtained over the ISI-free AWGN channel. Recall that the E_b/N_o loss of the 4-QAM system was 2.8 dB, hence indicating an improved performance by the TEQ-IQ receiver for 16-QAM. The constellation of the received 16-QAM signal and the signal obtained by converting the decoder's *a posteriori* LLRs to symbols was plotted in Figure 6.13 for $E_b/N_o = 12$ dB after one to six iterations. After the first iteration a crude approximation of the 16-QAM signal constellation was beginning to emerge. Subsequent iterations improved the reliability of the channel decoder's *a posteriori* LLRs, hence yielding a more accurate mapping of the 16-QAM signal. Therefore, at the sixth turbo equalisation iteration in Figure 6.13(g) a close approximation of the transmitted 16-QAM signal constellation was achieved and no severe residual ISI persisted. Hence, an improved performance was obtained, compared to the 4-QAM system, which still exhibited residual ISI after four

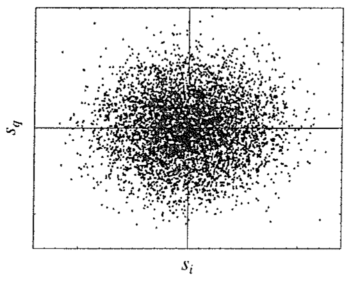


(a) Turbo equaliser using the conventional trellis-based equaliser

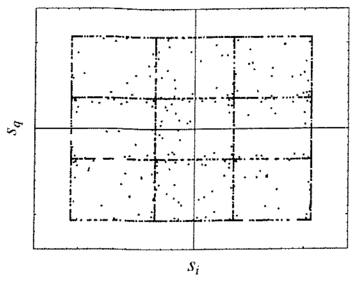


(b) Turbo equaliser using two I/Q EQs

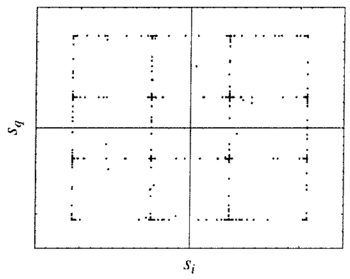
Figure 6.12: Performance of the turbo equaliser employing two I/Q EQs and that utilising the conventional trellis-based equaliser for a convolutional-coded 16-QAM system, possessing a channel interleaving depth of 13824 bits, transmitted over the equally-weighted three-path Rayleigh fading channel of Figure 6.9 using a normalised Doppler frequency of 3.3×10^{-5} and burst-invariant fading.



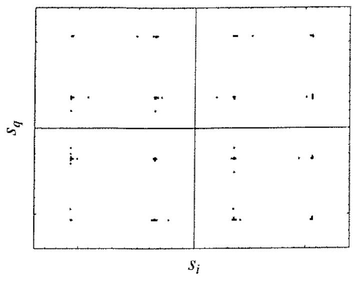
(a) Received signal constellation



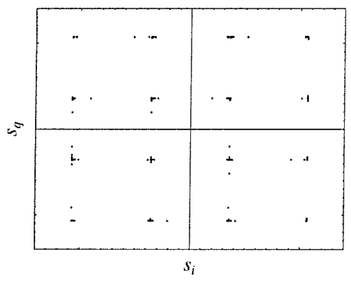
(b) Iteration 1



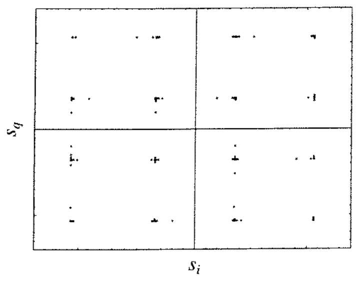
(c) Iteration 2



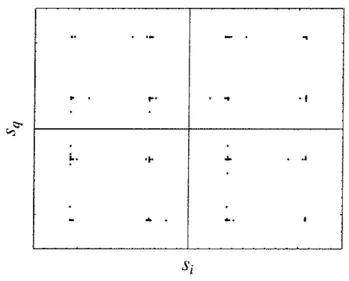
(d) Iteration 3



(e) Iteration 4



(f) Iteration 5



(g) Iteration 6

Figure 6.13: Phasor constellation of the received signals and that of the 16-QAM signals, which were converted from the channel decoder's LLRs to 16-QAM symbols after one to six iterations at $E_b/N_o = 12$ dB over the three-path Rayleigh fading channel of Figure 6.9 using a normalised Doppler frequency of 3.3×10^{-5} and burst-invariant fading.

turbo equalisation iterations. Also worth noting is the difference in the channel interleaver depths implemented, *i.e.* which was 6912 bits for 4-QAM and 13824 for 16-QAM. The larger interleaving depths reduced the correlation between the bits, hence yielding a better performance.

The iteration gain achieved by the 16-QAM TEQ-IQ receiver after performing the critical number of iterations, *i.e.* six turbo equalisation iterations, was approximately 5 dB. When compared to the 4-QAM system, which encountered the same dispersive Rayleigh fading channel, the iteration gain obtained by the 16-QAM system was 2.9 dB higher. The 16-QAM system experienced more severe ISI-induced impairments, since the Euclidean distance within the constellation was lower compared to the 4-QAM constellation. Hence, the 16-QAM performance after one iteration was much poorer than that of 4-QAM. However, when turbo equalisation was implemented, the effects of the ISI inflicted by the channel were mitigated and the receiver was capable of approximating the performance obtained over the non-dispersive Gaussian channel. This justified the higher iteration gain achieved by the 16-QAM system in comparison to the 4-QAM scheme.

Examining the performance of our 64-QAM system over the same dispersive Rayleigh fading channel in Figure 6.14, it was observed that the critical number of iterations was six, *i.e.* identical to that of the 16-QAM system. Simulations could not be conducted for the 64-QAM TEQ-EQ system, since the trellis-based equaliser required $64^2 = 4096$ states and 64 transitions per state. However, in order to compute the complexity of the TEQ-EQ receiver, we assume — based on the 16-QAM performance — that its critical number of iteration was three, giving a complexity of $3 \times 64^3 = 786432$ number of transitions per trellis interval. By contrast, the TEQ-IQ receiver incurred a complexity of $6 \times 2 \times \sqrt{64}^{\,3} = 6144$ number of transitions per trellis interval, yielding a significant complexity reduction factor of $\frac{786432}{6144} = 128$. After six turbo equalisation iterations the performance of the TEQ-IQ receiver at BER $= 10^{-3}$ was only 1.5 dB from the decoding performance curve over the non-dispersive Gaussian channel, as shown in Figure 6.14. This was an improvement, when compared to the E_b/N_o loss of 2.8 dB and 2 dB suffered by the 4-QAM and 16-QAM systems, respectively and can be attributed to the higher interleaving depths employed. As before, the constellation of the received 64-QAM signal and the signal obtained by converting the channel decoder's *a posteriori* LLRs to 64-QAM symbols was plotted in Figure 6.15 for $E_b/N_o = 18$ dB after one to six iterations. After the first iteration the 64-QAM signal constellation was beginning to emerge. Subsequent iterations improved the reliability of the channel decoder's *a posteriori* LLR values, hence yielding a more accurate mapping of the 64-QAM symbols. Therefore, by the sixth turbo equalisation

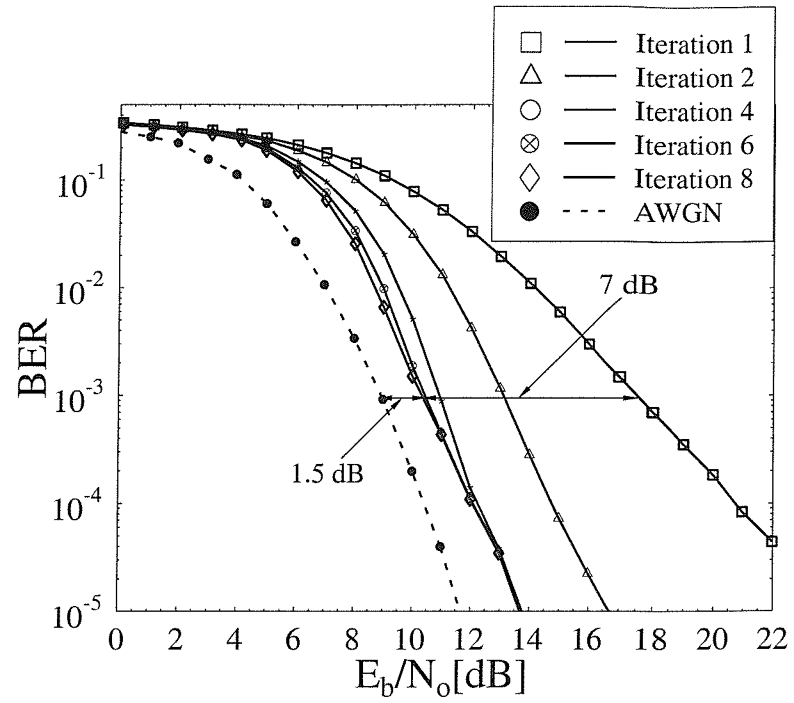


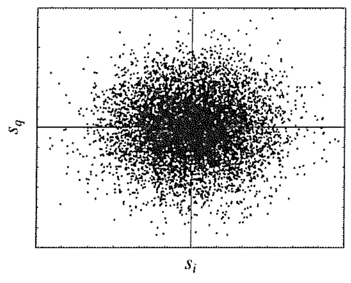
Figure 6.14: Performance of the turbo equaliser employing two I/Q EQs for a convolutional-coded 64-QAM system, possessing a channel interleaving depth of 20736 bits, transmitted over the equally-weighted three-path Rayleigh fading channel of Figure 6.9 using a normalised Doppler frequency of 3.3×10^{-5} and burst-invariant fading.

iteration in Figure 6.15(g), a close approximation of the 64-QAM signal constellation was achieved and no severe residual ISI persisted, hence justifying the lower E_b/N_o loss obtained.

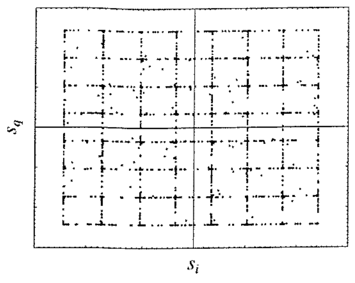
In Figure 6.14, the iteration gain observed was 7 dB. The 64-QAM constellation had a lower Euclidean distance between each point hence increasing the likelihood of making an error, when the channel ISI and noise were dominant. Therefore, the performance of the TEQ-IQ after the first iteration was poor. However, by implementing joint equalisation and decoding — namely turbo equalisation — the channel ISI was mitigated, hence ultimately approximating the ISI-free performance. This explains the larger iteration gain obtained in conjunction with higher-order modulation modes.

6.6 Summary

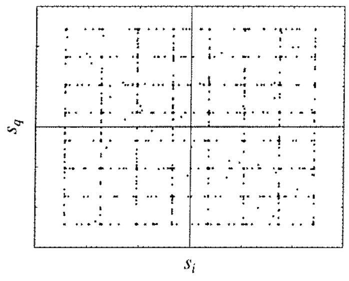
In this chapter, a novel reduced complexity trellis-based equaliser referred to as the I/Q EQ was proposed, in order to equalise M-QAM signals and to provide soft decisions, which were



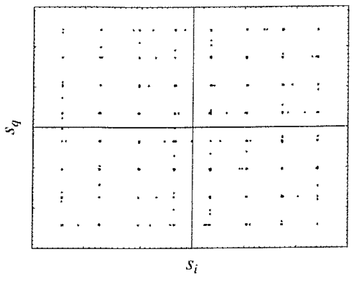
(a) Received signal constellation



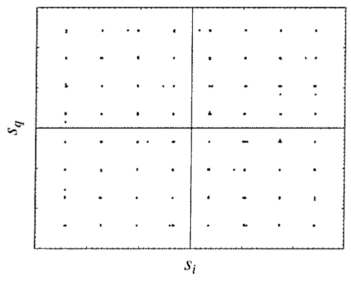
(b) Iteration 1



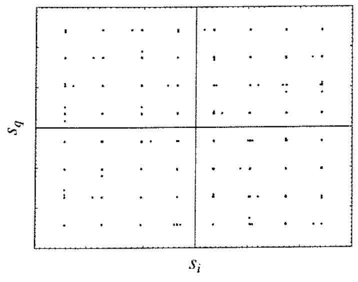
(c) Iteration 2



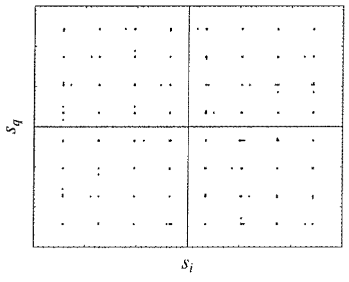
(d) Iteration 3



(e) Iteration 4



(f) Iteration 5



(g) Iteration 6

Figure 6.15: Phasor constellation of the received 64-QAM signals and that of the 64-QAM signals, which were converted from the channel decoder's LLRs to 6-bit symbols after one to six iterations at $E_b/N_o = 18$ dB over the three-path Rayleigh fading channel of Figure 6.9 using a normalised Doppler frequency of 3.3×10^{-5} and burst-invariant fading.

employed in the context of turbo equalisation. When the M-QAM signal $s(t)$, consisting of quadrature signals $s_I(t)$ and $s_Q(t)$, was transmitted over a complex channel $h(t)$, the complex channel output contained contributions from both $s_I(t)$ and $s_Q(t)$ in each receiver quadrature arm, as expressed in Equation 6.1. Therefore, when a channel with memory τ_d was encountered, the conventional trellis-based equaliser must consider M^{τ_d+1} M-QAM signal sequence combinations at each trellis stage in order to equalise the received signal. Hence, the complexity of the conventional trellis-based equaliser increased exponentially with τ_d. However, by removing the associated cross-coupling of $s_I(t)$ and $s_Q(t)$ and rendering the channel output to be only dependent on either $s_I(t)$ or $s_Q(t)$, the number of signal-sequence combinations considered was reduced to $\sqrt{M}^{\tau_d+1}$. This was the motivation for design of the I/Q EQ. The decoupling operation was performed by using the CIR estimates and the channel decoder's *a posteriori* LLRs from the previous iteration, which have been converted into M-QAM symbols. Subsequently, the cross-coupling of the complex received signals was cancelled and the signals were combined accordingly to give the resultant output signals $r'_I(t)$ and $r'_Q(t)$ of Equation 6.3 and Figure 6.4. For perfect cancellation, the signals $r'_I(t)$ and $r'_Q(t)$ were no longer affected by the cross-coupling of $s_I(t)$ and $s_Q(t)$. The signals $r'_I(t)$ and $r'_Q(t)$ were then passed to the I/Q EQs, which employed SISO algorithms — such as the Log-MAP algorithm of Section 2.5 — in order to provide soft decisions for turbo equalisation. The performance of the turbo equaliser using I/Q EQs — namely that of the TEQ-IQ scheme — was compared with the conventional turbo equaliser TEQ-EQ for M-QAM transmissions over the equally-weighted three-path Rayleigh fading channel of Figure 6.9 using a normalised Doppler frequency of 3.3×10^{-5} and burst-invariant fading. It was observed in Figures 6.10(b) and 6.12(b) that the TEQ-IQ receiver was capable of achieving the same performance as the more complex TEQ-EQ receiver for 4-QAM and 16-QAM, while achieving a corresponding complexity reduction factor of 2.67 and 16, respectively. For 64-QAM, the performance of the TEQ-IQ receiver at BER $= 10^{-3}$ is only 1.5 dB from the decoding performance curve over the non-dispersive Gaussian channel, as it was shown in Figure 6.14, while reducing its complexity by a factor of 128 times, relative to the TEQ-EQ receiver. Note that although we were unable to simulate the performance of the TEQ-EQ receiver for 64-QAM transmissions over the same dispersive Rayleigh fading channel due to complexity reasons, we have computed its complexity based on the assumption that its critical number of iterations was three, as in the observed performance of the TEQ-EQ receiver for the 16-QAM system. It was also concluded from the TEQ-IQ performance that despite the poor symbol mapping from low-reliability decoder LLR values in the initial turbo equalisation iterations, the employment of subsequent turbo equalisation iterations improved the symbol mapping, hence yielding a better BER performance. The iteration gain achieved after the critical number of iterations increased for the higher-order modulation modes, exhibiting gains of 2 dB, 5 dB and 7 dB for 4-QAM, 16-QAM and 64-QAM, respectively. This was expected, since the Euclidean distance between signal points in the constellation was reduced for higher-order modulation modes, hence increasing the likelihood of making an error when the channel ISI and noise were dominant. Therefore, the performance of the TEQ-IQ after the first iteration was poor. Furthermore, longer interleaving depths could be implemented for higher-order modulation modes, while maintaining the target system delay of $\approx$ 30ms. The higher-interleaving depths reduced the correlation between the bits hence improving the turbo equaliser's performance in mitigating the effects of ISI. Consequently, near ISI-free performance was achieved, as evidenced by Figures 6.10(b), 6.12(b) and 6.14.

Chapter 7

Summary and Conclusions

In this chapter, the main findings of our investigations are summarised and suggestions for further research are presented.

Chapter 2 presented a study of turbo coding in the context of systems sensitive to delays, such as speech systems. Turbo codes are known to perform well, when large turbo interleavers can be implemented, *i.e.* in systems not constrained by low delay requirements. However, for speech systems the delay of the system must be minimised and hence the turbo interleaver depth that can be implemented is also low. The objective of the research in Chapter 2 was to study the feasibility of employing turbo codes possessing short turbo interleavers in the context of low-rate speech systems, namely the 1.9 kbps speech system [1]. The results were compared with the performance of convolutional-coded systems using the same speech codec, modulation and channel conditions. Specifically, the speech codec operated on 20 ms frames and output 38 bits for each of these 20 ms frames. Subsequently, the encoded speech bits from the speech encoder were channel encoded using a $\frac{1}{2}$-rate turbo encoder in conjunction with a 9x9 turbo interleaver, and the GSM convolutional encoder, in order to compare the associated BER performance. Assuming negligible processing delay, 162 bits were released every 40ms, entailing two 20ms speech frames, since the 9x9 turbo interleaver matrix employed required two 20ms, 38-bit speech frames before channel interleaving could commence. It was observed in Figures 2.10 and 2.11, that turbo coding obtained a modest improvement of approximately 0.25 dB in comparison to the standard convolutional coding for transmissions over the narrowband Rayleigh fading channel and COST207 typical urban channel of Figure 2.8. This marginal improvement was attributed to the short interleaver depth used in the turbo encoder. Therefore, it was concluded that it was not justified to invest in the higher complexity and longer delay turbo codec, due to the limitation of short-latency turbo-coded systems. For longer coding frames, simulations were carried out for the full-rate GSM speech codec and for the GSM data channel. The full-rate GSM speech codec utilised a coding frame length of 378 bits and a 11x17-bit turbo interleaver. In the GSM data channel, a turbo encoder with a random turbo interleaver of depth 1083 bits was used. We employed a random turbo interleaver, since it was shown in reference [61] that for long turbo coding frames the random interleaver yielded superior performance in comparison to block interleavers. In Figures 2.13(a) and 2.13(b), we observed that the performance of the 11x17-bit turbo-interleaved turbo code was 0.5 dB better, than that of the $K = 5$ convolutional code for the narrowband Rayleigh fading channel and approximately 0.7 dB better over the COST207 typical urban channel at a BER of 10^{-3}. For the GSM data channel, the coded BER of turbo codes was approximately 1.5 dB better, than that of the standard convolutional code at BER=10^{-4} for both of these channels. By increasing the coding frame length further to the order of 50000 bits, we obtained a gain of approximately 3.7 dB as compared to the standard convolutional code over the so-called perfectly interleaved narrowband Rayleigh fading channel, as illustrated in Figure 2.15. This confirmed the superiority of turbo codes over convolutional codes when the coding frame length was sufficiently long.

In Chapter 3 the advantage gained by performing the equalisation and decoding jointly was examined in the context of GMSK-modulated systems, as compared to implementing the equalisation independently from the decoding, as in Chapter 2. This joint equalisation and decoding technique is known as turbo equalisation [10] and is performed iteratively. We then quantified the advantage gained by turbo equalisation over independent equalisation and decoding. Gains of 3.7 dB and 4.2 dB were achieved for the rate $R = \frac{1}{2}$ convolutional-coded GMSK system transmitting over the non-dispersive Gaussian channel and the five-path Rayleigh fading channel, respectively. Similarly, for the rate $R = \frac{1}{2}$ convolutional-coding based turbo-coded scheme, the corresponding gains achieved through turbo equalisation over independent equalisation and turbo decoding were 0.5 dB and 0.8 dB, over the non-dispersive Gaussian channel and the dispersive Rayleigh fading channel considered. Finally, the BCH-coding based turbo-coded systems employing turbo equalisation and transmitting over the non-dispersive Gaussian and the five-path fading channel also demonstrated gains of 2.8 dB and 3 dB, respectively. Secondly, it was observed that the rate $R = 0.5$ convolutional-coded GMSK system employing turbo equalisation obtained a better performance, than that of the $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme. For transmission over the non-dispersive Gaussian channel the convolutional-coded GMSK system outperformed the convolutional-coding based turbo-coded GMSK scheme by 0.8 dB at BER $= 10^{-4}$, while in the five-path Rayleigh fading scenario the convolutional-coded system attained a gain of approximately 1.0 dB over the convolutional-coding based turbo-coded scheme, as it was illustrated in Figures 3.19(a) and 3.19(b), respectively. These results were surprising, since the turbo-coded system — which consists of two convolutional encoders — was expected to perform better, as it was a more powerful coding compared to convolutional codes. However, from Figures 3.19(a) and 3.19(b), which compared the turbo equalisation performance of the rate $R = 0.5$, $K = 5$ conventional convolutional-coded GMSK system and $R = 0.5$ convolutional-coding based turbo-coded GMSK scheme over both the non-dispersive Gaussian channel and five-path Rayleigh fading channel, it was observed that at E_b/N_o values less than 4.5 dB and 6 dB, respectively, the BER of the convolutional-coded system was lower, than that of the turbo-coded scheme after one turbo equalisation iteration. This indicated that the decoder in the $R = 0.5$, $K = 5$ convolutional-coded GMSK system was providing more reliable LLR values for the equaliser. Consequently, the equaliser in the convolutional-coded scheme — upon receiving the higher confidence LLR values from the decoder — was capable of producing more reliable LLR values, which were subsequently passed to the decoder in the following turbo equalisation iteration. Beyond the E_b/N_o values of 4.5 dB and 6 dB, the turbo-coded scheme exhibited a better performance after one turbo equalisation iteration. Therefore, the turbo-coded system was expected to obtain a lower BER. However, since speech and data systems have a target BER of 10^{-3} and 10^{-4}, respectively, convolutional codes constitute a better choice for coded-GMSK systems employing turbo equalisation. A similar phenomenon was also observed in Figure 3.20 in conjunction with the weaker BCH-coding based turbo codes, which also yielded a better performance than the convolutional-coding based turbo codes, when turbo equalisation was performed. Here, the same trend was observed as before, where the performance of the BCH-coding based turbo-coded system was better than that of the convolutional-coding based turbo-coded scheme after one turbo equalisation iteration. This occurred at E_b/N_o values less than 4.0 dB and 5.5 dB for transmissions over the non-dispersive Gaussian channel and over the five-path Rayleigh fading channel, respectively. Therefore, the equaliser in the BCH-coding based turbo-coded system received more reliable LLR information compared to the convolutional-coding based turbo-coded scheme, hence enabling it to achieve a better BER performance after implementing successive turbo equalisation iterations.

Chapter 4 described the fundamental principles required for deriving the ML union bound performance for non-punctured convolutional-coded and convolutional-coding based turbo-coded DPSK systems. The aim was to characterise the relative performance bounds of coded GMSK systems and to justify our observation in Chapter 3 that the convolutional-coded GMSK scheme achieved better turbo equalisation performance, than the convolutional-coding based turbo-coded GMSK system. The ML performance analysis was based on modelling the modulator as an inner encoder and viewing the entire system as a serial concatenated coded scheme. Subsequently, the union bound principles of SCCCs [19] was employed for the convolutional-coded system, whereas the analysis for PCCCs [20] was used in conjunction with the above SCCC principles for turbo-coded schemes. However, since GMSK modulation could not be readily modelled as a binary inner encoder, it was substituted by DPSK, which is also inherently recursive. Our simulation results in Figure 4.11(a) demonstrated that the rate $R = \frac{1}{3}$ convolutional-coded scheme outperformed the rate $R = \frac{1}{3}$ turbo-coded system by a margin of 0.5 dB at BER $= 10^{-4}$. The trends of our simulation results were also corroborated by the theoretical performance bound in Figure 4.11(b). As the E_b/N_o value increased beyond 3 dB, the theoretical bound for the convolutional-coded system began to flatten at approximately BER $= 10^{-8}$ to BER $= 10^{-12}$, whereas the error-floor for the turbo-coded system was approximately at BER $= 10^{-10}$ to BER $= 10^{-14}$. It was concluded that the turbo-coded DPSK system exhibited a lower error-floor, than the convolutional-coded system, but at low E_b/N_o values, the turbo-coded system yielded poorer performance than the convolutional-coded DPSK system. These observations were also consistent with the results obtained for convolutional-coded GMSK schemes and turbo-coded GMSK systems in Chapter 3. Therefore, for speech and data systems — requiring BER $= 10^{-3}$ and BER $= 10^{-4}$, respectively — employing recursive modulation techniques, such as DPSK or GMSK and utilising turbo equalisation, convolutional coding is the more robust choice, compared to the more complex turbo coding schemes.

Having studied the turbo equalisation performance of various coded systems employing recursive modulators, Chapter 5 investigated the different receiver configurations of BPSK modulated systems using BCH-coding based turbo codes, convolutional-coding based turbo codes, and convolutional codes, denoted as **BT**, **CT** and **CC**, respectively. Non-iterative and iterative equaliser/decoders operating at code rates of $R \approx \frac{1}{2}, R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ were studied. In the iterative schemes, loops containing decoders only, as in turbo decoding, and loops containing joint equalisation and decoding stages — *i.e.* employing turbo equalisation — were investigated. The comparative study of the turbo equalisers for the above **BT**, **CT** and **CC** systems demonstrated that at high code rates of $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ the **BT** system was marginally better or comparable to the **CT** system at BER $= 10^{-4}$, at the expense of higher complexity compared to the **CT** system. At these high code rates and over the equally-weighted symbol-spaced five-path Rayleigh fading channel of Figure 5.2(b), the **CC** system performed poorly since its turbo equalisation iteration gain was low. This was because after the first iteration the system's performance was already close to the decoding results over the non-dispersive Gaussian channel. At code rate $R = \frac{5}{6}$ we observed a loss of $E_b/N_o = 1.0$ dB over the five-path Gaussian channel and a loss of $E_b/N_o = 3.8$ dB, over the five-path Rayleigh fading channel at BER $= 10^{-4}$ after eight turbo equalisation iteration when compared to the **BT** system. On the whole, the turbo-equalised **CT** system was the most robust scheme, giving comparable performance — within a few tenths of a dB — for all code rate investigated, when compared to the best system in each scenario. Furthermore, the turbo-equalised **CT** system had a lower receiver complexity, when compared to the **BT** system, hence making it the best choice in most applications.

Finally, in Chapter 6 a novel reduced complexity trellis-based equaliser referred to as the I/Q EQ, was proposed for the turbo equalisation of M-QAM signals. When the M-QAM signals consisting of quadrature components were transmitted over a complex channel, the complex channel output contained contributions from each quadrature arm, as expressed in Equation 6.1. Therefore, when a channel with memory τ_d was encountered, the trellis-based equaliser must consider M^{τ_d+1} M-QAM signal sequence combinations, in order to equalise the received signal. Hence, the complexity of the complex-valued trellis-based equaliser increases rapidly with τ_d. However, by removing the associated cross-coupling of the in-phase and quadrature-phase signal components and hence rendering the channel output to be only dependent on either quadrature component, the number of signal-sequence combinations considered was reduced to $\sqrt{M}^{\,\tau_d+1}$. This was our motivation for the design of the I/Q EQ. The performance of the turbo equaliser using two I/Q EQs — denoted as TEQ-IQ — was compared with that of complex-valued turbo equaliser TEQ-EQ for M-QAM transmissions over the equally-weighted three-path Rayleigh fading channel using a normalised Doppler frequency of 3.3×10^{-5}, where fading was assumed to be invariant throughout the transmission burst. It was observed in Figures 6.10(b) and 6.12(b) that the TEQ-IQ receiver was capable of achieving the same performance as the TEQ-EQ receiver for 4-QAM and 16-QAM, while maintaining a complexity reduction factor of 2.67 and 16, respectively. For 64-QAM, the performance of the TEQ-IQ receiver at BER $= 10^{-3}$ was only 1.5 dB from the decoding performance curve over the non-dispersive Gaussian channel, as shown in Figure 6.14, while reducing its complexity by a factor of 128, relative to the TEQ-EQ receiver. Note that although we were unable to simulate the performance of the TEQ-EQ receiver for 64-QAM transmissions over the same dispersive Rayleigh fading channel, we have computed its complexity based on the assumption that its critical number of iterations was three, as for the 16-QAM TEQ-EQ system. It was also noticed from the TEQ-IQ performance that despite the poor symbol mapping from low-reliability channel decoder LLR values to M-QAM symbols in the initial stages, the employment of subsequent turbo equalisation iterations improved the symbol mapping, hence yielding a better BER performance. The iteration gain achieved after the critical number of iterations increased with M, achieving 2 dB, 5 dB and 7 dB for 4-QAM, 16-QAM and 64-QAM, respectively. This was expected, since the Euclidean distance between signal points in the constellation was reduced for higher-order modulation modes, hence increasing the likelihood of making an error when the channel ISI and noise were dominant. Therefore, the performance of the TEQ-IQ after the first iteration was poor. Furthermore, longer interleaving depths could be employed for higher modulation modes, while maintaining the target channel transmission delay of $\approx$ 30ms. The higher interleaver depths reduced the correlation between the bits, hence improving the turbo equaliser's performance in mitigating the effects of ISI and approximating the ISI-free performance. Overall, it has been demonstrated that the turbo equaliser employing the reduced complexity I/Q EQs — namely, the TEQ-IQ receiver — is capable of achieving the same performance, as the complex-valued turbo equaliser, while incurring lower computational complexity. Furthermore, the TEQ-IQ can also perform equalisation and decoding for high-order modulation modes such as 64-QAM, which is not feasible by using the complex-valued trellis-based equaliser due to its complexity constraints.

7.1 Suggestions for Further Research

Apart from mitigating the effects of ISI induced by the channel and modulator, turbo equalisation techniques are also powerful in terms of improving channel estimation techniques [27]. Recent work in this area include the utilisation of the so-called Per Survivor Processing (PSP) technique [98, 99, 100, 101] in the SISO algorithm of the equaliser [102] for the purpose of joint channel estimation and data equalisation. Reference [103] also performed channel estimation by modifying the Maximum *A Posteriori* algorithm and employing a channel estimator, in order to evaluate the channel information recursively.

The ML bound derived for the coded DPSK systems, based on the union bound technique, produces a steep divergence from the true performance at low E_b/N_o values. Recently, several new bounds have been proposed in order for turbo codes to estimate the BER performance more accurately in this divergence region [104, 105, 106]. This can be adapted to improve the accuracy of the ML bound evaluated for the coded DPSK systems. The ML bound for coded GMSK systems constitutes a further topic for research. This involves augmenting the trellis of the encoder and modulator into a hypertrellis [107]. By using this hypertrellis, the weight spectrum of the concatenated system can be evaluated and subsequently utilised, in order to approximate the BER performance.

Iterative multi-user interference detection employing the turbo equalisation strategy is also an attractive research topic. Recent work utilising turbo equalisation has been successful in mitigating the interference inflicted by other users, such that the near single-user performance is achieved [108, 109, 110]. Low complexity iterative interference cancellation solutions were also proposed by Wang and Poor [111]. Their simulation results demonstrated that the proposed low-complexity iterative receiver structure invoked for interference suppression and decoding offers significant performance gains over the traditional non-iterative receiver structure. Moreover, at high signal-to-noise ratios the detrimental effects of Multiple Access Interference (MAI) and ISI in the channel can almost be completely overcome by iterative processing and the single-user performance can be approached.

Equaliser designs in the context of joint cancellation of Cochannel Interference (CCI), Adjacent Channel Interference (ACI) and Inter Symbol Interference (ISI) [112] are also promising in terms of their added performance potential. In mobile radio systems frequency reuse is employed, in order to provide contiguous radio coverage over a geographical area. As a consequence, the receivers suffer from CCI. By implementing demodulators which compensate for CCI, the traffic capacity of the system can be increased. In a recent contribution by Berangi *et al* [113], a blind cochannel interference canceller for constant envelope modulation systems was presented. In this technique, the channels of the cochannel interferers do not have to be identified. By exploiting the constant envelope property of the modulation schemes, an appropriate metric was devised and CCI equalisation was then performed with the aid of the Viterbi algorithm coupled with the devised metric. Another approach suggested by Wales [114] removes CCI through the joint detection of the desired and interfering signals. Here, the maximum likelihood receiver compensates for the CCI by subtracting the estimate of the interfering signals from the received signal prior to comparison with the most likely transmitted symbol sequences. This technique requires the knowledge of the desired signal's channel and the interfering signals' channels. Hence it was computationally expensive. A sub-optimum receiver with reduced complexity was then presented in the same paper [114], using a superstate trellis structure.

Turbo equalisation for Trellis-Coded Modulation (TCM) [115, 116] is a further topic investigated in the recent literature in order to improve the spectral efficiency of the system — which is one of the key objectives of mobile radio research.

Glossary

3G	Third generation mobile communications systems
ACI	Adjacent Channel Interference
ACTS	Advanced Communication Technologies and Services
AMPS	Advanced Mobile Phone Service
BCH	Bose-Chaudhuri-Hocquenghem. A class of forward error correcting codes (FEC)
CCI	Cochannel Interference
CISI	Controlled Inter Symbol Interference
CSV	Correlative State Vector. It is a sequence of previously transmitted bits, which is dependent on the channel delay spread and the spreading in the modulator.
DFE	Decision Feedback Equaliser
DPSK	Differential Phase Shift Keying
EFR	Enhanced Full-Rate speech codex
ETSI	European Telecommunications Standards Institute
FFT	Fast Fourier Transform
FPLMTS	Future Public Land Mobile Telephone System
GPRS	General Packet Radio Service
GSM	Global System of Mobile Communications. A Pan-European digital mobile radio standard, operating at 900MHz
HSCSD	High-Speed Circuit Switched Data

I/Q EQ	In-phase/Quadrature-phase Equaliser
IOWEF	Input Output Weight Enumerating Function. A relationship, which describes the encoder as a function of the input word and codeword Hamming weights.
ISI	Inter Symbol interference
ITU	International Telecommunications Union, formerly the CCITT, standardisation group
L-RC	L-Raised Cosine modulation. It is a class of Digital Phase Modulation, where L denotes the length of spreading in the modulator filter.
LLR	Log Likelihood Ratio
Log-MAP	An algorithm based on the Maximum *A Posteriori* algorithm. It gives identical performance but at lower complexity sinc eit is implemented in the logarithm domain.
M-QAM	Multi-level Quadrature Amplitude Modulation Scheme, where M represents the modulation level.
ML	Maximum likelihood.
MLSE	Maximum Likelihood Sequence Estimation
MS	A common abbreviation for Mobile Station
NTT	Nippon Telephone and Telegraph
ODMA	Opportunity Driven Multiplex Access
OFDM	Orthogonal Frequency Division Multiplex
OQPSK	Offset Quadrature Phase Shift Keying
PCCC	Parallel Concatenated Convolutional Code. Parallel constituent encoder are separated by an interleaver between encoders.
PSD	Power Spectral Density
QPSK	Quadrature Phase Shift Keying
RBF	Radial Basis Function

RSC	Recursive Systematic Convolutional
SCCC	Serial Concatenated Convolutional Code. Constituent encoders are serially cascaded with an interleaver between encoders.
SISO	Soft-In/Soft-Out. An algorithm, which accepts soft values and returns a value reflecting the reliability of the transmitted information.
SOVA	Soft Output Viterbi Algorithm. This is based on the Viterbi Algorithm and returns the reliability of the transmitted information after determining the most likely sequence.
TACS	Total Access Communication System
TCM	Trellis-Coded Modulation
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
$\mathrm{CSV}_{\mathrm{EQ}}$	Correlative state vector used in the VE states
TU	Typical urban channel impulse response
UMTS	Universal Mobile Telecommunication System
VE	Viterbi Equaliser
WCDMA	Wideband Code Division Multiple Access

Bibliography

[1] F. C. A. Brooks, B. L. Yeap, J. P. Woodard, and L. Hanzo, "A Sixth-rate, 3.8 kbps GSM-like Speech Transceiver," in *Proceedings of the 3rd ACTS Mobile Communication Summit*, vol. 2, (Rhodes, Greece), pp. 544–548, 8-11 June 1998.

[2] M. Mouly and M. Pautet, *The GSM System for Mobile Communications.* Michel Mouly and Marie-Bernadette Pautet, 1992.

[3] R. Steele and L. Hanzo(ed), *Mobile radio communications.* John-Wiley IEEE Press, 2 ed., 1999.

[4] T. Ojanperä and R. Prasad, *Wideband CDMA for Third Generation Mobile Communications.* Artech House, 1998.

[5] T. Ojanperä, "Overview of research activities for third generation mobile communication," *Wireless Communications TDMA vs CDMA, S.G. Glisic and P. A. Leppannen (editors), Kluwer Academic Publishers*, pp. 415–446, 1997.

[6] L. Hanzo, W. Webb, and T. Keller, *Single and Multicarrier Quadrature Amplitude Modulation.* John-Wiley IEEE Press, 2000.

[7] J. B. Anderson, T. Aulin, and C.-E. Sundberg, *Digital phase modulation.* Plenum Press, New York, 1986.

[8] K. Hirade and K. Murota, "GMSK modulation for digital mobile radio telephony," *IEEE Transactions On Communications*, vol. 29, pp. 1044–1050, July 1981.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," in *Proceedings of the International Conference on Communications*, (Geneva, Switzerland), pp. 1064–1070, 23-26 May 1993.

[10] C. Douillard, A. Picart, M. Jézéquel, P. Didier, C. Berrou, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *European Transactions on Communications*, vol. 6, pp. 507–511, September-October 1995.

[11] ETSI, *Digital Cellular Telecommunications System (Phase 2+); High Speed Circuit Switched Data (HSCSD) - Stage 1; (GSM 02.34 Version 5.2.1).* European Telecommunications Standards Institute, Sophia Antipolis, Cedex, France, July 1997.

[12] ETSI, *Digital Cellular Telecommunications System (Phase 2+); General Packet Radio Service (GPRS); Overall Description of the GPRS Radio Interface, Stage 2 (GSM 03.64 Version 5.2.0).* European Telecommunications Standards Institute, Sophia Antipolis, Cedex, France, January 1998.

[13] I. Gerson, M. Jasiuk, J.-M. Muller, J. Nowack, and E. Winter, "Speech and channel coding for the half-rate GSM channel," *Proceedings ITG-Fachbericht*, vol. 130, pp. 225–233, November 1994.

[14] R. A. Salami, C. Laflamme, B. Besette, J.-P. Adoul, K. Jarvinen, J. Vainio, P. Kapanen, T. Hankanen, and P. Haavisto, "Description of the GSM enhanced full rate speech codec," in *Proc. of ICC'97*, 1997.

[15] G. Bauch and V. Franz, "Iterative Equalisation and Decoding for the GSM-System," in *Proceedings of the IEEE 48th Vehicular Technology Conference*, (Ottawa, Canada), pp. 2262–2266, 18-21 May 1998.

[16] F. Burkert, G. Caire, T. H. Joachim Hagenauer, and G. Lechner, "'Turbo' Decoding with Unequal Error Protection applied to GSM Speech Coding," in *Proceedings of the IEEE Global Telecommunications Conference 1996*, (London, United Kingdom), pp. 2044–2048, 18-22 November 1996.

[17] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimising Symbol Error Rate," *IEEE Transactions on Information Theory*, pp. 284–287, March 1974.

[18] P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain," in *Proceedings of the International Conference on Communications*, (Seattle, United States), pp. 1009–1013, 18-22 June 1995.

[19] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, pp. 909–926, May 1998.

[20] S. Benedetto and G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 409–428, March 1996.

[21] D. Raphaeli and Y. Zarai, "Combined turbo equalization and turbo decoding," *IEEE Communications Letters*, vol. 2, pp. 107–109, April 1998.

[22] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, September 1959.

[23] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error-correcting binary group codes," *Information and Control*, vol. 3, pp. 68–79, March 1960.

[24] B. L. Yeap, T. H. Liew, J. Hàmorskỳ, and L. Hanzo, "Block-based turbo-coded, turbo-equalised partial-response modulation for dispersive mobile channels," in *Proceedings of Microcoll 1999*, (Budapest, Hungary), pp. 71–74, 21-24 March 1999.

[25] B. L. Yeap, T. H. Liew, J. Hàmorskỳ, and L. Hanzo, "Comparative study of turbo equalisers using convolutional codes and block-based turbo codes for GMSK modulation," in *Proceedings of the IEEE Conference on Vehicular Technology 1999*, (Amsterdam, Netherlands), pp. 1000–1004, 19-22 September 1999.

[26] B. L. Yeap, C. H. Wong, and L. Hanzo, "Reduced complexity in-phase/quadrature-phase turbo equalisation with iterative channel estimation," in *IEEE Global Telecommunications Conference*, (San Francisco, United States). Manuscript in preparation for publication.

[27] C. H. Wong, B. L. Yeap, and L. Hanzo, "Wideband burst-by-burst adaptive modulation with turbo equalization and iterative channel estimation," in *Proceedings of the IEEE Vehicular Technology Conference 2000*, (Tokyo, Japan), 15-18 May 2000. To be published.

[28] P. J. Cherriman, B. L. Yeap, and L. Hanzo, "Turbo-equalised Interactive Videotelephony over GSM," *IEEE Transactions on Circuits and Systems for Video Technology*, 2000. Manuscript in preparation for publication.

[29] A. Knickenberg, B. L. Yeap, J. Hàmorskỳ, M. Breiling, and L. Hanzo, "Non-iterative Joint Channel Equalisation and Channel Decoding," *IEE Electronics Letters*, vol. 35, pp. 1628–1630, 16 September 1999.

[30] T. H. Liew, B. L. Yeap, J. P. Woodard, and L. Hanzo, "Modified MAP Algorithm for Extended Turbo BCH Codes and Turbo Equalisers," in *Proceedings of the First International Conference on 3G Mobile Communication Technologies, Jan 2000*, (London, United Kingdom), 27-29 March To be published.

[31] M. S. Yee, B. L. Yeap, and L. Hanzo, "Iterative Radial Basis Function Assisted Turbo Equalisation," in *Proceedings of the IEEE Vehicular Technology Conference 2000*, (Tokyo, Japan), 15-18 May 2000. To be published.

[32] J. D. Parsons, *The Mobile Radio Propagation Channel.* London: Pentech Press, 1992.

[33] S. R. Saunders, *Antennas and Propagation for Wireless Communication Systems* . John Wiley & Sons, 1999.

[34] T. S. Rappaport, *Wireless Communications : Principles and Practice.* Prentice Hall, 1996.

[35] "GSM Recommendation 05.05: Transmission and reception," November 1988.

[36] A. B. Carlson, *Communication Systems : An Introduction to Signals and Noise in Electrical Communication.* McGraw-Hill, 1986.

[37] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, March 1973.

[38] R. Debuda, "Coherent demodulation of frequency shift keying with low deviation ratio," *IEEE Transactions on Communications*, vol. COM-20, pp. 429–435, June 1972.

[39] S. Pasupathy, "Minimum shift keying: A spectrally efficient modulation," *IEEE Communications Magazine*, vol. 17, pp. 14–22, July 1979.

[40] B. Sklar, *Digital communications: Fundamental and applications.* Prentice-Hall, 1988.

[41] M. K. Simon and C. Wang, "Differential detection of Gaussian MSK in a mobile radio environment," *IEEE Transactions Vehicular Technology*, vol. 33, pp. 307–320, November 1984.

[42] J. Kreyszig, *Advanced engineering mathematics.* Wiley, 7th edition ed., 1993.

[43] D. G. Manolakis and J. G. Proakis, *Introduction to digital signal processing.* Macmillan Publishing Company, New York, 1995.

[44] C. Berrou and A. Glavieux, "Near optimum error-correcting coding and decoding: Turbo codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[45] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (Turbo) codes," in *Proceedings of the IEEE Global Telecommunications Conference 1994*, (San Francisco, United States), pp. 1298–1303, December 1994.

[46] J. Hagenauer, E. Offer, and L. Papke, "Iterative Coding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–437, March 1996.

[47] M. Moher, "Decoding via cross-entropy minimization," in *Proceedings of the IEEE Global Telecommunications Conference 1993*, (Houston, United States), pp. 809–813, 29 November - 2 December 1993.

[48] C. Heegard and S. B. Wicker, *Turbo Coding.* Kluwer International, 1999.

[49] C. Berrou, "Some Clinical Aspects of Turbo Codes," in *Proceedings of the International Symposium on Turbo Codes & Related Topics*, (Brest, France), pp. 26–31, 3-5 September 1997.

[50] D. A. Johnson, S. W. Wales, and P. H. Waters, "Equalisers for GSM," *IEE Colloquium (Digest)*, no. 21, pp. 1/1–1/6, 1990.

[51] J. C. S. Cheung and R. Steele, "Soft-decision feedback equalizer for continuous phase modulated signals in wideband mobile radio channels," *IEEE Transactions on Communications*, vol. 42, p. 42, February/March/April 1994.

[52] W. Koch and A. Baier, "Optimum and Sub-Optimum Detection of Coded Data Disturbed by Time-Varying Inter-Symbol Interference," in *Proceedings of the IEEE Global Telecommunications Conference 1990*, (San Diego, United States), pp. 1679–1684, 2-5 December 1990.

[53] J. A. Erfanian, S. Pasupathy, and G. Gulak, "Reduced Complexity Symbol Detectors with Parallel Structures for ISI Channels," *IEEE Transactions on Communications*, vol. 42, pp. 1661–1671, February/March/April 1994.

[54] J. Hagenauer, "Source-Controlled Channel Decoding," *IEEE Transactions on Communications*, vol. 43, pp. 2449–2457, September 1995.

[55] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications," in *Proceedings of the IEEE Global Telecommunications Conference 1989*, (Dallas, United States), pp. 1680–1686, 27-30 November 1989.

[56] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A Low Complexity Soft-Output Viterbi Decoder Architecture," in *Proceedings of the International Conference on Communications*, (Geneva, Switzerland), pp. 737–740, 23-26 May 1993.

[57] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 1982.

[58] J. P. Woodard, "Turbo Coding for Gaussian and Fading Channels," tech. rep., Southampton University, 1998.

[59] M. Breiling, "Turbo Coding Simulation Results," tech. rep., Universität Karlsruhe and Southampton University, 1997.

[60] P. Robertson and T. Wórz, "Coded Modulation Scheme Employing Turbo Codes," *IEE Electronics Letters*, vol. 31, pp. 1546–1547, 31 August 1995.

[61] A. S. Barbulescu and S. S. Pietrobon, "Interleaver Design for Turbo Codes," *IEE Electronics Letters*, vol. 30, pp. 2107–2108, 8 December 1994.

[62] Office for Official Publications of the European Communities, Luxembourg, *COST 207: Digital land mobile radio communications, final report*, 1989.

[63] "GSM Recommendation 05.03: Channel coding," November 1988.

[64] P. Jung and M. Naβhan, "Dependence of the error performance of turbo-codes on the interleaver structure in short frame transmission systems," *IEE Electronics Letters*, vol. 30, pp. 287–288, 17 February 1994.

[65] G. Bauch, H. Khorram, and J. Hagenauer, "Iterative Equalization and Decoding in Mobile Communications Systems," in *Proceedings of the European Personal Mobile Communications Conference*, (Bonn, Germany), pp. 301–312, 30 September - 2 October 1997.

[66] M. J. Gertsman and J. L. Lodge, "Symbol-by-symbol MAP demodulation of CPM and PSK signals on Rayleigh flat-fading channels," *IEEE Transactions on Communications*, vol. 45, pp. 788–799, July 1997.

[67] I. D. Marsland, P. T. Mathiopoulos, and S. Kallel, "Non-coherent turbo equalization for frequency selective Rayleigh fast fading channels," *Proceedings of the International Symposium on Turbo Codes & Related Topics*, pp. 196–199, 1997.

[68] Q. Dai and E. Shwedyk, "Detection of bandlimited signals over frequency selective Rayleigh fading channels," *IEEE Transactions on Communications*, pp. 941–950, February/March/April 1994.

[69] F. Jordan and K.-D. Kammeyer, "Study on iterative decoding techniques applied to GSM full-rate channels," in *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, (Pisa, Italy), pp. 1066–1070, 29 September - 2 October 1998.

[70] G.-F. Qin, S.-D. Zhou, and Y. Yao, "Iterative decoding of GMSK modulated convolutional code," *IEE Electronics Letters*, vol. 35, pp. 810–811, 13th May 1999.

[71] K. R. Narayanan and G. L. Stüber, "A serial concatenation approach to iterative demodulation and decoding," *IEEE Transactions on Communications*, vol. 47, pp. 956–961, July 1999.

[72] A. J. Viterbi, "Error bounds for convolutional codes and asymtotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.

[73] L. Lin and R. S. Cheng, "Improvements in SOVA-based decoding for turbo codes," in *Proceedings of the IEEE International Conference on Communications*, vol. 3, (Montreal, Canada), pp. 1473–1478, 8-12 June 1997.

[74] Y. Liu, M. Fossorier, and S. Lin, "MAP algorithms for decoding linear block code based on sectionalized trellis diagrams," in *Proceedings of the IEEE Global Telecommunications Conference 1998*, vol. 1, (Sydney, Australia), pp. 562–566, 8-12 November 1998.

[75] Y. V. Svirid and S. Riedel, "Threshold decoding of turbo-codes," *Proceedings of the IEEE International Symposium on Information Theory*, p. 39, September 1995.

[76] J. S. Reeve, "A parallel Viterbi decoding algorithm," *IEEE Communications Letters*. Submitted for publication.

[77] S. Benedetto, R. Garello, and G. Montorsi, "A search for good convolutional codes to be used in the construction of turbo codes," *IEEE Transactions on Communications*, vol. 46, pp. 1101–1105, September 1998.

[78] S. Benedetto and G. Montorsi, "Design of Parallel Concatenated Convolutional Codes," *IEEE Transactions on Communications*, vol. 44, pp. 591–600, May 1996.

[79] M. S. C. Ho, S. S. Pietrobon, and T. Giles, "Improving the constituent codes of turbo encoders," in *Proceedings of the IEEE Global Telecommunications Conference 1998*, vol. 6, (Sydney, Australia), pp. 3525–3529, 8-12 November 1998.

[80] A. Ushirokawa, T. Okamura, and N. Kamiya, "Principles of Turbo codes and their application to mobile communications," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E81-A, pp. 1320–1329, July 1998.

[81] A. Shibutani, H. Suda, and F. Adachi, "Complexity reduction of turbo decoding," in *Proceedings of the IEEE Vehicular Technology Conference 1999*, (Amsterdam, Netherlands), pp. 1570–1574, 19-22 September 1999.

[82] J. Yuan, B. Vucetic, and W. Feng, "Combined turbo codes and interleaver design," *IEEE Transactions on Communications*, vol. 47, no. 4, pp. 484–487, 1999.

[83] B. H. M. Z. Wang, "Interleaver design for turbo codes," in *Proceedings of the International Conference on Information, Communications and Signal Processing, ICICS*, vol. 1, (Singapore, Singapore), pp. 453–455, 9-12 September 1997.

[84] M. Breiling and L. Hanzo, "Optimum non-iterative turbo decoder," in *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 2, (Helsinki, Finland), pp. 714–718, 1-4 September 1997.

[85] M. Breiling and L. Hanzo, "Non-iterative optimum super-trellis decoding of turbo codes," *IEE Electronics Letters*, vol. 33, pp. 848–849, 8 May 1997.

[86] P. Hoeher and J. Lodge, "Turbo DPSK': iterative differential PSK demodulation and channel decoding," *IEEE Transactions on Communications*, vol. 47, pp. 837–843, June 1999.

[87] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, March 1996.

[88] H. Nickl, J. Hagenauer, and F. Burkett, "Approaching Shannon's Capacity Limit by 0.27 dB Using Simple Hamming Codes," *IEEE Communications Letters*, vol. 1, pp. 130–132, September 1997.

[89] R. Pyndiah, "Iterative Decoding of Product Codes: Block Turbo Codes," in *Proceedings of the International Symposium on Turbo Codes & Related Topics*, (Brest, France), pp. 71–79, 3-5 September 1997.

[90] "Digital Video Broadcast (DVB): Framing structure, channel coding and modulation for 11/12 GHz Satellite Services," pp. 12–14, August 1997.

[91] Ömer. F. Açikel and W. E. Ryan, "Punctured turbo-codes for BPSK/QPSK channels," *IEEE Transactions on Communications*, vol. 47, pp. 1315–1323, September 1999.

[92] A. Klein, R. Pirhonen, J. Sköld, and R. Suoranta, "FRAMES Multiple Access Model - Wideband TDMA with and without spreading," in *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications 1997*, (Helsinki, Finland), pp. 37–41, 1-4 September 1997.

[93] A. Glavieux, C. Laot, and J. Labat, "Turbo Equalization over a frequency selective channel," in *Proceedings of the International Symposium on Turbo Codes & Related Topics*, (Brest, France), pp. 96–102, 3-5 September 1997.

[94] A. D. Whalen, *Detection of signals in noise.* Academic Press Inc, New York and London, 1971.

[95] J. Proakis, *Digital communications.* McGraw-Hill, 1995.

[96] S. Haykin, *Adaptive Filter Theory.* Prentice Hall, 1996.

[97] C. H. Wong, *Wideband adaptive full response multilevel transceivers and equalizers.* PhD thesis, University of Southampton, 1999.

[98] R. Raheli, A. Polydoros, and C.-K. Tzou, "Per-survivor processing: A general approach to MLSE in uncertain environments," *IEEE Transactions on Communications*, vol. 43, pp. 354–364, February/March/April 1995.

[99] R. Raheli, G. Marino, and P. Castoldi, "Per-survivor processing and tentative decisions: what is in between?," *IEEE Transactions on Communications*, vol. 44, pp. 127–129, February 1996.

[100] K. Chugg and A. Polydoros, "MLSE for an unknown channel - part I: Optimality considerations," *IEEE Transactions on Communications*, vol. 44, pp. 836–846, July 1996.

[101] K. Chugg and A. Polydoros, "MLSE for an unknown channel - part II: Tracking performance," *IEEE Transactions on Communications*, vol. 44, pp. 949–958, August 1996.

[102] L. Davies, I. Collings, and P. Hoeher, "Joint MAP equalization and channel estimation for frequency-selective fast fading channels," in *Proceedings of the Seventh Communication Theory Mini-Conference in conjunction with IEEE Global Telecommunications Conference '98,*, (Sydney, Australia), pp. 53–58, 8-12 November 1998.

[103] I. Bar-David and A. Elia, "Augmented APP (A^2P^2) module for a posteriori probability calculation and channel parameter tracking," *IEEE Communications Letters*, vol. 3, pp. 18–20, January 1999.

[104] T. M. Duman and M. Salehi, "New performance bounds for turbo codes," *IEEE Transactions on Communications*, vol. 46, pp. 717–723, June 1998.

[105] A. Viterbi and A. Viterbi, "Improved union bound on linear codes for the input-binary AWGN channel, with applications to turbo codes," in *Proceedings of the International Symposium on Information Theory*, (Cambridge, United States), p. 29, 16-21 August 1998.

[106] I. Sason and S. Shamai, "Improved upper bounds on the performance of parallel serial concatenated turbo codes via their ensemble distance spectrum," in *Proceedings of the International Symposium on Information Theory*, (Cambridge, United States), p. 30, 16-21 August 1998.

[107] B. Spinnler and J. B. Huber, "Design of hyper states for reduced-state sequence estimation," *International Journal of Electronics and Communications*, vol. 50, pp. 17–26, January 1996.

[108] P. D. Alexander, M. C. Reed, J. A. Asenstorfer, and C. B. Schlegel, "Iterative Multiuser Interference Reduction: Turbo CDMA," *IEEE Transactions on Communications*, vol. 47, pp. 1008–1014, July 1999.

[109] M. C. Reed, C. B. S. P. D. Alexander, and J. A. Asenstorfer, "Iterative Multiuser Detection for CDMA with FEC: Near-single-user Performance," *IEEE Transactions on Communications*, vol. 46, pp. 1693–1699, December 1998.

[110] M. C. Valenti and B. D. Woerner, "Iterative multiuser detection for convolutionally coded asynchronous DS-CDMA," in *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, vol. 1, (Boston, United States), pp. 213–217, 8-11 September 1998.

[111] X. Wang and V. H. Poor, "Iterative (Turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Transactions on Communications*, vol. 47, pp. 1046–1061, August 1999.

[112] H. Yoshino, K. Fukawa, and H. Suzuki, "Interference canceling equaliser (ICE) for mobile radio communications," in *Proceedings of the IEEE International Conference on Communication*, vol. 3, (New Orleans, USA), pp. 1427–1432, IEEE, Piscataway, NJ, USA, 1-5 May 1994.

[113] R. Berangi, P. Leung, and M. Faulkner, "Cochannel interference cancellation for mobile communication systems," in *Annual International Conference on Universal Personal Communications Record*, vol. 1, (Cambridge, United States), pp. 438–442, 29 September - 1 October 1996.

[114] S. W. Wales, "Technique for cochannel interference suppression in TDMA mobile radio systems," *IEE Proc.-Communi.*, vol. 142, pp. 106–114, April 1995.

[115] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets. Part I: Introduction," *IEEE Communications Magazine*, vol. 25, pp. 5–11, February 1987.

[116] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets. Part II: State of the art," *IEEE Communications Magazine*, vol. 25, pp. 12–22, February 1987.

Index

Author Index