Turbo Coding, Turbo Equalisation and Space-Time Coding

by

©L. Hanzo, T.H. Liew, B.L. Yeap
Department of Electronics and Computer Science,
University of Southampton, UK

Contents

[1] This chapter is based on B.L. Yeap, T.H. Liew, J.Hámorský and L. Hanzo: Comparative study of turbo equalization schemes using convolutional, convolutional turbo and block-turbo codes, to appear in IEEE Tr. on Wireless Communications, 2002

This books is dedicated to the numerous contributors of this field, many of whom are listed in the Author Index

Contributors of the book:
Chapter 4: J.P. Woddard, L. Hanzo
Chapter 5: M. Breiling, L. Hanzo
Chapter 7: T.H. Liew, L.L. Yang, L. Hanzo
Chapter 8: S.X. Ng, L. Hanzo

Part I

Convolutional and Block Coding

Part II

Turbo Convolutional and Turbo Block Coding

Chapter 4

Turbo Convolutional Coding

J.P. Woodard, L. Hanzo[1]

4.1 Introduction

In Chapter 1 a rudimentary introduction to convolutional codes and their decoding using the Viterbi algorithm was provided. In this chapter we introduce the concept of turbo coding using convolutional codes as their constituent codes. Our discussions will become deeper, relying on basic familiarity with convolutional coding. In Chapter 5 we then further elaborate on the relationship between conventional convolutional codes and turbo convolutional codes and highlight the relationship between their trellis structure.

Turbo coding was proposed in 1993 by Berrou, Glavieux and Thitimajashima, who reported excellent coding gain results [21], approaching Shannonian predictions. The information sequence is encoded twice, with an interleaver between the two encoders serving to make the two encoded data sequences approximately statistically independent of each other. Often half rate Recursive Systematic Convolutional (RSC) encoders are used, with each RSC encoder producing a systematic output which is equivalent to the original information sequence, as well as a stream of parity information. The two parity sequences can then be punctured before being transmitted along with the original information sequence to the decoder. This puncturing of the parity information allows a wide range of coding rates to be realised, and often half the parity information from each encoder is sent. Along with the original data sequence this results in an overall coding rate of 1/2.

At the decoder two RSC decoders are used. Special decoding algorithms must be used which accept soft inputs and give soft outputs for the decoded sequence. These soft inputs and outputs provide not only an indication of whether a particular bit was a 0 or a 1, but also a likelihood ratio which gives the probability that the bit has been correctly decoded. The turbo decoder operates iteratively. In the first iteration the first RSC decoder provides a soft

[1]This chapter is based on J.P. Woodard, L. Hanzo: Comparative Study of Turbo Decoding Techniques: An Overview; IEEE Transactions on Vehicular Technology, Nov. 2000, Vol. 49, No. 6, pp 2208-2234 ©IEEE

output giving an estimation of the original data sequence based on the soft channel inputs alone. It also provides an *extrinsic* output. The extrinsic output for a given bit is based not on the channel input for that bit, but on the information for surrounding bits and the constraints imposed by the code being used. This extrinsic output from the first decoder is used by the second RSC decoder as *a-priori information*, and this information together with the channel inputs are used by the second RSC decoder to give its soft output and extrinsic information. In the second iteration the extrinsic information from the second decoder in the first iteration is used as the a-priori information for the first decoder, and using this a-priori information the decoder can hopefully decode more bits correctly than it did in the first iteration. This cycle continues, with at each iteration both RSC decoders producing a soft output and extrinsic information based on the channel inputs and a-priori information obtained from the extrinsic information provided by the previous decoder. After each iteration the Bit Error Rate (BER) in the decoded sequence drops, but the improvements obtained with each iteration falls as the number iterations increases so that for complexity reasons usually only between 4 and 12 iterations are used.

In their original proposal Berrou et al. [21] invoked a modified version of the classic minimum bit error rate maximum aposteriory algorithm (MAP) due to Bahl et al [20] in the the above iterative structure for decoding the constituent codes. Since the conception of turbo codes a large body of work has been carried out in the area, aiming for example to reduce the decoder complexity, as suggested by Robertson, Villebrun and Hoeher [59] as well as by Berrou et al [122]. Le Goff, Glavieux and Berrou [62], Wachsmann and Huber [63] as well as Robertson and Worz [64] suggested using the codes in conjunction with bandwidth efficient modulation schemes. Further advances in understanding the excellent preformance of the codes are due, for example, to Benedetto and Montorsi [65, 66] as well as to Perez, Seghers and Costello [67]. A number of authors, including Hagenauer, Offer and Papke, as well as Pyndiah [68, 123] extended the turbo concept to parallel concatenated block codes. Jung and Naßhan [72] characterised the coded performance under the constraints of short transmission frame length, which is characteristic of speech systems. In collaboration with Blanz they also applied turbo codes to a CDMA system using joint detection and antenna diversity [74]. Barbulescu and Pietrobon [75] as well as a number of other authors addressed the equally important issues of interlever design. Due to space limitations here we have to curtail listing the range of further contributors in the field, without whose advances this treatise could not have been written. Of particular importance is to note the tutorial paper authored by Sklar [76].

Here we embark on describing turbo codes in more detail. Specifically, in Section 4.2 we detail the encoder used, while in Section 4.3 the decoder is portrayed. Then in Section 4.4 we characterise the performance of various turbo codes over Gaussian channels using BPSK. Then in Section 4.5 we discuss the employment of turbo codes over Rayleigh channels, and characterise the system's speech performance, when using the G.729 speech codec.

4.2 Turbo Encoder

The general structure used in turbo encoders is shown in Figure 4.1. Two component codes are used to code the same input bits, but an interleaver is placed between the encoders. Generally Recursive Systematic Convolutional (RSC) codes are used as the component codes, but

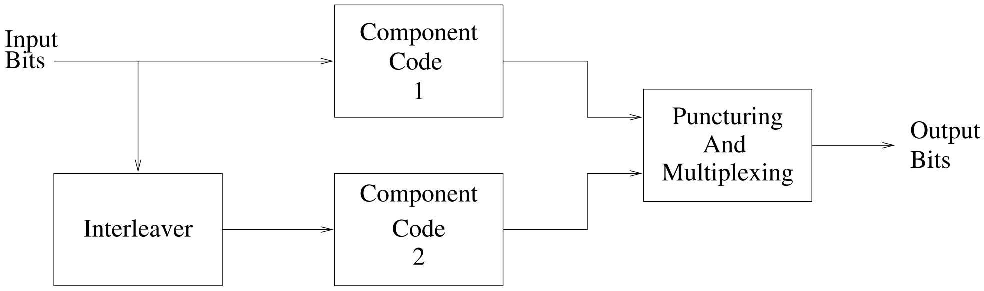


Figure 4.1: Turbo Encoder Schematic

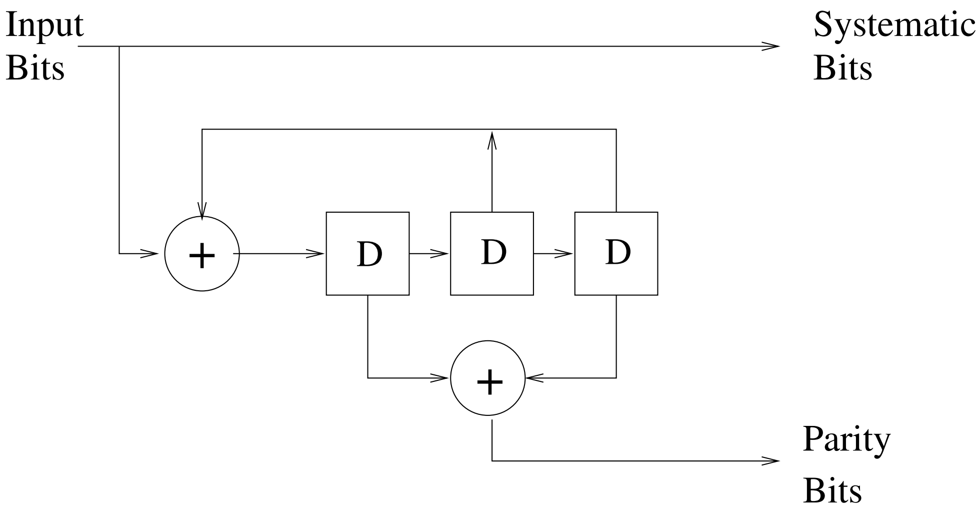


Figure 4.2: Recursive Systematic Convolutional (RSC) Encoder

it is possible to achieve good performance using a structure like that seen in Figure 4.1 with the aid of other components codes, such as for example block codes [68, 123]. Furthermore, it is also possible to employ more than two component codes. However, in this chapter we concentrate entirely on the standard turbo encoder structure using two RSC codes. Turbo codes using block codes as component codes are described in Chapter 6.

The outputs from the two component codes are then punctured and multiplexed. Usually both component RSC codes are half rate, giving one parity bit and one systematic bit output for every input bit. Then to give an overall coding rate of one half, half the output bits from the two encoders must be punctured. The arrangement that is often favoured and that we have used in our work, is to transmit all the systematic bits from the first RSC encoder, and half the parity bits from each encoder. Note that the systematic bits are rarely punctured, since this degrades the performance of the code more dramatically, than puncturing the parity bits.

Figure 4.2 shows the K=3 RSC code we have used as the component codes in most of our simulations. This code has generator polynomials 7 (for the feedback polynomial) and 5.

Parallel concatenated codes had been well investigated before Berou's breakthrough 1993

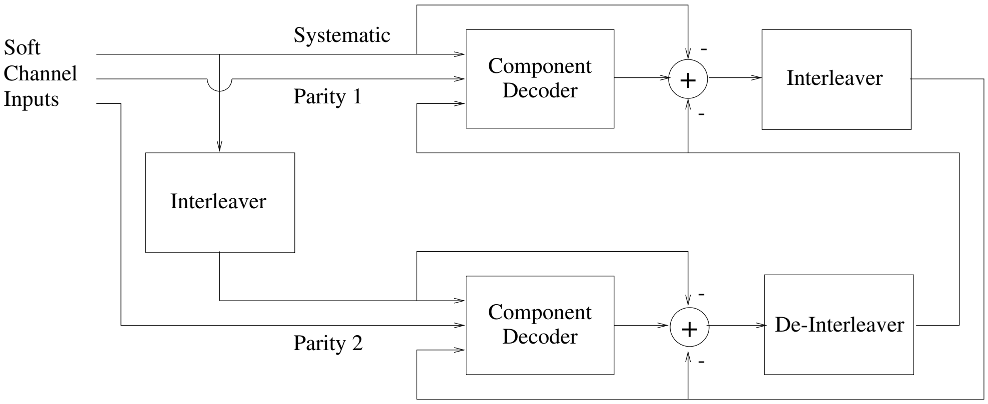


Figure 4.3: Turbo Decoder Schematic

paper [21], but the dramatic improvement in performance that turbo codes gave arose because of the interleaver used between the encoders, and because recursive codes were used as the component codes. Recently theoretical paper have been published [65, 66] which try to explain the remarkable performance of turbo codes. It appears that turbo codes can be thought of as having a performance gain proportional to the interleaver length used. However the decoding complexity per bit does not depend on the interleaver length. Therefore extremely good performance can be achieved with reasonable complexity by using very long interleavers. However for many important applications, such as speech transmission, extremely long frame lengths are not practical because of the delays they result in. Therefore in this chapter we have also investigated the use of turbo codes in conjunction with short frame lengths of the order of 100 bits.

4.3 Turbo Decoder

4.3.1 Introduction

The general structure of an iterative turbo decoder is shown in Figure 4.3. Two component decoders are linked by interleavers in a structure similar to that of the encoder. As seen in the figure, each decoder takes three inputs – the systematically encoded channel output bits, the parity bits transmitted from the associated component encoder, and the information from the other component decoder about the likely values of the bits concerned. This information from the other decoder is referred to as a-priori information. The component decoders have to exploit both the inputs from the channel and this a-priori information. They must also provide what are known as soft outputs for the decoded bits. This means that as well as providing the decoded output bit sequence, the component decoders must also give the associated probabilities for each bit that it has been correctly decoded. The soft outputs are typically represented in terms of the so-called Log Likelihood Ratios (LLRs), the magnitude of which gives the sign of the bit, and the amplitude the probability of a correct decision. These LLRs are described in Section 4.3.2. Two suitable decoders are the Soft Output Viterbi Algorithm (SOVA) [60] and the Maximum A-Posteriori (MAP) [20] algorithm, which are described in Sections 4.3.6 and 4.3.3, respectively.

The decoder of Figure 4.3 operates iteratively, and in the first iteration the first component decoder takes channel output values only, and produces a soft output as its estimate of the data bits. The soft output from the first encoder is then used as additional information for the second decoder, which uses this information along with the channel outputs to calculate its estimate of the data bits. Now the second iteration can begin, and the first decoder decodes the channel outputs again, but now with additional information about the value of the input bits provided by the output of the second decoder in the first iteration. This additional information allows the first decoder to obtain a more accurate set of soft outputs, which are then used by the second decoder as a-priori information. This cycle is repeated, and with every iteration the Bit Error Rate (BER) of the decoded bits tends to fall. However the improvement in performance obtained with increasing numbers of iterations decreases as the number of iterations increases. Hence, for complexity reasons, usually only about 8 iterations are used.

Due to the interleaving used at the encoder, care must be taken to properly interleave and de-interleave the LLRs which are used to represent the soft values of the bits, as seen in Figure 4.3. Furthermore, because of the iterative nature of the decoding, care must be taken not to re-use the same information more than once at each decoding step. For this reason the concept of so-called extrinsic and intrinsic information was used in the original paper by Berou et al. describing iterative decoding of turbo codes [21]. These concepts and the reason for the subtraction circles shown in Figure 4.3 are described in Section 4.3.4.

Other, non-iterative, decoders have been proposed [124,125] which give optimal decoding of turbo codes. However the improvement in performance over iterative decoders was found to be only about 0.35 dB, and they are hugely complex. Therefore the iterative scheme shown in Figure 4.3 is commonly used. We now proceed with describing the concepts and algorithms used in the iterative decoding of turbo codes, commencing with the portrayal of logarithmic likelihood ratios.

4.3.2 Log Likelihood Ratios

The concept of Log Likelihood Ratios (LLRs) was shown by Robertson [126] to simplify the passing of information from one component decoder to the other in the iterative decoding of turbo codes, and so is now widely used in the turbo coding literature. The LLR of a data bit u_k is denoted as $L(u_k)$ and is defined to be merely the log of the ratio of the probabilities of the bit taking its two possible values, ie:

$$L(u_k) \triangleq \ln\left(\frac{\mathrm{P}(u_k = +1)}{\mathrm{P}(u_k = -1)}\right). \tag{4.1}$$

Notice that the two possible values for the bit u_k are taken to be $+1$ and -1, rather than 1 and 0. This definition of the two values of a binary variable makes no conceptual difference, but it slightly simplifies the mathematics in the derivations which follow. Hence this convention is used throughout this chanpter. Figure 4.4 shows how $L(u_k)$ varies as the probability of $u_k = +1$ varies. It can be seen from this figure that the sign of the LLR $L(u_k)$ of a bit u_k will indicate whether the bit is more likely to be $+1$ or -1, and the magnitude of the LLR gives an indication of how likely it is that the sign of the LLR gives the correct value of u_k. When the LLR $L(u_k) \approx 0$, we have $\mathrm{P}(u_k = +1) \approx \mathrm{P}(u_k = -1) \approx 0.5$, and we cannot be

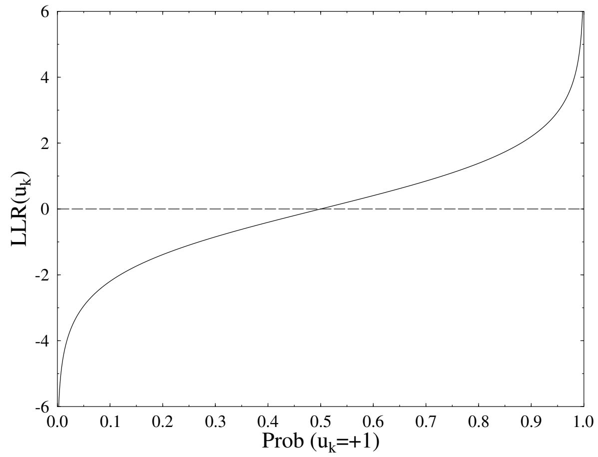


Figure 4.4: Log Likelihood Ratio $L(u_k)$ Versus the Probability of $u_k = +1$

certain about the value of u_k. Conversely, when $L(u_k) >> 0$, we have $\mathrm{P}(u_k = +1) >> \mathrm{P}(u_k = -1)$ and we can be almost certain that $u_k = +1$.

Given the LLR $L(u_k)$, it is possible to calculate the probability that $u_k = +1$ or $u_k = -1$ as follows. Remembering that $\mathrm{P}(u_k = -1) = 1 - \mathrm{P}(u_k = +1)$, and taking the exponent of both sides in Equation 4.1 we can write:

$$\mathrm{e}^{L(u_k)} = \frac{\mathrm{P}(u_k = +1)}{1 - \mathrm{P}(u_k = +1)}, \tag{4.2}$$

so

$$\begin{aligned} \mathrm{P}(u_k = +1) &= \frac{\mathrm{e}^{L(u_k)}}{1 + \mathrm{e}^{L(u_k)}} \\ &= \frac{1}{1 + \mathrm{e}^{-L(u_k)}}. \end{aligned} \tag{4.3}$$

Similarly

$$\begin{aligned} \mathrm{P}(u_k = -1) &= \frac{1}{1 + \mathrm{e}^{+L(u_k)}} \\ &= \frac{\mathrm{e}^{-L(u_k)}}{1 + \mathrm{e}^{-L(u_k)}}, \end{aligned} \tag{4.4}$$

and hence we can write

$$\mathrm{P}(u_k = \pm 1) = \left(\frac{\mathrm{e}^{-L(u_k)/2}}{1 + \mathrm{e}^{-L(u_k)}} \right) \cdot \mathrm{e}^{\pm L(u_k)/2}. \tag{4.5}$$

Notice that the bracketed term in this equation does not depend on whether we are interested in the probability that $u_k = +1$ or -1, and so it can be treated as a constant in certain applications, such as in Section 4.3.3 where we use this equation in the derivation of the MAP algorithm.

As well as the LLR $L(u_k)$ based on the unconditional probabilities $\mathrm{P}(u_k = \pm 1)$, we are also interested in LLRs based on conditional probabilities. For example, in channel coding theory we are interested in the probability that $u_k = \pm 1$ based, or conditioned, on some received sequence $\underline{y}$, and hence we may use the conditional LLR $L(u_k|\underline{y})$, which is defined as

$$L(u_k|\underline{y}) \triangleq \ln \left(\frac{\mathrm{P}(u_k = +1|\underline{y})}{\mathrm{P}(u_k = -1|\underline{y})} \right). \tag{4.6}$$

The conditional probabilities $\mathrm{P}(u_k = \pm 1|\underline{y})$ are known as the a-posteriori probabilities of the decoded bit u_k, and it is these a-posteriori probabilities that our soft-in soft-out decoders described in later sections attempt to find.

Apart from the conditional LLR $L(u_k|\underline{y})$ based on the a-posteriori probabilities $\mathrm{P}(u_k = \pm 1|\underline{y})$, we will also use conditional LLRs based on the probability that the receiver's matched filter output would be y_k given that the corresponding transmitted bit x_k was either $+1$ or -1. This conditional LLR is written as $L(y_k|x_k)$ and is defined as:

$$L(y_k|x_k) \triangleq \ln \left(\frac{\mathrm{P}(y_k|x_k = +1)}{\mathrm{P}(y_k|x_k = -1)} \right). \tag{4.7}$$

Notice the conceptual difference between the definitions of $L(u_k|\underline{y})$ in Equation 4.6 and $L(y_k|x_k)$ in Equation 4.7, despite these two conditional LLRs being represented with very similar notation. This contrast in the definitions of conditional LLRs is somewhat confusing, but since these definitions are widely used in the turbo coding literature, we have introduced them here.

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

$$P(y_k|x_k = +1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y_k - a)^2 \right), \tag{4.8}$$

where E_b is the transmitted energy per bit, σ^2 is the noise variance and a is the fading amplitude (we have $a = 1$ for non-fading AWGN channels). Similarly, we have

$$P(y_k|x_k = -1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y_k + a)^2 \right). \tag{4.9}$$

Therefore, when we use BPSK over a (possibly fading) Gaussian channel, we can rewrite

Equation 4.7 as

$$\begin{aligned}
L(y_k|x_k) &\triangleq \ln\left(\frac{\mathrm{P}(y_k|x_k=+1)}{\mathrm{P}(y_k|x_k=-1)}\right) \\
&= \ln\left(\frac{\exp\left(-\frac{E_b}{2\sigma^2}(y_k-a)^2\right)}{\exp\left(-\frac{E_b}{2\sigma^2}(y_k+a)^2\right)}\right) \\
&= \left(-\frac{E_b}{2\sigma^2}(y_k-a)^2\right) - \left(-\frac{E_b}{2\sigma^2}(y_k+a)^2\right) \\
&= \frac{E_b}{2\sigma^2}4a\cdot y_k \\
&= L_c y_k, \qquad (4.10)
\end{aligned}$$

where

$$L_c = 4a\frac{E_b}{2\sigma^2} \qquad (4.11)$$

is defined as the channel reliability value, and depends only on the SNR and fading amplitude of the channel. Hence, for BPSK over a (possibly fading) Gaussian channel, the conditional LLR $L(y_k|x_k)$, which is referred to as the soft output of the channel, is simply the matched filter output y_k multiplied by the channel reliability value L_c.

Having introduced log likelihood ratios, we now proceed to describe the operation of the Maximum A-Posteriori algorithm, which is one of the possible soft-in soft-out component decoders that can be used in an iterative turbo decoder.

4.3.3 The Maximum A-Posteriori Algorithm

4.3.3.1 Introduction and Mathematical Preliminaries

In 1974 an algorithm, which is known as the Maximum A-Posteriori (MAP) algorithm, was proposed by Bahl, Cocke, Jelinek and Raviv for estimating the a-posteriori probabilities of the states and the transitions of a Markov source observed, when subjected to memoryless noise. This algorithm has also become known as the BCJR algorithm, named after its inventors. They showed, how the algorithm could be used for decoding both block and convolutional codes. When employed for decoding convolutional codes, the algorithm is optimal in terms of minimising the decoded bit error rate, unlike the Viterbi algorithm [18], which minimises the probability of an incorrect path through the trellis being selected by the decoder. Thus the Viterbi algorithm can be thought of as minimising the number of *groups* of bits associated with these trellis paths, rather than the actual number of bits, which are decoded incorrectly. Nevertheless, as stated by Bahl et al. in [20], in most applications the performance of the two algorithms will be almost identical. However the MAP algorithm examines every possible path through the convolutional decoder trellis and therefore initially seemed to be unfeasibly complex for application in most systems. Hence it was not widely used before the discovery of turbo codes.

However the MAP algorithm provides not only the estimated bit sequence, but also the probabilities for each bit that it has been decoded correctly. This is essential for the iterative decoding of turbo codes proposed by Berrou et al. [21], and so MAP decoding was used in this seminal paper. Since then much work has been done to reduce the complexity of the MAP algorithm to a reasonable level. In this section we describe the theory behind the MAP algorithm as used for the soft output decoding of the component convolutional codes of turbo codes. Throughout our work it is assumed that binary codes are used.

We use Bayes' rule repeatedly throughout this section. This rule gives the joint probability of a *and* b, $P(a \wedge b)$, in terms of the conditional probability of a *given* b as

$$P(a \wedge b) = P(a|b) \cdot P(b). \tag{4.12}$$

A useful consequence of Bayes' rule is that:

$$P(\{a \wedge b\}|c) = P(a|\{b \wedge c\}) \cdot P(b|c), \tag{4.13}$$

which can be derived from Equation 4.12 by considering $x \equiv a \wedge b$ and $y \equiv b \wedge c$ as follows. From Equation 4.12 we can write

$$\begin{aligned} P(\{a \wedge b\}|c) &\equiv P(x|c) = \frac{P(x \wedge c)}{P(c)} \\ &= \frac{P(a \wedge b \wedge c)}{P(c)} \equiv \frac{P(a \wedge y)}{P(c)} \\ &= \frac{P(a|y) \cdot P(y)}{P(c)} \equiv P(a|\{b \wedge c\}) \cdot \frac{P(b \wedge c)}{P(c)} \\ &= P(a|\{b \wedge c\}) \cdot P(b|c). \end{aligned} \tag{4.14}$$

The MAP algorithm gives, for each decoded bit u_k, the probability that this bit was $+1$ or -1, given the received symbol sequence $\underline{y}$. As explained in Section 4.3.2 this is equivalent to finding the a-posteriori LLR $L(u_k|\underline{y})$, where

$$L(u_k|\underline{y}) = \ln\left(\frac{P(u_k = +1|\underline{y})}{P(u_k = -1|\underline{y})}\right). \tag{4.15}$$

Bayes' rule allows us to rewrite this equation as

$$L(u_k|\underline{y}) = \ln\left(\frac{P(u_k = +1 \wedge \underline{y})}{P(u_k = -1 \wedge \underline{y})}\right). \tag{4.16}$$

Let us now consider Figure 4.5 showing the transitions possible for the $K = 3$ RSC code shown in Figure 4.2, which we have used for the component codes in most of our work. For this $K = 3$ code there are four states, and as it is a binary code for each state two transitions are possible – one if the input bit is -1 (shown as a solid line), and one if the input bit is a $+1$ (shown as a dashed line). It can be seen from Figure 4.5 that if the previous state S_{k-1} and the present state S_k are known, then the value of the input bit u_k, which caused the transition between these two states, will be known. Hence the probability that $u_k = +1$ is equal to the probability that the transition from the previous state S_{k-1} to the present state S_k is one of the set of four possible transitions that can occur when $u_k = +1$ (ie those transitions shown with dashed lines). This set of transitions are mutually exclusive (ie only one of them could

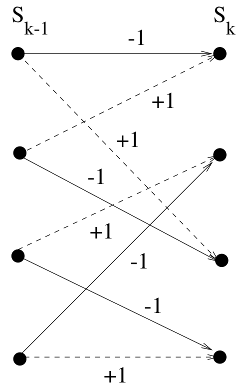


Figure 4.5: Possible Transitions in K=3 RSC Component Code

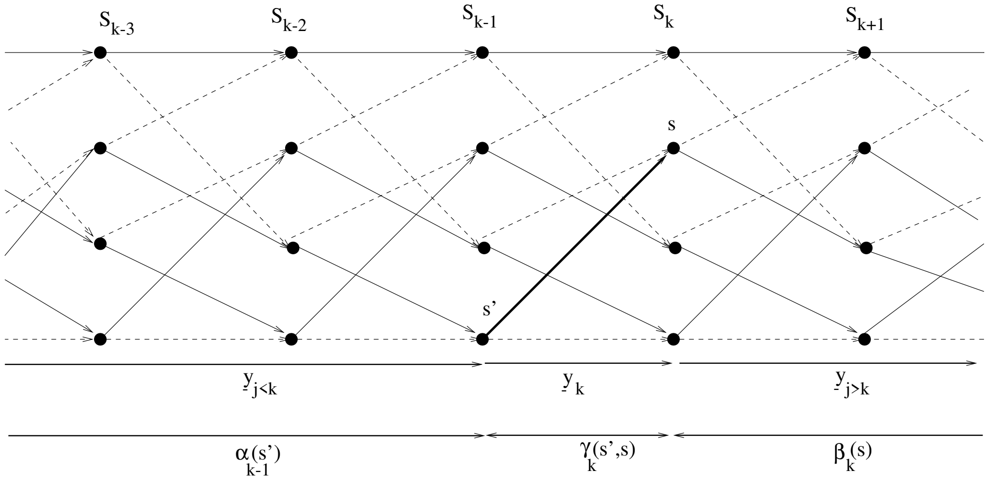


Figure 4.6: MAP Decoder Trellis for $K = 3$ RSC Code

have occured at the encoder), and so the probability that any one of them occurs is equal to the sum of their individual probabilities. Hence we can rewrite Equation 4.16 as

$$L(u_k|\underline{y}) = \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\u_k=+1}} P(S_{k-1}=\grave{s}\wedge S_k=s\wedge\underline{y})}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\u_k=-1}} P(S_{k-1}=\grave{s}\wedge S_k=s\wedge\underline{y})}\right), \tag{4.17}$$

where $(\grave{s},s)\Rightarrow u_k=+1$ is the set of transitions from the previous state $S_{k-1}=\grave{s}$ to the present state $S_k=s$ that can occur if the input bit $u_k=+1$, and similarly for $(\grave{s},s)\Rightarrow u_k=-1$. For brevity we shall write $P(S_{k-1}=\grave{s}\wedge S_k=s\wedge\underline{y})$ as $P(\grave{s}\wedge s\wedge\underline{y})$.

We now consider the individual probabilities $P(\grave{s}\wedge s\wedge\underline{y})$ from the numerator and denominator of Equation 4.17. The received sequence $\underline{y}$ can be split up into three sections: the received codeword associated with the present transition $\underline{y}_k$, the received sequence prior to the present transition $\underline{y}_{j<k}$ and the received sequence after the present transition $\underline{y}_{j>k}$. This split is shown in Figure 4.6, again for the example of our $K = 3$ RSC component code shown in Figure 4.2. We can thus write for the individual probabilities $P(\grave{s} \wedge s \wedge \underline{y})$

$$P(\grave{s} \wedge s \wedge \underline{y}) = P(\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k \wedge \underline{y}_{j>k}). \tag{4.18}$$

Using Bayes' rule of $P(a \wedge b) = P(a|b)P(b)$ and the fact that if we assume that the channel is memoryless, then the future received sequence $\underline{y}_{j>k}$ will depend only on the present state s and not on the previous state $\grave{s}$ or the present and previous received channel sequences $\underline{y}_k$ and $\underline{y}_{j<k}$, we can write

$$\begin{aligned} P(\grave{s} \wedge s \wedge \underline{y}) &= P(\underline{y}_{j>k}|\{\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\}) \cdot P(\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k) \\ &= P(\underline{y}_{j>k}|s) \cdot P(\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k). \end{aligned} \tag{4.19}$$

Again, using Bayes' rule and the assumption that the channel is memoryless, we can expand Equation 4.19 as follows:

$$\begin{aligned} P(\grave{s} \wedge s \wedge \underline{y}) &= P(\underline{y}_{j>k}|s) \cdot P(\grave{s} \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k) \\ &= P(\underline{y}_{j>k}|s) \cdot P(\{\underline{y}_k \wedge s\}|\{\grave{s} \wedge \underline{y}_{j<k}\}) \cdot P(\grave{s} \wedge \underline{y}_{j<k}) \\ &= P(\underline{y}_{j>k}|s) \cdot P(\{\underline{y}_k \wedge s\}|\grave{s}) \cdot P(\grave{s} \wedge \underline{y}_{j<k}) \\ &= \beta_k(s) \cdot \gamma_k(\grave{s}, s) \cdot \alpha_{k-1}(\grave{s}), \end{aligned} \tag{4.20}$$

where

$$\alpha_{k-1}(\grave{s}) = P(S_{k-1} = \grave{s} \wedge \underline{y}_{j<k}) \tag{4.21}$$

is the probability that the trellis is in state $\grave{s}$ at time $k-1$ and the received channel sequence up to this point is $\underline{y}_{j<k}$, as visualised in Figure 4.6,

$$\beta_k(s) = P(\underline{y}_{j>k}|S_k = s) \tag{4.22}$$

is the probability that given the trellis is in state s at time k the future received channel sequence will be $\underline{y}_{j>k}$, and lastly

$$\gamma_k(\grave{s}, s) = P(\{\underline{y}_k \wedge S_k = s\}|S_{k-1} = \grave{s}) \tag{4.23}$$

is the probability that given the trellis was in state $\grave{s}$ at time $k-1$, it moves to state s and the received channel sequence for this transition is $\underline{y}_k$.

Equation 4.20 shows that the probability $P(\grave{s} \wedge s \wedge \underline{y})$, that the encoder trellis took the transition from state $S_{k-1} = \grave{s}$ to state $S_k = s$ and the received sequence is $\underline{y}$, can be split into the product of three terms – $\alpha_{k-1}(\grave{s})$, $\gamma_k(\grave{s}, s)$ and $\beta_k(s)$. The meaning of these three probability terms is shown in Figure 4.6, for the transition $S_{k-1} = \grave{s}$ to $S_k = s$ shown by the bold line in this figure. The MAP algorithm finds $\alpha_k(s)$ and $\beta_k(s)$ for all states s throughout the trellis, ie for $k = 0, 1 \cdots N-1$, and $\gamma_k(\grave{s}, s)$ for all possible transitions from

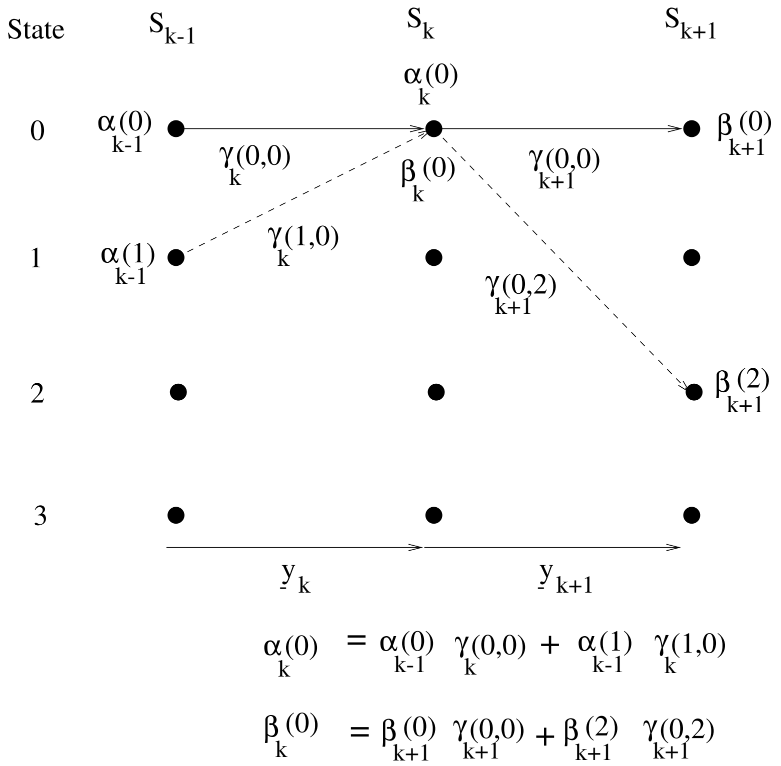


Figure 4.7: Recursive Calculation of $\alpha_k(0)$ and $\beta_k(0)$

state $S_{k-1} = \grave{s}$ to state $S_k = s$, again for $k = 0, 1 \cdots N-1$. These values are then used to find the probabilities $P(S_{k-1} = \grave{s} \wedge S_k = s \wedge \underline{y})$ of Equation 4.20, which are then used in Equation 4.17 to give the LLRs $L(u_k|\underline{y})$ for each bit u_k.These operations are summarised in the flowchart of Figure 4.8. We now describe how the values $\alpha_k(s)$, $\beta_k(s)$ and $\gamma_k(\grave{s}, s)$ can be calculated.

4.3.3.2 Forward Recursive Calculation of the $\alpha_k(s)$ Values

Consider first $\alpha_k(s)$. From the definition of $\alpha_{k-1}(\grave{s})$ in Equation 4.21 we can write

$$\begin{aligned}
\alpha_k(s) &= P(S_k = s \wedge \underline{y}_{j<k+1}) \\
&= P(s \wedge \underline{y}_{j<k} \wedge \underline{y}_k) \\
&= \sum_{\text{all } \grave{s}} P(s \wedge \grave{s} \wedge \underline{y}_{j<k} \wedge \underline{y}_k),
\end{aligned} \tag{4.24}$$

where in the last line we split the probability $P(s \wedge \underline{y}_{y<k+1})$ into the sum of joint probabilities $P(s \wedge \grave{s} \wedge \underline{y}_{j<k+1})$ over all possible previous states $\grave{s}$. Using Bayes' rule and the assumption

that the channel is memoryless again, we can proceed as follows:

$$\begin{aligned}\alpha_k(s) &= \sum_{\text{all } \grave{s}} P(s \wedge \grave{s} \wedge \underline{y}_{j<k} \wedge \underline{y}_k) \\ &= \sum_{\text{all } \grave{s}} P(\{s \wedge \underline{y}_k\}|\{\grave{s} \wedge \underline{y}_{j<k}\}) \cdot P(\grave{s} \wedge \underline{y}_{j<k}) \\ &= \sum_{\text{all } \grave{s}} P(\{s \wedge \underline{y}_k\}|\grave{s}) \cdot P(\grave{s} \wedge \underline{y}_{j<k}) \\ &= \sum_{\text{all } \grave{s}} \gamma_k(\grave{s}, s) \cdot \alpha_{k-1}(\grave{s}). \end{aligned} \tag{4.25}$$

Thus, once the $\gamma_k(\grave{s}, s)$ values are known, the $\alpha_k(s)$ values can be calculated recursively. Assuming that the trellis has the initial state $S_0 = 0$, the initial conditions for this recursion are

$$\begin{aligned}\alpha_0(S_0 = 0) &= 1 \\ \alpha_0(S_0 = s) &= 0 \quad \text{for all } s \neq 0. \end{aligned} \tag{4.26}$$

Figure 4.7 shows an example of how one $\alpha_k(s)$ value, for $s = 0$, is calculated recursively using values of $\alpha_{k-1}(\grave{s})$ and $\gamma_k(\grave{s}, s)$ for our example $K = 3$ RSC code. Notice that, as we are considering a binary trellis, only two previous states, $S_{k=1} = 0$ and $S_{k-1} = 1$, have paths to the state $S_k = 0$. Therefore $\gamma_k(\grave{s}, s)$ will be non-zero only for $\grave{s} = 0$ or $\grave{s} = 1$ and hence the summation in Equation 4.25 is over only two terms.

4.3.3.3 Backward Recursive Calculation of the $\beta_k(s)$ Values

The values of $\beta_k(s)$ can similarly be calculated recursively as shown below. From the definition of $\beta_k(s)$ in Equation 4.22, we can write $\beta_{k-1}(\grave{s})$ as

$$\beta_{k-1}(\grave{s}) = P(\underline{y}_{j>k-1}|S_{k-1} = \grave{s}), \tag{4.27}$$

and again splitting a single probability into the sum of joint probabilities and using the derivation from Bayes' rule in Equation 4.13, as well as the assumption that the channel is memoryless, we have:

$$\begin{aligned}\beta_{k-1}(\grave{s}) &= P(\underline{y}_{j>k-1}|\grave{s}) \\ &= \sum_{\text{all } s} P(\{\underline{y}_{j>k-1} \wedge s\}|\grave{s}) \\ &= \sum_{\text{all } s} P(\{\underline{y}_k \wedge \underline{y}_{j>k} \wedge s\}|\grave{s}) \\ &= \sum_{\text{all } s} P(\underline{y}_{j>k}|\{\grave{s} \wedge s \wedge \underline{y}_k\}) \cdot P(\{\underline{y}_k \wedge s\}|\grave{s}) \\ &= \sum_{\text{all } s} P(\underline{y}_{j>k}|s) \cdot P(\{\underline{y}_k \wedge s\}|\grave{s}) \\ &= \sum_{\text{all } s} \beta_k(s) \cdot \gamma_k(\grave{s}, s). \end{aligned} \tag{4.28}$$

Thus, once the values $\gamma_k(\grave{s}, s)$ are known, a backward recursion can be used to calculate the values of $\beta_{k-1}(\grave{s})$ from the values of $\beta_k(s)$ using Equation 4.28. Figure 4.7 again shows an example of how the $\beta_k(0)$ value is calculated recursively using values of $\beta_{k+1}(s)$ and $\gamma_{k+1}(0, s)$ for our example $K = 3$ RSC code.

The initial conditions which should be used for $\beta_N(s)$ are not as clear as for $\alpha_0(s)$. From Equation 4.22 $\beta_k(s)$ is the probability that the future received sequence is $\underline{y}_{j>k}$, given that the present state is s. For the last stage in the trellis however, ie when $k = N$, there is no future received sequence, and hence it is not clear what the initial values $B_N(s)$ should be set to. Berrou et al. [21] used the initial values $\beta_N(0) = 1$ and $\beta_N(s) = 0$ for all $s \neq 0$ for a trellis terminated in the all zero state, and in [126] the initial conditions for an unterminated trellis were given by Robertson as $\beta_N(s) = \alpha_N(s)$ for all s. However, as pointed out by Breiling [127], if we consider $\beta_{N-1}(s)$ from Equation 4.22 we have

$$\begin{aligned} \beta_{N-1}(\grave{s}) &= P(\underline{y}_N|\grave{s}) \\ &= \sum_{\text{all } s} P(\{\underline{y}_N \wedge s\}|\grave{s}) \\ &= \sum_{\text{all } s} \gamma_N(\grave{s}, s) \end{aligned} \tag{4.29}$$

and from the backward recursion for $\beta_{k-1}(\grave{s})$ in Equation 4.28 we have

$$\beta_{N-1}(\grave{s}) = \sum_{\text{all } s} \beta_N(s)\gamma_N(\grave{s}, s). \tag{4.30}$$

For both Equation 4.29 and Equation 4.30 to be satisfied, we must have

$$\beta_N(s) = 1 \quad \text{for all } s. \tag{4.31}$$

If the trellis is terminated, so that only the final state $S_N = 0$ is possible, this can be taken into account in the backward recursive calculation of the $\beta_k(s)$ values through the $\gamma_k(\grave{s}, s)$ values. In a terminated trellis for the last $K - 1$ transitions, where K is the constraint length of the convolutional code, for each state s only one transition (the one which takes the trellis towards the all-zero state) will be possible. Hence $\gamma_k(\grave{s}, s) = P(\{\underline{y}_k \wedge S_k = s\}|S_{k-1} = \grave{s})$ will be zero for all values of s except one, and with the initial values $\beta_N(s) = 1$ the correct values of $\beta_{N-1}(s), \beta_{N-2}(s), \cdots \beta_{N-K+1}$ will be calculated through Equation 4.28. Thus theory indicates that we should use $\beta_N(s) = 1$ for all s and account for the trellis termination by setting the values of $\gamma_k(\grave{s}, s)$ to zero for all transitions that are not possible due to trellis termination. However the same result can be achieved using β values of $\beta_N(0) = 1$ and $\beta_N(s) = 0$ for $s\ neq 0$, as suggested by Berrou et al. [21], and calculating $\gamma_k(\grave{s}, s)$ values in the same way as for all other transitions (i.e. directly from the channel inputs - see next section). This second method is simpler to implement and hence it is more commonly used in practice.

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

We now consider how the $\gamma_k(\grave{s}, s)$ values in Equation 4.20 can be calculated from the received channel sequence. Using the definition of $\gamma_k(\grave{s}, s)$ from Equation 4.23 and the derivation from

Bayes' rule given in Equation 4.13 we have

$$\begin{aligned}\gamma_k(\grave{s}, s) &= P(\{\underline{y}_k \wedge s\}|\grave{s}) \\ &= P(\underline{y}_k|\{\grave{s} \wedge s\}) \cdot P(s|\grave{s}) \\ &= P(\underline{y}_k|\{\grave{s} \wedge s\}) \cdot P(u_k), \end{aligned} \tag{4.32}$$

where u_k is the input bit necessary to cause the transition from state $S_{k-1} = \grave{s}$ to state $S_k = s$, and $P(u_k)$ is the a-priori probability of this bit. From Equation 4.5 this can be written as

$$\begin{aligned}P(u_k) &= \left(\frac{\mathrm{e}^{-L(u_k)/2}}{1+\mathrm{e}^{-L(u_k)}}\right) \cdot \mathrm{e}^{(u_k L(u_k)/2)} \\ &= C^{(1)}_{L(u_k)} \cdot \mathrm{e}^{(u_k L(u_k)/2)}, \end{aligned} \tag{4.33}$$

where, as stated before,

$$C^{(1)}_{L(u_k)} = \left(\frac{\mathrm{e}^{-L(u_k)/2}}{1+\mathrm{e}^{-L(u_k)}}\right) \tag{4.34}$$

depends only on the LLR $L(u_k)$ and not on whether u_k is $+1$ or -1.

The first term in second and third lines of Equation 4.32, $P(\underline{y}_k|\{\grave{s} \wedge s\})$, is equivalent to $P(\underline{y}_k|\underline{x}_k)$, where $\underline{x}_k$ is the transmitted codeword associated with the transition from state $S_{k-1} = \grave{s}$ to state $S_k = s$. Again assuming the channel is memoryless we can write

$$P(\underline{y}_k|\{\grave{s} \wedge s\}) \equiv P(\underline{y}_k|\underline{x}_k) = \prod_{l=1}^{n} P(y_{kl}|x_{kl}), \tag{4.35}$$

where x_{kl} and y_{kl} are the individual bits within the transmitted and received codewords $\underline{y}_k$ and $\underline{x}_k$, and n is the number of these bits in each codeword $\underline{y}_k$ or $\underline{x}_k$. Assuming that the transmitted bits x_{kl} have been transmitted over a Gaussian channel using BPSK, so that the transmitted symbols are either $+1$ or -1, we have for $P(y_{kl}|x_{kl})$

$$P(y_{kl}|x_{kl}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{E_b}{2\sigma^2}(y_{kl} - a x_{kl})^2\right), \tag{4.36}$$

where E_b is the transmitted energy per bit, σ^2 is the noise variance and a is the fading amplitude (a=1 for non-fading AWGN channels). Upon substituting Equation 4.36 in Equation 4.35 we have

$$\begin{aligned}P(\underline{y}_k|\{\grave{s} \wedge s\}) &= \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{E_b}{2\sigma^2}(y_{kl} - a x_{kl})^2\right) \\ &= \frac{1}{(\sqrt{2\pi}\sigma)^n} \exp\left(-\frac{E_b}{2\sigma^2}\sum_{l=1}^{n}(y_{kl} - a x_{kl})^2\right) \\ &= \frac{1}{(\sqrt{2\pi}\sigma)^n} \exp\left(-\frac{E_b}{2\sigma^2}\sum_{l=1}^{n}(y_{kl}^2 + a^2 x_{kl}^2 - 2a x_{kl} y_{kl})\right) \\ &= C^{(2)}_{\underline{y}_k} \cdot C^{(3)}_{\underline{x}_k} \cdot \exp\left(\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^{n} y_{kl} x_{yl}\right), \end{aligned} \tag{4.37}$$

where

$$C_{\underline{y}_k}^{(2)} = \frac{1}{(\sqrt{2\pi}\sigma)^n} \cdot \exp\left(-\frac{E_b}{2\sigma^2}\sum_{l=1}^{n} y_{kl}^2\right) \tag{4.38}$$

depends only on the channel SNR and on the magnitude of the received sequence $\underline{y}_k$, while

$$\begin{aligned} C_{\underline{x}_k}^{(3)} &= \exp\left(-\frac{E_b}{2\sigma^2}a^2\sum_{l=1}^{n} x_{kl}^2\right) \\ &= \exp\left(-\frac{E_b}{2\sigma^2}a^2 n\right) \end{aligned} \tag{4.39}$$

depends only on the channel SNR and on the fading amplitude. Hence we can write for $\gamma_k(\grave{s}, s)$

$$\begin{aligned} \gamma_k(\grave{s}, s) &= P(u_k) \cdot P(\underline{y}_k | \{\grave{s} \wedge s\}) \\ &= C \cdot \mathrm{e}^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^{n} y_{kl} x_{yl}\right) \\ &= C \cdot \mathrm{e}^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2} \sum_{l=1}^{n} y_{kl} x_{yl}\right), \end{aligned} \tag{4.40}$$

where

$$C = C_{L(u_k)}^{(1)} \cdot C_{\underline{y}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)}. \tag{4.41}$$

The term C does not depend on the sign of the bit u_k or the transmitted codeword $\underline{x}_k$ and so is constant over the summations in the numerator and denominator in Equation 4.17 and cancels out.

From Equations 4.17 and 4.20 we can write for the conditional LLR of u_k, given the received sequence $\underline{y}_k$,

$$\begin{aligned} L(u_k|\underline{y}) &= \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} P(S_{k-1} = \grave{s} \wedge S_k = s \wedge \underline{y})}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} P(S_{k-1} = \grave{s} \wedge S_k = s \wedge \underline{y})}\right) \\ &= \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s}, s) \cdot \beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\grave{s}) \cdot \gamma_k(\grave{s}, s) \cdot \beta_k(s)}\right). \end{aligned} \tag{4.42}$$

It is this conditional LLR $L(u_k|\underline{y})$ that the MAP decoder delivers.

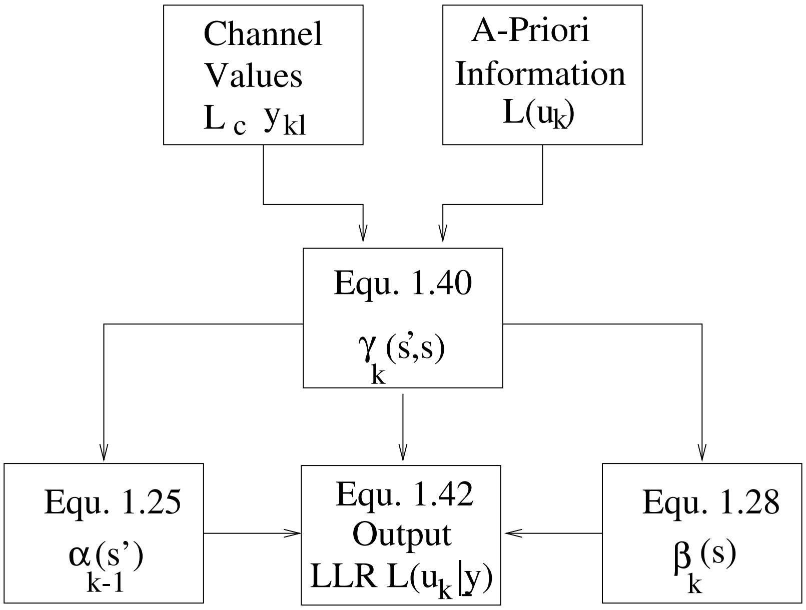


Figure 4.8: Summary of the Key Operations in the MAP Algorithm

4.3.3.5 Summary of the MAP Algorithm

¿From the description given above, we see that the MAP decoding of a received sequence $\underline{y}$ to give the a-posteriori LLR $L(u_k|\underline{y})$ can be carried out as follows. As the channel values y_{kl} are received, they and the a-priori LLRs $L(u_k)$ (which are provided in an iterative turbo decoder by the other component decoder – see Section 4.3.4) are used to calculate $\gamma_k(\grave{s}, s)$ according to Equation 4.40. The constant C can be omitted from the calculation of $\gamma_k(\grave{s}, s)$, as it will cancel out in the ratio in Equation 4.42. As the channel values y_{kl} are received, and the $\gamma_k(\grave{s}, s)$ values are calculated, the forward recursion from Equation 4.25 can be used to calculate $\alpha_k(\grave{s}, s)$. Once all the channel values have been received, and $\gamma_k(\grave{s}, s)$ has been calculated for all $k = 1, 2 \cdots N$, the backward recursion from Equation 4.28 can be used to calculate the $\beta_k(\grave{s}, s)$ values. Finally all the calculated values of $\alpha_k(\grave{s}, s)$, $\beta_k(\grave{s}, s)$ and $\gamma_k(\grave{s}, s)$ are used in Equation 4.42 to calculate the values of $L(u_k|\underline{y})$. These operations are summarised in the flowchart of Figure 4.8. Care must be taken to avoid numerical underflow problems in the recursive calculation of $\alpha_k(\grave{s}, s)$ and $\beta_k(\grave{s}, s)$, but such problems can be avoided by careful normalisation of these values. Such normalisation cancels out in the ratio in Equation 4.42 and so causes no change in the LLRs produced by the algorithm.

The MAP algorithm is, in the form described in this section, extremely complex due to the multiplications needed in Equations 4.25 and 4.28 for the recursive calculation of $\alpha_k(\grave{s}, s)$ and $\beta_k(\grave{s}, s)$, the multiplications and exponential operations required to calculate $\gamma_k(\grave{s}, s)$ using Equation 4.40, and the multiplication and natural logarithm operations required to calculate $L(u_k|\underline{y})$ using Equation 4.42. However much work has been done to reduce this complexity, and the Log-MAP algorithm [59], which will be described in Section 4.3.5, gives the same performance as the MAP algorithm but with a much lower complexity and without the numerical problems described above. We will first describe the principles behind the iterative decoding of turbo codes, and how the MAP algorithm described in this section can be used in such a scheme, before detailing the Log-MAP algorithm.

4.3.4 Iterative Turbo Decoding Principles

4.3.4.1 Turbo Decoding Mathematical Preliminaries

In this section we explain the concepts of extrinsic and intrinsic information as used by Berrou el al [21], and highlight how the MAP algorithm described in the previous section, and other soft-in soft-out decoders, can be used in the iterative decoding of turbo codes.

Consider first the expression for $\gamma_k(\grave{s}, s)$ in Equation 4.40, which is restated here for convenience

$$\gamma_k(\grave{s}, s) = C \cdot \mathrm{e}^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2}\sum_{l=1}^{n} y_{kl} x_{yl}\right). \tag{4.43}$$

As we are dealing with systematic codes one of the n transmitted bits will be the systematic bit u_k. If we assume that this systematic bit is the first of the n transmitted bits then we will have $x_{k1} = u_k$, and we can rewrite Equation 4.43 as

$$\begin{aligned}\gamma_k(\grave{s}, s) &= C \cdot \mathrm{e}^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2} y_{ks} u_k\right) \cdot \exp\left(\frac{L_c}{2}\sum_{l=2}^{n} y_{kl} x_{yl}\right) \\ &= C \cdot \mathrm{e}^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2} y_{ks} u_k\right) \cdot \chi_k(\grave{s}, s), \end{aligned} \tag{4.44}$$

where y_{ks} is the received version of the transmitted systematic bit $x_{k1} = u_k$ and

$$\chi_k(\grave{s}, s) = \exp\left(\frac{L_c}{2}\sum_{l=2}^{n} y_{kl} x_{yl}\right). \tag{4.45}$$

Using Equation 4.44 and remembering that in the numerator we have $u_k = +1$ for all terms in the summation, whereas in the denominator we have $u_k = -1$, we can rewrite Equation 4.42

as

$$
\begin{aligned}
L(u_k|\underline{y}) &= \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} \alpha_{k-1}(\grave{s})\cdot\gamma_k(\grave{s},s)\cdot\beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} \alpha_{k-1}(\grave{s})\gamma_k(\grave{s},s)\cdot\beta_k(s)}\right) \\
&= \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} \alpha_{k-1}(\grave{s})\cdot \mathrm{e}^{+L(u_k)/2}\cdot \mathrm{e}^{+L_c y_{ks}/2}\cdot\chi_k(\grave{s},s)\cdot\beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} \alpha_{k-1}(\grave{s})\cdot \mathrm{e}^{-L(u_k)/2}\cdot \mathrm{e}^{-L_c y_{ks}/2}\cdot\chi_k(\grave{s},s)\cdot\beta_k(s)}\right) \\
&= L(u_k) + L_c y_{ks} + \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} \alpha_{k-1}(\grave{s})\cdot\chi_k(\grave{s},s)\cdot\beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} \alpha_{k-1}(\grave{s})\cdot\chi_k(\grave{s},s)\cdot\beta_k(s)}\right) \\
&= L(u_k) + L_c y_{ks} + L_e(u_k),
\end{aligned} \tag{4.46}
$$

where

$$
L_e(u_k) = \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} \alpha_{k-1}(\grave{s})\cdot\chi_k(\grave{s},s)\cdot\beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} \alpha_{k-1}(\grave{s})\cdot\chi_k(\grave{s},s)\cdot\beta_k(s)}\right). \tag{4.47}
$$

Thus we can see that the a-posteriori LLR $L(u_k|\underline{y})$ calculated with the MAP algorithm can be thought of as comprising of three terms – $L(u_k)$, $L_c y_{ks}$ and $L_e(u_k)$. The a-priori LLR term $L(u_k)$ comes from $P(u_k)$ in the expression for the branch transition probability $\gamma_k(\grave{s},s)$ in Equation 4.32. This probability should come from an independent source and is called the a-priori probability of the k'th bit being $+1$ or -1. In most cases we will have no independent or a-priori knowledge of the likely value of the bit u_k, and so the a-priori LLR $L(u_k)$ will be zero, corresponding to an a-priori probability $P(u_k) = 0.5$. However, in the case of an iterative turbo decoder, each component decoder can provide the other decoder with an estimate of the a-priori LLR $L(u_k)$, as described later.

The second term $L_c y_{ks}$ in Equation 4.46 is the soft output of the channel for the systematic bit u_k, which was directly transmitted across the channel and received as y_{ks}. When the channel SNR is high, the channel reliability value L_c of Equation 4.11 will be high and this systematic bit will have a large influence on the a-posteriori LLR $L(u_k|\underline{y})$. Conversely, when the channel is poor and L_c is low, the soft output of the channel for the received systematic bit y_{ks} will have less impact on the a-posteriori LLR delivered by the MAP algorithm.

The final term in Equation 4.46, $L_e(u_k)$, is derived, using the constraints imposed by the code used, from the a-priori information sequence $L(u_n)$ and the received channel information sequence $\underline{y}$, *excluding* the received systematic bit y_{ks} and the a-priori information $L(u_k)$ for the bit u_k. Hence it is called the *extrinsic* LLR for the bit u_k. Equation 4.46 shows that the extrinsic information from a MAP decoder can be obtained by subtracting the a-priori information $L(u_k)$ and the received systematic channel input $L_c y_{ks}$ from the soft output $L(u_k|\underline{y})$ of the decoder. This is the reason for the subtraction paths shown in Figure 4.3. Equations similar to Equation 4.46 can be derived for the other component decoders which can be used in iterative turbo decoding.

Notice that the expression for the branch transition probabilities $\gamma_k(\grave{s}, s)$ in Equation 4.40 uses the a-priori information $L(u_k)$ and all n bits, including the systematic bit y_{ks}, of the received codeword $\underline{y}_k$. These branch transition probabilities are used in the recursive calculations of $\alpha_k(s)$ and $\beta_k(s)$ in Equations 4.25 and 4.28 and so, as these terms appear in Equation 4.47 for $L_e(u_k)$, it might seem that the received systematic bit y_{ks} and the a-priori information $L(u_k)$ for the bit u_k appear indirectly in the extrinsic output $L_e(u_k)$. However careful examination of Equation 4.47 shows that for the bit u_k we use the values of $\alpha_{k-1}(\grave{s})$ and $\beta_k(s)$. From Equations 4.25 and 4.28 for the recursive calculation of the these values we see that the branch transition probabilities $\gamma_n(\grave{s}, s)$ for $1 \leq n \leq k-1$ and $k+1 \leq n \leq N$ will be used to calculate the $\alpha_{k-1}(\grave{s})$ and $\beta_k(s)$ values. Notice however that the branch transition probability $\gamma_k(\grave{s}, s)$, for the transition associated with the bit u_k, is not used. Hence $L_e(u_k)$ uses the values of the branch transition probabilities $\gamma_n(\grave{s}, s)$ for all the branches *except* the k'th branch. Therefore, although it does depend on all the other a-priori information terms $L(u_n)$ and received systematic bits, the term $L_e(u_k)$ really is independent of the a-priori information $L(u_k)$ and the received systematic bit y_{ks}, and so can justifiably be called the extrinsic LLR for the bit u_k.

We summarise below what is meant by the terms a-priori, a-posteriori and extrinsic information which we use throughout this treatise.

a-priori The a-priori information about a bit is information known before decoding starts, from a source other than the received sequence or the code constraints. It is also sometimes referred to as intrinsic information to contrast with the extrinsic information described next.

extrinsic The extrinsic information about a bit u_k is the information provided by a decoder based on the received sequence and on a-priori information *excluding* the received systematic bit y_{ks} and the a-priori information $L(u_k)$ for the bit u_k. Typically the component decoder provides this information using the constraints imposed on the transmitted sequence by the code used. It processes the received bits and a-priori information surrounding the systematic bit u_k, and uses this information and the code constraints to provide information about the value of the bit u_k.

a-posteriori The a-posteriori information about a bit is the information that the decoder gives taking into account *all* available sources of information about u_k. It is the a-posteriori LLR, ie $L(u_k|\underline{y})$, that the MAP algorithm gives as its output.

4.3.4.2 Iterative Turbo Decoding

We now describe how the iterative decoding of turbo codes is carried out. Figure 4.3 from Section 4.3.1, showing the structure of an iterative turbo decoder, is repeated for convenience here as Figure 4.9. Figure 4.9 also shows the symbols we have used for the various inputs to and outputs from the component decoders.

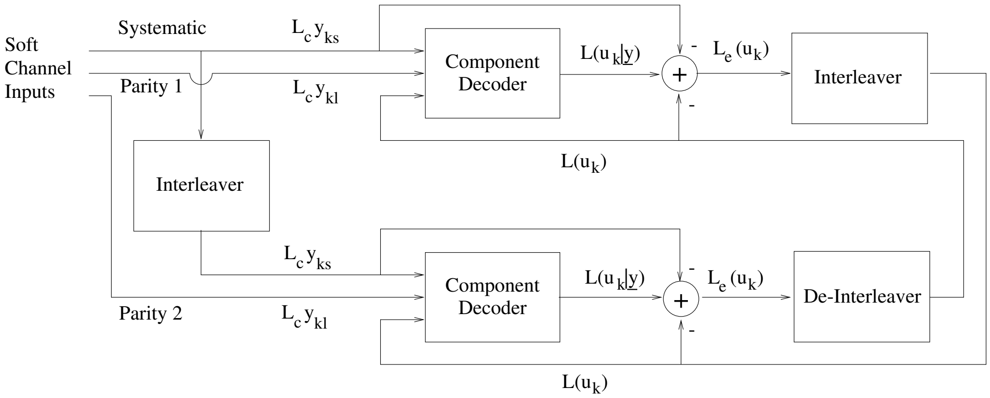


Figure 4.9: Turbo Decoder Schematic

Consider initially the first component decoder in the first iteration. This decoder receives the channel sequence $L_c\underline{y}^{(1)}$ containing the received versions of the transmitted systematic bits, $L_c y_{ks}$, and the parity bits, $L_c y_{kl}$, from the first encoder. Usually, to obtain a half-rate code, half of these parity bits will have been punctured at the transmitter, and so the turbo decoder must insert zeros in the soft channel output $L_c y_{kl}$ for these punctured bits. The first component decoder can then process the soft channel inputs and produce its estimate $L_{11}(u_k|\underline{y})$ of the conditional LLRs of the data bits u_k, $k = 1, 2 \cdots N$. In this notation the subscript 11 in $L_{11}(u_k|\underline{y})$ indicates that this is the a-posteriori LLR in the first iteration from the first component decoder. Note that in this first iteration the first component decoder will have no a-priori information about the bits, and hence $L(u_k)$ in Equation 4.40 giving $\gamma_k(\grave{s}, s)$ will be zero, corresponding to an a-priori probability of 0.5.

Next the second component decoder comes into operation. It receives the channel sequence $L_c\underline{y}^{(2)}$ containing the *interleaved* version of the received systematic bits, and the parity bits from the second encoder. Again, the turbo decoder will have to insert zeros into this sequence, if the parity bits generated by the encoder are punctured before transmission. However now, in addition to the received channel sequence $L_c\underline{y}^{(2)}$, the decoder can use the conditional LLR $L_{11}(u_k|\underline{y})$ provided by the first component decoder to generate a-priori LLRs $L(u_k)$ to be used by the second component decoder. Ideally these a-priori LLRs $L(u_k)$ would be completely independent from all the other information used by the second component decoder. As can be seen in Figure 4.9 in iterative turbo decoders the extrinsic information $L_e(u_k)$ from the other component decoder is used as the a-priori LLRs, after being interleaved to arrange the decoded data bits $\underline{u}$ in the same order as they were encoded by the second encoder. Again, according to Equation 4.46, the reason for the subtraction paths shown in Figure 4.9 is that the a-posteriori LLRs from one decoder have the systematic soft channel inputs $L_c y_{ks}$ and the a-priori LLRs $L(u_k)$ (if any were available) subtracted to yield the extrinsic LLRs $L_e(u_k)$ which are then used as a-priori LLRs for the other component decoder. The second component decoder thus uses the received channel sequence $L_c\underline{y}^{(2)}$ and the a-priori LLRs $L(u_k)$ (derived by interleaving the extrinsic LLRs $L_e(u_k)$ of the first component decoder) to produce its a-posteriori LLRs $L_{12}(u_k|\underline{y})$. This is then the end of the first iteration.

For the second iteration the first component encoder again processes its received channel sequence $L_c\underline{y}^{(1)}$, but now it also has a-priori LLRs $L(u_k)$ provided by the extrinsic portion $L_e(u_k)$ of the a-posteriori LLRs $L_{12}(u_k|\underline{y})$ calculated by the second component encoder, and hence it can produce an improved a-posteriori LLR $L_{21}(u_k|\underline{y})$. The second iteration then continues with the second component decoder using the improved a-posteriori LLRs $L_{21}(u_k|\underline{y})$ from the first encoder to derive, through Equation 4.46, improved a-priori LLRs $L(u_k)$ which it uses in conjunction with its received channel sequence $L_c\underline{y}^{(2)}$ to calculate $L_{22}(u_k|\underline{y})$.

This iterative process continues, and with each iteration on average the BER of the decoded bits will fall. However, as will be seen in Figure 4.20, the improvement in performance for each additional iteration carried out falls as the number of iterations increases. Hence for complexity reasons usually only about eight iterations are carried out, as no significant improvement in performance is obtained with a higher number of iterations. This is the arrangement we have used in most of our simulations, ie the decoder carries out a fixed number of iterations. However it is possible to use a variable number of iterations up to a maximum, with some termination criterion used to decide when it is deemed that further iterations will produce marginal gain. This allows the average number of iterations, and so the average complexity of the decoder, to be dramatically reduced [68] with only a small degradation in performance. Suitable termination criteria have been found to be the so-called cross-entropy of the outputs from the two component decoders [68], and the variance of the a-posteriori LLRs $L(u_k|\underline{y})$ of a component decoder [126].

Figure 4.10 shows how the a-posteriori LLRs $L(u_k|\underline{y})$ output from the component decoders in an iterative decoder vary with the number of iterations used. The output from the second component decoder is shown after 1,2,4 and 8 iterations. The input sequence of the encoder consisted entirely of logical 0's, and consequently the negative a-posteriori LLR $L(u_k|\underline{y})$ values correspond to a correct hard decision, while the positive values to an incorrect hard decision. The input sequence was coded using a turbo encoder with two constraint length 3 recursive convolutional codes, and a block interleaver with 31 rows and 31 columns. This turbo encoder is used in the majority of our investigations and its performance is characterised in Section 4.4. The encoded bits were transmitted over an AWGN channel at a channel SNR of -1 dB, and then decoded using an iterative turbo decoder using the MAP algorithm. It can be seen that as the number of iterations used increases, the number of positive a-posteriori LLR $L(u_k|\underline{y})$ values, and hence the BER, decreases until after eight iterations there are no incorrectly decoded values. Furthermore, as the number of iterations increases, the decoders become more certain about the value of the bits and hence the magnitudes of the LLRs gradually become larger. The erroneous decisions in the figure appear in bursts, since deviating from the error-free path trellis path typically inflicts several bit errors.

When the series of iterations halts, after either a fixed number of iterations or when a termination criterion is satisfied, the output from the turbo decoder is given by the de-interleaved a-posteriori LLRs of the second component decoder, $L_{i2}(u_k|\underline{y})$, where i is the number of iterations used. The sign of these a-posteriori LLRs gives the hard decision output, ie whether the decoder believes that the transmitted data bit u_k was $+1$ or -1, and in some applications the magnitude of these LLRs, which gives the confidence the decoder has in its decision, may also be useful.

Ideally, for the iterative decoding of turbo codes, the a-priori information used by a component decoder should be completely independent from the channel outputs used by that

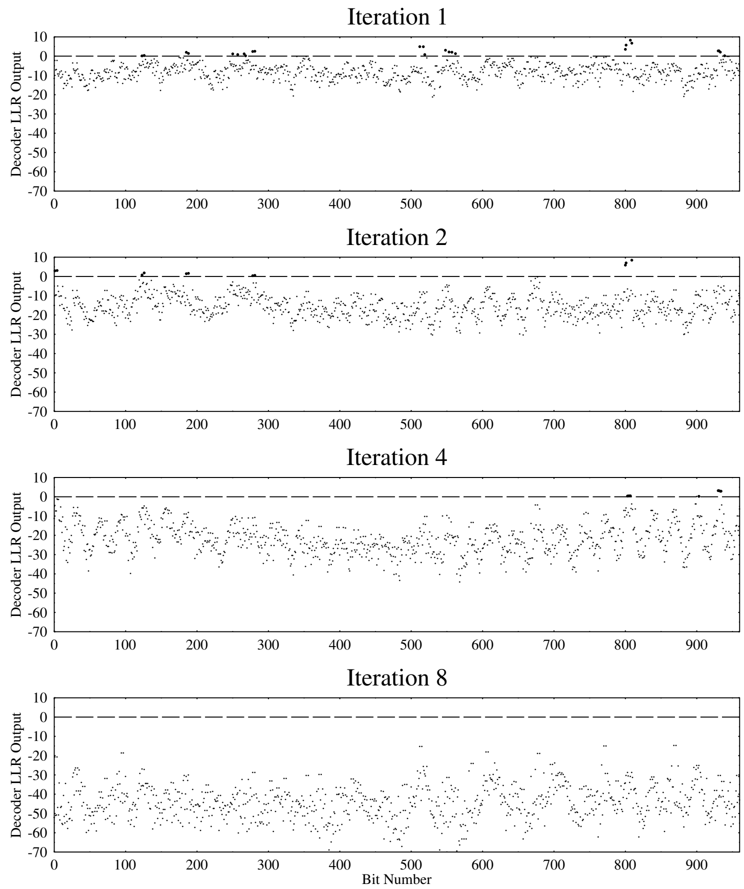


Figure 4.10: Soft Outputs From the MAP Decoder in an Iterative Turbo Decoder for a Transmitted Stream of all -1

decoder. However in turbo decoders the extrinsic LLR $L_e(u_k)$ for the bit u_k, as explained above, uses all the available received parity bits and all the received systematic bits except the received value y_{ks} of the bit u_k. However the same received systematic bits are also used by the other component decoder, which uses the interleaved or de-interleaved version of $L_e(u_k)$ as its a-priori LLRs. Hence the a-priori LLRs $L(u_k)$ are not truly independent from the channel outputs $\underline{y}$ used by the component decoders. However, due to the fact that the component convolutional codes have a short memory, usually of only 4 bits or less, the extrinsic LLR $L_e(u_k)$ is only significantly affected by the received systematic bits relatively close to the bit u_k. When this extrinsic LLR $L_e(u_k)$ is used as the a-priori LLR $L(u_k)$ by the other component decoder, because of the interleaving used, the bit u_k and its neighbours will probably have been well separated. Hence the dependence of the a-priori LLRs $L(u_k)$ on the received systematic channel values $L_c y_{ks}$ which are also used by the other component decoder will have relatively little effect, and the iterative decoding provides good results.

Another justification for using the iterative arrangement described above is how well it has been found to work. In the limited experiments that have been carried out with optimal decoding of turbo codes [124,125,128] it has been found that optimal decoding performs only a fraction of a decibel (around 0.35–0.5 dB) better than iterative decoding with the MAP algorithm. Furthermore various turbo coding schemes have been found [70,128] that approach the Shannonian limit, which gives the best performance theoretically available, to a similar fraction of a decibel. Therefore it seems that, for a variety of codes, the iterative decoding of turbo codes gives an almost optimal performance. Hence it is this iterative decoding structure, which is almost exclusively used with turbo codes, which we have used throughout our simulations.

Having described how the MAP algorithm can be used in the iterative decoding of turbo codes, we now proceed to describe other soft-in soft-out decoders, which are less complex and can be used instead of the MAP algorithm. We first describe two related algorithms, the Max-Log-MAP [57,58] and the Log-MAP [59], which are derived from the MAP algorithm, and then another, referred to as the Soft Output Viterbi Algorithm (SOVA) [60,122], derived from the Viterbi algorithm.

4.3.5 Modifications of the MAP algorithm

4.3.5.1 Introduction

The MAP algorithm as described in Section 4.3.3 is much more complex than the Viterbi algorithm and with hard decision outputs performs almost identically to it. Therefore for almost 20 years it was largely ignored. However, its application in turbo codes renewed interest in the algorithm, and it was realised that its complexity can be dramatically reduced without affecting its performance. Initially the Max-Log-MAP algorithm was proposed by Koch and Baier [57] and Erfanian et al [58]. This technique simplified the MAP algorithm by transferring the recursions into the log domain and invoking an approximation to dramatically reduce the complexity. Because of this approximation its performance is sub-optimal compared to that of the MAP algorithm. However, Robertson et al [59] in 1995 proposed the Log-MAP algorithm, which corrected the approximation used in the Max-Log-MAP algorithm and hence gave a performance identical to that of the MAP algorithm, but at a fraction of its complexity. These two algorithms are described in this section.

4.3.5.2 Mathematical Description of the Max-Log-MAP Algorithm

The MAP algorithm calculates the a-posteriori LLRs $L(u_k|\underline{y})$ using Equation 4.42. To do this it requires the following values:

1) The $\alpha_{k-1}(\grave{s})$ values, which are calculated in a forward recursive manner using Equation 4.25,

2) the $\beta_k(s)$ values, which are calculated in a backward recursion using Equation 4.28, and

3) the branch transition probabilities $\gamma_k(\grave{s}, s)$, which are calculated using Equation 4.40.

The Max-Log-MAP algorithm simplifies this by transferring these equations into the log arithmetic domain and then using the approximation

$$\ln\left(\sum_i \mathrm{e}^{x_i}\right) \approx \max_i(x_i), \tag{4.48}$$

where $\max_i(x_i)$ means the maximum value of x_i. Then, with $A_k(s)$, $B_k(s)$ and $\Gamma_k(\grave{s}, s)$ defined as follows

$$A_k(s) \triangleq \ln\left(\alpha_k(s)\right), \tag{4.49}$$

$$B_k(s) \triangleq \ln\left(\beta_k(s)\right), \tag{4.50}$$

and

$$\Gamma_k(\grave{s}, s) \triangleq \ln\left(\gamma_k(\grave{s}, s)\right), \tag{4.51}$$

we can rewrite Equation 4.25 as

$$\begin{aligned} A_k(s) &\triangleq \ln\left(\alpha_k(s)\right) \\ &= \ln\left(\sum_{\text{all }\grave{s}} \alpha_{k-1}(\grave{s})\gamma_k(\grave{s}, s)\right) \\ &= \ln\left(\sum_{\text{all }\grave{s}} \exp\left[A_{k-1}(\grave{s}) + \Gamma_k(\grave{s}, s)\right]\right) \\ &\approx \max_{\grave{s}}\left(A_{k-1}(\grave{s}) + \Gamma_k(\grave{s}, s)\right). \end{aligned} \tag{4.52}$$

Equation 4.52 implies that for each path in Figure 4.6 from the previous stage in the trellis to the state $S_k = s$ at the present stage, the algorithm adds a branch metric term $\Gamma_k(\grave{s}, s)$ to the previous value $A_{k-1}(\grave{s})$ to find a new value $\tilde{A}_k(s)$ for that path. The new value of $A_k(s)$ according to Equation 4.52 is then the maximum of the $\tilde{A}_k(s)$ values of the various paths reaching the state $S_k = s$. This can be thought of as selecting one path as the "survivor" and discarding any other paths reaching the state.

The value of $A_k(s)$ should give the natural logarithm of the probability that the trellis is in state $S_k = s$ at stage k, given that the received channel sequence up to this point has been $\underline{y}_{j<k}$. However, because of the approximation of Equation 4.48 used to derive Equation 4.52, only the maximum likelihood path through the state $S_k = s$ is considered when calculating this probability. Thus the value of A_k in the Max-Log-MAP algorithm actually gives the probability of the most likely path through the trellis to the state $S_k = s$, rather than the probability of *any* path through the trellis to state $S_k = s$. This approximation is one of the reasons for the sub-optimal performance of the Max-Log-MAP algorithm compared to the MAP algorithm.

We see from Equation 4.52 that in the Max-Log-MAP algorithm the forward recursion used to calculate $A_k(s)$ is exactly the same as the forward recursion in the Viterbi algorithm – for each pair of merging paths the survivor is found using two additions and one comparison.

Notice that for binary trellises the summation, and maximisation, over all previous states $S_{k-1} = \grave{s}$ in Equation 4.52 will in fact be over only two states, because there will be only two previous states $S_{k-1} = \grave{s}$ with paths to the present state $S_k = s$. For all other values of $\grave{s}$ we will have $\gamma_k(\grave{s}, s) = 0$.

Similarly to Equation 4.52 for the forward recursion used to calculate the $A_k(s)$, we can rewrite Equation 4.28 as

$$\begin{aligned} B_{k-1}(\grave{s}) &\triangleq \ln\left(\beta_{k-1}(\grave{s})\right) \\ &= \ln\left(\sum_{\text{all } s} \beta_k(s)\gamma_k(\grave{s}, s)\right) \\ &= \ln\left(\sum_{\text{all } s} \exp\left[B_k(s) + \Gamma_k(\grave{s}, s)\right]\right) \\ &\approx \max_s \left(B_k(s) + \Gamma_k(\grave{s}, s)\right), \end{aligned} \tag{4.53}$$

and obtain the backward recursion used to calculate the $B_{k-1}(\grave{s})$ values. Again this is equivalent to the recursion used in the Viterbi algorithm – the value of $B_{k-1}(\grave{s})$ is found by, for every state $S_k = s$ having a path from $S_{k-1} = \grave{s}$ (two in a binary trellis), adding a branch metric $\Gamma_k(\grave{s}, s)$ to the value of $B_k(s)$ and selecting which path gives the highest $B_{k-1}(\grave{s})$ value.

Using Equation 4.40, the branch metrics $\Gamma_k(\grave{s}, s)$ in the above recursive equations for $A_k(s)$ and $B_{k-1}(\grave{s})$ can be written as

$$\begin{aligned} \Gamma_k(\grave{s}, s) &\triangleq \ln\left(\gamma_k(\grave{s}, s)\right) \\ &= \ln\left(C \cdot \mathrm{e}^{(u_k L(u_k)/2)} \exp\left[\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^{n} y_{kl} x_{kl}\right]\right) \\ &= \hat{C} + \frac{1}{2} u_k L(u_k) + \frac{L_c}{2} \sum_{l=1}^{n} y_{kl} x_{kl}, \end{aligned} \tag{4.54}$$

where $\hat{C} = \ln C$ does not depend on u_k or on the transmitted codeword $\underline{x}_k$ and so can be considered a constant and omitted. Hence the branch metric is equivalent to that used in the Viterbi algorithm, with the addition of the a-priori LLR term $u_k L(u_k)$. Furthermore the correlation term $\sum_{l=1}^{n} y_{kl} x_{kl}$ is weighted by the channel reliability value L_c of Equation 4.11.

Finally, from Equation 4.42, we can write for the a-posteriori LLRs $L(u_k|\underline{y})$ which the Max-Log-MAP algorithm calculates

$$\begin{aligned}
L(u_k|\underline{y}) &= \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\u_k=+1}} \alpha_{k-1}(\grave{s})\cdot\gamma_k(\grave{s},s)\cdot\beta_k(s)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\u_k=-1}} \alpha_{k-1}(\grave{s})\cdot\gamma_k(\grave{s},s)\cdot\beta_k(s)}\right) \\
&= \ln\left(\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\u_k=+1}} \exp\left(A_{k-1}(\grave{s})+\Gamma_k(\grave{s},s)+B_k(s)\right)}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\u_k=-1}} \exp\left(A_{k-1}(\grave{s})+\Gamma_k(\grave{s},s)+B_k(s)\right)}\right) \\
&\approx \max_{\substack{(\grave{s},s)\Rightarrow\\u_k=+1}} \left(A_{k-1}(\grave{s})+\Gamma_k(\grave{s},s)+B_k(s)\right) \\
&- \max_{\substack{(\grave{s},s)\Rightarrow\\u_k=-1}} \left(A_{k-1}(\grave{s})+\Gamma_k(\grave{s},s)+B_k(s)\right). \qquad (4.55)
\end{aligned}$$

This means that in the Max-Log-MAP algorithm for each bit u_k the a-posteriori LLR $L(u_k|\underline{y})$ is calculated by considering every transition from the trellis stage S_{k-1} to the stage S_k. These transitions are grouped into those that might have occured if $u_k = +1$, and those that might have occured if $u_k = -1$. For both of these groups the transition giving the maximum value of $A_{k-1}(\grave{s}) + \Gamma_(\grave{s}, s) + B_k(s)$ is found, and the a-posteriori LLR is calculated based on only these two "best" transitions. For a binary trellis there will be $2\cdot 2^{K-1}$ transitions at each stage of the trellis, where K is the constraint length of the convolutional code. Therefore there will be 2^{K-1} transitions to consider in each of the maximisations in Equation 4.55.

The Max-Log-MAP algorithm can be summarised as follows. Forward and backward recursions, both similar to the forward recursion used in the Viterbi algorithm, are used to calculate $A_k(s)$ using Equation 4.52 and $B_k(s)$ using Equation 4.53. The branch metric $\Gamma_k(\grave{s}, s)$ used is given by Equation 4.54, where the constant term $\hat{C}$ can be omitted. Once both the forward and backward recursions have been carried out, the a-posteriori LLRs can be calculated using Equation 4.55. Thus the complexity of the Max-Log-MAP algorithm is not hugely higher than that of the Viterbi algorithm – instead of one recursion two are carried out, the branch metric of Equation 4.54 has the additional a-priori term $u_k L(u_k)$ term added to it, and for each bit Equation 4.55 must be used to give the a-posteriori LLRs. This calculation of $L(u_k|\underline{y})$ from the $A_{k-1}(\grave{s})$, $B_k(s)$, and $\Gamma_k(\grave{s}, s)$ values requires for every bit 2 additions for each of the $2 \cdot 2^{K-1}$ transitions at each stage of the trellis, two maximisations and one subtraction. Viterbi states [129] that the complexity of the Log-MAP-Max algorithm is no greater than three times that of a Viterbi decoder. Unfortunately the storage requirements are much greater due to the need to store both the forward and backward recursively calculated metrics $A_k(s)$ and $B_k(s)$ before the $L(u_k|\underline{y})$ values can be calculated. However, Viterbi also states [129, 130] that it can be shown that by increasing the computational load slightly the associated memory requirements can be dramatically reduced.

4.3.5.3 Correcting the Approximation – the Log-MAP Algorithm

The Max-Log-MAP algorithm gives a slight degradation in performance compared to the MAP algorithm due to the approximation of Equation 4.48. When used for the iterative decoding of turbo codes, Robertson et al [59] found this degradation to result in a drop in performance of about 0.35 dB. However, the approximation of Equation 4.48 can be made exact by using the Jacobian logarithm:

$$\begin{aligned}\ln\left(\mathrm{e}^{x_1}+\mathrm{e}^{x_2}\right) &= \max(x_1,x_2)+\ln\left(1+\mathrm{e}^{-|x_1-x_2|}\right)\\ &= \max(x_1,x_2)+f_c\left(|x_1-x_2|\right)\\ &= g(x_1,x_2),\end{aligned}\tag{4.56}$$

where $f_c(x)$ can be thought of as a correction term. This is then the basis of the Log-MAP algorithm proposed by Robertson, Villebrun and Hoeher [59]. Similarly to the Max-Log-MAP algorithm, values for $A_k(s) \triangleq \ln(\alpha_k(s))$ and $B_k(s) \triangleq \ln(\beta_k(s))$ are calculated using a forward and a backward recursion. However, the maximisation in Equations 4.52 and 4.53 is complemented by the correction term in Equation 4.56. This means that the exact rather than approximate values of $A_k(s)$ and $B_k(s)$ are calculated. In binary trellises, as explained earlier, the maximisation will be over only two terms. Therefore we can correct the approximations in Equations 4.52 and 4.53 by merely adding the term $f_c(\delta)$, where δ is the magnitude of the difference between the metrics of the two merging paths. Similarly, the approximation in Equation 4.55 giving the a-posteriori LLRs $L(u_k|\underline{y})$, can be eliminated using the Jacobian logarithm. However, as explained earlier, there will be 2^{K-1} transitions to consider in each of the maximisations of Equation 4.55. Thus we must generalise Equation 4.48 in order to cope with more than two x_i terms. This is done by nesting the $g(x_1,x_2)$ operations as follows

$$\ln\left(\sum_{i=1}^{I} e^{x_i}\right) = g(x_I, g(x_{I-1}, \cdots g(x_3, g(x_2,x_1)))\cdots).\tag{4.57}$$

The correction term $f_c(\delta)$ need not be computed for every value of δ, but instead can be stored in a look-up table. Robertson et al [59] found that such a look-up table need contain only eight values for δ, ranging between 0 and 5. This means that the Log-MAP algorithm is only slightly more complex than the Max-Log-MAP algorithm, but it gives exactly the same performance as the MAP algorithm. Therefore it is a very attractive algorithm to use in the component decoders of an iterative turbo decoder.

Having described two techniques based on the MAP algorithm but with reduced complexity, we now describe an alternative soft-in soft-out decoder based on the Viterbi algorithm.

4.3.6 The Soft-Output Viterbi Algorithm

4.3.6.1 Mathematical Description of the SOVA Algorithm

In this section we describe a variation of the Viterbi algorithm, referred to as the Soft Output Viterbi Algorithm (SOVA) [60, 122]. This algorithm has two modifications over the classical Viterbi algorithm which allow it to be used as a component decoder for turbo codes. Firstly the path metrics used are modified to take account of a-priori information when selecting the maximum likelihood path through the trellis. Secondly the algorithm is modified so that it provides a soft output in the form of the a-posteriori LLR $L(u_k|\underline{y})$ for each decoded bit.

The first modification is easily accomplished. Consider the state sequence $\underline{s}_k^s$ which gives the states along the surviving path at state $S_k = s$ at stage k in the trellis. The probability that this is the correct path through the trellis is given by

$$p(\underline{s}_k^s|\underline{y}_{j\leq k}) = \frac{p(\underline{s}_k^s \wedge \underline{y}_{j\leq k})}{p(\underline{y}_{j\leq k})}. \tag{4.58}$$

As the probability of the received sequence $\underline{y}_{j\leq k}$ for transitions up to and including the k'th transition is constant for all paths $\underline{s}_k$ through the trellis to stage k, the probability that the path $\underline{s}_k^s$ is the correct one is proportional to $p(\underline{s}_k^s \wedge \underline{y}_{j\leq k})$. Therefore our metric should be defined so that maximising the metric will maximise $p(\underline{s}_k^s \wedge \underline{y}_{j\leq k})$. The metric should also be easily computable in a recursive manner as we go from the $(k-1)$'th stage in the trellis to the k'th stage. If the path $\underline{s}_k^s$ at the k'th stage has the path $\underline{s}_{k-1}^{\grave{s}}$ for its first $k-1$ transitions then, assuming a memoryless channel, we will have

$$p(\underline{s}_k^s \wedge \underline{y}_{j\leq k}) = p(\underline{s}_{k-1}^{\grave{s}} \wedge \underline{y}_{j\leq k-1}) \cdot p(S_k = s \wedge \underline{y}_k|S_{k-1} = \grave{s}). \tag{4.59}$$

A suitable metric for the path $\underline{s}_k^s$ is therefore $M(\underline{s}_k^s)$, where

$$\begin{aligned} M(\underline{s}_k^s) &\triangleq \ln\Big(p(\underline{s}_k^s \wedge \underline{y}_{j\leq k})\Big) \\ &= M(\underline{s}_{k-1}^{\grave{s}}) + ln\Big(p(S_k = s \wedge \underline{y}_k|S_{k-1} = \grave{s})\Big). \end{aligned} \tag{4.60}$$

Using Equation 4.23 we then have:

$$M(\underline{s}_k^s) = M(\underline{s}_{k-1}^{\grave{s}}) + ln\left(\gamma_k(\grave{s}, s)\right), \tag{4.61}$$

where $\gamma_k(\grave{s}, s)$ is the branch transition probability for the path from $S_{k-1} = \grave{s}$ to $S_k = s$. From Equation 4.54 we can write

$$\ln\left(\gamma_k(\grave{s}, s)\right) \triangleq \Gamma_k(\grave{s}, s) = \hat{C} + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl}, \tag{4.62}$$

and as the term $\hat{C}$ is constant, it can be omitted and we can rewrite Equation 4.61 as

$$M(\underline{s}_k^s) = M(\underline{s}_{k-1}^{\grave{s}}) + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl}. \tag{4.63}$$

Hence our metric in the SOVA algorithm is updated as in the Viterbi algorithm, with the additional $u_k L(u_k)$ term included so that the a-priori information available is taken into account. Notice that this is equivalent to the forward recursion in Equation 4.52 used to calculate $A_k(s)$ in the Max-Log-MAP algorithm.

The possibility of modifying the metric used in the Viterbi algorithm to include a-priori information was mentioned by Forney [18] in his 1973 paper, although he proposed no application for such a modification. However the requirement to use a-priori information in the soft-in soft-out component decoders of turbo decoders has provided an obvious application.

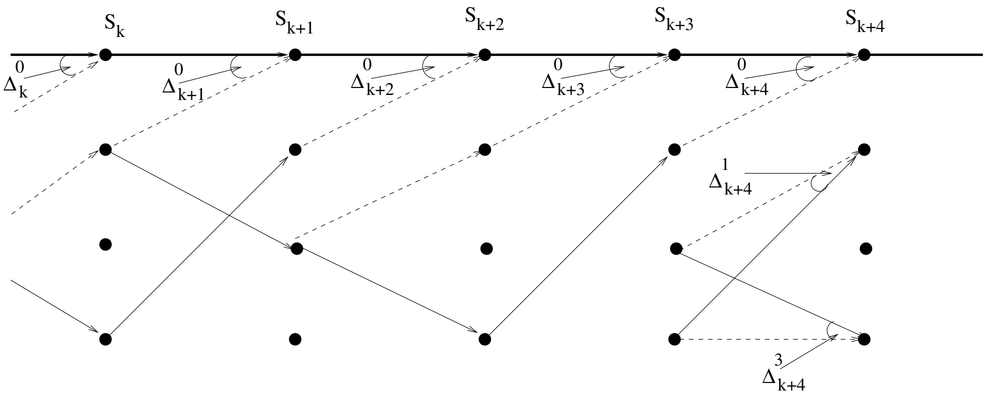


Figure 4.11: Simplified Section of the Trellis for our $K = 3$ RSC Code with SOVA Decoding

Let us now discuss the second modification of the algorithm required, ie to give soft outputs. In a binary trellis there will be two paths reaching state $S_k = s$ at stage k in the trellis. The modified Viterbi algorithm, which takes account of the a-priori information $u_k L(u_k)$, calculates the metric from Equation 4.63 for both merging paths, and discards the path with the lower metric. If the two paths $\underline{s}_k^s$ and $\underline{\hat{s}}_k^s$ reaching state $S_k = s$ have metrics $M(\underline{s}_k^s)$ and $M(\underline{\hat{s}}_k^s)$, and the path $\underline{s}_k^s$ is selected as the survivor because its metric is higher, then we can define the metric difference Δ_k^s as

$$\Delta_k^s = M(\underline{s}_k^s) - M(\underline{\hat{s}}_k^s) \geq 0. \tag{4.64}$$

The probability that we have made the correct decision when we selected path $\underline{s}_k^s$ as the survivor and discarded path $\underline{\hat{s}}_k^s$, is then

$$P(\text{correct decision at } S_k = s) = \frac{P(\underline{s}_k^s)}{P(\underline{s}_k^s) + P(\underline{\hat{s}}_k^s)}. \tag{4.65}$$

Upon taking into account our metric definition in Equation 4.60 we have

$$\begin{aligned} P(\text{correct decision at } S_k = s) &= \frac{\mathrm{e}^{M(\underline{s}_k^s)}}{\mathrm{e}^{M(\underline{s}_k^s)} + \mathrm{e}^{M(\underline{\hat{s}}_k^s)}} \\ &= \frac{\mathrm{e}^{\Delta_k^s}}{1 + \mathrm{e}^{\Delta_k^s}}, \end{aligned} \tag{4.66}$$

and the LLR that this is the correct decision is given by

$$\begin{aligned} L(\text{correct decision at } S_k = s) &= \ln\left(\frac{P(\text{correct decision at } S_k = s)}{1 - P(\text{correct decision at } S_k = s)}\right) \\ &= \Delta_k^s. \end{aligned} \tag{4.67}$$

Figure 4.11 shows a simplified section of the trellis for our $K = 3$ RSC code, with the metric differences Δ_k^s marked at various points in the trellis.

When we reach the end of the trellis and have identified the Maximum Likelihood (ML) path through the trellis, we need to find the LLRs giving the reliability of the bit decisions along the ML path. Observations of the Viterbi algorithm have shown that all the surviving paths at a stage l in the trellis will normally have come from the same path at some point before l in the trellis. This point is taken to be at most δ transitions before l, where usually δ is set to be five times the constraint length of the convolutional code. Therefore the value of the bit u_k associated with the transition from state $S_{k-1} = \grave{s}$ to state $S_k = s$ on the ML path may have been different if, instead of the ML path, the Viterbi algorithm had selected one of the paths which merged with the ML path up to δ transitions later, ie up to the trellis stage $k + \sigma$. By the arguments above if the algorithm had selected any of the paths which merged with the ML path after this point the value of u_k would not be affected, because such paths will have diverged from the ML path after the transition from $S_{k-1} = \grave{s}$ to $S_k = s$. Thus, when calculating the LLR of the bit u_k, the soft output Viterbi algorithm must take account of the probability that the paths merging with the ML path from stage k to stage $k + \delta$ in the trellis were incorrectly discarded. This is done by considering the values of the metric difference $\Delta_i^{s_i}$ for all states s_i along the ML path from trellis stage $i = k$ to $i = k + \delta$. It is shown by Hagenauer in [61] that this LLR can be approximated by

$$L(u_k|\underline{y}) \approx u_k \min_{\substack{i=k\cdots k+\delta \\ u_k \neq u_k^i}} \Delta_i^{s_i}, \tag{4.68}$$

where u_k is the value of the bit given by the ML path, and u_k^i is the value of this bit for the path which merged with the ML path and was discarded at trellis stage i. Thus the minimisation in Equation 4.68 is carried out only for those paths merging with the ML path which would have given a different value for the bit u_k if they had been selected as the survivor. The paths which merge with the ML path, but would have given the same value for u_k as the ML path, obviously do not affect the reliability of the decision of u_k.

For clarification of these operations refer again to Figure 4.11 showing a simplified section of the trellis for our $K = 3$ RSC code. In this figure, as before, solid lines represent transitions taken when the input bit is a -1, and dashed lines represent transitions taken when the input bit is a +1. We assume that the all-zero path is identified as the maximum likelihood path, and this path is shown as a bold line. Also shown are the paths which merge with this ML path. It can be seen from the figure that the ML path gives a value of -1 for u_k, but the paths merging with the ML path at trellis stages S_k, S_{k+1}, S_{k+3} and S_{k+4} all give a value of +1 for the bit u_k. Hence, if we assume for simplicity that $\sigma = 4$, from Equation 4.68 the LLR $L(u_k|\underline{y})$ will be given by -1 multiplied by the minimum of the metric differences Δ_k^0, Δ_{k+1}^0, Δ_{k+3}^0 and Δ_{k+4}^0.

4.3.6.2 Implementation of the SOVA Algorithm

The SOVA algorithm can be implemented as follows. For each state at each stage in the trellis the metric $M(\underline{s}_k^s)$ is calculated for both of the two paths merging into the state using Equation 4.63. The path with the highest metric is selected as the survivor, and for this state at this stage in the trellis a pointer to the previous state along the surviving path is stored, just as in the classical Viterbi algorithm. However, in order to allow the reliability of the decoded bits to be calculated, the information used in Equation 4.68 to give $L(u_k|\underline{y})$ is also stored. Thus the difference Δ_k^s between the metrics of the surviving and the discarded paths is stored, together with a binary vector containing $\delta + 1$ bits, which indicate whether or not the discarded path would have given the same series of bits u_l for $l = k$ back to $l = k - \delta$ as

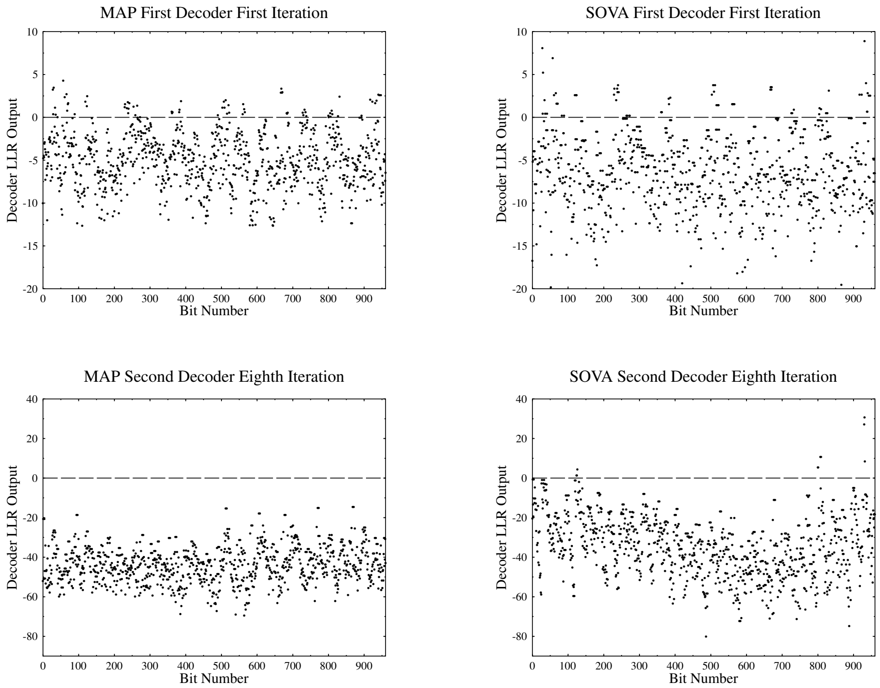


Figure 4.12: Soft Outputs from the SOVA Algorithm Compared to the MAP Algorithm for a Transmitted Stream of all -1

the surviving path does. This series of bits is called the update sequence in [61], and as noted by Hagenauer it is given by the result of a modulo two addition (ie an exclusive-or operation) between the previous $\delta + 1$ decoded bits along the surviving and discarded paths. When the SOVA has identified the ML path, the stored update sequences and metric differences along this path are used in Equation 4.68 to calculate the values of $L(u_k|\underline{y})$.

The SOVA algorithm described in this section is the least complex of all the soft-in soft-out decoders discussed in this chapter. In [59] it is shown by Robertson et al. that the SOVA algorithm is about half as complex as the Max-Log-MAP algorithm. However, the SOVA algorithm is also the least accurate of the algorithms we have described in this chapter and, when used in an iterative turbo decoder, performs about 0.6 dB worse [59] than a decoder using the MAP algorithm. Figure 4.12 compares the LLRs output from the component decoders in an iterative turbo decoder using both the MAP and the SOVA algorithm. The same encoder, all -1 input sequence, and channel SNR as described for Figure 4.10 were used. It can be seen that the outputs of the SOVA algorithm are significantly more noisy than those from the MAP algorithm. For the second decoder at the eighth iteration the MAP algorithm gives LLRs which are all negative, and hence gives no bit errors. However it can be seen from Figure 4.10 that, even after eight iterations, the SOVA algorithm still gives some positive LLRs, and hence will make several bit errors.

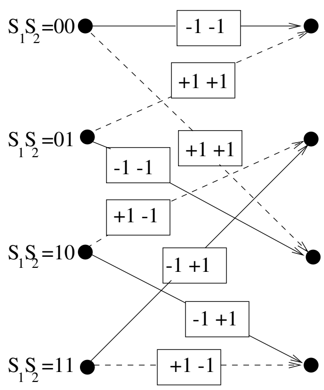


Figure 4.13: State Transition Diagram for our (2,1,3) RSC Component Codes

Let us now augment our understanding of iterative turbo decoding by considering a specific example.

4.3.7 Turbo Decoding Example

In this section we discuss an example of turbo decoding using the SOVA algorithm detailed in Section 4.3.6. This example serves to illustrate the details of the SOVA algorithm and the iterative decoding of turbo codes discussed in Section 4.3.4.

We consider a simple half-rate turbo code using the $K = 3$ Recursive Systematic Convolutional (RSC) code, with generator polynomials expressed in octal form as 7 and 5, shown in Figure 4.2. Two such codes are combined, as shown in Figure 4.1, with a 3x3 block interleaver to give a simple turbo code. The parity bits from both the component codes are punctured, so that alternate parity bits from the first and the second component encoder are transmitted. Thus the first, third, fifth, seventh and ninth parity bits from the first component encoder are transmitted, and the second, fourth, sixth and eighth parity bits from the second component encoder are transmitted. The first component encoder is terminated using two bits chosen to take this encoder back to the all zero state. The transmitted sequence will therefore contain nine systematic and nine parity bits. Of the systematic bits seven will be the input bits, and two will be the bits chosen to terminate the first trellis. Of the nine parity bits five will come from the first encoder, and four from the second encoder.

The state transition diagram for the component RSC codes is shown in Figure 4.13. As in all our diagrams in this section, a solid line denotes a transition resulting from a -1 input bit, and a dashed lines represents an input bit of +1. The figures within the boxes along the transition lines give the output bits associated with that transition - the first bit is the systematic bit, which is the same as the input bit, and the second is the parity bit.

For the sake of simplicity we assume that an all -1 input sequence is used. Thus there will be seven input bits which are -1, and the encoder trellis will remain in the $S_1S_2 = 00$ state. The two bits necessary to terminate the trellis will be -1 in this case and, as can be seen from Figure 4.13, the resulting parity bits will also be -1. Thus all 18 of the transmitted bits will be -1 for an all -1 input sequence. Assuming that BPSK modulation is used with the transmitted symbols being -1 or +1, the transmitted sequence will be a series of 18 -1's. The

Input Bit	Systematic Bit	Parity Bits Coder 1	Coder 2	Transmitted Sequence	Received Sequence
-1	-1	-1	–	-1,-1	-2.1,-0.1
-1	-1	–	-1	-1,-1	-1.4,-1.4
-1	-1	-1	–	-1,-1	-1.7,-0.5
-1	-1	–	-1	-1,-1	+0.9,+0.5
-1	-1	-1	–	-1,-1	+1.2,-1.7
-1	-1	–	-1	-1,-1	-1.1,-1.1
-1	-1	-1	–	-1,-1	-0.7,-0.8
-	-1	–	-1	-1,-1	-2.4,-1.9
-	-1	-1	–	-1,-1	-1.6,-0.9

Table 4.1: Input and Transmitted Bits for Turbo Decoding Example

received channel output sequence for our example, together with the input and transmitted bits detailed above, is shown in Table 4.1. Notice that approximately half the parity bits from each component encoder are punctured – this is represented by a dash in Table 4.1. Also note that the received channel sequence values shown in Table 4.1 are the matched filter outputs, which were denoted by y_{kl} in previous sections. If hard decision demodulation were used then negative values would be decoded as -1's, and positive values as +1's. It can be seen that from the 18 coded bits which were transmitted, all of which were -1, three would be decoded as +1 if hard decision demodulation were used.

In order to illustrate the difference between iterative turbo decoding and the decoding of convolutional codes, we initially consider how the received sequence shown in Table 4.1 would be decoded by a convolutional decoder using the Viterbi algorithm. Imagine the half-rate $K = 3$ RSC code detailed above used as an ordinary convolutional code to encode an input sequence of seven -1's. If trellis termination was used then two -1's would be employed to terminate the trellis, and the transmitted sequence would consist of 18 -1's, just as for our turbo coding example. If the received sequence was as shown in Table 4.1, then the Viterbi algorithm decoding this sequence would have the trellis diagram shown in Figure 4.14. The metrics shown in this figure are given by the cross correlation of the received and expected channel sequences for a given path, and the Viterbi algorithm maximises this metric to find the Maximum Likelihood (ML) path, which is shown by the bold line in Figure 4.14. Notice that at each state in the trellis where two paths merge, the path with the lower metric is discarded and its metric is shown crossed out in the figure. As can be seen from Figure 4.14, the Viterbi algorithm makes an incorrect decision at stage $k = 6$ in the trellis and selects a path other than the all zero path as the survivor. This results in three of the seven bits being decoded incorrectly as +1's.

Having seen how Viterbi decoding of a RSC convolutional code would fail and produce three errors given our received sequence, we now proceed to detail the operation of an iterative turbo decoder for the same channel sequence. Consider first the operation of the first component decoder in the first iteration. The component decoder uses the SOVA algorithm to decide upon not only the most likely input bits, but also the LLRs of these bits, as described in Section 4.3.6. We will describe here how the SOVA algorithm calculates these LLRs for the channel values given in Table 4.1.

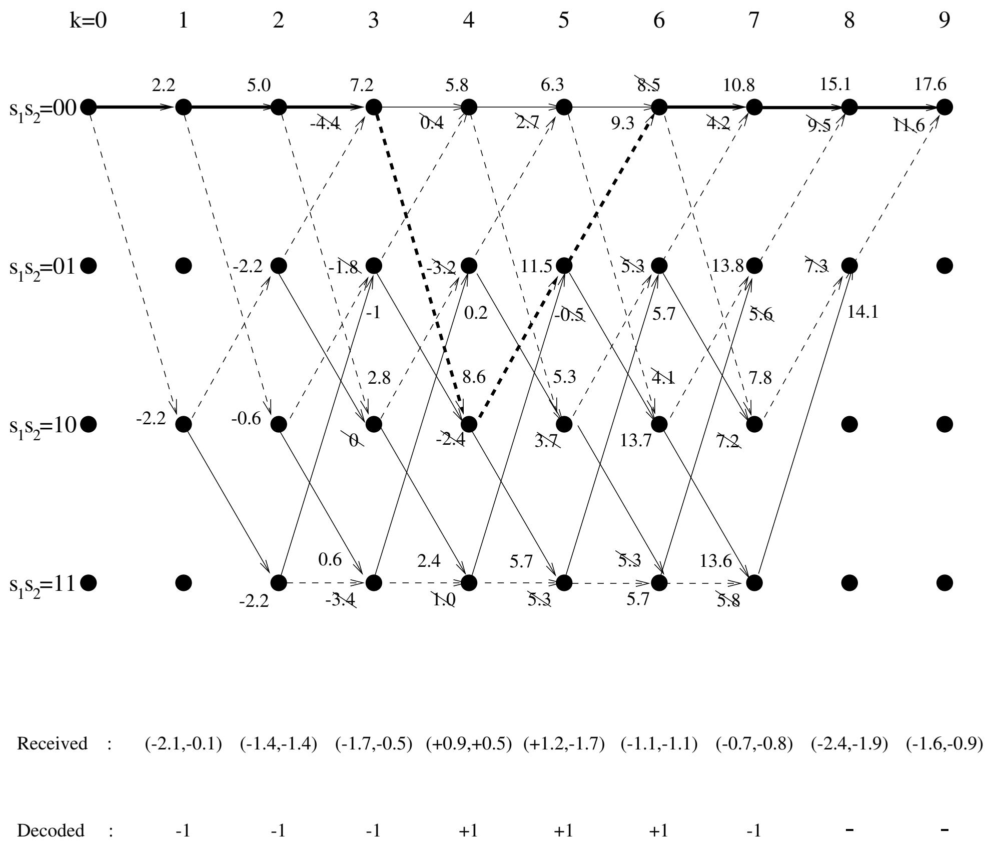


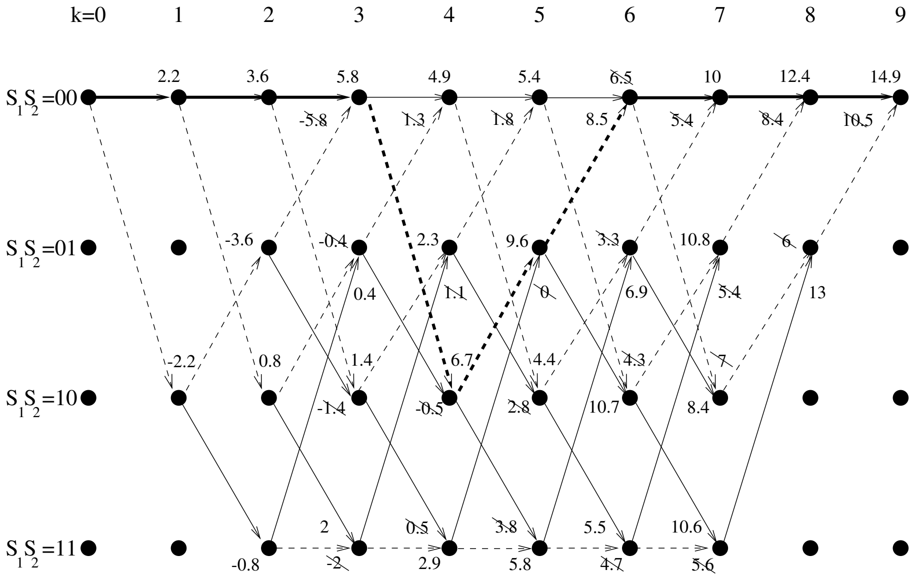
Figure 4.14: Trellis Diagram for the Viterbi Decoding of the Received Sequence Shown in Table 4.1

The metric for the SOVA algorithm is given by Equation 4.63, which is repeated here for convenience:

$$M(\underline{s}_k^s) = M(\underline{s}_{k-1}^{\grave{s}}) + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{Q} y_{kl}x_{kl}. \tag{4.69}$$

Here $M(\underline{s}_{k-1}^{\grave{s}})$ is the metric for the surviving path through the state $S_{k-1} = \grave{s}$ at stage $k-1$ in the trellis, u_k and x_{kl} are the input bit and the transmitted channel sequence associated with a given transition, y_{kl} is the received channel sequence for that transition and L_c is the channel reliability value, as defined in Equation 4.11. As initially we are considering the operation of the first decoder in the first iteration there is no a-priori information and hence we have $L(u_k) = 0$ for all k, which corresponds to an a-priori probability of 0.5. The received sequence given in Table 4.1 was derived from the transmitted channel sequence (which has $E_b = 1$) by adding AWGN with variance $\sigma = 1$. Hence, as the fading amplitude is $a = 1$, from Equation 4.11 we have for the channel reliability measure $L_c = 2$.

Figure 4.15 shows the trellis for this first component decoder in the first iteration. Due to the puncturing of the parity bits used at the encoder, the second, fourth, sixth and eighth



Received - $L_c y_{kl}/2$	(-2.1,-0.1)	(-1.4,0)	(-1.7,-0.5)	(+0.9,0)	(+1.2,-1.7)	(-1.1,0)	(-0.7,-0.8)	(-2.4,0)	(-1.6,-0.9)
A-Priori - $L(u_k)/2$	0	0	0	0	0	0	0	0	0
Decoded	-1	-1	-1	+1	+1	+1	-1	-1	-1

Figure 4.15: Trellis Diagram for the SOVA Decoding in the First Iteration of the First Decoder

parity bits have been received as zeros. The a-priori and channel values shown in Figure 4.15 are given as $L(u_k)/2$ and $L_c y_{kl}/2$ so that the metric values, given by Equation 4.69, can be calculated by simple addition and subtraction of the values shown. As we have $L(u_k) = 0$ and $L_c = 2$, these metrics are again given by the cross correlation of the expected and received channel sequences. Notice however that because of the puncturing used the metric values shown in Figure 4.15 are not the same as those in Figure 4.14. Despite this the ML path, shown by the bold line in Figure 4.15, is the same as the one that was chosen by the Viterbi algorithm shown in Figure 4.14, with three of the input bits being decoded as +1's rather than -1's.

We now discuss how, having determined the ML path, the SOVA algorithm finds the LLRs for the decoded bits. Figure 4.16 is a simplified version of the trellis from Figure 4.15, which shows only the ML path and the paths that merge with this ML path and are discarded. Also shown are the metric differences, denoted by Δ_k^s in Section 4.3.6, between the ML and the discarded paths. These metric differences, together with the previously defined update sequences that indicate for which of the bits the survivor and discarded paths would have given different values, are stored by the SOVA algorithm for each node at each stage in the trellis. When the ML path has been identified, the algorithm uses these stored values along the ML path to find the LLR for each decoded bit. Table 4.2 shows these stored values for our example trellis shown in Figures 4.15 and 4.16. The calculation of the decoded LLRs shown

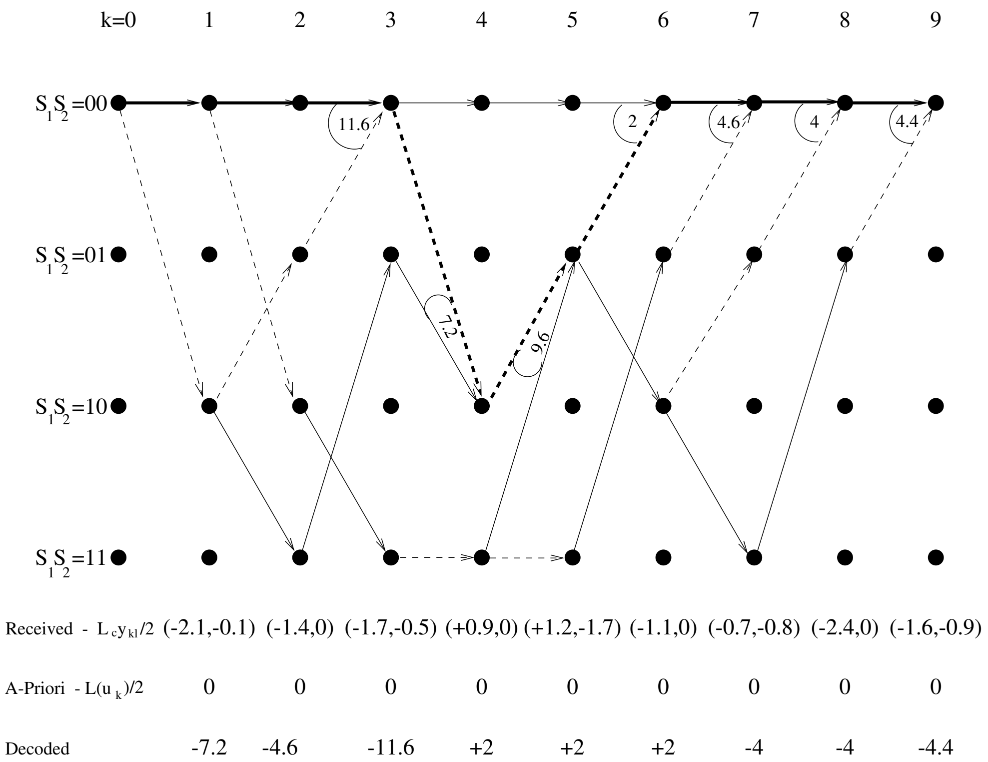


Figure 4.16: Simplified Trellis Diagram for the SOVA Decoding in the First Iteration of the First Decoder

Trellis Stage k	Decoded Bit u_k	Metric Difference Δ_k^s	Update Sequence	Decoded LLR
1	-1	–	–	-7.2
2	-1	–	–	-4.6
3	-1	11.6	111	-11.6
4	+1	7.2	1001	+2
5	+1	9.6	10010	+2
6	+1	2	111000	+2
7	-1	4.6	1100010	-4
8	-1	4	11100000	-4
9	-1	4.4	100100000	-4.4

Table 4.2: SOVA Output for the First Iteration of the First Decoder

in this table is detailed at a later stage.

Notice in Table 4.2 that at trellis stages $k = 1$ and $k = 2$ there is no metric difference or update sequence stored because, as can be seen from Figures 4.15 and 4.16, there are no paths merging with the ML path at these stages. For all subsequent stages there is a merging path, and values of the metric differences and update sequences are stored. For the update sequence a 1 indicates that the ML and the discarded merging path would have given different values for a particular bit. At stage k in the trellis we have taken the Most Significant Bit (MSB), on the left hand side, to represent u_k, the next bit to represent u_{k-1}, etc until the Least Significant Bit (LSB), which represents u_1. For our RSC code any two paths merging at trellis stage k give different values for the bit u_k, and so the MSB in the update sequences in Table 4.2 is always 1. Notice furthermore that although in our example the update sequences are all of different lengths, this is only because of the very short frame length we have used. More generally, as explained in Section 4.3.6, all the stored update sequences will be $\delta + 1$ bits long, where δ is usually set to be five times the constraint length of the convolutional code (15 in our case). At this stage it is beneficial for the reader to verify the update sequences of Table 4.2 using Figure 4.16.

We now explain how the SOVA algorithm can use the stored update sequences and metric differences along the ML path to calculate the LLRs for the decoded bits. Equation 4.68, which is repeated here as Equation 4.70 for convenience, shows that the decoded a-posteriori LLR $L(u_k|\underline{y})$ for a bit u_k is given by the minimum metric difference of merging paths along the ML path:

$$L(u_k|\underline{y}) \approx u_k \min_{\substack{i=k\cdots k+\delta \\ u_k \neq u_k^i}} \Delta_i^{s_i}. \tag{4.70}$$

This minimum is taken only over the metric differences for stages $i = k, k+1, \cdots k+\delta$ where the value u_k^i of the bit u_k given by the path merging with the ML path at stage i is different from the value given for this bit by the ML path. Whether or not the condition $u_k = u_k^i$ is met is determined using the stored update sequences. Denoting the update sequence stored at stage l along the ML path as $\underline{e}_l$, for each bit u_k the SOVA algorithm examines the MSB of $\underline{e}_k$, the 2nd MSB of $\underline{e}_{k+1}$, etc up to the $(\delta + 1)$'th bit (which will be the LSB) of $\underline{e}_{k+\delta}$. For our example this examination of the update sequences is limited because of our short frame length, but the same principles are used. Taking the fourth bit u_4 as an example, to determine the decoded LLR $L(u_4|\underline{y})$ for this bit the algorithm examines the MSB of $\underline{e}_4$ in row four of Table 4.2, the 2nd MSB of $\underline{e}_5$ in row five, etc up to the 6'th MSB of $\underline{e}_9$ in row nine. It can be seen, from the corresponding rows in Table 4.2, that only the paths merging at stages $k = 4$ and $k = 6$ of the trellis give values different from the ML path for the bit u_4. Hence the decoded LLR $L(u_4|\underline{y})$ from the SOVA algorithm for this bit is calculated using Equation 4.70 as the value of the bit given by the ML path (+1) times the minimum of the metric differences stored at stages 4 and 6 of the trellis (7.2 and 2), yeilding $L(u_4|\underline{y}) = +2$. For the next bit, u_5, the MSB of $\underline{e}_5$ in row five of Table 4.2, the 2nd MSB of $\underline{e}_6$ in row six, etc up to the 5th MSB of $\underline{e}_9$ in row nine are examined, which indicate that a different value for this bit would have been given by the discarded paths at stages 5 and 6 of the trellis. Hence $L(u_5|\underline{y})$ also equals +2, as the metric difference, 2, at stage 6 in the trellis is less than the metric difference, 9.6, at stage 5. For the next bit, u_6, the update sequences indicate that all the merging paths from the 6th stage of the trellis to the end of the trellis would give values different to those given by the ML path. However, the minimum of all these metric differences is still 2, and so $L(u_6|\underline{y})$ also equals +2. This illustrates in the SOVA algorithm how one discarded path having a low metric difference can entirely determine the LLRs for all the bits, for which it gives a different value from the ML path, which is a consequence of taking the minimum in Equation 4.70. At stage 6 of the trellis, where the algorithm incorrectly chooses the non-zero path as the

Trellis Stage k	Decoder LLR Output $L(u_k|\underline{y})$	A-Priori Info. $L(u_k)$	Received Sys. Info. $L_c y_{ks}$	Extrinsic Information
1	-7.2	0	-4.2	-3
2	-4.6	0	-2.8	-1.8
3	-11.6	0	-3.4	-8.2
4	+2	0	+1.8	+0.2
5	+2	0	+2.4	-0.4
6	+2	0	-2.2	+4.2
7	-4	0	-1.4	-2.6
8	-4	0	-4.8	+0.8
9	-4.4	0	-3.2	-1.2

Table 4.3: Calculation of the Extrinsic Information from the First Decoder in the First Iteration using Equation 4.71

survivor, the metric difference between the chosen (incorrect) path and the discarded path is the lowest metric difference (2) encountered along the ML path. Hence the LLRs for the three incorrectly decoded bits, i.e. for u_4, u_5 and u_6, have the lowest magnitudes of any of the decoded bits.

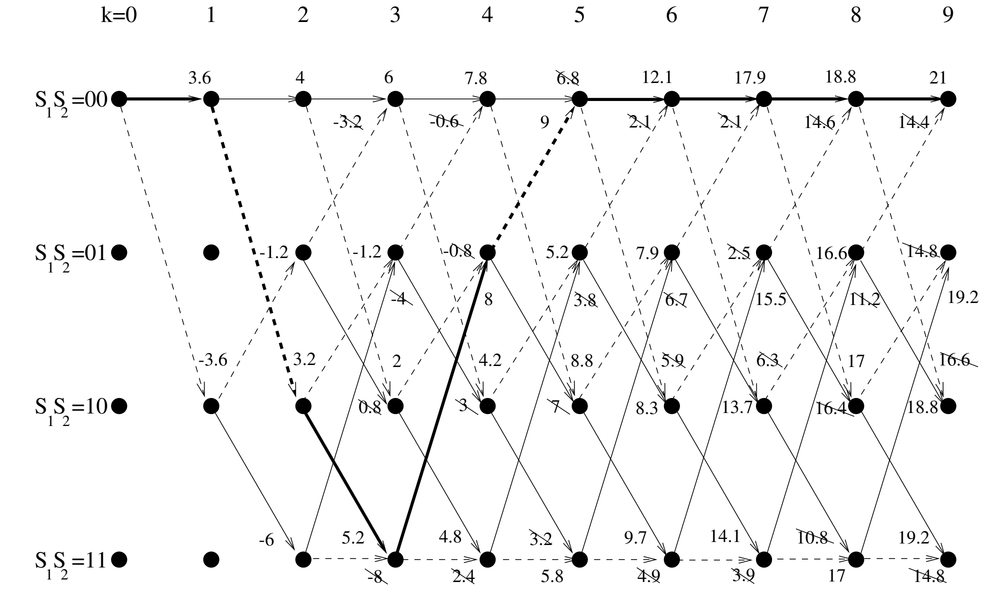
The remaining decoded LLR values in Table 4.2 are computed following a similar procedure. However also worth noting explicitly is the LLR for the bit u_3. Examination of the MSB of $\underline{e}_3$ in row three of Table 4.2, the 2nd MSB of $\underline{e}_4$ in row four, etc up to the 7th MSB of $\underline{e}_9$, reveals that only the path merging with the ML path at the 3rd stage of the trellis would give a different value for u_3. Hence the minimum for the LLR $L(u_3|\underline{y})$ in Equation 4.70 is over one term, and the magnitude of $L(u_3|\underline{y})$ is determined by the metric difference of the merging path at stage $k = 3$ of the trellis, which is 11.6.

We now move on to describing the operation of the second component decoder in the first iteration. This decoder uses the extrinsic information from the first decoder as a-priori information to assist its operation, and therefore should be able to provide a better estimate of the encoded sequence than the first decoder was. Equation 4.46 from Section 4.3.4 gives the extrinisic information from the MAP decoder as

$$L_e(u_k) = L(u_k|\underline{y}) - L(u_k) - L_c y_{ks}. \tag{4.71}$$

The same equation can be derived for all the soft-in soft-out decoders which are used as component decoders for turbo codes. This equation states that the extrinsic information $L_e(u_k)$ is given by the soft output $L(u_k|\underline{y})$ from the decoder with the a-priori information $L(u_k)$ (if any was available) and the received systematic channel information $L_c y_{ks}$ subtracted. Table 4.3 shows the extrinsic information calculated from Equation 4.71 from the first decoder, which is then interleaved by a 3x3 block interleaver and used as the a-priori information for the second component decoder. The second component decoder also uses the interleaved received systematic channel values, and the received parity bits from the second encoder which were not punctured (ie the second, fourth, sixth and eighth bits).

Figure 4.17 shows the trellis for the SOVA decoding of the second decoder in the first iteration. The extrinsic information values from Table 4.3 are shown after being interleavered and divided by two as $L(u_k)/2$. Also shown is the channel information $L_c y_{ks}/2$ used by this



Received - $L_c y_{kl}/2$	(-2.1,0)	(+0.9,-1.4)	(-0.7,0)	(-1.4,+0.5)	(+1.2,0)	(-2.4,-1.1)	(-1.7,0)	(-1.1,-1.9)	(-1.6,0)
A-Priori - $L(u_k)/2$	-1.5	+0.1	-1.3	-0.9	-0.2	+0.4	-4.1	+2.1	-0.6
Decoded	-13.2	+2.2	-8.8	-8.8	+2.2	-6.6	-15.8	-4.2	-6.6

Figure 4.17: Trellis Diagram for the SOVA Decoding in the First Iteration of the Second Decoder

decoder. Notice that as the trellis is not terminated for the second component encoder, paths terminating in all four possible states of the trellis are considered at the decoder. However, the metric for the $S_1S_2 = 00$ state is the maximum of the four final metrics, and hence this all zero state is used as the final state of the trellis. Note furthermore that the metrics in Figure 4.17 are now calculated as the cross correlation of the received and expected channel information, plus the a-priori information $u_k L(u_k)/2$.

The ML path chosen by the second component decoder is shown by a bold line in Figure 4.17, together with the LLR values output by the decoder. These are calculated, using update sequences and minumum metric differences, in the same way as was explained for the first decoder using Figure 4.16 and Table 4.2. It can be seen tha the decoder makes an incorrect decision at stage $k = 5$ in the trellis and selects a path other than the all zero path as the survivor. However the incorrectly chosen path gives decoded bits of +1 for only two transitions, and hence only two, rather than three, decoding errors are made. Furthermore the difference in the metrics between the correct and the chosen path at trellis stage $k = 5$ is only 2.2, and so the magnitude of the decoded LLRs $L(u_k|\underline{y})$ for the two incorrectly decoded bits, u_2 and u_5, is only 2.2. This is significantly lower than the magnitudes of the LLRs for the other bits, and indicates that the algorithm is less certain about these two bits being +1 than it is about the other bits being -1.

Having calculated the LLRs from the second component decoder, the turbo decoder has

Trellis Stage k	Decoder LLR Output $L(u_k\|\underline{y})$	A-Priori Info. $L(u_k)$	Received Sys. Info. $L_c y_{ks}$	Extrinsic Information
1	-13.2	-3	-4.2	-6
2	+2.2	+0.2	+1.8	+0.2
3	-8.8	-2.6	-1.4	-4.8
4	-8.8	-1.8	-2.8	-4.2
5	+2.2	-0.4	+2.4	+0.2
6	-6.6	+0.8	-4.8	-2.6
7	-15.8	-8.2	-3.4	-4.2
8	-4.2	+4.2	-2.2	-6.2
9	-6.6	-1.2	-3.2	-2.2

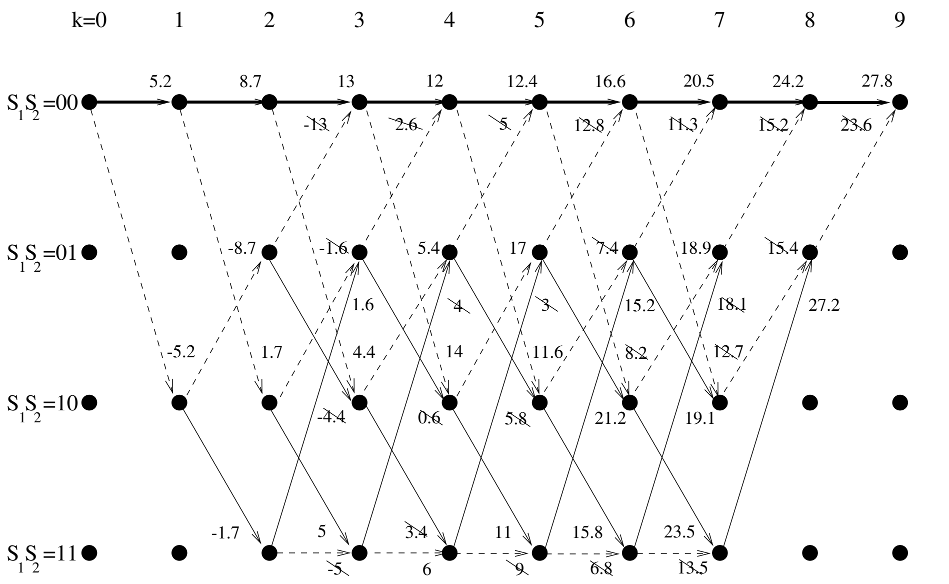
Table 4.4: Calculation of the Extrinsic Information from the Second Decoder in the First Iteration using Equation 4.71

now completed one iteration. The soft output LLR values from the second component decoder shown in bottom line of Figure 4.17 could now be de-interleaved and used as the output from the turbo decoder. This de-interleaving would result in an output sequence which gave negative LLRs for all the decoded bits except u_4 and u_5, which would be incorrectly decoded as +1's as their LLRs are both +2.2. Thus, even after only one iteration, the turbo decoder has decoded the received sequence with one less error than the convolutional decoder did. However generally better results are achieved with more iterations, and so we now progress to describe the operation of the turbo decoder in the second iteration.

In the second, and all subsequent, iterations the first component decoder is able to use the extrinsic information from the second decoder in the previous iteration as a-priori information. Table 4.4 shows the calculation of this extrinsic information using Equation 4.71 from the second decoder in the first iteration. It can be seen that it gives negative LLRs for all the bits except u_2 and u_5, and for these two bits the LLRs are close to zero. This extrinsic information is then de-interleaved and used as the a-priori information for the first decoder in the next (second) iteration. The trellis for this decoder is shown in Figure 4.18. It can be seen that this decoder uses the same channel information as it did in the first iteration. However now, in contrast to Figure 4.15, it also has a-priori information, to assist it in finding the correct path through the trellis. The selected ML path is again shown by a bold line, and it can be seen that now the correct all zero path is chosen. The second iteration is then completed by finding the extrinsic information from the first decoder, interleaving it and using it as a-priori information for the second decoder. It can be shown that this decoder will also now select the all zero path as the ML path, and hence the output from the turbo decoder after the seond iteration will be the correct all -1 sequence. This concludes our example of the operation of an iterative turbo decoder using the SOVA algorithm.

4.3.8 Comparison of the Component Decoder Algorithms

In this section we have detailed the iterative structure and the component decoders used for decoding turbo codes. A numerical example illustrating this decoding was given in the previous section. We now conclude by summarising the operation of the algorithms which can be used as component decoders, highlighting the similarities and differences between these



Received - $L_c y_{kl}/2$	(-2.1,-0.1)	(-1.4,0)	(-1.7,-0.5)	(+0.9,0)	(+1.2,-1.7)	(-1.1,0)	(-0.7,-0.8)	(-2.4,0)	(-1.6,-0.9)
A-Priori - $L(u_k)/2$	-3	-2.1	-2.1	+0.1	+0.1	-3.1	-2.4	-1.3	-1.1
Decoded	-9.4	-26	-7.4	-3.8	-3.8	-3.8	-9	-9	-4.2

Figure 4.18: Trellis Diagram for the SOVA Decoding in the Second Iteration of the First Decoder

algorithms, and noting their relative complexities and performances.

The MAP algorithm is the optimal component decoder for turbo codes. It finds the probability of each bit u_k being a +1 or -1 by calculating the probability for each transition from state $S_{k-1} = \grave{s}$ to $S_k = s$ that could occur if the input bit was +1, and similarly for every transition that could occur if the input bit was -1. As these transitions are mutually exclusive, the probability of any one of them occurring is simply the sum of their individual probabilities, and hence the LLR for a bit u_k is given by the ratio of two sums of probabilities, as in Equation 4.17.

Due to the Markov nature of the trellis and the assumption that the output from the trellis is observed in memoryless noise, the individual probabilities of the transitions in Equation 4.17 can be expressed as the product of three terms – $\alpha_{k-1}(\grave{s})$, $\beta_k s$ and $\gamma_k(\grave{s}, s)$, as in Equation 4.20. By definition the $\alpha_{k-1}(\grave{s})$ term gives the probability that the trellis reaches state $S_{k-1} = \grave{s}$ and that the received sequence up to this point is $\underline{y}_{j<k}$. The state transition probability, $\gamma_k(\grave{s}, s)$, is defined as the probability that given the trellis is in state $S_{k-1} = \grave{s}$ it moves to state $S_k = s$ and the received sequence for that transition is $\underline{y}_k$. Finally, the $\beta_k(s)$ term gives the probability that given the trellis is in state $S_k = s$, the received sequence from this point to the end of the trellis is $\underline{y}_{j>k}$. The state transition probabilities $\gamma_k(\grave{s}, s)$ are calculated from the received and expected channel sequences, y_{kl} and x_{kl}, for a given transition and the a-priori LLR $L(u_k)$ of the bit associated with this transition, as seen in Equation 4.40

for an AWGN channel. The $\alpha_{k-1}(\grave{s})$ terms can then be calculated using a forward recursion through the trellis, as in Equation 4.25, and similarly the $\beta_k(s)$ terms are calculated using Equation 4.28 by a backward recursion through the trellis. The output LLR $L(u_k|\underline{y})$ from the MAP algorithm is then determined for each bit u_k by finding the probability of each transition from stage S_{k-1} to S_k in the trellis. These transitions are divided into two groups – those that would have resulted if the bit u_k was a +1, and those that would have resulted if the bit u_k was a -1. The LLR $L(u_k|\underline{y})$ for the bit u_k is then given by the logarithm of the ratio of these probabilities.

The MAP algorithm is optimal for the decoding of turbo codes, but is extremely complex. Furthermore, because of the multiplications used in the recursive calculation of the $\alpha_{k-1}(\grave{s})$ and $\beta_k(s)$ terms, and the exponents used to calculate the $\gamma_k(\grave{s}, s)$ terms, it often suffers from numerical problems in practice. The Log-MAP algorithm is theoretically identical to the MAP algorithm, but transfers its operations to the log domain. Thus multiplications are replaced by additions, and hence the numerical problems of the MAP algorithm are circumvented, while the associated complexity is dramatically reduced.

The Max-Log-MAP algorithm further reduces the complexity of the Log-MAP algorithm using the maximisation approximation given in Equation 4.48. This has two effects on the operation of the algorithm compared to that of the Log-MAP algorithm. Firstly, as can be seen by examining Equation 4.55, it means that only two transitions are considered when finding the LLR $L(u_k|\underline{y})$ for each bit u_k – the best transition from $S_{k-1} = \grave{s}$ to $S_k = s$ that would give $u_k = +1$ and the best that would give $u_k = -1$. Similarly in the recursive calculations of the $A_k(s) = \ln(\alpha_k(s))$ and $B_k(s) = \ln(\beta_k(s))$ terms of Equations 4.52 and 4.53 the approximation means that only one transition, the most likely one, is considered when calculating $A_k(s)$ from the $A_{k-1}(\grave{s})$ terms and $B_{k-1}(\grave{s})$ from the $B_k(s)$ terms. This means that although $A_{k-1}(\grave{s})$ should give the logarithm of the probability that the trellis reaches state $S_{k-1} = \grave{s}$ along *any* path from the initial state $S_0 = 0$, in fact it gives the logarithm of the probability of only the *most likely* path to state $S_{k-1} = \grave{s}$. Similarly $B_k(s) = \ln(\beta_k(s))$ should give the logarithm of the probability of the received sequence $\underline{y}_{j>k}$ given only that the trellis is in state $S_k = s$ at stage k. However the maximisation in Equation 4.53 used in the recursive calculation of the $B_k(s)$ terms means that only the most likely path from state $S_k = s$ to the end of the trellis is considered, and not all paths.

Hence the Max-Log-MAP algorithm finds the LLR $L(u_k|\underline{y})$ for a given bit u_k by comparing the probability of the most likely path giving $u_k = +1$ to the probability of the most likely path giving $u_k = -1$. For the next bit, u_{k+1}, again the best path that would give $u_{k+1} = +1$ and the best path that would give $u_{k+1} = -1$ are compared. One of these "best paths" will always be the maximum likelihood path, and so will not change from one stage to the next, whereas the other may change. In contrast the MAP and the Log-MAP algorithms consider every path in the calculation of the LLR for each bit. All that changes from one stage to the next is the division of paths into those that give $u_k = +1$ and those that give $u_k = -1$. Thus the Max-Log-MAP algorithm gives a degraded performance compared to the MAP and Log-MAP algorithms.

In the SOVA algorithm the Maximum Likelihood (ML) path is found by maximising the metric given in Equation 4.63. The recursion used to find this metric is identical to that used to find the $A_k(s)$ terms in Equation 4.52 in the Max-Log-MAP algorithm. Once the ML path has been found, the hard decision for a given bit u_k is determined by which transition the ML path took between trellis stages S_{k-1} and S_k. The LLR $L(u_k|\underline{y})$ for this bit is determined by examining the paths which merge with the ML path that would have given a different hard decision for the bit u_k. The LLR is taken to be the minimum metric difference for these merging paths which would have given a different hard decision for the bit u_k. Using the notation associated with the Max-Log-MAP algorithm, once a path merges with the ML path, it will have the same value of $B_k(s)$ as the ML path. Hence, as the metric in the SOVA is identical to the $A_k(s)$ values in the Max-Log-MAP, taking the difference between the metrics of the two merging paths in the SOVA algorithm is equivalent to taking the difference between two values of $(A_{k-1}(\grave{s}) + \Gamma_k(\grave{s}, s) + B_k(s))$ in the Max-Log-MAP algorithm, as in Equation 4.55. The only difference is that in the Max-Log-MAP algorithm one path will be the ML path, and the other will be the most likely path that gives a different hard decision for u_k. In the SOVA algorithm again one path will be the ML path, but the other may not be the most likely path that gives a different hard decision for u_k. Instead, it will be the most likely path that gives a different hard decision for u_k and survives to merge with the ML path. Other, more likely paths, which give a different hard decision for the bit u_k to the ML path may have been discarded before they merge with the ML path. Thus the SOVA algorithm gives a degraded performance compared to the Max-Log-MAP algorithm. However, as pointed out in [59] by Robertson et al., the SOVA and Max-Log-MAP algorithms will always give the same hard decisions, as in both algorithms these hard decisions are determined by the ML path, which is calculated using the same metric in both algorithms.

It was also noted in [59] that the outputs from the SOVA algorithm contain no bias when compared to those from the Max-Log-MAP algorithm, they are just more noisy. However consideration of the arguments above make it clear that when the most likely path that gives a different hard decision for u_k survives to merge with the ML path, the outputs from the SOVA and the Max-Log-MAP algorithms will be identical. Otherwise when this most likely path which gives a different hard decision for u_k does not survive to merge with the ML path, the merging path that is used to calculate the soft output from the SOVA algorithm will be less likely than the path which should have been used. Thus it will have a lower metric, and so the metric difference used in Equation 4.68 will be higher than it should be. Therefore, although the sign of the soft outputs from the SOVA algorithm will be identical to those from the Max-Log-MAP algorithm, their magnitudes will either be identical or higher. This can be seen in Figure 4.19, which shows the Probability Density Function (PDF) for the differences between the absolute values of the soft outputs from the SOVA and the Max-Log-MAP algorithms. Both decoders were used as the first decoder in the first iteration, and again the same encoder, all -1 input sequence, and channel SNR as described for Figure 4.10 were used. It can be seen that for more than half of the decoder outputs the two algorithms give identical values. It can also be seen that the absolute values given by the SOVA algorithm are never less than those given by the Max-Log-MAP algorithm.

A comparison of the complexities of the Log-MAP, the Max-Log-MAP, and the SOVA algorithms is given in [59]. The relative complexity of the algorithms depends on the constraint length K of the convolutional codes used, but it is shown that the Max-Log-MAP algorithm is about twice as complex as the SOVA algorithm. The Log-MAP algorithm is about 50% more complex than the Max-Log-MAP algorithm due to the look-up operation required for finding the correction factors $f_c(x)$. Viterbi noted furthermore [129, 130] that the Max-Log-MAP algorithm can be viewed as two generalised Viterbi decoders (one for the forward recursion, and one for the backward recursion) together with a generalised dual-maxima computation (for calculating the soft outputs using Equation 4.55). The complexity

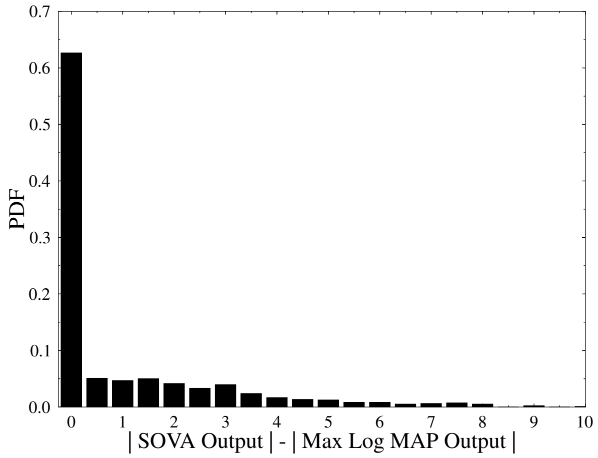


Figure 4.19: Probability Density Function of Differences Between Absolute Values of Soft Outputs from the SOVA and Max Log MAP Algorithms

of the dual-maxima computation is lower than that of the Viterbi decoder, and hence Viterbi estimated the complexity of the Max-Log-MAP algorithm to be lower than three times that of the Viterbi algorithm. Our related calculations suggest for a K=3 code that the Max Log MAP decoder is about 2.6 times as complex, and the Log MAP decoder is about 4 times as complex, as the standard Viterbi algorithm.

The performance of the algorithms when used in the iterative decoding of turbo codes follows in the same order as their complexities, with the best performance given by the Log-MAP algorithm, then the Max-Log-MAP algorithm, and the worst performance exhibited by the SOVA algorithm. We will compare the performance of these three algorithms when decoding various turbo codes in Section 4.4.

4.3.9 Conclusions

In this section we have described the techniques used for the decoding of turbo codes. Although it is possible to optimally decode turbo codes in a single non-iterative step, for complexity reasons a non-optimum iterative decoder is almost always preferred. Such an iterative decoder employs two component soft-in soft-out decoders, and we have described the MAP, Log-MAP, Max-Log-MAP and SOVA algorithms, which can all be used as the component decoders. The MAP algorithm is optimal for this task, but it is extremely complex. The Log-MAP algorithm is a simplification of the MAP algorithm, and offers the same optimal performance with a reasonable complexity. The other two algorithms, the Max-Log-MAP and the SOVA, are both less complex again, but give a slightly degraded performance.

Having described the principles behind the encoding and decoding of turbo codes we now move on to presenting our simulation results demonstrating the excellent performance of turbo codes for various scenarios.

4.4 Turbo Coded BPSK Performance Over Gaussian Channels

In the previous two sections we have discussed the structure of both the encoder and the decoder in a turbo codec. In this section we present simulation results for turbo codes using Binary Phase Shift Keying (BPSK) over Additive White Gaussian Noise (AWGN) channels. We show that there are many parameters, some of which are interlinked, which affect the performance of turbo codes. Some of these parameters are:

- The component decoding algorithm used.
- The number of decoding iterations used.
- The frame-length or latency of the input data.
- The specific design of the interleaver used.
- The generator polynomials and constraint lengths of the component codes.

In this section we investigate how all of these parameters affect the performance of turbo codes. The standard parameters we have used in our simulations are shown in Table 4.5. All our results presented in this section consider turbo codes using BPSK modulation over an AWGN channel. The turbo encoder uses two component Recursive Convolutional Codes (RSCs) in parallel. Our standard RSC component codes are $K = 3$ codes with generator polynomials $G_0 = 7$ and $G_1 = 5$ in octal representation. These generator polynomials are optimum in terms of maximising the minimum free distance of the component codes [104]. The effects of varying these generator polynomials are examined in Section 4.4.5. The standard interleaver used between the two component RSC codes is a 1000 bit random interleaver with odd-even separation [75]. The effects of changing the length of the interleaver, and its structure, are examined in Sections 4.4.4 and 4.4.6. Unless otherwise stated, the results of this section are valid for half-rate codes, where half the parity bits generated by each of the two component RSC codes are punctured. However, for comparison, we also include some results for turbo codes where all the parity bits from both component encoders are transmitted, leading to a one-third rate code. At the decoder two component, soft-in soft-out, decoders are used in parallel in the structure shown in Figure 4.3. In most of our simulations we use the Log-MAP decoder, but the effect of using other component decoders is investigated in Section 4.4.3. Usually 8 iterations of the component decoders are used, but in the next section we consider the effect of the number of iterations.

4.4.1 Effect of the Number of Iterations Used

Figure 4.20 shows the performance of a turbo decoder using the MAP algorithm versus the number of decoding iterations which were used. For comparison, the uncoded BER and the BER obtained using convolutional coding with a standard (2,1,3) non-recursive convolutional

Channel	Additive White Gaussian Noise (AWGN)
Modulation	Binary Phase Shift Keying (BPSK)
Component Encoders	2 identical Recursive -Convolutional Codes (RSCs)
RSC Parameters	n=2,k=1,K=3 $G_0 = 7\ G_1 = 5$
Interleaver	1000 bit random interleaver with odd-even separation [75]
Puncturing Used	Half parity bits from each component encoder transmitted – give half-rate code
Component Decoders	Log-MAP decoder
Iterations	8

Table 4.5: Standard Turbo Encoder and Decoder Parameters Used

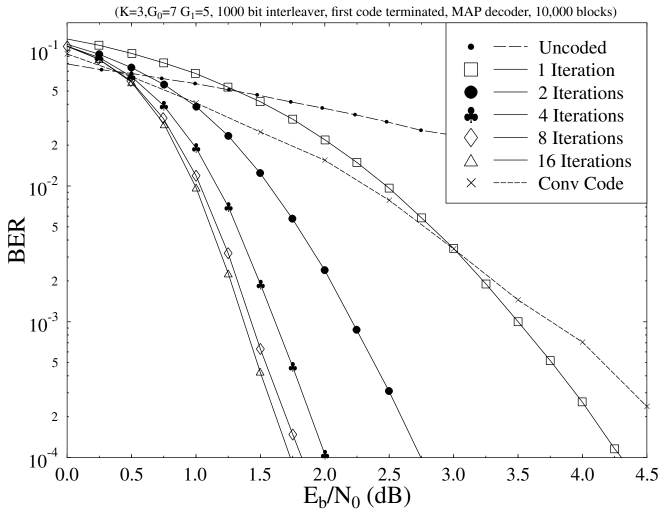


Figure 4.20: Turbo Coding BER Performance Using Different Numbers of Iterations of the MAP Algorithm. Other Parameters as in Table 4.5.

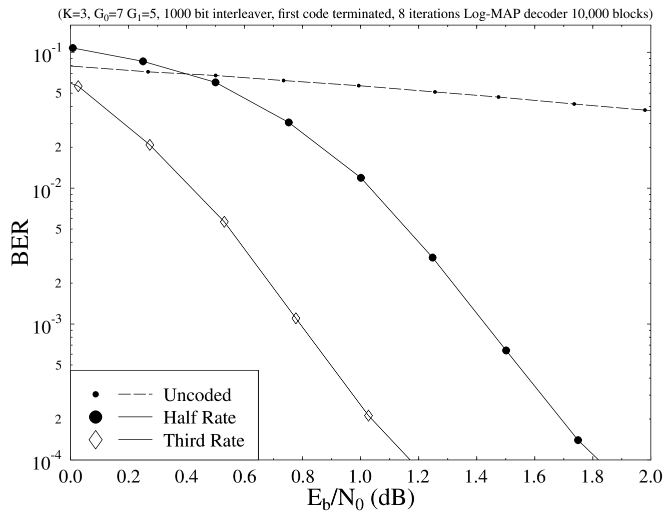


Figure 4.21: BER Performance Comparison Between One-Third and Half-Rate Turbo Codes using Parameters of Table 4.5.

code, are also shown. Like the component codes in the turbo encoder, the convolutional encoder uses the optimum octal generator polynomials of 7 and 5. It can be seen that the performance of the turbo code after one iteration is roughly similar to that of the convolutional code at low SNRs, but improves more rapidly than that of the convolutional coding as the SNR is increased. As the number of iterations used by the turbo decoder increases, the turbo decoder performs significantly better. However after 8 iterations there is little improvement achieved by using further iterations. For example it can be seen from Figure 4.20 that using 16 iterations rather than 8 gives an improvement of only about 0.1 dB. Similar results are obtained when using the SOVA algorithm – again there is little improvement in the BER performance of the decoder from using more than 8 iterations. Hence for complexity reasons usually only between about 4 and 12 iterations are used. Accordingly, unless otherwise stated, in our simulations we used 8 iterations. In the next section we consider the effects of puncturing.

4.4.2 Effects of Puncturing

As described in Section 4.2, in a turbo encoder two or more component encoders are used for generating parity information from an input data sequence. In our work we have used two RSC component encoders, and this is the arrangement most commonly used for turbo codes having coding rates below two-thirds. Typically, in order to generate a half-rate code, half the parity bits from each component encoder are punctured. This was the arrangement used in their seminal paper by Berrou et al. on the concept of turbo codes [21]. However, it is

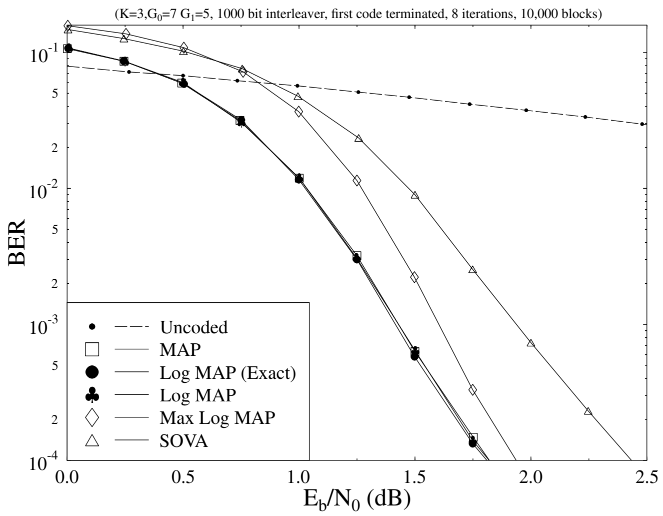


Figure 4.22: BER Performance Comparison Between Different Component Decoders for a Random Interleaver with L=1000. Other Parameters as in Table 4.5.

of course possible to omit the puncturing and transmit all the parity information from both component encoders, which gives a one-third rate code. The performance of such a code, compared to the corresponding half-rate code, is shown in Figure 4.21. In this figure the encoders use the same parameters as were described above for Figure 4.20. It can be seen that transmitting all the parity information gives a gain of about 0.6 dB, in terms of E_b/N_0, at a BER of 10^{-4}. This corresponds to a gain of about 2.4 dB in terms of channel SNR. Very similar gains are seen for turbo codes with different frame-lengths. Let us now consider the performance of the various soft-in soft-out component decoding algorithms which were described in Section 4.3.

4.4.3 Effect of the Component Decoder Used

Figure 4.22 shows a comparison between turbo decoders using the different component decoders described in Section 4.3 for a turbo encoder using the parameters described above. In this figure the "Log MAP (exact)" curve refers to a decoder which calculates the correction term $f_c(x)$ in Equation 4.56 of Section 4.3.5 exactly, ie using

$$f_c(x) = \ln(1 + \mathrm{e}^{-x}), \tag{4.72}$$

rather than using a look-up table as described in [59]. The Log MAP curve refers to a decoder which does use a look-up table with 8 values of $f_c(x)$ stored, and hence introduces an approximation to the calculation of the LLRs. It can be seen that, as expected, the MAP and the Log-MAP (exact) algorithms give identical performances. Furthermore, as Robertson

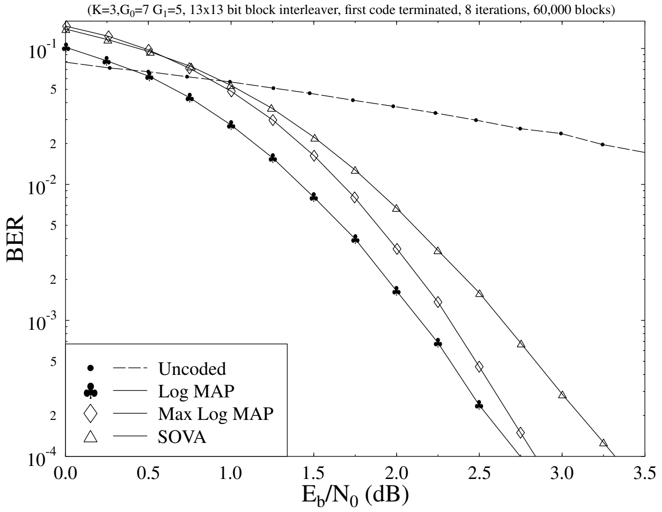


Figure 4.23: BER Performance Comparison Between Different Component Decoders for a L=169, 13x13, Block Interleaver. Other Parameters as in Table 4.5.

found [59], the look-up procedure for the values of the $f_c(x)$ correction terms introduces no degradation to the performance of the decoder.

It can also be seen from Figure 4.22 that the Max Log MAP and the SOVA algorithms both give a degradation in performance compared to the MAP and Log MAP algorithms. At a BER of 10^{-4} this degradation is about 0.1 dB for the Max Log MAP algorithm, and about 0.6 dB for the SOVA algorithm.

Figure 4.23 compares the Log MAP, Max Log MAP and SOVA algorithms for a turbo decoder with a frame-length of only 169 bits, rather than 1000 bits as was used for Figure 4.22. It can be seen that although all three decoders give a worse BER performance than those shown in Figure 4.22, the differences in the performances between the decoders are very similar to those shown in Figure 4.22. Similarly, Figure 4.24 compares these three decoding algorithms for a one-third rate code, and again the degradations relative to a decoder using the Log-MAP algorithm are about 0.1 dB for the Max-Log-MAP algorithm, and about 0.6 dB for the SOVA algorithm.

4.4.4 Effect of the Frame-Length of the Code

In the original paper on turbo coding by Berrou et al [21], and many of the subsequent papers, impressive results have been presented for coding with very large frame lengths. However for many applications, such as for example speech transmission systems, the large delays inherent in using high frame-lengths are unacceptable. Therefore an important area of turbo coding research is achieving as impressive results with short frame-lengths as have been

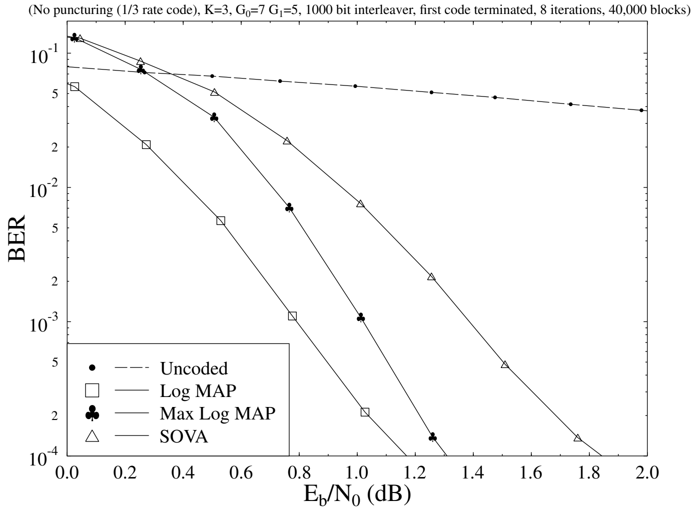


Figure 4.24: BER Performance Comparison Between Different Component Decoders for a Random Interleaver with L=1000 Using a $\frac{1}{3}$ Rate Code. Other Parameters as in Table 4.5.

demonstrated for long frame-length systems.

Figure 4.25 shows how dramatically the performance of turbo codes depends on the frame-length L used in the encoder. The 169 bit code would be suitable for use in a speech transmission systems at approximately 8 kbits/s with a 20 ms frame-length [131], while the 1000 bit code would be suitable for video transmission. The larger frame-length systems would be useful in data or non-real time transmission systems. It can be seen from Figure 4.25 that the performance of turbo codes is very impressive for systems with long frame lengths. However even for a short frame-length system, using 169 bits per frame, it can be seen that turbo codes give good results, comparable to or better than a constraint length $K = 9$ convolutional code. The use of the $K = 9$ convolutional code as a bench-marker is justified below.

As noted in Section 4.3, a single decode with the Log-MAP decoder is about 4 times as complex as decoding the same code using a standard Viterbi decoder. The curves shown in Figure 4.25, and in most of our results, use two component decoders with 8 iterations. Therefore the overall complexity of a turbo decoder is approximately $2 \times 8 \times 4 = 64$ times that of a Viterbi decoder for one of the component convolutional codes. This means that the complexity of our turbo decoder using eight iterations of two $K = 3$ component codes is approximately the same as the complexity of a Viterbi decoder for an ordinary $K = 9$ convolutional code. In order to provide a comparison between the performance of turbo codes and convolutional codes for similar complexity decoders, we will compare our $K = 3$ turbo codes with an 8 iteration decoder to a $K = 9$ convolutional code.

Figure 4.25 shows the performance of such a convolutional code. A non recursive (2,1,9)

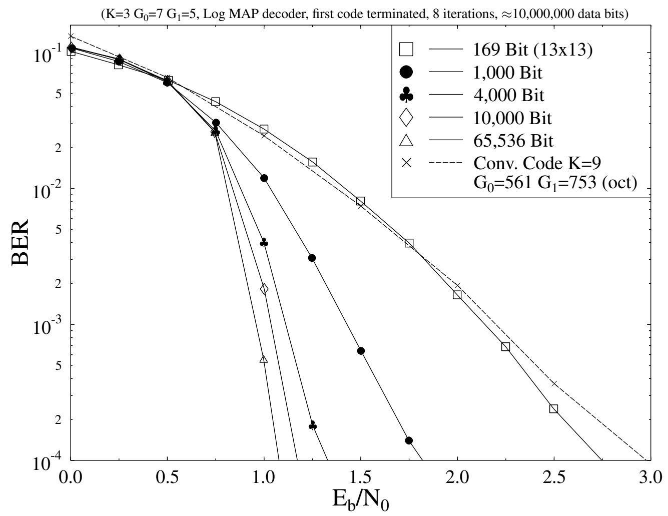


Figure 4.25: Effect of Frame-Length on the BER Performance of Turbo Coding. All Interleavers except $L = 169$ Block Interleaver Use Random Separated Interleavers [75]. Other Parameters as in Table 4.5.

convolutional code using the generator polynomials $G_0 = 561$ and $G_1 = 753$ in octal notation, which maximise the free distance of the code [104], was used. These generator polynomials provide the best performance in the AWGN channels we use in this section. A frame-length of 169 bits is used, and the code is terminated. It can be seen that even for the short frame-length of 169 bits, turbo codes out-perform similar complexity convolutional codes. As the frame-length is increased, the performance gain from using turbo codes, rather than high constraint length convolutional codes, increases dramatically.

Figure 4.26 shows how the performance of a one-third rate turbo code varies with the frame-length of the code. Again, the performance of the turbo code is better the longer the frame-length of the code, but impressive results are still obtained with a frame length of only 169 bits. Again the results for a $K = 9$ convolutional code are shown, this time using a third rate $n = 3, k = 1$ code with the optimal generator polynomials of $G_0 = 557$, $G_1 = 663$ and $G_2 = 711$ [104] in octal notation. Again it can be seen that the high constraint length convolutional code is out-performed by turbo codes with frame-lengths of 169 and higher.

Let us now consider the effect of using different RSC component codes.

4.4.5 The Component Codes

Both the constraint length and the generator polynomials used in the component codes of turbo codes are important parameters. Often in turbo codes the generator polynomials which lead to the largest minimum free distance for ordinary convolutional codes are used, although

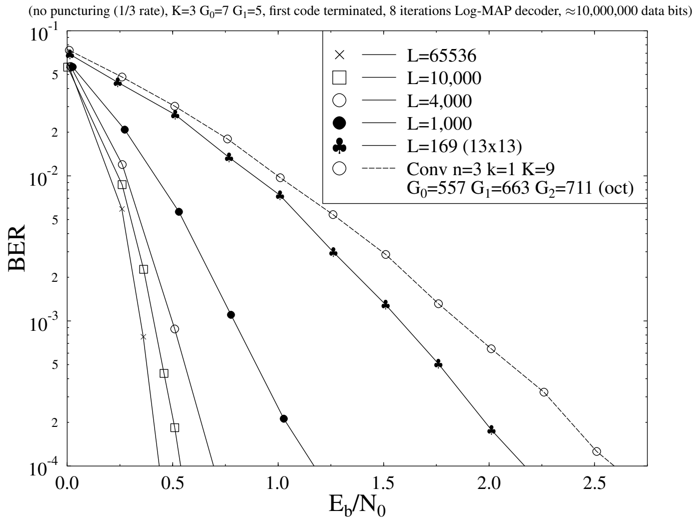


Figure 4.26: Effect of Frame-Length on BER Performance of $\frac{1}{3}$ Rate Turbo Coding. All Interleavers except $L = 169$ Block Interleaver Use Random Interleavers. Other Parameters as in Table 4.5.

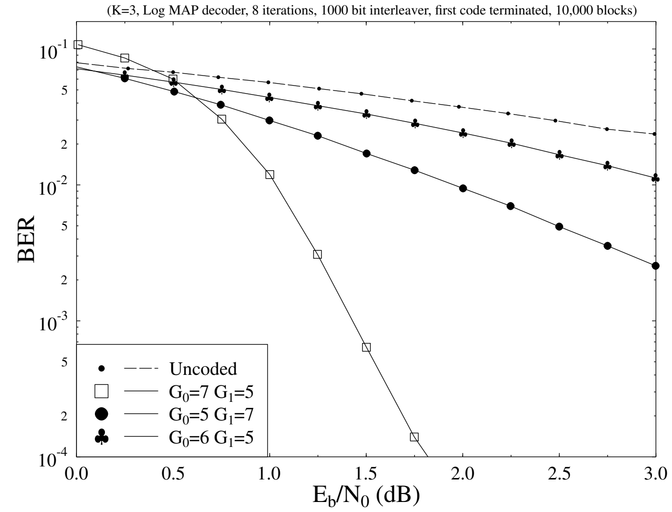


Figure 4.27: Effect of Generator Polynomials on BER Performance of Turbo Coding. Other Parameters as in Table 4.5.

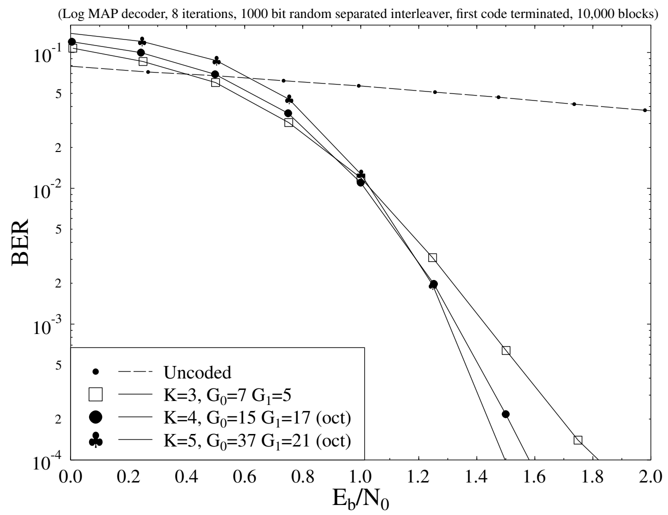


Figure 4.28: Effect of Constraint Length on the BER Performance of Turbo Coding. Other Parameters as in Table 4.5.

when the effect of interleaving is considered these generator polynomials do not necessarily lead to the best minimum free distance for turbo codes. Figure 4.27 shows the huge difference in performance that can result from different generator polynomials being used in the component codes. The other parameters used in these simulations were the same as detailed above in Table 4.5.

Most of the results provided in this chapter were obtained using constraint length three component codes. For these codes we have used the optimum generator polynomials in terms of maximising the minimum free distance of the component convolutional codes, ie 7 and 5 in octal representation. These generator polynomials were also used for constraint length 3 turbo coding by Hagenauer et al in [68] and Jung in [73]. It can be seen from Figure 4.27 that the order of these generator polynomials is important - the value 7 should be used for the feedback generator polynomial in Figure 4.1 (denoted in our work by G_0). If G_0 and G_1 are swapped round, the performance of a convolutional code (both regular and recursive systematic codes) would be unaffected, but for turbo codes this gives a significant degradation in performance.

The effect of increasing the constraint length of the component codes used in turbo codes is shown in Figure 4.28. For the constraint length four turbo code we again used the optimum minimum free distance generator polynomials for the component codes (15 and 17 in octal, 13 and 15 in decimal representations). The resulting turbo code gives an improvement of about 0.25 dB at a BER of 10^{-4} over the K=3 curve.

For the constraint length 5 turbo code we used the octal generator polynomials 37 and 21 (31 and 17 in decimal), which were the polynomials used by Berrou et al [21] in the

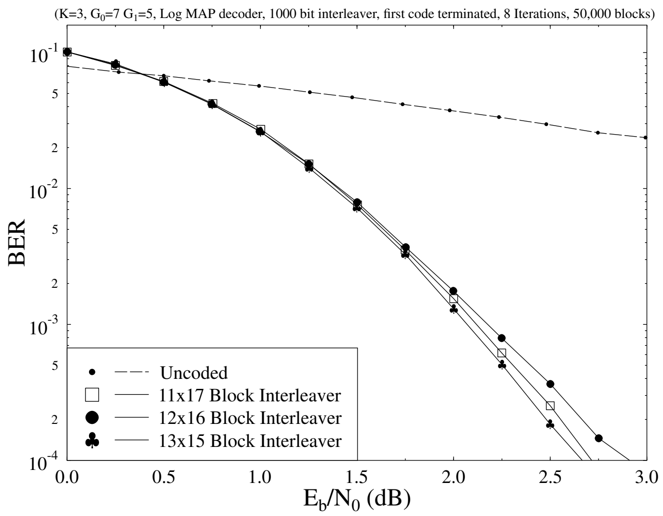


Figure 4.29: Effect of Block Interleaver Choice for $L \approx 190$ Frame-Length Turbo Codes. Other Parameters as in Table 4.5.

original paper on turbo coding. We also tried using the octal generator polynomials 23 and 35 (19 and 29), which are again the optimum minimum free distance generator polynomials for the component codes, as suggested by Hagenauer et al. in [68]. We found that these generator polynomials gave almost identical results to those used by Berrou et al. It can be seen from Figure 4.28 that increasing the constraint length of the turbo code does improve its performance, with the K=4 code performing about 0.25 dB better than the K=3 code at a BER of 10^{-4}, and the K=5 code giving a further improvement of about 0.1 dB. However, these improvements are provided at the cost of approximately doubling or quadrupling the decoding complexity. Therefore, unless otherwise stated, we have used component codes with a constraint length of 3 in our work. Let us now focus on the effects of the interleaver used within the turbo encoder and decoder.

4.4.6 Effect of the Interleaver

It is well known that the interleaver used in turbo codes has a vital influence on the performance of the code. The interleaver design together with the generator polynomials used in the component codes, and the puncturing used at the encoder, have a dramatic affect on the free distance of the resultant turbo code. Several algorithms have been proposed, for example in References [126] and [72], that attempt to choose good interleavers based on maximising the minimum free distance of the code. However this process is complex, and the resultant interleavers are not necessarily optimum. For example, in [132] random interleavers designed using the technique given in [72] are compared to a 12x16 block interleaver, and the "opti-

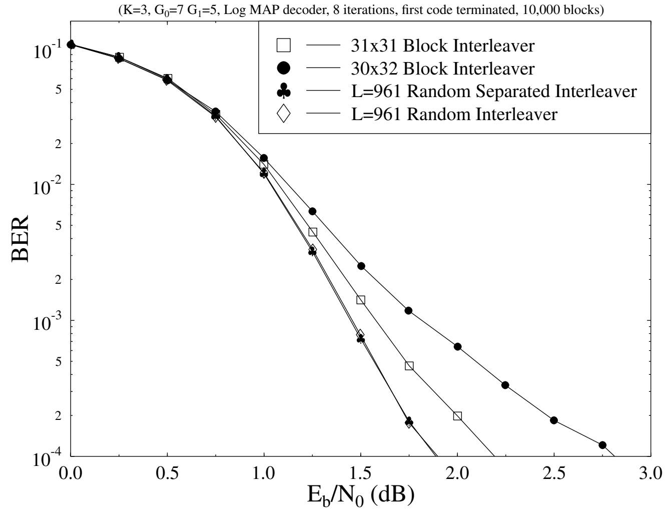


Figure 4.30: Effect of Interleaver Choice for $L \approx 961$ Frame-Length Turbo Codes. Other Parameters as in Table 4.5.

mised" interleavers are found to perform worse than the block interleaver.

In [75] a simple technique for designing good interleavers, which is referred to as "odd-even separation" is proposed. With alternate puncturing of the parity bits from each of the component codes, which is the puncturing most often used, if an interleaver is designed so that the odd and even input bits are kept separate, then it can be shown that one (and only one) parity bit associated with each information bit will be left unpunctured. This is preferable to the more general situation, where some information bits will have their parity bits from both component codes transmitted, whereas others will have neither of their parity bits transmitted.

A convenient way of achieving odd-even separation in the interleaver is to use a block interleaver with an odd number of rows and columns [75]. The benefits of using an odd number of rows and columns with a block interleaver can be seen in Figure 4.29. This shows a comparison between turbo coders using several block interleavers with frame-lengths of approximately 190 bits. The 12x16 block interleaver, proposed for short frame transmission systems in [132] and used by the same authors in other papers such as [73,74,133], clearly has a somewhat lower performance than the other block interleavers, which use an odd number of rows and columns. It is also interesting to note that of the two block interleavers with an odd number of rows and columns, the interleaver which is closer to being square (ie the 13x15 interleaver) performs better than the more rectangular 11x17 interleaver.

We also attempted using random interleavers of various frame-lengths. The effect of the interleaver choice for a turbo coding system with a frame-length of approximately 960 bits is shown in Figure 4.30. It can be seen from this figure that, as was the case with the codes with frame-lengths around 192 bits shown in Figure 4.29, the block interleaver with an

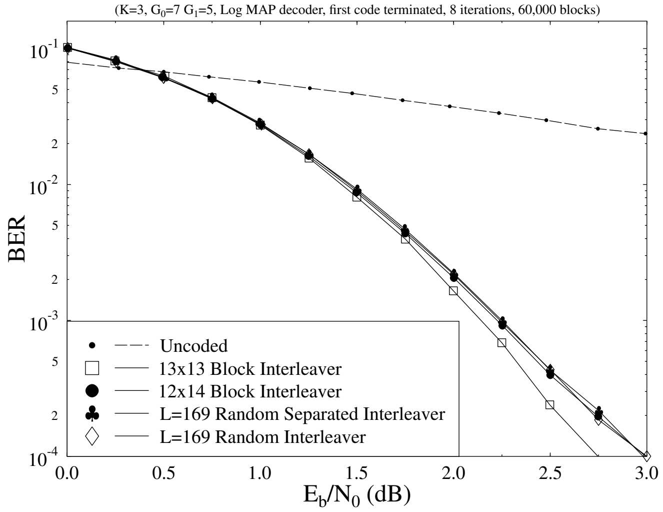


Figure 4.31: Effect of Interleaver Choice for $L \approx 169$ Frame-Length Turbo Codes. Other Parameters as in Table 4.5.

odd number of rows and columns (the 31x31 interleaver) performs significantly better than the interleaver with an even number of rows and columns (the 30x32 interleaver). However both of these interleavers are outperformed by the two random interleavers. In the "random separated" interleaver odd-even separation, as proposed by Barbulescu and Pietrobon [75], is used. This interleaver performs very slightly better than the other random interleaver, which does not use odd-even separation. However the effect of odd-even separation is much less significant for the random interleavers than it is for the block interleavers.

Similar curves are shown in Figure 4.31 for turbo coding schemes with approximately 169 bits per frame. It can be seen again that the scheme using block interleaving with odd-even separation (ie the 13x13 interleaver) performs better than the the scheme using block interleaving without odd-even separation (ie the 12x14 interleaver). However for this short frame-length system the two random interleavers perform worse than the best block interleaver. From our results it appears that although random interleavers give the best performance for turbo codes with long frame-lengths, for short frame-length systems the best performance is given using a block interleaver with an odd number of rows and columns.

When puncturing is not used, and we have a third rate code, the benefit of using odd-even separation with block interleavers, ie using block interleavers with an odd number of rows and columns, disappears. This can be seen from Figure 4.32, which compares the performance of a turbo code with no puncturing using three different interleavers, all with a length of approximately 169 bits. As in the case of the half-rate turbo codes using puncturing in Figure 4.31, for a small frame length, such as 169 bits, the best performance is given by using a block rather than a random interleaver. However it can be seen from Figure 4.32

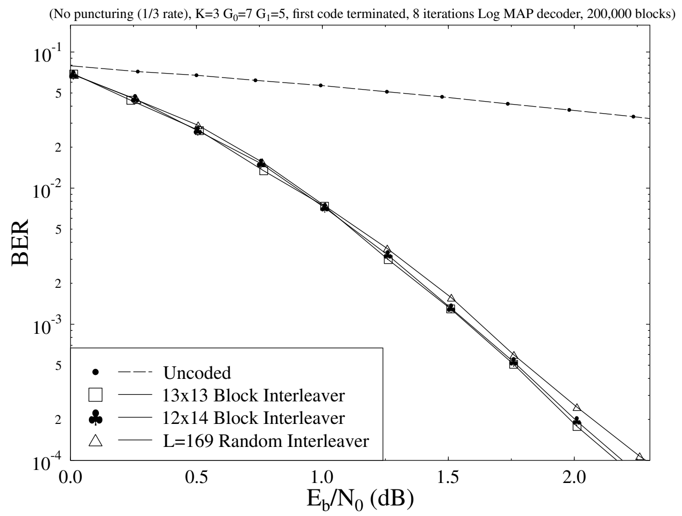


Figure 4.32: Effect of Interleaver Choice for Third-Rate $L \approx 169$ Frame-Length Turbo Codes. Other Parameters as in Table 4.5.

that, unlike for half-rate codes, for turbo codes without puncturing there is little difference between the block interleavers with and without odd-even separation, ie between the 13x13 and 12x14 interleavers.

In [134] Herzberg suggests that a "reverse block" interleaver, ie a block interleaver in which the output bits are read from the block in the reverse order relative to an ordinary block interleaver, gives an improved performance over ordinary block interleavers. He also suggests that for high SNRs, and hence for low BERs, reverse block interleavers having a short frame-length give a better performance, than random interleavers having a significantly higher frame length. However, as can be seen from Figure 4.33, which portrays the performance of ordinary and reverse block interleavers for various frame-lengths, we found very little difference between the performances of block and reverse block interleavers. One difference between our results and those in [134] is that we have used punctured half-rate turbo codes, whereas Herzberg used turbo codes without puncturing. However we found that even with third-rate turbo codes using no puncturing, and using 14x14 interleavers as Herzberg did, the performance of block and reverse block interleavers were almost identical. It appears in [134] that for turbo codes with long random interleavers, and with an ordinary block interleaver, Herzberg used the generator polynomials $G_0 = 5$ and $G_1 = 7$, whereas for the reverse block interleaver he used the generator polynomials $G_0 = 7$ and $G_1 = 5$. The generator polynomials $G_0 = 5$ and $G_1 = 7$ were used so that the performance of turbo codes with long random interleavers could be approximated using the Union bound and the error coefficients calculated by Benedetto and Montorosi in [66] for these generator polynomials. However, as was seen in Figure 4.27, these generator polynomials give a significantly worse

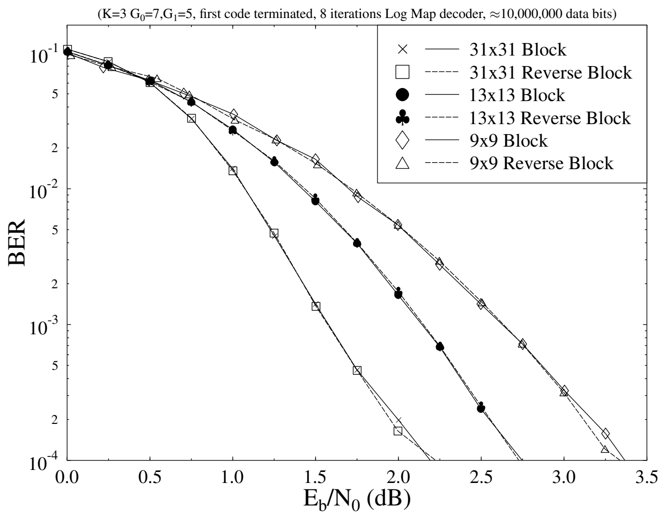


Figure 4.33: BER Performance of Block and Reverse Block Interleavers. Other Parameters as in Table 4.5.

performance that the generator polynomials $G_0 = 7$ and $G_1 = 5$ we have used for most of our simulations, and Herzberg used with his reverse block interleaver. Thus it appears that the reason Herzberg found such promising results for the reverse block interleaver was not because of this interleaver's superiority, but because of the inferiority of the generator polynomials he used with random and block interleavers.

Let us now focus our attention on the effect of the estimation of the channel reliability measure L_c.

4.4.7 Effect of Estimating the Channel Reliability Value L_c

In the previous section we highlighted, how the component decoders of an iterative turbo decoder interacted using soft inputs, the channel inputs $L_c y_{kl}$ as well as the a-priori inputs $L(u_k)$, and providing the a-posteriori LLRs $L(u_k|\underline{y})$ as soft outputs. In the MAP and Log-MAP algorithms the channel inputs and a-priori information are used to calculate the transition probabilities $\gamma_k(\grave{s}, s)$ that are then used to recursively calculate the $\alpha_k(s)$ and $\beta_k(s)$ and finally the a-posteriori LLRs $L(u_k|\underline{y})$. Similarly, in the Max-Log-MAP and the SOVA algorithms the channel and a-priori information values are used to update metrics, which are then used to give the soft output a-posteriori LLRs. In this section we investigate how important an accurate estimate of the channel reliability measure L_c is to the good performance of an iterative turbo decoder.

Figure 4.34 shows the performance of iterative turbo decoders using three different component decoders – the Log-MAP, Max-Log-MAP and the SOVA algorithms. For each com-

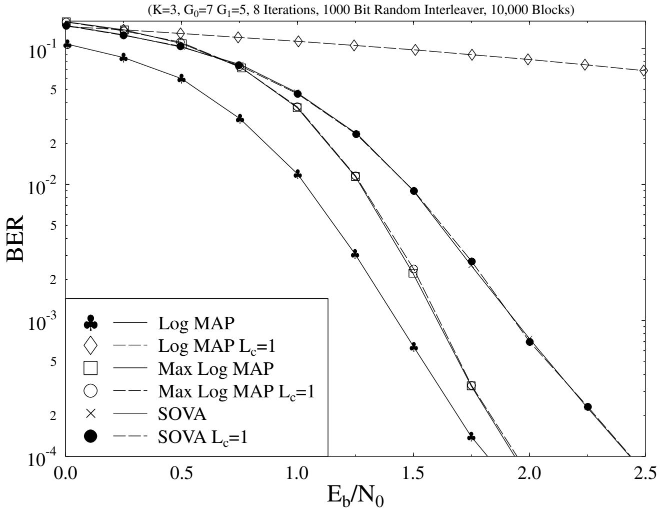


Figure 4.34: Effect of Using Incorrect Channel Reliability Measures L_c on an Iterative Turbo Decoder Using Various Component Decoders. Other Parameters as in Table 4.5.

ponent decoder type the solid line shows the performance of the codec, when the channel reliability value L_c is calculated exactly using the known channel SNR. For all our previous results we assumed that the channel SNR, and hence the correct value of L_c, would be known at the decoder. The dashed curves in Figure 4.34 show how the three component decoders perform when L_c is not known. For these curves the value of $L_c = 1$, which corresponds to a value of E_b/N_0 of -3 dB, was used at all channel SNRs. It can be seen from Figure 4.34 that for the SOVA and the Max-Log-MAP algorithms the turbo decoder performs equally well whether or not the correct value of L_c is known. However for the Log-MAP algorithm the performance of the iterative turbo decoder is drastically affected by the value of L_c used.

The reason for these effects can be understood by considering the different operation of the three algorithms, as it was described in Section 4.3. In the SOVA algorithm the channel values $L_c y_{kl}$ are used to recursively calculate the metrics M_n using Equation 4.63. The metric M_n for a state $S_k = s$ along a path is given by the metric M_{n-1} for the previous state along the path, added to an a-priori information term and to a cross-correlation term between the expected and the received channel values, x_{kl} and y_{kl}. The channel reliability measure L_c is used to scale this cross-correlation between the received and expected channel values. Then, once the Maximum-Likelihood (ML) path has been identified, the soft outputs from the algorithm are given by the minimum metric difference between the ML path and the paths merging with this ML path, as seen in Equation 4.68. When we use an incorrect value of L_c, effectively we are scaling the inputs to the component decoders by a factor. For instance, if we simply use $L_c = 1$, then we scale the channel input values by a factor of one over the correct value of L_c. In the SOVA algorithm this has the effect of scaling all the metrics M_n by the same factor, and as the a-posteriori LLRs from the algorithm are given by the difference between metrics for different paths, these output LLRs are also scaled by the same factor. Which path is chosen by the algorithm as the ML path will be unaffected by this scaling of the metrics, and so the hard decisions given by the algorithm will be unaffected by using the incorrect value of L_c.

Consider now the operation of the SOVA algorithm as a component decoder within an iterative turbo decoder. Assuming that no a-priori information about the values of the bits is available to the iterative turbo decoder, the first component decoder in the first iteration takes channel values only. The a-priori information values $L(u_k)$ are set equal to zero. If the correct value of the channel reliability measure for the channel SNR used is L_c, but an incorrect value of $\hat{L}_c$ is used instead, then effectively the channel input values will have been scaled by a factor X, where

$$X = \frac{\hat{L}_c}{L_c}. \tag{4.73}$$

The first component decoder will process these scaled channel values, and give soft output LLRs $L(u_k|\underline{y})$. From our discussions above these soft outputs, and hence the extrinsic information $L_e(u_k)$ derived from them using Equation 4.71, will be equal to the correct soft outputs scaled by X. Next the second component decoder will take a-priori information, equal to the interleaved extrinsic information $L_e(u_k)$ from the first decoder, and the channel inputs and will use these values to calculate its soft outputs. Both the channel values $\hat{L}_c y_{kl}$ and the a-priori information $L(u_k)$ will have been scaled by X relative to their values if the correct L_c had been used, and so again all the metrics used in the SOVA algorithm will be scaled by X, and the soft outputs from this decoder will simply be the correct soft outputs scaled by the factor X. Hence we see that, because of the linearity in the SOVA algorithm, the effect of using an incorrect value of the channel reliability measure is that the output LLRs from the decoder are scaled by a constant factor. The relative importance of the two inputs to the decoder, ie the a-priori information and the channel information, will not change, since the LLRs for both these sources of information will be scaled by the same factor. The soft outputs from the final component decoder in the final iteration will have the same sign as those that would have been calculated using the correct value of L_c, and will merely have been scaled by X. Hence the hard outputs from an iterative turbo decoder using the SOVA algorithm are unaffected by the value of the channel reliability value L_c used, as can be seen from Figure 4.34.

The same linearity that is present in the SOVA algorithm is also found in the Max-Log-MAP algorithm. Instead of one metric two are calculated, but again only simple additions of the cross-correlation of the expected and received channel values are used. Hence, if an incorrect value of the channel reliability value is used, all the metrics are simply scaled by a factor X. As in the SOVA algorithm, the soft outputs are given by the differences in metrics between different paths, and so the same argument as was used above for the SOVA algorithm will also apply to the Max-Log-MAP algorithm – if the input channel values are scaled by X, then the soft outputs of all the component decoders will also be scaled by X, and the final hard decisions given by the turbo decoder will be unaffected.

Let us now consider the Log-MAP algorithm. This is identical to the Max-Log-MAP algorithm, except for a correction factor $f_c(x) = \ln(1 + \mathrm{e}^{-x})$ used in the calculation of the forward and backward metrics $A_k(s)$ and $B_k(s)$ and the soft output LLRs. The function

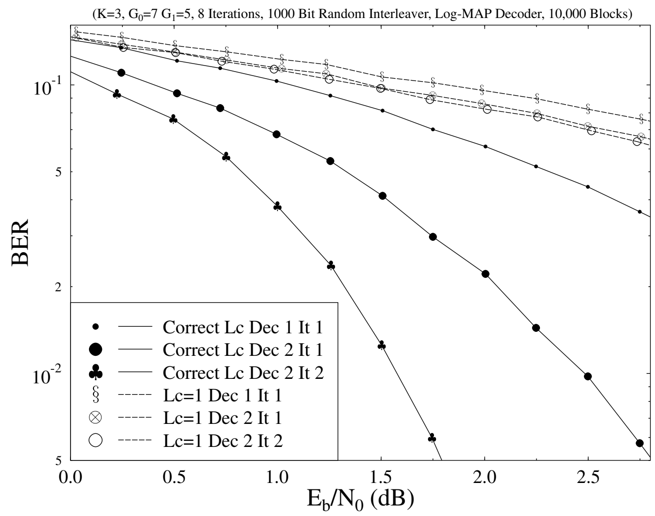


Figure 4.35: BER Within an Iterative Turbo Decoder Using the Log-MAP Decoder and the Parameters of Table 4.5, with Correct and Incorrect Channel Reliability Values L_c.

$f_c(x)$ is non-linear – it decreases asymptotically towards zero as x increases. Hence the linearity that is present in the Max-Log-MAP and SOVA algorithms is not present in the Log-MAP algorithm. We found that the effect of this non-linearity is two-fold if an incorrect value of the channel reliability value is used. Firstly, even when only the channel values are used to calculate the soft outputs from the algorithm, the component decoder makes more hard decision errors than if the correct value of L_c were used. This is the case for the first component decoder in the first iteration, where the a-priori information values $L(u_k)$ are assumed to be equal to zero. Figure 4.35 shows the performance of an iterative turbo decoder using the Log-MAP algorithm after the first component decoder in the first iteration, denoted by "Dec 1 It 1" in the key, when both the correct value of the channel reliability measure L_c, and an incorrect value of $L_c = 1$, are used. It can be seen from this figure that when the channel reliability value $L_c = 1$ is used the BER for the first component decoder in the first iteration is significantly increased.

As well as making more hard decision errors, if an incorrect value of the channel reliability measure is used with the Log-MAP algorithm, then the extrinsic information derived from the soft output values from the first component decoder have incorrect amplitudes. This means that the a-priori information that is used by the second decoder in the first iteration, and by both decoders in subsequent iterations, will have incorrect amplitudes relative to the soft channel inputs. In an iterative turbo decoder the feeding of a-priori information from one component decoder to the next allows a rapid decrease in the BER of the decoder as the number of iterations increases, as was seen in Figure 4.20. However, when the incorrect value of L_c is used in an iterative turbo decoder employing the Log-MAP algorithm, due to

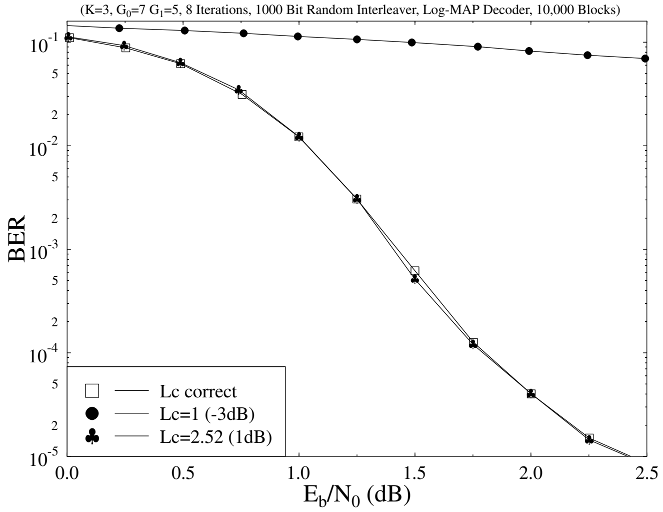


Figure 4.36: Turbo Decoder Performance Using the Log-MAP Algorithm and the Parameters of Table 4.5 with a Constant Estimated Channel Reliability Measure L_c.

the incorrect scaling of the a-priori information relative to the channel inputs, no such rapid fall in the BER with the number of iterations occurs. Infact the performance of the decoder is largely unaffected by the number of iterations used. This can be seen from Figure 4.35, which shows the BER from the second decoder after one and two iterations. A significant performance improvement can be observed between the first and second iterations, when the correct value of L_c is used. By contrast, when the value of $L_c = 1$ is used, there is only a marginal improvement between the first and second iteration.

In reference [135] Summers and Wilson consider the degradation in the performance of an iterative turbo decoder using the MAP algorithm when the channel SNR is not correctly estimated. As explained in Section 4.3, the MAP and Log-MAP algorithm give identical outputs and hence our analysis of the Log-MAP algorithm also applies to the MAP algorithm. In [135] the authors propose a method for blind estimation of the channel SNR, using the ratio of the average squared received channel value to the square of the average of the magnitudes of the received channel values. For a one-third rate code and a block length of 420 data bits (so 1260 coded bits are transmitted) it is shown that the SNR derived using this method rarely differs from the true SNR by more than 3 dB. It is also shown that using these estimated SNRs to derive a channel reliability measure gives a turbo decoder performance using the MAP algorithm almost identical to that given using channel reliability measures derived from the true SNR.

Our work presented here shows that if the Max-Log-MAP or SOVA algorithms are used as the component decoders, then no such SNR estimation is necessary for a turbo decoder. If the MAP, or equivalently the Log-MAP, algorithm is used then, as can be seen from Figure 4.34, if a very inaccurate value of L_c is used the turbo decoder performance will be drastically affected. However we have found that the value of L_c used does not have to be very close to the true value for a good BER performance to be obtained. Figure 4.36 compares the performance of a turbo decoder using the Log-MAP algorithm with the correct value of L_c, to that of a scheme which uses a constant value of $L_c = 2.52$. This corresponds to a value of E_b/N_0 of 1 dB. It can be seen that using this estimated value of L_c gives a performance virtually identical to that given with the correct value of L_c for values of E_b/N_0 from 0 to 2.5 dB, or BERs from 10^{-1} to 10^{-5}. Hence, even when using the Log-MAP algorithm, only a rough estimate of L_c is needed.

Having investigated the performance of turbo codes when used with BPSK modulation over AWGN channels, in the next section we discuss the use of both convolutional and turbo codes with higher order modulation schemes. This allows turbo codes to be used in systems which are both bandwidth and power efficient.

4.5 Turbo Coding Performance Over Rayleigh Channels

4.5.1 Introduction

In the previous sections we have discussed the performance of turbo coding in conjunction with various modulation constellations over AWGN channels. We now move on to an investigation of using turbo coding over fading channels. In this work we have assumed Rayleigh fading, and that the receiver has exact estimates of the fading amplitude and phase inflicted by the channel. This assumption is justified as several techniques, for example Pilot Symbol Assisted Modulation (PSAM) [136], are available which provide practical CSI recovery with performance very close to that assuming perfect recovery. In Section 4.5.2 we look at the performance of various turbo codes over Rayleigh fading channels which are perfectly interleaved. Then in Section 4.5.3 we consider the effects correlations experienced in real Rayleigh fading channels have, and evaluate the performance of turbo codes in such channels.

4.5.2 Performance Over Perfectly Interleaved Narrow-Band Rayleigh Channels

Figure 4.37 shows the performance of three turbo codes with different frame-lengths L over a perfectly interleaved Rayleigh fading channel using BPSK modulation. All the turbo codecs use two $K = 3$ RSC component codes with generator polynomials $G_0 = 7$ and $G_1 = 5$. At the decoder eight iterations of the Log-MAP decoder are used. Also shown in Figure 4.37 is the performance of a constraint length $K = 9$ convolutional code which, as explained earlier, has a decoder complexity which is similar or slightly higher than that of the turbo decoder. It can be seen that the turbo codes with frame-lengths of $L = 1000$ or $L = 10,000$ give a significant increase in performance over the convolutional code. Even the turbo code with a short frame length of 169 bits outperforms the convolutional code for BERs below 10^{-3}.

Comparing the performance of the $L = 169$, $L = 1000$ and $L = 10,000$ turbo codes in Figure 4.37 to those in Figure 4.25 for the same codes over an AWGN channel, we see that the perfectly interleaved fading of the received channel values degrades the BER performance

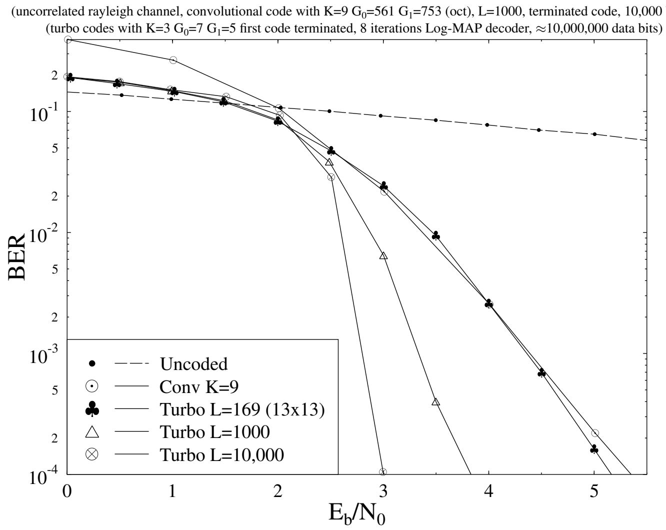


Figure 4.37: BER Performance of Turbo Codes with Different Frame-Lengths L over Perfectly Interleaved Rayleigh Fading Channels. Other Turbo Codec Parameters as in Table 4.5.

of the code by around 2 dB at a BER of 10^{-4}, with a larger degradation for the shorter frame-length codes.

The short frame-length $L = 169$ turbo codec in Figure 4.37 uses a 13x13 block interleaver. We found in Section 4.4.6 that when communicating over a Gaussian channel, for a half-rate turbo code having short frame-lengths, a block interleaver with an odd number of rows and columns should be used. Figure 4.38 shows the effect of using different interleavers, all with a frame-length of approximately 169 bits, on the BER performance of a turbo codec over a perfectly interleavered Rayleigh channel. It can be seen from this figure that again for the short frame-length of 169 bits the best performance is given by a block interleaver with an odd number of rows and columns. This block interleaver acheives odd-even separation [75] so that each data bit has one, and only one, of the two parity bits associated with it transmitted. The 12x14 block interleaver shown in Figure 4.38 does not give odd-even separation, and so even through its frame-length is almost identical to that of the 13x13 block interleaver (168 rather than 169 bits) it performs almost 1 dB worse than the 13x13 block interleaver at a BER of 10^{-4}. The two random interleavers shown in Figure 4.38 also perform worse than the 13x13 block interleaver, although the random interleaver with odd-even separation does perform better than the non-separated interleaver.

Figure 4.39 shows how the choice of the component decoders used at the turbo decoder affects the performance of the codec over a perfectly interleavered Rayleigh channel. It can be seen that again the Log-MAP decoder gives the best performance, followed by the Max-Log-MAP decoder with the SOVA decoder, the simplest of the three, giving the worst performance. It can also be seen that the differences in performances between the different decoders

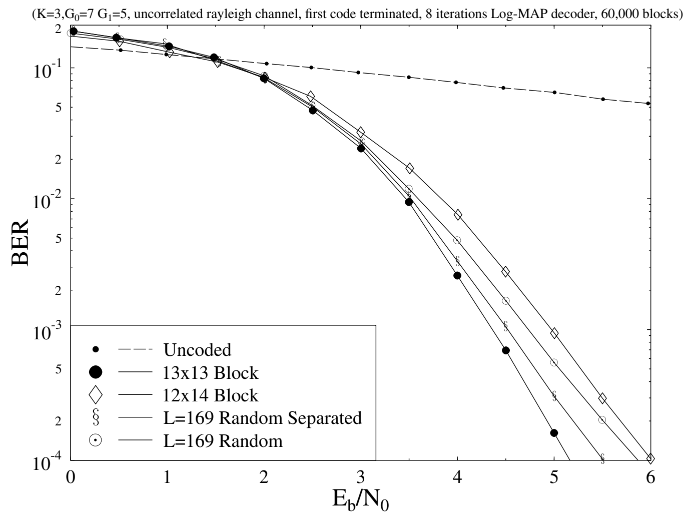


Figure 4.38: BER Performance of Turbo Codes with Different Interleavers over Perfectly Interleaved Rayleigh Fading Channels. Other Turbo Codec Parameters as in Table 4.5.

are slightly larger than they were over an AWGN channel - the Max-Log-MAP decoder performs about 0.2 dB worse than the Log-MAP decoder, and the SOVA decoder is about 0.8 dB worse than the Log-MAP decoder.

Figure 4.40 shows the effect of puncturing on a turbo code with frame-length $L = 1000$ over the perfectly interleaved Rayleigh channel. In Figure 4.21 we saw that over the AWGN channel the third-rate code outperformed the half-rate code by about 0.6 dB in terms of E_b/N_0. We see from Figure 4.40 that again for the perfectly interleaved Rayleigh channel the difference in performance is bigger – about 1.5 dB in terms of E_b/N_0 or about 3.25 dB in terms of channel SNR.

4.5.3 Performance Over Correlated Narrow-Band Rayleigh Channels

Figure 4.41 shows the performance of a half-rate turbo coding system with $L = 1000$ over various Rayleigh fading channels. It can be seen that by far the best performance is achieved over the perfectly interleaved Rayleigh channel, where there is no correlation between sucessive fading values. The narrowband Rayleigh channel exhibits a normalised Doppler frequency of $f_d = 2.44 * 10^{-4}$, since we assumed a carrier frequency of 1.9 GHz, a symbol rate of 360 Kbaud and a vehicular speed of 50 km/hr. It can be seen that the turbo codes give a significant coding gain over the uncoded BER results even for this channel. We found that for Rayleigh fading channels exhibiting faster fading, ie a higher normalised Doppler frequency, the coding gain increased. Furthermore, it can be seen that interleaving the output bits of the turbo encoder before transmission over the Rayleigh fading channel improves the

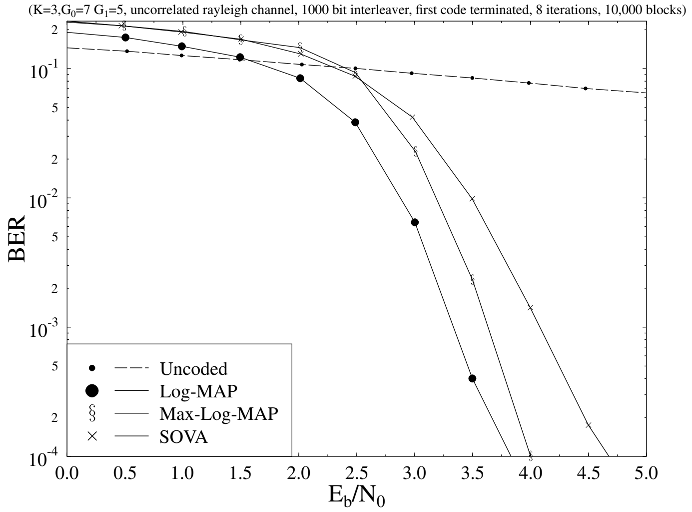


Figure 4.39: BER Performance of Turbo Codes with $L = 1000$ Using Different Component Decoders over Perfectly Interleaved Rayleigh Fading Channels. Other Turbo Codec Parameters as in Table 4.5.

performance for the narrowband system by about 2.5 dB at a BER of 10^{-4}. This gain was acheived by merely interleaving over the 2000-bit length of the output block of the turbo encoder. Higher interleaving gains can be achieved at the cost of extra delay, by interleaving over longer periods. Near-perfect interleaving over a significantly longer period would give the performance indicated by the uncorrelated Rayleigh curve in Figure 4.41.

Also shown in Figure 4.41 is the performance of our turbo codec over an Orthogonal Frequency Division Multiplexing (OFDM) system with Rayleigh fading. The effects of OFDM with turbo coding will be explored in the next section, but it can be seen that the OFDM gives a coded performance much closer to that for the perfectly interleaved Rayleigh channel. Again interleaving over the 2000 output bits from the turbo encoder improves the coded performance.

4.6 Summary and Conclusions

In this chapter we have characterised the performance of turbo coding schemes using both BPSK and QPSK modulation constellations, when communicating over both AWGN and Rayleigh channels. As expected, the turbo codes have been shown to perform significantly better than convolutional codes. We have demonstrated the effects of the various decoding algorithms, the constraint length and generator polynomials of the constituent codes, as well as the influence of the transmission frame length on the achievable performance. Furthermore,

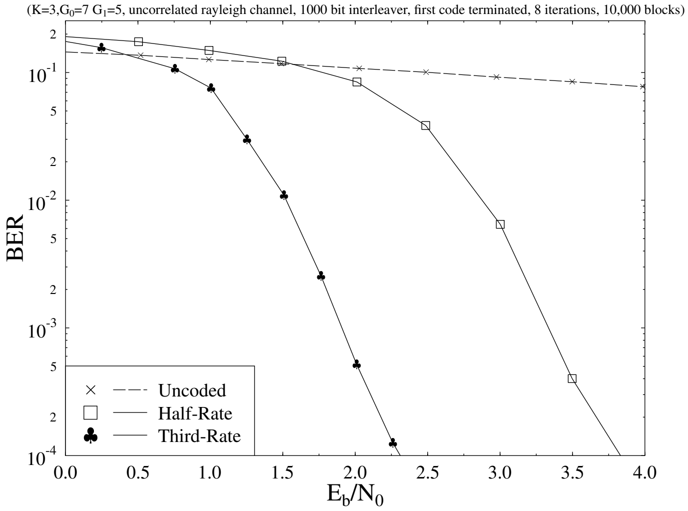


Figure 4.40: The BER Performance Comparison Between One-Third and One-Half Rate of Turbo Codes over Perfectly Interleaved Rayleigh Fading Channels. Other Turbo Codec Parameters as in Table 4.5.

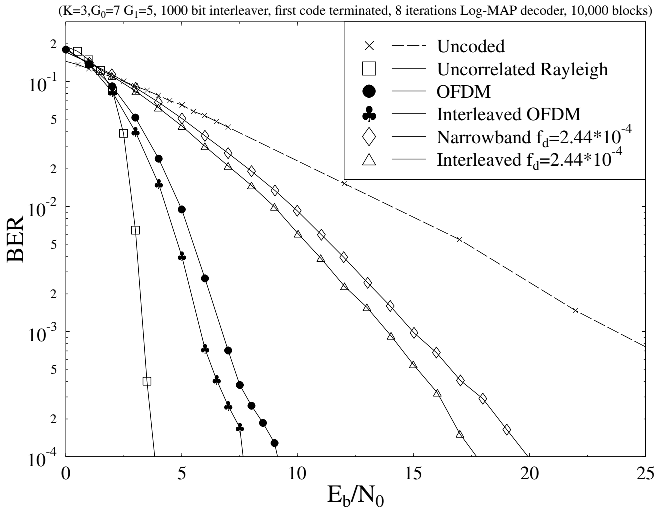


Figure 4.41: Performance of Turbo Coding over Rayleigh Fading Channels. Turbo Codec Parameters as in Table 4.5.

based on our detailed investigations we have demosntrated the importance of the choice of the interleaver in the context of turbo codes. More explicitly, we have reached the following conclusions regarding the choice of interleavers:

- When block interleavers are used in conjunction with half-rate codes, an odd number of rows and columns should be used.
- For long frame length systems random interleavers perform better than block interleavers, but for shorter frame length systems, such as those that might be used for speech transmission, block interleavers perform better.

Finally in Section 4.5 we provided performance results obtained, when using turbo codes in conjunction with BPSK and QPSK modulation for transmissions over Rayleigh fading channels.

Having explored the structure and decoding of convolutional constituent code based turbo codes, in Chapter 5 we elaborate further on the relationship between conventional convolutional codes and turbo convolutional codes and highlight the relationship between their trellis structure.

Part III

Coded Modulation: TCM, TTCM, BICM, BICM-ID

Part IV

Space-Time Block and Space-Time Trellis Coding

Chapter 9

Space-Time Block Codes

9.1 Introduction[1]

In this chapter, as well as in the forthcoming one we will mainly concentrate our attention on systems designed for transmissions over wireless links, where the channel errors tend to occur in bursts, rather than randomly distributed. Hence here we assume a basic background in the area of mobile communication channel properties. We also assume that the reader is well-versed in channel coding in general, since a range of previously studied channel coding schemes will be combined with various space-time coding arrangements in Chapters 9 and 10.

The third generation (3G) mobile communications standards [266] are expected to provide a wide range of bearer services, spanning from voice to high-rate data services, supporting rates of at least 144 kb/s in vehicular, 384 kb/s in outdoor-to-indoor and 2 Mb/s in indoor as well as picocellular applications [266].

In an effort to support such high rates, the bit/symbol capacity of band-limited wireless channels can be increased by employing multiple antennas [267]. The classic approach is to use multiple antennas at the receiver and invoke Maximum Ratio Combining (MRC) [268–270] of the received signals for improving the performance. However, applying receiver diversity at the Mobile Stations (MS) increases their complexity. Hence receiver diversity techniques typically have been applied at the Base Stations (BS). The BSs provide services for many MSs and hence up-grading the BSs is economically viable. However, the drawback of this scheme is that it only provides diversity gain for the BSs' receivers.

In the past, different transmit diversity techniques have been introduced, in order to provide diversity gain for MSs by upgrading the BSs. These transmit diversity techniques can be classified into three main categories: schemes using information feedback [271, 272], arrangements invoking feedforward or training information [273–275] and blind schemes [276, 277]. Recently, Tarokh *et al.* proposed space-time trellis coding [77, 87, 278–281] by jointly designing the channel coding, modulation, transmit diversity and the optional receiver

[1]This chapter is based on T.H. Liew and L. Hanzo: Space-time Block Codes and Concatenated Channel Codes: A Historical Perspective and Comparative Study, Proc. of the IEEE, Febr. 2001

diversity scheme. The performance criteria for designing space-time trellis codes were derived in [77], under the assumption that the channel is fading slowly and that the fading is frequency nonselective. These advances were then also extended to fast fading channels. The encoding and decoding complexity of these *space-time trellis codes* is comparable to that of conventional trellis codes [53–55] often employed in practice over non-dispersive Gaussian channels.

Space-time trellis codes [77, 87, 278–281] perform extremely well at the cost of relatively high complexity. In addressing the issue of decoding complexity, Alamouti [78] discovered a remarkable scheme for transmissions using two transmit antennas. A simple decoding algorithm was also introduced by Alamouti [78], which can be generalised to an arbitrary number of receiver antennas. This scheme is significantly less complex, than space-time trellis coding using two transmitter antennas, although there is a loss in performance [79]. Despite the associated performance penalty, Alamouti's scheme is appealing in terms of its simplicity and performance. This proposal motivated Tarokh *et al.* [79, 80] to generalise Alamouti's scheme to an arbitrary number of transmitter antennas, leading to the concept of *space-time block codes*.

Intrigued by the decoding simplicity of the space-time block codes proposed in [78–80], in this chapter, we commence our discourse by detailing their encoding and decoding process. Subsequently, we investigate the performance of the space-time block codes over perfectly interleaved, non-dispersive Rayleigh fading channels. A system which consists of space-time block codes and different channel coders will be proposed. Finally, the performance and estimated complexity of the different systems will be compared and tabulated.

Following a rudimentary introduction to space-time block codes in Section 9.3 and to channel coded space-time codes in Section 9.4, the associated estimated complexity issues and memory requirements are addressed in Section 9.4.3. The bulk of this chapter is constituted by the performance study of various space-time and channel coded transceivers in Section 9.5. Our aim is firstly to identify a space-time code, channel code combination constituting a good engineering trade-off in terms of its effective throughput, BER performance and estimated complexity in Section 9.5.1. Specifically, the issue of bit-to-symbol mapping is addressed in the context of convolution codes and convolutional coding as well as Bose-Chaudhuri-Hocquenghem (BCH) coding based turbo codes in conjunction with an attractive unity-rate space-time code and multi-level modulation in Section 9.5.2. These schemes are also benchmarked against a range of powerful trellis coded modulation (TCM) and turbo trellis coded modulation (TTCM) schemes. Our conclusions concerning the merits of the various schemes are drawn in Section 9.5.4 in the context of their coding gain versus estimated complexity.

9.2 Background

In this section, we present a brief overview of space-time block codes by considering the classical Maximum Ratio Combining (MRC) technique [78, 104, 282]. The introduction of this classical technique is important, since at a later stage it will assist us in highlighting the philosophy of space-time block codes.

9.2.1 Maximum Ratio Combining

In conventional transmission systems we have a single transmitter, which transmits information to a single receiver. In Rayleigh fading channels the transmitted symbols experience severe magnitude fluctuation and phase rotation. In order to mitigate this problem, we can employ several receivers that receive replicas of the same transmitted symbol through independent fading paths. Even if a particular path is severely faded, we may still be able to recover a reliable estimate of the transmitted symbols through other propagation paths. However, at the station we have to combine the received symbols of the different propagation paths, which involves additional complexity. Again, the classical method often used in practice is referred to as the MRC technique [78, 104, 282].

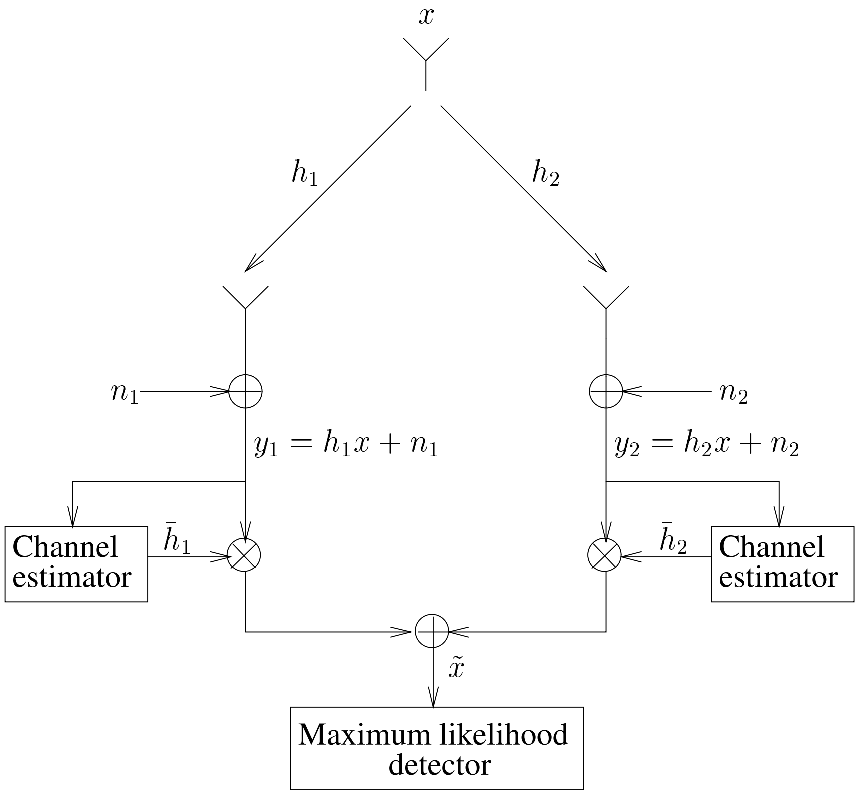


Figure 9.1: Baseband representation of the MRC technique using two receivers.

Figure 9.1 shows the baseband representation of the classical MRC technique in conjunction with two receivers. At a particular instant, a symbol x is transmitted. As we can see from the figure, the transmitted symbol x propagates through two different channels, namely h_1 and h_2. For simplicity, all channels are assumed to be constituted by a single propagation path and can be modelled as complex multiplicative distortion, which consists of a magnitude and phase response given as follows:

$$\begin{aligned} h_1 &= |h_1|e^{j\theta_1} && (9.1)\\ h_2 &= |h_2|e^{j\theta_2}\ , && (9.2) \end{aligned}$$

where $|h_1|$, $|h_2|$ are the fading magnitudes and θ_1, θ_2 are the phase values. Noise is added by each receiver, as shown in Figure 9.1. Hence, the resulting received baseband signals are:

$$\begin{aligned} y_1 &= h_1 x + n_1 && (9.3)\\ y_2 &= h_2 x + n_2\ , && (9.4) \end{aligned}$$

where n_1 and n_2 are complex noise samples. In matrix form this can be written as follows:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = x \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} + \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}\ . \tag{9.5}$$

Assuming that we have perfect channel information, i.e. a perfect channel estimator, the received signals y_1 and y_2 can be multiplied by the conjugate of the complex channel transfer functions $\bar{h}_1$ and $\bar{h}_2$, respectively, in order to remove the channel's effects. Then the corresponding signals are combined at the input of the maximum likelihood detector of Figure 9.1 as follows:

$$\begin{aligned} \tilde{x} &= \bar{h}_1 y_1 + \bar{h}_2 y_2 \\ &= \bar{h}_1 h_1 x + \bar{h}_1 n_1 + \bar{h}_2 h_2 x + \bar{h}_2 n_2 \\ &= \left(|h_1|^2 + |h_2|^2\right) x + \bar{h}_1 n_1 + \bar{h}_2 n_2\ . && (9.6) \end{aligned}$$

The combined signal $\tilde{x}$ is then passed to the maximum likelihood detector, as shown in Figure 9.1. Based on the Euclidean distances between the combined signal $\tilde{x}$ and all possible transmitted symbols, the most likely transmitted symbol is determined by the maximum likelihood detector. The simplified decision rule is based on choosing x_i if and only if

$$dist(\tilde{x}, x_i) \leq dist(\tilde{x}, x_j), \quad \forall i \neq j, \tag{9.7}$$

where $dist(A, B)$ is the Euclidean distance between signals A, B and the index j spans all possible transmitted signals. From Equation 9.7 we can see that maximum likelihood transmitted symbol is the one having the minimum Euclidean distance from the combined signal $\tilde{x}$.

9.3 Space-Time Block Codes

In the previous section we have briefly introduced the classic MRC technique. In this section we will present the basic principles of space-time block codes. In analogy to the MRC matrix formula of Equation 9.5, a space-time block code describing the relationship between the original transmitted signal x and the signal replicas artificially created at the transmitter for transmission over various diversity channels, is defined by an $n \times p$ dimensional transmission matrix. The entries of the matrix are constituted by linear combinations of the $k-$ary input symbols $x_1, x_2, ..., x_k$ and their conjugates. The $k-$ary input symbols x_i $i = 1...k$, are used to represent the information-bearing binary bits to be transmitted over the transmit diversity channels. In a signal constellation having 2^b constellation points, b number of binary bits are used to represent a symbol x_i. Hence, a block of $k \times b$ binary bits are entered into the space-time block encoder at a time and it is therefore referred to as a space-time block code. The number of transmitter antennas is p and n represents the number of time slots used to transmit k input symbols. Hence, a general form of the transmission matrix of a space-time block code is as follows:

$$\begin{pmatrix} g_{11} & g_{21} & \cdots & g_{p1} \\ g_{12} & g_{22} & \cdots & g_{p2} \\ . & . & . & . \\ . & . & . & . \\ g_{1n} & g_{2n} & \cdots & g_{pn} \end{pmatrix}, \tag{9.8}$$

where the entries g_{ij} represent linear combinations of the symbols $x_1, x_2, ..., x_k$ and their conjugates. More specifically, the entries g_{ij}, where $i = 1...p$, are transmitted simultaneously from transmit antennas $1, ..., p$ in each time slot $j = 1, ..., n$. For example, in time slot $j = 2$, signals $g_{12}, g_{22}, ..., g_{p2}$ are transmitted simultaneously from transmit antennas $Tx\ 1, Tx\ 2, ..., Tx\ p$. We can see in the transmission matrix defined in Equation 9.8 that encoding is carried out in both space and time; hence the term space-time coding.

The $n \times p$ transmission matrix in Equation 9.8 $-$ which defines the space-time block code $-$ is based on a complex generalised orthogonal design, as defined in [78–80]. Since there are k symbols transmitted over n time slots, the code rate of the space-time block code is given by:

$$R = k/n. \tag{9.9}$$

At the receiving end, we can have an arbitrary number of q receivers. It was shown in [78] that the associated diversity order is $p \times q$. A combining technique [78, 80] similar to MRC can be applied at the receiving end, which may be generalised to q number of receivers. At the current state-of-the-art the associated diversity channels are often assumed to be flat fading channels. A possible approach to satisfying this condition for high-rate transmissions over frequency-selective channels is to split the high-rate bit stream into a high number of low-rate streams transmitted over flat-fading subchannels This can be achieved with the aid of Orthogonal Frequency Division Multiplexing (OFDM) [206], It is also typically assumed that the complex fading envelope is constant over n consecutive time slots.

9.3.1 A Twin-Transmitter Based Space-Time Block Code

As mentioned above, the simplest form of space-time block codes was proposed by Alamouti in [78], which is a simple twin-transmitter based scheme associated with $p = 2$. The transmission matrix is defined as follows:

$$\mathbf{G}_2 = \begin{pmatrix} x_1 & x_2 \\ -\bar{x}_2 & \bar{x}_1 \end{pmatrix}. \tag{9.10}$$

We can see in the transmission matrix $\mathbf{G}_2$ that there are $p = 2$ (number of columns in the matrix $\mathbf{G}_2$) transmitters, $k = 2$ possible input symbols, namely x_1, x_2, and the code spans over $n = 2$ (number of rows in the matrix $\mathbf{G}_2$) time slots. Since $k = 2$ and $n = 2$, the code rate given by Equation 9.9 is unity. The associated encoding and transmission process is shown in Table 9.1. At any given time instant T, two signals are simultaneously transmitted

Time	antenna	
slot, T	Tx 1	Tx 2
1	x_1	x_2
2	$-\bar{x}_2$	$\bar{x}_1$

Table 9.1: The encoding and transmission process for the $\mathbf{G}_2$ space-time block code of Equation 9.10.

from the antennas Tx 1 and Tx 2. For example, in the first time slot associated with $T = 1$, signal x_1 is transmitted from antenna Tx 1 and *simultaneously* signal x_2 is transmitted from antenna Tx 2. In the next time slot corresponding to $T = 2$, signals $-\bar{x}_2$ and $\bar{x}_1$ (the conjugates of symbols x_1 and x_2) are simultaneously transmitted from antennas Tx 1 and Tx 2, respectively.

9.3.1.1 The Space-Time Code $\mathbf{G}_2$ Using One Receiver

Lets us now consider an example of encoding and decoding the $\mathbf{G}_2$ space-time block code of Equation 9.10 using one receiver. This example can be readily extended to an arbitrary number of receivers. In Figure 9.2 we show the baseband representation of a simple two-transmitter space-time block code, namely that of the $\mathbf{G}_2$ code seen in Equation 9.10 using one receiver. We can see from the figure that there are two transmitters, namely Tx 1 as well as Tx 2 and they transmit two signals simultaneously. As mentioned earlier, the complex fading envelope is assumed to be constant across the corresponding two consecutive time slots. Therefore, we can write

$$h_1 = h_1(T = 1) = h_1(T = 2) \tag{9.11}$$
$$h_2 = h_2(T = 1) = h_2(T = 2)\ . \tag{9.12}$$

Independent noise samples are added at the receiver in each time slot and hence the received signals can be expressed with the aid of Equation 9.10 as:

$$y_1 = h_1 x_1 + h_2 x_2 + n_1 \tag{9.13}$$
$$y_2 = -h_1 \bar{x}_2 + h_2 \bar{x}_1 + n_2\ , \tag{9.14}$$

where y_1 is the first received signal and y_2 is the second. Notice that the received signal y_1 consists of the transmitted signal x_1 and x_2, while y_2 of their conjugates. In order to determine the transmitted symbols, we have to extract the signals x_1 and x_2 from the received signals y_1 and y_2. Therefore, both signals y_1 and y_2 are passed to the combiner, as shown in Figure 9.2. In the combiner $-$ aided by the channel estimator, which provides perfect estimation of the diversity channels in this example $-$ simple signal processing is performed in order to separate the signals x_1 and x_2. Specifically, in order to extract the signal x_1, we

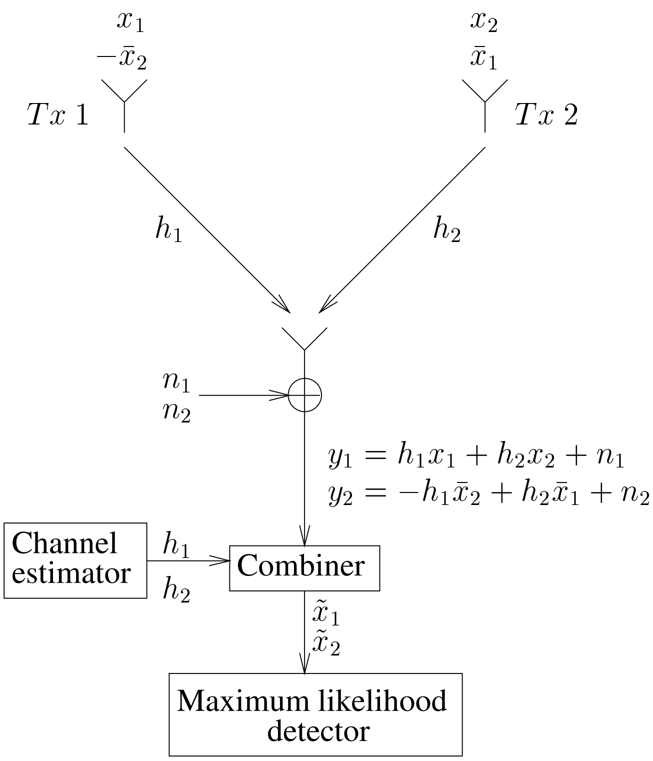


Figure 9.2: Baseband representation of the simple twin-transmitter space-time block code $\mathbf{G}_2$ of Equation 9.10 using one receiver.

combine the received signals y_1 and y_2 as follows:

$$\begin{aligned}
\tilde{x}_1 &= \bar{h}_1 y_1 + h_2 \bar{y}_2 \\
&= \bar{h}_1 h_1 x_1 + \bar{h}_1 h_2 x_2 + \bar{h}_1 n_1 - h_2 \bar{h}_1 x_2 + h_2 \bar{h}_2 \bar{x}_1 + h_2 \bar{n}_2 \\
&= \left(|h_1|^2 + |h_2|^2\right) x_1 + \bar{h}_1 n_1 + h_2 \bar{n}_2 \ .
\end{aligned} \tag{9.15}$$

Similarly, for signal x_2 we generate:

$$\begin{aligned}
\tilde{x}_2 &= \bar{h}_2 y_1 - h_1 \bar{y}_2 \\
&= \bar{h}_2 h_1 x_1 + \bar{h}_2 h_2 x_2 + \bar{h}_2 n_1 + h_1 \bar{h}_1 x_2 - h_1 \bar{h}_2 x_1 - h_1 \bar{n}_2 \\
&= \left(|h_1|^2 + |h_2|^2\right) x_2 + \bar{h}_2 n_1 - h_1 \bar{n}_2 \ .
\end{aligned} \tag{9.16}$$

Clearly, from Equation 9.15 and 9.16 we can see that we have separated the signals x_1 and x_2 by simple multiplications and additions. Due to the orthogonality of the space-time block code $\mathbf{G}_2$ in Equation 9.10 [79], the unwanted signal x_2 is cancelled out in Equation 9.15 and vice versa, signal x_1 is removed from Equation 9.16. Both signals $\tilde{x}_1$ and $\tilde{x}_2$ are then passed to the maximum likelihood detector of Figure 9.2, which applies Equation 9.7 to determine the most likely transmitted symbols.

¿From Equations 9.15 and 9.16 we can derive a simple rule of thumb for manipulating the received signal in order to extract a symbol x_i. For each received signal y_j, we would have a linear combination of the transmitted signals x_i convolved with the corresponding Channel Impulse Response (CIR) h_i. The non-dispersive CIR is assumed to be constituted by a single CIR tap corresponding to a complex multiplicative factor. The conjugate of the CIR $\bar{h}_i$ should be multiplied with the received signal y_j, if x_i is in the expression of the received signal y_j. However, if the conjugate of x_i, namely $\bar{x}_i$ is present in the expression, we should then multiply the CIR h_i with the conjugate of the received signal y_j, namely $\bar{y}_j$. The product should then be added to or subtracted from the rest, depending on the sign of the term in the expression of the received signal y_j.

9.3.1.2 The Space-Time Code $\mathbf{G}_2$ Using Two Receivers

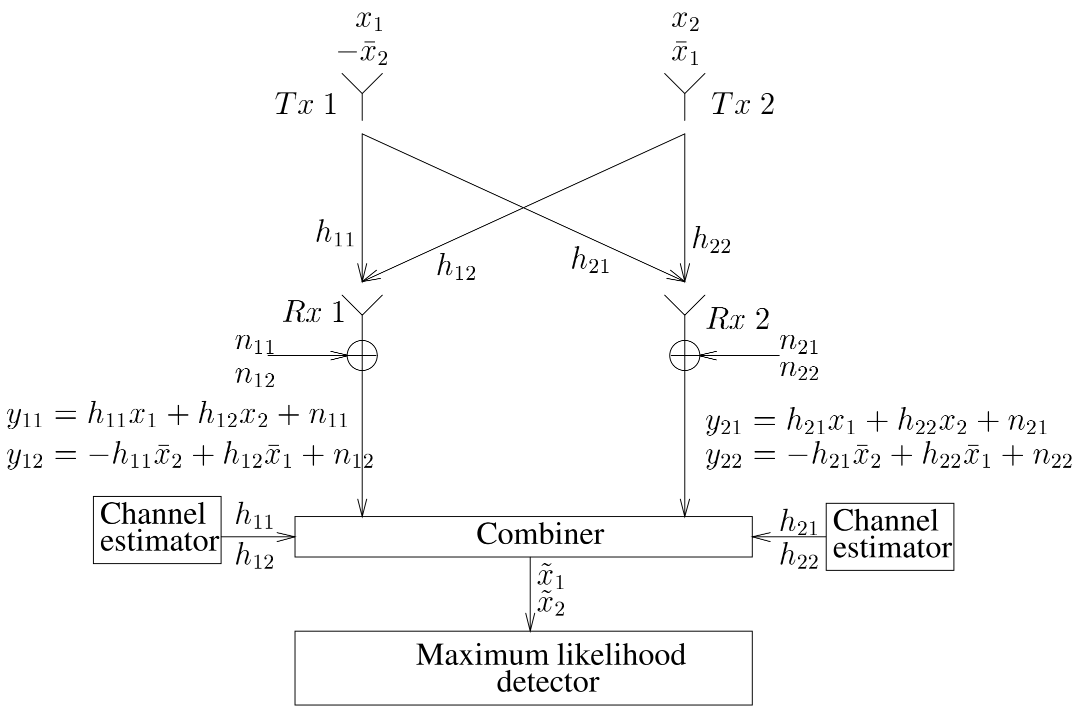


Figure 9.3: Baseband representation of the simple twin-transmitter space-time block code $\mathbf{G}_2$ of Equation 9.10 using two receivers.

In Section 9.3.1.1 we have shown an example of the encoding and decoding process for the $\mathbf{G}_2$ space-time block code of Equation 9.10 using one receiver. However, this example can be readily extended to an arbitrary number of receivers. The encoding and transmission sequence will be identical to the case of a single receiver. For illustration, we discuss the specific case of two transmitters and two receivers, as shown in Figure 9.3. We will show however that the generalisation to q receivers is straightforward. In Figure 9.3, the subscript i in the notation h_{ij}, n_{ij} and y_{ij} represents the receiver index. By contrast, the subscript j denotes the transmitter index in the CIR h_{ij}, but it denotes the time slot T in n_{ij} and y_{ij}. Therefore, at the first receiver Rx 1 we have:

$$\begin{aligned} y_{11} &= h_{11}x_1 + h_{12}x_2 + n_{11} && (9.17) \\ y_{12} &= -h_{11}\bar{x}_2 + h_{12}\bar{x}_1 + n_{12} \ , && (9.18) \end{aligned}$$

while at receiver Rx 2 we have

$$\begin{aligned} y_{21} &= h_{21}x_1 + h_{22}x_2 + n_{21} && (9.19) \\ y_{22} &= -h_{21}\bar{x}_2 + h_{22}\bar{x}_1 + n_{22} \ . && (9.20) \end{aligned}$$

We can, however, generalise these equations to:

$$\begin{aligned} y_{i1} &= h_{i1}x_1 + h_{i2}x_2 + n_{i1} && (9.21) \\ y_{i2} &= -h_{i1}\bar{x}_2 + h_{i2}\bar{x}_1 + n_{i2} \ , && (9.22) \end{aligned}$$

where $i = 1, ..., q$ and q is the number of receivers, which is equal to two in this example. At the combiner of Figure 9.3, the received signals are combined to extract the transmitted signals x_1 and x_2 from the received signals y_{11}, y_{12}, y_{21} and y_{22}, as follows:

$$\begin{aligned} \tilde{x}_1 &= \bar{h}_{11}y_{11} + h_{12}\bar{y}_{12} + \bar{h}_{21}y_{21} + h_{22}\bar{y}_{22} && (9.23) \\ \tilde{x}_2 &= \bar{h}_{12}y_{11} - h_{11}\bar{y}_{12} + \bar{h}_{22}y_{21} - h_{21}\bar{y}_{22} \ . && (9.24) \end{aligned}$$

Again, we can generalise the above expressions to q receivers, yielding:

$$\begin{aligned} \tilde{x}_1 &= \sum_{i=1}^{q} \left(\bar{h}_{i1}y_{i1} + h_{i2}\bar{y}_{i2}\right) && (9.25) \\ \tilde{x}_2 &= \sum_{i=1}^{q} \left(\bar{h}_{i2}y_{i1} - h_{i1}\bar{y}_{i2}\right) \ . && (9.26) \end{aligned}$$

Finally, we can simplify Equations 9.23 and 9.24 to:

$$\tilde{x}_1 = \left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right) x_1 + \bar{h}_{11}n_{11} + h_{12}\bar{n}_{12} + \bar{h}_{21}n_{21} + h_{22}\bar{n}_{22} \quad (9.27)$$

$$\tilde{x}_2 = \left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right) x_2 + \bar{h}_{12}n_{11} - h_{11}\bar{n}_{12} + \bar{h}_{22}n_{21} - h_{21}\bar{n}_{22} \ . \quad (9.28)$$

In the generalised form of q receivers we have:

$$\begin{aligned} \tilde{x}_1 &= \sum_{i=1}^{q} \left[\left(|h_{i1}|^2 + |h_{i2}|^2\right) x_1 + \bar{h}_{i1}n_{i1} + h_{i2}\bar{n}_{i2}\right] && (9.29) \\ \tilde{x}_2 &= \sum_{i=1}^{q} \left[\left(|h_{i1}|^2 + |h_{i2}|^2\right) x_2 + \bar{h}_{i2}n_{i1} - h_{i1}\bar{n}_{i2}\right] \ . && (9.30) \end{aligned}$$

Signals $\tilde{x}_1$ and $\tilde{x}_2$ are finally derived and passed to the maximum likelihood detector seen in Figure 9.3. Again, Equation 9.7 is applied to determine the maximum likelihood transmitted symbols.

We observe in Equation 9.29 that signal x_1 is multiplied by a term related to the fading amplitudes, namely $|h_{i1}|^2 + |h_{i2}|^2$. Hence, in order to acquire a high reliability signal $\tilde{x}_1$, the amplitudes of the CIRs must be high. If the number of receivers is equal to one, i.e. $q = 1$, then Equation 9.29 is simplified to Equation 9.15. In Equation 9.15, we can see that there are two fading amplitude terms, i.e. two independent paths associated with transmitting the symbol x_1. Therefore, if either of the paths is in a deep fade, the other path still may provide a high-reliability for the transmitted signal x_1. This explains, why the performance of a system having two transmitters and one receiver is better, than that of the system employing one transmitter and one receiver. On the other hand, in the conventional single-transmitter, single-receiver system there is only a single propagation path, which may be severely attenuated by a deep fade. To elaborate further, if the number of receivers is increased to $q = 2$, Equation 9.27 accrues from Equation 9.29. We can see in Equation 9.27 that there are now twice as many propagation paths, as in Equation 9.15. This increases the probability of providing a high reliability for the signal $\tilde{x}_1$.

9.3.2 Other Space-Time Block Codes

In Section 9.3.1 we have detailed Alamouti's simple two-transmitter space-time block code namely the $\mathbf{G}_2$ code of Equation 9.10. This code is significantly less complex, than the space-time trellis codes of [77, 87, 278–281] using two transmit antennas. However, again, there is a performance loss compared to the space-time trellis codes of [77, 87, 278–281]. Despite its performance loss, Alamouti's scheme [78] is appealing in terms of its simplicity. This motivated Tarokh *et al.* [79] to search for similar schemes using more than two transmit antennas. In [79] the theory of orthogonal code design was invoked, in order to construct space-time block codes having more than two transmitters. The half-rate space-time block code employing three transmitters was defined as [79]:

$$\mathbf{G}_3 = \begin{pmatrix} x_1 & x_2 & x_3 \\ -x_2 & x_1 & -x_4 \\ -x_3 & x_4 & x_1 \\ -x_4 & -x_3 & x_2 \\ \bar{x}_1 & \bar{x}_2 & \bar{x}_3 \\ -\bar{x}_2 & \bar{x}_1 & -\bar{x}_4 \\ -\bar{x}_3 & \bar{x}_4 & \bar{x}_1 \\ -\bar{x}_4 & -\bar{x}_3 & \bar{x}_2 \end{pmatrix}, \tag{9.31}$$

and the four-transmitter half-rate space-time block code was specified as [79]:

$$\mathbf{G}_4 = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \\ \bar{x}_1 & \bar{x}_2 & \bar{x}_3 & \bar{x}_4 \\ -\bar{x}_2 & \bar{x}_1 & -\bar{x}_4 & \bar{x}_3 \\ -\bar{x}_3 & \bar{x}_4 & \bar{x}_1 & -\bar{x}_2 \\ -\bar{x}_4 & -\bar{x}_3 & \bar{x}_2 & \bar{x}_1 \end{pmatrix} . \tag{9.32}$$

By employing the space-time block codes $\mathbf{G}_3$ and $\mathbf{G}_4$, we can see that the bandwidth efficiency has been reduced by a factor of two compared to the space-time block code $\mathbf{G}_2$. Besides, the number of transmission slots across which the channels is required to have a constant fading envelope is eight, namely four times higher, than that of the space-time code $\mathbf{G}_2$.

In order to increase the associated bandwidth efficiency, Tarokh *et al.* constructed the rate $3/4$ so-called generalised complex orthogonal sporadic codes [79, 80]. The corresponding rate $3/4$ three-transmitter space-time block code is given by [79]:

$$\mathbf{H}_3 = \begin{pmatrix} x_1 & x_2 & \frac{x_3}{\sqrt{2}} \\ -\bar{x}_2 & \bar{x}_1 & \frac{x_3}{\sqrt{2}} \\ \frac{\bar{x}_3}{\sqrt{2}} & \frac{\bar{x}_3}{\sqrt{2}} & \frac{(-x_1-\bar{x}_1+x_2-\bar{x}_2)}{2} \\ \frac{\bar{x}_3}{\sqrt{2}} & -\frac{\bar{x}_3}{\sqrt{2}} & \frac{(x_2+\bar{x}_2+x_1-\bar{x}_1)}{2} \end{pmatrix} , \tag{9.33}$$

while the rate $3/4$ four-transmitter space-time block code is defined as [79]:

$$\mathbf{H}_4 = \begin{pmatrix} x_1 & x_2 & \frac{x_3}{\sqrt{2}} & \frac{x_3}{\sqrt{2}} \\ -\bar{x}_2 & \bar{x}_1 & \frac{x_3}{\sqrt{2}} & -\frac{x_3}{\sqrt{2}} \\ \frac{\bar{x}_3}{\sqrt{2}} & \frac{\bar{x}_3}{\sqrt{2}} & \frac{(-x_1-\bar{x}_1+x_2-\bar{x}_2)}{2} & \frac{(-x_2-\bar{x}_2+x_1-\bar{x}_1)}{2} \\ \frac{\bar{x}_3}{\sqrt{2}} & -\frac{\bar{x}_3}{\sqrt{2}} & \frac{(x_2+\bar{x}_2+x_1-\bar{x}_1)}{2} & \frac{(-x_1-\bar{x}_1-x_2+\bar{x}_2)}{2} \end{pmatrix} . \tag{9.34}$$

In Table 9.2 we summarise the parameters associated with all space-time block codes proposed by Alamouti [78] and Tarokh, Jafarkhani as well as Calderbank [79, 80]. The decoding algorithms and the corresponding performance results of the space-time block codes were given in [80].

9.3.3 MAP Decoding of Space-Time Block Codes

Recently Bauch [283] derived a simple symbol-by-symbol Maximum-A-Posteriori (MAP) decoding rule for space-time block codes. The soft-outputs provided by the space-time MAP decoder can be used as the input to channel decoders, such as for example turbo codes, which may be concatenated for further improving the system's performance.

By using Bayes' rule, the a-posteriori probability of the transmitted k-ary symbols $x_1, ..., x_k$ given the received signals $y_{11}, ..., y_{1n}, y_{21}, ..., y_{qn}$ can be expressed as [283]:

$$P\left(x_1, ..., x_k | y_{11}, ..., y_{qn}\right) = P\left(y_{11}, ..., y_{qn} | x_1, ..., x_k\right) \cdot P\left(x_1, ..., x_k\right) \ , \tag{9.35}$$

Space-time code	Rate	No. of transmitters, p	No. of input symbols, k	Code span, n
$\mathbf{G}_2$	1	2	2	2
$\mathbf{G}_3$	$1/2$	3	4	8
$\mathbf{G}_4$	$1/2$	4	4	8
$\mathbf{H}_3$	$3/4$	3	3	4
$\mathbf{H}_4$	$3/4$	4	3	4

Table 9.2: Table of different space-time block codes.

where $P(x_1, ..., x_k)$ is the associated *a-priori* information, of the transmitted symbols which can be obtained from other independent sources, for example from channel decoders. Furthermore, according to Bauch [283], over non-dispersive Rayleigh fading channels we have:

$$P(y_{11}, ..., y_{qn}|x_1, ..., x_k) = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^{qn}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\sum_{i=1}^{n}\left|y_{li} - \sum_{j=1}^{p} h_{lj} g_{ji}\right|^2\right\}, \tag{9.36}$$

where σ is the noise variance and g_{ij} are the entries of the transmission matrix in Equation 9.8. We can, however, simplify Equation 9.35 and obtain the expression for the a-posteriori probability of each transmitted symbol x_i as [283]:

$$P(x_i|y_{11}, ..., y_{qn}) = P(y_{11}, ..., y_{qn}|x_i) \cdot P(x_i), \tag{9.37}$$

where $i = 1, ..., k$.

Let us now consider as an example the simplest possible space-time code, namely $\mathbf{G}_2$ associated with $k = 2$, $n = 2$ and $p = 2$. Assuming that there is no a-priori information, i.e. that $P(x_1, ..., x_k) = C$ where C is a constant, we obtain the a-posteriori information of the transmitted k-ary symbols from Equation 9.35 and 9.36, as [283]:

$$\begin{aligned}
& P(x_1, ..., x_k|y_{11}, ..., y_{qn}) \\
= \; & C \cdot \frac{1}{\left(\sigma\sqrt{2\pi}\right)^{qn}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[\left|y_{l1} - \sum_{j=1}^{p} h_{lj} g_{j1}\right|^2 + \left|y_{l2} - \sum_{j=1}^{p} h_{lj} g_{j2}\right|^2\right]\right\} \\
= \; & C' \cdot \exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[|y_{l1} - h_{l1}g_{11} - h_{l2}g_{21}|^2 + |y_{l2} - h_{l1}g_{12} - h_{l2}g_{22}|^2\right]\right\} \\
= \; & C' \cdot \exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[|y_{l1} - h_{l1}x_1 - h_{l2}x_2|^2 + |y_{l2} + h_{l1}\bar{x}_2 - h_{l2}\bar{x}_1|^2\right]\right\},
\end{aligned} \tag{9.38}$$

where $C' = C \cdot \frac{1}{\left(\sigma\sqrt{2\pi}\right)^{qn}}$. In order to obtain the expression of the a-posteriori probability for symbol x_1, x_2-related terms can be eliminated in Equation 9.38 due to the orthogonality of the code, arriving at:

$$\begin{aligned} & P\left(x_1|y_{11},...,y_{q2}\right) \\ = \; & C' \cdot \exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[\left|y_{l1}-h_{l1}x_1\right|^2+\left|y_{l2}-h_{l2}\bar{x}_1\right|^2\right]\right\} \\ = \; & C'' \cdot \exp\left\{-\frac{1}{2\sigma^2}\sum_{l=1}^{q}\left[-h_{l1}x_1\bar{y}_{l1}-\bar{h}_{l1}\bar{x}_1y_{l1}-h_{l2}\bar{x}_1\bar{y}_{l2}-\bar{h}_{l2}x_1y_{l2}+\left|x_1\right|^2\sum_{i=1}^{2}\left|h_{li}\right|^2\right]\right\}, \end{aligned} \tag{9.39}$$

where $|y_{l1}|^2$ and $|y_{l2}|^2$ are constants, which do not depend on x_1 and hence incorporated into C''. Following a few further manipulations, we can simplify Equation 9.39 to:

$$\begin{aligned} & P\left(x_1|y_{11},...,y_{q2}\right) \\ = \; & C \cdot \exp\left\{-\frac{1}{2\sigma^2}\left[\left|\left[\sum_{l=1}^{q}\left(\bar{h}_{l1}y_{l1}+h_{l2}\bar{y}_{l2}\right)\right]-x_1\right|^2+\left(-1+\sum_{l=1}^{q}\sum_{i=1}^{2}\left|h_{li}\right|^2\right)\left|x_1\right|\right]\right\}. \end{aligned} \tag{9.40}$$

Similarly, we can eliminate the x_1-related terms in Equation 9.38 and simplify it to:

$$\begin{aligned} & P\left(x_2|y_{11},...,y_{q2}\right) \\ = \; & C \cdot \exp\left\{-\frac{1}{2\sigma^2}\left[\left|\left[\sum_{l=1}^{q}\left(\bar{h}_{l2}y_{l1}-h_{l1}\bar{y}_{l2}\right)\right]-x_2\right|^2+\left(-1+\sum_{l=1}^{q}\sum_{i=1}^{2}\left|h_{li}\right|^2\right)\left|x_2\right|\right]\right\}. \end{aligned} \tag{9.41}$$

It can be seen that Equations 9.40 and 9.41 resemble the equations given in [80] for the maximum likelihood decoding of the space-time code $\mathbf{G}_2$. Besides considering the space-time code $\mathbf{G}_2$, the maximum likelihood decoding algorithms were also given for the space-time codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ in [80]. It can be shown that Bauch's MAP algorithms [283] applicable to the space-time codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ also resemble the maximum likelihood algorithms given in [80].

9.4 Channel Coded Space-Time Block Codes

In Section 9.3, we have given a detailed illustration of the concept of space-time block codes. The MAP decoding rules were also applied to space-time block codes in Section 9.3.3. This enables the space-time decoder to provide soft-outputs, which in turn can be used by the concatenated channel decoders. Hence, in this section we concatenate space-time block codes with Convolutional Codes (CC) [12, 284, 285], Turbo Convolutional (TC) codes [21, 22], Turbo BCH (TBCH) codes [68], Trellis Coded Modulation (TCM) [53,54] and Turbo Trellis Coded Modulation (TTCM) [64]. The performance and estimated complexity of each scheme will be studied and compared. We will also address the issue of mapping channel coded bits of the TC and TBCH schemes to different protection classes in multilevel modulation [206].

Convolutional codes (CC) were first suggested by Elias [12] in 1955. The so-called Viterbi Algorithm (VA) was proposed by Viterbi [17, 18] in 1967 for the maximum likelihood decoding of convolutional codes. As an alternative decoder, the more complex Maximum A-Posteriori (MAP) algorithm was proposed by Bahl [20], which provided the optimum Bit Error Rate (BER) performance, although this was not significantly better than that of the Viterbi algorithm. In the early 1970s, convolutional codes were used in deep-space and satellite communications. They were then also adopted by the Global System of Mobile communications (GSM) [56] for the pan-European digital cellular mobile radio system.

In 1993, Berrou *et al.* [21, 22] proposed a novel channel code, referred to as a turbo code. As detailed in Chapter 6, the turbo encoder consists of two component encoders. Generally, convolutional codes are used as the component encoders, and the corresponding turbo codes are termed here as a TC code. However, BCH [56, 92] codes can also be employed as their component codes, resulting in the so-called turbo BCH codes (TBCH). They have been shown for example by Hagenauer [68, 70] to perform impressively at near-unity coding rates, although at a higher decoding complexity than that of the corresponding-rate TCs.

In 1987, Ungerboeck [53, 54] invented trellis coded modulation (TCM) by combining the design of channel coding and modulation. TCM optimises the Euclidean distance between codewords and hence maximises the coding gain. In [64], Robertson *et al.* applied the basic idea of turbo codes [21, 22] to TCM by retaining the important properties and advantages of both structures. In the resultant TTCM scheme, two Ungerboeck codes [53, 54] are employed in combination with TCM as component codes in an overall structure similar to that of turbo codes.

9.4.1 System Overview

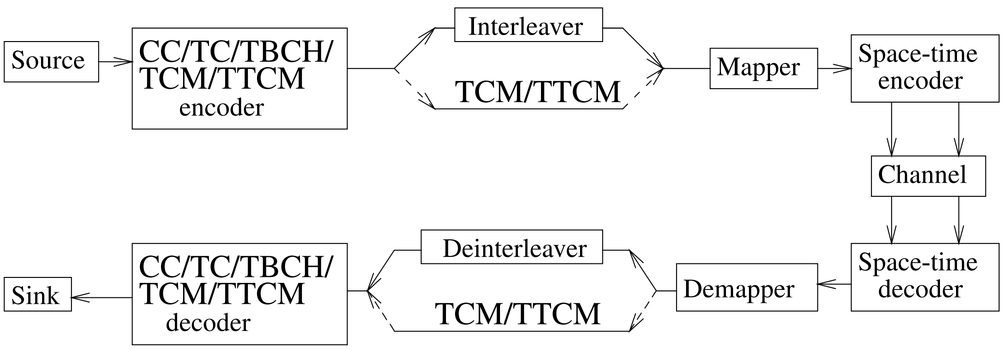


Figure 9.4: System overview of space-time block codes and different channel coding schemes.

The schematic of the proposed concatenated space-time block codes and the different channel coding schemes is shown in Figure 9.4. As mentioned above, the investigated channel coding schemes are CC, TC codes, TBCH codes, TCM and TTCM. The information source at the transmitter of Figure 9.4 generates random data bits. The information bits are then encoded by each of the above five different channel coding schemes. However, only the output binary bits of the CC, TBCH and TC coding schemes are channel interleaved, as seen in Figure 9.4. The role of the interleaver will be detailed in Section 9.5.2.

The output bits of the TCM and TTCM scheme are passed directly to the mapper in Figure 9.4, which employs two different mapping techniques. Gray-mapping [62, 92, 286] is used for the CC, TBCH and TC schemes, whereas set-partitioning [53–55, 64] is utilised for the TCM and TTCM scheme. Different modulation schemes are employed, namely Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), 8-level Phase Shift Keying (8PSK), 16-level Quadrature Amplitude Modulation (16QAM) and 64-level Quadrature Amplitude Modulation (64QAM) [206].

Following the mapper, the channel coded symbols are passed to the space-time block encoder, as shown in Figure 9.4. Below, we will investigate the performance of all the previously mentioned space-time block codes, namely that of the $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ codes proposed in [78–80]. The corresponding transmission matrices are given in Equations 9.10, 9.31, 9.32, 9.33 and 9.34, respectively. The coding rate and number of transmitters of the associated space-time block codes is shown in Table 9.2. The channels are uncorrelated or $-$ synonymously $-$ perfectly interleaved narrow-band or non-dispersive Rayleigh fading channels. This assumption does not contradict to requiring a constant channel magnitude and phase over p (number of rows in the transmission matrix) consecutive symbols, since upon applying a sufficiently high channel interleaving depth the channels' fading envelope can become indeed near-uncorrelated. We assumed that the narrow-band fading amplitudes received from each transmitter antenna were mutually uncorrelated Rayleigh distributed processes. The average signal power received from each transmitter antenna was the same. Furthermore, we assumed that the receiver had a perfect estimate of the channels' fading amplitudes. In practice, the channels' fading amplitude can be estimated for example with the aid of pilot symbols [206].

At the receiver, the number of receiver antennas constitutes a design parameter, which was fixed to one, unless specified otherwise. The space-time block decoders then apply the MAP or Log-MAP decoding algorithm of Section 9.3.3 for the decoding of the signals received from the different antennas. Due to its implementational simplicity, the Log-MAP decoding algorithm is preferred in the proposed system. The soft outputs associated with the received bits or symbols are passed through the channel deinterleaver or directly to the TCM/TTCM decoder, respectively, as seen in Figure 9.4. The channel-deinterleaved soft outputs of the received bits are then passed to the CC, TC or TBCH decoders. The Viterbi algorithm [17, 18] is applied in the CC and TCM decoder. By contrast, all turbo decoder schemes apply the Log-MAP [22, 64, 68] decoding algorithm. The decoded bits are finally passed to the information sink for calculation of the BER, as shown in Figure 9.4.

9.4.2 Channel Codec Parameters

In Figure 9.4, we have given an overview of the proposed system. As we can see in the figure, there are different channel encoders to be considered, namely the CC, TC, TBCH, TCM and TTCM schemes. In this section, we present the parameters of all the channel codecs to be used in our investigations.

Table 9.3 shows the parameters of each channel encoder proposed in the system. We commence with the most well-known channel code, namely the convolutional code. A convolutional code is described by three parameters n, k and K and it is denoted as CC(n,k,K).

Code	Octal generator polynomial	No. of states	Decoding algorithm	No. of iterations
Convolutional Code (CC)				
CC(2,1,5)	23,33	16	VA	–
CC(2,1,7)	171,133	64	VA	–
CC(2,1,9)	561,753	256	VA	–
Turbo Convolutional Code (TC)				
TC(2,1,3)	7,5	4	Log-Map	8
TC(2,1,4)	13,15	8	Log-Map	8
TC(2,1,5)	23,35	16	Log-Map	8
Turbo BCH Code (TBCH)				
TBCH(31,26)	45	32	Log-Map	8
TBCH(32,26)	45	64	Log-Map	8
TBCH(31,21)	3551	1024	Log-Map	8
TBCH(63,57)	103	64	Log-Map	8
TBCH(127,120)	211	128	Log-Map	8
Trellis Coded Modulation (TCM)				
8PSK-TCM	103,30,66	64	VA	–
16QAM-TCM	101,16,64	64	VA	–
Turbo Trellis Coded Modulation (TTCM)				
8PSK-TTCM	11,2,4	8	Log-Map	8
16QAM-TTCM	23,2,4,10	16	Log-Map	8

Table 9.3: Parameters of the different channel encoders used in Figure 9.4.

At each instant, a CC(n,k,K) encoder accepts k input bits and outputs n coded bits. The constraint length of the code is K and the number of encoder states is equal to 2^{K-1}. The channel coded rate is given by

$$R = \frac{k}{n} \, . \tag{9.42}$$

However, different code rates can be obtained by suitable puncturing [233] and we will elaborate on this issue later in the section. The first entry of Table 9.3 is the convolutional code CC$(2,1,5)$, which was adopted by the Groupe Speciale Mobile (GSM) committee in 1982 [56, 287]. Then in 1996, a more powerful convolutional code, the CC$(2,1,7)$ arrangement was employed by the Digital Video Broadcasting (DVB) [46] standard for television, sound and data services. Recently, the Universal Mobile Telecommunication System (UMTS) proposed the use of the CC$(2,1,9)$ scheme, which is also shown in Table 9.3. The implementation of this scheme is about 16 times more complex than that of the CC$(2,1,5)$ scheme adopted by GSM some 15 years ago. This clearly shows that the advances of integrated circuit technology have substantially contributed towards the performance improvement of mobile communication systems.

As mentioned earlier, a turbo encoder consists of two component encoders. Generally, two identical Recursive Systematic Convolutional (RSC) codes are used. Berrou *et al.* [21,22]

used two constraint length $K = 3$, RSC codes, each having four trellis states. We denote a turbo convolutional code as TC(n, k, K) where n, k and K have their usual interpretations, as in CC. In [21,22], the MAP algorithm [20] was employed for iterative decoding. However, in our systems the Log-MAP decoding algorithm [59] was utilised. The Log-MAP algorithm is a more attractive version of the MAP algorithm, since it operates in the logarithmic domain, in order to reduce the computational complexity and to mitigate the numerical problems associated with the MAP algorithm [59]. The number of turbo iterations was set to eight, since this yielded a performance close to the optimum performance associated with an infinite number of iterations. In our investigations we will consider the turbo convolutional code TC($2, 1, 3$), as proposed in [21, 22]. However, the more complex TC($2, 1, 4$) code [288] was proposed by UMTS to be employed in the third generation (3G) mobile communication systems [56,266,289]. The TC($2, 1, 5$) code is also interesting, since it is expected to provide further significant coding gains over that of the TC($2, 1, 3$) and TC($2, 1, 4$) code.

BCH codes [56] are used as the component codes in the TBCH codes of Table 9.3. Again, TBCH codes have been shown for example by Hagenauer [68,70] to perform impressively at near-unity coding rates, although at high complexity. Hence in our study the BCH component codes BCH($31, 26$), BCH($31, 21$), BCH($63, 57$) and BCH($127, 120$) are employed, as shown in Table 9.3. Finally, we also investigate TCM and TTCM. Both of them are employed in 8PSK and 16QAM modulation modes. This results in 8PSK-TCM, 16QAM-TCM and 8PSK-TTCM, 16QAM-TTCM, respectively.

In Table 9.3, we have given the encoding and decoding parameters of the different channel encoders employed. However, as mentioned earlier, we can design codes of variable code rates R by employing suitable puncturing patterns. By combining puncturing with different modulation modes, we could design a system having a range of various throughputs, expressed in terms of the number of Bits Per Symbol (BPS), as shown in Table 9.4 and 9.5. Some of the parameters in Table 9.4 and 9.5 are discussed in depth during our further discourse, but significantly more information can be gleaned concerning these systems by carefully studying both tables.

Code	Code Rate R	Puncturing Pattern	Modulation Mode	BPS	Random interleaver depth
Convolutional Code (CC)					
CC(2,1,5)	0.50	1,1	QPSK	1.00	20,000
CC(2,1,7)	0.50	1,1	QPSK	1.00	20,000
	0.75	101, 110	64QAM	4.50	13,320
	0.83	10101, 11010	64QAM	5.00	12,000
CC(2,1,9)	0.50	1,1	QPSK	1.00	20,000
			16QAM	2.00	20,000
			64QAM	3.00	20,004
Trellis Coded Modulation (TCM)					
8PSK-TCM	0.67	1,1	8PSK	2.00	–
16QAM-TCM	0.75	1,1	16QAM	3.00	–

Table 9.4: Simulation parameters associated with the CC and TCM channel encoders in Figure 9.4.

In Table 9.4 we summarised the simulation parameters of the CC and TCM schemes employed. Since there are two coded bits ($n = 2$) for each data bit ($k = 1$), we have two possible puncturing patterns, as shown in the table. A binary 1 means that the coded bit is transmitted, whereas a binary 0 implies that the coded bit is punctured. Accordingly, the puncturing pattern $(1, 1)$ simply implies that no puncturing is applied and hence results in a half-rate CC. However, for example in the DVB standard [46] different puncturing patterns were proposed for the CC$(2, 1, 7)$ code, which result in different coding rates. These are also shown in Table 9.4.

Code	Code Rate R	Puncturing Pattern	Modulation Mode	BPS	Random turbo interleaver depth	Random (separation) interleaver depth
Turbo Convolutional Code (TC)						
TC(2,1,3)	0.50	$10, 01$	16QAM	2.00	10,000	20,000
TC(2,1,4)	0.33	$1, 1$	64QAM	2.00	10,000	30,000
	0.50	$10, 01$	16QAM	2.00	10,000	20,000
			64QAM	3.00	10,002	20,004
	0.67	$1000, 0001$	64QAM	4.00	10,000	15,000
	0.75	$100000,$ 000001	16QAM	3.00	9,990	13,320
			64QAM	4.50	9,990	13,320
	0.83	$1000000000,$ 0000000001	64QAM	5.00	10,000	12,000
	0.90	$100000000000000000,$ 000000000000000001	64QAM	5.40	10,044	11,160
TC(2,1,5)	0.50	$10, 01$	16QAM	2.00	10,000	20,000
Turbo BCH Code (TBCH)						
TBCH(31,26)	0.72	1,1	16QAM	2.89	9,984	13,824
			64QAM	4.33	9,984	13,824
TBCH(32,26)	0.68	1,1	8PSK	2.05	9,984	14,592
TBCH(31,21)	0.51	1,1	16QAM	2.04	9,996	19,516
TBCH(63,57)	0.83	1,1	64QAM	4.96	10,032	12,144
TBCH(127,120)	0.90	1,1	64QAM	5.37	10,080	11,256
Turbo Trellis Coded Modulation (TTCM)						
8PSK-TTCM	$2/3$	$10, 01$	8PSK	2.00	10,000	–
16QAM-TTCM	$3/4$	$10, 01$	16QAM	3.00	13,332	–

Table 9.5: Simulation parameters associated with the TC, TBCH and TTCM channel encoders in Figure 9.4.

In Table 9.5 we showed the simulation parameters of three different turbo schemes, namely that of the TC, TBCH and TTCM arrangements. Again, different code rates can be designed using suitable puncturing patterns, where the puncturing patterns seen in Table 9.5 consist of two parts. Specifically, the associated different puncturing patterns represent the puncturing patterns of the parity bits emanating from the first and the second encoder, respectively. These patterns are different from the puncturing patterns seen in Table 9.4. For the TC$(2,1,3)$ scheme different puncturing patterns are employed for the various code rates R. The puncturing patterns were optimised experimentally by simulations, in order to attain the best possible BER performance. The design procedure for punctured turbo codes was proposed by Acikel *et al.* [71] in the context of BPSK and QPSK.

9.4.3 Complexity Issues and Memory Requirements

In this section we address the complexity issues and memory requirements of the proposed system. We will mainly focus on the relative estimated complexity and memory requirements of the proposed channel decoders rather than attempting to determine their exact complexity. Therefore, in order to simplify our comparative study, several assumptions are made. In our simplified approach the estimated complexity of the whole system is deemed to depend only on that of the channel decoders. In other words, the complexity associated with the modulator, demodulator, space-time encoder and decoder as well as channel encoders are assumed to be insignificant compared to the complexity of channel decoders.

Since the estimated complexity of the channel decoders depends directly on the number of trellis transitions, the number of trellis transitions per information data bit will be used as the basis of our comparison. Several channel encoders schemes in Table 9.3 are composed of convolutional codes. For the binary convolutional code CC$(2,1,K)$, two trellis transitions diverge from each of the 2^{K-1} states. Hence, we can approximate the complexity of a CC$(2,1,K)$ code as:

$$\begin{aligned} comp\,\{\text{CC(2,1,K)}\} &= 2 \times 2^{K-1} \\ &= 2^K\ . \end{aligned} \tag{9.43}$$

The number of trellis transitions in the Log-MAP decoding algorithm is assumed to be three times higher, than that of the conventional Viterbi algorithm, since the Log-MAP algorithm has to perform forward as well as backward recursion and soft output calculations, which results in traversing through the trellis three times. The reader is referred to Section 6.3.3.4 for further details of the algorithm. For TC codes we apply the Log-MAP decoding algorithm for iterative decoding, assisted by the two component decoders. Upon taking into account the number of turbo decoding iterations as well, the complexity of TC decoding is then approximated by:

$$\begin{aligned} comp\,\{TC(2,1,K)\} &= 3 \times 2 \times 2^{K-1} \times 2 \times \text{No. of Iterations} \\ &= 3 \times 2^{K+1} \times \text{No. of Iterations}\ . \end{aligned} \tag{9.44}$$

In TCM we construct a non-binary decoding trellis [55]. The TCM schemes of Table 9.3 have 2^{BPS-1} trellis branches diverging from each trellis state, where BPS is the number of transmitted bits per modulation symbol. However, for each trellis transition we would have $BPS-1$ transmitted information data bits, since the TCM encoder typically adds one parity bit per non-binary symbol. Therefore, we can estimate the complexity of the proposed TCM schemes as:

$$comp\,\{\text{TCM}\} = 2^{BPS-1} \times \frac{\text{No. of States}}{BPS-1}\ . \tag{9.45}$$

Similarly to TC, TTCM consists of two TCM codes and the Log-MAP decoding algorithm [64] is employed for iterative decoding. The associated TTCM complexity is then estimated as:

$$\begin{aligned} comp\{\text{TTCM}\} &= 3 \times 2^{BPS-1} \times \frac{\text{No. of States}}{BPS-1} \times 2 \times \text{No. of Iterations} \\ &= \frac{3 \times 2^{BPS} \times \text{No. of States} \times \text{No of Iterations}}{BPS-1} . \end{aligned} \quad (9.46)$$

For TBCH(n,k) codes the estimated complexity calculation is not as straightforward as in the previous cases. Its component codes are BCH(n,k) codes and the decoding trellis can be divided into three sections [27]. Assuming that $k > n-k$, for every decoding instant j the number of trellis states is given as [27]:

$$\text{No. of States}_j = \begin{cases} 2^j & j = 0,1,...,n-k-1 \\ 2^{n-k} & j = n-k, n-k+1, ..., k \\ 2^{n-j} & j = k+1, k+2, ..., n \end{cases} . \quad (9.47)$$

It can be readily shown that:

$$\begin{aligned} 2^{n-k} - 1 &= \sum_{j=0}^{n-k-1} 2^j && (9.48) \\ &= \sum_{j=k+1}^{n} 2^{n-j} . && (9.49) \end{aligned}$$

Upon using the approximation $\sum_{j=0}^{n-k-1} 2^j = \sum_{j=k+1}^{n} 2^{n-j} = 2^{n-k} - 1 \approx 2^{n-k}$, we can write the number of decoding trellis states per information data bit as:

$$\begin{aligned} \text{No. of States} &= \frac{2 \times 2^{n-k} + \{k-(n-k)\} \times 2^{n-k}}{k} \\ &= \frac{(2k-n+2) \times 2^{n-k}}{k} . \end{aligned} \quad (9.50)$$

Having derived the number of decoding trellis states per information data bit, we can approximate the complexity of TBCH codes as:

$$\begin{aligned} comp\{\text{TBCH}(n,k)\} &= 3 \times 2 \times \frac{(2k-n+2) \times 2^{n-k}}{k} \times 2 \times \text{No. of Iterations} \\ &= \frac{3 \times (2k-n+2) \times 2^{n-k+2} \times \text{No. of Iterations}}{k} . \end{aligned} \quad (9.51)$$

Having approximated the complexity of each channel decoder, we will now derive their approximate memory requirements. Typically, the memory requirement of a channel decoder depends directly on the number of trellis states in the entire coded block. Therefore in this section the number of trellis states per coded block serves as the basis of a relative memory requirement comparison between the channel decoders studied. For a binary convolutional code, observation of the VA has shown that typically all surviving paths of the current trellis state emerge from trellis states not 'older' than approximately five times the constraint length, K. Therefore at any decoding instant, only a section of $5 \times K$ trellis transitions has to be stored. We can then approximate the associated memory requirement as:

$$mem\,\{CC(2,1,K)\} = 2^{K-1} \times 5 \times K\;. \tag{9.52}$$

Again, as highlighted in Section 6.3.3.4, the Log-MAP algorithm requires the storage of γ, α and β values. Hence for the same number of decoding trellis states, the Log-MAP algorithm would require about three times more memory, than the classic Viterbi algorithm. Consequently, we can estimate the memory requirement of the TC code as:

$$mem\,\{\text{TC}(2,1,K)\} = 3 \times 2^{K-1} \times \text{Block Length}\;. \tag{9.53}$$

Similarly to CCs, we can approximate the memory requirements of TCM as:

$$mem\,\{\text{TCM}\} = \text{No. of States} \times \text{Block Length}\;. \tag{9.54}$$

Following similar arguments, the memory requirements of TTCM employing the Log-MAP algorithm can be approximated as:

$$mem\,\{\text{TTCM}\} = 3 \times \text{No. of States} \times \text{Block Length}\;. \tag{9.55}$$

The estimation of the memory requirements of TBCH codes is again different from that of the other channel codes considered. Specifically, their memory requirement does not directly depend on the number of decoding trellis states in a coded TBCH block. Instead, it depends on the number of decoding trellis states in the constituent BCH codewords. From Equation 9.50, we can estimate the associated memory requirements as:

$$mem\,\{\text{TBCH}(n,k)\} = 3 \times (2k-n+2) \times 2^{n-k}\;. \tag{9.56}$$

Applying Equations 9.43 to 9.56, we summarised the estimated complexity and memory requirements of the channel decoders characterised in Table 9.3. Explicitly, assuming that there are 10,000 information data bits per coded block, the associated estimated complexity and memory requirements are then given in Table 9.6. Note that the block length of TCM and TTCM is expressed in terms of the number of symbols per coded block, since these schemes are symbol-oriented rather than bit-oriented.

9.5 Performance Results

In this section, unless otherwise stated, all simulation results are obtained over uncorrelated or – synonymously – perfectly interleaved narrow-band or non-dispersive Rayleigh fading channels. As stated before, this does not contradict to requiring a constant channel magnitude and phase over n consecutive time slots in Equation 9.8, since upon applying a sufficiently high interleaving depth the channel's fading envelope can be indeed uncorrelated. Our assumptions were that:

1) The fading amplitudes were constant across n consecutive transmission slots of the space-time block codes' transmission matrix;

Code	No. of states	No. of states per data bit	Iteration No	Block length	Complexity	Memory requirement
Convolutional Code (CC)						
CC(2,1,5)	16	16	–	10,000	32	400
CC(2,1,7)	64	64	–	10,000	128	2,240
CC(2,1,9)	256	256	–	10,000	512	11,520
Turbo Convolutional Code (TC)						
TC(2,1,3)	4	4	8	10,000	384	120,000
TC(2,1,4)	8	8	8	10,000	768	240,000
TC(2,1,5)	16	16	8	10,000	1,536	480,000
Turbo BCH Code (TBCH)						
TBCH(31,26)	32	28	8	31	2,718	2,208
TBCH(32,26)	64	54	8	32	5,199	4,224
TBCH(31,21)	1,024	634	8	31	60,855	39,936
TBCH(63,57)	64	60	8	63	5,713	10,176
TBCH(127,120)	128	123	8	127	11,776	44,160
Trellis Coded Modulation (TCM)						
8PSK-TCM	64	32	–	5,000	128	320,000
16QAM-TCM	64	21	–	3,333	171	213,312
Turbo Trellis Coded Modulation (TTCM)						
8PSK-TTCM	8	4	8	5,000	768	120,000
16QAM-TTCM	16	5	8	3,333	2,048	159,984

Table 9.6: Complexity and memory requirements of the different channel decoders in characterised Table 9.3.

2) The average signal power received from each transmitter antenna was the same;

3) The receiver had a perfect knowledge of the channels' fading amplitudes.

We note that the above assumptions are unrealistic, yielding the best-case performance, nonetheless, facilitating the performance comparison of the various techniques under identical circumstances.

In the following sections, we compare the performance of various combinations of space-time block codes and channel codes. As mentioned earlier, various code rates can be used for both the space-time block codes and for the associated channel codes. The different modulation schemes employed result in various effective throughput. Hence, for a fair comparison, all different systems are compared on the basis of the same effective BPS throughput given by:

$$\text{BPS} = R_{st} \times R_{cc} \times \text{modulation throughput}, \tag{9.57}$$

where R_{st} and R_{cc} are the code rates of the space-time block code and the channel code, respectively.

9.5.1 Performance Comparison Of Various Space-Time Block Codes Without Channel Codecs

In this section, the performance of various space-time block codes without channel codes is investigated and compared. All the investigated space-time block codes, namely the $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ codes [78–80] have their corresponding transmission matrices given in Equation 9.10, 9.31, 9.32, 9.33 and 9.34, respectively. The encoding parameters are summarised in Table 9.2.

9.5.1.1 Maximum Ratio Combining and the Space-Time Code $\mathbf{G}_2$

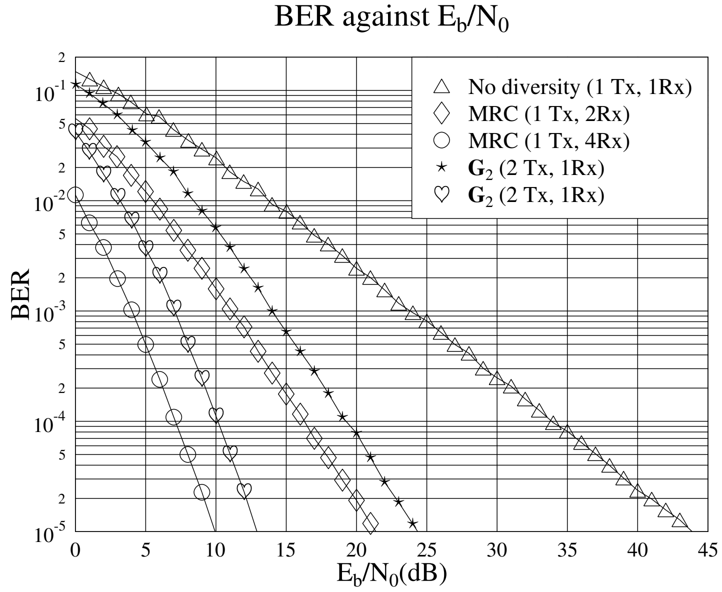


Figure 9.5: Performance comparison of the MRC technique and space-time code $\mathbf{G}_2$ using **BPSK** over uncorrelated Rayleigh fading channels.

Figure 9.5 shows the performance of MRC and the space-time code $\mathbf{G}_2$ using BPSK over uncorrelated Rayleigh fading channels. It is assumed that the total power received from both transmit antennas in the space-time coded system using $\mathbf{G}_2$ of Equation 9.10 is the same as the transmit power of the single transmit antenna assisted MRC system. It can be seen in Figure 9.5 that the performance of the space-time code $\mathbf{G}_2$ is about 3 dB worse, than that of the MRC technique using two receivers, even though both systems have the same diversity order of two. The 3 dB penalty is incurred, because the transmit power of each antenna in the $\mathbf{G}_2$ space-time coded arrangement is only half of the transmit power in the MRC assisted system. It is shown in Figure 9.5 however that at a BER of 10^{-5} a diversity gain of 20 dB is achieved by the space-time code $\mathbf{G}_2$. If we increase the diversity order to four by using two receivers, the space-time code $\mathbf{G}_2$ achieves a diversity gain of 32 dB. However, there is still a 3 dB performance penalty as compared to the conventional MRC technique using four receivers. The advantage of the space-time coded scheme is nonetheless that the increased complexity of the space-time coded transmitter is more affordable at the BS than at the MS, where the MRC receiver would have to be located.

9.5.1.2 Performance of 1 BPS Schemes

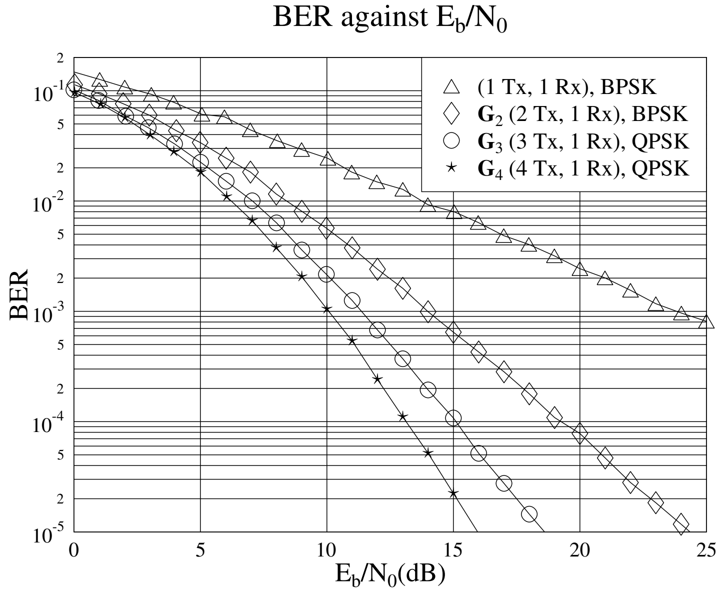


Figure 9.6: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ of Table 9.2 at an effective throughput of **1 BPS** using **one receiver** over uncorrelated Rayleigh fading channels.

Figures 9.6 and 9.7 compare the performance of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ having an effective throughput of 1 BPS over uncorrelated Rayleigh fading channels using one and two receivers, respectively. BPSK modulation was employed in conjunction with the space-time code $\mathbf{G}_2$. As shown in Table 9.2, the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ are half-rate codes. Therefore, QPSK modulation was used in the context of $\mathbf{G}_3$ and $\mathbf{G}_4$ in order to retain a throughput of 1 BPS. It can be seen in Figure 9.6 that at a BER of 10^{-5} the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ give about 2.5 and 7.5 dB gain over the $\mathbf{G}_2$ code, respectively. If the number of receivers is increased to two, as shown in Figure 9.7, the associated E_b/N_0 gain reduces to about 1 and 3.5 dB, respectively. The reason is that over the perfectly interleaved flat-fading channel encountered much of the attainable diversity gain is already achieved using the $\mathbf{G}_2$ code and two receivers. The associated gains of the various schemes at a BER of 10^{-5} are summarised in Table 9.7.

9.5.1.3 Performance of 2 BPS Schemes

In Figure 9.8 we compare the performance of the space-time codes $\mathbf{G}_2, \mathbf{G}_3, \mathbf{G}_4, \mathbf{H}_3$ and $\mathbf{H}_4$ proposed in [78–80] using the encoding parameters summarised in Table 9.2. The perfor-

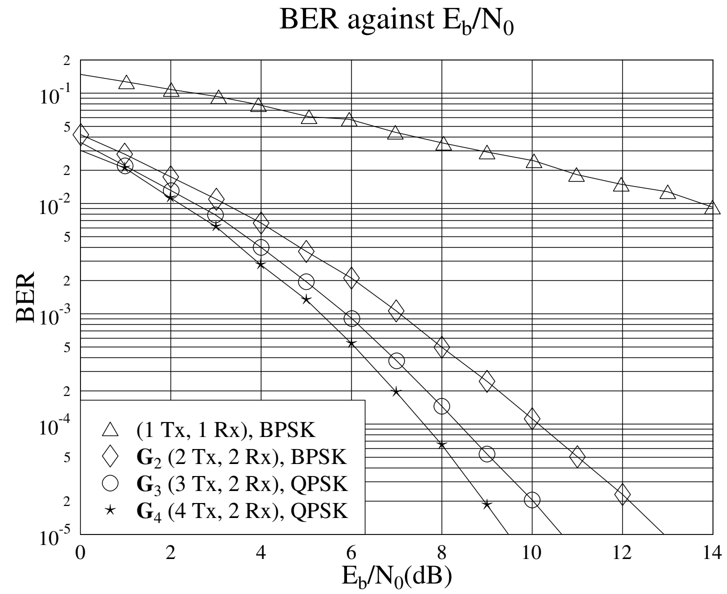


Figure 9.7: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ of Table 9.2 at an effective throughput of **1 BPS** using **two receivers** over uncorrelated Rayleigh fading channels.

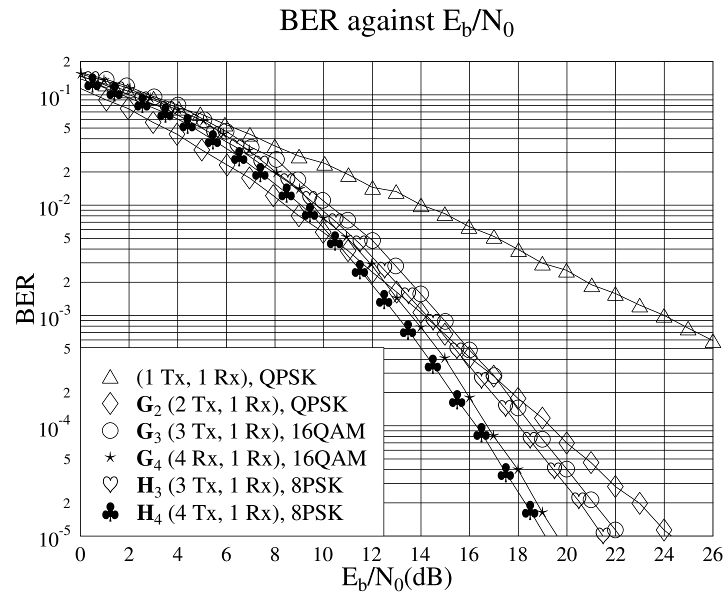


Figure 9.8: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ at an effective throughput of approximately **2 BPS** using **one receiver** over uncorrelated Rayleigh fading channels. The associated parameters of the space-time codes are summarised in Table 9.2.

mance results were obtained over uncorrelated Rayleigh fading channels using one receiver and the effective throughput of the system is about 2 BPS. For the $\mathbf{G}_2$ code QPSK modulation was used, while the $\mathbf{G}_3$ and $\mathbf{G}_4$ codes employ 16QAM conveying 4 BPS. Hence the effective throughput is 2 BPS, since $\mathbf{G}_3$ and $\mathbf{G}_4$ are half-rate codes. Since the code rate of the $\mathbf{H}_3$ and $\mathbf{H}_4$ codes is $\frac{3}{4}$, 8PSK modulation was employed in this context, resulting in a throughput of $3 \times 3/4 = 2.25$ BPS, which is approximately 2 BPS. We can see in Figure 9.8 that at high BERs or low E_b/N_0 values the $\mathbf{G}_2$ code slightly outperforms the others. However, the situation is reversed, when the system is operated at a low BER or high E_b/N_0 values. At a BER of 10^{-5} the code $\mathbf{G}_4$ only gives a diversity gain of 5 dB over the $\mathbf{G}_2$ code. This is a 2.5 dB loss compared to the 7.5 dB gain achieved by the system transmitting at an effective throughput of 1 BPS in the previous section. This is because the more vulnerable 16QAM scheme was used for the space-time code $\mathbf{G}_4$. Since the 16QAM signal constellation is more densely packed compared to QPSK, it is more prone to errors. Moreover, the space-time code $\mathbf{G}_4$ has no error correction capability to correct the extra errors induced by employing a more vulnerable, higher-order modulation scheme. Hence, this results in a poorer performance. If the throughput of the system is increased by employing an even higher-order modulation scheme, the space-time code $\mathbf{G}_4$ will suffer even higher performance degradations, as it will be shown in the next section. Since the space-time code $\mathbf{G}_3$ of Table 9.2 is also a half-rate code, similarly to the $\mathbf{G}_4$ code, it suffers from the same drawbacks.

In Figure 9.8, we also show the performance of the rate $\frac{3}{4}$ space-time codes $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 9.2. Both the $\mathbf{H}_4$ and $\mathbf{G}_4$ codes have the same diversity order of four in conjunction with one receiver. However, at a BER of 10^{-5} the performance of the $\mathbf{H}_4$ code is about 0.5 dB better, than that of the $\mathbf{G}_4$ code. This is again due to the higher-order modulation employed in conjunction with the half-rate code $\mathbf{G}_4$, in order to maintain the same throughput. As alluded to earlier, the higher-order modulation schemes are more susceptible to errors and hence the performance of the system in conjunction with the $\mathbf{G}_3$ or $\mathbf{G}_4$ code of Table 9.2 is worse, than that of the $\mathbf{H}_3$ or $\mathbf{H}_4$ code having the same diversity orders, respectively. The associated gains of the various schemes at a BER of 10^{-5} are summarised in Table 9.7.

9.5.1.4 Performance of 3 BPS Schemes

Figures 9.9 and 9.10 show our performance comparisons for the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 9.2 at an effective throughput of 3 BPS over uncorrelated Rayleigh fading channels using one and two receivers, respectively. When using the $\mathbf{G}_2$ code we employed 8PSK modulation. Since $\mathbf{G}_3$ and $\mathbf{G}_4$ are half-rate codes, 64QAM was employed, in order to obtain an effective throughput of 3 BPS. By contrast, for the $\mathbf{H}_3$ and $\mathbf{H}_4$ codes, which have a code rate of $\frac{3}{4}$, 16QAM was used in order to ensure the same throughput of $4/4 = 3$ BPS.

In Figure 9.9 we can see that at a BER of 10^{-5} the diversity gain of the $\mathbf{G}_4$ code over the $\mathbf{G}_2$ code is further reduced to about 3 dB. There is only a marginal diversity gain for the $\mathbf{G}_3$ code over the $\mathbf{G}_2$ code. As alluded to in the previous section, 64QAM in conjunction with the space-time code $\mathbf{G}_3$ or $\mathbf{G}_4$, has a densely packed signal constellation and hence this scheme is prone to errors. At the higher BER of 10^{-2} the $\mathbf{G}_2$ code outperforms the $\mathbf{G}_3$ and $\mathbf{G}_4$ codes by approximately 3 and 4 dB, respectively.

Due to the associated higher-order modulation scheme employed, we can see in Figure 9.9 that at a BER of 10^{-5} the $\mathbf{H}_3$ and $\mathbf{H}_4$ codes of Table 9.2 outperform both the $\mathbf{G}_3$ and the $\mathbf{G}_4$

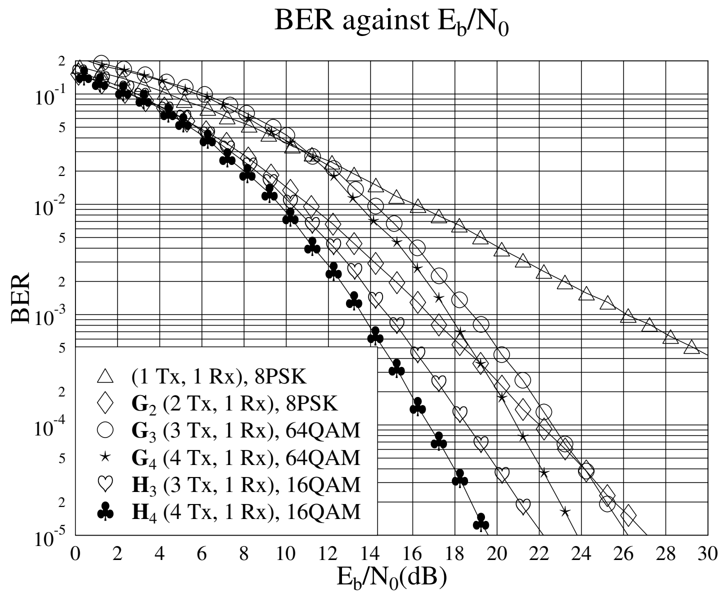


Figure 9.9: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 9.2 at an effective throughput of **3 BPS** using **one receiver** over uncorrelated Rayleigh fading channels.

codes. Specifically, we can see that the $\mathbf{H}_3$ code attains about 2 dB gain over the $\mathbf{G}_4$ code, even though it has a lower diversity order.

If we increase the number of receivers to two, a scenario characterised in Figure 9.10, the performance degradation of the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ is even more pronounced. At a BER of 10^{-5} the performance gain of the $\mathbf{H}_4$ code over the $\mathbf{G}_4$ code is approximately 4 dB compared to the 0.5 dB gain, when the system's effective throughput is only 2 BPS, as it was shown in Figure 9.8 of the previous section.

Having studied Figures 9.6 to 9.10, we may conclude two important points. Firstly, the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ of Table 9.2 suffer from having a code-rate of half, since this significantly reduces the effective throughput of the system. In order to maintain the same throughput as the unity-rate $\mathbf{G}_2$ code, higher-order modulation schemes, such as for example 64QAM have to employed. This results in a preponderance of channel errors, since the constellation points of the higher-order modulation schemes are more densely packed. Due to their lack of error correcting capability, the $\mathbf{G}_3$ and $\mathbf{G}_4$ codes suffer performance losses compared to the $\mathbf{G}_2$ code. Secondly, if the number of receivers is increased to two, the performance gain of the $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ or $\mathbf{H}_4$ codes over the $\mathbf{G}_2$ code becomes lower. The reason behind this phenomenon is that much of the attainable diversity gain was already achieved using the $\mathbf{G}_2$ code and two receivers. The associated gains of the various schemes at a BER of 10^{-5} are summarised in Table 9.7.

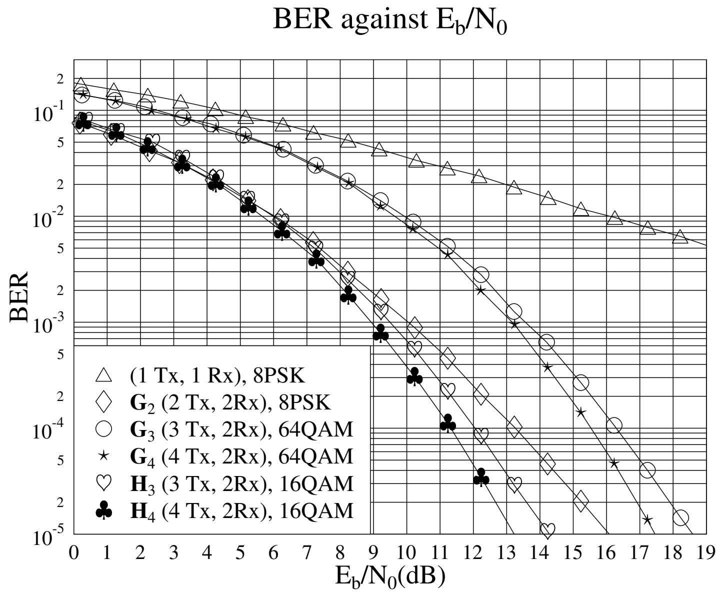


Figure 9.10: Performance comparison of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 9.2 at an effective throughput of **3 BPS** using **two receivers** over uncorrelated Rayleigh fading channels.

		One receiver			Two receivers		
Code	Rate	1 BPS	2 BPS	3 BPS	1 BPS	2 BPS	3 BPS
$\mathbf{G}_2$	1	19.5	19.6	19.1	30.9	30.9	30.1
$\mathbf{G}_3$	1/2	25.2	21.8	20.0	33.2	29.6	27.6
$\mathbf{G}_4$	1/2	27.9	24.3	22.4	34.3	30.7	28.8
$\mathbf{H}_3$	3/4	–	22.4	24.0	–	30.1	31.9
$\mathbf{H}_4$	3/4	–	24.8	22.6	–	31.2	33.0

Table 9.7: Coding gain of the space-time block codes of Table 9.2 over uncorrelated Rayleigh fading channels.

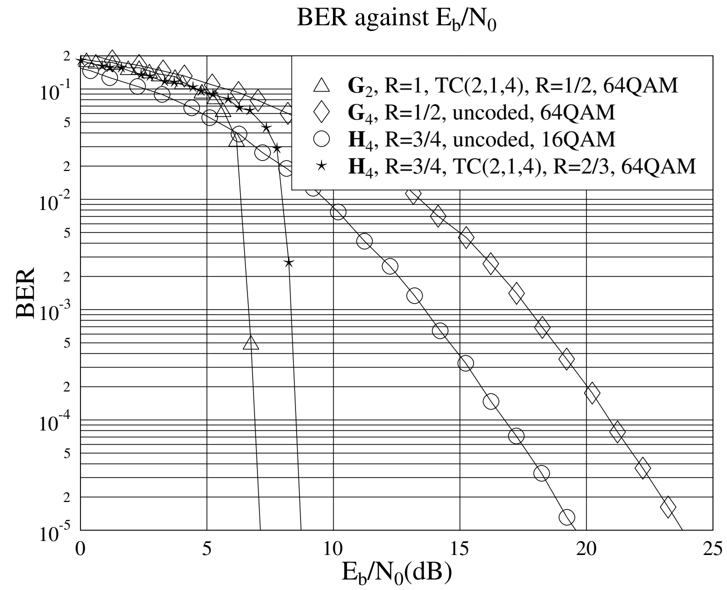


Figure 9.11: Performance comparison of the half-rate TC(2,1,4) code concatenated with the space-time code $\mathbf{G}_2$ and the space-time block codes $\mathbf{G}_4$ and $\mathbf{H}_4$. The associated parameters are shown in Tables 9.2, 9.3, and 9.5. All simulation results were obtained at an effective throughput of **3 BPS** over uncorrelated Rayleigh fading channels.

9.5.1.5 Channel Coded Space-Time Block Codes

In the previous sections, we have shown that without channel coding the performance of the unity-rate space-time $\mathbf{G}_2$ code is inferior to the lower rate space-time codes, namely to that of the $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ schemes. Since the space-time code $\mathbf{G}_2$ has a unity code rate, half-rate turbo codes can be employed for improving the performance of the system. In Figure 9.11, we compare the performance of the half-rate TC(2,1,4) code concatenated with the space-time code $\mathbf{G}_2$ and with the space-time block codes $\mathbf{G}_4$ and $\mathbf{H}_4$. Both the space-time codes $\mathbf{G}_4$ and $\mathbf{H}_4$ have a diversity gain of four and a code rate of $\frac{1}{2}$ and $\frac{3}{4}$, respectively. The associated parameters are shown in Tables 9.2, 9.3 and 9.5. Suitable modulation schemes were chosen so that all systems had the same throughput of 3 BPS. All simulation results were obtained over uncorrelated Rayleigh fading channels.

¿From Figure 9.11, we can see that a huge performance improvement is achieved by concatenating the space-time code $\mathbf{G}_2$ with the half-rate code TC(2,1,4). At a BER of 10^{-5} this concatenated scheme attains a coding gain of 16 dB and 13 dB compared to the space-time codes $\mathbf{G}_4$ and $\mathbf{H}_4$, respectively. This clearly shows that it is better to invest the parity bits associated with the code-rate reduction in the concatenated turbo code, rather than in non-unity-rate space-time block codes. In Figure 9.11 we also show the performance of the space-time code $\mathbf{H}_4$ concatenated with the punctured two-third rate code TC(2,1,4). The figure shows that the TC(2,1,4) code improves the performance of the system tremendously, attaining a coding gain of 11 dB compared to the non-turbo-coded space-time code $\mathbf{H}_4$, at

BER= 10^{-5}. However, its performance is still inferior to that of the half-rate TC(2,1,4) coded space-time code $\mathbf{G}_2$.

In conclusion, in Figure 9.11 we have seen that the reduction in coding rate is best assigned to turbo channel codes, rather to space-time codes. Therefore, in all our forthcoming simulations, all channel codecs of Table 9.3 are concatenated with the unity-rate space-time code $\mathbf{G}_2$, instead of the non-unity-rate space-time codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 9.2.

9.5.2 Mapping Binary Channel Codes to Multilevel Modulation

As mentioned earlier, in our investigations different modulation schemes are employed in conjunction with the binary channel codecs CC, TC and TBCH. Specifically, the modulation schemes used are BPSK, QPSK, 8PSK, 16QAM and 64QAM. Gray-mapping [92, 104, 206] is employed to map the bits to the QPSK, 8PSK, 16QAM and 64QAM symbols. In higher-order modulation schemes, such as 8PSK, 16QAM and 64QAM we have several transmitted bits per constellation point. However, the different bit positions of the constellation points have different noise-protection distances [206]. More explicitly, the protection distance is the Euclidean distance from one constellation point to another, which results in the corruption of a particular bit. A larger noise-protection distance results in a higher integrity of the bit and vice-versa. Therefore, for the different bit positions in the symbol we have different protection for the transmitted bits within the phaser constellation of the non-binary modulation schemes. It can be readily shown that in 8PSK and 16QAM we have two protection classes, namely class I and II [92, 206], where the class I transmitted bits are more protected. Similarly, in 64QAM we have three protection classes namely I, II and III [206], where the transmitted bits in class I are most protected, followed by class II and class III.

In our system the parity bits are generated by binary channel encoders, such as the CC, TC and TBCH schemes for protecting the binary data bits. However it is not intuitive, whether the integrity of the data or parity bits is more important in yielding a better overall BER performance. For example, if the parity bits are more important, it is better to allocate the parity bits to the better protection classes in higher-order modulation scheme and vice-versa. Therefore, in this section, we will investigate the performance of different channel codes along with different bit mapping schemes. The effect of the bit interleaver seen in Figure 9.4 is studied in conjunction with binary channel codes as well.

9.5.2.1 Turbo Convolutional Codes - Data and Parity Bit Mapping

We commence here by studying half-rate turbo convolutional codes, which are characterised in Table 9.3. An equal number of parity and data bits are generated by the half-rate TC codes and they are then mapped to the protection classes of the 16QAM scheme considered. Again, in the Gray-mapping assisted 16-QAM constellation there are two protection classes [206], class I and II, depending on the bit position. Explicitly, there are four bits per symbol in the 16-QAM constellation and two of the bit positions are more protected, than the remaining two bits.

In Figure 9.12 we compare the performance of various parity and data bit mapping schemes for the (a) TC(2,1,3), (b) TC(2,1,4) and (c) TC(2,1,5) codes. The curve marked by triangles represents the performance of the TC codes, when allocating the parity bits to the higher-integrity protection class I and the data bits to the lower-integrity protection class

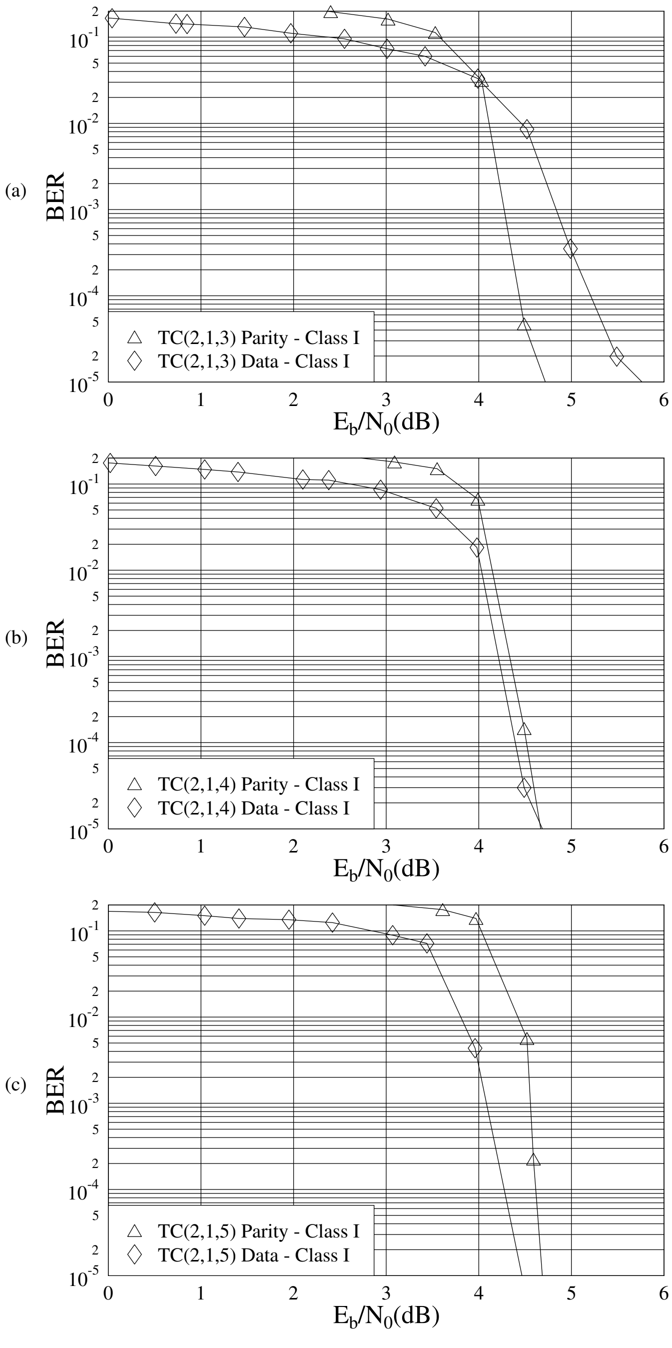


Figure 9.12: Performance comparison of various data and parity bit allocation schemes for the (a) TC(2,1,3), (b) TC(2,1,4) and (c) TC(2,1,5) codes, where the parameters are shown in Table 9.3. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver and 16QAM over uncorrelated Rayleigh fading channels at an effective throughput of 2 BPS.

II. On the other hand, the performance curve marked by diamonds indicates the allocation of data bits to protection class I, while the parity bits are assigned to protection class II.

In Figure 9.12(a), we can see that at low E_b/N_0 values the performance of the TC(2,1,3) code, when allocating the parity bits to protection class I is worse, than upon allocating the data bits to protection class I. However, for E_b/N_0 values in excess of about 4 dB, the situation is reversed. At a BER of 10^{-5}, there is a performance gain of about 1 dB when using the TC(2,1,3) arrangement with the parity bits allocated to protection class I. We surmise that by protecting the parity bits better, we render the TC(2,1,3) code more powerful. It is common that stronger channel codes perform worse, than weaker codes at low E_b/N_0 values, but outperform their less powerful counterparts for higher E_b/N_0 values.

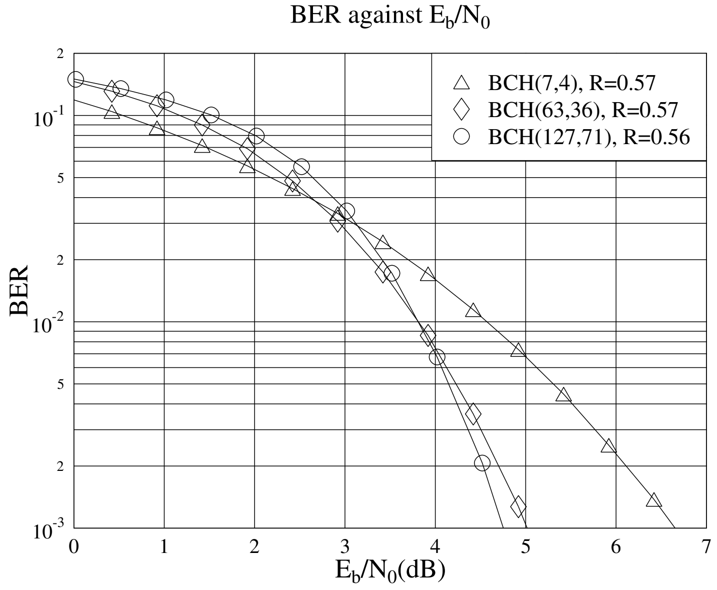


Figure 9.13: Performance comparison of hard decision algebraic decoding of different BCH codes having approximately the same code rate of $R = 0.57$, using BPSK over AWGN channels.

This is further justified in Figure 9.13. Here, we showed the performance of hard decision algebraic decoding of the BCH(7,4), BCH(63,36) and BCH(127,71) codes using BPSK over AWGN channels. All BCH codes characterised in the figure have approximately the same code rate, which is $R = 0.57$. From the figure we can see that at a BER of 10^{-3} the performance of the BCH codes improves with an increasing codeword length n. However, at a high BER or low E_b/N_0 value we can see that the performance of the BCH(7,4) code is better, than that of the BCH(63,36) and BCH(127,71) codes, which are stronger channel codes. This is, because stronger codes have many codewords having a large free distance. At low SNRs we have bad channel conditions and hence the channel might corrupt even those codewords having a large free distance. Once they are corrupted, they produce many erroneous information bits, a phenomenon which results in a poorer BER performance.

In Figure 9.12(b) we showed the performance of the TC(2,1,4) code using the same data and parity bit allocation, as in Figure 9.12(a). The figure clearly shows that the TC(2,1,4) scheme exhibits a better performance for E_b/N_0 values below about 4.7 dB, if the data bits are more strongly protected than the parity bits. It is also seen from the figure that the situation is reversed for E_b/N_0 values above this point. This phenomenon is different from the behaviour of the TC(2,1,3) scheme, since the crossing point of both curves occurs at a significantly lower BER. The same situation can be observed for the BCH codes characterised in Figure 9.13, where we can see that the performance curve of the BCH(127,71) code crosses the performance curve of the BCH(63,36) scheme at $E_b/N_0 \approx 4$ dB. This value is lower, than the crossing point of the performance curves of the BCH(63,36) and BCH(7,4) codes. Hence the trend is that the crossing point of stronger codes is shifted to right of the figure. Hence the crossing point of the performance curves of stronger codes will occur at lower BERs and shifted to the right on the E_b/N_0 scale. From the above argument we can speculate also in the context of TC codes that since the TC(2,1,4) scheme is a stronger code than the TC(2,1,3) arrangement, the crossing point of the associated performance curves for TC(2,1,4) is at a lower BER, than that of the TC(2,1,3) code and appears to be shifted to right of the E_b/N_0 scale.

Let us now consider the same performance curves in the context of the significantly stronger TC(2,1,5) code in Figure 9.12(c). The figure clearly shows that better performance is yielded in the observed range, when the data bits are more strongly protected. Unlike in Figure 9.12(a) and 9.12(b), there is no visible crossing point in Figure 9.12(c). However, judging from the gradient of both curves, if we were to extrapolate the curves in Figure 9.12(c), they might cross at BER$\approx 10^{-6}$. The issue of data and parity bit mapping to multilevel modulation schemes was also addressed by Goff *et al.* [62, 286]. However, the authors only investigated the performance of the TC(2,1,5) code and stated that better performance is achieved by protecting more strongly the data bits. Additionally, we note here that the situation was reversed for the TC(2,1,3) code, where better performance was achieved by protecting the parity bits more strongly.

Hence, from the three subfigures of Figure 9.12 we can draw the following conclusions for the mapping of the data and parity bits to the different protection classes of the modulated symbol. For weaker half-rate turbo codes, such as the TC(2,1,3) arrangement, it is better to protect the parity bits more strongly. On the other hand, for stronger half-rate turbo codes, such as the TC(2,1,4) and TC(2,1,5) schemes, better performance is achieved by protecting more strongly the data bits. From our simulation results, we found that the same scenario also applies to turbo codes having code rates lower or higher than half-rates, as shown in Table 9.5. Based on these facts, we continue our investigations into the effect of interleavers, in an effort to achieve an improved performance.

9.5.2.2 Turbo Convolutional Codes − Interleaver Effects

In Figure 9.4 we have seen that a bit-based interleaver is employed for the CC, TC and TBCH codes. Since our performance results are obtained over uncorrelated Rayleigh fading channels, the purpose of the bit-based interleaver is to disperse bursts of channel errors within a modulated symbol, when it experiences a deep fade. This is vital for TC codes, because according to the turbo code structure proposed by Berrou *et al.* in [21, 22], at the output of the turbo encoder, a data bit is followed by the parity bits generated for its protection against errors. Therefore in multi-level modulation schemes a particular modulated symbol could consist of the data bit and its corresponding parity bits generated for its protection. If the symbol experiences a deep fade, the demodulator would provide low reliability values for both the data bit and the associated parity bits. In conjunction with low reliability information the turbo decoder may fail to correct errors induced by the channel. However, we can separate both the data bit and the parity bits generated for its protection into different modulation symbols. By doing so, there is a better chance that the demodulator can provide high-reliability parity bits, which are represented by another modulation symbol, even if the data bit experienced a deep fade and vice-versa. This will assist the turbo decoder in correcting errors.

More explicitly, the random interleaver shown in Figure 9.4 has two different effects on the binary channel codes, namely:

1) It separates the data bit and the parity bits generated for its protection into different modulated symbols;

2) It randomly maps the data and parity bits into different protection classes in multi-level modulation schemes.

The first effect of the random interleaver will improve the performance of the binary channel codecs. By contrast, the second effect might have a negative impact on the performance of the channel codecs, because the data and parity bits are randomly mapped to the different protection classes, rather than assigning the more vulnerable bits consistently to the higher-integrity protection class.

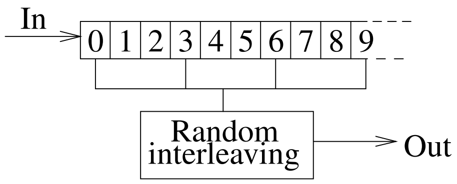


Figure 9.14: Random separation based interleaving.

In order to eliminate the potentially detrimental second effect of the random interleaver, we propose to invoke a so-called random separation based interleaver. Explicitly, Figure 9.14 shows an example of the random separation based interleaving employed. The objective of random separation based interleaving is to randomly interleave the bits within the same protection class of the multilevel modulated symbols. If 8PSK modulation is used, 3 bits per symbol are transmitted. Hence, for every 3-bit spaced position, the bits will be randomly interleaved. For example, in Figure 9.14 we randomly interleaved the bit positions $0, 3, 6, 9, ...$ Similarly, bit positions $1, 4, 7, ...$ and $2, 5, 8, ...$ will be randomly interleaved as well.

In Figure 9.15 we investigated the effects of both a random interleaver and those of a random separation based interleaver on the performance of the TC(2,1,3) code. The encoding parameters of the TC(2,1,3) code are shown in Table 9.3. The simulation results were obtained in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and 16QAM over uncorrelated Rayleigh fading channels. The performance curves marked by the triangles and diamonds were obtained by protecting the parity bits and data bits more strongly, respectively. Recall that the same performance curves were also shown in Figure 9.12(a).

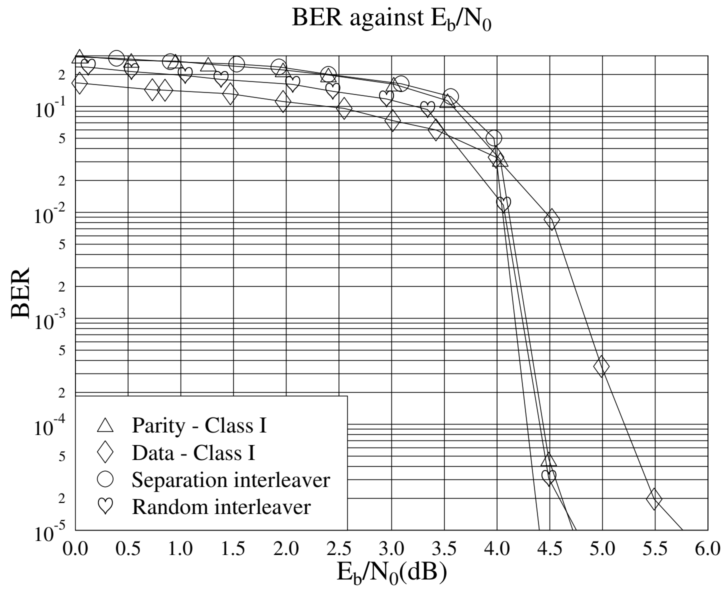


Figure 9.15: Performance comparison between different bit-to-symbol mapping methods for the **TC(2,1,3)** code in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and **16QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TC(2,1,3) code are shown in Table 9.3.

As mentioned earlier, the random interleaver has two different effects on the performance of binary channel codes. It randomly maps the data and parity bits into different protection classes which might have a negative impact on the performance of the channel codecs. Additionally, it may separate the data bits and parity bits generated for their protection into different modulated symbols, which on the other hand might improve the performance. In Figure 9.15 the random interleaver based performance curve is marked by the hearts, which is similar to that of the TC(2,1,3) coded scheme protecting the parity bits more strongly. This suggest that the above-mentioned positive effect of the random interleaver is more pronounced than the negative effect in the context of the TC(2,1,3) coded scheme. On the other hand, based on the evidence of Figure 9.12(a) the random separation based interleaver was ultimately applied in conjunction with the allocation of the parity bits, rather than the data bits into protection class I. The interleaver randomly interleaved the coded bits within the same protection class of a block of transmitted symbols. Therefore, the parity bits remained more protected compared to the data bits and yet they have been randomly interleaved within the set of parity. In Figure 9.15 the performance of the random separation based interleaver is marked by circles, which is about 0.5 dB better, than that of the TC(2,1,3) coded scheme with the parity bits allocated to protection class I.

Similarly to Figure 9.15, in Figures 9.16 and 9.17 we show the performance of the TC(2,1,4) and TC(2,1,5) codes, respectively, using different bit-to-symbol mapping methods. All simulation results were obtained in conjunction with the space-time code $\mathbf{G}_2$ using

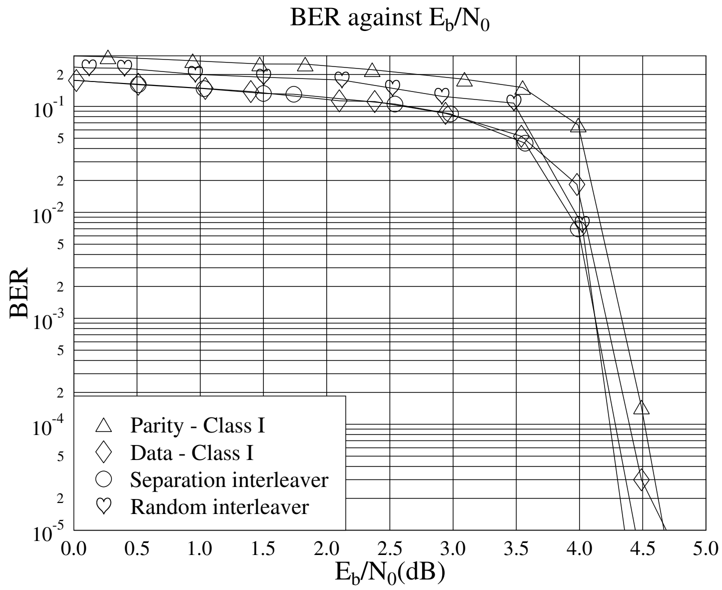


Figure 9.16: Performance comparison between different bit-to-symbol mapping methods for the **TC(2,1,4)** code in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and **16QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TC(2,1,4) code are shown in Table 9.3.

one receiver and 16QAM over uncorrelated Rayleigh fading channels. The encoding parameters of the TC(2,1,4) and TC(2,1,5) codes are shown in Table 9.3. Unlike in Figure 9.15, the random separation based interleaver was applied in conjunction with the allocation of the data bits, rather than the parity bits to protection class I. It can be seen from Figures 9.16 and 9.17 that the performance of the random interleaver and random separation based interleaver is similar. This again suggest that the above-mentioned positive effect yielded by the random based interleaver is more pronounced than its detrimental effect in the context of both the TC(2,1,4) and TC(2,1,5) schemes.

In conclusion, our simulation results presented in this section demonstrated that at a BER of 10^{-5} the half-rate turbo codes using a random separation based interleaver attain the best performance, albeit for certain schemes only by a small margin. Therefore in our forthcoming performance comparisons we will be employing the random separation based interleaver in conjunction with the various TC codes.

9.5.2.3 Turbo BCH Codes

Figure 9.18 characterises the performance of the TBCH(32,26) code in conjunction with different bit-to-symbol mapping to the two protection classes of 8PSK. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using one receiver and 8PSK over uncorrelated Rayleigh fading channels. Again, the encoding parameters of the TBCH(32,26) code are shown in Tables 9.3 and 9.5. The TBCH(32,26) code was chosen for our inves-

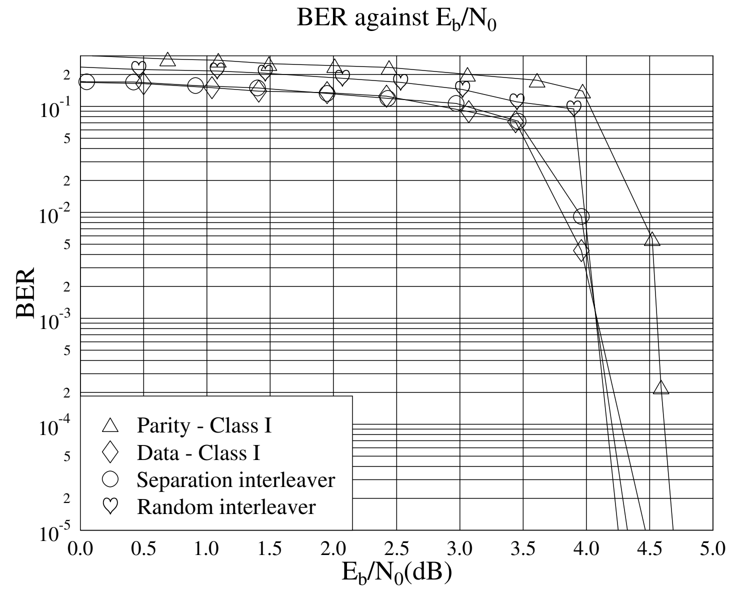


Figure 9.17: Performance comparison between different bit-to-symbol mapping methods for the **TC(2,1,5)** code in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and **16QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TC(2,1,5) code are shown in Table 9.3.

tigations, because the parity bits of the constituent encoders were not punctured and hence this resulted in a code rate of $R \approx \frac{2}{3}$. Roughly speaking, for every two data bits, there is one parity bit. Similarly to 16QAM, in the Gray-mapping assisted 8-PSK constellation there are also two protection classes, depending on the bit position in the 3-bit symbols. From the three bits of the 8-PSK constellation two of the bit positions are more protected, than the remaining bit. In Figure 9.18, we portray the performance of the TBCH(32,26) scheme for four different bit-to-symbol mapping methods. Firstly, one data bit and one parity bit was mapped to the two better protected 8-PSK bit positions. The corresponding BER curve was marked by the triangles in Figure 9.18. According to the second method, the data bits were mapped to the two better protected bit positions of the 8-PSK symbol. This scenario was marked by the diamonds in Figure 9.18. As we can see from the figure, the first mapping method yields a substantial E_b/N_0 gain of 1.5 dB at a BER of 10^{-5} over the second method. By applying the random separation based interleaver of Figure 9.14, while still better protecting one of the data bits and the parity bit than the remaining data bits, we disperse the bursty bit errors associated with a transmitted symbol over several BCH codewords of the turbo BCH code. As shown in Figure 9.18, the performance curve marked by the circles shows a slight improvement compared to the above-mentioned first method, although the difference is marginal. Finally, we show the performance of applying random interleaving, which randomly distributes the data and parity bits between the two 8-PSK protection classes. It can be seen that the associated performance is worse, than that of the first bit-to-symbol mapping

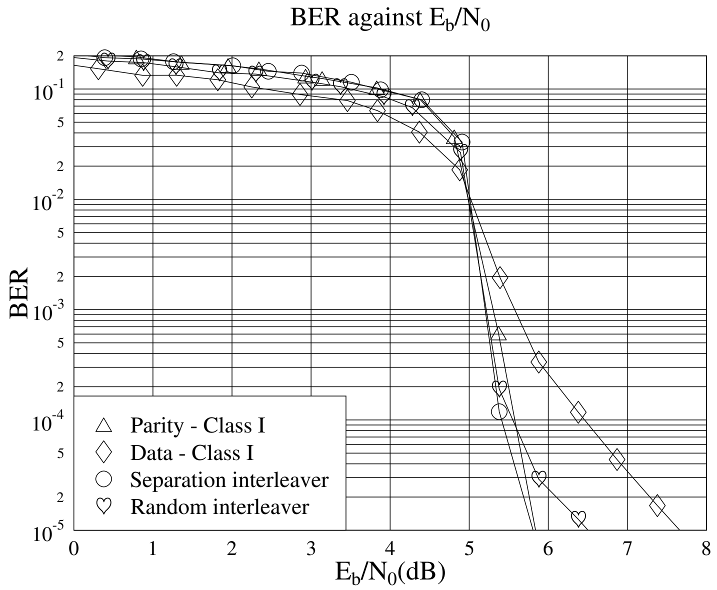


Figure 9.18: Performance comparison between different bit-to-symbol mapping methods for the TBCH(32,26) code in conjunction with the space-time code $\mathbf{G}_2$ using one receiver and **8PSK** over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**. The encoding parameters of the TBCH(32,26) code are shown in Tables 9.3 and 9.5.

method.

In Figure 9.18, we have shown that it is better to protect the parity bits more strongly for the TBCH(32,26) code and a slight further improvement can be achieved by applying a random separation based interleaver. More simulation results were obtained in conjunction with the other TBCH codes shown in Tables 9.3 and 9.5 with the aid of the space-time code $\mathbf{G}_2$ and 64QAM over uncorrelated Rayleigh fading channels. From the simulation results we have found that all TBCH codes shown in Tables 9.3 and 9.5 perform better, if the parity bits are more protected. In general, a slight further improvement can be obtained for TBCH codes, when a random separation based interleaver is applied. A possible explanation is that the component encoders of the TBCH codes are BCH encoders, where a block of parity bits is generated by a block of data bits. Hence, every parity bit has an influence on the whole codeword. Moreover, we used high-rate TBCH codes and hence there are more data bits compared to the parity bits. Hence, in our forthcoming TBCH comparisons, we will use bit-to-symbol mappers protecting the parity bits better.

9.5.2.4 Convolutional Codes

Let us now investigate the space-time code $\mathbf{G}_2$ in conjunction with the half-rate convolutional code CC(2,1,9) proposed for UMTS. The CC(2,1,9) code is a non-systematic non-recursive convolutional code, where the original information bits cannot be explicitly recognised in the encoded sequence. Its associated performance curve is shown in Figure 9.19 marked by

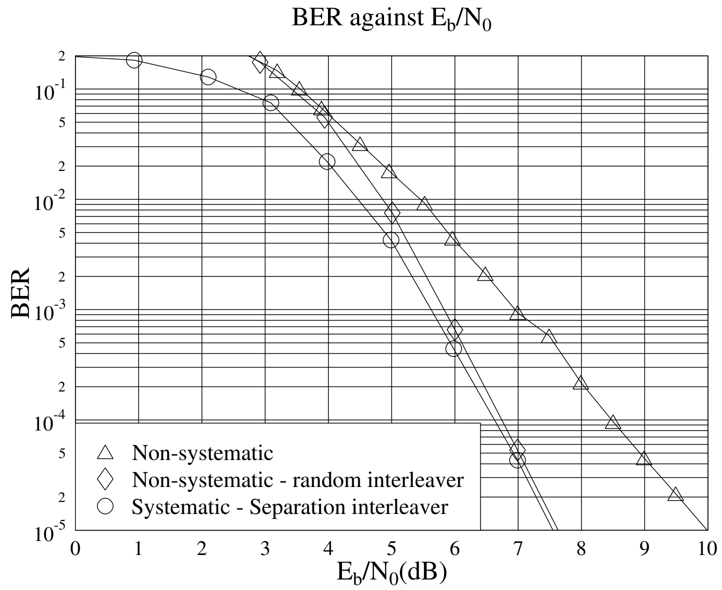


Figure 9.19: Performance comparison between the **systematic and non-systematic half-rate CC(2,1,9)** code in conjunction with the space-time code $\mathbf{G}_2$ and **16QAM** over uncorrelated Rayleigh fading channels at a throughput of **2 BPS**. The encoding parameters of the CC(2,1,9) code are shown in Tables 9.3 and 9.4.

the triangles. A random interleaver was then applied, in order to disperse the bursty channel errors and the associated performance curve is marked by the diamonds in Figure 9.19. At a BER of 10^{-5} there is a performance gain of 2.5 dB, if the random interleaver is applied. As a further scheme we invoked a systematic CC(2,1,9) code, which was obtained using a recursive convolutional code [56, 104]. Hence, in this scenario we have explicitly separable data bits and parity bits. In Figure 9.19 the performance curve marked by the circles is obtained by mapping the data bits of the systematic CC(2,1,9) code to protection class I of the associated 16QAM scheme in conjunction with the random separation based interleaver of Figure 9.14. From the figure we can see that there is only a marginal performance improvement over the non-systematic CC(2,1,9) code using the random interleaver.

9.5.3 Performance Comparison of Various Channel Codecs Using the G_2 Space-time Code and Multi-level Modulation

In this section we compare the $\mathbf{G}_2$ space-time coded performance of all channel codecs summarised in Table 9.3. In order to avoid having an excessive number of curves in one figure, only one channel codec will be characterised from each group of the CC, TC, TBCH, TCM and TTCM schemes. The choice of the channel codec considered depends on its performance, complexity and code rate. Unless otherwise stated, all channel codecs are concatenated with the space-time code $\mathbf{G}_2$ using one receiver. All comparison are carried out on the basis of the same BPS throughput over uncorrelated Rayleigh fading channels. Let us now briefly discuss in the forthcoming sections, how each channel codec is selected from the codec families considered.

9.5.3.1 Comparison of Turbo Convolutional Codes

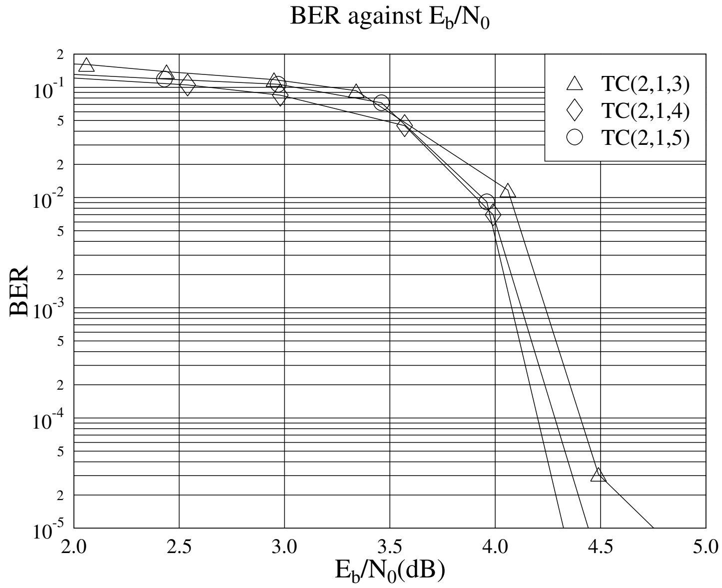


Figure 9.20: Performance comparison between the half-rate codes TC(2,1,3), TC(2,1,4) and TC(2,1,5), where the encoding parameters are shown in Table 9.3 and 9.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using **16QAM** over uncorrelated Rayleigh fading channels and the throughput was **2 BPS**.

In Figure 9.20 we compare the performance of the half-rate turbo codes TC(2,1,3), TC(2,1,4) and TC(2,1,5), where the encoding parameters are shown in Tables 9.3 and 9.5. The simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using 16QAM over uncorrelated Rayleigh fading channels. The three performance curves in the figure are the best performance curves chosen from Figures 9.17, 9.16 and 9.15 for the half-rate codes TC(2,1,5), TC(2,1,4) and TC(2,1,3), respectively. It can be seen from the figure that the performance of the turbo codes improves, when we increase the constraint length of the component codes from 3 to 5. However, this performance gain is obtained at the cost of a higher decoding complexity. At a BER of 10^{-5} the TC(2,1,4) code has an E_b/N_0 improvement of approximately 0.25 dB over the TC(2,1,3) scheme at a penalty of twice the complexity. However, at the cost of the same complexity increment over that of the TC(2,1,4) arrangement the TC(2,1,5) scheme only achieves a marginal performance gain of 0.1 dB at BER= 10^{-5}. Therefore, in our following investigations only the TC(2,1,4) scheme will be characterised as it exhibits a significant coding gain at a moderate complexity. Furthermore, the TC(2,1,4) code has been adopted by the 3G UTRA mobile communication system [56].

9.5.3.2 Comparison of Different Rate TC(2,1,4) Codes

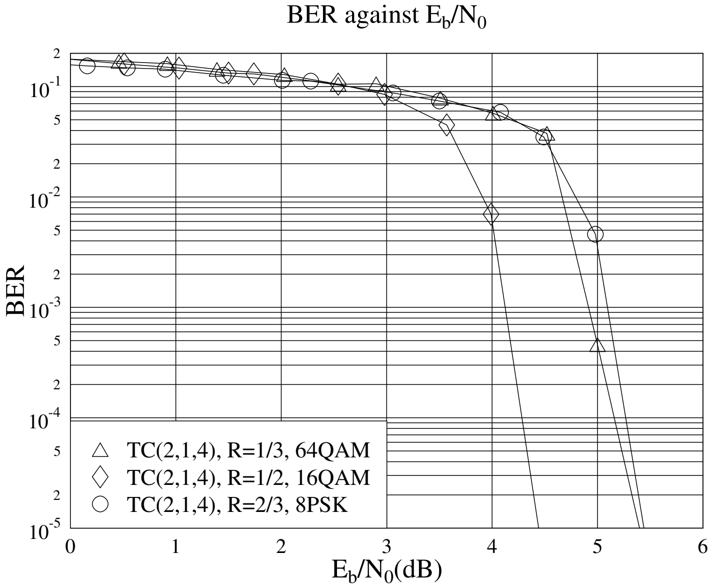


Figure 9.21: Performance of the TC(2,1,4) code using coding rates of $\frac{1}{3}$, $\frac{1}{2}$ and $\frac{2}{3}$, where the associated encoding parameters are shown in Tables 9.3 and 9.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at an effective throughput of **2 BPS** over uncorrelated Rayleigh fading channels.

In their seminal paper on turbo coding [21, 22], Berrou *et al.* applied alternate puncturing of the parity bits. This results in half-rate turbo codes. However, additionally a range of different puncturing patterns can be applied, which results in different code rates [71]. In Figure 9.21 we portray the performance of the punctured TC(2,1,4) code having coding rates of $\frac{1}{3}$, $\frac{1}{2}$ and $\frac{2}{3}$. The associated coding parameters are shown in Tables 9.3 and 9.5. Suitable multi-level modulation schemes are chosen so that all systems have the same effective throughput of 2 BPS. Explicitly, 64QAM, 16QAM and 8PSK are used. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ over uncorrelated Rayleigh fading channels. As expected, from Figure 9.21 we can clearly see that the best performance is achieved by the half-rate TC(2,1,4) scheme. At a BER of 10^{-5} the half-rate TC(2,1,4) code achieved a performance gain of approximately 1 dB over the third-rate and the two-third-rate TC(2,1,4) codes. Even though the third-rate TC(2,1,4) code has a higher amount of redundancy than the half-rate TC(2,1,4) scheme, its performance is worse, than that of the half-rate TC(2,1,4) arrangement. We speculate that this is because the constellation points in 64QAM are more densely packed, than those of 16QAM. Therefore, they are more prone to errors and hence the extra coding power of the third-rate TC(2,1,4) code is insufficient to correct the extra errors. This results in a poorer performance. On the other hand, there are less errors induced by 8PSK, but the two-third-rate TC(2,1,4) code is a weak code due to the puncturing of the parity bits. Again, this results in an inferior performance.

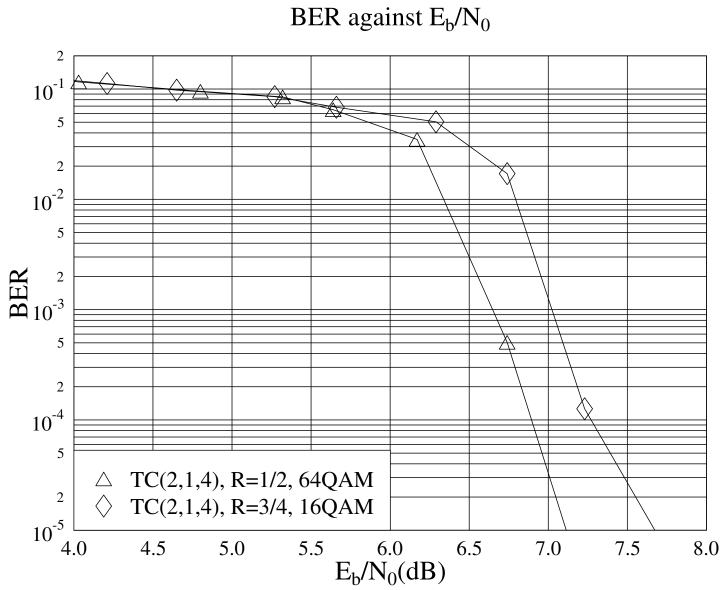


Figure 9.22: Performance of the punctured TC(2,1,4) code at coding rates of $\frac{1}{2}$ and $\frac{3}{4}$, where the associated parameters are shown in Tables 9.3 and 9.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at an effective throughput of **3 BPS** over uncorrelated Rayleigh fading channels.

In Figure 9.22 we show the performance of the TC(2,1,4) code at coding rates of $\frac{1}{2}$ and $\frac{3}{4}$. The associated coding parameters were shown in Tables 9.3 and 9.5. Again, suitable modulation schemes were chosen so that both systems have the same effective throughput, namely 3 BPS. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ over uncorrelated Rayleigh fading channels. As compared to Figure 9.21, the throughput of the systems in Figure 9.22 has been increased from 2 BPS to 3 BPS. In order to maintain a high BPS throughput, 64QAM was employed in conjunction with the half-rate TC(2,1,4) code. We can see from the figure that the performance gain of the half-rate TC(2,1,4) code over the three-quarter-rate TC(2,1,4) code has been reduced to only 0.5 dB, as compared to 1 dB over the two-third-rate TC(2,1,4) code characterised in Figure 9.21. Moreover, the three-quarter-rate TC(2,1,4) code is weaker, than the two-third-rate TC(2,1,4) code, since less parity bits are transmitted over the channel. Based on the fact that the performance gain of the half-rate TC(2,1,4) code has been reduced, we surmise that high-rate turbo codes will outperform the half-rate TC(2,1,4) code, if the throughput of the system is increased to 4 BPS or even further.

¿From Figures 9.21 and 9.22 we can see that the best performance is achieved by the half-rate TC(2,1,4) code for an effective throughput of 2 and 3 BPS. However, we are also interested in the system's performance at higher effective BPS throughputs. Hence, during our later discourse in Section 9.5.3.6 the performance of high-rate TC and TBCH codes will be studied for throughput values in excess of 5 BPS.

9.5.3.3 Convolutional Codes

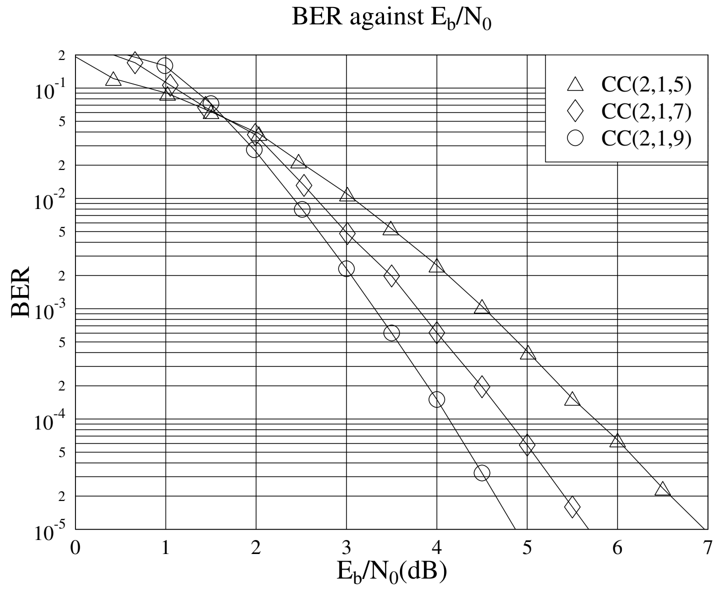


Figure 9.23: Performance comparison between the non-recursive half-rate convolutional codes CC(2,1,5), CC(2,1,7) and CC(2,1,9), where the coding parameters are shown in Tables 9.3 and 9.4. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using **QPSK** over uncorrelated Rayleigh fading channels. The effective throughput is **1BPS**

In Figure 9.23 we compare the performance of the $\mathbf{G}_2$ space-time coded non-recursive half-rate convolutional codes CC(2,1,5), CC(2,1,7) and CC(2,1,9). These schemes were standardised in the GSM [56, 287], DVB [46] and the 3G UTRA systems [56, 266, 289], respectively. The associated coding parameters were shown in Tables 9.3 and 9.4. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ using QPSK over uncorrelated Rayleigh fading channels. We can see from the figure that at a BER of 10^{-5} the performance of the non-recursive convolutional codes improves by approximately 1 dB, if the complexity is increased by a factor of $2^2 = 4$. However, the extra performance gain attainable becomes smaller, as the affordable complexity further increases. In our forthcoming channel code comparisons, only the CC(2,1,9) code will be used, since it has the best performance amongst the above three schemes and it has a comparable complexity to that of the turbo convolutional codes studied. Moreover, the CC(2,1,9) code is also proposed for the third generation UTRA mobile communication system [56].

9.5.3.4 $\mathbf{G}_2$ Coded Channel Codec Comparison − Throughput of 2 BPS

Having narrowed down the choice of the $\mathbf{G}_2$ space-time coded convolutional codes and the turbo codes, we are now ready to compare the performance of the different proposed channel codecs belonging to different codec families. Our comparison is carried out on the basis of the

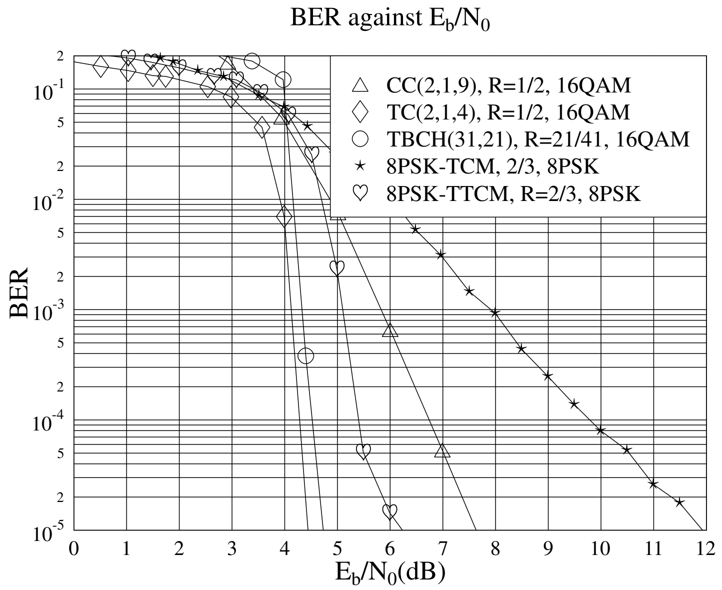


Figure 9.24: Performance comparison between different CC, TC, TBCH, TCM and TTCM schemes where the coding parameters are shown in Tables 9.3, 9.4 and 9.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at a throughput of **2 BPS** over uncorrelated Rayleigh fading channels.

same throughput and all channel codecs are concatenated with the space-time code $\mathbf{G}_2$, when transmitting over uncorrelated Rayleigh fading channels. Figure 9.24 shows the performance of our channel codecs selected from the CC, TC, TBCH, TCM and TTCM families on the basis of the same throughput of 2 BPS, regardless of their coding rates. The associated coding parameters are shown in Tables 9.3, 9.4 and 9.5. The throughput is 2 BPS.

¿From Figure 9.24 we can see that the half-rate TC(2,1,4) code outperforms the other channel codecs. At a BER of 10^{-5} the TC(2,1,4) code achieves a gain of approximately 0.5 dB over the TBCH(31,21) scheme at a much lower complexity. At the same BER, the TC(2,1,4) code also outperforms 8PSK-TTCM by approximately 1.5 dB. The poor performance of TTCM might be partially due to using generator polynomials, which are optimum for AWGN channels [64]. However, to date only limited research has been carried out on finding optimum generator polynomials for TTCM over fading channels [290].

In Figure 9.24 we also characterise the performance of the CC(2,1,9) and 8PSK-TCM schemes. The figure clearly demonstrates that the invention of turbo codes invoked in our TC, TBCH and TTCM $\mathbf{G}_2$-coded schemes, resulted in substantial improvements over the conventional $\mathbf{G}_2$-coded channel codecs, such as the CC and TCM schemes considered. At a BER of 10^{-5}, the TC(2,1,4) code outperforms the CC(2,1,9) and 8PSK-TCM arrangements by approximately 3.0 dB and 7.5 dB, respectively.

9.5.3.5 G_2-Coded Channel Codec Comparison − Throughput of 3 BPS

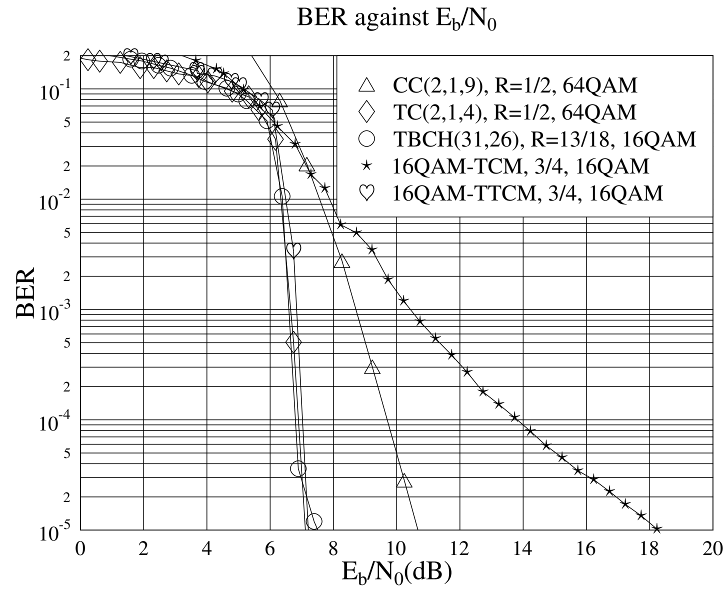


Figure 9.25: Performance comparison between different CC, TC, TBCH, TCM and TTCM schemes where the coding parameters are shown in Tables 9.3, 9.4 and 9.5. All simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ at an effective throughput of **3 BPS** over uncorrelated Rayleigh fading channels.

In Figure 9.25 we portray the performance of various channel codecs belonging to the CC, TC, TBCH, TCM and TTCM codec families on the basis of a constant throughput of 3 BPS, regardless of their coding rates. The associated coding parameters are shown in Tables 9.3, 9.4 and 9.5. The simulation results were obtained with the aid of the space-time code $\mathbf{G}_2$ over uncorrelated Rayleigh fading channels.

¿From Figure 9.25 we can infer a few interesting points. As mentioned earlier, the half-rate TC(2,1,4) code suffers from the effects of puncturing as we increase the throughput of the system. In order to maintain a throughput of 3 BPS, 64QAM has to be employed in the systems using the half-rate TC(2,1,4) code. The rather vulnerable 64QAM modulation scheme appears to over-stretch the coding power of the half-rate TC(2,1,4) code attempting to saturate the available channel capacity. At a BER of 10^{-5} there is no obvious performance gain over the TBCH(31,26)/16QAM and 16QAM-TTCM schemes. Hence, we have reasons to postulate that if the throughput of the system is increased beyond 3 BPS, high-rate turbo codes should be employed for improving the performance, rather than invoking a higher throughput modulation scheme.

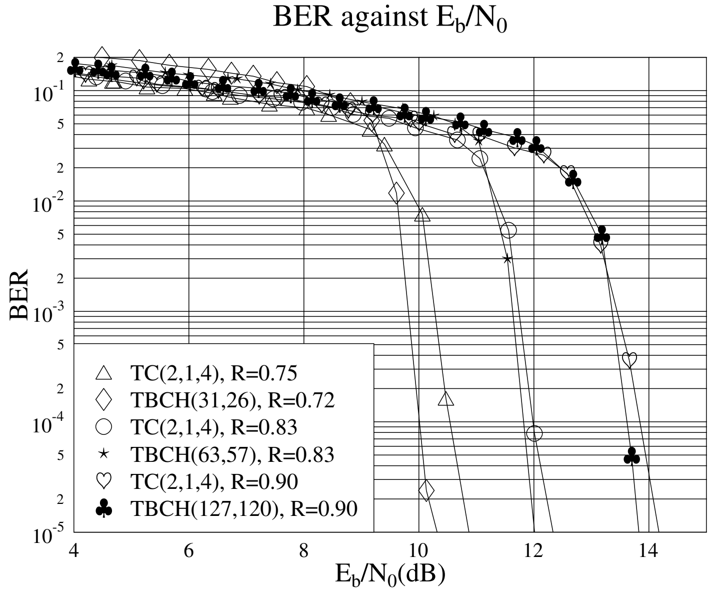


Figure 9.26: Performance comparison between high-rate TC and TBCH codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The parameters of the TC and TBCH codes were shown in Tables 9.3 and 9.5.

9.5.3.6 Comparison of G_2-Coded High-Rate TC and TBCH Codes

In the previous section we have shown that at the BER of 10^{-5}, the required E_b/N_0 is increased by about 2.5 dB for the half-rate turbo code TC(2,1,4), as the throughput of the system is increased from 2 BPS to 3 BPS. A range of schemes having a throughput in excess of 5 BPS is characterised in Figure 9.26. Specifically, the figure shows the performance of high-rate TC and TBCH codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The parameters of the TC and TBCH codes used are shown in Tables 9.3 and 9.5. The performance of half-rate turbo codes along with such a high throughput is not shown, because a modulation scheme having at least 1024 constellation points would be needed, which is practically infeasible over non-stationary wireless channels. Moreover, the turbo codes often would be overloaded with the plethora of errors induced by the densely packed constellation points.

In Figure 9.26 we can clearly see that there is not much difference in performance terms between the high-rate TC(2,1,4) and TBCH codes employed, although the TBCH codes exhibit marginal gains. This gain is achieved at a cost of high decoding complexity, as evidenced by Table 9.6. The slight performance improvement of the TBCH(31,26) code over the three-quarter rate TC(2,1,4) scheme is probably due to its slightly lower code rate of $R = 0.72$, compared to the rate of $R = 0.75$ associated with the TC(2,1,4) code. It is important to note that all BCH component codes used in the TBCH codes have a minimum distance d_{min} of 3. We speculate that the performance of the TBCH codes might improve, if d_{min} is increased to 5. However, due to the associated complexity we will refrain from employing $d_{min} = 5$

BCH component codes in the TBCH schemes studied.

9.5.3.7 Comparison of High-Rate TC and Convolutional Codes

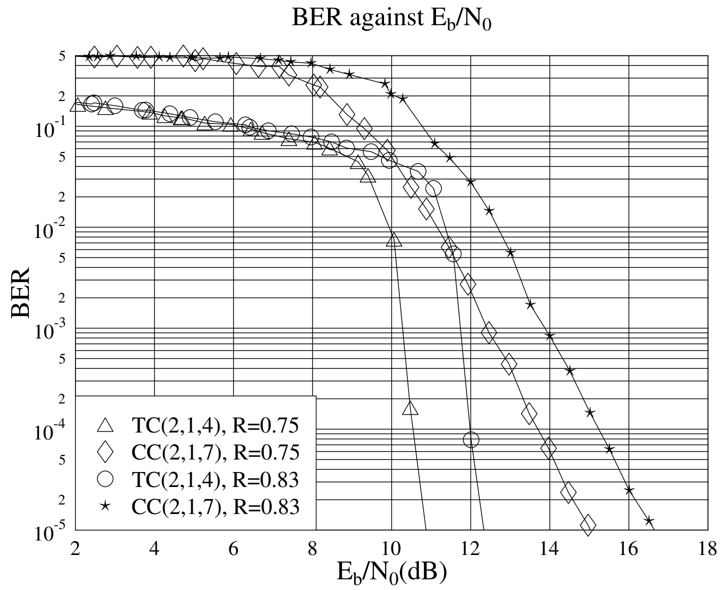


Figure 9.27: Performance comparison between high-rate TCs and convolutional codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The parameters of the TC and CC codes were shown in Tables 9.3, 9.4 and 9.5.

In Figure 9.27, we compare the performance of the high-rate punctured TC(2,1,4) and CC(2,1,7) codes concatenated with the space-time code $\mathbf{G}_2$ employing 64QAM over uncorrelated Rayleigh fading channels. The puncturing patterns employed for the CC(2,1,7) scheme were proposed in the DVB standard [46]. The parameters of the TC(2,1,4) and CC(2,1,7) codes are shown in Tables 9.3, 9.4 and 9.5. From the figure we can see that both high-rate TC(2,1,4) codes outperform their equivalent rate CC(2,1,7) counterparts by about 2 dB at a BER of 10^{-5}, whilst maintaining a similar estimated decoding complexity, as it was evidenced by Table 9.6. This fact indicates that at a given tolerable complexity, better BER performance can be attained by an iterative turbo decoder. These findings motivated the investigations of our next section, where the performance of the various schemes was studied in the context of the achievable coding gain versus the estimated decoding complexity.

9.5.4 Coding Gain Versus Complexity

In Section 9.4.3 we have estimated the various channel decoders' complexity based on a few implifying assumptions. All the complexities estimated in our forthcoming discourse were calculated based on Equations 9.43 to 9.51. Again, our performance comparison of the channel codes was made on the basis of the coding gain defined as the E_b/N_0 difference, expressed in decibels, at BER= 10^{-5} between the various channel coded and uncoded systems having the same throughput, while using the space-time code $\mathbf{G}_2$.

9.5.4.1 Complexity Comparison of Turbo Convolutional Codes

Figure 9.28 shows the (a) coding gain versus the number of iterations and (b) the coding gain versus estimated complexity for the TC(2,1,3), TC(2,1,4) and TC(2,1,5) codes, where the coding parameters used are shown in Tables 9.3, 9.5 and 9.6. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver and 64QAM over uncorrelated Rayleigh fading channels at an effective throughput of 3 BPS. We can see from Figure 9.28(a) that there is a huge performance improvement of approximately $3 - 4$ dB between the first and second turbo decoding iteration. However, the further coding gain improvements become smaller, as the number of iterations increases. It can be seen from the figure that the performance of turbo codes does not significantly improve after 8 iterations, as indicated by the rather flat coding gain curve. Figure 9.28(a) also shows that as we increase the constraint length K of the turbo codes from 3 to 5, the associated performance improves.

In Figure 9.28(b) the coding gains of the various turbo codes using different number of iterations were compared on the basis of their estimated complexity. This was necessary, since we have seen in Section 9.4.3 that the estimated complexity of turbo codes depends exponentially on the constraint length K, but only linearly on the number of iterations. ¿From Figure 9.28(b), we can see that the estimated complexity of the TC(2,1,5) code ranges from approximately 200 to 2000, when using one to ten iterations. On the other hand, the estimated complexity of the TC(2,1,3) scheme ranges only from approximately 50 to 500 upon invoking one to ten iterations. This clearly shows that the estimated complexity of the turbo codes is dominated by the constraint length K. Figure 9.28(b) also shows that the coding gain curve of the TC(2,1,3) code saturates faster, which is demonstrated by the steep increase in coding gain, as the estimated complexity increases. For achieving the same coding gain of 19 dB, we can see that the TC(2,1,3) scheme requires the lowest estimated complexity. We would require 2-3 times higher computational power for the TC(2,1,5) code to achieve the above-mentioned coding gain of 19 dB.

9.5.4.2 Complexity Comparison of Channel Codes

In the previous section we have compared the coding gain versus estimated complexity of the $\mathbf{G}_2$-coded turbo schemes TC(2,1,3), TC(2,1,4) and TC(2,1,5). Here we compare the TC(2,1,4) arrangement that faired best amongst them to the CC(2,1,9) code and to the TBCH(32,26)/8PSK as well as to the TTCM-8PSK arrangements, representing the other codec families studied. Specifically, Figure 9.29 shows the coding gain versus estimated complexity for the CC(2,1,K), TC(2,1,4), TBCH(32,26) and TTCM-8PSK schemes, where the associated parameters are shown in Tables 9.3, 9.4, 9.5 and 9.6. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of 2 BPS. For the turbo schemes TC(2,1,4), TBCH(32,26) and TTCM-8PSK the increased estimated complexity is achieved by increasing the number of iterations from 1 to 10. However, convolutional codes are decoded non-iteratively. Therefore in Figure 9.29 we vary the constraint length K of the convo-

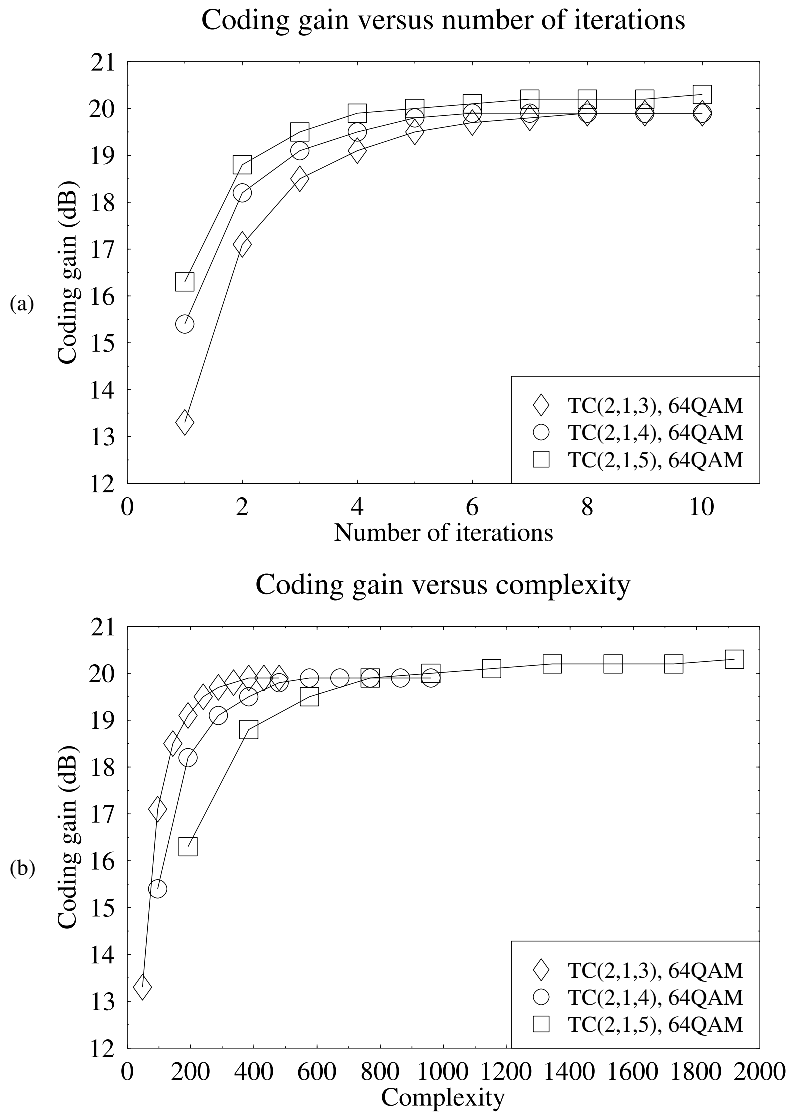


Figure 9.28: Coding gain versus (a) the number iterations and versus (b) estimated complexity for the TC(2,1,3), TC(2,1,4) and TC(2,1,5) codes, where the coding parameters are shown in Tables 9.3, 9.5 and 9.6. All simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver and **64QAM** over uncorrelated Rayleigh fading channels at an effective throughput of **3 BPS**.

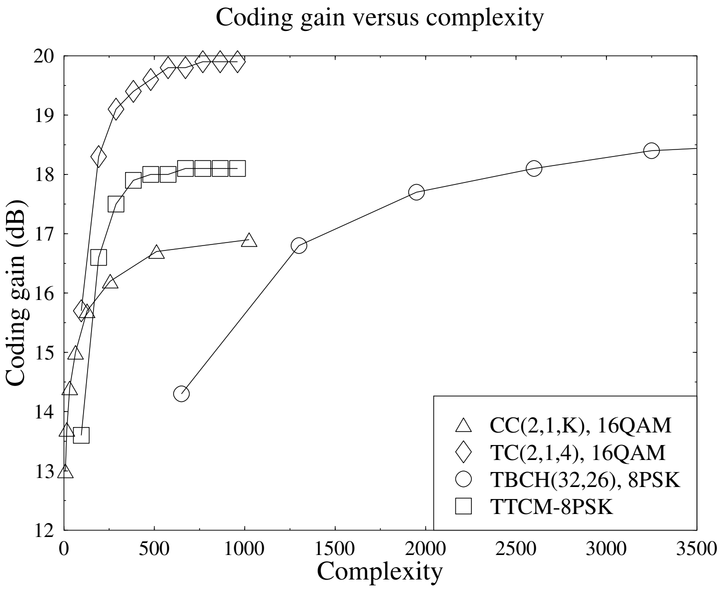


Figure 9.29: Coding gain versus estimated complexity for the CC(2,1,K), TC(2,1,4), TBCH(32,26) and TTCM-8PSK where the parameters are shown in Table 9.3, 9.4, 9.5 and 9.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of **2 BPS**.

lutional codes from 3 to 10, which results in increased estimated complexity. The generator polynomials of the CC(2,1,K) codec, where $K = 3...10$, are given in [104] and they define the corresponding maximum minimum free distance of the codes. From Figure 9.29 we can see that there is a steep increase in the coding gain achieved by the TC(2,1,4) code, as the estimated complexity is increased. Moreover, the TC(2,1,4) scheme asymptotically achieves a maximum coding gain of approximately 20 dB. At a low estimated complexity of approximately 200, the TC(2,1,4) code attains a coding gain of approximately 18 dB, which exceeds that of the other channel codes studied. The TBCH(32,26) arrangement is the least attractive one, since a huge estimated complexity is incurred, when aiming for a high coding gain.

In contrast to the 2 BPS schemes of Figure 9.29, Figure 9.30 shows the corresponding coding gain versus estimated complexity curves for the CC(2,1,K), TC(2,1,4), TBCH(31,26) and TTCM-16QAM 3 BPS arrangements, where the coding parameters are shown in Tables 9.3, 9.4, 9.5 and 9.6. Again, all simulation results were obtained upon employing the space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of 3 BPS. As before, the increased estimated complexity of the turbo schemes is incurred by increasing the number of iterations from 1 to 10. For the convolutional codes the constraint length K is varied from 3 to 10. Similarly to Figure 9.29, the TC(2,1,4) scheme achieves a considerable coding gain at a relatively low estimated complexity. For example, in order to achieve a coding gain of 18 dB, the TTCM and TBCH(31,26) arrangements would require an approximately 3 and 4 times higher computational power

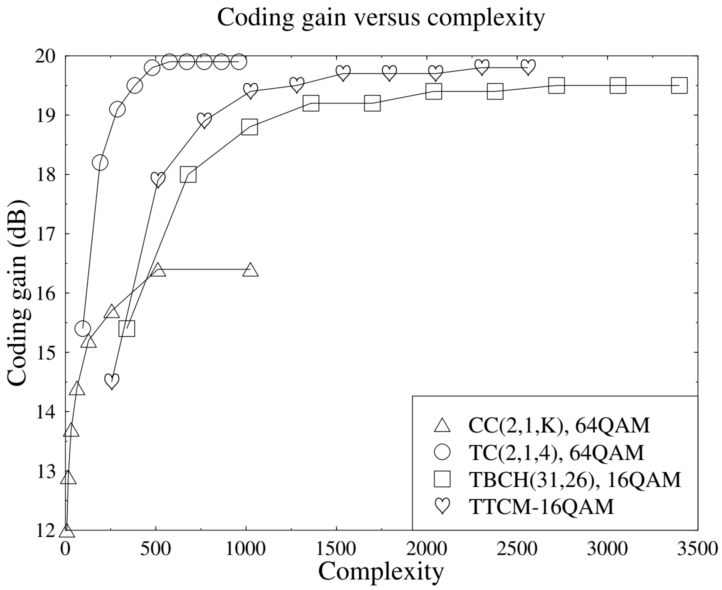


Figure 9.30: Coding gain versus estimated complexity for the CC(2,1,K), TC(2,1,4), TBCH(31,26) and TTCM-16QAM schemes where the coding parameters are shown in Tables 9.3, 9.4, 9.5 and 9.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using one receiver over uncorrelated Rayleigh fading channels at an effective throughput of **3 BPS**.

compared to the TC(2,1,4) code.

¿From Figures 9.29 and 9.30 we can clearly see that turbo codes are the most attractive one of all the channel codes studied in conjunction with the space-time code $\mathbf{G}_2$, offering an impressive coding gain at a moderate estimated decoding complexity.

In Figure 9.31, we show the E_b/N_0 value required for maintaining BER= 10^{-5} versus the effective throughput BPS for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,4) code where the coding parameters are shown in Tables 9.3, 9.5 and 9.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using **one receiver** over uncorrelated Rayleigh fading channels. Half-rate TC(2,1,4) code was employed for BPS up to three. Then TC(2,1,4) code with various rates was employed with 64QAM in order to achive increasing effective throughput BPS. It can be seen from the figure that the E_b/N_0 value required for maintaining BER= 10^{-5} increases linearly as the effective throughput BPS increases.

9.6 Summary and Conclusions

The state-of-the-art of transmission schemes based on multiple transmitters and receivers was reviewed in Section 9.1 This was followed by a rudimentary introduction to MRC [78] technique, using a simple example in Section 9.2.1. Space-time block codes were introduced in

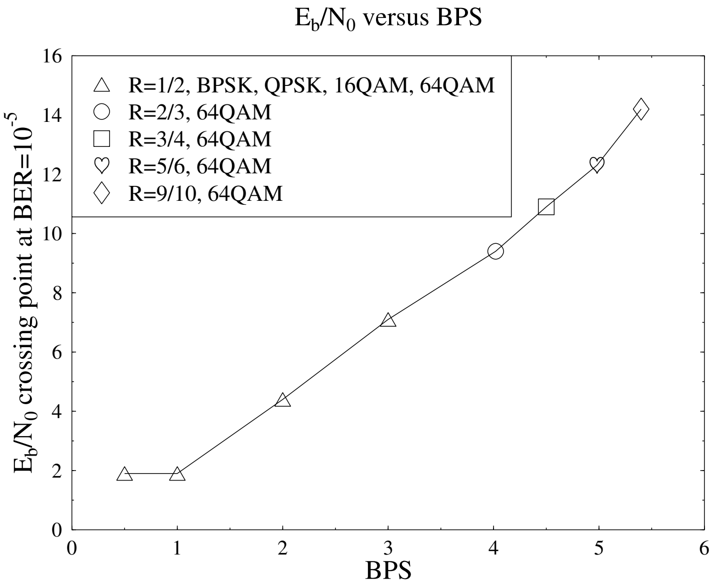


Figure 9.31: The E_b/N_0 value required for maintaining BER= 10^{-5} versus the effective throughput BPS for the space-time block code $\mathbf{G}_2$ concatenated with the TC(2,1,4) code where the coding parameters are shown in Tables 9.3, 9.5 and 9.6. All simulation results were obtained upon employing space-time code $\mathbf{G}_2$ using **one receiver** over uncorrelated Rayleigh fading channels.

Section 9.3 employing the unity-rate space-time code $\mathbf{G}_2$. In Sections 9.3.1.1 and 9.3.1.2 two examples of employing the space-time code $\mathbf{G}_2$ were provided using one and two receivers, respectively. The transmission matrix of a range of different-rate space-time codes, namely that of the codes $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ of Table 9.2 were also given. Additionally, a brief description of the MAP decoding algorithm [283] was provided in Section 9.3.3 in the context of space-time block codes.

In Section 9.4 we proposed a system, which consists of the concatenation of the abovementioned space-time block codes and a range of different channel codes. The channel coding schemes investigated were convolutional codes, turbo convolutional codes, turbo BCH codes, trellis coded modulation and turbo trellis coded modulation. The estimated complexity and memory requirement of the channel decoders were summarised in Section 9.4.3.

Finally, we presented our simulation results in Section 9.5, which were divided into four categories. In Section 9.5.1, we first compared the performance results of the space-time codes $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{H}_3$ and $\mathbf{H}_4$ without using channel codecs. It was found that as we increased the effective throughput of the system, the performance of the half-rate space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$ degraded in comparison to that of the unity rate space-time code $\mathbf{G}_2$. This was because in order to maintain the same effective throughput, higher modulation schemes had to be employed in conjunction with the half-rate space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$, which were more prone to errors and hence degraded the performance of the system. On the other hand, for the sake of maintaining the same diversity gain and same effective throughput we found that the performance of the space-time codes $\mathbf{H}_3$ and $\mathbf{H}_4$ was better, than that of the space-time codes $\mathbf{G}_3$ and $\mathbf{G}_4$, respectively. Since the space-time code $\mathbf{G}_2$ has a code rate of unity, we were able to concatenate it with half-rate TC codes, while maintaining the same effective throughput, as the half-rate space-time code using no channel coding. Hence for the same effective throughput, the unity-rate $\mathbf{G}_2$ space-time coded and half-rate channel coded scheme provided substantial performance improvement over the three-quarter rate space-time code $\mathbf{H}_4$ and half-rate space-time code $\mathbf{G}_4$, which were unable to benefit from channel coding. We concluded that the reduction in coding rate was best invested in turbo channel codes, rather than space-time block codes. Therefore, all channel codes studied were concatenated with the unity-rate space-time code $\mathbf{G}_2$ only.

In the second category of our investigations in Section 9.5.1.5 we studied the effect of the binary channel codes' data and parity bits mapped into different protection classes of multi-level modulation schemes. It was found that TC codes having different constraint lengths K require different mapping methods, as evidenced by Figure 9.12. By contrast, in the turbo BCH codes studied mapping of the parity bits to the higher-integrity protection class of a multi-level modulation scheme yielded a better performance. The so-called random separation based interleaver was proposed, in order to improve the performance of the system.

The third set of results compared the performance of all proposed channel codes in conjunction with the space-time code $\mathbf{G}_2$. In order to avoid confusion, we only selected one channel code from each group of channel codes in Table 9.3. Specifically, only half-rate TC codes were studied, as they gave better coding gain performance compared to other TC codes having lower and higher rates. It was then found that the performance of the half-rate TC codes was better than that of the CC, TBCH, TCM and TTCM codes. Then, we compared the performance of high rates TC codes with high-rate turbo BCH codes in conjunction with 64QAM. It was found that the turbo BCH codes provided a slight performance improvement over high rate TC codes, but at the cost of high complexity. Finally, the chapter was concluded by comparing the $\mathbf{G}_2$ space-time coded channel codes upon taking their estimated complexity into consideration. In Figures 9.29 and 9.30, we can clearly see that the half-rate TC codes give the best coding gain at a moderate estimated complexity.

Following our discussions on space-time block codes in this chapter, which have been contrived for communications over non-dispersive wireless channels, in the next chapter space-time trellis codes are discussed.

Part V

Turbo Equalisation

Chapter 15

Comparative Study of Turbo Equalisers

15.1 Motivation[1]

In Chapter 13, 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 14 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 $\mathrm{BER} > 10^{-5}$. The theoretical ML bounds of the convolutional-coded and turbo-coded DPSK systems in Chapter 14 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 $\mathrm{BER} = 10^{-4}$ and $\mathrm{BER} = 10^{-3}$, respectively and employing turbo equalisation — that convolutional codes are more robust, compared to the investigated turbo-coded schemes. In this chapter a sound appreciation of the concept presented in Chapter 13 constitutes a prerequisite, although useful information system performance-related information may be gleaned without understanding the associated iterative equalisation principles.

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 14.3. In addition to convolutional codes and convolutional-coding based turbo codes, block-coding based turbo codes are also researched in conjunction with BPSK systems util-

[1] This chapter is based on B.L. Yeap, T.H. Liew, J.Hámorský and L. Hanzo: Comparative study of turbo equalization schemes using convolutional, convolutional turbo and block-turbo codes, to appear in IEEE Tr. on Wireless Communications, 2002

ising 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.* [68] 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 [70]. In reference [123] 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** $R = \frac{1}{2}$ [107, 108]. Specifically, Bose-Chaudhuri-Hocquengham (BCH) codes [23, 24] 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 15.2 provides an overview of the systems researched. Subsequently, Section 15.3 summarises the simulation parameters. Finally, Section 15.4 provides results and discussions, while Section 15.5 summarises the the systems' performance.

15.2 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 [17, 18] or turbo equalisation. The turbo equalisation operation is based on the principles described in Section 13.2 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 [21, 123]. In these systems turbo equalisation is also based on the principles of Section 13.2, 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.

15.3 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 13.7. 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 [361] specified in Table 15.1. For a fair comparative 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 [71]. 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 13.7. 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 15.2 — 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. 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 15.3. 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 [75] — 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 15.3.

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 15.1: DVB puncturing pattern [361] 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.

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 [339], since the Log-MAP algorithm achieves identical performance to the original Maximum *A Posteriori* (MAP) algorithm [20], 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

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

Table 15.2: 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.

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 15.1
Conv Rate $R = \frac{5}{6} = 0.833$	None	20736	See Table 15.1
Turbo Conv Rate $R = \frac{1}{2} = 0.5$	10368	20736	See Table 15.2
Turbo Conv Rate $R = \frac{3}{4} = 0.75$	15552	20736	See Table 15.2
Turbo Conv Rate $R = \frac{5}{6} = 0.833$	17280	20736	See Table 15.2
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 15.3: 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 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 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 :

$$\text{Number of states in decoder trellis} = \frac{\text{Encoder Input Block Length}}{k} \cdot (2k - n + 3) \cdot 2^{n-k}. \tag{15.1}$$

Table 15.4 shows the turbo equaliser complexity of each turbo equalisation iteration, for the three different classes of encoders.

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 15.4: 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).

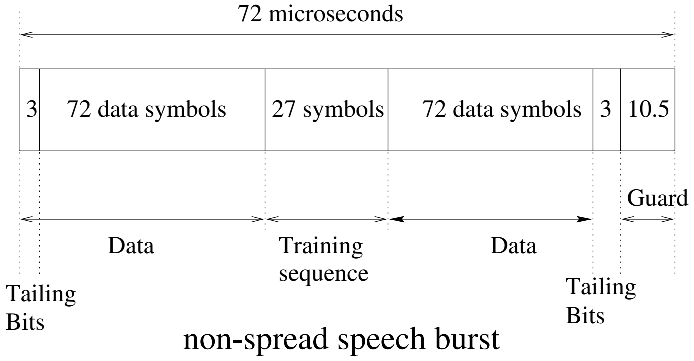
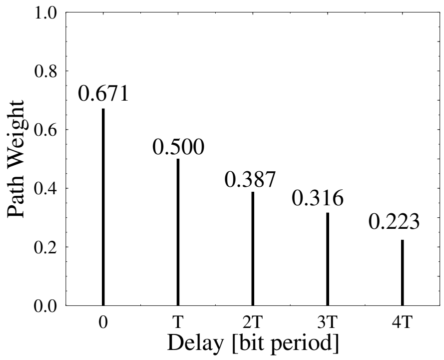
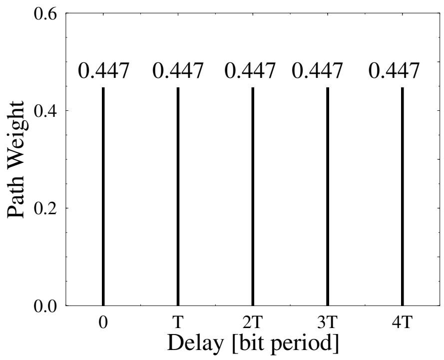


Figure 15.1: Transmission burst structure of the FMA1 non-spread speech burst of the FRAMES proposal [252].

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 [252] and shown in Figure 15.1. Our comparative study was conducted over the five-path Gaussian channel and the



(a) Five-path Gaussian channel



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

Figure 15.2: Channel impulse response of the five-path Gaussian channel and the equally-weighted five-path Rayleigh fading channel.

equally-weighted five-path Rayleigh fading channel using a normalised Doppler frequency of $f_d = 1.5{\times}10^{-4}$, as illustrated in Figures 15.2(a) and 15.2(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.

15.4 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 15.2(a) and 15.2(b), respectively.

15.4.1 Five-path Gaussian Channel

Figure 15.3 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 15.2(a). We 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

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 15.5: Ranking of the BPSK turbo equalisation performance for all systems over the five-path Gaussian channel from Figure 15.2(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'.

for $R \approx \frac{3}{4}$ BPSK turbo equalisers in Figure 15.4. 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 15.5 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 $= 10^{-4}$ 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

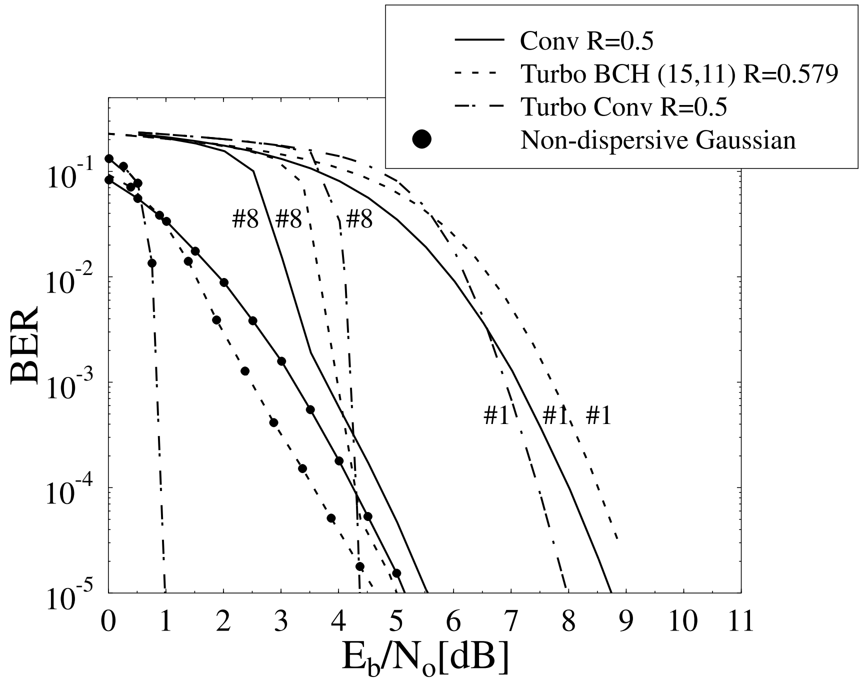


Figure 15.3: Comparing the BPSK turbo equalisation performance of the $R = \frac{1}{2}$ **CC** system, the $R = \frac{1}{2}$ **CT** scheme and the $R = \frac{11}{19}$ **BT** system for one ($\#1$) and eight ($\#8$) turbo equalisation iterations, over the five-path Gaussian channel of Figure 15.2(a). The decoding performance over the non-dispersive Gaussian channel — *i.e* the lower bound performance — is shown as well.

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 15.5, 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 15.2(a). Note that for the concatenated-coded **BT** and **CT** schemes, turbo equalisation and turbo decoding have the same processing sequence when only one iteration is implemented, hence giving the same performance. From Figures 15.6(a), 15.6(b) and 15.6(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 15.7(a), 15.7(b) and 15.7(c), we observed the same trend, where gains of 3.0 dB, 1.4 dB and 0.7 dB were

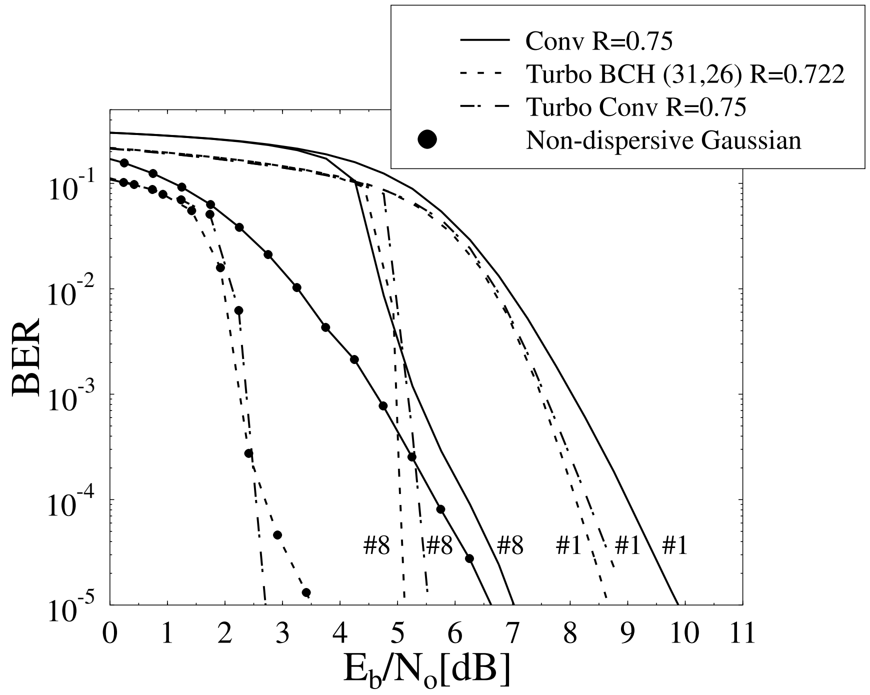


Figure 15.4: Comparing the BPSK turbo equalisation performance of the $R = \frac{3}{4}$ **CC** system, the $R = \frac{3}{4}$ **CT** scheme and the $R = \frac{26}{36}$ **BT** system for one $(\#1)$ and eight $(\#8)$ turbo equalisation iterations, over the five-path Gaussian channel of Figure 15.2(a). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

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 15.3, 15.4 and 15.5 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.

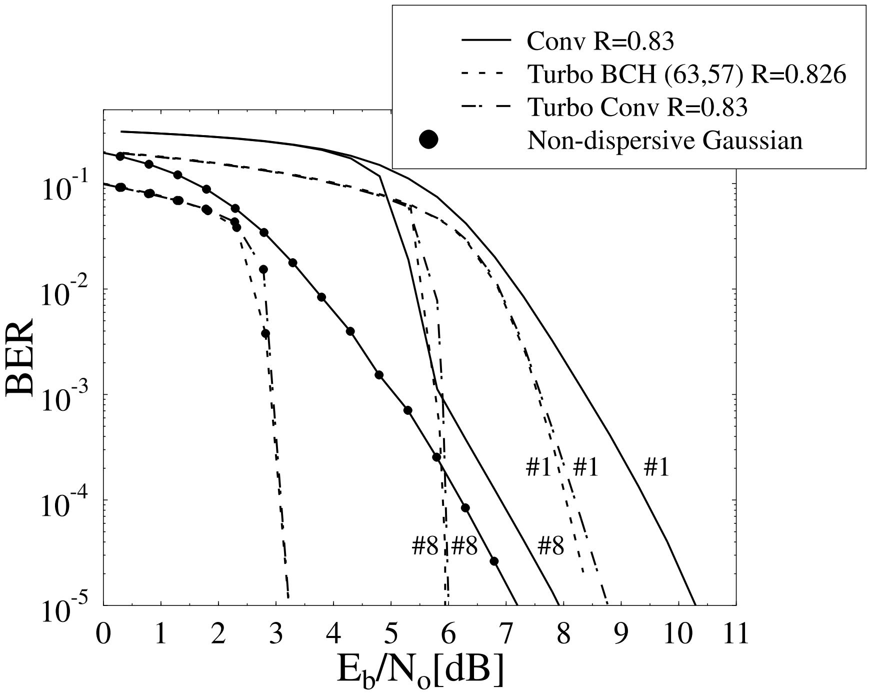


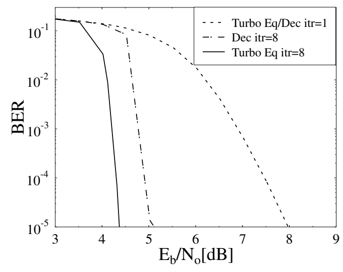
Figure 15.5: Comparing the BPSK turbo equalisation performance of the $R = \frac{5}{6}$ **CC** system, the $R = \frac{5}{6}$ **CT** scheme and the $R = \frac{57}{69}$ **BT** system for one $(\#1)$ and eight $(\#8)$ turbo equalisation iterations, over the five-path Gaussian channel of Figure 15.2(a). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

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)

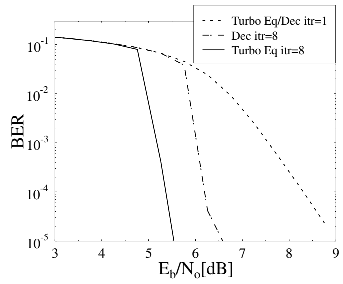
Table 15.6: Ranking of the BPSK turbo equalisation performance for all systems for the five-path Rayleigh fading channel using burst-invariant fading in Figure 15.2(b). The notation '1' represents the system that required the lowest E_b/N_o value and '3' for the system that needed the highest E_b/N_o 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'.

15.4.2 Equally-weighted Five-path Rayleigh Fading Channel

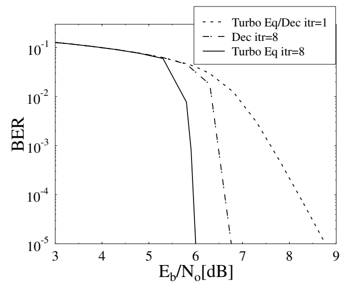
We now compare the turbo equalisation performance of the **CC**, **CT** and **BT** systems, for code rates of $R \approx \frac{1}{2}$, $R \approx \frac{3}{4}$ and $R \approx \frac{5}{6}$ over the five-path Rayleigh fading channel using burst-invariant fading depicted in Figure 15.2(b).



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

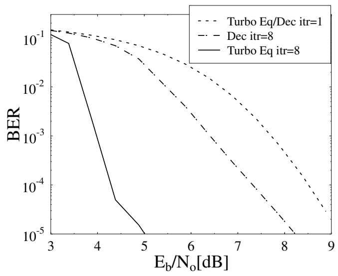


(b) $R = \frac{3}{4}$

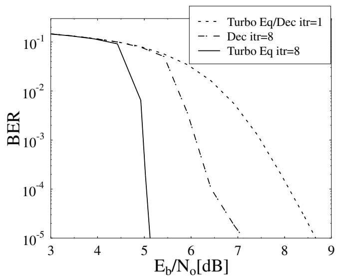


(c) $R = \frac{5}{6}$

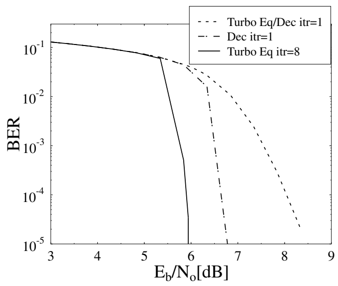
Figure 15.6: Decoding performance of the BPSK **CT** system using isolated turbo decoding compared with the turbo equalisation performance after the first iteration — which is identical for both — and after the eighth iteration for code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$, over the five-path Gaussian channel of Figure 15.2(a).



(a) $R = 0.579$



(b) $R = 0.722$



(c) $R = 0.826$

Figure 15.7: 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 15.2(a).

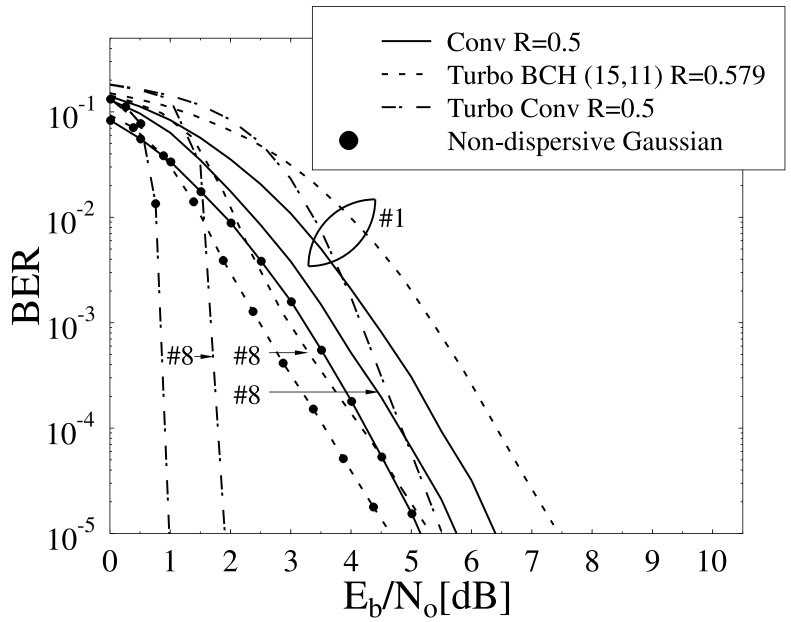


Figure 15.8: Comparing the BPSK turbo equalisation performance of the $R = \frac{1}{2}$ **CC** system, the $R = \frac{1}{2}$ **CT** scheme and of the $R = \frac{11}{19}$ **BT** system for one (#1) and eight (#8) turbo equalisation iterations, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 15.2(b). The decoding performance over the non-dispersive Gaussian channel — *i.e* the lower bound performance — is shown as well.

As shown in Figure 15.8, the $R = \frac{1}{2}$ **CT** system achieved a significant gain of 2.4 dB and 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 15.9 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 15.10 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

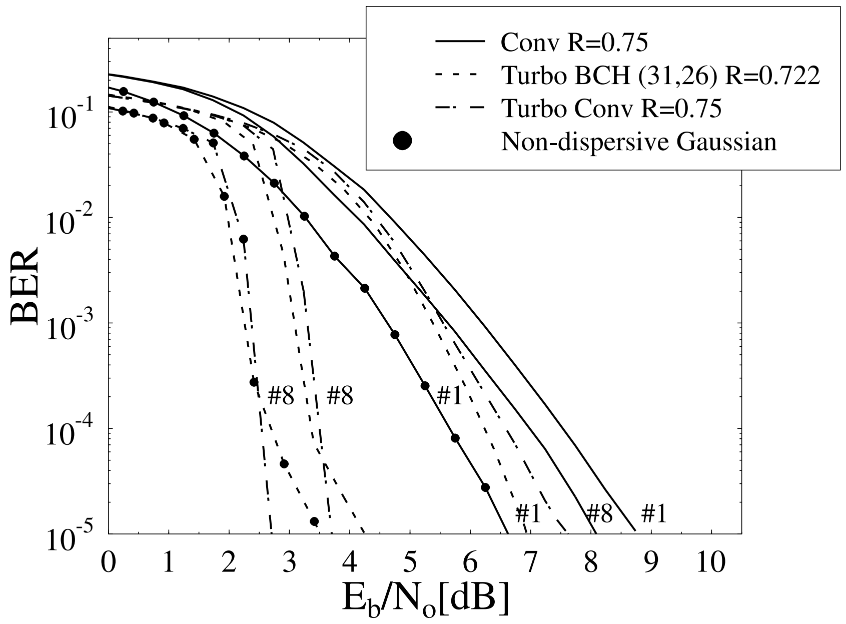


Figure 15.9: Comparing the BPSK turbo equalisation performance of the $R = \frac{3}{4}$ **CC** system, the $R = \frac{3}{4}$ **CT** scheme and the $R = \frac{26}{36}$ **BT** system for one $(\#1)$ and eight $(\#8)$ turbo equalisation iterations, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 15.2(b). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

was obtained. In Table 15.6 we ranked the **CC**, **CT** and **BT** systems, according to the 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 15.2(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 15.11(a), 15.11(b) and 15.11(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 15.12(a), 15.12(b) and 15.12(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 15.8, 15.9 and 15.10 showed gains of 0.7 dB, 0.5 dB and 0.5 dB, which were achieved by employing turbo equalisation

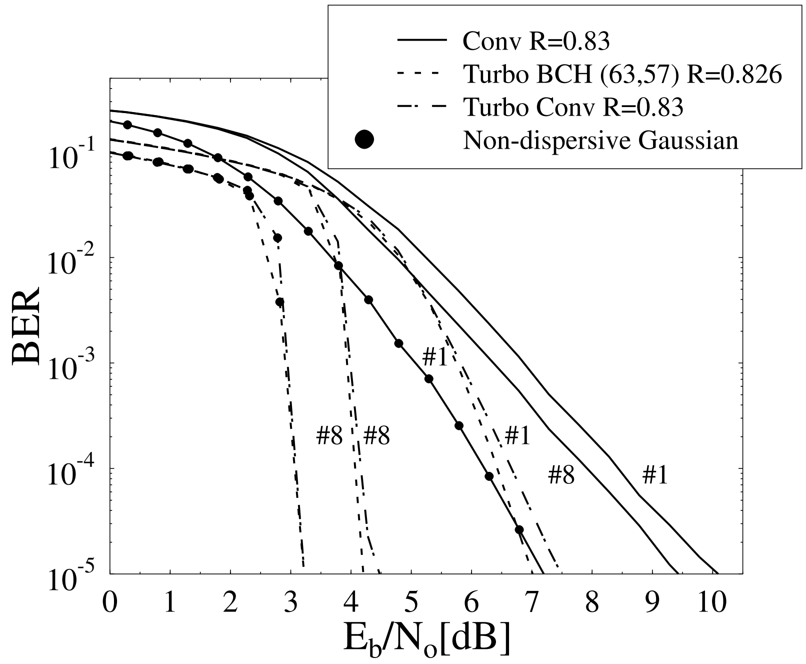
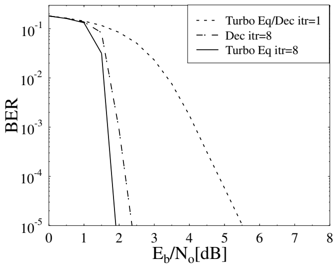


Figure 15.10: Comparing the BPSK turbo equalisation performance of the $R = \frac{5}{6}$ **CC** system, the $R = \frac{5}{6}$ **CT** scheme and the $R = \frac{57}{69}$ **BT** system for one ($\#1$) and eight ($\#8$) turbo equalisation iterations, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 15.2(b). The decoding performance over the non-dispersive Gaussian channel, *i.e* the lower bound performance, is shown as well.

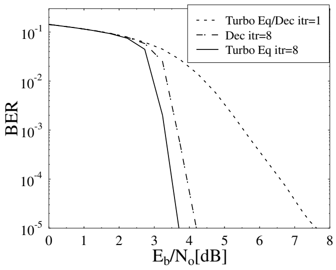
instead of convolutional decoding at BER $= 10^{-4}$ for the **CC** system at code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$, respectively.

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.

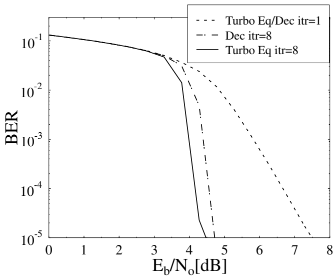
The turbo equalisation simulations over the five-path Rayleigh fading channel of Figure 15.2(b) also showed that the **CC** system has poor iteration gain — *i.e.* a modest gain in E_b/N_o — performance with respect to the first iteration (consistent with the results presented for the $R = \frac{1}{2}$ **CC** system in references [107, 342]). 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 15.10. At this BER, the $R = \frac{5}{6}$ **CC** system only achieves an iteration gain of 0.5 dB after eight turbo



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

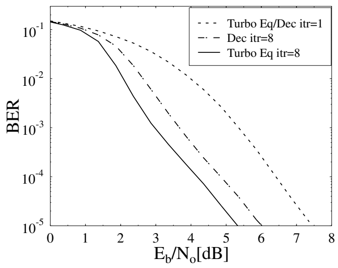


(b) $R = \frac{3}{4}$

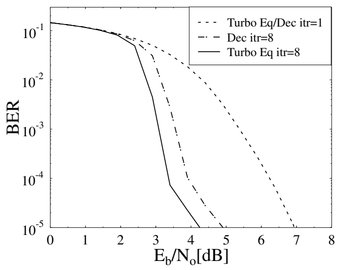


(c) $R = \frac{5}{6}$

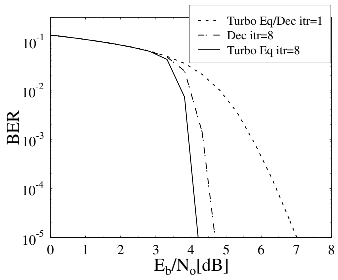
Figure 15.11: Decoding performance of the BPSK **CT** system using isolated turbo decoding compared with the turbo equalisation performance after the first iteration — which is identical for both — and after the eighth iteration for code rates of $R = \frac{1}{2}$, $R = \frac{3}{4}$ and $R = \frac{5}{6}$, over the five-path Rayleigh fading channel using burst-invariant fading illustrated in Figure 15.2(b).



(a) $R = 0.579$



(b) $R = 0.722$



(c) $R = 0.826$

Figure 15.12: 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 15.2(b).

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 $\mathrm{BER} = 10^{-4}$. However, this was obtained at the cost of higher receiver complexity compared to the **CT** system as seen in Table 15.4. At high code rates the **CC** system performs poorly over the five-path Rayleigh fading channel. For example, at $\mathrm{BER} = 10^{-4}$ 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.

15.5 Summary and Conclusions

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 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 $\mathrm{BER} = 10^{-4}$, 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 15.2(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 $\mathrm{BER} = 10^{-4}$ 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, 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.

Having characterised the performance of various turbo equalisers in this chapter, our discussions evolve further in the next chapter in the area of reducing the associated implementation complexity.

Bibliography

[1] E. Berlekamp, *Algebraic Coding Theory*. New York, USA: McGraw-Hill, 1968.

[2] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. IT-15, pp. 122–127, January 1969.

[3] R. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA, USA: Addison-Wesley, 1983. ISBN 0-201-10102-5.

[4] A. Michelson and A. Levesque, *Error Control Techniques for Digital Communication*. New York, USA: John Wiley and Sons, 1985.

[5] S. Lin and D. Constello Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall, October 1982. ISBN: 013283796X.

[6] W. Peterson and E. Weldon Jr., *Error Correcting Codes*. Cambridge, MA, USA: MIT. Press, 2nd ed., August 1972. ISBN: 0262160390.

[7] G. Clark Jr. and J. Cain, *Error Correction Coding for Digital Communications*. New York, USA: Plenum Press, May 1981. ISBN: 0306406152.

[8] K. Wong, *Transmission of Channel Coded Speech and Data over Mobile Channels*. PhD thesis, University of Southampton, UK, 1989.

[9] R. Steele and L. Hanzo, eds., *Mobile Radio Communications*. Piscataway, NJ, USA: IEEE Press, 1999.

[10] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, pp. 379–427, 1948.

[11] R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, 1950.

[12] P. Elias, "Coding for noisy channels," *IRE Conv. Rec. pt.4*, pp. 37–47, 1955.

[13] J. Wozencraft, "Sequential decoding for reliable communication," *IRE Natl. Conv. Rec.*, vol. 5, pt.2, pp. 11–25, 1957.

[14] J. Wozencraft and B. Reiffen, *Sequential Decoding*. Cambridge, MA, USA: MIT Press, 1961.

[15] R. Fano, "A heuristic discussion of probabilistic coding," *IEEE Transactions on Information Theory*, vol. IT-9, pp. 64–74, April 1963.

[16] J. Massey, *Threshold Decoding*. Cambridge, MA, USA: MIT Press, 1963.

[17] A. Viterbi, "Error bounds for convolutional codes and an asymphotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.

[18] G. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, March 1973.

[19] J.-H. Chen, "High-quality 16 kb/s speech coding with a one-way delay less than 2 ms," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing, ICASSP'90*, vol. 1, (Albuquerque, New Mexico, USA), pp. 453–456, IEEE, 3–6 April 1990.

[20] L.R. Bahl and J. Cocke and F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimising Symbol Error Rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

[21] C. Berrou and 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, May 1993.

[22] 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.

[23] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, September 1959.

[24] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, pp. 68–79, March 1960.

[25] R. Bose and D. Ray-Chaudhuri, "Further results on error correcting binary group codes," *Information and Control*, vol. 3, pp. 279–290, September 1960.

[26] W. Peterson, "Encoding and error correction procedures for the Bose-Chaudhuri codes," *IRE Trans. Inform. Theory*, vol. IT-6, pp. 459–470, September 1960.

[27] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Transactions on Information Theory*, vol. IT-24, pp. 76–80, January 1978.

[28] B. Honary and G. Markarian, *Trellis Decoding of Block Codes*. Kluwer Academic Publishers Group, Distribution Centre, Post Office Box 322, 3300 AH Dordrecht, The Netherlands: Kluwer Academic Publishers, 1997.

[29] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and Trellis-based Decoding Algorithms for Linear Block Codes*. Norwell, Massachusetts 02061 USA: Kluwer Academic, 1998.

[30] G. Forney, "Coset Codes-Part II: Binary Lattices and Related Codes," *IEEE Transactions on Information Theory*, vol. 34, pp. 1152–1187, September 1988.

[31] H. Manoukian and B. Honary, "BCJR trellis construction for binary linear block codes," *IEE Proc-Comms*, vol. 144, pp. 367–371, December 1997.

[32] B. Honary, G. Markarian, and P. Farrell, "Generalised array codes and their trellis structure," *Electronics Letters*, vol. 29, pp. 541–542, March 1993.

[33] B. Honary and G. Markarian, "Low-complexity trellis decoding of Hamming codes," *Electronics Letters*, vol. 29, pp. 1114–1116, June 1993.

[34] B. Honary, G. Markarian, and M. Darnell, "Low-complexity trellis decoding of linear block codes," *IEE Proceeding Communication*, vol. 142, pp. 201–209, August 1995.

[35] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On complexity of trellis structure of linear block codes," *IEEE Transactions on Information Theory*, vol. 39, pp. 1057–1937, May 1993.

[36] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On the optimum bit orders with respects to the state complexity of trellis diagrams for binary linear codes," *IEEE Transactions on Information Theory*, vol. 39, pp. 242–245, January 1993.

[37] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information Theory*, vol. IT-18, pp. 170–182, January 1972.

[38] D. Gorenstein and N. Zierler, "A class of cyclic linear error-correcting codes in p^m synbols," *J. Soc. Ind. Appl. Math.*, vol. 9, pp. 107–214, June 1961.

[39] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, pp. 300–304, June 1960.

[40] E. Berlekamp, "On decoding binary Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 11, pp. 577–579, 1965.

[41] J. Massey, "Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 11, pp. 580–585, 1965.

[42] M. Oh and P. Sweeney, "Bit-level soft-decision sequential decoding for Reed Solomon codes," in *Workshop on Coding and Cryptography*, (Paris, France), January 1999.

[43] M. Oh and P. Sweeney, "Low complexity soft-decision sequential decoding using hybrid permutation for rs codes," in *Seventh IMA Conference on Cryptography and Coding*, (Royal Agricultural College, Cirencester, UK), December 1999.

[44] D. Burgess, S. Wesemeyer, and P. Sweeney, "Soft-decision decoding algorithms for RS codes," in *Seventh IMA Conference on Cryptography and Coding*, (Royal Agricultural College, Cirencester, UK), December 1999.

[45] Consultative Committee for Space Data Systems, *Blue Book: Recommendations for Space Data System Standards: Telemetry Channel Coding*, May 1984.

[46] European Telecommunication Standard Institute (ETSI), *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for MVDS at 10GHz and above*, ETS 300 748 ed., October 1996. http://www.etsi.org/.

[47] F. Taylor, "Residue arithmetic: A tutorial with examples," *IEEE Computer Magazine*, vol. 17, pp. 50–62, May 1984.

[48] N. Szabo and R. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. New York, USA: McGraw-Hill, 1967.

[49] R. Watson and C. Hastings, "Self-checked computation using residue arithmetic," *Proceedings of the IEEE*, vol. 54, pp. 1920–1931, December 1966.

[50] H. Krishna, K.-Y. Lin, and J.-D. Sun, "A coding theory approach to error control in redundant residue number systems - part I: theory and single error correction," *IEEE Transactions on Circuits Systems*, vol. 39, pp. 8–17, January 1992.

[51] J.-D. Sun and H. Krishna, "A coding theory approach to error control in redundant residue number systems — part II: multiple error detection and correction," *IEEE Transactions on Circuits Systems*, vol. 39, pp. 18–34, January 1992.

[52] T. H. Liew, L. L. Yang, and L. Hanzo, "Soft-decision Redundant Residue Number System Based Error Correction Coding," in *Proc. of IEEE VTC'99*, (Amsterdam, The Netherlands), pp. 2546–2550, Sept 1999.

[53] G. Ungerboeck, "Treliis-coded modulation with redundant signal sets part 1: Introduction," *IEEE Communications Magazine*, vol. 25, pp. 5–11, February 1987.

[54] G. Ungerboeck, "Treliis-coded modulation with redundant signal sets part ii state of the art," *IEEE Communications Magazine*, vol. 25, pp. 12–21, February 1987.

[55] C. Schlegel, *Trellis Coding*. The Institute of Electrical and Electronics Engineers, Inc., New York: IEEE Press, 1997.

[56] R. Steele and L. Hanzo, eds., *Mobile Radio Communications: Second and Third Generation Cellular and WATM Systems*. New York, USA: IEEE Press - John Wiley & Sons, 2nd ed., 1999.

[57] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference," *IEEE Globecom*, pp. 1679–1684, December 1990.

[58] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol dectectors with parallel structures for ISI channels," *IEEE Transactions on Communications*, vol. 42, pp. 1661–1671, 1994.

[59] P. Robertson and E. Villebrun and P. Höher, "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, June 1995.

[60] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," in *IEEE Globecom*, pp. 1680–1686, 1989.

[61] J. Hagenauer, "Source-controlled channel decoding," *IEEE Transactions on Communications*, vol. 43, pp. 2449–2457, September 1995.

[62] S. L. Goff, A. Glavieux, and C. Berrou, "Turbo-codes and high spectral efficiency modulation," in *Proceedings of IEEE International Conference on Communications*, pp. 645–649, 1994.

[63] U. Wachsmann and J. Huber, "Power and bandwidth efficient digital communications using turbo codes in multilevel codes," *European Transactions on Telecommunications*, vol. 6, pp. 557–567, September–October 1995.

[64] P. Robertson and T. Worz, "Bandwidth-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 206–218, Feb 1998.

[65] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Transactions on Communications*, vol. 44, pp. 591–600, May 1996.

[66] 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.

[67] L. Perez, J. Seghers, and D. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1698–1709, November 1996.

[68] 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.

[69] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Transactions on Communications*, vol. 46, pp. 1003–1010, August 1998.

[70] 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.

[71] 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.

[72] P. Jung and M. Nasshan, "Performance evaluation of turbo codes for short frame transmission systems," *IEE Electronics Letters*, vol. 30, pp. 111–112, January 1994.

[73] P. Jung, "Comparison of turbo-code decoders applied to short frame transmission systems," *IEEE Journal on Selected Areas in Communications*, pp. 530–537, 1996.

[74] Peter Jung, Markus Naßhan and Josef Blanz, "Application of Turbo-Codes to a CDMA Mobile Radio System Using Joint Detection and Antenna Diversity," in *Proceedings of the IEEE Conference on Vehicular Technology*, pp. 770–774, 1994.

[75] A. Barbulescu and S. Pietrobon, "Interleaver design for turbo codes," *IEE Electronics Letters*, pp. 2107–2108, December 1994.

[76] Bernard Sklar, "A Primer on Turbo Code Concepts," *IEEE Communications Magazine*, pp. 94–102, Dec 1997.

[77] V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space-Time Codes for High Data Rate Wireless Communication: Performance Criterion and Code Construction," *IEEE Transactions on Information Theory*, vol. 44, pp. 744–765, March 1998.

[78] S. M. Alamouti, "A Simple Transmit Diversity Technique for Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 1451–1458, October 1998.

[79] H. J. V. Tarokh and A. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Transactions on Information Theory*, vol. 45, pp. 1456–1467, May 1999.

[80] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block coding for wireless communications: Performance results," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 451–460, March 1999.

[81] G. Bauch, A. Naguib, and N. Seshadri, "MAP Equalization of Space-Time Coded Signals over Frequency Selective Channels," in *Proceedings of Wireless Communications and Networking Conference*, (New Orleans, USA), September 1999.

[82] G. Bauch and N. Al-Dhahir, "Reduced-complexity turbo equalization with multiple transmit and receive antennas over multipath fading channels," in *Proceedings of Information Sciences and Systems*, (Princeton, USA), pp. WP3 13–18, March 2000.

[83] D. Agrawal, V. Tarokh, A. Naguib, and N. Seshadri, "Space-time coded OFDM for high data-rate wireless communication over wideband channels," in *Proceedings of IEEE Vehicular Technology Conference*, (Ottawa, Canada), pp. 2232–2236, May 1998.

[84] Y. Li, N. Seshadri, and S. Ariyavisitakul, "Channel estimation for OFDM systems with transmitter diversity in mobile wireless channels," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 461–471, March 1999.

[85] Y. Li, J. Chuang, and N. Sollenberger, "Transmitter diversity for OFDM systems and its impact on high-rate data wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1233–1243, July 1999.

[86] A. Naguib and N. Seshdri and A. Calderbank, "Increasing Data Rate Over Wireless Channels: Space-Time Coding for High Data Rate Wireless Communications," *IEEE Signal Processing Magazine*, vol. 17, pp. 76–92, May 2000.

[87] A. F. Naguib, V. Tarokh, N. Seshadri, and A. R. Calderbank, "A Space-Time Coding Modem for High-Data-Rate Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 1459–1478, October 1998.

[88] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *GLOBECOM 94*, (San Francisco, California), pp. 339–343, November 1994.

[89] T. Kasami, *Combinational Mathematics and its Applications*. USA: University of North Carolina Press, 1969.

[90] W. Peterson, *Error Correcting Codes*. Cambridge, MA, USA: MIT Press, 1st ed., 1961.

[91] F. MacWilliams and J. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

[92] B. Sklar, *Digital Communications—Fundamentals and Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1988.

[93] I. Blake, ed., *Algebraic Coding Theory: History and Development*. Dowden, Hutchinson and Ross, 1973.

[94] V. Pless, *Introduction to the Theory of Error-correcting Codes*. New York, USA: John Wiley and Sons, 1982. ISBN: 0471813044.

[95] C. E. Shannon, *Mathematical Theory of Communication*. University of Illinois Press, 1963.

[96] C. Heegard and S. B. Wicker, *Turbo Coding*. Kluwer International, 1999.

[97] M. Bossert, *Channel Coding for Telecommunications*. New York, USA: John Wiley and Sons, 1999. ISBN 0-471-98277-6.

[98] B. Vucetic and J. Yuan, *Turbo Codes Principles and Applications*. The Netherlands: Kluwer Academic Publishers, 2000.

[99] R. Lidl and H. Niederreiter, *Finite Fields*. Cambridge, UK: Cambridge University Press, October 1996.

[100] D. Hoffman, D. Leonard, C. Lindner, K. Phelps, C. Rodger, and J. Wall, *Coding Theory*. New York, USA: Marcel Dekker, Inc., 1991.

[101] J. Huber, *Trelliscodierung*. Berlin: Springer Verlag, 1992.

[102] J. Anderson and S. Mohan, *Source and Channel Coding — An Algorithmic Approach*. Dordrecht: Kluwer Academic Publishers, 1993.

[103] S. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1994.

[104] J. G. Proakis, *Digital Communications*. Mc-Graw Hill International Editions, 3rd ed., 1995.

[105] P. Sweeney, *Error Control Coding: An Introduction*. New York, USA: Prentice Hall, 1991.

[106] 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.

[107] 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, 1995.

[108] D. Raphaeli and Y. Zarai, "Combined turbo equalization and turbo decoding," *IEEE Communications Letters*, vol. 2, pp. 107–109, April 1998.

[109] J. Heller and I. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Transactions on Communication Technology*, vol. COM-19, pp. 835–848, October 1971.

[110] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 561–580, April 1975.

[111] R. Blahut, *Fast Algorithms for Digital Signal Processing*. Reading, MA, USA: Addison-Wesley, 1985. ISBN 0-201-10155-6.

[112] J. Schur, "Ueber potenzreihen, die im innern des einheits- kreises beschraenkt sind," *Journal fuer Mathematik*, pp. 205–232. Bd. 147, Heft 4.

[113] R. Chien, "Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Info. Theory*, vol. 10, pp. 357–363, October 1964.

[114] A. Jennings, *Matrix Computation for Engineers and Scientists*. New York, USA: John Wiley and Sons Ltd., 1977.

[115] G. Forney Jr, "On decoding BCH codes," *IEEE Transactions on Information Theory*, vol. IT-11, pp. 549–557, 1965.

[116] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding goppa codes," *Information Control*, no. 27, pp. 87–99, 1975.

[117] S. Golomb, *Shift Register Sequences*. Laugana Hills, CA, USA: Aegean Park Press, 1982.

[118] J. Stenbit, "Table of generators for Bose-Chaudhuri codes," *IEEE Transactions on Information Theory*, vol. 10, pp. 390–391, October 1964.

[119] B. Sklar, *Digital Communications Fundamentals and Applications*, ch. 5.6.5, pp. 294–296. Prentice Hall, Englewood Cliffs: Prentice-Hall International Editions, 1988.

[120] R. Steele and L. Hanzo, eds., *Mobile Radio Communications*, ch. 4.4.4, pp. 425–428. IEEE Press, 445 Hoes Lane, Piscataway, NJ, 08855, USA: IEEE Press and Pentech Press, 1999.

[121] R. Blahut, *Theory and Practice of Error Control Codes*, ch. 6, pp. 130–160. IBM Corporation, Owego, NY 13827, USA: Addison-Wesley Publishing Company, 1983.

[122] 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*, pp. 737–740, May 1993.

[123] R. Pyndiah, "Iterative decoding of product codes: Block turbo codes," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 71–79, September 1997.

[124] M. Breiling and L. Hanzo, "Optimum non-iterative decoding of turbo codes," *IEEE Transactions on Information Theory*, vol. 46, pp. 2212–2228, September 2000.

[125] M. Breiling and L. Hanzo, "Optimum Non-iterative Turbo-Decoding," *Proc. of PIMRC'97*, pp. 714–718, Sept 1997. Helsinki, Finland.

[126] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," *IEEE Globecom*, pp. 1298–1303, 1994.

[127] M. Breiling, "Turbo coding simulation results," tech. rep., Universität Karlsruhe, Germany and Southampton University, UK, 1997.

[128] C. Berrou, "Some clinical aspects of turbo codes," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 26–31, September 1997.

[129] A. Viterbi, "Approaching the Shannon limit: Theorist's dream and practitioner's challenge," in *Proceedings of the International Conference on Millimeter Wave and Far Infrared Science and Technology*, pp. 1–11, 1996.

[130] A. J. Viterbi, "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes," *IEEE Journal on Selected Areas in Communications*, pp. 260–264, February 1997.

[131] J. Woodard, T. Keller, and L. Hanzo, "Turbo-coded orthogonal frequency division multiplex transmission of 8 kbps encoded speech," in *Proceeding of ACTS Mobile Communication Summit '97*, (Aalborg, Denmark), pp. 894–899, ACTS, 7–10 October 1997.

[132] P. Jung and M. Nasshan, "Dependence of the error performance of turbo-codes on the interleaver structure in short frame transmission systems," *IEE Electronics Letters*, pp. 287–288, February 1994.

[133] Peter Jung and Markus Naßhan, "Results on Turbo-Codes for Speech Transmission in a Joint Detection CDMA Mobile Radio System with Coherent Receiver Antenna Diversity," *IEEE Transactions on Vehicular Technology*, vol. 46, pp. 862–870, Nov 1997.

[134] H. Herzberg, "Multilevel Turbo Coding with Short Interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 303–309, Feb 1998.

[135] Todd A. Summmers and Stephen G. Wilson, "SNR Mismatch and Online Estimation in Turbo Decoding," *IEEE Transactions on Communications*, vol. 46, pp. 421–423, April 1998.

[136] J. Cavers, "An analysis of pilot symbol assisted modulation for rayleigh fading channels," *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 686–693, November 1991.

[137] R. Gallager, "Low-density parity-check codes," *IEEE Transactions on Information Theory*, pp. 21–28, 1962.

[138] G. Battail, "On random-like codes," *Canadian Workshop on Information Theory*, 1995.

[139] A. Khandani, "Design of turbo-code interleaver using hungarian method," *Electronics Letters*, pp. 63–65, 1998.

[140] K. Andrews, C. Heegard, and D. Kozen, "Interleaver design methods for turbo codes," *International Symposium on Information Theory*, p. 420, 1998.

[141] A. Ambroze, G. Wade, and M. Tomlinson, "Turbo code tree and code performance," *IEE Electronics Letters*, vol. 34, pp. 353–354, 19 February 1998.

[142] J. Sadowsky, "A maximum-likelihood decoding algorithm for turbo codes," *IEEE Communications Theory Workshop, Tucson*, 1997.

[143] P. Höher, "New iterative ("turbo") decoding algorithms," *International Symposium on Turbo Codes, Brest*, pp. 63–70, 1997.

[144] J. Proakis, *Digital Communications*. McGraw Hill, 2nd ed., 1989.

[145] A. Jimenez and K. Zigangirov, "Time-varying periodical convolutional codes with low-density parity-check matrix," *accepted for publication in theIEEE Transactions on Information Theory*, 1999.

[146] P. Adde, R. Pyndiah, O. Raoul, and J.-R. Inisan, "Block turbo decoder design," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 166–169, September 1997.

[147] A. Goalic and R. Pyndiah, "Real-time turbo-decoding of product codes on a digital signal processor," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 267–270, September 1997.

[148] F. Macwilliams and N. Sloane, *The theory of error correcting codes*, vol. 16, ch. 18, pp. 567–580. Bell Laboratories, Murray Hill, NJ 07974 USA: North-Holland publishing company, 1978.

[149] S. Ng, T. Liew, L. Yang, and L. Hanzo, "Turbo coding performance using block component codes," in *Proceedings of VTC 2000 Spring*, (Tokyo, Japan), pp. 849–853, 15-18 May 2000.

[150] C. Hastings, "Automatic detection and correction of errors in digital computers using residue arithmetic," in *Proceeding of IEEE Conference*, (Region Six), pp. 465–490, 1966.

[151] R. Watson and C. Hastings, *Residue Arithmetic and Reliable Computer Design*. Washington, D. C.: Spartan Books, 1967.

[152] Y. Keir, P. Cheney, and M. Tannenbaum, "Division and overflow detection in residue number systems," *IRE Transactions on Electronic Computers*, vol. EC-11, pp. 501–507, August 1962.

[153] A. Baraniecka and G. Jullien, "On decoding techniques for residue number system realizations of digital signal processing hardware," *IEEE Transactions on Circuits System*, vol. CAS-25, November 1978.

[154] W. Jenkins, "A new algorithm for scaling in residue number systems with applications to recursive digital filtering," in *IEEE International Symposium on Circuits and Systems*, (Phoenix, Arizona), pp. 56–59, 1977.

[155] H. Huang and F. Taylor, "High speed DFT's using residue numbers," in *IEEE International Conference on Acoustics, Speech, Signal Processing*, (Denver, CO), pp. 238–242, April 1980.

[156] W. Jenkins and B. Leon, "The use of residue number system in the design of finite impulse response filters," *IEEE Transactions on Circuits Systems*, vol. CAS-24, pp. 191–201, April 1977.

[157] T. Vu, "Efficient implementations of the chinese remainder theorem for sign detection and residue decoding," *IEEE Transactions on Computers*, vol. 34, pp. 646–651, July 1985.

[158] C. Su and H. Lo, "An algorithm for scaling and single residue error correction in residue number system," *IEEE Transactions on Computers*, vol. 39, pp. 1053–1064, August 1990.

[159] F. Taylor and C. Huang, "An autoscale residue multiplier," *IEEE Transactions on Computers*, vol. 31, pp. 321–325, April 1982.

[160] G. Jullien, "Residue number scaling and other operations using ROM arrays," *IEEE Transactions on Computers*, vol. C-27, pp. 325–337, April 1978.

[161] W. Jenkins, "A highly efficient residue-combinatorial architecture for digital filters," *Proceeding of the IEEE*, vol. 66, pp. 700–702, June 1978.

[162] G. Alia and E. Martinelli, "A VLSI modulo m multiplier," *IEEE Transactions on Computers*, vol. 40, pp. 873–878, July 1991.

[163] D. Radhakrishnan and Y. Yuan, "Novel Approachs to the Design of VLSI RNS Multiplier," *IEEE Transactions on Circuits and Systems-II*, vol. 39, pp. 52–57, January 1992.

[164] F. Taylor and A. Ramnarayanan, "An efficient residue-to-decimal converter," *IEEE Transactions on Circuits and Systems*, vol. 28, pp. 1164–1169, December 1981.

[165] A. Shenoy and R. Kumaresan, "Residue to binary conversion for RNS arithmetic using only modular look-up table," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1158–1162, September 1988.

[166] G. Alia and E. Martinelli, "On the lower bound to the VLSI complexity of number conversion from weighted to residue representation," *IEEE Transactions on Computers*, vol. 42, pp. 962–967, August 1993.

[167] G. Alia and E. Martinelli, "A VLSI algorithm for direct and reverse conversion from weighted binary number system to residue number system," *IEEE Transactions on circuits and Systems*, vol. 31, pp. 1033–1039, December 1984.

[168] F. Barsi and P. Maestrini, "Error correction properties of redundant residue number systems," *IEEE Transactions on Computers*, vol. 22, pp. 307–315, March 1973.

[169] S. Yau and Y. Liu, "Error correction in redundant residue number systems," *IEEE Transactions on Computers*, vol. C-22, pp. 5–11, January 1984.

[170] H. Krishna and J.-D. Sun, "On theory and fast algorithms for error correction in residue number system product codes," *IEEE Transactions on Comput.*, vol. 42, pp. 840–852, July 1993.

[171] R. Cosentino, "Fault tolerance in a systolic residue arithmetic processor array," *IEEE Transactions on Computers*, vol. 37, pp. 886–890, July 1988.

[172] F. Barsi and P. Maestrini, "Error detection and correction by product codes in residue number system," *IEEE Transactions on Computers*, vol. 23, pp. 915–924, September 1974.

[173] V. Ramachandran, "Single residue error correction in residue number systems," *IEEE Transactions on Computers*, vol. 32, pp. 504–507, May 1983.

[174] A. Shenoy and R. Kumaresan, "Fast base extension using a redundant modulus in RNS," *IEEE Transactions on Computers*, vol. 38, pp. 292–297, February 1989.

[175] E. Claudio, G. Orlandi, and F. Piazza, "A systolic redundant residue arithmetic error correction circuit," *IEEE Transactions on Computers*, vol. 42, pp. 427–432, April 1993.

[176] L. Yang and L. Hanzo, "Redundant residue number system based error correction codes," in *Proceedings of IEEE VTC'01 (Fall)*, (Atlantic City, USA), pp. 1472–1476, October 2001.

[177] M. Etzel and W. Jenkins, "Redundant residue number systems for error detection and correction in digital filters," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-28, pp. 538–544, October 1980.

[178] C. Huang, D. Peterson, H. Rauch, J. Teague, and D. Fraser, "Implementation of a fast digital processor using the residue number system," *IEEE Transactions on Circuits Systems*, vol. CAS-28, pp. 32–38, January 1981.

[179] W. Jenkins, "Recent advances in residue number system techniques for recursive digital filtering," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-27, pp. 19–30, February 1979.

[180] M. Soderstrand, "A high-speed, low-cost, recursive digital filter using residue number arithmetic," *Proceedings of IEEE*, vol. 65, pp. 1065–1067, July 1977.

[181] M. Soderstrand, W. Jenkins, and G. Jullien, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. New York, USA: IEEE Press, 1986.

[182] R. Krishnan, G. Jullien, and W. Miller, "Complex digital signal processing using quadratic residue number systems," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, pp. 166–176, February 1986.

[183] L.-L. Yang and L. Hanzo, "Residue number system arithmetic assisted m-ary modulation," *IEEE Communications Letters*, vol. 3, pp. 28–30, February 1999.

[184] A. Baraniecka and G. Jullien, "Residue number system implementations of number theoretic transforms," *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. ASSP-28, pp. 285–291, June 1980.

[185] M. Soderstrand and E. Fields, "Multipliers for residue number arithmetic digital filters," *Electronics Letters*, vol. 13, pp. 164–166, March 1977.

[186] B. Tseng, G. Jullien, and W. Miller, "Implementation of FFT structure using the residue number system," *IEEE Transactions on Computers*, vol. 28, pp. 831–844, November 1979.

[187] L.-L. Yang and L. Hanzo, "Performance of residue number system based DS-CDMA over multipath fading channels using orthogonal sequences," *ETT*, vol. 9, pp. 525–536, November–December 1998.

[188] L.-L. Yang and L. Hanzo, "Residue number system based multiple code DS-CDMA systems," in *Proceeding of VTC'99 (Spring)*, (Houston, TX, USA), IEEE, 16–20 May 1999.

[189] L. Hanzo and L.-L. Yang, "Ratio statistic test assisted residue number system based parallel communication systems," in *Proceeding of VTC'99 (Spring)*, (Houston, TX, USA), pp. 894–898, IEEE, 16–20 May 1999.

[190] K. Yen, L.-L. Yang, and L. Hanzo, "Residual number system assisted CDMA – a new system concept," in *Proceedings of 4th ACTS Mobile Communications Summit'99*, (Sorrento, Italy), pp. 177–182, June 8–11 1999.

[191] L.-L. Yang and L. Hanzo, "A residue number system based parallel communication scheme using orthogonal signaling: part I - system outline," *To appear in IEEE Trans. on Vehicular Technology*, 2002.

[192] L.-L. Yang and L. Hanzo, "A residue number system based parallel communication scheme using orthogonal signaling: part II - multipath fading channels," *To appear in IEEE Trans. on Vehicular Technology*, 2002.

[193] D. Mandelbaum, "Error correction in residue arithmetic," *IEEE Transactions on Computers*, vol. C-21, pp. 538–543, June 1972.

[194] L. Yang and L. Hanzo, "Minimum-distance decoding of redundant residue number system codes," in *Proceedings of IEEE ICC'2001*, (Helsinki, Finland), pp. 2975–2979, June 2001.

[195] T. Liew, L. Yang, and L. Hanzo, "Turbo decoded redundant residue number system codes," in *Proceedings of VTC 2000 Spring*, (Tokyo, Japan), pp. 576–580, 15-18 May 2000.

[196] T. Keller, T. H. Liew, and L. Hanzo, "Adaptive redundant residue number system coded multicarrier modulation," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2292–2301, November 2000.

[197] T. Keller, T. Liew, and L. Hanzo, "Adaptive rate RRNS coded OFDM transmission for mobile communication channels," in *Proceedings of VTC 2000 Spring*, (Tokyo, Japan), pp. 230–234, 15-18 May 2000.

[198] W. Jenkins and E. Altman, "Self-checking properties of residue number error checkers based on mixed radix conversion," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 159–167, February 1988.

[199] W. Jenkins, "The design of error checkers for self-checking residue number arithmetic," *IEEE Transactions on Computers*, vol. 32, pp. 388–396, April 1983.

[200] O. Aitsab and R. Pyndiah, "Performance of Reed-Solomon block turbo code," in *GLOBECOM '96*, (London, U.K.), pp. 121–125, November 1996.

[201] O. Aitsab and R. Pyndiah, "Performance of concatenated Reed-Solomon/Convolutional codes with iterative decoding," in *GLOBECOM '97*, (New York, USA), pp. 644–648, 1997.

[202] R. Pyndiah, P. Combelles, and P. Adde, "A very low complexity block turbo decoder for product codes," in *GLOBECOM '96*, (London, U.K.), pp. 101–105, November 1996.

[203] S. Lin, D. Constello Jr., and M. Miller, "Automatic-repeat-request error-control schemes," *IEEE Communications Magazine*, vol. 22, pp. 5–17, December 1984.

[204] T. Keller, T. Liew, and L. Hanzo, "Adaptive redundant residue number system coded multicarrier modulation," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2292–2301, November 2000.

[205] T. James, "Study into redundant residue number system codes," tech. rep., University of Southampton, May 1999.

[206] L. Hanzo, W. Webb, and T. Keller, *Single- and Multi-Carrier Quadrature Amplitude Modulation: Principles and Applications for Personal Communications, WLANs and Broadcasting*. IEEE Press, 2000.

[207] G. Ungerböck, "Channel Coding with Multilevel/Phase Signals," *IEEE Transactions on Information Theory*, vol. IT-28, pp. 55–67, January 1982.

[208] D. Divsalar and M. K. Simon, "The design of trellis coded MPSK for fading channel: Performance criteria," *IEEE Transactions on Communications*, vol. 36, pp. 1004–1012, September 1988.

[209] D. Divsalar and M. K. Simon, "The design of trellis coded MPSK for fading channel: Set partitioning for optimum code design," *IEEE Transactions on Communications*, vol. 36, pp. 1013–1021, September 1988.

[210] P. Robertson, T. Wörz, "Bandwidth-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 206–218, February 1998.

[211] E. Zehavi, "8-PSK trellis codes for a Rayleigh fading channel," *IEEE Transactions on Communications*, vol. 40, pp. 873–883, May 1992.

[212] X. Li and J.A. Ritcey, "Bit-interleaved coded modulation with iterative decoding," *IEEE Communications Letters*, vol. 1, November 1997.

[213] X. Li and J.A. Ritcey, "Trellis-Coded Modulation with Bit Interleaving and Iterative Decoding," *IEEE Journal on Selected Areas in Communications*, vol. 17, April 1999.

[214] X. Li and J.A. Ritcey, "Bit-interleaved coded modulation with iterative decoding — Approaching turbo-TCM performance without code concatenation," in *Proceedings of CISS 1998*, (Princeton University, USA), March 1998.

[215] S. X. Ng, T. H. Liew, L-L. Yang and L. Hanzo, "Comparative Study of TCM, TTCM, BICM and BICM-ID schemes," *IEEE Vehicular Technology Conference*, p. 265 (CDROM), May 2001.

[216] S. X. Ng, C. H. Wong and L. Hanzo, "Burst-by-Burst Adaptive Decision Feedback Equalized TCM, TTCM, BICM and BICM-ID," *International Conference on Communications (ICC)*, pp. 3031–3035, June 2001.

[217] C. S. Lee, S. X. Ng, L. Piazzo and L. Hanzo, "TCM, TTCM, BICM and Iterative BICM Assisted OFDM-Based Digital Video Broadcasting to Mobile Receivers," *IEEE Vehicular Technology Conference*, p. 113 (CDROM), May 2001.

[218] X. Li and J.A. Ritcey, "Bit-interleaved coded modulation with iterative decoding using soft feedback," *IEE Electronics Letters*, vol. 34, pp. 942–943, May 1998.

[219] G. Ungerböck, "Trellis-coded modulation with redundant signal sets. Part 1 and 2," *IEEE Communications Magazine*, vol. 25, pp. 5–21, February 1987.

[220] J.-H. Chen and A. Gersho, "Gain-adaptive vector quantization with application to speech coding," *IEEE Transactions on Communications*, vol. 35, pp. 918–930, September 1987.

[221] S. S. Pietrobon, G. Ungerböck, L. C. Perez and D. J. Costello, "Rotationally invariant nonlinear trellis codes for two-dimensional modulation," *IEEE Transactions on Information Theory*, vol. IT-40, pp. 1773–1791, November 1994.

[222] C. Schlegel, "Chapter 3: Trellis Coded Modulation," in *Trellis Coding*, (New York), pp. 43–89, IEEE Press, September 1997.

[223] J. K. Cavers and P. Ho, "Analysis of the Error Performance of Trellis-Coded Modulations in Rayleigh-Fading Channels," *IEEE Transactions on Communications*, vol. 40, pp. 74–83, January 1992.

[224] G. D. Forney, "The Viterbi ALgorithm," in *Proceedings of the IEEE*, vol. 61, pp. 268–277, March 1973.

[225] L. Piazzo, "TTCM-OFDM over Wideband Fading Channels," tech. rep., University of Southampton, December 1999.

[226] J. G. Proakis, "Optimum Receivers for the Additive White Gaussian Noise Channel," in *Digital Communication*, (New York), pp. 260–274, September 1995.

[227] K. Abend and B. D. Fritchman, "Statistical detection for communication channels with intersymbol interference," *Proceedings of the IEEE*, vol. 58, pp. 779–785, May 1970.

[228] L. Piazzo, "An algorithm for SBS Receivers/Decoders," *IEE Electronics Letters*, vol. 32, pp. 1058–1060, Jun 1996.

[229] S.S. Pietrobon, R.H. Deng, A. Lafanechére, G. Ungerböck and D.J. Costello, "Trellis-Coded Multidimensional Phase Modulation," *IEEE Transactions on Information Theory*, vol. 36, pp. 63–89, January 1990.

[230] L.-F. Wei, "Trellis-coded modulation with multidimensional constellations," *IEEE Transactions on Information Theory*, vol. IT-33, pp. 483–501, July 1987.

[231] P. Robertson, "An Overview of Bandwidth Efficient Turbo Coding Schemes," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 103–110, September 1997.

[232] G. Caire and G. Taricco and E. Biglieri, "Bit-Interleaved Coded Modulation," *IEEE Transactions on Information Theory*, vol. 44, pp. 927–946, May 1998.

[233] J. Hagenauer, "Rate-compatible puncture convolutional codes (RCPC) and their application," *IEEE Transactions on Communications*, vol. 36, pp. 389–400, April 1988.

[234] L. Lee, "New rate-compatible puncture convolutional codes for viterbi decoding," *IEEE Transactions on Communications*, vol. 42, pp. 3073–3079, December 1994.

[235] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "A Soft-Input Soft-Output APP Module for Iterative Decoding of concatenated codes," *IEEE Communications Letter*, vol. 1, pp. 22–24, January 1997.

[236] J. G. Proakis, "Chapter 10: Communication Through Band-Limited Channels," in *Digital Communications*, (New York), pp. 583–635, McGraw-Hill International Editions, 3rd Edition, September 1995.

[237] C. H. Wong, *Wideband Adaptive Full Response Multilevel Transceivers and Equalizers*. PhD thesis, University of Southampton, United Kingdom, November 1999.

[238] D.F. Mix, *Random Signal Processing*. Englewood Cliffs NJ, USA: Prentice-Hall, 1995.

[239] J.C. Cheung, *Adaptive Equalisers for Wideband TDMA Mobile Radio*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, UK, 1991.

[240] R. Steele and W. Webb, "Variable rate QAM for data transmission over Rayleigh fading channels," in *Proceeedings of Wireless '91*, (Calgary, Alberta), pp. 1–14, IEEE, 1991.

[241] S. Sampei and S. Komaki and N. Morinaga, "Adaptive Modulation/TDMA Scheme for large capacity personal Multi-Media Communication Systems," *IEICE Transactions on Communications (Japan)*, vol. E77-B, pp. 1096–1103, September 1994.

[242] J.M. Torrance and L. Hanzo, "Latency and Networking Aspects of Adaptive Modems over Slow Indoors Rayleigh Fading Channels," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 4, pp. 1237–1251, 1998.

[243] J.M. Torrance and L. Hanzo, "Interference Aspects of adaptive modems over slow Rayleigh fading channels," *IEEE Vehicular Technology Conference*, vol. 48, pp. 1527–1545, September 1999.

[244] A.J. Goldsmith and S. Chua, "Variable-rate variable-power MQAM for fading channels," *IEEE Transactions on Communications*, vol. 45, pp. 1218–1230, October 1997.

[245] C. Wong and L. Hanzo, "Upper-bound performance of a wideband burst-by-burst adaptive modem," *IEEE Transactions on Communications*, vol. 48, pp. 367–369, March 2000.

[246] H. Matsuoka and S. Sampei and N. Morinaga and Y. Kamio, "Adaptive Modulation System with Variable Coding Rate Concatenated Code for High Quality Multi-Media Communications Systems," in *Proceedings of IEEE VTC'96*, vol. 1, (Atlanta, GA, USA), pp. 487–491, IEEE, 28 April–1 May 1996.

[247] V. Lau and M. Macleod, "Variable rate adaptive trellis coded QAM for high bandwidth efficiency applications in rayleigh fading channels," in *Proceedings of IEEE Vehicular Technology Conference (VTC'98)*, (Ottawa, Canada), pp. 348–352, IEEE, 18–21 May 1998.

[248] A.J. Goldsmith and S. Chua, "Adaptive Coded Modulation for fading channels," *IEEE Transactions on Communications*, vol. 46, pp. 595–602, May 1998.

[249] C.H. Wong, T. H. Liew and L. Hanzo, "Burst-by-Burst Turbo Coded Wideband Adaptive Modulation with Blind Modem Mode Detection," *Proceedings of 4th ACTS Mobile Communications Summit 1999, Sorrento, Italy*, pp. 303–308, June 1999.

[250] D. Goeckel, "Adaptive Coding for Fading Channels using Outdated Fading Estimates," *IEEE Transactions on Communications*, vol. 47, pp. 844–855, June 1999.

[251] "COST 207: Digital land mobile radio communications, final report." Office for Official Publications of the European Communities, 1989. Luxembourg.

[252] A. Klein and R. Pirhonen and J. Skoeld and R. Suoranta, "FRAMES Multiple Access MODE 1 — Wideband TDMA with and without Spreading," in *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, vol. 1, (Helsinki, Finland), pp. 37–41, 1–4 September 1997.

[253] B. J. Choi, M. Münster, L. L. Yang, and L. Hanzo, "Performance of Rake receiver assisted adaptive-modulation based CDMA over frequency selective slow Rayleigh fading channel," *Electronics Letters*, vol. 37, pp. 247–249, February 2001.

[254] A. Duel-Hallen and S. Hu and H. Hallen, "Long Range Prediction of Fading Signals," *IEEE Signal Processing Magazine*, vol. 17, pp. 62–75, May 2000.

[255] S. M. Alamouti and S. Kallel, "Adaptive Trellis-Coded Multiple-Phased-Shift Keying Rayleigh fading channels," *IEEE Transactions on Communications*, vol. 42, pp. 2305–2341, June 1994.

[256] L. Piazzo and L. Hanzo, "TTCM-OFDM over Dispersive Fading Channels," *IEEE Vehicular Technology Conference*, vol. 1, pp. 66–70, May 2000.

[257] R. W. Chang, "Synthesis of Band-Limited Orthogonal Signals for Multichannel Data Transmission," *Bell Systems Technical Journal*, vol. 46, pp. 1775–1796, December 1966.

[258] M.S. Zimmermann and A.L. Kirsch, "The AN/GSC-10/KATHRYN/Variable Rate Data Modem for HF Radio," *IEEE Transactions on Communication Technology*, vol. CCM–15, pp. 197–205, April 1967.

[259] L.J. Cimini, "Analysis and Simulation of a Digital Mobile Channel Using Orthogonal Frequency Division Multiplexing," *IEEE Transactions on Communications*, vol. 33, pp. 665–675, July 1985.

[260] F. Mueller-Roemer, "Directions in audio broadcasting," *Journal Audio Engineering Society*, vol. 41, pp. 158–173, March 1993.

[261] M. Alard and R. Lassalle, "Principles of modulation and channel coding for digital broadcasting for mobile receivers," *EBU Review, Technical No. 224*, pp. 47–69, August 1987.

[262] I. Kalet, "The multitone channel," *IEEE Transactions on Communications*, vol. 37, pp. 119–124, February 1989.

[263] B. L. Yeap, T. H. Liew and L. Hanzo, "Turbo Equalization of Serially Concatenated Systematic Convolutional Codes and Systematic Space Time Trellis Codes," *IEEE Vehicular Technology Conference*, p. 119 (CDROM), May 2001.

[264] ETSI, *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, August 1997. ETS 300 744.

[265] "COST 207 : Digital land mobile radio communications, final report," tech. rep., Luxembourg, 1989.

[266] P. Chaudhury, W. Mohr, and S. Onoe, "The 3GPP proposal for IMT-2000," *IEEE Communications Magazine*, vol. 37, pp. 72–81, December 1999.

[267] G. Foschini Jr. and M. Gans, "On limits of wireless communication in a fading environment when using multiple antennas," *Wireless Personal Communications*, vol. 6, pp. 311–335, March 1998.

[268] B. Glance and L. Greestein, "Frequency-selective fading effects in digital mobile radio with diversity combining," *IEEE Transactions Communications*, vol. COM-31, pp. 1085–1094, September 1983.

[269] F. Adachi and K. Ohno, "BER performance of QDPSK with postdetection diversity reception in mobile radio channels," *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 237–249, February 1991.

[270] H. Zhou, R. Deng, and T. Tjhung, "Performance of combined diversity reception and convolutional coding for QDPSK land mobile radio," *IEEE Transactions on Vehicular Technology*, vol. 43, pp. 499–508, August 1994.

[271] J. Winters, "Switched diversity with feedback for DPSK mobile radio systems," *IEEE Transactions on Information Theory*, vol. 32, pp. 134–150, February 1983.

[272] G. Raleigh and J. Cioffi, "Spatio-temporal coding for wireless communications," in *GLOBECOM '96*, (London, U.K.), pp. 533–537, November 1996.

[273] A. Wittneben, "Base station modulation diversity for digital SIMULCAST," in *Proceedings of IEEE Vehicular Technology Conference*, pp. 505–511, May 1993.

[274] N. Seshadri and J Winters, "Two signalling schemes for improving the error performance of frequency-division-duplex (FDD) transmission systems using transmitter antenna diversity," *International Journal of Wireless Information Networks*, vol. 1, pp. 49–60, January 1994.

[275] J. Winters, "The diversity gain of transmit diversity in wireless systems with Rayleigh fading," *IEEE Transactions on Vehicular Technology*, vol. 47, pp. 119–123, February 1998.

[276] T. Hattori and K. Hirade, "Multitransmitter simulcast digital signal transmission by using frequency offset strategy in land mobile radio-telephone system," *IEEE Transactions on Vehicular Technology*, vol. 27, pp. 231–238, 1978.

[277] A. Hiroike, F. Adachi, and N. Nakajima, "Combined effects of phase sweeping transmitter diversity and channel coding," *IEEE Transactions on Vehicular Technology*, vol. 41, pp. 170–176, May 1992.

[278] N. Seshadri, V. Tarokh, and A. Calderbank, "Space-Time Codes for High Data Rate Wireless Communications: Code Construction," in *Proceedings of IEEE Vehicular Technology Conference '97*, (Phoenix, Arizona), pp. 637–641, 1997.

[279] V. Tarokh and N. Seshadri and A. Calderbank, "Space-time codes for high data rate wireless communications: Performance criterion and code construction," in *Proc IEEE International Conference on Communications '97*, (Montreal, Canada), pp. 299–303, 1997.

[280] N. S. V. Tarokh, A. Naguib and A. Calderbank, "Space-time codes for high data rate wireless communications: Mismatch analysis," in *Proc IEEE International Conference on Communications '97*, (Montreal, Canada), pp. 309–313, 1997.

[281] V. Tarokh, A. Naguib, N. Seshadri, and A. R. Calderbank, "Space-time codes for high data rate wireless communication: Performance criteria in the presence of channel estimation errors, mobility, and multile paths," *IEEE Transactions on Communications*, vol. 47, pp. 199–207, February 1999.

[282] D. Brennan, "Linear diversity combining techniques," *Proc. IRE*, vol. 47, pp. 1075–1102, 1959.

[283] G. Bauch, "Concatenation of space-time block codes and Turbo-TCM," in *Proceedings of IEEE International Conference on Communications*, (Vancouver, Canada), pp. 1202–1206, June 1999.

[284] G. Forney, "Convolutional codes: I. Algebraic structure," *IEEE Transactions on Information Theory*, vol. 16, pp. 720–738, November 1970.

[285] G. Forney, "Burst-correcting codes for the classic burst channel," *IEEE Transactions on Communication Technology*, vol. COM-19, pp. 772–781, October 1971.

[286] W. Jakes Jr., ed., *Microwave Mobile Communications*. New York, USA: John Wiley & Sons, 1974.

[287] S. Red, M. Oliphant, and M. Weber, *An Introduction to GSM*. Artech House, 1995.

[288] 3GPP, *Multiplexing and channel coding (TDD)*. 3G TS 25.222, http://www.3gpp.org.

[289] T. Ojanperä and R. Prasad, *Wideband CDMA for Third Generation Mobile Communications*. London, UK: Artech House, 1998.

[290] S. Al-Semari and T. Fuja, "I-Q TCM: Reliable communication over the rayleigh fading channel close to the cuttoff rate," *IEEE Transactions on Information Theory*, vol. 43, pp. 250–262, January 1997.

[291] R. Horn and C. Johnson, *Matrix Analysis*. New York: Cambridge University Press, 1988.

[292] W. Choi and J. Cioffi, "Space-Time Block Codes over Frequency Selective Fading Channels," in *Proceedings of VTC 1999 Fall*, (Amsterdam, Holland), pp. 2541–2545, 19-22 September 1999.

[293] Z. Liu, G. Giannakis, A. Scaglione, and S. Barbarossa, "Block precoding and transmit-antenna diversity for decoding and equalization of unknown multipath channels," in *Proc 33rd Asilomar Conference Signals, Systems and Computers*, (Pacific Grove, Canada), pp. 1557–1561, 1-4 November 1999.

[294] Z. Liu and G. Giannakis, "Space-time coding with transmit antennas for multiple access regardless of frequency-selective multipath," in *Proc 1st Sensor Array and Multichannel SP Workshop*, (Boston, USA), 15-17 March 2000.

[295] T. Liew, J. Pliquett, B. Yeap, L.-L. Yang, and L. Hanzo, "Comparative study of space time block codes and various concatenated turbo coding schemes," in *PIMRC 2000*, (London, UK), pp. 741–745, 18-21 September 2000.

[296] T. Liew, J. Pliquett, B. Yeap, L.-L. Yang, and L. Hanzo, "Concatenated space time block codes and TCM, turbo TCM, convolutional as well as turbo codes," in *GLOBECOM 2000*, (San Francisco, USA), 27 Nov -1 Dec 2000.

[297] W. Webb and R. Steele, "Variable rate QAM for mobile radio," *IEEE Transactions on Communications*, vol. 43, pp. 2223–2230, July 1995.

[298] J. Torrance and L. Hanzo, "Performance upper bound of adaptive QAM in slow Rayleigh-fading environments," in *Proceedings of IEEE ICCS'96/ISPACS'96*, (Singapore), pp. 1653–1657, IEEE, 25–29 November 1996.

[299] J. Torrance, L. Hanzo, and T. Keller, "Interference aspects of adaptive modems over slow Rayleigh fading channels," *IEEE Transactions on Vehicular Technology*, vol. 48, pp. 1527–1545, September 1999.

[300] T. Keller and L. Hanzo, "Adaptive orthogonal frequency division multiplexing schemes," in *Proceeding of ACTS Mobile Communication Summit '98*, (Rhodes, Greece), pp. 794–799, ACTS, 8–11 June 1998.

[301] T. Keller and L. Hanzo, "Blind-detection assisted sub-band adaptive turbo-coded OFDM schemes," in *Proceeding of VTC'99 (Spring)*, (Houston, TX, USA), pp. 489–493, IEEE, 16–20 May 1999.

[302] H. Matsuako, S. Sampei, N. Morinaga, and Y. Kamio, "Adaptive modulation systems with variable coding rate concatenated code for high quality multi-media communication systems," in *Proceedings of IEEE Vehicular Technology Conference*, (Atlanta, USA), pp. 487–491, April 1996.

[303] S.-G. Chua and A. Goldsmith, "Variable-rate variable-power mQAM for fading channels," in *Proceedings of IEEE VTC'96*, (Atlanta, GA, USA), pp. 815–819, IEEE, 28 April–1 May 1996.

[304] J. Torrance and L. Hanzo, "Optimisation of switching levels for adaptive modulation in a slow Rayleigh fading channel," *Electronics Letters*, vol. 32, pp. 1167–1169, 20 June 1996.

[305] J. Torrance and L. Hanzo, "On the upper bound performance of adaptive QAM in a slow Rayleigh fading," *IEE Electronics Letters*, pp. 169–171, April 1996.

[306] L. Hanzo, C. Wong, and M. Yee, *Adaptive Wireless Transceivers*. John Wiley, IEEE Press, 2002. 2001 (For detailed contents, please refer to http://www-mobile.ecs.soton.ac.uk.).

[307] M. Yee, T. Liew, and L. Hanzo, "Burst-by-burst adaptive turbo coded radial basis function assisted feedback equalisation," *IEEE Transactions on Communications*, vol. 49, November 2001.

[308] V. Lau and M. Macleod, "Variable-rate adaptive trellis coded qam for flat-fading channels," *IEEE Transactions on Communications*, vol. 49, pp. 1550–1560, September 2001.

[309] V. Lau and S. Maric, "Variable rate adaptive modulation for DS-CDMA," *IEEE Transactions on Communications*, vol. 47, pp. 577–589, April 1999.

[310] S. Chua and A. Goldsmith, "Adaptive coded modulation for fading channels," *IEEE Transactions on Communications*, vol. 46, pp. 595–602, May 1998.

[311] X. Liu, P. Ormeci, R. Wesel, and D. Goeckel, "Bandwidth-efficient, low-latency adaptive coded modulation schemes for time-varying channels," in *Proceedings of IEEE International Conference on Communications*, (Helsinki, Finland), June 2001.

[312] C. Tidestav, M. Sternad, and A. Ahlén, "Reuse within a cell - interference rejection or multiuser detection," *IEEE Transactions on Communications*, vol. 47, pp. 1511–1522, October 1999.

[313] T. Liew and L. Hanzo, "Space-time block coded adaptive modulation aided OFDM," in *IEEE Globecom 2001*, (San Antonio, USA), pp. 136–140, 25-29 November 2001.

[314] J. Cheung and R. Steele, "Soft-decision feedback equalizer for continuous-phase modulated signals in wide-band mobile radio channels," *IEEE Transactions on Communications*, vol. 42, pp. 1628–1638, February/March/April 1994.

[315] T. Liew and L. Hanzo, "Space-time codes and concatenated channel codes for wireless communications," *to appear in Proceedings of the IEEE*, February 2002.

[316] J. Torrance and L. Hanzo, "Demodulation level selection in adaptive modulation," *Electronics Letters*, vol. 32, pp. 1751–1752, 12 September 1996.

[317] B. J. Choi, T. H. Liew, and L. Hanzo, "Concatenated space-time block coded and turbo coded symbol-by-symbol adaptive OFDM and multi-carrier CDMA systems," in *Proceedings of IEEE VTC 2001-Spring*, p. P.528, IEEE, May 2001.

[318] J. Anderson, T. Aulin, and C. Sundberg, *Digital phase modulation*. Plenum Press, 1986.

[319] K. Murota and K. Hirade, "GMSK modulation for digital mobile radio telephony," *IEEE Transactions on Communications*, vol. 29, pp. 1044–1050, July 1981.

[320] 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.

[321] 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.

[322] 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.

[323] R. 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 *Proceedings of ICC'97*, 1997.

[324] G. Bauch and V. Franz, "Iterative equalisation and decoding for the GSM-system," in *Proceedings of IEEE Vehicular Technology Conference (VTC'98)*, (Ottawa, Canada), pp. 2262–2266, IEEE, 18–21 May 1998.

[325] F. Burkert, G. Caire, J. Hagenauer, T. Hidelang, and G. Lechner, "Turbo decoding with unequal error protection applied to GSM speech coding," in *Proceeding of IEEE Global Telecommunications Conference, Globecom 96*, (London, UK), pp. 2044–2048, IEEE, 18–22 November 1996.

[326] D. Parsons, *The Mobile Radio Propagation Channel*. London: Pentech Press, 1992.

[327] S. Saunders, *Antennas and Propagation for Wireless Communication Systems Concept and Design*. New York, USA: John Wiley and Sons, 1999.

[328] T. Rappaport, *Wireless Communications Principles and Practice*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996.

[329] ETSI, *GSM Recommendation 05.05, Annex 3*, November 1988.

[330] A. Carlson, *Communication Systems*. New York, USA: McGraw-Hill, 1975.

[331] R. Debuda, "Coherent demodulation of frequency shift keying with low deviation ratio," *IEEE Transactions on Communications*, vol. COM-20, pp. 429–435, June 1972.

[332] S. Pasupathy, "Minimum shift keying: a spectrally efficient modulation," *IEEE Communications Magazine*, vol. 17, pp. 14–22, July 1979.

[333] 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.

[334] E. Kreyszig, *Advanced engineering mathematics*. John Wiley & Sons, Inc., 7th ed., 1993.

[335] J. Proakis and D. Manolakis, *Digital Signal Processing — Principles, Algorithms and Applications*. Macmillan, 1992.

[336] M. Moher, "Decoding via cross-entropy minimization," in *Proceedings of the IEEE Global Telecommunications Conference 1993*, (Houston, TX, USA), pp. 809–813, 29 November – 2 December 1993.

[337] D. A. Johnson, S. W. Wales, and P. H. Waters, "Equalisers for GSM," *IEE Colloquium (Digest)*, no. 21, pp. 1/1–1/6, 1990.

[338] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 3 ed., 1991.

[339] P. Robertson and T. Wórz, "Coded modulation scheme employing turbo codes," *IEE Electronics Letters*, vol. 31, pp. 1546–1547, 31st August 1995.

[340] "GSM Recommendation 05.03: Channel coding," November 1988.

[341] M. Mouly and M. Pautet, *The GSM System for Mobile Communications*. Michel Mouly and Marie-Bernadette Pautet, 1992.

[342] G. Bauch, H. Khorram, and J. Hagenauer, "Iterative equalization and decoding in mobile communications systems," in *European Personal Mobile Communications Conference*, (Bonn, Germany), pp. 301–312, 30 September - 2 October 1997.

[343] M. Gertsman and J. 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.

[344] I. Marsland, P. Mathiopoulos, and S. Kallel, "Non-coherent turbo equalization for frequency selective Rayleigh fast fading channels," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 196–199, September 1997.

[345] 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.

[346] 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.

[347] 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.

[348] K. Narayanan and G. Stuber, “A serial concatenation approach to iterative demodulation and decoding,” *IEEE Transactions on Communications*, vol. 47, pp. 956–961, July 1999.

[349] 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.

[350] 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.

[351] Y. V. Svirid and S. Riedel, “Threshold decoding of turbo-codes,” *Proceedings of the IEEE International Symposium on Information Theory*, p. 39, September 1995.

[352] 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.

[353] M. Ho, 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.

[354] 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.

[355] 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.

[356] 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.

[357] B. He and 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.

[358] M. Breiling and L. Hanzo, “Optimum non-iterative turbo-decoding,” in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, 1997*, vol. 2, pp. 714–718, IEEE, 1997.

[359] M. Breiling and L. Hanzo, “Non-iterative optimum super-trellis decoding of turbo codes,” *Electronics Letters*, vol. 33, pp. 848–849, May 1997.

[360] 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.

[361] ETSI, *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz Satellite Services*, August 1997. ETS 300 421.

[362] 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.

[363] M. S. Yee, B. L. Yeap, and L. Hanzo, “Radial basis function assisted turbo equalisation,” in *Proceedings of IEEE Vehicular Technology Conference*, (Japan, Tokyo), pp. 640–644, IEEE, 15-18 May 2000.

[364] A. Glavieux, C. Laot, and J. Labat, “Turbo equalization over a frequency selective channel,” in *Proceedings of the International Symposium on Turbo Codes*, (Brest, France), pp. 96–102, 3-5 September 1997.

[365] C. Wong, B. Yeap, and L. Hanzo, “Wideband Burst-by-Burst Adaptive Modulation with Turbo Equalization and Iterative Channel Estimation,” in *Accepted for the Proceedings of the IEEE Vehicular Technology Conference 2000*, 2000.

[366] M. Sandell, C. Luschi, P. Strauch, and R. Yan, "Iterative channel estimation using soft decision feedback," in *Proceedings of the Global Telecommunications Conference 1998*, (Sydney, Australia), pp. 3728–3733, 8 - 12 November 1998.

[367] S. Haykin, *Digital Communications*. New York, USA: John Wiley and Sons, 1988.

[368] S. Benedetto, D. Divsalar, G. Motorsi, and F. Pollara, "A soft-input soft-output app module for iterative decoding of concatenated codes," *IEEE Communication Letters*, pp. 22–24, 1997.

[369] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Transactions on Telecommunications*, vol. 8, pp. 119–125, March/April 1997.

[370] A. Whalen, *Detection of signals in noise*. New York, USA: Academic Press, 1971.

[371] B. L. Yeap, C. H. Wong, and L. Hanzo, "Reduced complexity in-phase/quadrature-phase turbo equalisation with iterative channel estimation," in *IEEE International Communications Conference 2001*, (Helsinki, Finland), 11-15 June 2001. Accepted for publication.

[372] 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), pp. 185–189, 27-29 March 2000.

[373] W. Lee, "Estimate of channel capacity in Rayleigh fading environment," *IEEE Transactions on Vehicular Technology*, vol. 39, pp. 187–189, August 1990.

[374] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation techniques in wireless packet data services," *IEEE Communications Magazine*, vol. 38, pp. 54–64, January 2000.

[375] J. Blogh and L. Hanzo, *3G Systems and Intelligent Networking*. John Wiley and IEEE Press, 2002. 2001 (For detailed contents, please refer to http://www-mobile.ecs.soton.ac.uk.).

[376] L. Hanzo, P. Cherriman, and J. Streit, *Wireless Video Communications: From Second to Third Generation Systems, WLANs and Beyond*. IEEE Press, 2001. For detailed contents please refer to http://www-mobile.ecs.soton.ac.uk.).

[377] L. Hanzo, C. Wong, and P. Cherriman, "Channel-adaptive wideband video telephony," *IEEE Signal Processing Magazine*, vol. 17, pp. 10–30, July 2000.

[378] L. Hanzo, P. Cherriman, and E. Kuan, "Interactive cellular and cordless video telephony: State-of-the-art, system design principles and expected performance," *Proceedings of IEEE*, September 2000.

Subject Index

C

D

W

Author Index

G

H

M

N

O

P

R

S

T

U

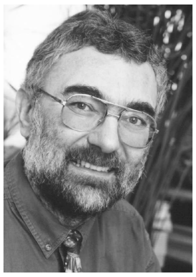
V

W

Y

Z

About the Authors



Lajos Hanzo (http://www-mobile.ecs.soton.ac.uk) received his degree in electronics in 1976 and his doctorate in 1983. During his 25-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the Department of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has co-authored eight books on mobile radio communications, published over 300 research papers, organised and chaired conference sessions, presented overview lectures and been awarded a number of distinctions. Currently he is managing an academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European IST Programme and the Mobile Virtual Centre of Excellence (VCE), UK. He is an enthusiastic supporter of industrial and academic liaison and he offers a range of industrial courses. He is also an IEEE Distinguished Lecturer. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk

T.H. Liew received the B.Eng degree in Electronics Engineering from the University of Southampton, U.K. In 2001 he was awarded the degree of PhD by the same university and currently he is continuing his research as a postdoctoral research fellow. His current research interests are associated with coding and modulation for wireless channels, space-time coding, adaptive transceivers, etc. He published his research results widely.

Dr Bee Leong Yeap...

Other Related Wiley/IEEE Press Books

- L. Hanzo, W.T. Webb, T. Keller: Single- and Multi-carrier Quadrature Amplitude Modulation: Principles and Applications for Personal Communications, WATM and Broadcasting; IEEE Press-John Wiley, 2000 [2]
- R. Steele, L. Hanzo (Ed): Mobile Radio Communications: Second and Third Generation Cellular and WATM Systems, John Wiley-IEEE Press, 2nd edition, 1999, ISBN 07 273-1406-8
- L. Hanzo, F.C.A. Somerville, J.P. Woodard: Voice Compression and Communications: Principles and Applications for Fixed and Wireless Channels; IEEE Press-John Wiley, 2001
- L. Hanzo, P. Cherriman, J. Streit: Wireless Video Communications: Second to Third Generation and Beyond, IEEE Press, 2001
- L. Hanzo, C.H. Wong, M.S. Yee: Adaptive wireless transceivers: Turbo-Coded, Turbo-Equalised and Space-Time Coded TDMA, CDMA and OFDM systems, John Wiley, 2002
- J.S. Blogh, L. Hanzo: Third-Generation Systems and Intelligent Wireless Networking - Smart Antennas and Adaptive Modulation, John Wiley, 2002

[2]For detailed contents please refer to http://www-mobile.ecs.soton.ac.uk

Blurb

For the sake of completeness and wide reader appeal, virtually no prior knowledge is assumed in the field of channel coding. In **Chapter 1** we commence our discourse by introducing the family of convolutional codes and the hard- as well as soft-decision Viterbi algorithm in simple conceptual terms with the aid of worked examples.

Chapter 2 provides a rudimentary introduction to the most prominant classes of block codes, namely to Reed-Solomon (RS) and Bose-Chaudhuri-Hocquenghem (BCH) codes. A range of algebraic decoding techiques are reviewed and worked examples are provided.

Chapter 3 elaborates on the trellis-decoding of BCH codes using worked examples and characterises their performance. Furthermore, the classic Chase algorithm is introduced and its performance is investigated.

Chapter 4 introduces the concept of turbo convolutional codes and gives a detailed discourse on the Maximum Aposteriory (MAP) algorithm and its computationally less demanding counterparts, namely the Log-MAP and Max-Log-MAP algorithms. The Soft Output Viterbi Algorithm (SOVA) is also highlighted and its concept is augmented with the aid of a detailed worked example. Then the effects of the various turbo codec parameters are investigated.

Chapter 5 comparatively studies the trellis structure of convolutional and turbo codes, while **Chapter 6** characterises turbo BCH codes. **Chapter 7** is a unique portrayal of the novel family of Redundant Residue Number System (RNS) based codes and their turbo decoding. **Chapter 8** considers the family of joint coding and modulation based arrangements, which are often referred to as coded modulation schemes. Specifically, Trellis Coded Modulation (TCM), Turbo Trellis Coded Modulation (TTCM), Bit-Interleaved Coded Modulation (BICM) as well as iterative joint decoding and demodulation assisted BICM (BICM-ID) are studied and compared under various narrow-band and wide-band propagation conditions.

In **Chapter 9 and 10** space-time block codes and space-time trellis codes are introduced. Their performance is studied comparative in conjunction with a whole host of channel codecs, providing guide-lines for system designers. As a lower-complexity design alternative to multiple-transmitter, multiple-receiver (MIMO) based schemes the concept of near-instantaneously Adaptive Quadrature Amplitude Modulation (AQAM), combined with near-instantaneously adaptive turbo channel coding is introduced in **Chapter 11**.

Based on the introductory concepts of **Chapter 12**, **Chapter 13** is dedicated to the detailed principles of iterative joint channel equalisation and channel decoding techniques known as turbo equalisation. **Chapter 14** provides theoretical performance bounds for turbo equalisers, while **Chapter 15** offers a wide-ranging comparative study of various turbo equaliser arrangements. The problem of reduced implemenattional complexity is addressed in **Chapter 16**. Finally, turbo equalised space-time trellis codes are the subject of **Chapter 17**.