On Lowering the Error-Floor of Low-Complexity Turbo-Codes

by

ZELJKO BLAZEK
M.A.Sc, University of Victoria, 1998
B.Eng., University of Victoria, 1989

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

We accept this dissertation as conforming
to the required standard

Dr. V.K. Bhargava, Co-Supervisor,	Dept. of Elect. & Comp. Eng.
Dr. T.A. Gulliver, Co-Supervisor,	Dept. of Elect. & Comp. Eng.
Dr. K.F. Li, Department Member,	Dept. of Elect. & Comp. Eng.
Dr. J. Muzio, Outside Member,	Dept. of Computer Science
Dr. I.J. Fair, External Examiner,	Dept. of ECE, Univ. of Alberta

Supervisor: Dr. V.K. Bhargava, Dr. T.A. Gulliver

ABSTRACT

Turbo-codes are a popular error correction method for applications requiring bit error rates from 10^{-3} to 10^{-6}, such as wireless multimedia applications. In order to reduce the complexity of the turbo-decoder, it is advantageous to use the simplest possible constituent codes, such as 4-state recursive systematic convolutional (RSC) codes. However, for such codes, the error floor can be high, thus making them unable to achieve the target bit error range.

In this dissertation, two methods of lowering the error floor are investigated. These methods are interleaver selection, and puncturing selective data bits. Through the use of appropriate code design criteria, various types of interleavers, and various puncturing parameters are evaluated. It was found that by careful selection of interleavers and puncturing parameters, a substantial reduction in the error floor can be achieved.

From the various interleaver types investigated, the variable s-random type was found to provide the best performance. For the puncturing parameters, puncturing of both the data and parity bits of the turbo-code, as well as puncturing only the parity bits of the turbo-code, were considered. It was found that for applications requiring BERs around 10^{-3}, it is sufficient to only puncture the parity bits. However, for applications that require the full range of BER values, or for applications where the FER is the important design parameter, puncturing some of the data bits appears to be beneficial.

Examiners:

Dr. V.K. Bhargava, Co-Supervisor,	Dept. of Elect. & Comp. Eng.
Dr. T.A. Gulliver, Co-Supervisor,	Dept. of Elect. & Comp. Eng.
Dr. K.F. Li, Department Member,	Dept. of Elect. & Comp. Eng.
Dr. J. Muzio, Outside Member,	Dept. of Computer Science
Dr. I.J. Fair, External Examiner,	Dept. of ECE, Univ. of Alberta

Table of Contents

List of Figures

List of Tables

Notation

FEC	forward error correction
BER	bit error rate
FER	frame error rate
CC	constituent code
CC1	first constituent code
CC2	second constituent code
AWGN	additive white gaussian noise
Q	Q-function
E_b/N_0	ratio of energy per bit to one-sided noise spectral density
SNR	signal to noise ratio
K	length of data word
N	length of code word
R	code rate
d	weight of codeword
w	weight of data word
d_{min}	minimum distance of a code
w_{min}	input weight causing d_{min}
$d_{min,x}$	minimum distance of a code caused by a weight-x input
$A(w,d)$	input-output weight enumerating function (IOWEF)
$A(d)$	weight enumerating function (WEF)
RSC	recursive systematic convolutional
k	number of inputs bits for a convolutional code
n	number of output bits for a convolutional code
m	memory length of a convolutional code
R_{eff}	effective code rate
L_a	a priori information
L_e	extrinsic information

SISO	soft-input soft-output decoder
MAP	maximum a posteriori probability
SOVA	soft-output viterbi algorithm
$P_2(d)$	pairwise error probability
ADS	average distance spectrum
BL	block
PR	pseudo-random
SR	s-random
MSR	modified s-random
VSR	variable s-random
FSTC	fully systematic turbo code
PSTC	partially systematic turbo code
p_u	ratio of unpunctured data bits to total data bits
p_{p_1}	ratio of unpunctured parity bits to total parity bits for the first constituent code.
p_{p_2}	ratio of unpunctured parity bits to total parity bits for the second constituent code.
f:XX	puncture mask family

Acknowledgement

It has been a long and interesting journey. My many thanks to those who have helped along the way.

Chapter 1

Introduction

Wireless communication systems, whether they carry voice, video or data are becoming more commonplace these days. There are many challenges to engineering wireless communication systems, one of which is dealing with the harsh wireless transmission medium. A wireless link is inherently more error prone than its wireline cousin, due to noise, fading and interference. A number of techniques also exist to help combat these problems, with different techniques functioning at different layers of the transmission system. One common technique that functions to alleviate errors is forward error correction (FEC). Typically, wireless systems exchange information between a source and destination as information packets. These packets often have a relatively short length, generally somewhere between 100 and 1000 data bits. Each packet may contain, for example, a short (20 ms) voice sample or a single message between two computer systems. FEC is used to detect and then correct errors in these information packets.

Forward error correction works by adding a certain number of redundant bits to each packet (using an encoder) before it is transmitted, based on the data contained within the packet. These redundant bits are usually referred to as the parity, or parity bits. When the packet is received, these parity bits are used to detect and correct any errors that may have occurred (using a decoder). The number of errors that can be corrected is determined by how many parity bits were added. Generally, the more parity added, the more error correction is possible. The disadvantage to adding more parity bits is that these bits use system resources that could otherwise be used for transmitting data. Thus, a key design criteria for an FEC code is to balance the amount of added redundancy with the desired error correction capability.

Many different FEC codes exist. A number of these have been applied to digital wireless systems. Which type of code to use has generally been determined by the bit error rate

(BER) required by the application. Voice and multi-media applications generally require moderately-low BERs on the order of 10^{-3} to 10^{-5}, whereas data applications require low BERs below 10^{-6} [1, 2]. Often data applications will incorporate some form of Automatic Repeat Request (ARQ) system at higher network layers, to further lower the BER. Some examples of the application of FEC codes include Global System for Mobile (GSM), where a convolutional code was used, and Cellular Digital Packet Data (CDPD) modems where a Reed-Solomon (RS) code was used. [1].

Turbo-codes have now become a popular alternative for many third generation (3G) wireless standards, both as a replacement for convolutional codes at moderately-low BERs and for data applications with a BER requirement on the order of 10^{-6} [2]. Turbo-codes were first introduced in [3]. They have become very popular because they provide remarkable error correction capability for relatively low decoding complexity. They are formed by the parallel concatenation of simple convolutional codes, usually referred to as constituent codes (CCs), and are decoded using an iterative decoding algorithm.

1.1 Significance of Research

The complexity of a turbo-code decoder implementation is directly related to the complexity of implementing the constituent code decoders. Thus, it is advantageous from an implementation perspective, to use constituent codes with a small number of states. However, turbo-codes using constituent codes with a small number of states tend to not perform as well, especially in the error floor region. This is even more evident for the small block lengths to be considered in this dissertation.

This dissertation investigates methods to improve the performance of such turbo-codes, with the goal of reducing the BER error floor below 10^{-6}, while at the same time keeping the good performance in the waterfall region. This investigation will be applied to a binary turbo-code with a 4-state constituent code, and overall code rate of $1/2$. These methods should also be applicable to turbo-codes with larger constituent codes, and different code rates.

The specific methods for lowering the error floor, that are investigated in this dissertation, are interleaver selection/design, and selective puncturing of data bits.

1.2 Outline

This dissertation consists of 9 chapters.

Chapter 2 provides background information.

Chapter 3 discusses the code design methods used.

Chapter 4 discusses interleaver construction, and gives some simulation results.

Chapter 5 applies the code design methods of Chapter 3, to the interleavers discussed in Chapter 4, and compares the code design and simulation results for these interleavers.

Chapter 6 introduces and compares partially systematic and fully systematic punctured turbo-codes.

Chapter 7 gives a detailed study of the partially systematic turbo-codes and presents simulation results using randomly chosen interleavers.

Chapter 8 combines the results of previous chapters by applying code design methods to the selection of interleavers and puncturing patterns, and then comparing them with simulation results.

Chapter 9 provides conclusions and suggestions for future work.

Chapter 2

Background

This chapter presents some background information on the channel model, weight spectrum and turbo-codes.

2.1 Channel Models

The simulation results and bounds presented in this work are given for two types of channels, the additive white gaussian noise (AWGN) channel, and the Rayleigh fading channel. A detailed description of the channel models used can be found in [4]. The important details are presented here. For both channels, the signalling method used is binary phase shift keying (BPSK), with a bit value of zero mapping to (-1) and a bit value of one mapping to (+1).

The AWGN channel is commonly used to model a wireline channel, however, it also serves as an approximate lower bound on the performance of a wireless channel. The probability of bit error for the AWGN channel is [4]

$$P_{AWGN} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{2.1}$$

where Q is the Q-function [4], E_b is the energy per bit and N_0 is the one-sided noise spectral density. The ratio E_b/N_0 is also called the signal to noise ratio (SNR).

The Rayleigh fading channel is commonly used to model the outdoor wireless environment. It provides an upper bound on the performance of a wireless channel. With the Rayleigh fading model, there is the assumption of slow, frequency non-selective, fading, with ideal interleaving. Slow fading means that the fading amplitude is constant over one or more bit periods. Ideal interleaving means that the fading amplitude of any two bits

is uncorrelated. For convenience, the Rayleigh fading channel will simply be called the fading channel. The probability of bit error for the fading channel is [4]

$$P_{fade} = \frac{1}{2}\left(1 - \sqrt{\frac{\overline{\gamma}}{1+\overline{\gamma}}}\right) \tag{2.2}$$

where $\overline{\gamma} = (E_b/N_0)\overline{\alpha^2}$, which is the average value of the SNR, α is the fading amplitude, and E_b and N_0 are as defined above.

2.2 Weight Spectrum

The weight spectrum of a code will be used in the code design methods described in Chapter 3. This section gives some useful background and definitions. More details can be found in [5]

A packet of information, or data, is usually called an *information word*, *data word*, or *input word*. After encoding, the resulting packet is called a *codeword* or *output word*. The length of the input word is K bits, and the length of the codeword is N bits. The ratio K/N is called the *code rate*, R. The terms *block length* or *block size* refer to the length of the input word, K.

The *weight* of a codeword is the number of non-zero bits in the codeword, and is given by d. Similarly, the weight of an input word is the number of non-zero bits in the input word, and is given by w.

For a linear code, such as a turbo-code, the *minimum distance* is the weight of the lowest weight nonzero codeword. The minimum distance is given by d_{min}, and the associated codeword is given by c_{min}. Sometimes, only the codewords that are caused by an input word of a given weight are considered. In this case, the minimum distance of the codewords caused by a weight-x input word is given by $d_{min,x}$, and the associated codeword is $c_{min,x}$.

The *weight spectrum*, or *distance spectrum*, of a code is the count of the number of codewords of every possible weight. It is usually given by the *input-output weight enumerating function* (IOWEF) $A(w,d)$, or the *weight enumerating function* (WEF) $A(d)$. The relationship between the two functions is

$$A(d) = \sum_{w=1}^{K} A(w,d). \tag{2.3}$$

The term *weight enumerator* will be used to refer to either of these two functions. The value of the weight enumerator, $A(d)$ or $A(w, d)$, is also referred to as the multiplicity of the codewords of weight d.

2.3 Turbo Codes

Turbo-codes are a relatively new branch of error-correcting codes, having only been introduced in 1993 [3]. They have become very popular because they provide remarkable error correction capability for relatively low decoding complexity. The following sections present a brief overview of turbo-codes. Details can be found in [6].

2.3.1 Turbo Encoder

An example of the general structure of a turbo-code encoder is shown in Figure 2.1, where d is the data bits and p_1 and p_2 are the parity bits. The components of the encoder are the interleaver and the constituent code (CC) encoders CC1 and CC2. The CC encoders shown in Figure 2.1 are rate 1/2 systematic encoders. The code rate is simply the ratio of input bits to output bits. A systematic encoder outputs the data and parity bits separately, such that the data bits are inserted, without modification, into the codeword. Although any finite error-correcting code can be used as a constituent code, the most common type is a recursive systematic convolutional (RSC) code. The constituent codes are described in more detail in Section 2.3.2. The interleavers are described in more detail in Chapter 4.

The output of the turbo-encoder in Figure 2.1 consists of the original systematic data followed by the parity data from the two CC encoders, giving a code word of the form

$$(d_0, p_{10}, p_{20}, d_1, p_{11}, p_{21}, d_2, p_{12}, p_{22}, ...). \tag{2.4}$$

This code is a rate 1/3 turbo-code. By using puncturing, the rate of the turbo-code can be increased. One common puncturing scheme alternatively chooses between the parity from the first and second CC encoders, giving a code word of the form

$$(d_0, p_{10}, d_1, p_{21}, d_2, p_{12}, d_3, p_{23}, ...), \tag{2.5}$$

and results in a rate 1/2 turbo-code.

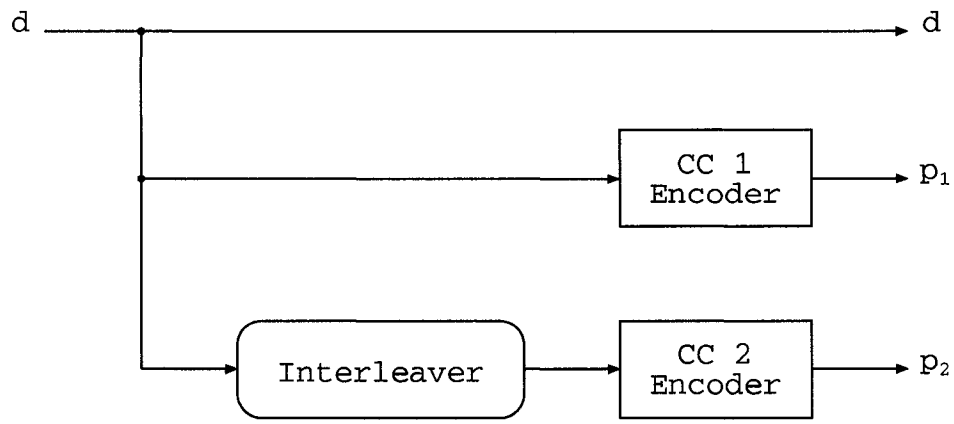

Figure 2.1. *Turbo Encoder*

2.3.2 Constituent Codes

As already mentioned, the constituent codes used in the turbo-code are usually recursive systematic convolutional codes. Only details of the RSC codes are presented here, although most of the description is applicable to all convolutional codes.

The RSC encoder is represented by a shift register with feedforward and feedback taps as determined by the generator polynomials for the particular code. Additional parameters associated with the code are often given as the triplet (n, k, m). For every k input bits, the encoder generates n output bits, thus, giving a code rate of k/n. The memory length, m, of the code is the length of the shift register. The number of states in the code is 2^m, for the binary codes considered here. A block diagram of a (2, 1, 2) RSC encoder is shown in Fig. 2.2, with the feedback and feedforward paths labelled. The feedback and feedforward generator polynomials for this code are 7 and 5, respectively.

An alternative view of a convolutional code is provided by the code trellis. This is simply a graph of the code output that has been folded onto itself by eliminating the replication in the graph. The trellis for the above code is given in Fig. 2.3. The nodes on the graph identify the state of the encoder and the branches indicate the input/output relationship for a transition from one node/state to another. For a given input stream, the output of the encoder can be found by starting from the zero state, and tracing a path along the trellis, taking the branches as indicated by the input for the given transition, and generating the output as indicated by the output for each transition.

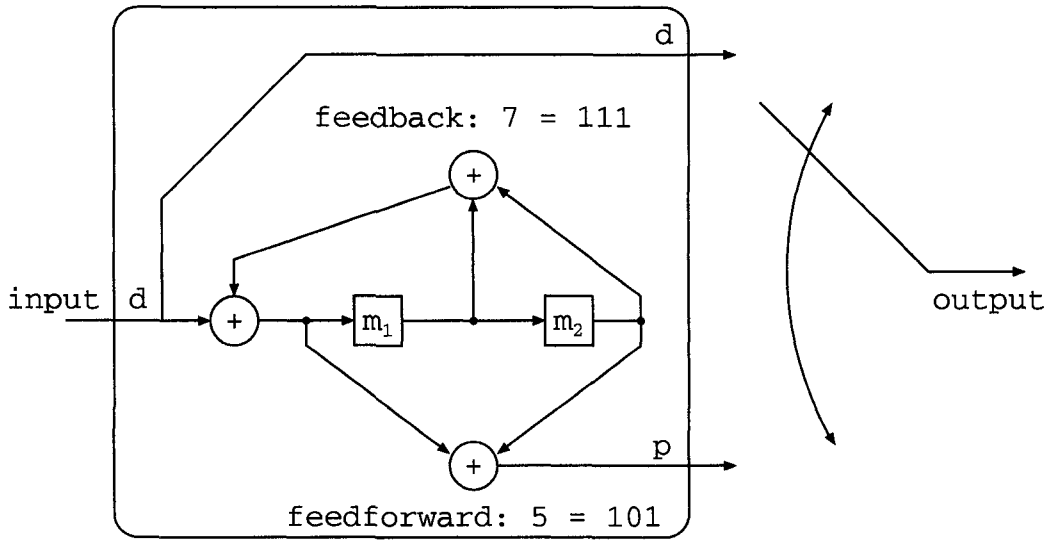

Figure 2.2. *RSC Encoder*

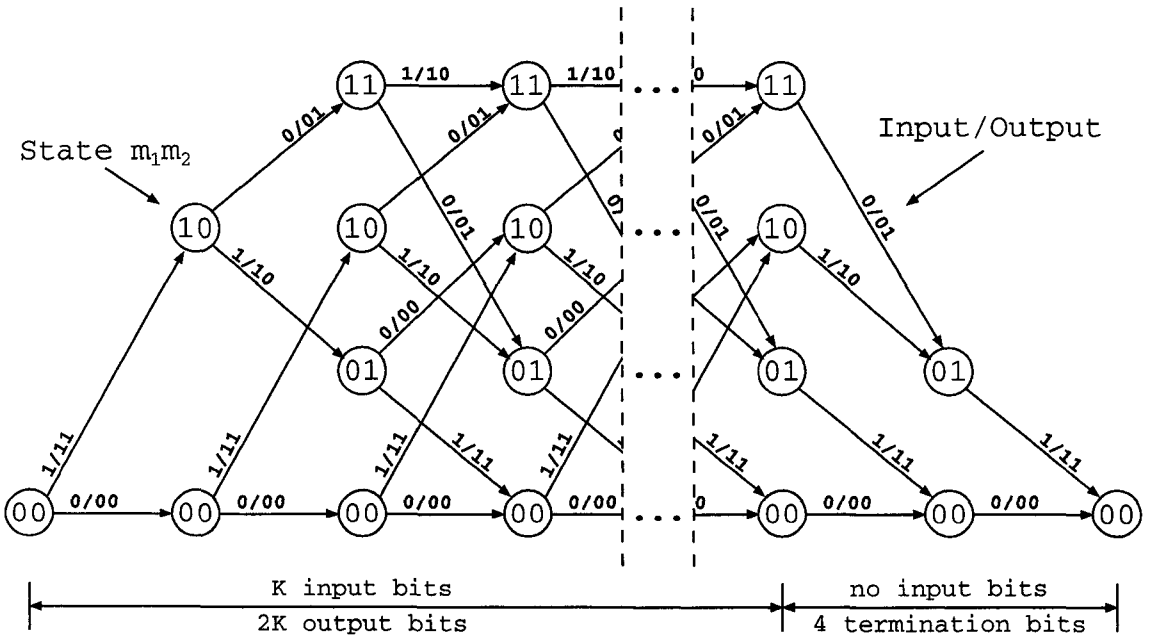

Figure 2.3. *Trellis Diagram*

For data that is partitioned into packets, the data in the encoder is usually "flushed", to ensure that all the input data reaches the output. This is known as trellis termination and is accomplished by appending tail bits to the input stream. The number of tail bits to append is equal to the memory of the code. If tail bits are appended to the input stream, then this affects the length of the code word generated by the encoder and thus the overall rate of the code. The effective rate, R_{eff}, of the code now becomes

$$R_{eff} = \frac{k}{n}\left(\frac{K}{K+m}\right). \tag{2.6}$$

For $K >> m$, the effect of the tail bits on the code rate is negligible.

The main reason that RSC codes are used as constituent codes is that they are recursive. The benefit of this is that a single "1" bit in the input followed by a sequence of zeros, will generate an output sequence that does not return to zero, but continues to generate non-zero output. Thus, a weight-1 input word generates an output word of much higher weight. The weight of the output word is proportional to the length of that portion of the input word, following the initial "1" bit. Thus, if the length of the input word is infinite, the weight of the output word will also be infinite.

In fact, for an RSC code, a second "1" is required in the input stream to bring the output back to all zeroes. This has a significant impact on the weight spectrum of the turbo-code. By varying the spacing between the successive ones, through the use of an appropriate interleaver, different weight code words can be generated with input data of the same weight. This allows the number of low weight code words generated by the turbo-encoder to be reduced, and thus improves error rate performance.

As an example, if the input sequence is

$$1000000\ldots \tag{2.7}$$

then the output of the RSC encoder will be

$$11010100\,\mathbf{010100}\ldots \tag{2.8}$$

with the last six bits (as highlighted in bold) repeating continuously.

2.3.3 Turbo Decoder

In Figure 2.4, we show the general structure of the turbo-decoder that corresponds to the turbo-encoder given in Figure 2.1. The turbo-code decoder has analogous components

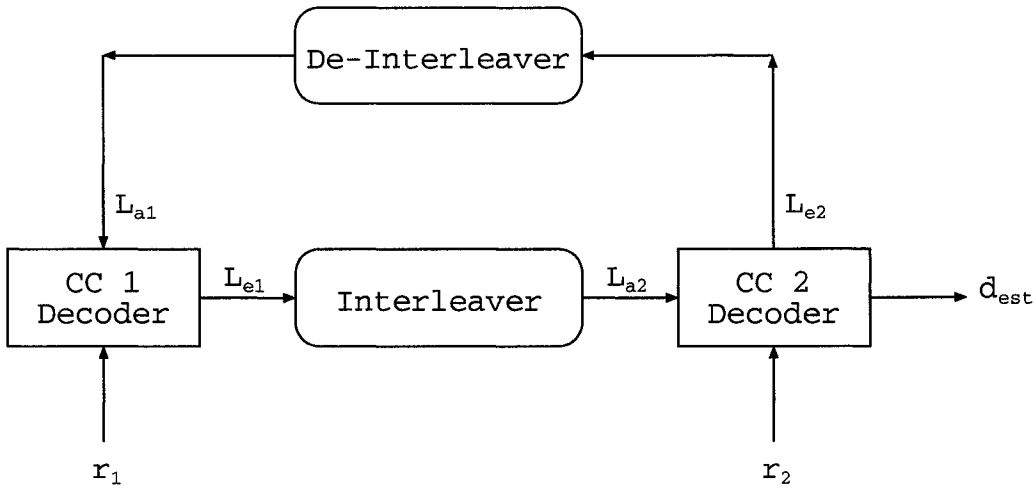

Figure 2.4. *Basic Turbo Decoder Structure*

to the turbo-code encoder, using CC decoders that correspond to the CC encoders of the turbo-encoder. In the turbo-decoder case, both a de-interleaver and interleaver are required.

The received CC code words, r_1 and r_2, are formed from the received turbo-code word. Both r_1 and r_2 contain the received information bits, plus the received parity bits from the corresponding CC. Note that the extrinsic information, L_e, output from one CC decoder is fed to the input of the other as a priori information, L_a, recursively. Thus, the decoding process can be repeated over several iterations, refining the results with each iteration. The final output of the decoder, d_{est}, is simply

$$d_{est} = r_2 + L_{a2} + L_{e2}. \tag{2.9}$$

To illustrate the benefits of iteration in the decoding process, the BER simulation results for a typical turbo-code are given in Fig. 2.5. This code was simulated for 20 full iterations of the turbo-decoding algorithm, where a full iteration means that both the CC1 and CC2 decoder were used. Several iterations, from the 1st iteration up to the 20th iteration, are identified on the figure. From the figure, it is seen that the BER performance steadily improves with each iteration. The amount of improvement is greatest for the low iterations, and gradually becomes less as the iteration number increases. For example, at a SNR of 3dB, the difference in BER between the first and second iteration is approximately an order of magnitude, whereas the difference in BER between the 10th and 20th iteration is approximately a factor of 2.

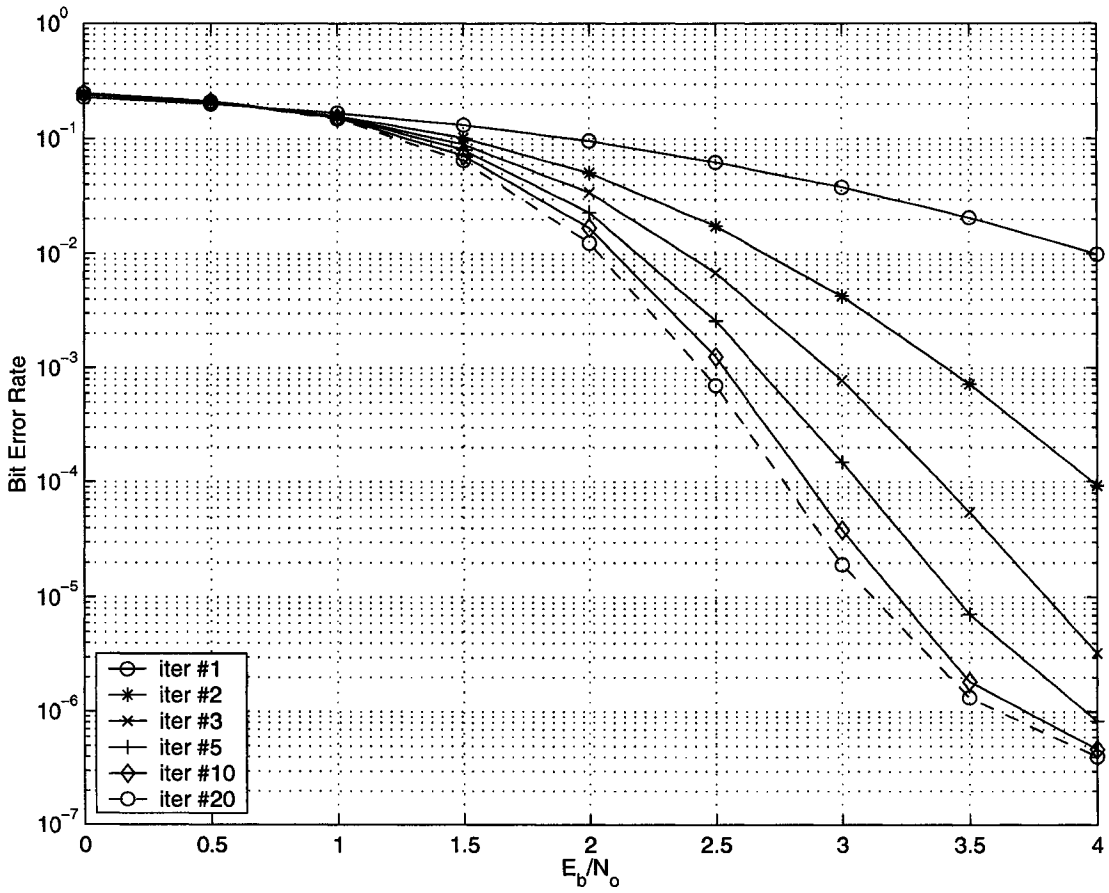

Figure 2.5. *Improvement in BER over several iterations*

In order for the iterative decoding process to work, the extrinsic information must be in the form of a soft-value. This is achieved by using CC decoders which are soft-input/soft-output (SISO) decoders. What this means is that the decoder accepts and generates a measure of the reliability of each data bit.

2.3.4 Soft-Input Soft-Output Decoders

The two main categories of soft-in/soft-out decoders are based on the maximum a posteriori probability (MAP) algorithm [7] and the soft-output Viterbi algorithm (SOVA) [8]. For the binary turbo-code case, the MAP algorithm is generally accepted to be about twice as complex as the SOVA. However, when the SOVA is used within a turbo-decoder, an extra 0.3dB to 0.7dB in SNR is required in order to achieve a comparable BER, depending on the block size used and the SNR [9].

To reduce the complexity of the original MAP algorithm, simplifications such as the Log-MAP and Max-Log-MAP algorithms have been introduced. The Log-MAP algorithm provides the same performance as the original MAP algorithm, but reduces some of the numerical problems in the original algorithm. The Max-Log-MAP algorithm provides a simplification of the Log-MAP algorithm with a small loss in performance, but still performs better than the SOVA. A comprehensive comparison of these algorithms is given in [10].

To illustrate the performance differences between these three SISO decoders, the BER simulation results for a typical turbo-code are given in Fig. 2.6. For each BER curve, one of the SISO decoders was used as part of the turbo-decoder, and 20 full iterations of the turbo-decoding algorithm were simulated. The figure clearly shows the relative performance of the three SISO decoders, with the LogMAP-based decoder giving the best results, and the SOVA-based decoder giving the worst results. For example, the SOVA-based decoder requires an extra 0.3dB over the LogMAP-based decoder, to achieve a BER of 10^{-4}. Note that at a high BER above 10^{-3}, the SOVA-based decoder provides better performance than the MaxLogMAP-based decoder; however, BER values above 10^{-3} are not useful for the applications considered here. On the other hand, at a low BER below 10^{-6}, the LogMAP-based decoder and MaxLogMAP-based decoder provide similar performance. In this case, the LogMAP-based decoder achieves this performance level with less iterations than required by the MaxLogMAP-based decoder.

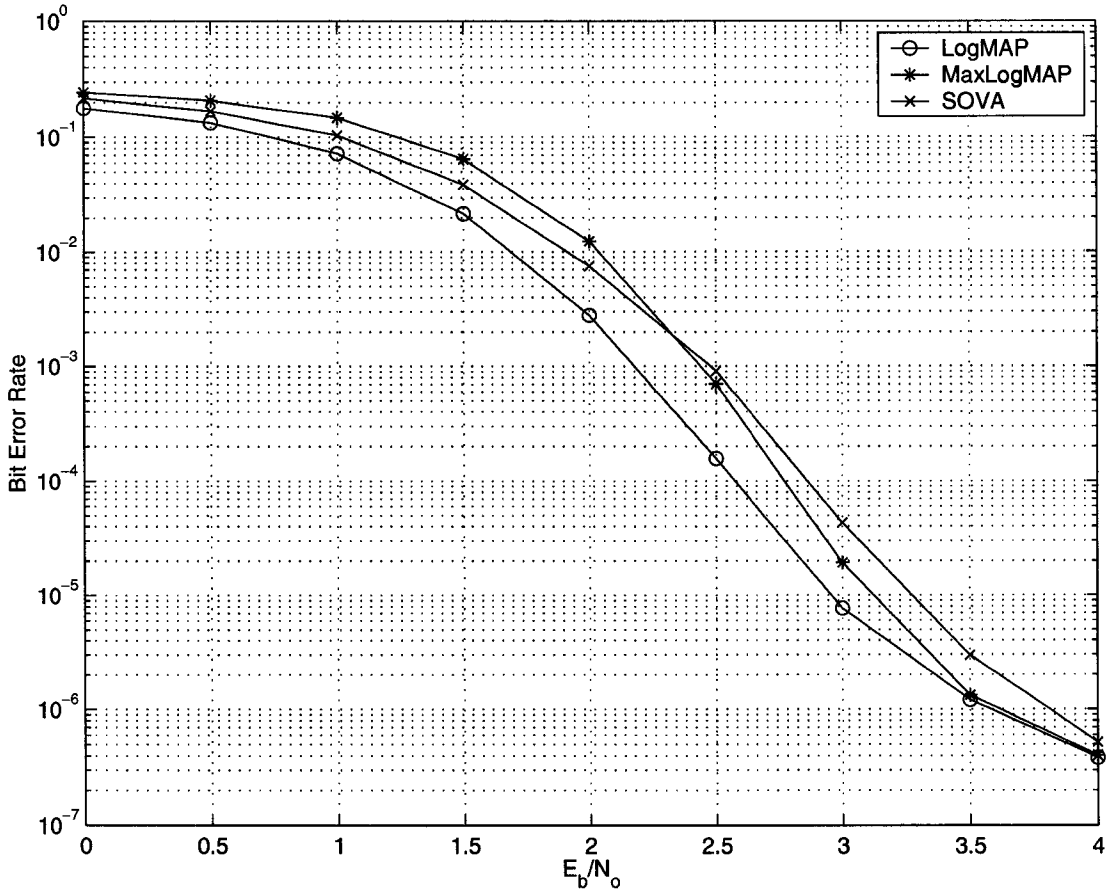

Figure 2.6. *Comparison of BER for 3 SISO Algorithms*

The memory length, m, of the constituent codes is another important parameter in determining the complexity of the turbo-decoder. Since the complexity of the CC decoder grows exponentially with the memory length, it is desirable to keep the memory length fairly short. This is because each CC decoder will be iterated on many times, and so a complex CC decoder will cause a considerable increase in complexity of the overall turbo-code decoder. However, CCs with higher memory length (up to about $m = 5$) tend to give better error rate performance when used within a turbo-code. Thus, there is a trade-off between error correction performance and decoder complexity. For example, an $m = 4$ code has four times the decoding complexity of an $m = 2$ code. Thus, if these codes are used within a turbo-decoder, four iterations of the $m = 2$ code can be performed in about the same time it takes to perform one iteration of the $m = 4$ code.

Chapter 3

Code Design

This chapter examines techniques used for designing turbo-codes, and evaluating their performance. These techniques are applied in later chapters as part of the code design process. They are applicable to a variety of codes, not just turbo-codes.

The particular code design technique to use depends on the target BER being considered. Fig. 3.1 shows a typical BER curve of a turbo-code. On this figure are identified two regions of the BER curve, commonly referred to as the *waterfall* and *error-floor* regions. The waterfall region is generally characterized by a steeper slope of the BER curve, with the resulting larger decrease in BER for small increase in SNR. The error floor region is generally characterized by a more gentle slope of the BER curve, with the resulting small decrease in BER for larger increase in SNR. Note that there is also a small transition region. In the error floor region, performance is dominated by the structure of the code. In the waterfall region, two factors contribute to performance: the structure of the code, and the convergence characteristics of the iterative decoder.

Some code design techniques are better suited to the waterfall region, and some are better suited to the error floor region. By varying the turbo-code parameters, the position and slope of the regions can be changed [11].

There is no strict definition of these regions, in terms of the BER, but for the purposes of this work, the waterfall region will be defined as the region where the BER is between 10^{-3} and 10^{-5}. The error floor region will be defined as the region with a BER around 10^{-6}, or lower. The BERs associated with these regions are not strict, and some small variations will be used on occasion, especially with the error floor for simulation results. Often, the error floor will be defined by the highest SNR value used in the simulation. As a result, sometimes any BER around 10^{-5} or lower, will be considered as part of the error floor region.

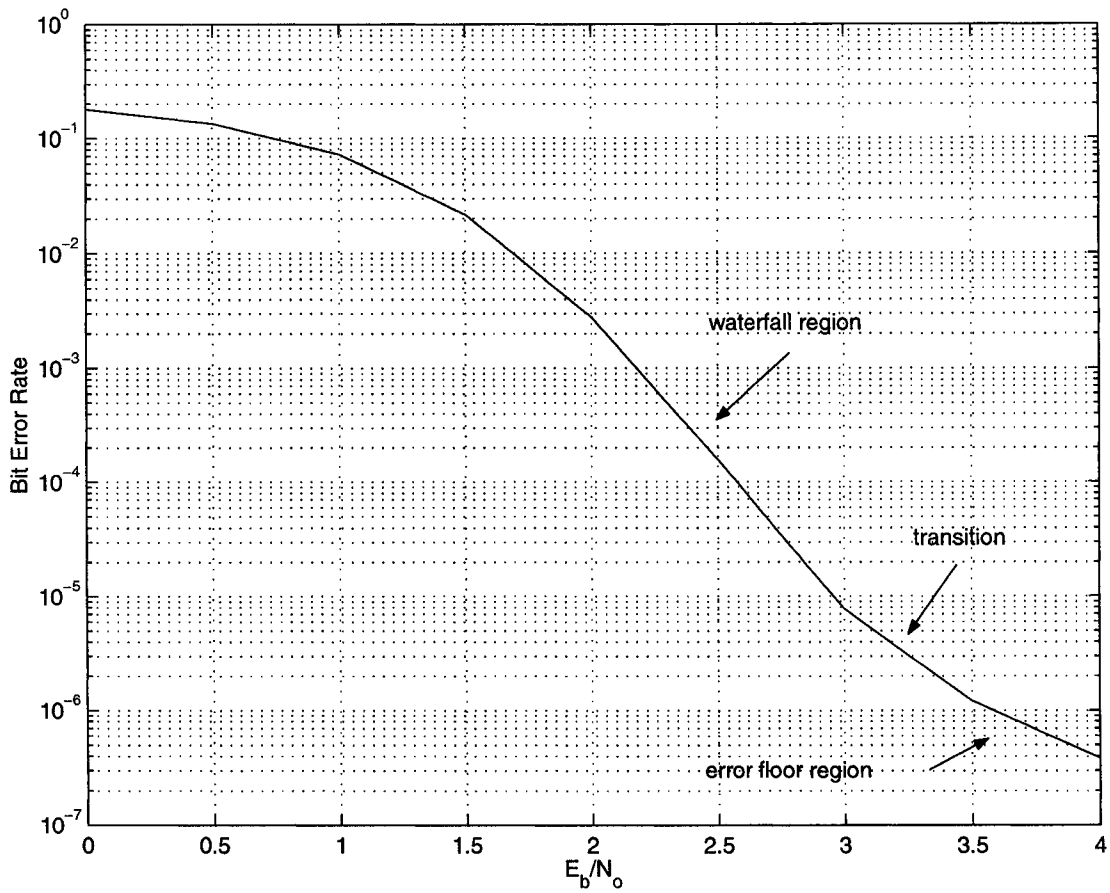

Figure 3.1. *Waterfall and Error-floor Regions*

3.1 Code Parameters

This section provides details on the particular turbo-code used throughout this dissertation.

This is a standard rate $1/3$ turbo-code with two identical rate $1/2$ constituent codes. These constituent codes are 4-state RSC codes with feedback and feedforward polynomials (expressed in octal) of 7 and 5, respectively. Since this is a 4-state code, the memory length $m = 2$. This is the constituent code shown in Fig. 2.2.

The trellis of each constituent code is independently terminated, using the method described in [12]. This method requires $2m$ bits to terminate the trellis of each code: m extra data bits, which are selected based on the final state of the encoder; and the associated m extra parity bits. Thus, for the CC considered here, this means two extra data bits, and two extra parity bits for each CC. These extra bits are added to the turbo-code codeword, giving an effective code rate of

$$R_{eff} = \frac{K}{3K + 8} \tag{3.1}$$

which, for the block lengths of interest, is only slightly less than $1/3$. For convenience, this will still be referred to as a rate $1/3$ code.

3.2 Union Bound on Performance

If the distance spectrum of a code is known, then the BER and frame error rate (FER) performance of the code, under maximum likelihood (ML) decoding, can be upper-bounded using the union bound. In maximum likelihood decoding, the decoder always selects the codeword that is closest to the received word.

Referring to [13, 14], the union bound on the FER and BER is

$$FER \leq \sum_{w=1}^{K} \sum_{d=d_{min}}^{N} A(w,d) P_2(d) \tag{3.2}$$

and

$$BER \leq \sum_{w=1}^{K} \sum_{d=d_{min}}^{N} \frac{w}{K} A(w,d) P_2(d) \tag{3.3}$$

respectively, where w, d, K, N, and $A(w,d)$ are as defined in Section 2.2, and $P_2(d)$ is the pairwise error probability for the given channel. Without loss of generality, assume that

the all zero codeword is transmitted. The pairwise error probability is the probability that a given incorrect codeword, of weight d, is selected by the decoder instead of the codeword that was actually transmitted.

Equations (3.2) and (3.3) require knowledge of the complete distance spectrum of the given code. Often this is difficult to obtain, due to the substantial computation time required, and so, only a partial distance spectrum is available. In this case, the summations are truncated to the number of available terms.

For the AWGN channel, the exact expression for the pairwise error probability is [4]

$$P_2(d) = Q\left(\sqrt{d\frac{2RE_b}{N_0}}\right). \tag{3.4}$$

The term (RE_b/N_0) is the SNR of the coded bit.

For the fading channel, the exact expression for the pairwise error probability is very difficult to evaluate [15]. An upper bound on the pairwise error probability is [15]

$$P_2(d) = \frac{1}{2}\left(1 - \left[\frac{R\overline{\gamma}}{1+R\overline{\gamma}}\right]^{1/2}\right)\left(\frac{1}{1+R\overline{\gamma}}\right)^{d-1} \tag{3.5}$$

where the term $(R\overline{\gamma})$ is the average SNR of the coded bit.

The union bounds of (3.2) and (3.3) were applied in [16] to determine the upper bound on the average BER and FER. In that work, the bounds were averaged over all interleavers, and quickly became greater than 1 for small values of E_b/N_0. In [17], it is noted that union bounds that are averaged over an ensemble of codes are only valid for values of E_b/N_0 corresponding to rates above the computational cutoff rate, R_0. It is also noted that this explains the behaviour of the bounds in [16] for small values of E_b/N_0.

In this work, the union bounds are applied to specific codes, and thus are still valid for small values of E_b/N_0, although the bounds do become less tight as E_b/N_0 is decreased.

To better illustrate the bounds, the BER bound of (3.3) was applied to two sample codes. For the first code, the bound was calculated for the AWGN channel using (3.4) for the pairwise error probability. This bound is shown in Fig. 3.2, along with the simulation results for this code, and a second bound that will be explained later. For the second code, the bound was calculated for the fading channel using (3.5) for the pairwise error probability. This bound is shown in Fig. 3.3, along with the simulation results for this code, and a second bound that will be explained later. For the simulation results of both

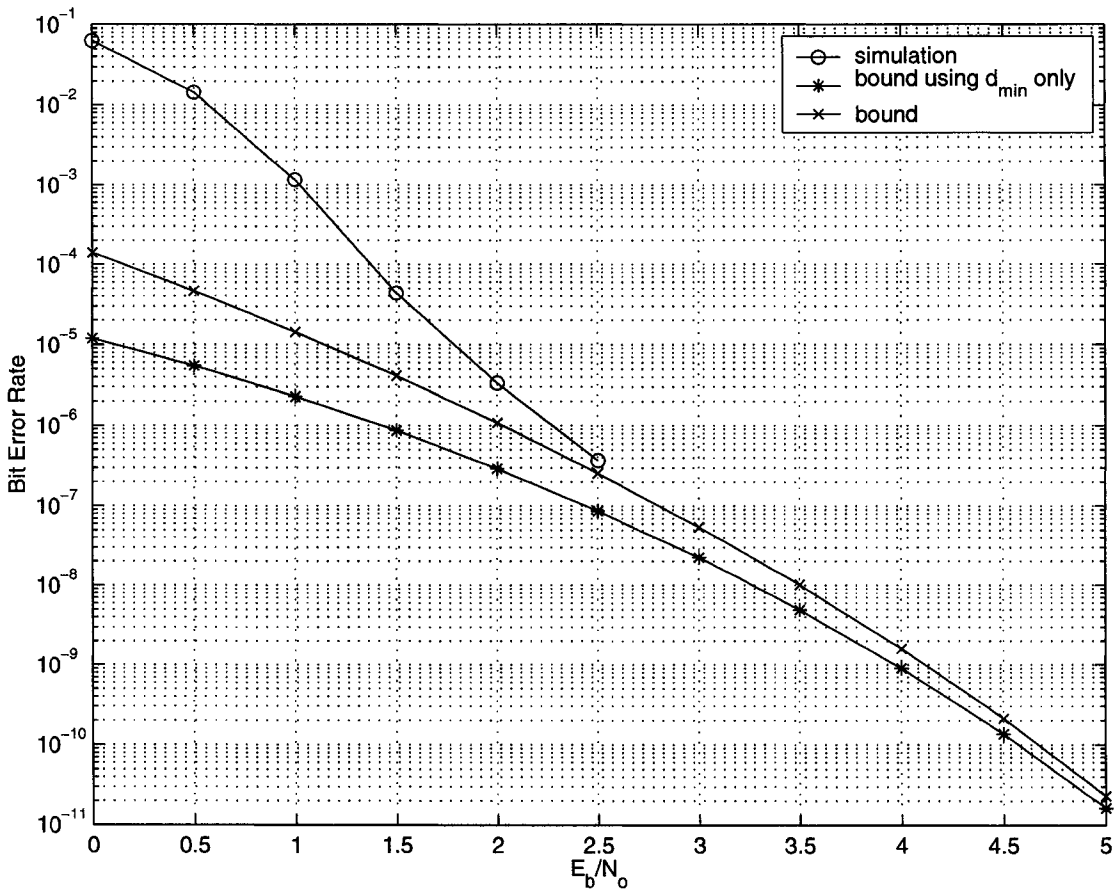

Figure 3.2. *Simulation and Bounds for AWGN Channel*

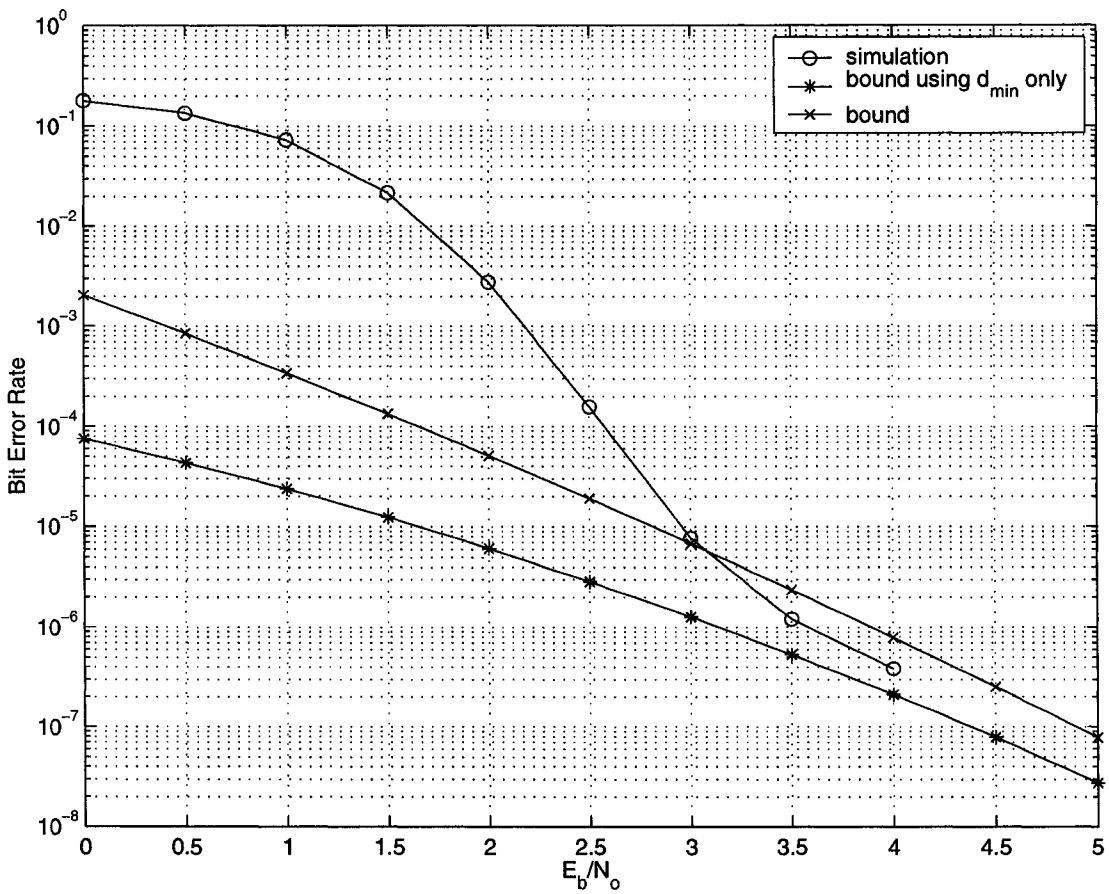

Figure 3.3. *Simulation and Bounds for Fading Channel*

codes, the LogMAP algorithm was used, with 20 full iterations of the turbo-decoder. This was done to get as much performance as possible from the codes.

In order to evaluate the bounds, the distance spectrums of the two codes had to be calculated. The distance spectrum was calculated by enumerating all the codewords caused by inputs of the given weight, and tabulating the weights of these codewords. For both of the codes considered, only the partial distance spectrum was calculated, due to the excessive computation time required to determine the complete distance spectrum. Specifically, the maximum input weight w, was chosen to be 4 and the maximum codeword weight d, was chosen to be 50. The maximum value of w was chosen so that the computation would complete in a reasonable amount of time. In addition, it is expected that higher input weights will not significantly contribute to the values of the bound. For the given values of w, all values of d are available for calculating the bound; however, it was found that using a value greater than 50 did not noticeably change the bound. Thus, the summation was truncated at 50 to reduce the computation time.

For the AWGN channel, the bound provides a good approximation of the error floor region. Even though the bound is an upper bound, it is below the simulation results. This occurs because the bound is an upper bound on the performance of a maximum-likelihood decoder, and the iterative turbo decoding algorithm is sub-optimum with respect to maximum-likelihood decoding.

For the fading channel, the bound provides a good approximation of the slope of the error floor region. However, in contrast to the AWGN case, the bound is above the simulation results. This is likely due to the fact that (3.5) is an upper bound for P_2, and not an exact expression, as is the case for the AWGN channel.

For both channels, there is a substantial difference between the bound and the simulation results in the waterfall region. This is due to the convergence characteristics of the iterative decoder.

3.3 Minimum Distance and Multiplicity

The most common design criteria for block codes is the minimum distance of the code. The minimum distance design approach to turbo-codes has been analysed in [16, 18] and [11] among others.

In [16, 18], the *effective free distance* of a turbo-code is defined as the minimum distance associated with weight-2 inputs. It is argued that the performance of a turbo-code is largely determined by the minimum distance due to weight-2 inputs.

In [11], it is shown how the minimum distance and the multiplicity at that minimum distance affects the performance in the error floor region. The concept of spectral thinning is also introduced, and the argument is made that, for blocklength K, as $K \rightarrow \infty$, the contribution of non weight-2 inputs on the distance spectrum goes to zero.

If only the minimum distance and associated multiplicity of the code is known, then the union bounds in (3.2) and (3.3) can be simplified to

$$FER \leq A(w_{min}, d_{min})P_2(d_{min}) \tag{3.6}$$

and

$$BER \leq \frac{w_{min}}{K}A(w_{min}, d_{min})P_2(d_{min}) \tag{3.7}$$

where w_{min} is the input weight associated with d_{min} , and the other terms have already been defined.

For the two codes considered in Figures 3.2 and 3.3, these simplified bounds were also calculated and are shown on those figures. In both cases the simplified bounds are below the regular bounds, although the gap decreases as the SNR is increased. This verifies the expected behaviour that the d_{min} term dominates at high SNR, but other terms become more important at low SNR. In the error-floor region, at a BER of 10^{-6}, the difference between the bounds for the AWGN channel is about 0.5dB, and for the fading channel, almost 1dB. This shows that for the regions of interest, the d_{min} term is not sufficient for estimating the performance, although it is still useful as a rough approximation.

In this dissertation, the minimum distance and multiplicity at that minimum distance will be used extensively as a design criteria, especially in the error floor region. This design criteria will commonly be referred to as the d_{min} criteria, and codes found using the d_{min} criteria will be called d_{min} codes.

3.4 Distance Spectrum Slope

In [19], the average distance spectrum (ADS) slope design criteria was introduced which takes into account the multiplicity of a number of low weight codewords. This design

procedure fits a line through the first 30 terms of the ADS, and determines the slope of the line. The slope provides a measure of the multiplicity of the low weight codewords. The objective is to find the code with the minimum slope. The ADS, as defined in [19], only includes the non-zero terms of the distance spectrum, and is calculated over all possible interleavers, using the uniform interleaver concept of [16].

In this dissertation, a slightly modified version of the ADS slope design criteria will be used. Specifically, the partial distance spectrum of an actual code will be used instead of the average distance spectrum. In addition, all terms of the distance spectrum, starting from $d = 1$ up to some maximum number of terms will be used, even if the associated multiplicity of the term is zero. In Section 3.2, when calculating the bounds for the sample codes, the maximum value of d was set to 50, which simply means that the first 50 terms of the distance spectrum were used. It seems reasonable to apply that same number here, in deciding the maximum value of d.

The basic design procedure is still the same. A line will be fitted through the first 50 terms of the partial distance spectrum of a group of codes, and the slope of this line will be determined. The code with the lowest slope will be selected.

This design criteria will be used mainly for the waterfall region. It will be referred to as the distance spectrum slope criteria, or simply, the slope criteria. Codes found using the slope criteria will be called slope codes.

3.5 Monte Carlo Simulation

One common method to evaluate the performance of a given code is through Monte Carlo simulation.

Monte Carlo simulations of the turbo-encoder/turbo-decoder were performed to generate all the simulation results presented in this thesis. The simulations were performed for specific values of the SNR and operated on one packet at a time until the stop criteria was met. The SNR values were chosen to give BER results below the error floor. The stop criteria required at least 1000 packets to be processed and 100 error events to be detected, where an error event is defined as a single frame error.

All of the simulations of the turbo-decoder were performed for a fixed number of full iterations of the decoder, where one full iteration consists of operating both CC decoders.

The exact number of iterations and SISO decoder used, vary depending on the simulations, and these details are provided when discussing particular simulation results. Perfect channel estimation was always assumed.

To allow for fair comparisons between the different coded and uncoded systems, the energy per packet was held constant. The error rate curves present the error performance against the data bit SNR. In the simulations, the code bit SNR was determined by multiplying the data bit SNR by the effective rate of the code.

Simulation can also be used for the purposes of code design, in comparing the relative merits of different codes. In order to compare the simulation results of two or more codes, it is useful to know the *confidence interval* and associated *confidence level* of the simulation results. The confidence level is expressed as the probability that the true BER is within the confidence interval. The confidence interval is an upper and lower bound on the true value of the BER, expressed in terms of the estimated BER. Similar definitions hold for the FER.

3.5.1 Confidence Interval

The following formula for the confidence interval is derived in [20], and so only the results are presented here.

The confidence level for a given simulation result, $1-\alpha$, is defined as

$$\text{Prob}\left[y_+ \leq p \leq y_-\right] = 1-\alpha \tag{3.8}$$

where

$$y_\pm = \hat{p}\left\{1 + \frac{d_{\alpha^2}}{2n}\left[1 \pm \sqrt{\frac{4n}{d_{\alpha^2}} + 1}\right]\right\} \tag{3.9}$$

and

- $\hat{p} = 10^{-k}$ = estimated probability of BER
- p = true value of BER
- n = number of samples in error
- $N = n10^k$ = total number of samples

The term d_α is chosen to satisfy the relation

$$\int_{-d_\alpha}^{d_\alpha} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = 1-\alpha \tag{3.10}$$

Eq. (3.10) can be rewritten as

$$2Q(d_\alpha) = \alpha \tag{3.11}$$

where $Q(x)$ is the Q-function.

As an example, for a confidence level of 95% and assuming $n = 100$ error events,

$$\begin{aligned} \alpha &= 0.05 \\ d_\alpha &= 1.96 \\ y_+ &= 1.22\,\hat{p} \\ y_- &= 0.82\,\hat{p}. \end{aligned} \tag{3.12}$$

This means that the true value of p is between $1.22\hat{p}$ and $0.82\hat{p}$, with a 95% confidence level. Note also that $y_+ = 1/y_-$.

3.5.2 Code Design Using Confidence Interval

The confidence interval will be used in evaluating codes based on their BER and FER performance. Normally, two codes at a time are compared.

The process is quite simple, and is outlined as follows. Assume two codes, A and B, with $BER_A < BER_B$. The confidence intervals of BER_A and BER_B will be computed and compared, and one of two decisions will be made:

- If the confidence intervals of BER_A and BER_B do not overlap, then code A is considered better in terms of BER performance.
- If the confidence intervals overlap, then A and B will be judged to be approximately equal in terms of BER performance.

The same process can be repeated for FER.

In this work, the 95% confidence interval will be used, along with 100 error events. Referring to (3.12), this gives a confidence interval of

$$[y_-,\ y_+] = [0.82\hat{p},\ 1.22\hat{p}]. \tag{3.13}$$

The process of comparing the confidence intervals reduces to testing the relation

$$\begin{aligned} 0.82\,BER_B &> 1.22\,BER_A \\ BER_B &> 1.49\,BER_A \end{aligned}$$

which can be rounded off for simplicity to

$$BER_B > 1.5\, BER_A \tag{3.14}$$

If this relation is true, then code A is better. If this relation is not true, then code A may or may not be better than code B, and so they will be considered *equivalent*.

Note that the equations in Section 3.5.1 assume that all the error events are independent. To ensure independent error events, frame errors will be counted rather than bit errors, since bit errors often occur in pairs for turbo-codes, and so it is more difficult to determine independent bit errors. Thus, by counting 100 frame error events, at least 100 independent bit error events are counted.

Chapter 4

Interleavers

As mentioned, one of the key components of turbo-codes is the interleaver. The choice of interleaver can often affect the bit error rate (BER) performance of a particular turbo-code by more than an order of magnitude.

The general function of interleavers is to map a block of input data to a block of output data (interleaving) using a fixed rule or mapping that re-orders the data in the block. The reverse of the rule is used to convert the re-ordered data back to its original form (de-interleaving).

Interleavers can be grouped into two general classes:

- random,
- structured.

The mapping used with random interleavers must be stored separately from the actual data, with the size of the storage equal to the size of the interleaver. Structured interleavers have a fixed algebraic rule which maps the interleaver input to its output. Thus, the mapping can be computed as required and does not require significant extra storage.

Random interleavers are generally considered to give better performance relative to structured interleavers (e.g. block interleavers), especially for large data block sizes [6]. For small block sizes (< 1000 bits), the performance differences have often been considered negligible [9], but we have found substantial improvements in some cases.

Two of the most common techniques for generating random interleavers are pseudo-random interleavers (PR) and S-random interleavers (SR) [12]. There are other techniques for generating random interleavers that have been presented in the literature, (e.g. [21, 22, 23]). However, these techniques are generally just variations of the S-random technique.

The PR and SR interleavers will be examined in the detail within this chapter, in addi-

Figure 4.1. *Operation of Block Interleaver*

0	1	2
3	4	5
6	7	8

Bit Write Order

0	3	6
1	4	7
2	5	8

Bit Read Order

tion to two variations of the SR interleaver. As a basis for comparison, the random interleavers will also be compared against block interleavers. The comparison will concentrate on interleaver sizes less than 1000 bits, specifically, interleaver/block sizes of 192, 400 and 900 bits will be considered.

4.1 Interleaver Construction

This section reviews the construction techniques for each of the interleavers under consideration.

4.1.1 Block Interleaver

With a block interleaver, the data is ordered into a two-dimensional matrix. The data is written into the interleaver matrix a row at a time, and is read out a column at a time. This is illustrated in Fig. 4.1, with the numbers in the matrix indicating the write and read order of individual data items. An m by n block interleaver holds a total of mn elements.

Alternatively, interleaving may be viewed as the re-ordering of the elements in a sequence of data. For the interleaver illustrated in Fig. 4.1, if the original data sequence is

$$0, 1, 2, 3, 4, 5, 6, 7, 8, \tag{4.1}$$

then the interleaved version of this data is

$$0, 3, 6, 1, 4, 7, 2, 5, 8. \tag{4.2}$$

4.1.2 Pseudo-Random Interleaver (PR)

To create a pseudo-random interleaver, a pseudo-random mapping is generated which maps the original data into the interleaved data. This mapping is simply a pseudo-random re-ordering of the indices of the original data block. As an example, if the original data sequence is given by (4.1), then one possible pseudo-random interleaver would produce the sequence

$$8, 0, 3, 6, 7, 5, 1, 2, 4. \tag{4.3}$$

4.1.3 S-Random Interleaver (SR)

This scheme was introduced in [12]. It is similar to the pseudo-random interleaver, except that restrictions are imposed during the construction of the mapping. Specifically, the following constraint must be satisfied:

$$\text{if} \quad |i - j| \leq S, \quad \text{then} \quad |p_i - p_j| \geq S, \tag{4.4}$$

where:

- i and j are indices into the original data sequence,
- p_i and p_j are indices into the interleaved data sequence corresponding to i and j, respectively.
- S is the S-value of the interleaver.

The purpose of this constraint is to reduce the number of low weight codewords caused by weight-2 inputs.

A mapping for an S-random interleaver can be constructed by building the mapping, one element at a time, ensuring that each new element matches the above criteria. If this process gets stuck, it backtracks a certain number of elements (possibly all the way to the beginning). For the interleavers generated for this dissertation, the process is considered stuck if a valid element cannot be found after trying 10000 random elements. Once the process is stuck, the current partial interleaver is discarded and a new interleaver mapping is started. In [12], an S-value around $\sqrt{K/2}$ is recommended, to give good BER performance without taking excessively long to generate the interleaver.

Table 4.1. *Interleaver Generation Time*

Type	Interleaver Size		
	192	400	900
MSR	23s	39s	56s
SR	29s	1273s	5790s

To illustrate an S-random interleaver with an example, if the original data sequence is given by (4.1), then one possible SR interleaver (using an S-value of 2) would produce the sequence

$$5, 8, 1, 6, 2, 7, 4, 0, 3. \tag{4.5}$$

4.1.4 Modified S-Random construction (MSR)

The time required to generate an S-Random interleaver can become excessive, especially for longer block sizes, and hence larger S-values. In some cases, it begins to approach the time required for simulation. In an effort to speed up the generation of the interleaver, a simple modification to the construction algorithm is presented. This is a straightforward modification of the mapping generation technique of the S-random interleaver. When the process gets stuck, rather than backtracking, the S-value is simply lowered by one for the next index only. After that index is generated, the original S-value is restored.

The important aspect of this technique is that temporarily relaxing the S-value does not significantly degrade the performance of the resulting interleavers. This is independent of the exact way in which the S-random interleavers are generated.

To show the efficiency of this modified technique in generating interleavers, the time taken to generate 200 interleavers of the required type and size is given in Table 4.1. It is quite evident that a substantial reduction in interleaver generation time is possible. This is especially true for larger interleaver sizes, where there is a factor of 100 difference in generation time for an interleaver of 900 bits in size.

4.1.5 Variable S-random construction (VSR)

This is another variation on the S-random interleaver. A very similar approach was independently discovered and documented in [23], where it is called a high-spread interleaver.

In the variable S-random construction, a modified set of constraints are used, namely

$$|i - j| + |p_i - p_j| \geq 2S, \tag{4.6}$$

where:

- i and j are indices into the original data sequence,
- p_i and p_j are indices into the interleaved data sequence corresponding to i and j, respectively.
- S is the S-value of the interleaver.

In addition, as with the MSR interleaver, the S-value is temporarily reduced by one if the sequence generation gets stuck.

To illustrate a variable S-random interleaver with an example, if the original data sequence is

$$0, ..., 15, \tag{4.7}$$

then one possible VSR interleaver (using an S-value of 2) would produce the sequence

$$12, 7, 10, 3, 14, 11, 8, 5, 0, 15, 6, 2, 13, 4, 1, 9. \tag{4.8}$$

4.2 Simulation Results

In this section, the BER and FER performance results of the different classes of interleavers are examined.

The error rate performance results are obtained by simulating the rate $1/3$ turbo-code of Section 3.1. Interleaver block sizes of 192, 400 and 900 bits are used, and results are obtained for both the AWGN and fading channels. The SOVA decoder is used in conjunction with 8 full iterations of the turbo-decoding algorithm.

For each of the random interleaver types, 200 interleavers were generated, and their BER and FER performance was determined. For the case of the SR and MSR interleavers,

Table 4.2. *Interleaver AWGN Channel BER Results*

length	SNR (dB)	type	best	worst	mean
192	3.0	BL	9.59e-06		
		PR	7.61e-06	4.23e-05	1.97e-05
		SR	1.28e-06	4.67e-06	2.11e-06
		MSR	1.36e-06	5.97e-06	2.34e-06
		VSR	1.18e-06	5.41e-06	1.83e-06
400	2.5	BL	1.76e-05		
		PR	7.05e-06	4.20e-05	1.73e-05
		SR	9.92e-07	2.97e-06	1.40e-06
		MSR	1.05e-06	6.53e-06	1.66e-06
		VSR	6.92e-07	3.08e-06	1.09e-06
900	2.0	BL	8.00e-05		
		PR	8.50e-06	4.62e-05	1.64e-05
		SR	1.41e-06	4.68e-06	2.05e-06
		MSR	1.54e-06	4.56e-06	2.22e-06
		VSR	1.44e-06	4.15e-06	2.16e-06

S-values of 9, 14, and 19 were used for interleaver sizes of 192, 400 and 900 bits respectively. For the VSR interleaver, the S-values do not have the same interpretation as for the SR and MSR interleavers. To provide a fair comparison, the S-values were chosen to be in the middle between the minimum and maximum separations of the SR interleaver. Thus, for the VSR interleaver, the S-values of 7, 11, and 15 were used for interleaver sizes of 192, 400 and 900 bits respectively.

The performance for the best, mean and worst interleaver for each random interleaver types is presented. For the block interleaver all combinations of row and column sizes were used in determining the interleaver with the best performance, and only this interleaver is shown. Tables 4.2, 4.3, 4.4 and 4.5 provide the detailed BER and FER results for both the AWGN and Rayleigh fading channels. These results are for the error floor region, and as such, are given for the highest SNR value that was used in the simulation. Note that this SNR value is different for each of the interleaver block lengths, and is thus noted in the tables.

Table 4.3. *Interleaver AWGN Channel FER Results*

length	SNR (dB)	type	best	worst	mean
192	3.0	BL	5.48e-04		
		PR	4.68e-04	2.75e-03	1.26e-03
		SR	7.13e-05	3.56e-04	1.33e-04
		MSR	7.60e-05	4.71e-04	1.46e-04
		VSR	7.20e-05	4.81e-04	1.16e-04
400	2.5	BL	1.52e-03		
		PR	9.48e-04	5.81e-03	2.38e-03
		SR	1.16e-04	3.76e-04	1.64e-04
		MSR	1.17e-04	1.08e-03	2.12e-04
		VSR	7.82e-05	5.04e-04	1.19e-04
900	2.0	BL	1.10e-02		
		PR	2.48e-03	1.47e-02	5.11e-03
		SR	3.48e-04	1.78e-03	4.79e-04
		MSR	3.83e-04	1.55e-03	5.41e-04
		VSR	3.37e-04	1.17e-03	5.26e-04

Table 4.4. *Interleaver Fading Channel BER Results*

length	SNR (dB)	type	best	worst	mean
192	4.5	BL	9.97e-06		
		PR	9.29e-06	4.99e-05	2.14e-05
		SR	1.41e-06	6.92e-06	2.49e-06
		MSR	1.69e-06	7.53e-06	2.70e-06
		VSR	1.33e-06	6.63e-06	2.16e-06
400	4.0	BL	1.33e-05		
		PR	5.24e-06	3.78e-05	1.36e-05
		SR	6.06e-07	1.61e-06	8.81e-07
		MSR	6.05e-07	4.72e-06	1.10e-06
		VSR	4.04e-07	2.08e-06	6.20e-07
900	3.5	BL	3.82e-05		
		PR	4.49e-06	3.79e-05	9.65e-06
		SR	4.82e-07	2.65e-06	6.83e-07
		MSR	5.02e-07	2.54e-06	7.84e-07
		VSR	4.91e-07	2.12e-06	7.61e-07

Table 4.5. *Interleaver Fading Channel FER Results*

length	SNR (dB)	type	best	worst	mean
192	4.5	BL	5.71e-04		
		PR	5.05e-04	3.30e-03	1.32e-03
		SR	6.71e-05	4.18e-04	1.42e-04
		MSR	8.21e-05	6.23e-04	1.55e-04
		VSR	7.09e-05	5.92e-04	1.25e-04
400	4.0	BL	1.44e-03		
		PR	5.64e-04	5.43e-03	1.95e-03
		SR	6.64e-05	2.81e-04	1.05e-04
		MSR	7.67e-05	7.45e-04	1.45e-04
		VSR	4.27e-05	3.66e-04	6.92e-05
900	3.5	BL	5.74e-03		
		PR	1.29e-03	1.30e-02	3.28e-03
		SR	1.15e-04	1.07e-03	1.63e-04
		MSR	1.20e-04	9.74e-04	2.03e-04
		VSR	1.16e-04	7.83e-04	1.99e-04

Several observations can be made from the results:

1. The block interleaver consistently gives a higher BER and FER (i.e. performs worse) than the best results from any of the random interleavers.
2. At a blocklength of 192, the BER and FER performance of the block interleaver is similar to the performance of the best PR interleaver, but as the blocklength is increased the PR interleaver becomes better.
3. The error rate performance of the mean SR interleaver is approximately the same or better than the performance of the best PR interleaver.
4. The best SR and VSR interleavers consistently have the best error rate performance.
5. All three of the s-random based interleavers give similar error rate performance, not just for the best interleavers, but also for the mean and worst interleavers.
6. The temporary relaxing of the S-value, as used in the MSR and VSR interleavers does not appear to have a detrimental affect on the error rate performance of the interleavers.
7. In all cases, the error rate performance of the best interleaver is at least an order of magnitude better than that of the mean PR interleaver. This suggests that choosing a good interleaver can give an order of magnitude improvement in the error rate, as compared to the average performance bounds calculated in [16].

From these results, it is quite clear that the s-random based construction generates interleavers with the best error rate performance, even for small interleaver sizes. This has been suggested in [12], but no direct comparison is available in the literature.

Chapter 5

Interleaver Design

This chapter applies the minimum distance and distance spectrum slope design techniques discussed in Chapter 3, to select the best interleavers. These techniques are applied to the interleavers that were presented in Chapter 4. This allows for a good evaluation of the effectiveness of a given technique, since the selected interleavers can be compared against the performance of all the interleavers to see how many of the best performing interleavers are chosen by the given technique, and also, how many of the badly performing interleavers are chosen by the technique. The ideal selection criteria would only choose the best interleavers. However, this is not likely to happen with a particular code design criteria, so the best that can be expected is that most of the selected interleavers, provide better than the average performance, and not too many bad interleavers are chosen. The focus of the investigation will be on the PR interleavers. These interleavers were chosen since they have the largest range of error rate values; thus it is easier to determine how well the selection criteria are performing.

In order to apply the selection criteria, it is necessary to calculate the distance spectrum of the code, or at least the partial distance spectrum. For the block lengths considered it is only practical, from a computation time perspective, to calculate the output weights for the first few input weights. Specifically, for information block lengths of 192 and 400, input weights up to 4 are used in the calculation, whereas for the block length of 900, input weights up to 3 are used in the calculation. The weight calculations are performed by explicitly enumerating all of the appropriate codewords, and then calculating and tabulating their weights.

5.1 Minimum Distance Properties

This section examines the minimum distance properties of the different interleaver types, by examining all the interleavers generated.

5.1.1 Minimum Distance Histograms

One perspective on the minimum distance information can be found by looking at a histogram of the minimum distances of each interleaver of a given interleaver type. This is shown in Fig. 5.1 through Fig. 5.4, for the four random interleaver types.

Each figure is for one interleaver type, and shows the histogram plots of the minimum distances for the three block lengths considered, namely 192, 400 and 900. The x-axis is the weight of the minimum distance codeword for a given interleaver. The y-axis is the number of interleavers with the given minimum distance. Also, the minimum distances associated with each input weight are given separately on the histograms. Thus, each interleaver appears once on the histogram for every input weight.

Referring to Fig. 5.1, a number of observations can be made about the PR interleavers:

1. The most common minimum distance is 10, and this is primarily caused by weight 2 inputs.
2. The minimum distances associated with weight 2 inputs appear relatively constant across all three block lengths.
3. The minimum distances associated with weight 3 inputs and weight 4 inputs slowly increase with increasing block length.
4. Weight 3 inputs cause a significant number of minimum distance codewords that are below the most common distance of 10, but this number decreases with increasing block length, as noted in the previous observation.
5. Weight 1 inputs generally produce the highest minimum distance codewords.

Referring to Fig. 5.2, a number of observations can be made about the SR interleavers:

1. The minimum distances associated with weight 1, 2 and 3 inputs increase with block length.
2. The minimum distances associated with weight-4 inputs appear relatively constant with block length. It has been suggested in [24], that for s-random interleavers, the

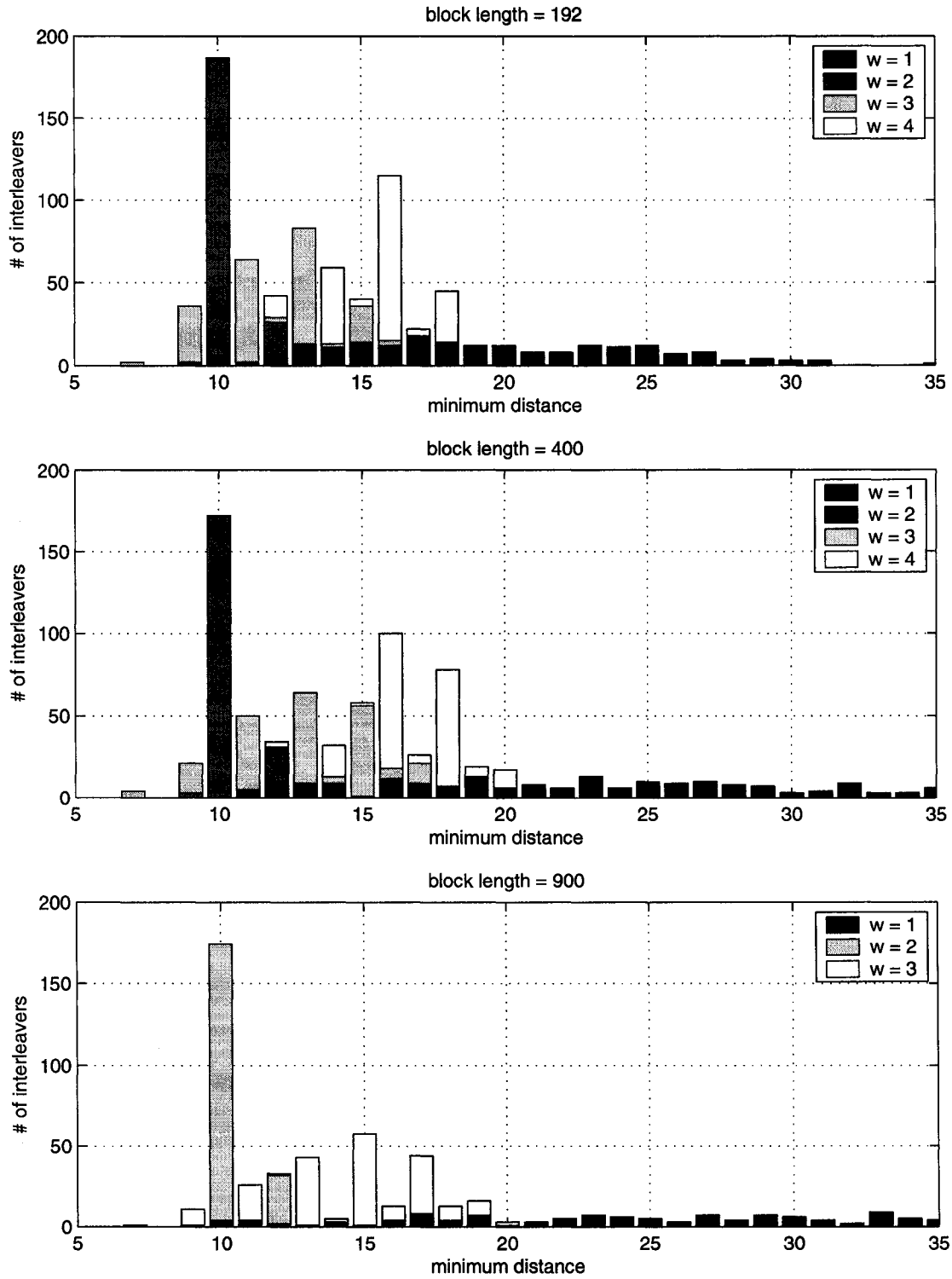

Figure 5.1. *Histogram of PR Interleavers*

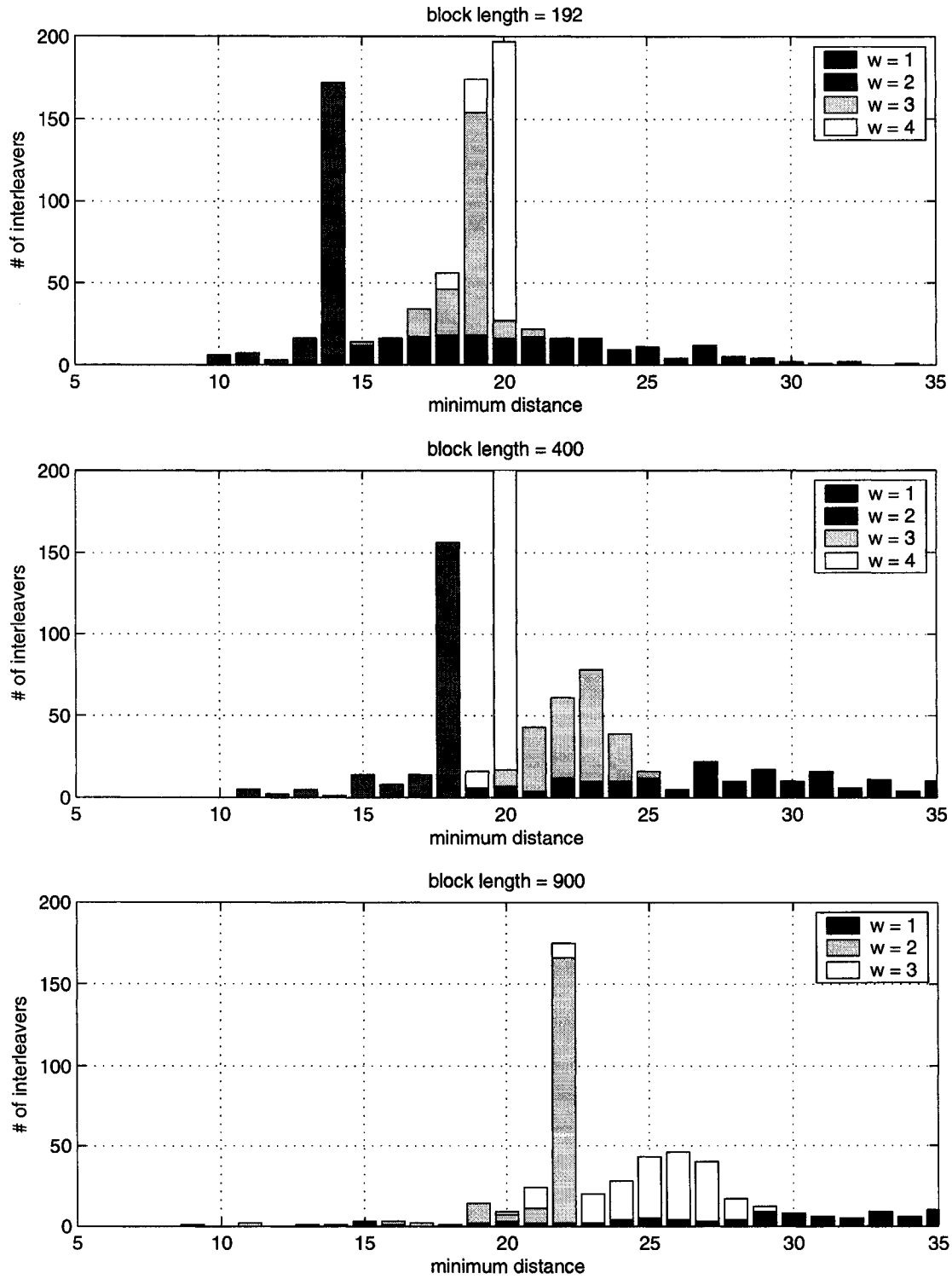

Figure 5.2. *Histogram of SR Interleavers*

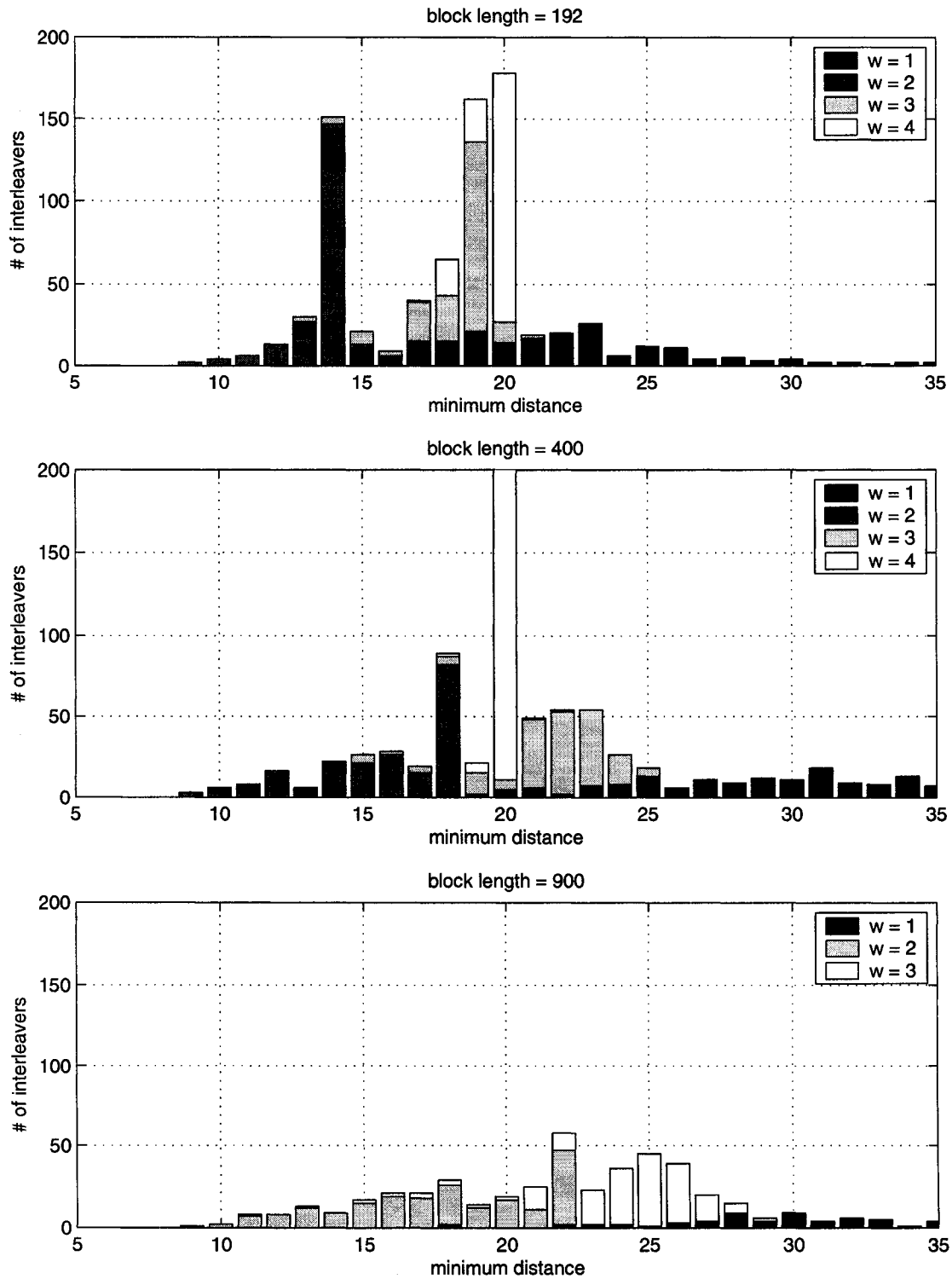

Figure 5.3. *Histogram of MSR Interleavers*

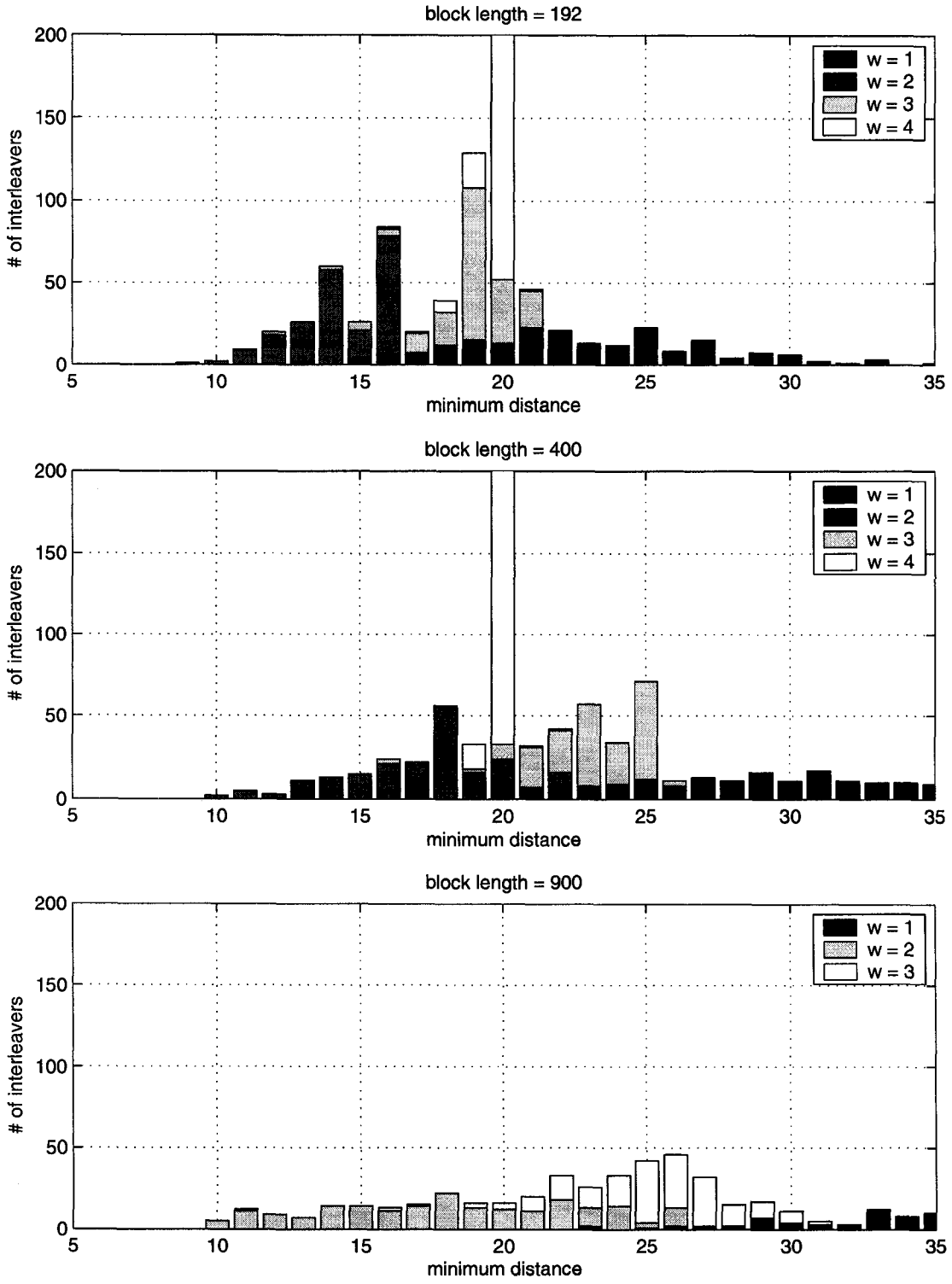

Figure 5.4. *Histogram of VSR Interleavers*

minimum distance caused by weight-4 inputs is constant, and thus does not increase with block length.

The MSR and VSR interleavers in Fig. 5.3 and Fig. 5.4, exhibit similar trends to the SR interleaver in Fig. 5.2.

Comparing between the s-random based interleaver types, a number of additional observations can be made:

1. For the MSR and VSR interleavers, the minimum distances for weight 2 inputs are spread over a larger range of values, relative to the SR interleaver. The increased spread of the minimum distances is likely caused by the relaxation of the s-values for the MSR/VSR interleaver construction techniques.
2. For the SR and MSR interleavers, the most common minimum distance for weight 2 inputs is the same between the two interleaver types, for all three block lengths. These values are 14, 18 and 22 for the block lengths of 192, 400 and 900, respectively. Also there are very few interleavers with minimum distances caused by weight 2 inputs above this value.
3. Compared to the other two interleaver types, for the VSR interleavers, the minimum distances associated with weight 2 inputs behave differently depending on the block length.
    - For a block length of 192, the most common minimum distance is higher by 2 at 16.
    - For a block length of 400, the most common minimum distance is the same at 18, but there are quite a few interleavers with higher minimum distances.
    - For a block length of 900, there is no definitive most common minimum distance, and there are quite a few interleavers with a higher minimum distance than the value of 22, which is the most common minimum distance for the SR and MSR interleavers.

Comparing between the PR and SR interleaver types, a number of additional observations can be made:

1. The SR interleaver construction effectively reduces the contribution of weight 2 inputs to the minimum distance, as intended by the construction method.

2. In addition, the SR interleaver construction also improves the situation for weight 1, 3, and 4 inputs as well.
3. For the PR construction, the minimum distance of weight 2 inputs appears relatively constant, whereas for the SR construction, the minimum distance of weight 4 inputs appears relatively constant.

The above observations can be summarized as follows:

1. The minimum distance of PR interleavers is lower than the SR interleavers for all block lengths, and the difference in d_{min} only gets larger as the block length is increased.
2. The SR interleaver construction is effective in reducing the contribution of weight 2 inputs on the minimum distance. For small block lengths (400 or less) this is sufficient as the weight 2 inputs are the primary cause of the minimum distance codewords.
3. For larger block lengths (> 400), the weight 4 inputs start to have a more significant influence on the minimum distance, and in this case the SR interleaver construction starts to become less than optimum, although still better than the PR construction.
    - In this case, construction techniques that break weight 4 inputs should be considered, such as [21, 22].
    - This is in agreement with the observations made in [25], that for larger block lengths, higher weight inputs are the cause of the minimum distance terms.
    - At first, this appears to contradict the spectral thinning argument of [11], which suggests that as the block length is increased the contribution of non weight 2 inputs on the distance spectrum is reduced. Two interpretations are:
        - Spectrum thinning is based on the average over all pseudo-random interleavers, and not on an arbitrary interleaver. In addition, the s-random based construction works to reduce the contribution of weight 2 inputs on the distance spectrum. Thus, spectrum thinning does not necessarily occur for a specific interleaver, especially one which is not pseudo-random.
        - Alternatively, the block length may simply not be long enough for spectrum thinning to take full effect.
4. The s-value relaxation technique of the MSR/VSR interleavers produces interleavers

Table 5.1. *Input Weight Contributions to Minimum Distance: 192*

Type	Value	$w = 1$	$w = 2$	$w = 3$	$w = 4$	Total
PR	count	7	159	42	2	210
	%	3.5	79.5	21.0	1.0	105.0
SR	count	11	194	0	0	205
	%	5.5	97.0	0.0	0.0	102.5
MSR	count	1	197	7	0	205
	%	0.5	98.5	3.5	0.0	102.5
VSR	count	2	190	13	0	205
	%	1.0	95.0	6.5	0.0	102.5

with minimum distances as good as or better than those generated by the SR technique, although the number of such interleavers is slightly lower.

5. The VSR construction technique appears to give interleavers with higher minimum distances than the SR/MSR techniques. However, this may be caused by the choice of s-value, and not by the superiority of the construction technique itself.

5.1.2 Input Weight Contributions to Minimum Distance

From Fig. 5.1 through Fig. 5.4, it is evident that different input weights cause the minimum distance code word in different cases. Table 5.1 through Table 5.3 are used to better illustrate this. Each entry shows the number of interleavers whose minimum distance codeword is generated by the given input weight. Note that the "Total" column is sometimes more than 100%, which indicates that more than one input weight causes the minimum distance codeword.

Some observations:

- For all interleaver types and block lengths, the weight 2 input causes the most minimum distance codewords, followed by the weight 3 input and lastly the weight 1 input input.
- With one notable exception, very few minimum distance codewords are caused by weight 4 inputs for the block lengths of 192 and 400.
- For the VSR interleaver of length 400, a surprisingly large number of interleavers

Table 5.2. *Input Weight Contributions to Minimum Distance: 400*

Type	Value	$w = 1$	$w = 2$	$w = 3$	$w = 4$	Total
PR	count	1	172	27	2	202
	%	0.5	86.0	13.5	1.0	101.0
SR	count	5	195	0	0	200
	%	2.5	97.5	0.0	0.0	100.0
MSR	count	1	188	14	0	203
	%	0.5	94.0	7.0	0.0	101.5
VSR	count	0	184	3	37	224
	%	0.0	92.0	1.5	18.5	112.0

Table 5.3. *Input Weight Contributions to Minimum Distance: 900*

Type	Value	$w = 1$	$w = 2$	$w = 3$	Total
PR	count	7	184	12	203
	%	3.5	92.0	6.0	101.5
SR	count	12	178	20	210
	%	6.0	89.0	10.0	105.0
MSR	count	2	186	18	206
	%	1.0	93.0	9.0	103.0
VSR	count	0	188	19	207
	%	0.0	94.0	9.5	103.5

have their minimum distance codeword generated by weight 4 inputs, with 37 interleavers in total. This can be explained by referring to the output weights of 19 through 22 in Fig. 5.4. It can be seen that there are quite a few interleavers whose weight 2 input minimum distance is at or above 19 and 20, which is where the weight 4 minimum distances are concentrated.

- A second observation with the VSR interleaver of length 400, is the value of 224 in the Total column. This indicates that 24 interleavers have their minimum distance codeword generated by more than one input weight. This is in contrast to the other interleaver types and sizes, where this happens for at most 10 interleavers. The cause of this may be the same as given in the previous observation.
- For the s-random based interleavers, the contribution of the weight 3 inputs is higher for the block length of 900, than for the smaller block lengths, with approximately 10% of interleavers having their minimum distance codeword generated by weight 3 inputs. This again suggests, that for longer block lengths, it is necessary to use construction techniques which take into account input weights higher than 2.

5.2 Code Design using Minimum Distance

This section examines the minimum distances and multiplicities of the best interleavers over all the interleaver types, and looks at the results of applying the minimum distance criteria to the PR interleavers.

5.2.1 Best Minimum Distance

This section takes a detailed look at the minimum distances and multiplicities of the best interleavers for each of the interleaver types and for each of the considered input weights.

For each interleaver type and for each input weight, one interleaver is selected as the best interleaver, according to the design criteria. The following tests are applied in order. If a given test selects several interleavers, then the next test is applied, otherwise the selection process stops.

1. Select the interleaver(s) with the largest $d_{min,w}$ for input weight w.

2. From the set of interleavers selected in the previous step, select the interleaver(s) with the smallest multiplicity, i.e. $A(w, d_{min,w})$.
3. From the set of interleavers selected in the previous step, select the interleaver by examining the minimum distance and multiplicity for the other input weights. The other input weights are ordered according to their associated minimum distance, from lowest to highest minimum distance. Minimum distances are considered first, across all the input weights, and then multiplicity is considered.

Tables 5.4, 5.5 and 5.6 list the best interleavers for block lengths of 192, 400 and 900 respectively. For each interleaver type and input weight, the best selected interleaver is shown. For this interleaver, the tuple

$$d_{min,w} \ , \ A(w, d_{min,w}) \tag{5.1}$$

is given for all of the input weights. The column labelled W gives the input weight that was used in the selection process. The column labels $w = 1, w = 2, ...$ give the values of w used in (5.1).

The $w = 0$ column gives the tuple

$$d_{min} \ , \ A(d_{min}) \tag{5.2}$$

Usually this matches one of the entries in the other columns, but in some cases the multiplicity may be higher. This occurs if more than one input weight gives the minimum distance term, in which case, the multiplicity in the $w = 0$ column is the sum of the multiplicities in the other columns.

The $W = 0$ row shows the interleaver that was selected when the selection process was not restricted to a particular input weight. This is usually, but not always, one of the interleavers selected for a given input weight.

Throughout this section, and later sections as well, each of the interleavers is denoted by a number. The 200 interleavers of each interleaver type are numbered, and this number simply denotes one of those 200 interleavers. This number is given in the column labelled Ilv #.

In Table 5.6, the $w = 4$ column is special, in that the $w = 4$ weight spectrum was calculated after the interleavers were selected, and so the weigh-4 inputs were not considered in the selection process.

By examining Tables 5.4, 5.5 and 5.6 a number of observations can be made:

Table 5.4. *Minimum Distance and Multiplicity: 192*

W	Ilv #	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
ALL	BL 012	(9, 1)	(9, 1)	(15, 1)	(19, 1)	(20, 181)
	BL 016	(9, 1)	(9, 1)	(15, 1)	(19, 1)	(20, 181)
0	PR 192	(12, 2)	(24, 1)	(12, 2)	(13, 2)	(16, 1)
1	PR 114	(10, 1)	(37, 1)	(10, 1)	(15, 1)	(16, 1)
2	PR 176	(12, 2)	(25, 1)	(12, 1)	(13, 1)	(12, 1)
3	PR 195	(10, 4)	(14, 1)	(10, 4)	(17, 5)	(16, 2)
4	PR 170	(10, 2)	(21, 1)	(10, 2)	(13, 1)	(19, 1)
0	SR 109	(16, 3)	(24, 1)	(16, 3)	(19, 3)	(20, 5)
1	SR 185	(14, 5)	(34, 1)	(14, 5)	(19, 2)	(20, 3)
2	SR 109	(16, 3)	(24, 1)	(16, 3)	(19, 3)	(20, 5)
3	SR 076	(14, 2)	(20, 1)	(14, 2)	(21, 6)	(20, 5)
4	SR 122	(16, 9)	(16, 1)	(16, 8)	(18, 1)	(20, 1)
0	MSR 020	(16, 5)	(19, 1)	(16, 5)	(19, 8)	(20, 6)
1	MSR 006	(14, 2)	(35, 1)	(14, 2)	(19, 5)	(20, 7)
2	MSR 020	(16, 5)	(19, 1)	(16, 5)	(19, 8)	(20, 6)
3	MSR 191	(14, 4)	(22, 1)	(14, 4)	(21, 4)	(20, 7)
4	MSR 132	(15, 1)	(23, 1)	(15, 1)	(17, 1)	(20, 1)
0	VSR 171	(16, 8)	(27, 1)	(16, 8)	(17, 1)	(20, 8)
1	VSR 089	(14, 1)	(36, 1)	(14, 1)	(19, 3)	(20, 6)
2	VSR 171	(16, 8)	(27, 1)	(16, 8)	(17, 1)	(20, 8)
3	VSR 091	(14, 1)	(15, 1)	(14, 1)	(21, 1)	(20, 5)
4	VSR 185	(13, 1)	(21, 1)	(13, 1)	(19, 1)	(21, 3)

Table 5.5. *Minimum Distance and Multiplicity: 400*

W	Ilv #	$w = 0$	$w = 1$	$w = 2$	$w = 3$	$w = 4$
ALL	BL 020	(9, 1)	(9, 1)	(22, 2)	(23, 1)	(20, 389)
0	PR 106	(14, 8)	(33, 1)	(14, 8)	(15, 1)	(18, 1)
1	PR 181	(10, 1)	(47, 1)	(10, 1)	(13, 1)	(16, 1)
2	PR 106	(14, 8)	(33, 1)	(14, 8)	(15, 1)	(18, 1)
3	PR 031	(10, 2)	(27, 1)	(10, 2)	(17, 1)	(16, 1)
4	PR 009	(12, 2)	(23, 1)	(12, 2)	(17, 4)	(20, 4)
0	SR 177	(18, 3)	(38, 1)	(18, 3)	(21, 1)	(20, 8)
1	SR 172	(18, 11)	(46, 1)	(18, 11)	(23, 1)	(20, 8)
2	SR 177	(18, 3)	(38, 1)	(18, 3)	(21, 1)	(20, 8)
3	SR 193	(18, 7)	(28, 1)	(18, 7)	(25, 2)	(20, 5)
4	SR 010	(18, 9)	(19, 1)	(18, 9)	(22, 2)	(20, 1)
0	MSR 040	(18, 6)	(29, 1)	(18, 6)	(23, 1)	(20, 4)
1	MSR 131	(14, 1)	(48, 1)	(14, 1)	(23, 1)	(20, 4)
2	MSR 040	(18, 6)	(29, 1)	(18, 6)	(23, 1)	(20, 4)
3	MSR 019	(16, 2)	(20, 1)	(16, 2)	(25, 2)	(20, 6)
4	MSR 041	(12, 1)	(41, 1)	(12, 1)	(20, 1)	(22, 17)
0	VSR 150	(21, 1)	(35, 1)	(21, 1)	(26, 6)	(22, 26)
1	VSR 008	(19, 1)	(45, 1)	(19, 1)	(23, 2)	(20, 5)
2	VSR 166	(20, 2)	(27, 1)	(22, 26)	(25, 1)	(20, 2)
3	VSR 066	(16, 1)	(27, 1)	(16, 1)	(26, 1)	(20, 8)
4	VSR 150	(21, 1)	(35, 1)	(21, 1)	(26, 6)	(22, 26)

Table 5.6. *Minimum Distance and Multiplicity: 900*

W	Ilv #	$w = 0$	$w = 1$	$w = 2$	$w = 3$	$w = 4$
ALL	BL 030	(9, 1)	(9, 1)	(27, 2)	(23, 1)	(20, 889)
0	PR 038	(12, 1)	(25, 1)	(12, 1)	(16, 1)	(18, 3)
1	PR 114	(10, 5)	(73, 1)	(10, 5)	(15, 1)	(16, 1)
2	PR 038	(12, 1)	(25, 1)	(12, 1)	(16, 1)	(18, 3)
3	PR 087	(10, 2)	(46, 1)	(10, 2)	(20, 1)	(20, 7)
0	SR 116	(22, 2)	(31, 1)	(22, 2)	(26, 1)	(20, 6)
1	SR 171	(22, 7)	(74, 1)	(22, 7)	(26, 1)	(20, 8)
2	SR 129	(21, 1)	(42, 1)	(22, 1)	(21, 1)	(20, 3)
3	SR 086	(17, 1)	(29, 1)	(17, 1)	(30, 1)	(20, 6)
0	MSR 012	(22, 4)	(48, 1)	(22, 4)	(25, 1)	(20, 5)
1	MSR 122	(13, 1)	(79, 1)	(13, 1)	(25, 1)	(20, 6)
2	MSR 012	(22, 4)	(48, 1)	(22, 4)	(25, 1)	(20, 5)
3	MSR 020	(13, 1)	(28, 1)	(13, 1)	(29, 1)	(20, 6)
0	VSR 086	(26, 35)	(31, 1)	(26, 35)	(27, 1)	(20, 2)
1	VSR 172	(22, 2)	(72, 1)	(22, 2)	(29, 1)	(20, 3)
2	VSR 086	(26, 35)	(31, 1)	(26, 35)	(27, 1)	(20, 2)
3	VSR 164	(19, 1)	(43, 1)	(19, 1)	(31, 2)	(20, 1)

1. For the block interleaver, $d_{min} = d_{min,1}$ for all three block lengths. This occurs because the last bit in an input word does not change its position after being interleaved by a block interleaver. As a result if this last bit is a 1, then only the tail bits will contribute to the weight of the resulting codeword. It is commonly believed, e.g. [11], that d_{min} is never caused by a weight-1 input, and in fact, that $d_{min,1} = \infty$. This is true only for infinite block lengths. For a finite block length $d_{min,1}$ is also finite.
2. For the s-random based interleavers, the problem mentioned in Section 5.1.1 about $d_{min,4}$ being stuck at a constant value is readily apparent. In virtually all of the interleavers, the value of $d_{min,4}$ is 20. Only in two cases is it above 20, the MSR041 and VSR150 interleaver for a block length of 400, and in both those cases, the multiplicity is noticeably higher.
3. For the PR interleavers, d_{min} of the best interleavers is relatively constant over the three block lengths, while for the s-random based interleavers, there is a slight increase in d_{min} of the best interleavers as the block length is increased. Based on the discussion in Section 5.1.1, this is to be expected.
4. For block lengths 192 and 400, the interleavers selected by considering weight-2 inputs tend to have the highest values of d_{min}. The only time that this is not the case is for the VSR interleaver for a block length of 400, where $d_{min,4} < d_{min,2}$ because $d_{min,4}$ is stuck at a value of 20. This confirms that optimizing for weight-2 inputs is a sufficient design criteria for these block lengths.
5. For the s-random based interleavers with a block length of 900, the interleavers selected by considering weight-2 inputs are as good as or better than the interleavers selected by the other input weights. Unfortunately, for these interleavers, $d_{min} = d_{min,4}$ in all three cases. This confirms that optimizing for weight-2 inputs at this block length is not sufficient, and weight-4 inputs must also be considered.

5.2.2 Code Design Results

A detailed examination of the application of the design criteria to the PR interleavers is presented in this section. The PR interleavers were chosen as they provide the broadest performance range, as compared to the s-random based interleavers. Thus, it is easier to determine the effectiveness of the design criteria.

Tables 5.7, 5.8 and 5.9 give the results for block lengths 192, 400 and 900 respectively. The tables show the BER and FER performance of the selected interleavers in the error floor region. In this case, a rather loose interpretation of the error floor is used, since many of the BER values are around 10^{-5}. This is particularly valid for PR interleavers, which do tend to have a slightly higher error floor.

In addition, the associated group statistics are given for each set of simulation conditions, in order to show the performance of the selected interleaver against all 200 interleavers. The values of *min*, *median*, and *max* are self-explanatory. The values of *25%* and *75%* are simply the boundaries of the top quartile and bottom quartile, respectively, of the 200 interleavers. The values of the selected interleavers are positioned horizontally in the tables to indicate in which quarter of the group statistics they belong.

By examining the tables, and with reference to (3.14) for comparing simulation results, some observations can be made:

- In all cases, the interleavers selected over all input weights, i.e. $w = 0$, were within the top quartile of all interleavers, and were equivalent to the best interleavers.
- The interleavers selected using $w = 1$ gave mixed results, but tended to get worse as K was increased. In fact, for $K = 900$, the selected interleaver was consistently in the bottom quartile of all interleavers. For $K = 192$, although the selected interleaver was in the top quartile, it was not equivalent to the best interleaver.
- The interleavers selected using $w = 3$ also gave mixed results, but tended to get better as K was increased. For the case of $K = 900$, the interleaver was within the top quartile, but only for the AWGN channel was it equivalent to the best interleaver.
- The interleavers selected using $w = 2$, were the only interleavers, selected using a specific input weight, that were consistently in the top quartile for all simulations. However, for $K = 192$ and the AWGN channel, the interleaver was not equivalent to the best interleaver. Note that in this case, the $w = 0$ interleaver is different from the $w = 2$ interleaver, as noted in Table 5.4.
- The interleavers selected using $w = 4$, gave better results for $K = 400$ than for $K = 192$. In fact, for $K = 400$, the error rate values were lower than for the $w = 2$ interleaver, although the two interleavers are considered equivalent.

The results for the other interleaver types were similar, although not as conclusive, since the spread of the simulation results is smaller. As a result, the best selected inter-

Table 5.7. *Design Selection Comparison, PR, 192: (A) BER, AWGN, snr=3.0dB; (B) FER, AWGN, snr=3.0dB; (C) BER, Fade, snr=4.5dB; (D) FER, Fade, snr=4.5dB*

(A)

W	Ilv #	min	25%	median	75%	max
		7.61e-06	1.60e-05	1.93e-05	2.29e-05	4.23e-05
0	192	1.09e-05				
1	114	1.38e-05				
2	176	1.47e-05				
3	195			1.86e-05		
4	170	1.50e-05				

(B)

W	Ilv #	min	25%	median	75%	max
		4.68e-04	1.00e-03	1.22e-03	1.45e-03	2.75e-03
0	192	5.95e-04				
1	114	8.38e-04				
2	176	7.83e-04				
3	195			1.37e-03		
4	170	9.99e-04				

(C)

W	Ilv #	min	25%	median	75%	max
		9.29e-06	1.73e-05	2.07e-05	2.44e-05	4.99e-05
0	192	1.09e-05				
1	114	1.53e-05				
2	176	1.35e-05				
3	195			2.09e-05		
4	170			1.87e-05		

(D)

W	Ilv #	min	25%	median	75%	max
		5.05e-04	1.06e-03	1.29e-03	1.53e-03	3.30e-03
0	192	5.82e-04				
1	114	9.62e-04				
2	176	6.50e-04				
3	195				1.59e-03	
4	170			1.16e-03		

Table 5.8. *Design Selection Comparison, PR, 400: (A) BER, AWGN, snr=2.5dB; (B) FER, AWGN, snr=2.5dB; (C) BER, Fade, snr=4.0dB; (D) FER, Fade, snr=4.0dB*

(A)

W	Ilv #	min	25%	median	75%	max
		7.05e-06	1.40e-05	1.64e-05	2.00e-05	4.20e-05
0	106	1.05e-05				
1	181		1.45e-05			
2	106	1.05e-05				
3	031			1.85e-05		
4	009	8.21e-06				

(B)

W	Ilv #	min	25%	median	75%	max
		9.48e-04	1.93e-03	2.28e-03	2.73e-03	5.81e-03
0	106	1.12e-03				
1	181	1.76e-03				
2	106	1.12e-03				
3	031		2.27e-03			
4	009	1.01e-03				

(C)

W	Ilv #	min	25%	median	75%	max
		5.24e-06	1.08e-05	1.29e-05	1.55e-05	3.78e-05
0	106	7.17e-06				
1	181		1.11e-05			
2	106	7.17e-06				
3	031			1.33e-05		
4	009	5.24e-06				

(D)

W	Ilv #	min	25%	median	75%	max
		5.64e-04	1.56e-03	1.85e-03	2.29e-03	5.43e-03
0	106	8.17e-04				
1	181	1.51e-03				
2	106	8.17e-04				
3	031		1.65e-03			
4	009	6.45e-04				

Table 5.9. *Design Selection Comparison, PR, 900: (A) BER, AWGN, snr=2.0dB; (B) FER, AWGN, snr=2.0dB; (C) BER, Fade, snr=3.5dB; (D) FER, Fade, snr=3.5dB*

(A)

W	Ilv #	min	25%	median	75%	max
		8.50e-06	1.40e-05	1.60e-05	1.90e-05	4.62e-05
0	038	1.08e-05				
1	114				2.09e-05	
2	038	1.08e-05				
3	087	1.13e-05				

(B)

W	Ilv #	min	25%	median	75%	max
		2.48e-03	4.18e-03	4.96e-03	5.99e-03	1.47e-02
0	038	2.70e-03				
1	114				7.04e-03	
2	038	2.70e-03				
3	087	3.40e-03				

(C)

W	Ilv #	min	25%	median	75%	max
		4.49e-06	8.00e-06	9.32e-06	1.09e-05	3.79e-05
0	038	6.21e-06				
1	114				1.15e-05	
2	038	6.21e-06				
3	087	7.42e-06				

(D)

W	Ilv #	min	25%	median	75%	max
		1.29e-03	2.64e-03	3.15e-03	3.89e-03	1.30e-02
0	038	1.66e-03				
1	114				4.39e-03	
2	038	1.66e-03				
3	087	2.38e-03				

leavers were not always within the top quartile, but were instead within the top half of all interleavers.

From the above observations, it is clear that using weight-1 or weight-3 inputs as a selection criteria, will not give good results. Using weight-2 inputs as a selection criteria gives good results for all three block lengths. Lastly, using weight-4 inputs tends to give better results for the larger of the two block lengths considered.

This is consistent with previous observations that for longer block lengths it is important to consider weight-4 inputs, whereas for smaller block lengths it is sufficient to only consider weight-2 inputs.

5.3 Code Design using Distance Spectrum Slope

This section examines the distance spectrum slope design criteria. It will mainly focus on applying the design criteria to the selection of interleavers. The performance of the selected interleavers in the waterfall region is then compared. Specifically, the required SNR to achieve a BER of 10^{-3}, for each of the selected interleavers, will be examined, and compared against the overall group.

In order to obtain results for a specific BER value, linear interpolation was applied to the simulation results for SNR points above and below the desired BER value. Consider two known points (B_1, S_1) and (B_2, S_2), where B_x is the base-10 logarithm of the actual BER, and S_x is the SNR. The objective is to find the interpolated point (B_i, S_i), where B_i is the known value. Assume that $B_1 > B_i > B_2$, which implies that $S_1 < S_i < S_2$, since the BER curve has a negative slope. Then

$$S_i = S_1 + \frac{B_i - B_1}{slope} \tag{5.3}$$

where

$$slope = \frac{B_2 - B_1}{S_2 - S_1}. \tag{5.4}$$

In applying the distance spectrum slope criteria, the decision has to be made as to how many terms of the distance spectrum to use. As discussed in Chapter 3, the first 50 terms of the distance spectrum was chosen as a reasonable value. Given that the minimum distance

of good interleavers is around 20, using the first 50 terms is similar to the choice of using the first 30 non-zero terms, as mentioned in [19]. Also, for comparison purposes, the first 30 terms of the distance spectrum are also used, which corresponds to approximately the first 10 non-zero terms. This is used to determine whether the higher number of terms is necessary or whether a smaller number of terms is sufficient. Thus, the maximum value of d will be either 30 or 50.

In addition, the slope will be determined for two types of distance spectrums:

- The distance spectrum resulting from weight-2 inputs only. Thus, $w = 2$ in this case.
- The distance spectrum resulting from all the available input weights. For the block lengths of 192 and 400, this includes up to weight-4 inputs, and so w will take on the values 1, 2, 3 and 4. For the block length of 900, this includes up to weight-3 inputs, and so w will take on the values 1, 2 and 3.

This is used to determine whether it is sufficient to use only weight-2 inputs, or whether higher weight inputs are necessary.

Combining the possible values of d and w as defined above gives a total of four different partial weight spectrums, for each block length. Each of these partial weight spectrums will be used in the calculation of the distance spectrum slope. The following labels will be used to refer to these partial weight spectrums:

- w0d50 : all input weights, first 50 terms.
- w0d30 : all input weights, first 30 terms.
- w2d50 : weight-2 inputs only, first 50 terms.
- w2d30 : weight-2 inputs only, first 30 terms.

Tables 5.10, 5.11 and 5.12 give the results for the block lengths of 192, 400 and 900, respectively. The tables show the interleavers that were selected based on the slope criteria, using each of the four partial weight spectrums. In some cases, more than one interleaver is listed, which simply means that all the listed interleavers had the same slope value. The tables also show the SNR required, by the selected interleaver, to achieve the desired BER of 10^{-3}, and how this SNR value compares against all 200 interleavers. The column headings are similar to those already mentioned in Section 5.2.2. The entries in the W column are the partial weight spectrums defined above.

By examining the tables, a number of observations can be made:

- The slope codes selected using the w0d50 weight spectrum consistently give the lowest SNR values, with almost all of the selected codes being in the top 25% of all codes. The only exception is for AWGN channel and a block length of 900. In this case, the selected code is still in the top half of all the codes.
- Using the w0d30 weight spectrum gives reasonably good results, as well, although not as good as the w0d50 case. Most of the selected codes are within the top half of all the codes, with the exception again being the AWGN channel and a block length of 900. For this particular case, the range of the SNR values is quite small, which probably accounts for the relatively poor results for the selected code.
- Using the w2d30 and w2d50 weight spectrums does not give particularly good results. Although in some cases, the selected codes do quite well compared to the group as a whole, this is not consistent across all the block lengths and channel conditions. Overall, the additional terms of the w2d50 weight spectrum do result in slightly better selections being made.
- For both types of distance spectrums, using the first 50 terms results in better selections being made than using the first 30 terms.
- The distance spectrum that incorporates all the available input weights leads to better selection decisions than the one that only incorporates the affect of weight-2 inputs. This is true even for the block length of 900, where the maximum input weight is 3, although in this case, the difference in the selected codes is not large.

The general conclusion that can be drawn is that noticeably better selection decisions are made when more weight spectrum data is available.

Table 5.10. *Design Selection SNR Comparison, PR, 192: (A) AWGN; (B) Fade;*

	W	Ilv #	min	25%	median	75%	max
			1.68	1.75	1.78	1.80	1.87
(A)	w0d30	073			1.76		
	w0d50	042		1.75			
	w2d30	047				1.83	
		178				1.82	
	w2d50	107			1.77		

	W	Ilv #	min	25%	median	75%	max
			2.97	3.06	3.09	3.12	3.19
(B)	w0d30	073		3.03			
	w0d50	042		3.05			
	w2d30	047			3.11		
		178				3.14	
	w2d50	107		3.00			

Table 5.11. *Design Selection SNR Comparison, PR, 400: (A) AWGN; (B) Fade;*

	W	Ilv #	min	25%	median	75%	max
			1.32	1.37	1.39	1.41	1.48
(A)	w0d30	009		1.37			
	w0d50	164		1.35			
	w2d30	112			1.38		
	w2d50	014				1.43	
		027			1.38		

	W	Ilv #	min	25%	median	75%	max
			2.55	2.60	2.62	2.64	2.71
(B)	w0d30	009		2.56			
	w0d50	164		2.56			
	w2d30	112		2.56			
	w2d50	014			2.62		
		027			2.61		

Table 5.12. *Design Selection SNR Comparison, PR, 900: (A) AWGN; (B) Fade;*

(A)

W	Ilv #	min	25%	median	75%	max
		9.94	1.06	1.07	1.09	1.11
w0d30	188			1.08		
w0d50	002		1.07			
w2d30	188			1.08		
w2d50	011			1.08		
	024		1.07			
	101				1.10	
	175	1.05				

(B)

W	Ilv #	min	25%	median	75%	max
		2.21	2.24	2.25	2.26	2.30
w0d30	188	2.24				
w0d50	002	2.22				
w2d30	188	2.24				
w2d50	011				2.27	
	024				2.27	
	101		2.25			
	175	2.24				

Chapter 6

Punctured Turbo-Codes

This chapter examines two approaches to puncturing turbo-codes, which will be called partially systematic turbo-codes (PSTC) and fully systematic turbo-codes (FSTC). In FSTCs only the parity bits are punctured, whereas in PSTCs both the data and parity bits are punctured. Both the BER and FER performance of the PSTC and FSTC codes are compared. The comparison looks at both the error floor and waterfall portions of the error rate curves. The BER and FER performance was obtained through simulation.

A number of papers in the area of punctured turbo-codes, have been presented in the literature, (e.g. [26, 19]), but these have focused almost exclusively on puncturing only the parity bits. If puncturing any data bits was considered at all, it was either immediately dismissed as giving bad performance for iterative decoding [26], or very few data bits were punctured [19]. In contrast, [27] considered puncturing half or more of the data bits, in order to reduce the error floor. This was further studied in [28].

The rate $1/3$ turbo-code given in Section 3.1 is punctured to give the rate $1/2$ code. For both the PSTC and FSTC, equal amounts of puncturing are applied to both constituent codes. Using the terminology of [27], the permeability rate p is defined as the ratio of unpunctured bits to total bits. The ratio of punctured data bits is p_u, and the ratio of punctured parity bits is p_{p_1} and p_{p_2} for the first and second constituent encoders, respectively. For the PSTC, we use $p_u = 1/2$ and $p_{p_1} = p_{p_2} = 3/4$. These choices were found in [27] to give good overall performance in both the waterfall and error floor regions. For the FSTC, by definition $p_u = 1$, and thus we have $p_{p_1} = p_{p_2} = 1/2$.

To minimize the influence of choice of interleaver on the performance results, a new random interleaver is generated for each packet. The performance results were obtained over 20 full iterations of the turbo-decoding algorithm using the Max-Log-MAP decoder.

The performance examination considers:

- various puncturing matrices,
- blocklengths of 286, 670 and 1054 information bits,
- performance over a number of iterations,
- performance over AWGN and Rayleigh fading channels,

6.1 Notation

The puncturing matrices will be represented as a binary puncturing mask and written as a hexadecimal number. Consider, for example, the following puncturing matrix for a $p_u = 1/2$ PSTC

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}. \tag{6.1}$$

Writing this as a puncture mask gives the tuple $(D6, 45)$, where the first number is the puncture mask for the first constituent code, and the second number is the puncture mask of the second constituent code.

To see how the puncture mask is derived from the puncture matrix, we first split the puncture matrix into two parts, one for each constituent code. For the first constituent code, this becomes

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}. \tag{6.2}$$

Reading down the columns from left to right gives the binary sequence $1, 1, 0, 1, 0, 1, 1, 0$, which, expressed as a hexadecimal number, is D6. For the second constituent code, the puncturing matrix is

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}. \tag{6.3}$$

The first row of all zeros indicates that all the information bits are punctured. Reading down the columns from left to right gives the binary sequence $0, 1, 0, 0, 0, 1, 0, 1$, which, expressed as a hexadecimal number, is 45.

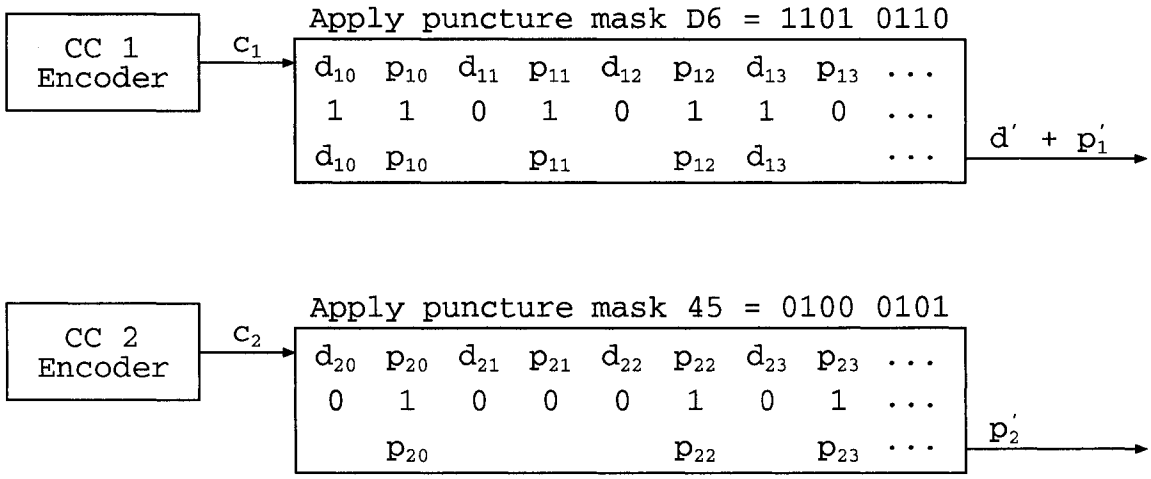

Figure 6.1. *Applying the puncture mask*

The length of a puncture mask is the length of the associated binary sequence. Thus, both of the above masks are of length 8.

The application of the puncture masks in the encoding operation is best illustrated with a diagram, as shown in Fig. 6.1. The outputs of the first (c_1) and second (c_2) constituent codes are input into the respective puncture masks. The masks are applied, and the outputs of the mask operation are the outputs of the turbo-encoder. Note that if we split the output of the first puncture mask, $d' + p'_1$, then this is equivalent to the data and parity branches of the standard turbo-encoder. Thus, we have not changed the structure of the code or the encoder, we have simply introduced a slightly different format.

6.2 Comparison of all Puncture Masks

All possible puncture masks of length 8 were generated for these two classes of codes. For the $p_u = 1/2$ code, there are 96 possible combinations of the puncture masks, whereas for the $p_u = 1$ code, there are 36 possible combinations of the puncture masks.

6.2.1 Comparing Statistics

The comparison is based on examining the minimum, maximum and mean of the BER and FER, in the error floor region, for each of the blocklengths considered. The results are

Table 6.1. *AWGN channel statistics for FSTC and PSTC. (A) BER, (B) FER*

length	SNR (dB)	min	max	mean
F286	4	3.12e-06	5.95e-06	4.30e-06
P286	4	1.25e-06	2.86e-06	1.91e-06
F670	3.5	2.44e-06	4.74e-06	3.49e-06
P670	3.5	7.50e-07	1.48e-06	1.04e-06
F1054	3	3.37e-06	6.78e-06	4.76e-06
P1054	3	8.35e-07	1.63e-06	1.25e-06

(A)

length	SNR (dB)	min	max	mean
F286	4	4.05e-04	6.81e-04	5.27e-04
P286	4	1.36e-04	2.83e-04	1.91e-04
F670	3.5	7.20e-04	1.33e-03	1.06e-03
P670	3.5	2.10e-04	4.09e-04	2.78e-04
F1054	3	1.65e-03	2.75e-03	2.24e-03
P1054	3	3.90e-04	7.55e-04	5.73e-04

(B)

summarized in Table 6.1 for the AWGN channel, and in Table 6.2 for the fading channel. The single letter "F" or "P" before the block length in the tables, indicates the results for the FSTC and PSTC, respectively. As can be seen, the error rate performance of the PSTC is better than that of the FSTC. In all cases, the worst PSTC is as good as or better than the best FSTC. Generally, as the block length is increased, the error rate performance benefit of the PSTC also increases. This is the case for both the BER and FER and for both channels, although it is more significant for the FER than the BER.

6.2.2 Comparing Histograms

To better illustrate the results, histogram plots for a blocklength of 286 are given in Figures 6.2, 6.3, 6.4 and 6.5 for the BER and FER in the error floor region, and for both channels. To provide a fair comparison, the histograms are normalized by the number of codes in each case. In [27], puncturing matrices are given for the two classes of codes, although no mention is made of how these matrices were determined. Expressed as masks, these puncturing matrices are $(D9, 15)$ and $(EE, 11)$, respectively for the PSTC and FSTC.

Table 6.2. *Fading channel statistics for FSTC and PSTC. (A) BER, (B) FER*

length	SNR (dB)	min	max	mean
F286	7	3.57e-06	5.92e-06	4.86e-06
P286	7	1.38e-06	2.91e-06	2.05e-06
F670	6	4.19e-06	6.82e-06	5.08e-06
P670	6	1.01e-06	2.39e-06	1.48e-06
F1054	5.5	3.89e-06	1.00e-05	5.23e-06
P1054	5.5	8.77e-07	3.74e-06	1.28e-06

(A)

length	SNR (dB)	min	max	mean
F286	7	4.64e-04	7.17e-04	6.06e-04
P286	7	1.49e-04	2.91e-04	2.14e-04
F670	6	1.23e-03	1.99e-03	1.52e-03
P670	6	2.83e-04	5.44e-04	4.15e-04
F1054	5.5	1.90e-03	3.41e-03	2.42e-03
P1054	5.5	4.19e-04	7.69e-04	5.80e-04

(B)

The results for these masks are also shown in the figures, along with the best puncture mask in each case.

From these figures, it is evident that the PSTCs with $p_u = 1/2$ perform much better than the FSTCs. For the BER, the histogram curves for the PSTCs and FSTCs touch, but do not overlap. However, for the FER, there is a noticeable gap between the histograms for the PSTCs and FSTCs. The results for the other blocklengths are similar. Also of note is that the puncturing matrices from [27] are equivalent to the best possible masks, although, in each case, the actual error rate is slightly higher.

The reason that the PSTCs have better FER performance than BER performance relative to FSTCs has to do with the convergence of the iterative decoding algorithm. In the error floor region, for an FSTC, each frame error causes a small number of bit errors. This indicates that the decoder has successfully converged to a codeword, although not the correct codeword, thus leading to a few bits errors. For a PSTC in the error floor region, some frame errors cause a large number of bit errors, with the remaining frame errors causing a small number of bits errors. This indicates that the decoder does not always successfully converge to a codeword. It is in these cases where the decoder does not converge, that a

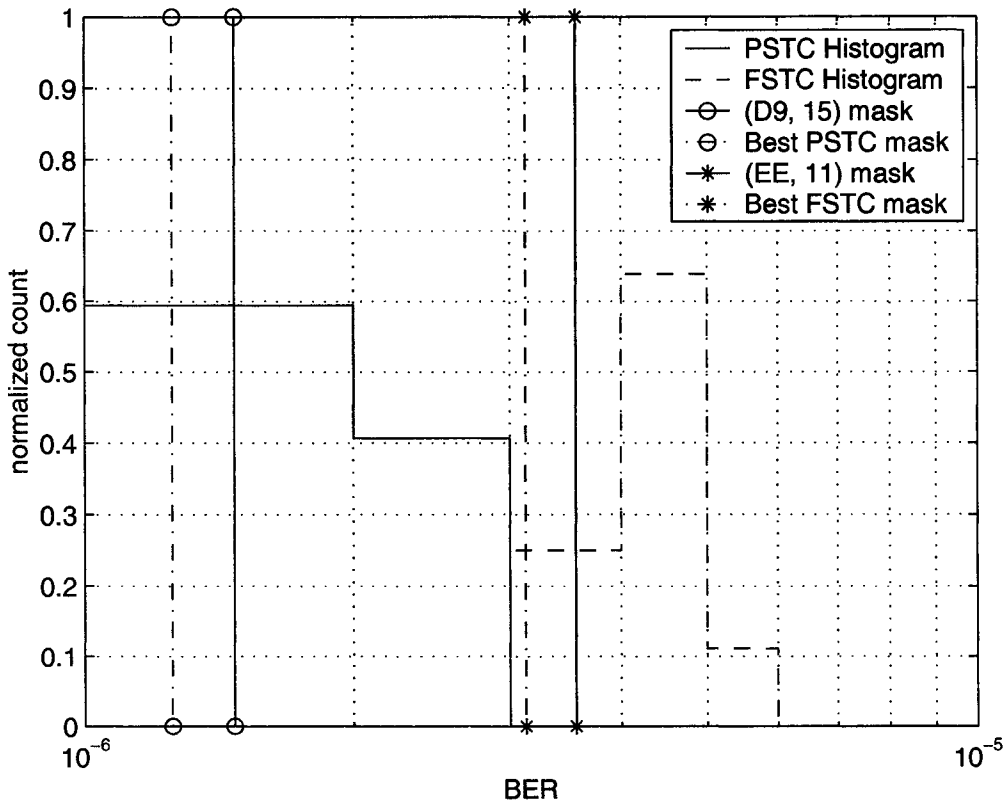

Figure 6.2. *BER Histogram for a Blocklength of 286 Bits, AWGN Channel, SNR=4dB*

large number of bits errors can result from a single frame error. This does not mean that the decoder will never converge to a codeword. Thus, in these cases, more decoding iterations may provide further improvements in the BER.

6.3 Comparison of the Best Puncture Masks

In this section, we compare in detail the BER and FER performance of the best puncture masks for the partially-systematic and fully-systematic turbo-codes. The best masks were determined by examining the BER performance over a range of SNR values, and finding those that give the best performance over this range. This range of SNR values is shown in Table 6.3, and was chosen to cover both the waterfall and error floor regions. The best performance was obtained by comparing the performance of a particular mask over the range of SNR values with that of the best mask at each SNR value. The distance between

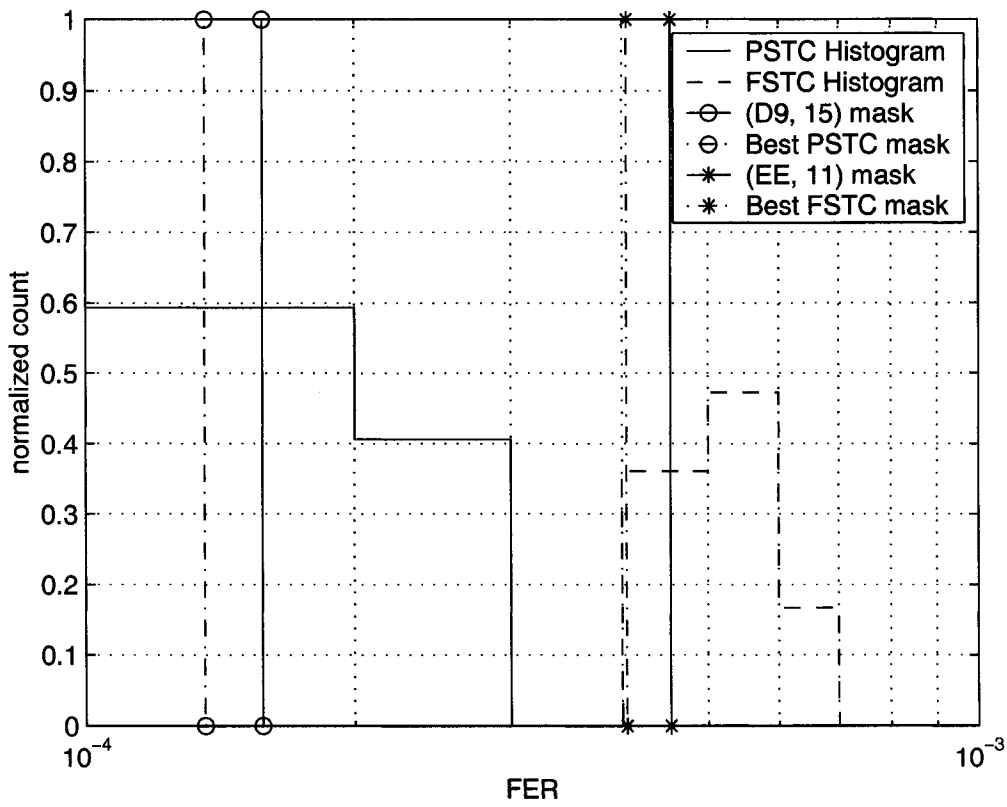

Figure 6.3. *FER Histogram for a blocklength of 286 Bits, AWGN Channel, SNR=4dB*

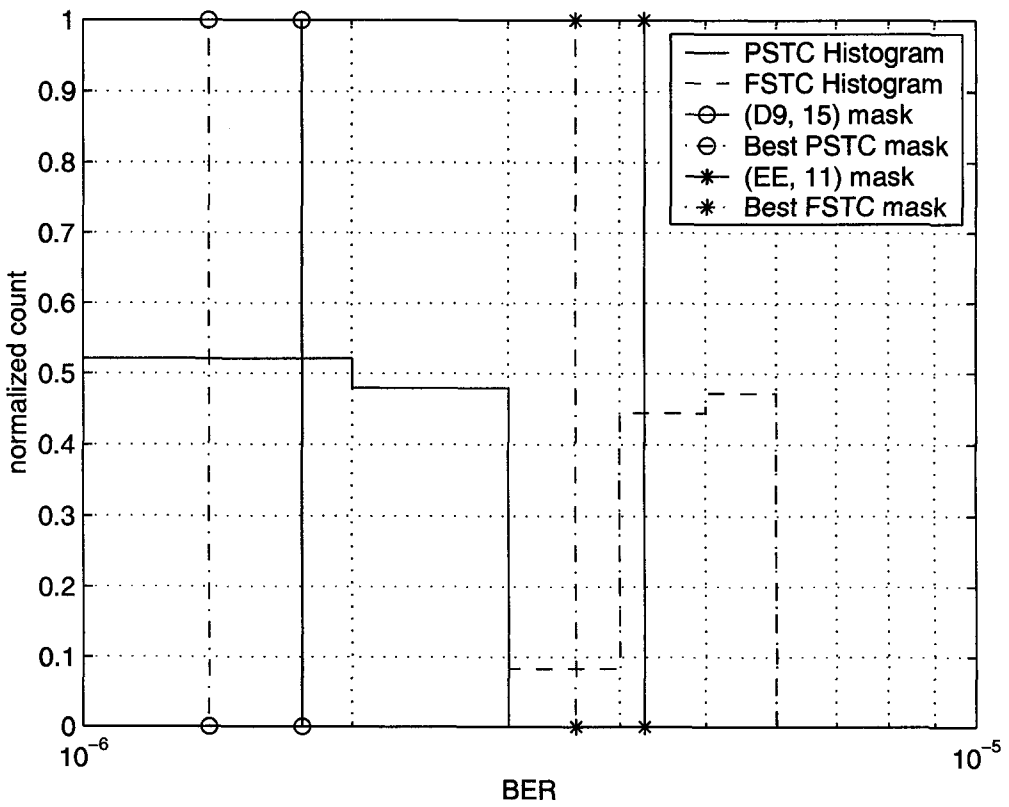

Figure 6.4. *BER Histogram for a Blocklength of 286 Bits, Rayleigh Fading Channel, SNR=7dB*

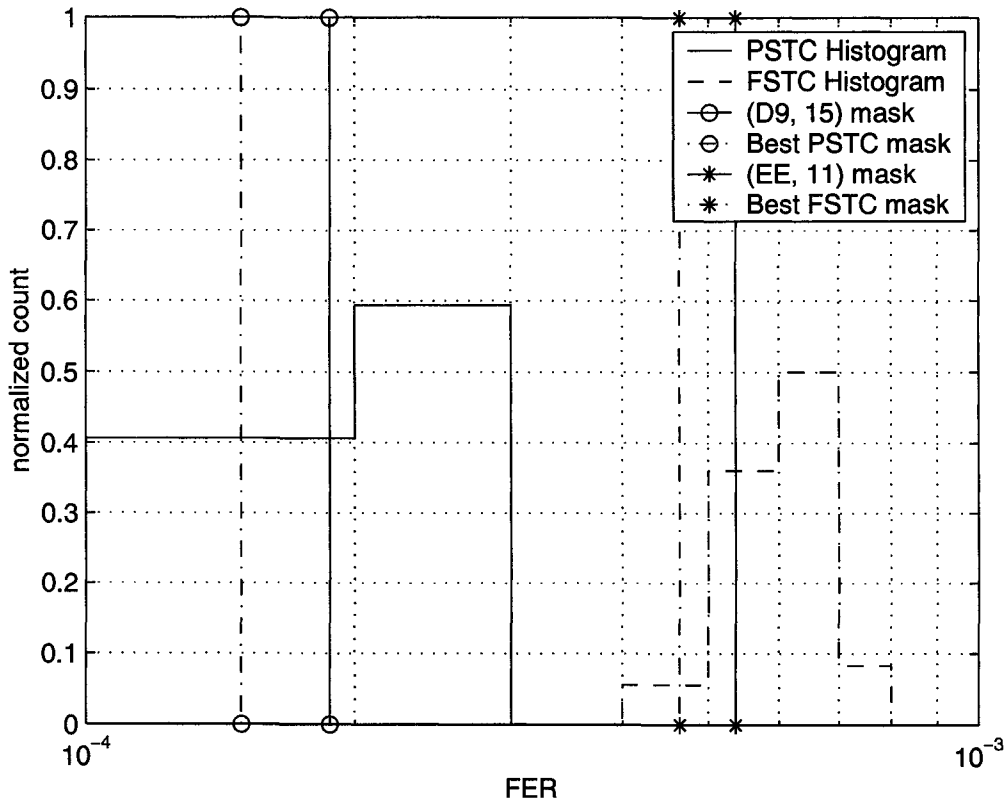

Figure 6.5. *FER Histogram for a Blocklength of 286 Bits, Rayleigh Fading Channel, SNR=7dB*

Table 6.3. *Best Puncture Masks over range of SNR values*

channel	length	SNR range (dB)	FSTC	PSTC
AWGN	286	2.0 - 4.0	EE, 11	D6, 45
	670	1.5 - 3.5	BB, 11	5B, 51
	1054	1.5 - 3.0	BB, 11	6D, 45
Fading	286	4.5 - 7.0	BB, 44	B5, 54
	670	4.0 - 6.0	BB, 44	5E, 54
	1054	3.5 - 5.5	BB, 11	7C, 54

Table 6.4. *Coding Gains for Best Puncture Masks*

		Coding Gain at	
channel	length	BER = 10^{-5}	FER = 10^{-3}
AWGN	286	0.5 dB	0.7 dB
	670	0.5 dB	0.6 dB
	1054	0.5 dB	0.7 dB
Fading	286	0.6 dB	1.0 dB
	670	0.5 dB	1.1 dB
	1054	0.4 dB	1.0 dB

the two sets of BERs is

$$distance = \sum_{s \in SNR\ Range} [\log_{10}(BER_s) - \log_{10}(BestBER_s)]^2 \tag{6.4}$$

where BER_s is the BER of the mask at the given SNR value, and $BestBER_s$ is the best BER over all the masks at the given SNR value. The mask with the smallest *distance* was taken to be the best mask.

The best masks for the three blocklengths investigated are given in Table 6.3. The associated coding gains of the PSTC over the corresponding FSTC are given in Table 6.4. When comparing two error rate curves, the coding gain is defined as the difference in SNR needed to achieve a given value of the error rate. For a particular channel and error rate, the coding gain appears to be relatively independent of the blocklength. Also, in general, the FER improvement is better than the BER improvement.

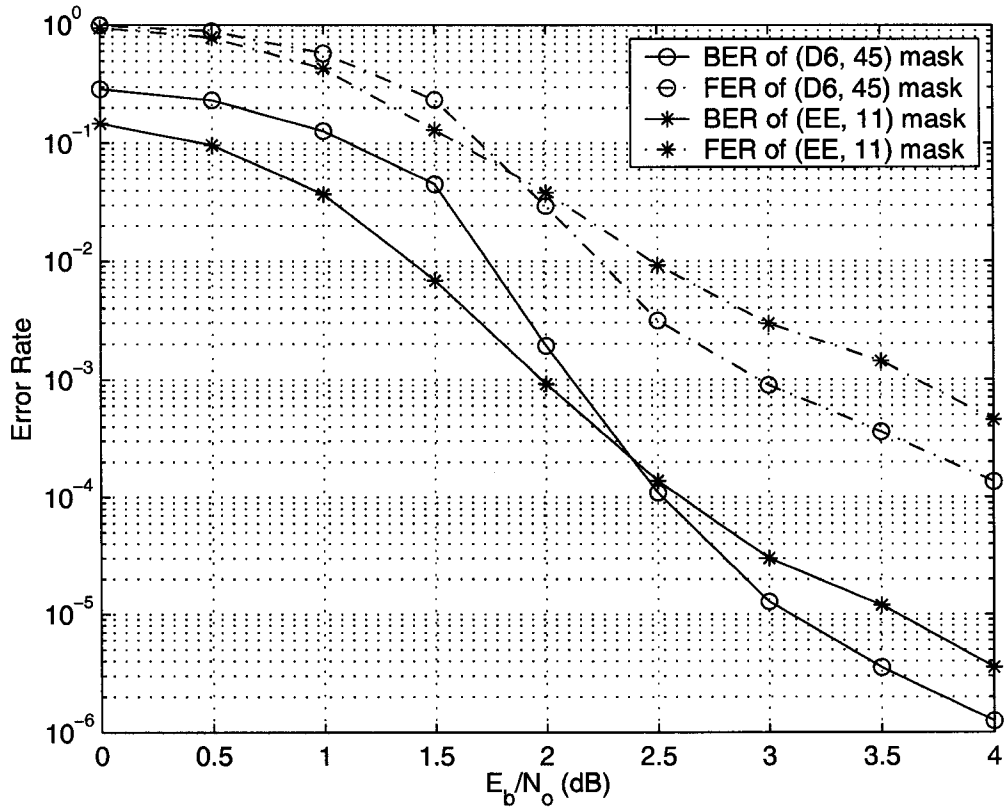

Figure 6.6. *BER and FER for a Blocklength of 286 Bits, AWGN Channel*

As in the previous section, we focus on the results for a blocklength of 286 bits. The BER and FER performance for the best puncture masks is given in Figs. 6.6 and 6.7, for the AWGN and fading channels, respectively. The results for other blocklengths are similar. Of particular interest is that the point when the PSTC starts to perform better than the FSTC occurs at approximately the same error rate for all the blocklengths. For the AWGN channel, the intersection points are for a BER of approximately 1.5e-04, and a FER of approximately 5.5e-02. For the fading channel, the intersection points are for a BER of approximately 3.0e-05, and a FER of approximately 1.0e-02.

6.3.1 Comparing Contour Plots

It is also useful to compare the codes over a range of SNR values and for a number of iterations. To best visualize these comparisons, the difference in performance between the

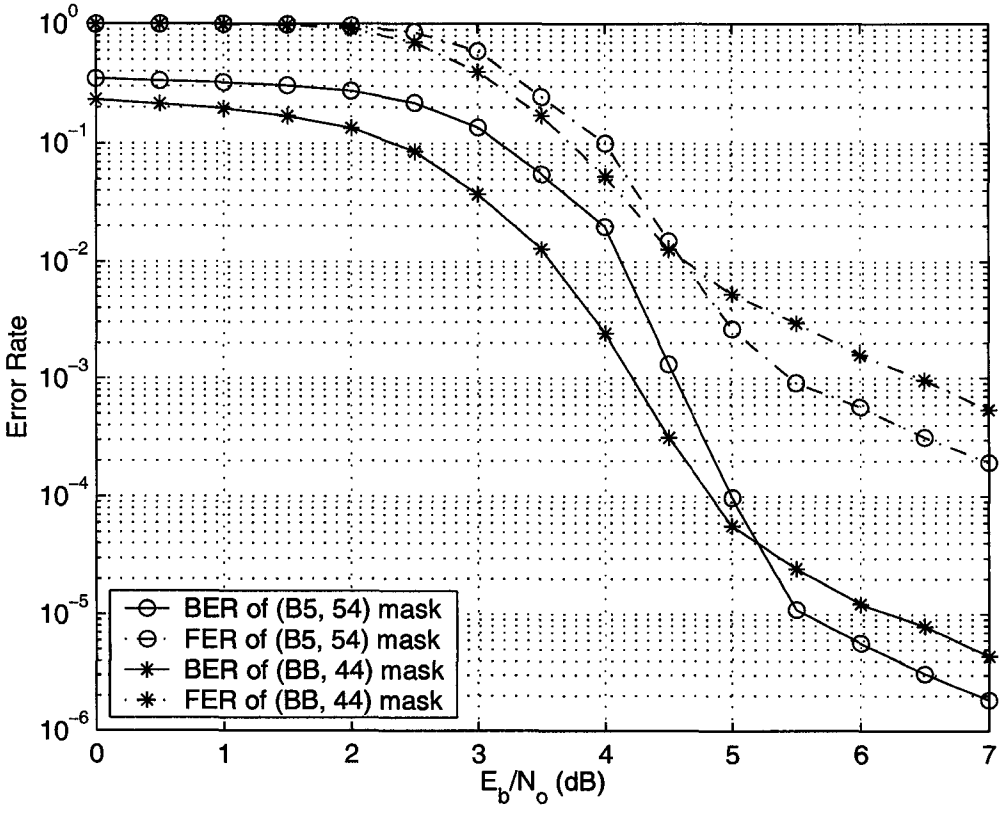

Figure 6.7. *BER and FER for a Blocklength of 286 Bits, Fading Channel*

two codes being compared was determined for each (SNR, iteration) point and plotted on a contour graph.

The BER and FER contour plots for a blocklength of 286 bits are given in Figs. 6.8 to 6.11, for the AWGN and Rayleigh fading channels. The vertical axis is the number of half iterations, and the horizontal axis is the SNR. The contour values are given by

$$\log_{10}(BER_{FSTC}) - \log_{10}(BER_{PSTC}). \tag{6.5}$$

For example, if $BER_{FSTC} = 10^{-5}$ and $BER_{PSTC} = 10^{-6}$, then the contour would show 1.0.

The most important contour to note is the one labelled 0. This is where the PSTC starts to perform better than the FSTC. For the points to the left of the 0 contour, the PSTC performs worse than the FSTC. For the points to the right of the 0 contour, the PSTC performs better than the FSTC.

From the figures, a number of observations can be made:

- The performance benefits of the PSTCs are mainly for higher SNR values and number of iterations.
- Even at the highest SNR value, the PSTCs do not perform better than the FSTCs until at least 8-10 half iterations. For lower numbers of iterations, the PSTCs perform worse, and in some cases significantly so.
- The better the PSTCs perform at higher iterations, the worse they perform at lower iterations.

All of these observations can be attributed to the slower convergence rate of the PSTC relative to the FSTC. The PSTC needs a higher SNR and/or number of iterations for convergence to start. Even at the highest SNR values, the PSTC needs an extra one or two full iterations to converge.

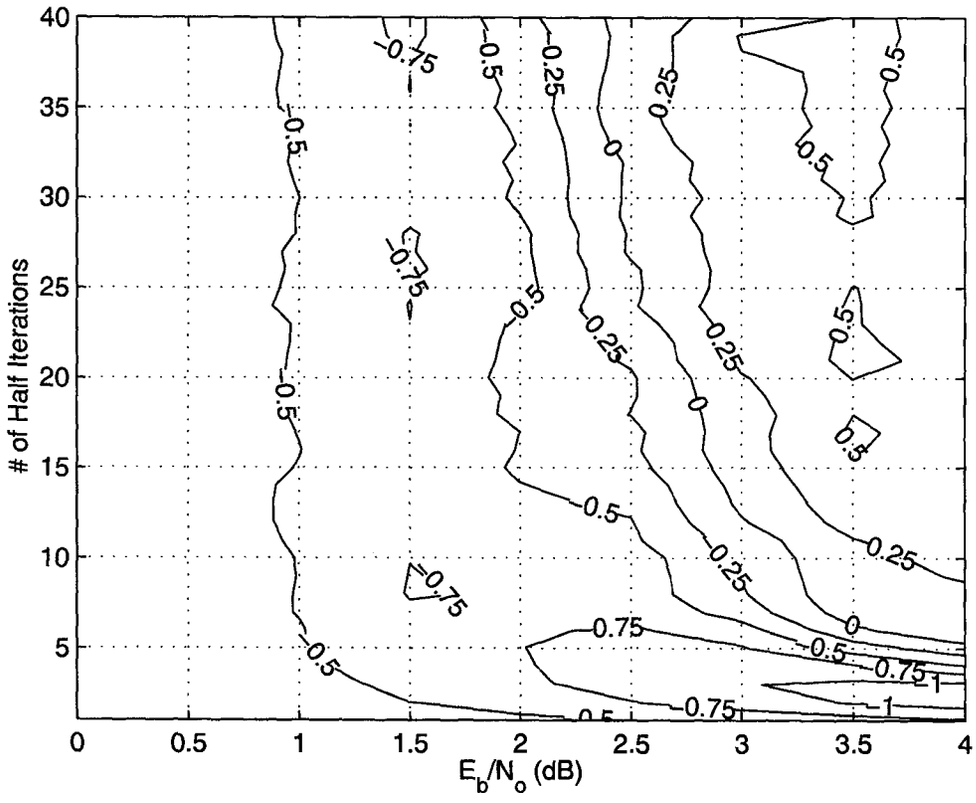

Figure 6.8. *BER Ratio Contour for a Blocklength of 286 Bits, AWGN*

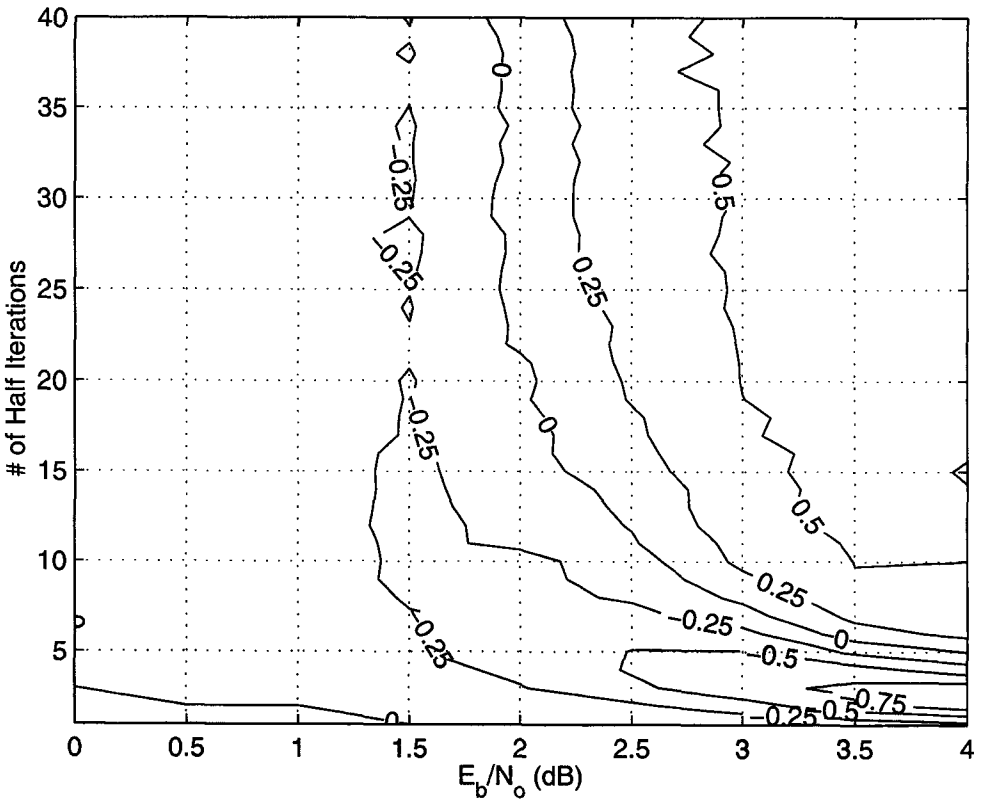

Figure 6.9. *FER Ratio Contour for a Blocklength of 286 Bits, AWGN*

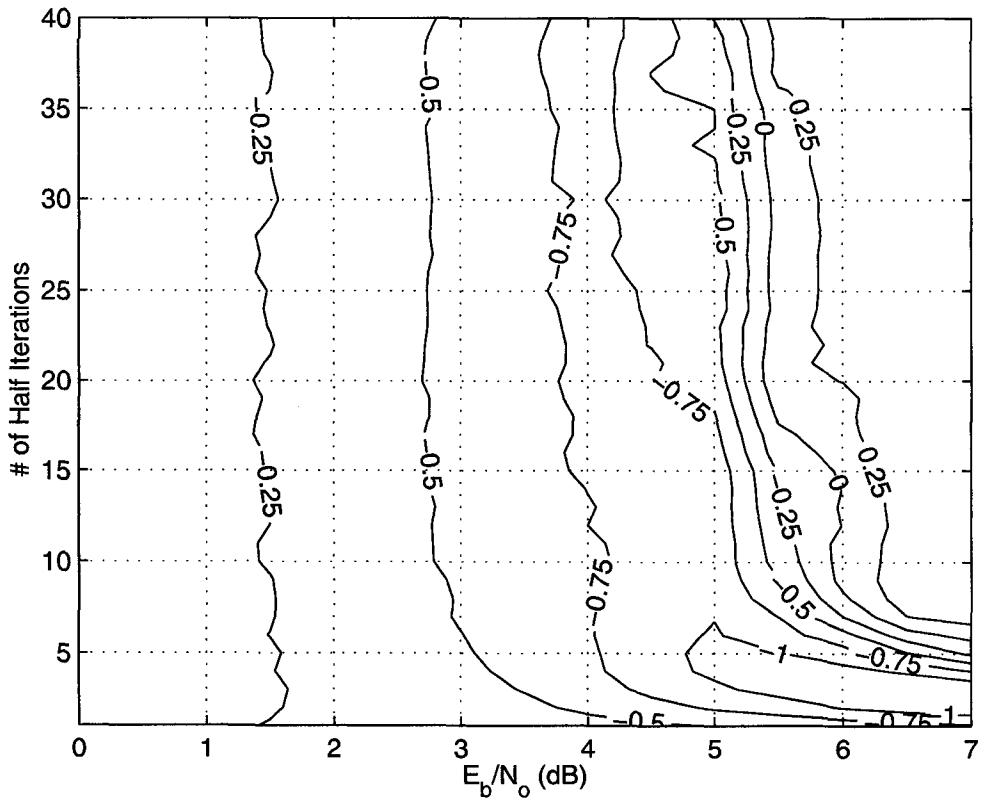

Figure 6.10. *BER Ratio Contour for a Blocklength of 286 Bits, Rayleigh Fading*

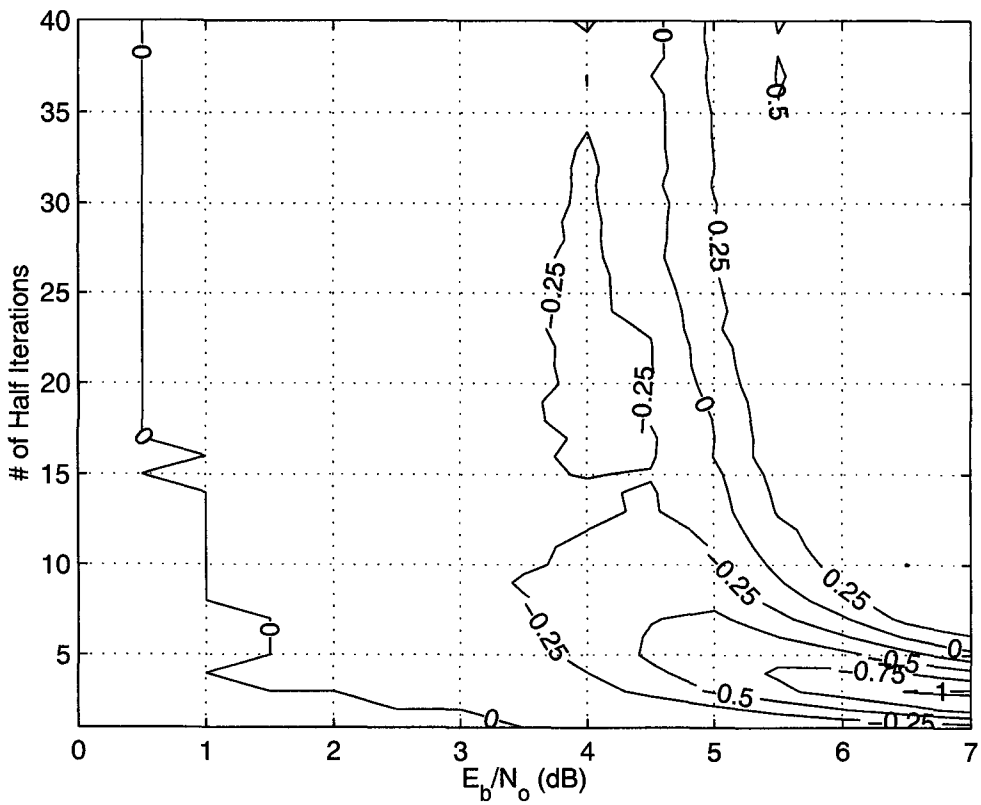

Figure 6.11. *FER Ratio Contour for a Blocklength of 286 Bits, Rayleigh Fading*

Chapter 7

Partially Systematic Turbo-Codes

This chapter examines in more detail the simulation results for the PSTC from Chapter 6. Recall that $p_u = 1/2$, $p_{p_1} = p_{p_2} = 3/4$, and puncture masks of length 8 were considered. All possible puncture masks were generated for this code. There are

$$\binom{4}{2} \times \binom{4}{3} = 24 \tag{7.1}$$

combinations of possible puncture masks for the first constituent code, and

$$\binom{4}{3} = 4 \tag{7.2}$$

combinations of possible puncture masks for the second constituent code, giving a total of 96 possible combinations.

The simulation performance results were obtained over 20 full iterations of the turbo-decoding algorithm using the Max-Log-MAP decoder. Information blocklengths of 286, 670 and 1054 data bits were used on both the AWGN and fading channels. To minimize the influence of choice of interleaver on the performance results, a new random interleaver is generated for each packet.

The performance of these codes is examined in both the waterfall and error floor regions for both the BER and FER. These regions were defined in Chapter 3 for the BER. For the FER, the waterfall region will be defined as the region where the FER is between 10^{-2} and 10^{-3}, while the error floor will be defined by the highest SNR value used in the simulation. The SNR values corresponding to the waterfall and error floor regions are given in Table 7.1. These ranges were selected so that they contain the error values of interest.

By examining the performance of the resulting codes, we can devise some simple design guidelines. These guidelines allow us to identify those puncture masks that have the potential to give good performance, and thus should be the subject of further analysis.

Table 7.1. *SNR Ranges/Values*

channel	length	waterfall SNR range (dB)	error floor SNR value (dB)
AWGN	286	2.0 - 3.0	4.0
	670	1.5 - 2.5	3.5
	1054	1.5 - 2.5	3.0
Fading	286	4.5 - 6.0	7.0
	670	4.0 - 5.0	6.0
	1054	3.5 - 5.0	5.5

7.1 CC1 Family Data

In this section, we examine the relative performance of each of the families of masks. This is done by taking the top and bottom 25% of the masks under given conditions, and counting how many of these masks belong to each family.

For the first constituent code, the 24 puncture masks are divided into 6 families, as shown in Table 7.2. Each family is labelled using the notation "f:XX", where XX is a member of the family. The value of XX was arbitrarily chosen to be the lowest numbered member of the given family. For the second constituent code, the 4 puncture masks all belong to the same family.

The members of a given puncture mask family are determined by taking a single puncture mask, and cyclically shifting through all possible values. Note that the cyclic shift must be done two positions at a time, in order to align the data and parity bits in the mask. For example, applying all possible cyclic shifts to 5B gives

$$\begin{aligned} 0101 \quad 1011 &= 5B \\ 1101 \quad 0110 &= D6 \\ 1011 \quad 0101 &= B5 \\ 0110 \quad 1101 &= 6D. \end{aligned}$$

All of the 96 masks were partitioned according to the CC1 families. Thus, we have $96/6 = 16$ puncture masks per CC1 family for comparison. Since we are looking at the top and bottom 25% of the puncture masks, if the performance of a given family is evenly

Table 7.2. *CC1 Families*

label	masks
f:1F	1F 7C C7 F1
f:37	37 73 CD DC
f:3D	3D 4F D3 F4
f:5B	5B 6D B5 D6
f:5E	5E 79 97 E5
f:67	67 76 9D D9

distributed, we would expect to see about 4 masks in the top 25% and 4 masks in the bottom 25% for each family.

For the first set of comparisons, we look at each of the families at each SNR point that was simulated. These results are shown in Table 7.3 for a blocklength of 286 and the AWGN channel, and in Table 7.4 for a blocklength of 1054, and the fading channel. The maximum value for each entry in the tables is 16, and the expected value is 4. Any table entry with a number significantly above the expected value indicates a superior mask family, for the given conditions, and any number significantly below indicates an inferior mask family.

From the tables, a number of observations can be made:

- The performance differences of the mask families tends to be more pronounced for the BER than the FER.
- The f:37 family is consistently the worst, rarely appearing in the top 25%, and appearing the most in the bottom 25%.
- The f:1F and f:3D families perform well for low SNRs, but don't do as well for the waterfall and error floor regions.
- The f:67 family generally performs the best in the error floor region at high SNR, but tends to perform very badly for the low SNR and waterfall regions.
- The f:5B and f:5E families are generally the best in the waterfall region, although they do reasonably well in the other regions as well, since they rarely appear in the bottom 25%. The exception to this is the FER in Table 7.4.
- The FER in Table 7.4 exhibits an interesting characteristic for low SNRs. Specifi-

Table 7.3. *P286 AWGN (A) BER Top 25% (B) BER Bottom 25% (C) FER Top 25% (D) FER Bottom 25%*

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
f:1F	9	6	4	5	2	4	2	1	0	
f:37	0	0	0	0	0	0	1	1	0	
f:3D	14	12	4	3	7	0	1	0	1	(A)
f:5B	1	3	8	8	9	10	7	7	9	
f:5E	0	3	8	8	6	10	10	5	4	
f:67	0	0	0	0	0	0	3	10	10	

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
f:1F	0	0	0	0	4	1	5	8	9	
f:37	10	11	9	11	8	8	7	6	7	
f:3D	0	0	0	0	1	5	2	8	5	(B)
f:5B	0	0	0	0	0	1	1	0	1	
f:5E	0	0	0	0	1	0	1	1	2	
f:67	14	13	15	13	10	9	8	1	0	

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
f:1F	3	4	1	2	4	3	1	1	0	
f:37	4	2	0	0	0	3	3	0	0	
f:3D	3	3	3	3	3	5	0	1	1	(C)
f:5B	7	6	10	10	9	4	5	6	8	
f:5E	3	7	10	9	8	7	5	4	1	
f:67	4	2	0	0	0	2	10	12	14	

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
f:1F	4	6	3	3	4	2	9	8	9	
f:37	7	9	6	10	7	6	4	4	6	
f:3D	3	1	2	1	3	7	7	8	7	(D)
f:5B	3	1	1	0	0	3	1	1	1	
f:5E	5	1	1	0	0	0	3	3	1	
f:67	2	6	11	10	10	6	0	0	0	

Table 7.4. *P1054 Fading (A) BER Top 25% (B) BER Bottom 25% (C) FER Top 25% (D) FER Bottom 25%*

(A)

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5
f:1F	12	14	11	12	9	12	9	5	8	6	1	0
f:37	0	0	0	0	0	0	0	0	0	1	1	2
f:3D	12	10	13	12	15	12	9	6	5	4	0	1
f:5B	0	0	0	0	0	0	3	7	5	8	4	5
f:5E	0	0	0	0	0	0	3	6	6	5	6	5
f:67	0	0	0	0	0	0	0	0	0	0	12	11

(B)

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5
f:1F	0	0	0	0	0	0	0	0	0	0	5	6
f:37	9	8	8	8	8	8	8	8	6	7	6	4
f:3D	0	0	0	0	0	0	0	0	1	1	9	9
f:5B	0	0	0	0	0	0	0	0	1	1	3	2
f:5E	0	0	0	0	0	0	0	0	1	0	1	3
f:67	15	16	16	16	16	16	16	16	15	15	0	0

(C)

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5
f:1F	4	4	4	4	4	6	2	5	5	2	0	1
f:37	4	4	4	4	4	2	0	0	2	2	1	4
f:3D	4	4	4	4	4	4	5	3	5	3	0	0
f:5B	4	4	4	4	4	7	9	8	7	6	7	4
f:5E	4	4	4	4	4	4	8	8	5	5	5	5
f:67	4	4	4	4	4	1	0	0	0	6	11	10

(D)

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5
f:1F	4	4	4	4	4	1	0	0	2	2	6	7
f:37	4	4	4	4	4	6	11	9	6	5	5	5
f:3D	8	8	8	8	8	5	0	0	1	6	9	7
f:5B	4	4	4	4	4	3	0	0	1	2	1	1
f:5E	4	4	4	4	4	1	0	0	1	1	2	4
f:67	0	0	0	0	0	8	13	15	13	8	1	0

cally, all of the mask families appear to be equally likely since the count is usually 4 for both the top and bottom 25%.

Similar observations can be made for the other blocksize and channel combinations.

7.2 CC1 Individual Data

As we saw in the previous section, certain families of puncture masks tend to give better performance, with the f:5B and f:5E families doing best in the waterfall region, and the f:67 family doing best in the error floor region. As such, this section will concentrate on these families, and will look at the performance in the waterfall region for the f:5B and f:5E family masks, and in the error floor region for the f:67 family masks.

7.2.1 Waterfall Region

For the waterfall region, we linearly interpolated the simulation results, to determine the SNR required to obtain the desired BER and FER. Specifically, we determined the required SNR values to obtain bit error rates of 10^{-3}, 10^{-4} and 10^{-5} and frame error rates of 10^{-2} and 10^{-3}. As in the previous section, the relative performance is determined by taking the top and bottom 25% of the masks, and counting the number of times each mask appears.

In order to determine the top and bottom 25% of the masks, all of the masks were ordered according to their performance over the entire range of BERs or FERs. Using an approach similar to the one used in Section 6.3, the distance between a particular mask and the best mask at the given error rate value, was calculated over the range of error rate values. For the BER, the distance between the two sets of SNRs is

$$distance = \sum_{b \in BER\ Range} [SNR_b - BestSNR_b]^2 \tag{7.3}$$

where SNR_b is the SNR of the mask at the given BER value, and $BestSNR_b$ is the best SNR over all the masks at the given BER value. The distance between the two sets of SNRs for the FER is calculated in a similar manner. All 96 masks were ordered according to their *distance* value with the smallest distance values being the best, and the largest distance values being the worst.

Although all the masks were ordered, only the counts for the f:5B and f:5E mask families are presented in Table 7.5, which shows how many times the given masks appear in the top 25% of all the masks, for the BER and FER waterfall regions.

The six columns labelled *286A* to *1054F* indicate the counts for the given conditions. The numeric part of the label indicates the blocklength, and the letter indicates the channel, with A=AWGN and F=fading. The sum of all the individual masks (or equivalently, the sum of the mask families) in any one column has a maximum value of 24, which is the number of masks that are in the top 25% of all masks (i.e. 25% of 96 = 24). For each row labelled by an individual CC1 mask, the maximum value of a table entry is 4, and the expected value is 1. For each row labelled by a CC1 mask family, the maximum value of a table entry is 16, and the expected value is 4. The *Total* column shows the sum of all the entries in the given row. The *Total* entry for an individual CC1 mask has a maximum value of 24, and an expected value of 6. The *Total* entry for a CC1 mask family has a maximum value of 96, and an expected value of 24.

As before, any number significantly above the expected value indicates a superior mask/family, and any number significantly below indicates an inferior mask/family.

Some observations can be made:

- The majority of best masks in the waterfall region belong to the f:5B and f:5E families. This can be clearly seen by combining the entries for the f:5B and f:5E families for any given blocklength and channel combination. The counts of the masks ranges from a low of 11 to a high of 22 out of the 24 masks in the top 25%. This observation was already suggested in the previous section, but is more noticeable here, especially for the bit error rate.
- All masks contribute to the performance of the corresponding mask family, although certain masks tend to stand out, such as the 5B, 5E and B5 masks. For both the BER and FER, the values in the *Total* column for each of these individual masks is at least half the maximum value of 24 for that table entry. This means that for each of these masks, across all conditions, they are in the top 25% of all masks, at least 50% of the time.
- Overall, the 5B mask tends to perform best across all conditions, and for both the bit and frame error rates.

Table 7.5. *CC1 Waterfall Region (A) BER Top 25% (B) FER Top 25%*

	286A	286F	670A	670F	1054A	1054F	Total	
f:5B	10	13	8	8	12	7	58	
5B	4	3	3	3	4	3	20	
6D	0	3	2	2	4	1	12	
B5	2	4	1	0	3	2	12	
D6	4	3	2	3	1	1	14	(A)
f:5E	12	7	9	7	10	4	49	
5E	4	1	4	3	4	0	16	
79	2	2	1	3	4	1	13	
97	4	1	4	0	1	1	11	
E5	2	3	0	1	1	2	9	

	286A	286F	670A	670F	1054A	1054F	Total	
f:5B	9	10	8	8	7	9	51	
5B	3	2	2	3	1	3	14	
6D	1	4	3	0	1	1	10	
B5	2	4	1	1	4	4	16	
D6	3	0	2	4	1	1	11	(B)
f:5E	8	6	7	6	4	8	39	
5E	4	2	4	4	0	3	17	
79	1	1	2	1	2	2	9	
97	1	3	1	1	1	2	9	
E5	2	0	0	0	1	1	4	

Table 7.6. *CC1 Error Floor Region (A) BER Top 25% (B) FER Top 25%*

	286A	286F	670A	670F	1054A	1054F	Total	
f:67	10	13	11	13	11	11	69	
67	2	4	3	1	4	3	17	(A)
76	1	2	1	4	2	3	13	
9D	4	4	3	4	3	4	22	
D9	3	3	4	4	2	1	17	

	286A	286F	670A	670F	1054A	1054F	Total	
f:67	14	16	12	13	13	10	78	
67	4	4	3	1	4	3	19	(B)
76	2	4	2	4	2	3	17	
9D	4	4	3	4	4	4	23	
D9	4	4	4	4	3	0	19	

7.2.2 Error Floor Region

For the error floor region, we examine the error rate performance for the highest SNR value simulated. The mask counts for the error floor region of the f:67 family are given in Table 7.6. The table has the same format and layout as Table 7.5 in Section 7.2.1.

Some observations can be made:

- All masks contribute to the performance of the mask family, although the contribution is not evenly distributed.
- Overall, the 9D mask tends to perform the best across all conditions, and for both bit and frame error rates. In fact, the individual mask counts in the *Total* column are 22 and 23 for the BER and FER, respectively, out of a maximum count of 24. This indicates that almost all of the 9D masks are in the top 25% across all interleaver and channel conditions.
- The 76 mask tends to perform the worst across all conditions, and for both the bit and frame error rates, although even here, the 76 mask will be in the top 25% of all masks at least 50% of the time.
- The f:67 masks tend to have a more significant presence for the FER than for the BER. This is in contrast to the waterfall region, where the predominant mask families

Table 7.7. *CC2 P286 AWGN (A) BER Top 25% (B) BER Bottom 25%*

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
15	5	4	1	2	3	8	6	8	8	
45	7	5	4	6	6	4	5	7	6	(A)
51	5	8	9	10	6	5	4	6	7	
54	7	7	10	6	9	7	9	3	3	

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
15	7	7	8	7	7	6	5	7	5	
45	7	6	7	8	4	6	7	6	5	(B)
51	5	5	4	3	6	5	7	8	6	
54	5	6	5	6	7	7	5	3	8	

tend to have a more significant presence in the BER. We suspect that this is simply due to the particular definitions of these regions for given conditions.

7.3 CC2 Data

For the second constituent code (CC2), all four masks belong to the same family. Thus, if the 96 total masks are partitioned according to the CC2 mask, there are $96/4 = 24$ puncture masks per CC2 mask for comparison. As in the previous section, we will look at the top and bottom 25% of all the masks, and count the number of times each CC2 mask appears. If the masks are equally distributed across the performance range, we would expect each CC2 mask to appear about 6 times in the top 25% and about 6 times in the bottom 25%.

Some sample results are shown in Table 7.7 for a blocklength of 286 and the AWGN channel, and in Table 7.8 for a blocklength of 1054, and the fading channel. These are the same conditions as presented in Tables 7.3 and 7.4 in Section 7.1. What we find is that all four masks appear with relatively equal frequency in both the top and bottom 25%. The tables suggest that the 15 CC2 mask tends to perform slightly better in the error floor region and the 54 CC2 mask tends to perform slightly worse, but other results, especially for a blocklength of 670, show the opposite. For the waterfall region, there is no clear distinction between any of the CC2 puncturing masks, as they all tend to appear with approximately

Table 7.8. *CC2 P1054 Fading (A) BER Top 25% (B) BER Bottom 25%*

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	
15	7	5	8	4	5	9	11	
45	6	6	4	9	7	6	6	(A)
51	6	7	6	4	5	3	2	
54	5	6	6	7	7	6	5	

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	
15	6	6	5	5	5	7	3	
45	7	7	7	5	7	4	7	(B)
51	5	6	6	7	9	5	8	
54	6	5	6	7	3	8	6	

equal frequency.

To see if a larger sample size would show anything different, we also repeated the above but took the top 50% of the masks. In this case, the expected value of the count is 12, assuming uniform distribution. Again, there are no significant and consistent differences between the counts for any of the CC2 puncture masks. For example, the results for a blocklength of 670 on the AWGN channel, are shown in Table 7.9. This highlights the point mentioned above, that, for a blocklength of 670, the 54 CC2 puncture mask tends to perform better in the error floor region, whereas the 15 CC2 mask performs worse in the same region. However, this observation does not hold for other blocklength/channel combinations.

One possible explanation for the similar performance across all the masks has to do with the choice of interleaver. Since the interleaver is applied to the information bits before being passed to the CC2 encoder, the interaction between the interleaver and the puncturing mask has a direct impact on the weight spectrum of the encoder output. However, in this chapter we are considering a randomly generated interleaver for each packet, and thus the contribution of a particular puncturing mask is averaged out over all the interleavers. This does not occur for the CC1 encoder, since the interleaver does not affect the weight spectrum of the output of the CC1 encoder.

Table 7.9. *CC2 P670 AWGN (A) BER Top 50% (B) FER Top 50%*

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	
15	14	11	12	11	11	14	6	6	
45	11	11	8	16	12	8	15	10	(A)
51	10	13	13	11	15	13	16	14	
54	13	13	15	10	10	13	11	18	

	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	
15	12	16	10	12	13	12	5	8	
45	12	15	10	13	12	12	15	10	(B)
51	12	8	15	13	13	12	15	10	
54	12	9	13	10	10	12	13	20	

Table 7.10. *Binary representation of puncture masks*

mask	binary	info bits only	number of runs
1F	0001 1111	00 11	2
37	0011 0111	01 01	4
3D	0011 1101	01 10	4
5B	0101 1011	00 11	6
5E	0101 1110	00 11	4
67	0110 0111	01 01	4

7.4 Discussion

In this section, we discuss the relative performance of the different puncture masks, and their associated families.

By examining the actual bit patterns of these masks, we may be able to gain some insight. Table 7.10 gives the binary representation for the first mask in each family, along with the number of runs. A run is defined as a sequence of contiguous bits of the same value.

There is an obvious difference between the three families that perform poorly in the waterfall and error floor regions (i.e. f:1F, f:37, f:3D) and the three families that perform well in these regions (i.e. f:5B, f:5E and f:67). The poorly performing masks all start with

00. Recall that each group of two digits represent a single stage in the trellis. Thus, a 00 indicates that both the information and parity bits for the given stage are punctured. The conclusion that can be drawn from this is that for any given stage in the trellis, either the information bit or the parity bit can be punctured, but never both.

Further examining the masks that perform well, a number of observations can be made:

- The number of runs is highest for f:5B, but is the same for f:5E and f:67.
- Both f:5B and f:5E have runs of 1's of length 1, whereas the shortest runs of 1's for f:67 is 2.
- Considering only the puncturing of the information bits, f:67 punctures every second information bit, whereas f:5B and f:5E puncture two consecutive information bits.

From these observations, it appears that puncturing alternate information bits gives better performance in the error floor region, while maximizing the number of runs of length one in the puncturing mask gives better performance in the waterfall region. Since the output of the CC1 encoder is not affected by the choice of interleavers, it would seem reasonable that these observations can be applied to specific interleavers as well.

Chapter 8

Partially Systematic Turbo-Codes with Select Interleavers

This chapter combines the results of previous chapters by applying code design methods to the selection of interleavers and puncturing masks. The selected codes are then simulated and compared. The PSTC code parameters are the same as considered in the previous chapter, namely, $p_u = 1/2$ and $p_{p_1} = p_{p_2} = 3/4$, although only a blocklength of 286 will be considered. In Chapter 6 and Chapter 7, the results were shown to be relatively independent of the blocklength. Thus, for the purposes of examining the puncturing masks, it is sufficient to examine a single blocklength.

8.1 Design Procedure

The design procedure that was used is detailed in this section. Both the minimum distance and distance spectrum slope criteria were used in the selection process.

The following major steps were applied. Each step is described in more detail later.

1. A number of VSR and MSR interleavers were generated, and the $w = 1$ and $w = 2$ weight spectrums were determined. For each interleaver type, the interleavers with the highest d_{min} values were selected.
2. For the selected interleavers, the $w = 3$ and $w = 4$ weight spectrums were determined. Note that the $w = 1$ and $w = 2$ weight spectrums for these interleavers were already determined in the previous step. For each interleaver type, the interleavers with the highest d_{min} values were selected.
3. For the selected interleavers, the distance spectrum slope was calculated. For each

interleaver type, 5 interleavers with the best (i.e. lowest) slope were selected.

4. The selected interleavers were combined with all the puncture masks, and the weight spectrum was determined for $w = 1, 2, 3, 4$. From the weight spectrums, the distance spectrum slope was calculated. Two codes were selected: one that best satisfies the minimum distance criteria, and one that best satisfies the distance spectrum slope criteria.

8.1.1 Step #1

To determine whether the VSR interleavers are indeed better than the MSR interleavers, both types of interleavers were generated.

A total of 1600 VSR and 1600 MSR interleavers were generated. The S-values for both interleavers were varied over a range of values and for each particular value, 200 interleavers were generated. For the VSR interleaver, the S-value was varied from 9 to 16. For the MSR interleaver, the S-value was varied from 12 to 19.

The $w = 1, 2$ weight spectrum was calculated for all the generated interleavers. For the VSR interleaver, the highest d_{min} was 20, and there were 139 interleavers with this value. For the MSR interleaver, the highest d_{min} was 18, and there were 331 interleavers with this value.

Although the VSR interleavers are better than the MSR interleavers from the minimum distance criteria, both interleaver types were carried forward to the next step.

8.1.2 Step #2

For the 139 VSR and 331 MSR interleavers selected in the previous step, the $w = 3$ and $w = 4$ weight spectrums were determined. The values of d_{min} for the selected interleavers were recalculated based on the additional weight spectrum data.

For the VSR interleaver there were 114 interleavers with a d_{min} of 20, which means that 25 interleavers or approximately 20% had minimum distances lower than 20 that were caused by the higher input weights. For the MSR interleaver there were 295 interleavers with a d_{min} of 18, which means that 36 interleavers or approximately 10% had minimum distances lower than 18 that were caused by the higher input weights.

The 114 VSR and 295 MSR interleavers were carried forward to the next step.

Table 8.1. *Selected VSR Interleavers*

ilv	slope	$w = 0$	$w = 1$	$w = 2$	$w = 3$	$w = 4$
VSR15 000	56.35	(20, 11)	(26, 1)	(20, 7)	(21, 1)	(20, 4)
VSR11 090	56.19	(20, 9)	(26, 1)	(20, 3)	(23, 1)	(20, 6)
VSR11 112	56.08	(20, 8)	(27, 1)	(20, 5)	(23, 1)	(20, 3)
VSR11 194	55.12	(20, 10)	(22, 1)	(20, 7)	(25, 1)	(20, 3)
VSR11 154	53.5	(20, 13)	(33, 1)	(20, 7)	(21, 1)	(20, 6)

Table 8.2. *Selected MSR Interleavers*

ilv	slope	$w = 0$	$w = 1$	$w = 2$	$w = 3$	$w = 4$
MSR19 171	53.36	(18, 14)	(20, 1)	(18, 14)	(19, 1)	(19, 1)
MSR13 031	53.26	(18, 10)	(31, 1)	(18, 10)	(22, 2)	(20, 4)
MSR15 062	52.53	(18, 11)	(33, 1)	(18, 11)	(21, 1)	(20, 14)
MSR18 111	52.5	(18, 12)	(29, 1)	(18, 12)	(20, 1)	(20, 11)
MSR18 023	51.41	(18, 2)	(23, 1)	(18, 2)	(21, 1)	(20, 14)

8.1.3 Step #3

The distance spectrum slope was calculated for each of the interleavers carried over from the previous step. For each interleaver type, the 5 interleavers with the lowest slope values were selected and these 10 interleavers were then carried forward to the next step.

The selected interleavers are given in Table 8.1 for the VSR interleavers, and Table 8.2 for the MSR interleavers. The *ilv* column identifies the interleaver. The first number is the S-value that was used to generate the interleaver. The second number is the number of the interleaver out of the 200 that were generated for each S-value. The *slope* column gives the value of the slope. The $w = 1$ to $w = 4$ columns give the $(d_{min,w}, A(w, d_{min,w}))$ tuples for each input weight w. As before, the $w = 0$ column shows the $(d_{min}, A(d_{min}))$ tuple.

A number of observations can be made from these tables:

- The slope values for the MSR interleavers are lower than those of any of the VSR interleavers.
- When comparing their $(d_{min,w}, A(w, d_{min,w}))$ tuple values, the VSR interleavers are better than the MSR interleavers, for all input weights.

- Almost all of the interleavers exhibit the characteristic mentioned earlier concerning $d_{min,4}$, namely, that the maximum value of $d_{min,4}$ seems to be around 20.
- The S-values that gave the best interleavers are not the highest values considered. This is more prominent for the VSR interleavers, than the MSR interleavers.
- For the VSR interleavers, the interleavers that have the best slope and d_{min} values are not the same, whereas for the MSR interleavers, the same interleaver has the best slope and d_{min} values.

8.1.4 Step #4

In this step, both the PSTCs and FSTCs were considered.

For the PSTC, a subset of the possible puncture masks were considered. In Chapter 7, three CC1 puncture mask families were identified as providing good performance (f:5B, f:5E and f:67), and three were identified as not providing good performance (f:1F, f:37 and f:3D). As noted in Chapter 7, these results are independent of the choice of interleaver. Based on these results, only the three CC1 mask families that gave good performance are considered for this step. All of the CC2 masks will be used. This gives a total of 48 puncture masks to consider. Combining the selected interleavers from the previous step, with all the good puncture masks gives a total of 480 codes.

For the FSTC, all 36 possible puncture masks were used. Combining the selected interleavers from the previous step, with all these puncture masks gives a total of 360 codes.

For each of the considered codes, the weight spectrum for $w = 1, 2, 3, 4$ was determined, and the distance spectrum slope was calculated. Four PSTCs and four FSTCs were then selected. Of the four selected PSTCs, two use a VSR interleaver and two use an MSR interleaver. For each interleaver type, one code was selected based on the minimum distance criteria, and one was selected based on the distance spectrum slope criteria. The same approach was used for selecting the four FSTCs.

The four PSTCs are shown in Table 8.3, and the four FSTCs are shown in Table 8.4. The *criteria* column indicates which of the two design criteria were used in the selection process. The *mask* column indicates the puncture mask, as a hexadecimal number. The remaining columns are the same as used in Table 8.1.

A number of observations can be made regarding the slope and d_{min} values:

Table 8.3. *Selected PSTC Codes*

criteria	ilv	mask	slope	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
d_{min}	MSR18 023	67, 45	354.18	(13, 4)	(18, 1)	(13, 4)	(16, 4)	(14, 19)
	VSR11 112	67, 54	374.21	(14, 5)	(19, 1)	(14, 1)	(17, 7)	(14, 4)
slope	MSR18 111	5B, 15	343.9	(12, 2)	(22, 1)	(12, 2)	(16, 4)	(13, 1)
	VSR11 154	5B, 15	349.09	(12, 1)	(25, 1)	(14, 4)	(15, 1)	(12, 1)

Table 8.4. *Selected FSTC Codes*

criteria	ilv	mask	slope	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
d_{min}	MSR18 023	AF, 11	2162.19	(10, 8)	(15, 1)	(10, 8)	(12, 5)	(12, 49)
	VSR11 112	EE, 44	2312.6	(10, 2)	(17, 1)	(10, 2)	(13, 2)	(12, 21)
slope	MSR18 111	AF, 5	2064.98	(10, 18)	(19, 1)	(10, 18)	(13, 8)	(12, 66)
	VSR11 154	AF, 5	2078.45	(8, 1)	(19, 1)	(8, 1)	(11, 2)	(12, 44)

1. The slope values of the PSTCs are much lower than those of the FSTCs. This would suggest that the PSTCs should perform better in the waterfall region.
2. The minimum distance values of the PSTCs are higher than the FSTCs for all the input weights considered. This would suggest that the PSTCs should perform better in the error floor region.
3. The codes with MSR interleavers have better slope values, than the corresponding code with a VSR interleaver. With one exception, the codes with VSR interleavers have better d_{min} and multiplicity values, than the corresponding code with an MSR interleaver. This is similar to the unpunctured codes in Tables 8.1 and 8.2, where the codes with VSR interleavers have better d_{min} values, and the codes with MSR interleavers have better slope values. The exception is the FSTC slope code, where the code with the MSR interleaver has a better d_{min} value.

In addition, a number of observations can be made regarding the interleavers used in the selected codes:

1. The interleavers used in the selected codes are the same for both the FSTCs and PSTCs, for the same selection criteria. For example, the (MSR18, 023) interleaver is used by the d_{min} code for both the PSTC and FSTC.

2. Looking at the values in Tables 8.1 and 8.2 for the selected interleavers, a number of similarities emerge. In general, the interleaver selected by the slope codes has the best slope value, and the interleaver selected by the d_{min} codes has the best d_{min} value. The exception to this is the MSR interleaver selected by the slope codes. In this case, the selected interleaver has the second best slope value in Table 8.2. Examining the complete list of slope values for the FSTCs, the second best slope code has the (MSR18, 023) interleaver and the puncturing pattern is the same as given in Table 8.4. Similarly, examining the complete list of slope values for the PSTCs, the third best slope code has the (MSR18, 023) interleaver and the puncturing pattern is the same as given in Table 8.3. This suggests that the best unpunctured slope interleavers are among the best punctured slope interleavers, and similarly the best unpunctured d_{min} interleavers are among the best punctured d_{min} interleavers.

Lastly, a number of observations can be made regarding the puncture masks used in the selected codes:

1. For the PSTC and FSTC slope codes, the same puncture mask is selected across the two interleavers. This is the (5B, 15) mask for the PSTC and the (AF, 5) mask for the FSTC.
2. For the PSTC d_{min} codes, the puncture masks only differ in the CC2 component and the same CC1 puncture mask is selected, across the two interleavers,
3. The FSTC d_{min} codes, are the only codes where the puncture masks are not the same or similar across the two interleavers.
4. For the PSTC codes, the selected puncture masks confirm the observations made in Chapter 7, that the f:67 family of puncture masks tends to perform best in the error floor region, as selected by the d_{min} codes, and that the f:5B and f:5E families tend to perform best in the waterfall region, as selected by the slope codes.

8.2 Simulation Results

The eight punctured rate $1/2$ codes listed in Tables 8.3 and 8.4 were simulated. In addition, the unpunctured rate $1/3$ codes using the interleavers from these tables were also simulated. All the simulations used the MaxLogMAP algorithm, with 20 full iterations of the iterative decoding algorithm. The simulations were performed on both the AWGN and

fading channel. Selected simulation results are presented and compared in the following sections, in terms of their BER and FER performance in both the waterfall and error floor regions. The definitions of the waterfall and error floor regions as given in Section 3 for the BER and Section 7 for the FER, will be used for the comparisons.

8.2.1 Interleavers

The BER and FER simulation results for the rate $1/3$ codes using the four interleavers are shown in Fig. 8.1 for the fading channel. The error rate performance of the two VSR interleavers is almost identical, and similarly, the error rate performance of the two MSR interleavers is almost identical. In the waterfall region, all four codes can be considered equivalent and it is only in the error floor region that the VSR interleavers, with their larger d_{min} values, start to be noticeably better. The results for the AWGN channel are similar.

8.2.2 PSTC

The BER and FER simulation results for the four PSTCs are shown in Fig. 8.2 for the AWGN channel.

Comparing the error rate performance of the codes with the same interleaver type, the slope code is better through the entire waterfall region, and well into the error floor region. It is only at the last SNR point (3.5dB) that the d_{min} code starts to approach the performance of the slope code. For the BER, the performance gap narrows, especially for the codes using the VSR interleaver where the slope and d_{min} codes are equivalent. For the FER, this is even more noticeable, as the d_{min} code performs as good as or better than the slope code. This can be explained by noting that the slope codes start to exhibit a flattening of the curve at 3.5dB, which is not evident for the d_{min} codes.

The relative performance of the slope and d_{min} codes is the expected behaviour, since the slope codes have better slope metric values, and worse d_{min} metric values.

Comparing the error rate performance of the codes using the VSR interleavers, with the codes using the MSR interleavers, it is evident that the VSR-based codes perform as good as or better in both the waterfall and error floor regions, than the corresponding MSR-based code using the same design criteria. For the slope codes, the VSR-based code is equivalent to the MSR-based code in the upper part of the waterfall region, and is only marginally

better in the lower part of the waterfall region, at a BER around 10^{-5}. However, in the error floor region, the MSR-based code starts to exhibit a very noticeable flattening of the error rate curves, which widens the performance gap. For the d_{min} codes, the VSR-based code is better throughout both regions of interest, and the performance gap widens as the SNR is increased, even though the MSR-based code does not show a noticeable flattening of the error rate curves.

Based on the d_{min} metric values, it is expected that the VSR-based codes would perform better in the error floor region than the MSR-based codes selected with the same design criteria. This has been confirmed by the simulation results. On the other hand, based on the slope metric values, it would be expected that the MSR-based codes would perform better in the waterfall region, than their VSR-based counterparts. This has not been confirmed by the simulations. This discrepancy can be explained as follows:

1. For the two slope codes, the slope values are very close, so it is reasonable to expect their BER performance to be close, and at the lower end of the error floor, d_{min} and its multiplicity starts to influence performance.
2. The situation for the d_{min} codes is similar to that of the slope codes in point #1 above. Although the difference in slope values is larger, the difference in d_{min} is also larger.
3. For the MSR-based slope code, and VSR-based d_{min} code, the difference in slope values is quite large, and only for these codes does the expected behaviour occur, where the code with the lower slope performs better in the waterfall region.

This can be summarized by noting that for cases where the slope values are quite close, the d_{min} values start to influence the relative performance.

The results for the fading channel are similar, both when comparing the different design criteria, and when comparing the different interleaver types, although the differences are not as significant.

8.2.3 FSTC

The BER and FER simulation results for the four FSTCs are shown in Fig. 8.3 for the AWGN channel. Three of the codes are virtually identical. The VSR-based d_{min} code is the only one that stands out as being clearly better than the other three in both the waterfall and error floor regions. The results for the fading channel are similar.

For the error floor region, the simulation results correspond to what is expected, since the VSR-based d_{min} code has the best d_{min} metric. However, for the waterfall region, the code with the worst slope metric has the best performance. One explanation for why the code with the worst slope metric has the best performance might have to do with the nature of the weight spectrum for these codes.

The weight spectrums and associated fitted lines, for two of the FSTC codes are shown in Fig. 8.4. These two codes are the VSR-based d_{min} code, which has the highest slope value, and the VSR-based slope code. Both codes exhibit the property that, for similar values of d, $A(d_{even}) > A(d_{odd})$, (i.e. the multiplicity for even weight codewords is higher than for odd weight codewords for codewords of similar weight). However, for the slope code, the ratio $A(d_{even})/A(d_{odd})$ is several times larger than for the d_{min} code. As a result, for all values of d,

$$A_{ratio,odd} = \frac{A(d_{odd,slope})}{A(d_{odd,d_{min}})} < 1 \tag{8.1}$$

and

$$A_{ratio,even} = \frac{A(d_{even,slope})}{A(d_{even,d_{min}})} > 1. \tag{8.2}$$

In determining the slope values of the two codes, the fact that, for most similar values of d, $A_{ratio,even} < 1/A_{ratio,odd}$ causes the slope value of the slope code to be lower than that of the d_{min} code. In determining the error performance of the two codes, the odd weight codewords do not contribute significantly to the error performance, since their multiplicity is much lower than for the even weight codewords. Thus it is mainly the even weight codewords that actually contribute to the error performance, and since $A_{ratio,even} > 1$, the d_{min} code has better error performance than the slope code.

The key attribute of the codes are the values of the CC1 mask. All three of the FSTC codes that have a CC1 mask of AF exhibit similar properties relative to the code with a CC1 mask of EE. The distance spectrum slope criteria is not able to properly handle this situation, and so codes that perform poorly have the potential to have good slope values.

8.2.4 Best PSTC vs FSTC

Based on the simulation results of Sections 8.2.2 and 8.2.3, the best PSTC and FSTC were selected and compared.

Table 8.5. *Intersection Points for PSTC vs FSTC*

	BER Curves		FER Curves	
channel	BER	SNR (dB)	FER	SNR (dB)
AWGN	2.6e-6	2.95	1.7e-3	2.4
fading	3.2e-7	6.0	8.5e-5	5.5

The selection was based on the error rate performance in both the waterfall and error floor regions. In each case, there is clearly one code the provides the best performance across both of these regions. In both cases, the best code uses a VSR interleaver. For the PSTC, the best code is the one selected using the slope design criteria, and is the code with puncture mask (5B, 15). For the FSTC, the best code is the one selected using the d_{min} design criteria, and is the code with puncture mask (EE, 44).

The error rate curves for these two codes are shown in Fig. 8.5 (A) for the AWGN channel and Fig. 8.5 (B) for the fading channel. The curves show both the BER and FER for the given channel. The intersection points, where the PSTC starts to perform better than the FSTC are given in Table 8.5.

For the BER on the AWGN channel, the FSTC is better than the PSTC throughout the waterfall region, and it is only in the error floor region that the PSTC starts to provide better BER performance. For the FER on the AWGN channel, the PSTC starts to perform better than the FSTC at a lower SNR than for the BER, towards the lower end of the waterfall region. For the BER on the fading channel, the FSTC is better than the PSTC throughout both the waterfall and error floor regions. The curves only intersect at the last SNR point of 6dB, below the error floor region. For the FER on the fading channel, the PSTC again starts to perform better than the FSTC at a lower SNR than for the BER. Note that for both the AWGN channel and the fading channel, the SNR point where the PSTC starts to perform better than the FSTC is about 0.5dB lower for the FER than for the BER.

As previously discussed in Chapter 7, the PSTC has problems with convergence of the iterative decoder, and so does not exhibit the good performance that it is capable of, for a lower SNR or lower number of iterations. For the results presented in this section, the PSTC does eventually give better error rate performance than the FSTC, but at lower error rates than presented in Chapter 7.

Part of the the reason for this is simply the choice of interleaver. The previous results

used a different random interleaver for each packet, giving the error rate performance of the average random interleaver. The results presented here use the best interleaver selected according to the code design procedure outlined earlier. This selected interleaver lowers the error rate curves considerably on its own. This is evident from the results of Chapter 4, where it was shown that the best interleaver can give an order of magnitude improvement in the error rate over the average interleaver.

Unfortunately, the choice of interleaver on its own, does not account for all of the performance difference between the current results, and those of Chapter 7. One method to improve the performance of the selected codes is investigated in the next section.

8.3 Simulation Results Using Scaling

It is known that scaling of the extrinsic information between each iteration can sometimes improve the convergence properties of the iterative decoding algorithm. However, there is as yet no systematic way to choose appropriate scaling methods or parameters [29, 30]. A number of different scaling methods were investigated in [31], and it was found that increasing the scaling value with each iteration gave the best results. The investigation in [31] was focused on a turbo-decoder using the SOVA algorithm, but this method was also applied to a turbo-decoder using the MaxLogMAP algorithm with good results.

In this section, the above scaling method was used to improve the convergence properties of the PSTC. Thus, the scaling factor is given by

$$scale\,factor = a + ib \tag{8.3}$$

where a is the initial value, b is the per iteration increment, and i is the iteration number, starting at 0. The extrinsic information is multiplied by this scaling factor before it is passed to the next iteration of the decoding algorithm. Through trial and error, the values of $a = 0.5$ and $b = 0.025$ were found to give good results. With 20 full iterations, the final scaling value will be $0.5 + 19 \times 0.025 = 0.975$. Note that a different set of parameters may give equal or better results.

Some of the PSTCs and FSTCs given in Sections 8.2.2 and 8.2.3 were simulated again, but this time incorporating the above scaling parameters. All the other simulation parameters were the same. For the PSTCs, only the slope codes were used, whereas for the FSTCs

only the d_{min} codes were used. Codes using both the VSR and MSR interleavers were considered. The results are shown and compared in the following sections.

8.3.1 Scaled PSTC

The BER and FER simulation results for the two PSTC slope codes are shown in Fig. 8.6 for the AWGN channel. Both the scaled and unscaled simulation results are shown.

For both the waterfall and error floor regions, the scaled codes clearly perform better, providing about 0.25dB coding gain throughout the whole region, relative to the unscaled codes. This is true for both interleaver types, and for both BER and FER. Thus, scaling helps with the convergence of the decoder in these cases.

It is only at the last SNR point, well below the error floor that the performance of the scaled and unscaled codes starts to converge. This is reasonable given that the scaling helps with the convergence of the decoder, and in general, it is easier for the decoder to converge as the SNR is increased. Thus, scaling provides less benefit at higher SNRs. Also, below the error floor region, most of the errors are due to the code structure, and not problems with the convergence of the decoder.

The results for the fading channel are similar, with a coding gain of about 0.45dB for the scaled codes in both the waterfall and error floor regions. However, the scaled codes continue to perform better even for the highest SNR point, which is below the error floor region.

8.3.2 Scaled FSTC

The BER and FER simulation results for the two FSTC slope codes are shown in Fig. 8.7 for the AWGN channel. Both the scaled and unscaled simulation results are shown.

For the entire range of the error rate curves, the performance of the scaled codes is almost identical to their unscaled counterparts. There is a slight gap in the curves, for the region around the SNR=1.5dB point, but this gap is not large enough to indicate that the scaled codes are better.

The results for the fading channel are similar, although there is a small separation between the scaled and unscaled curves over the entire range of the curves. However, this separation is not sufficient to indicate that the scaled codes are better.

Table 8.6. *Intersection Points for Results with Scaling*

	BER Curves		FER Curves	
channel	BER	SNR (dB)	FER	SNR (dB)
AWGN	1.3e-4	2.25	8.6e-2	1.5
fading	2.1e-6	5.3	1.0e-2	4.1

These results indicate that the scaling parameters used do not improve the performance of these codes, but neither do they degrade the performance of the codes. A different choice of scaling parameters may provide some improvement in the performance.

8.3.3 Best Scaled PSTC vs Scaled FSTC

In this section, the error rate performance of the best PSTC and FSTC is compared, using the simulation results with scaling. Although the chosen scaling parameters do not improve the performance of the FSTC, neither do they degrade the performance, so it is valid to use the simulation results with scaling for both codes.

As with the unscaled case, the VSR interleaver based codes provide better error rate performance than the MSR interleaver based codes. Thus, the same codes are compared in this section, as were compared in Section 8.2.4. For the PSTC, the best code is the one selected using the slope design criteria, and is the code with puncture mask (5B, 15). For the FSTC, the best code is the one selected using the d_{min} design criteria, and is the code with puncture mask (EE, 44).

The error rate curves for these two codes are shown in Fig. 8.8 (A) for the AWGN channel and Fig. 8.8 (B) for the fading channel. The curves show both the BER and FER for the given channel. The intersection points, where the PSTC starts to perform better than the FSTC are given in Table 8.6. The coding gain of the PSTC at select points is given in Table 8.7.

For the BER on the AWGN channel, the PSTC starts to perform better than the FSTC in the middle of the waterfall region, and continues to perform better than the FSTC into the error floor region. For the FER on the AWGN channel, the PSTC performs better than the FSTC through both the waterfall and error floor regions. For the BER on the fading channel, the FSTC is better than the PSTC throughout the waterfall region, and it is only

Table 8.7. *Coding Gains for Results with Scaling*

	Coding Gain at			
channel	BER=10^{-5}	BER=10^{-6}	FER=10^{-3}	FER=10^{-4}
AWGN	0.16 dB	0.36 dB	0.3 dB	0.5 dB
fading	-0.14 dB	0.16 dB	0.14 dB	0.45 dB

in the error floor region that the PSTC starts to perform better than the FSTC. For the FER on the fading channel, the PSTC again performs better than the FSTC throughout both the waterfall and error floor regions.

The scaled PSTC performs better against the FSTC, than the unscaled version compared in Section 8.2.4. However, the coding gains of the scaled PSTC are still less than achieved by the PSTC in Chapter 6. Part of the reason for this is again the choice of interleaver, and the fact that the interleaver causes the error rate curves to be steeper for the scaled PSTC codes. The intersection points between the PSTC and FSTC do actually occur at slightly lower SNR values for the scaled PSTC codes, than the PSTC codes of Chapter 6, but because of the steeper error rate curves, the actual coding gain is smaller.

From the above, it is clear that puncturing data bits is not always beneficial. For applications that only operate in the upper end of the waterfall region, around a BER of 10^{-3}, puncturing data bits is actually detrimental. However, for applications that need to operate in both the waterfall and error floor region, or for applications where the FER is more important than the BER, it appears that puncturing of the data bits is beneficial.

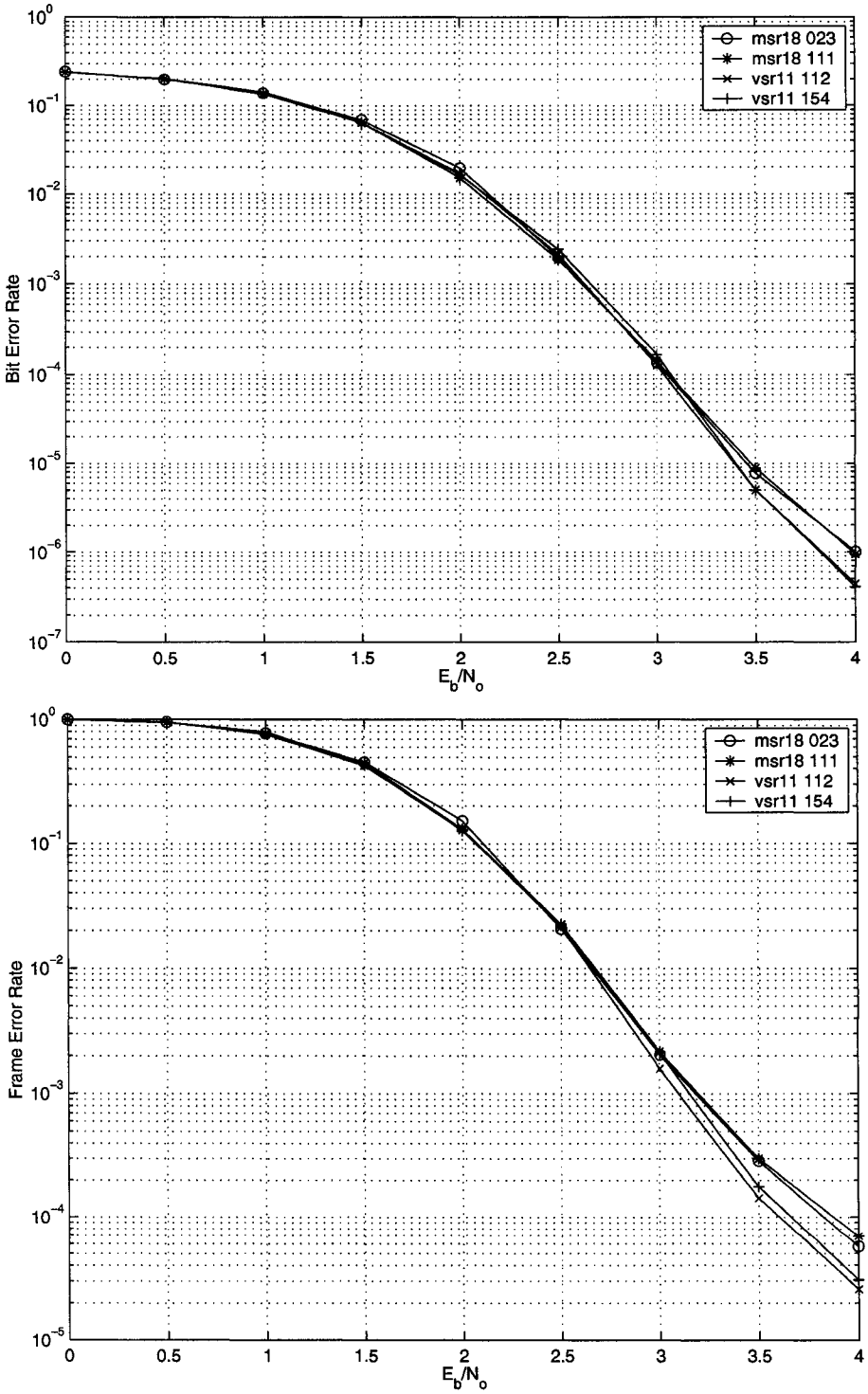

Figure 8.1. *Ilv Fading (A) BER (B) FER*

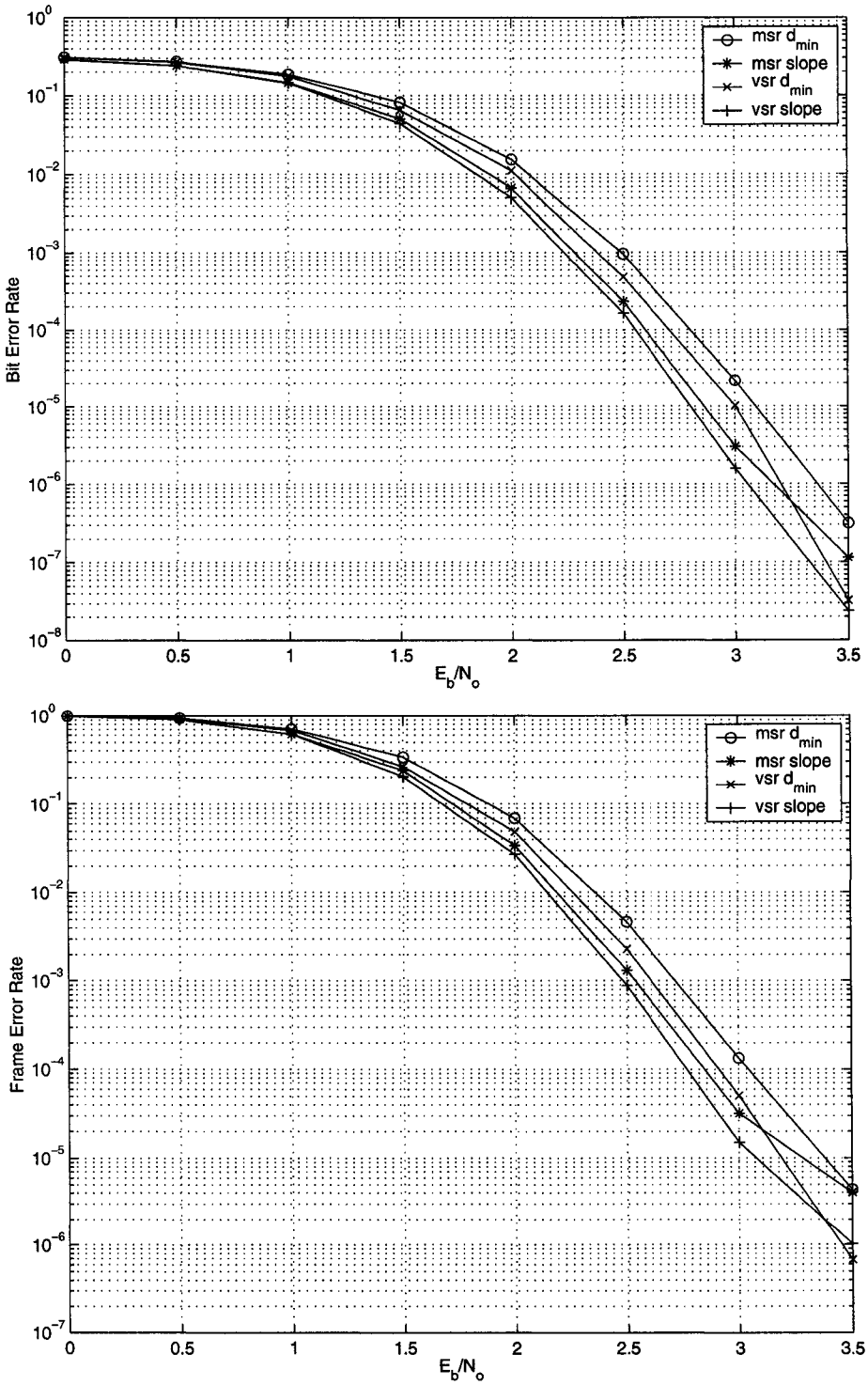

Figure 8.2. *PSTC AWGN (A) BER (B) FER*

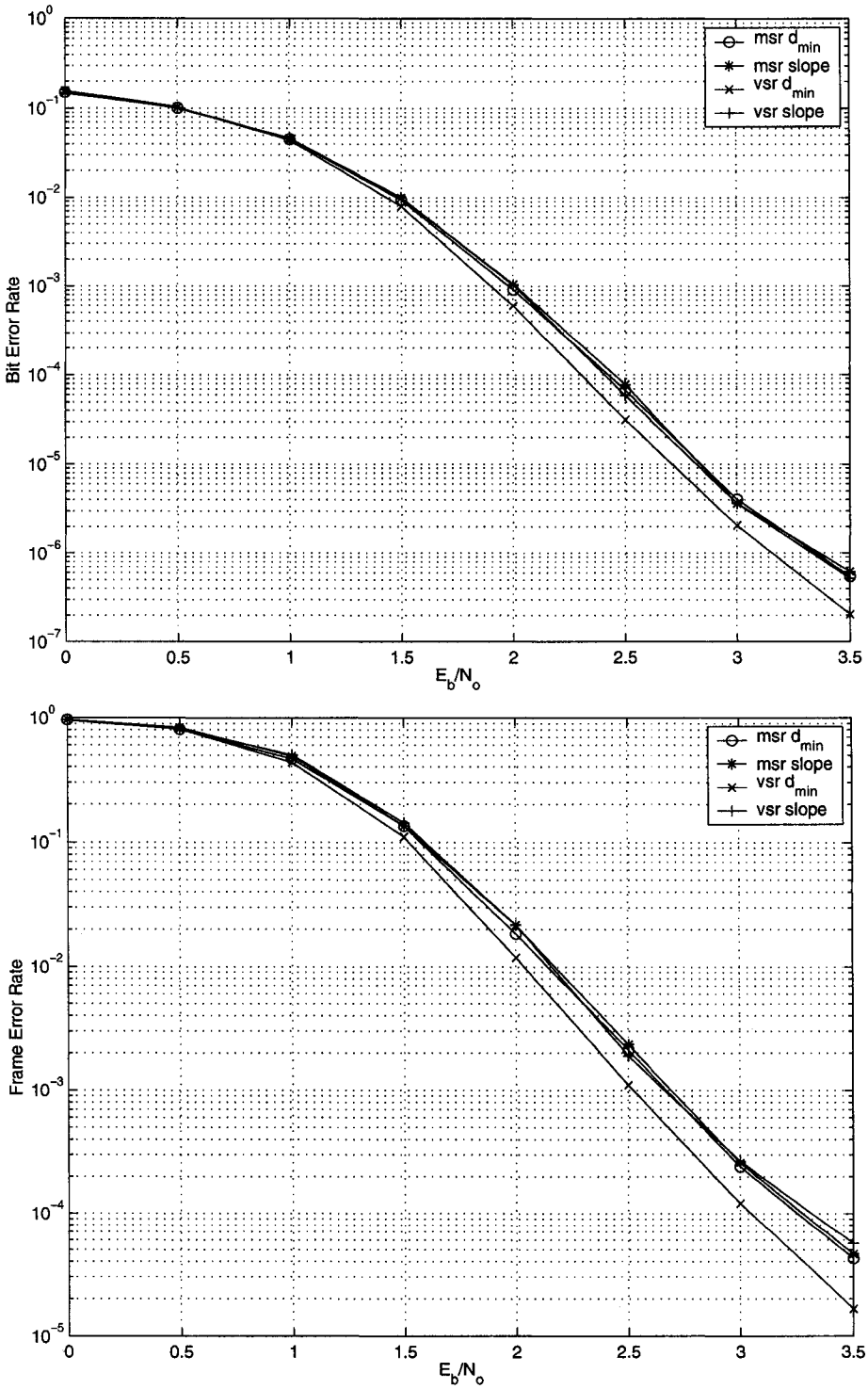

Figure 8.3. *FSTC AWGN (A) BER (B) FER*

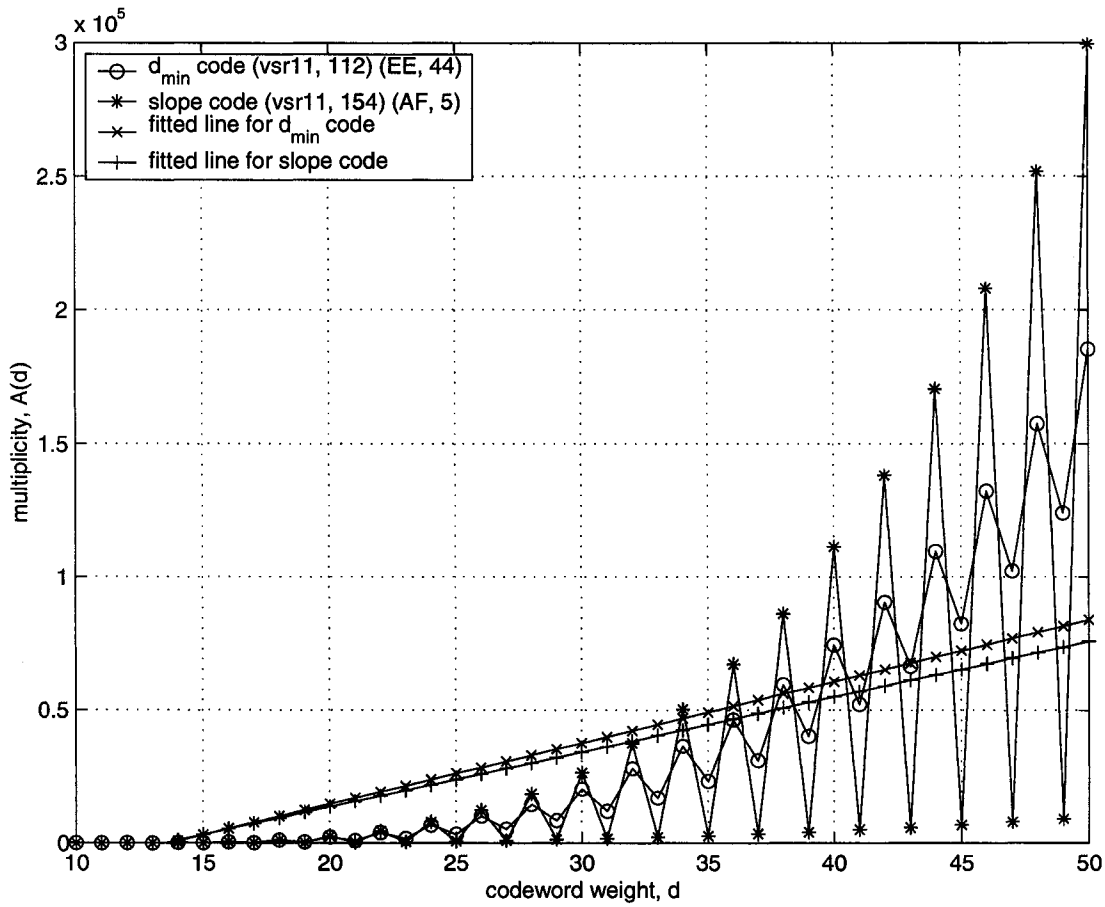

Figure 8.4. *Comparison of Weight Spectrum for FSTC*

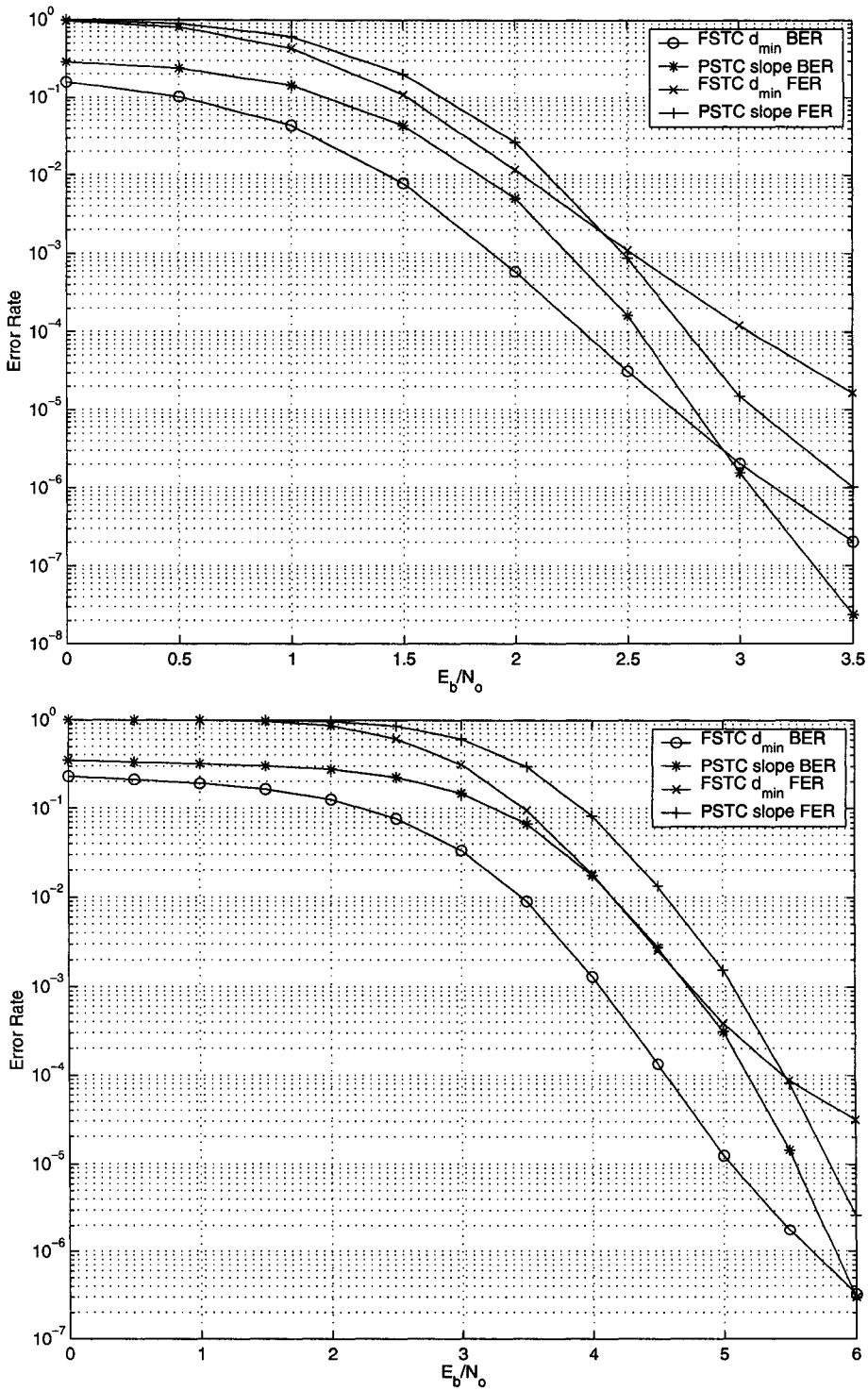

Figure 8.5. *Best (A) AWGN (B) Fade*

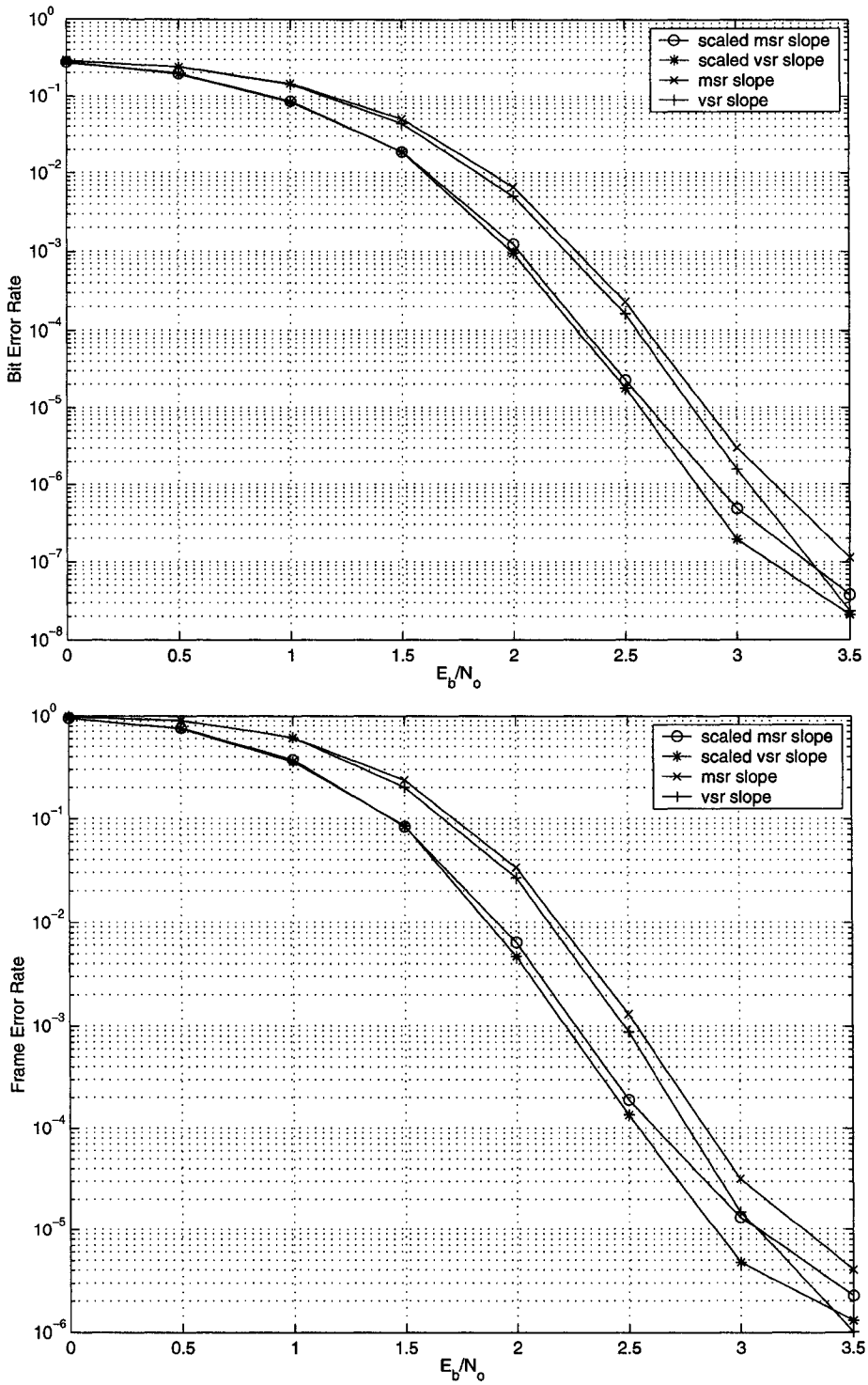

Figure 8.6. *PSTC Scaled AWGN (A) BER (B) FER*

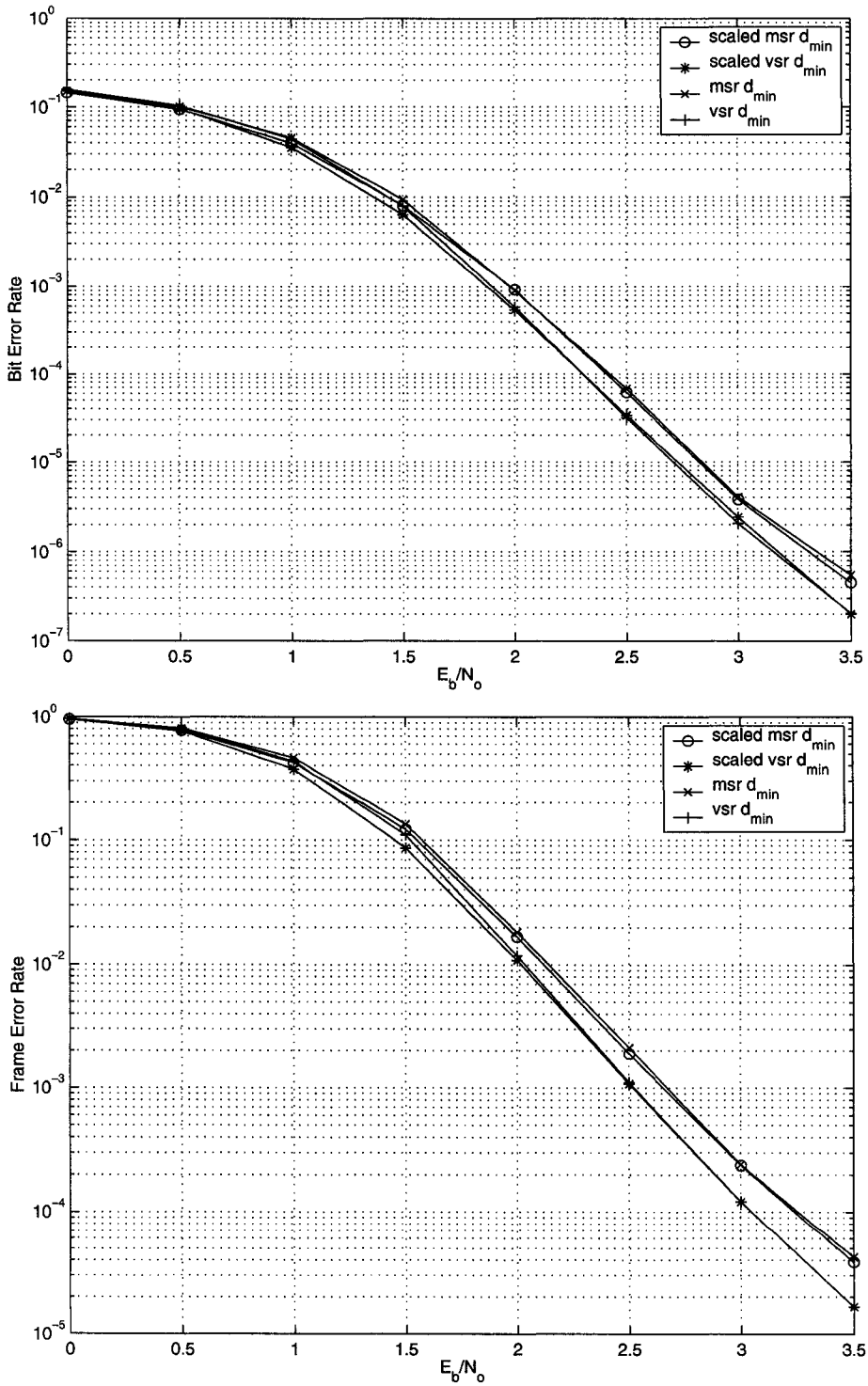

Figure 8.7. *FSTC Scaled AWGN (A) BER (B) FER*

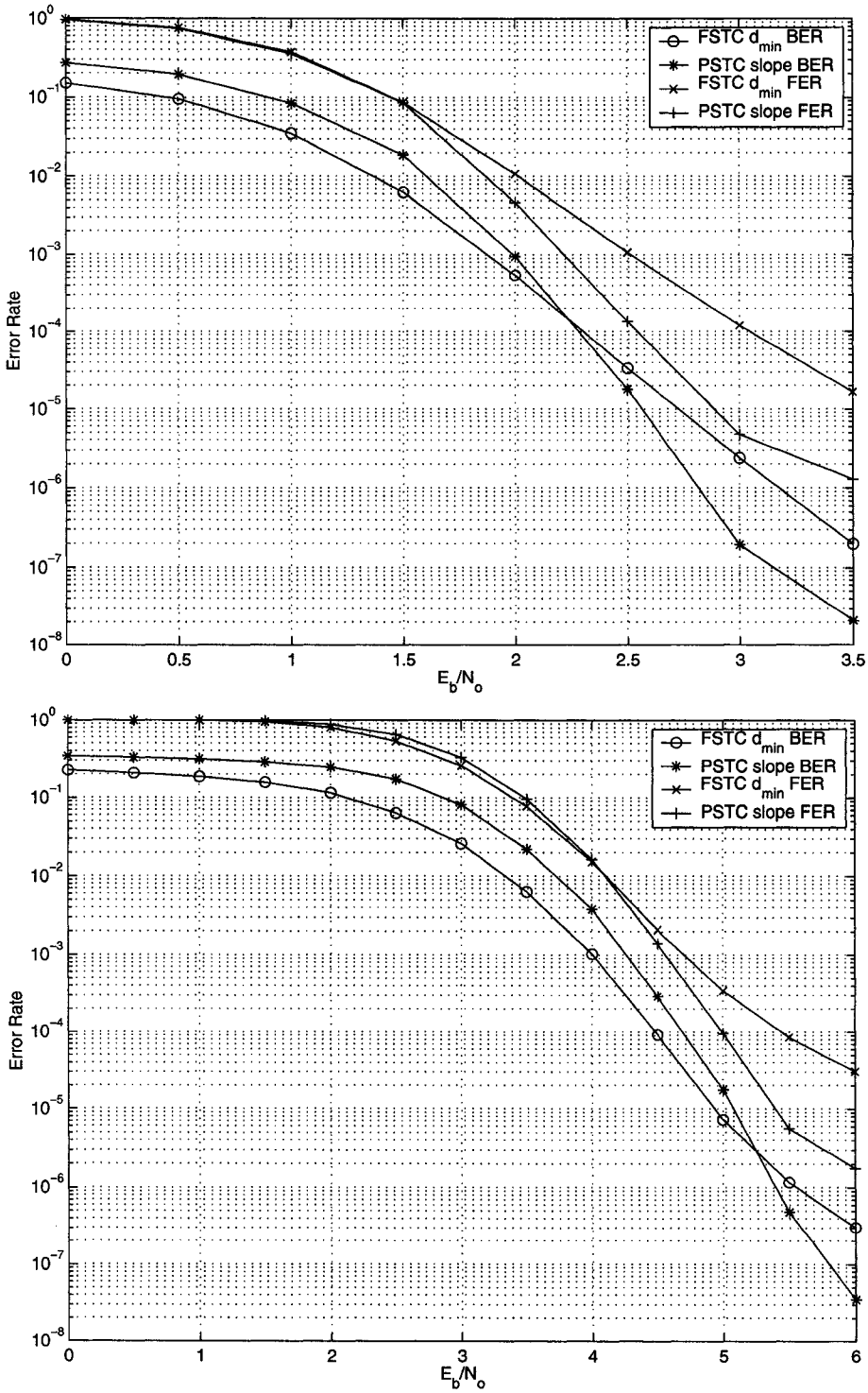

Figure 8.8. *Best Scaled (A) AWGN (B) Fade*

Chapter 9

Summary and Conclusions

This dissertation studied methods to improve the error floor performance of turbo-codes that use low complexity constituent codes. The two methods studied were interleaver selection, and selective puncturing of the information bits.

Chapter 1 provided an introduction to the problem studied in this dissertation.

In Chapter 2, various background information and concepts were introduced, including the channel models used, the definition of the weight/distance spectrum, and a brief overview of turbo-codes.

In Chapter 3, two code design methods were introduced that would be used in later chapters. These were the minimum distance design criteria, and the distance spectrum slope design criteria. In order to understand the relevance and importance of these design criteria, union bounds on the error rate performance of block codes were introduced. The applicability of these bounds to the performance evaluation of turbo-codes was also discussed, in the context of evaluating the bounds for specific turbo-codes, rather than an ensemble of turbo-codes. Details of the simulation techniques used throughout this dissertation for evaluating the performance of turbo-codes were also presented. This included a review of methods to calculate the confidence interval of the simulations and how to apply this confidence interval for the purposes of comparing the relative performance of different codes.

Chapter 4 introduced interleavers as they are applied to turbo-codes. The focus was on evaluating, through simulation, several techniques for generating random interleavers. These techniques are the pseudo-random interleaver, s-random interleaver, modified s-random interleaver, and variable s-random interleaver. For each particular technique, a large number of interleavers were generated (200 in each case), and their performance within a turbo-code was evaluated through simulation. The resulting BER and FER statis-

tics were compared. It was found that all three of the s-random based interleavers provided very good performance, and in particular, an improvement of at least an order of magnitude over the average random interleaver is possible, for all the block lengths considered. In addition, the temporary relaxing of the design constraints as implemented in the modified s-random and variable s-random interleavers did not appear to have a detrimental affect on the error rate performance of the interleavers.

In Chapter 5, the design criteria introduced in Chapter 3, were applied to the interleavers that were introduced and generated in Chapter 4. For all of the interleavers from Chapter 4, the minimum distance and multiplicity were investigated. For the pseudo-random interleavers, it was found that most of the minimum distance terms are caused by weight-2 inputs. In addition, the minimum distance due to weight-2 inputs was found to be fairly constant across all three block lengths. For the s-random based interleavers, it was found that the minimum distance due to weight-4 inputs was fairly constant across all three block lengths. For the short block lengths, the weight-2 inputs are responsible for the minimum distance codeword, but as the block length is increased, the minimum distance due to weight-2 inputs also increases, and it is the weight-4 input that causes most of the minimum distance codewords. For the pseudo-random interleavers from Chapter 4, the quality of the choices made by the d_{min} and slope design criteria were evaluated, by comparing the selected interleavers, against the statistics for all the pseudo-random interleavers. In general, both design criteria made good selections. Usually, the selected interleavers were within the top 25% of all the pseudo-random interleavers.

In Chapter 6, two approaches for puncturing turbo-codes were presented, and compared using simulation results. These are the partially-systematic turbo-codes (PSTC), which puncture some of the data bits as well as the parity bits, and the fully systematic turbo-codes (FSTC), which only puncture the parity bits. In general, it was found that puncturing some of the data bits improves the error rate performance towards the lower end of the waterfall region and in the error floor region. Usually the improvement in the FER performance is greater than the improvement in the BER performance. At a BER of 10^{-5}, there is about a 0.5dB coding gain for the PSTC. At a FER of 10^{-3}, there is about a 0.6dB coding gain for the PSTC in the AWGN channel, and about a 1.0dB coding gain in the fading channel. The amount of coding gain appears to be independent of the block length.

The PSTCs with $p_u = 1/2$, were studied in more detail in Chapter 7. Simulation results

of the error rate performance for all the PSTCs was obtained. The puncturing masks were partitioned according to the six different CC1 mask families. Two CC1 mask families, f:5B and f:5E, were identified as giving the best performance in the waterfall region, and one CC1 mask family, f:67, was identified as giving the best performance in the error floor region. The other three CC1 mask families were identified as performing poorly in both the waterfall and error floor regions. By examining the bit patterns of the puncturing masks that performed well, it was suggested that puncturing alternate data bits gave better performance in the error floor region, while maximizing the number of runs of length one gave better performance in the waterfall region.

In Chapter 8, the results of the previous chapters were combined by applying the code design methods to the selection of interleavers and puncturing masks. The error rate performance of the selected codes was then obtained through simulation and compared. For the PSTCs, the codes selected according to the slope criteria gave the best performance in the regions of interest. For the FSTCs, the codes selected according to the d_{min} criteria gave the best performance in the regions of interest. In both cases, it was found that the codes using VSR interleavers gave better performance than the codes using the MSR interleavers. In comparing the PSTCs to the FSTCs, the PSTCs were found to not perform better than the FSTCs in the regions of interest, mainly due to issues with the convergence of the decoding algorithm. To improve the convergence of the decoding algorithm for the PSTCs, scaling of the extrinsic information was introduced and applied to select codes. In this case, the PSTCs performed better than the FSTCs towards the lower end of the waterfall region and in the error floor region, but with smaller coding gains than given in Chapter 6.

The main contributions of this dissertation are as follows:

- In Chapter 4, two modifications to the s-random technique were introduced, both using an s-value relaxation technique. To provide a comprehensive comparison of the different interleaver types, a large group of interleavers of each type were generated and compared based on simulation results. From these results, it was concluded that the s-random based techniques are the best for all block lengths, and that the s-value relaxation technique did not adversely affect the achievable performance of the generated interleavers.
- In Chapter 5, a comprehensive study of the minimum distance properties of the interleavers from Chapter 4 was conducted, and some conclusions were drawn as to the

impact of different input weights. In addition, for the pseudo-random interleavers of Chapter 4, the two code design criteria of Chapter 3, were evaluated by comparing the selected interleavers against the group statistics.

- In Chapter 6, all possible puncture masks, for the given code construction of both the PSTC and FSTC, were generated and extensively compared. This served, in part, to verify the work of [27], that puncturing of data bits does indeed give the potential for better performance. In addition, an extensive comparison of the best puncture masks for each type of code, was given, and conditions under which the PSTC would give better performance were identified.
- In Chapter 7, the PSTCs from Chapter 6 were extensively studied. The concept of mask families was introduced, and it was determined, through simulation results, which mask families gave better performance for the regions of interest. Some observations were provided to explain the relative performance of the different mask families.
- In Chapter 8, a code design procedure incorporating both interleavers and puncturing patterns was introduced and applied. It was verified that the variable s-random interleavers provide better error rate performance than the modified s-random interleavers. This comparison was made using a range of s-values, to reduce the influence of the different definitions of the s-value. It was shown that, for the PSTC, the distance spectrum slope design criteria was best, whereas for the FSTC, the d_{min} design criteria was best. It was also shown that the PSTC can provide better error rate performance than the FSTC, in the waterfall and error floor regions, but only by scaling the extrinsic information to improve the convergence properties of the iterative decoder.

9.1 Suggestions for Future Work

A number of suggestions for future work are presented in this section. These suggestions fall into three main categories: code design techniques, interleaver generation, and puncturing.

The distance spectrum slope design criteria does not always lead to good selections being made. The main issue appears to be with how the slope is calculated. One possible solution is to determine the envelope of the distance spectrum, and then calculate the

slope based on this envelope. Other methods for calculating the slope should also be evaluated, such as using a quadratic approximation to the distance spectrum, rather than a linear approximation.

The code design techniques considered within this dissertation are based on evaluating the structure of the turbo-code. A different approach, based on evaluating the convergence properties of the iterative decoding algorithm, have become very popular in the last few years [32, 33, 34]. These techniques are mainly applicable to the waterfall region of the error rate curve. These techniques all assume that the extrinsic information generated by the SISO decoder can be approximated by a gaussian random variable. This gaussian approximation is not very accurate for the first few iterations, but as the number of iterations are increased the approximation becomes more accurate. In addition, there is the assumption of long block lengths, of several thousand bits or more. The intent of the design procedure is to identify an SNR threshold, below which the iterative decoder is not guaranteed to converge. Above this SNR threshold, the decoder should converge, although there is no guarantee as to how many iterations will be required for the decoder to converge. These techniques have also been applied to shorter block lengths [35, 36]. It would be worthwhile to examine these convergence-based design techniques, and compare the quality of their selections, against the design techniques presented in this dissertation.

The interleaver generation techniques presented in this dissertation are mainly concerned with reducing the impact of weight-2 inputs on the weight spectrum. It would be useful generate additional interleavers, by applying some of the techniques that consider higher weight inputs (e.g. [21, 24]), and comparing these new interleavers with those presented in this dissertation. We speculate that for the short block lengths used here, that these techniques that consider higher weight inputs, will not be able to generate significantly better interleavers.

All of the PSTCs considered within this dissertation use $p_u = 1/2$, and puncturing masks of length 8. It would be useful to investigate different values of p_u, with values slightly less than $1/2$, as this would improve the convergence properties of the iterative decoder. In order to facilitate this, longer puncturing masks would also be necessary, since the length of the puncturing mask determines the allowable values of p_u. Even without changing the value of p_u, it would be worthwhile to examine puncturing masks of longer length, as this would add more randomness into the structure of the turbo-code, which has

the potential to improve the performance.

Bibliography

[1] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall/IEEE Press, 1996.

[2] L.-N. Lee, A. R. Hammons Jr, F.-W. Sun, and M. Eroz, "Application and standardization of turbo codes in third-generation high-speed wireless data services," *IEEE Transactions on Vehicular Technology, Vol. 49, No. 6*, pp. 2198–2206, 2000.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," *Proc. IEEE International Conference on Communications*, pp. 1064–1070, 1993.

[4] J. G. Proakis, *Digital Communications*, McGraw Hill, 2001.

[5] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, 1995.

[6] C. Heegard and S. Wicker, *Turbo Coding*, Kluwer Academic Publishers, 1999.

[7] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory, vol 20*, pp. 284–287, 1974.

[8] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," *Proc. IEEE Global Telecommunications Conference*, pp. 1680–1686, 1989.

[9] J. Hagenauer, P. Robertson, and L. Papke, "Iterative (turbo) decoding of systematic convolutional codes with the map and sova algorithms," *Proc. of ITG Conference on Source and Channel Coding*, pp. 21–29, 1994.

[10] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," *Proc. IEEE International Conference on Communications*, pp. 1009–1013, 1995.

[11] L. C. Perez, J. Seghers, and D. J. Costello, "A distance spectrum interpretation of turbo-codes," *IEEE Transactions on Information Theory, vol. 42, no. 6*, pp. 1698–1709, 1996.

[12] D. Divsalar and F. Pollara, "Turbo codes for pcs applications," *Proc. IEEE International Conference on Communications*, pp. 54–59, 1995.

[13] A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding*, McGraw Hill, 1979.

[14] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (Turbo) codes," *Proc. IEEE Global Telecommunications Conference*, pp. 1298–1303, 1994.

[15] E. K. Hall and S. G. Wilson, "Design and analysis of turbo codes on rayleigh fading channels," *IEEE Journal on Selected Areas in Communications, vol. 16, no. 2*, pp. 160–174, 1998.

[16] 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, 1996.

[17] A. J. Viterbi, A. M. Viterbi, J. Nicolas, and N. T. Sindhushayana, "Perspectives on interleaved concatenated codes with iterative soft-output decoding," *Proc. International Symposium on Turbo Codes*, pp. 47–54, 1997.

[18] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *IEE Electronic Letters, Vol. 32, No. 5*, pp. 445–446, 1996.

[19] D. N. Rowitch and L. B. Milstein, "On the performance of hybrid fec/arq systems using rate compatible punctured turbo (rcpt) codes," *IEEE Transactions on Communications, vol. 48, no. 6*, pp. 948–959, 2000.

[20] M. C. Jeruchim, "Techniques for estimating the bit error rate in the simulation of digital communication systems," *IEEE Journal on Selected Areas in Communications, vol. SAC-2, no. 1*, pp. 153–170, 1984.

[21] C. Fragouli and R. D. Wesel, "Semi-random interleaver design criteria," *Proc. IEEE Global Telecommunications Conference*, pp. 2352–2356, 1999.

[22] M. Breiling, S. Peeters, and J. Huber, "Interleaver design using backtracking and spreading methods," *Proc. IEEE International Symposium on Information Theory*, p. 451, 2000.

[23] S. Crozier, "New high-spread high-distance interleavers for turbo-codes," *Proc. 20th Biennial Symposium on Communications*, pp. 3–7, 2000.

[24] M. Breiling and J. B. Huber, "Upper bound on the minimum distance of turbo codes," *IEEE Transactions on Communications, Vol. 49, No. 5*, pp. 808–815, 2001.

[25] M. Breiling, "A logarithmic upper bound on the minimum distance of turbo codes," *Submitted to IEEE Transactions on Information Theory*, April, 2001.

[26] O. F. Acikel and W. E. Ryan, "Punctured turbo-codes for bpsk/qpsk channels," *IEEE Transactions on Communications, Vol. 47, No. 9*, pp. 1315–1323, 1999.

[27] I. Land and P. Hoeher, "Partially systematic rate 1/2 turbo codes," *Proc. 2nd International Symposium on Turbo Codes and Related Topics*, pp. 287–290, 2000.

[28] I. Land and S. Chaoui, "Comparison of convolutional coupled codes and partially

systematic turbo codes for medium code lengths," *Proc. 4th International ITG Conference on Source and Channel Coding*, pp. 99–104, 2002.

[29] T. Richardson, "The geometry of turbo-decoding dynamics," *IEEE Transactions on Information Theory, Vol. 46, No. 1*, pp. 9–23, 2000.

[30] L. Kocarev, Z. Tasev, and A. Vardy, "Improving turbo codes by control of transient chaos in turbo-decoding algorithms," *IEE Electronic Letters, Vol. 38, No. 20*, pp. 1184–1186, 2002.

[31] Z. Blazek and V. K. Bhargava, "A dsp-based implementation of a turbo-decoder," *Proc. IEEE Global Telecommunications Conference*, pp. 111–222, 1998.

[32] H. El Gamal and A. R. Hammons Jr., "Analyzing the turbo decoder using the gaussian approximation," *IEEE Transactions on Information Theory, Vol. 47, No. 2*, pp. 671–686, 2001.

[33] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on density evolution," *IEEE Journal on Selected Areas in Communications, Vol. 19, No. 5*, pp. 891–907, 2001.

[34] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications, Vol. 49, No. 10*, pp. 1727–1737, 2001.

[35] J. W. Lee and R. E. Blahut, "Analysis of the extrinsic values in the finite length turbo decoding," *Proc. Conference on Information Sciences and Systems*, 2002.

[36] A. O. Yilmaz and W. E. Stark, "Application of gaussian approximation to iterative decoding with finite blocklengths," *Proc. IEEE Fall Vehicular Technology Conference*, 2002.