ANNUAL STATUS REPORT

Error Control Techniques for Satellite and
Space Communications
NASA Grant Number NAG5-557

Principal Investigator:
Daniel J. Costello, Jr.

November 1994

Summary of Progress

In this report, we will focus on the results included in the Ph.D. dissertation of Dr. Diane G. Mills. Dr. Mills completed her dissertation and received her Ph.D. degree in August 1994. A copy of the dissertation is included as Appendix A to this report. Two journal papers have been submitted based on Dr. Mills' research [1,2]. In addition, three conference presentations have resulted from this work [3-5]. The following paragraphs contain a brief summary of this research.

The purpose of unequal error protection (UEP) is to provide a higher degreee of error protection for the more important bits, without incurring the associated increase in complexity/cost/bandwidth that would occur if the protection were increased for the entire information stream. For example, UEP coding can be used to transmit images over noisy channels where different parts of an image require different levels of error protection. A potential application is the transmission of images over NASA's deep space network. The goal of this research was to investigate the unequal error protection capabilities of convolutional codes and to extend the results to trellis codes.

First, the effective free distance vector, $\mathbf{d}$, was defined as an alternative to the free distance as a primary performance parameter for UEP convolutional encoders. For a given (n, k, m) convolutional encoder the effective free distance vector is defined as the k-dimensional vector $\mathbf{d} = (d_0, d_1, \cdots, d_{k-1})$, where d_j, the j^{th} effective free distance, is the lowest Hamming weight among all code sequences that are generated by input sequences with at least one "1" in the j^{th} position. It is evident that the free distance of the code is the minimum of the effective free distances. Although the free distance for a code is unique to the code and independent of the encoder realization, the effective free distance vector is dependent on the encoder realization.

A modified transfer function, which provides a method to calculate $\mathbf{d}$, was developed. The modified transfer function incorporates a new branch labeling method which may be used to calculate the effective free distance vector when used in conjunction with standard algorithms that were originally developed to calculate the free distance of a code.

Several upper bounds on the effective free distance vector were derived. The bounds may be used to identify encoder configurations that have good potential for unequal error protection. Then computer searches for good unequal error protection encoders were conducted. A primary goal of the searches was to find encoders with at least one effective free distance greater than the free distance of the optimal code of the same rate and memory order. A decrease in free distance was acceptable. A number of binary convolutional encoders meeting this goal were found. Bit error rate (BER) performance for the encoders was simulated, and this confirmed the effective free distance as a measure of unequal error protection. At the same time, the BER plots indicated that the number of code sequences with Hamming weights equal to the individual effective free distance is also an important measure of performance.

Next, trellis coded modulation (TCM) systems with unequal error protection were investigated. It was determined that providing unequal error protection with TCM coding is more difficult due to the limitations of the signal constellations. A limited number of trellis codes

were found, however, that provide a measure of unequal error protection. Further work on this subject is in progress.

References

[1] D. G. Mills, D. J. Costello, Jr., and R. Palazzo, Jr., "Achieving Unequal Error Protection with Convolutional Codes", *IEEE Trans. Inform. Th.*, submitted for publication.

[2] D. G. Mills and D. J. Costello, Jr., "On the Unequal Error Protection Capabilities of Trellis Codes", *IEEE Trans. Inform. Th.*, submitted for publication.

[3] D. G. Mills and D. J. Costello, Jr., "An Upper Bound on the Free Distance of Double Memory Convolutional Codes", *Proc. Allerton Conf. on Commun., Cont., and Comput.*, pp. 20-26, Monticello, IL, September 1992.

[4] D. G. Mills and D. J. Costello, Jr., "Using a Modified Transfer Function to Calculate Unequal Error Protection Capabilities of Convolutional Codes", *Proc. IEEE International Symposium on Information Theory*, p. 144, San Antonio, TX, January 1993.

[5] D. G. Mills and D. J. Costello, Jr., "A Bound on the Unequal Error Protection Capabilities of Rate k/n Convolutional Codes", *Proc. IEEE International Symposium on Information Theory*, p. 274, Trondheim, Norway, June 1994.

Appendix A

The Unequal Error Protection Capability of Convolutional and Trellis Codes

THE UNEQUAL ERROR PROTECTION CAPABILITIES OF CONVOLUTIONAL AND TRELLIS CODES

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirement

for the Degree of

Doctor of Philosophy

by

Diane Grieselhuber Mills, BSEE, MSEE

Daniel J. Costello, Jr., Director

Department of Electrical Engineering

Notre Dame, Indiana

July, 1994

In loving memory of

Anna BeCraft Grieselhuber,

who inspired us all with her

class, determination, and strength.

THE UNEQUAL ERROR PROTECTION CAPABILITIES OF CONVOLUTIONAL AND TRELLIS CODES

Abstract

by

Diane Grieselhuber Mills

The research discussed in this dissertation studies the unequal error protection capabilities of convolutional and trellis codes. In certain environments, a discrepancy in the amount of error protection placed on different information bits is desirable. Examples of environments which have data of varying importance are a number of speech coding algorithms, packet switched networks, multi-user systems, embedded coding systems, and high definition television. Encoders which provide more than one level of error protection to information bits are called unequal error protection (UEP) codes.

In this work, the effective free distance vector, $\mathbf{d}$, is defined as an alternative to the free distance as a primary performance parameter for UEP convolutional and trellis encoders. For a given (n, k), convolutional encoder, $\mathbf{G}$, the effective free distance vector is defined as the k-dimensional vector $\mathbf{d} = (d_0, d_1, \cdots, d_{k-1})$, where d_j, the j^{th} effective free distance, is the lowest Hamming weight among all code sequences that are generated by input sequences with at least one "1 " in the j^{th} position. It is shown that, although the free distance for a code is unique to the code and independent of the encoder realization, the effective distance vector is dependent on the encoder realization.

A modified transfer function, which provides a method to calculate $\mathbf{d}$, is presented. The modified transfer function developes a new branch labelling method that allows

standard algorithms that were originally developed to calculate the free distance of a code to calculate the effective distance vector.

Several upper bounds on **d** are derived and compared. The results of searches for good unequal error protection codes are presented. A primary goal of the searches was to find encoders with at least one effective distance greater than the free distance of the optimal code of the same rate and memory order. Bit error rate (BER) plots for the enocders are presented, confirming the effective distance as a measure of unequal error protection. At the same time, the BER plots show that the number of code sequences with Hamming weights equal to the individual effective distance is more important than expected.

Trellis coded modulation (TCM) systems with unequal error protection are investigated. It is determined that providing unequal error protection with TCM coding is difficult due to the limitations of the signal constellations. Topics for future investigation are identified.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor D.J. Costello, Jr., for his guidance and advice throughout this investigation. His expertise was very helpful and much appreciated.

Dr. Peter Bauer, Dr. Robert L. Stevenson, and Dr. Mark A. Herro also deserve thanks for serving on my dissertation committee and commenting on my work. Dr. Reginaldo Palazzo and Lance Perez both deserve special thanks for the many hours of discussion and suggestions that they provided.

I would like to thank the National Science Foundation, which provided three years of financial support with an Award for Creativity in Engineering.

Finally, I would like to thank my husband, Jim, and my sons, Robert, Andrew, and Daniel. I couldn't have finished this work without their cooperation and support.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The purpose of a communication system is to transmit information or data from one point to another. Using the sampling theorem, analog data may be digitized without loss of quality or information, allowing the use of digital communication systems. Digital communications systems typically perform better than analog communication system, for a number of reasons. For instance, digital processing reduces signal degradation, allows source coding to remove redundancies, thereby decreasing the transmission rate, and allows channel coding to decrease the error rate. Digital systems are generally more reliable and easier to maintain than analog systems. In addition, because digital systems rely more on software than hardware, it is often relatively easy to upgrade a digital system.

As the volume of transmissions increases, bandwidth and energy-limited channels, introduce more errors. Transmission errors degrade the performance by reducing throughput, storage capacity, or reliability. Error control can be viewed two ways: for the same power and cost, the error rate may be decreased, or the error rate may be maintained, at a reduction in power and hardware costs. Channel coding, an error control technique, improves the reliability of digital data links and storage media.

Examples of systems in which coding is appropriate include computer storage systems, communication networks, deep-space transmission systems, telephone channels, satellite channels, and optical storage system [32].

This chapter reviews basic communications concepts. Section 2 briefly describes a general digital communications system and discusses the issues that are usually important. Section 3 outlines the dissertation.

1.2 Digital Communications Systems

A model of a typical digital communication system is shown in Figure 1.1.

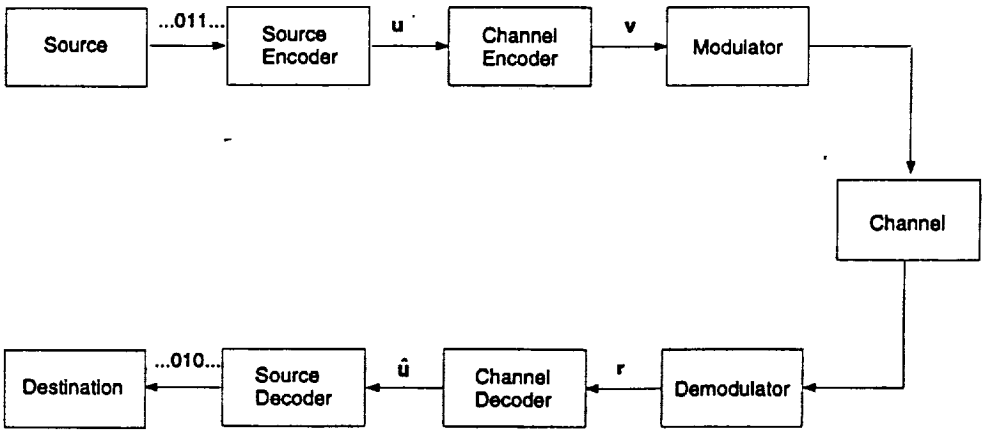


Figure 1.1: General communications system

The information source produces digital information which is to be transmitted to the destination. Using the information from the source, the source encoder generates a binary k-bit message $\mathbf{u}_t$, at each time instant t. It is not necessary that the the source encoder produces a binary output, but the assumption simplifies the discussion. Examples of sources include: a voice, a measuring instrument, and a computer. The information sequence, $\mathbf{u}$, is a semi-infinite binary sequence. The channel encoder converts the information sequence into the code sequence, $\mathbf{v}$, following some channel

coding rules. The goal of the channel encoder is to add enough redundancy so that the information may be reliably transmitted over the channel. The codewords are passed to the modulator, which generates continuous channel wave forms, $s(t)$, called channel signals. The channel signals are then transmitted over the channel to the receiver. The channel is any type of transmission medium, which may be, for example, a telephone line, satellite link, optical link, or magnetic storage media. Included in the channel model is a noise source, which is dependent on both the type of channel, and the specific channel used. The noise corrupts the original signal so that the continuous waveform at the output of the channel is $r(t)$. The demodulator produces the received sequence, $\mathbf{r}$. It is assumed that an optimum demodulator, such as a matched filter or correlation detector followed by a sampling switch and quantizer, is used. The channel decoder applies a decoding rule to the binary sequence $\mathbf{r}$ and produces an estimate, $\hat{\mathbf{v}}$, of the transmitted code sequence $\mathbf{v}$, and, consequently, an estimate, $\hat{u}$, of the message $\mathbf{u}$.

An optimum decoding rule must minimize the probability of a decoding error, $P(\mathcal{E})$. The conditional probability of decoding error, given that $\mathbf{r}$ is received is defined as $P(\mathcal{E}\,|\mathbf{r}) = P(\hat{\mathbf{v}} \neq \mathbf{v})$. It is easily seen that $P(\mathcal{E}) = \sum_{\mathbf{r}} P(\mathcal{E}\,|\mathbf{r})P(\mathbf{r})$. Since $P(\mathbf{r})$ is independent of the decoding rule, the decoder will to minimize the probability of error by minimizing $P(\mathcal{E}\,|\mathbf{r})$ for all $\mathbf{r}$, or equivalently, maximizing $P(\hat{\mathbf{v}} = \mathbf{v} \mid \mathbf{r})$ for all $\mathbf{r}$. Therefore, an optimum decoder must, for a given received sequence $\mathbf{r}$, decide which is the most likely code sequence, $\mathbf{v}$. That is, the decoder must choose the codeword estimate $\hat{\mathbf{v}}$ as the codeword $\mathbf{v}$ which maximizes

$$P(\mathbf{v} \mid \mathbf{r}) = \frac{P(\mathbf{r} \mid \mathbf{v})P(\mathbf{v})}{P(\mathbf{r})}. \tag{1.1}$$

If all codewords are equally likely, then maximizing $P(\mathbf{v} \mid \mathbf{r})$ is equivalent to maximiz-

ing $P(\mathbf{r} \mid \mathbf{v})$. Furthermore, if the channel is a discrete memoryless channel (DMC), $P(\mathbf{r} \mid \mathbf{v}) = \prod_i P(r_i \mid v_i)$, where $\mathbf{v} = (\ldots\ v_{i-1}\ v_i\ v_{i+1}\ \ldots)$ and $\mathbf{r} = (\ldots\ r_{i-1}\ r_i\ r_{i+1}\ \ldots)$. Therefore, minimizing the probability of decoding error is equivalent to maximizing $log \sum_i P(r_i \mid v_i)$. A decoder which maximizes $P(\mathbf{r} \mid \mathbf{v})$ is called a maximum likelihood decoder.

The source decoder then uses $\hat{u}$ to generate an estimate of the original source information. This dissertation focuses on the channel coding operation, and the modulation operation. For that reason, the source coding/decoding is ignored. The primary design criterion considered in this dissertation is error probability. Other factors with affect the cost and performance of the overall communication system include throughput and implementation complexity.

There are three common types of error probabilities used to measure the performance of a channel coding system. The bit error probability,$P_b(\mathcal{E})$, is the expected number of information bit decoding error per decoding information bit. The symbol error probability, $P_s(\mathcal{E})$, is the probability that a channel signal or symbol is decoded incorrectly. and the first event error probability, $P_f(\mathcal{E})$, is the probability that a channel signal or symbol is decodes incorrectly for the first time after a specific signaling interval. The bit error probability is the best measure of the probability that the information transmitted is properly received, but symbol error probability or the first error event probability are often easier to calculate for specific systems. Primarily, bit error probabilities are examined in this work.

1.3 Outline of Dissertation

In certain environments, a discrepancy in the amount of error protection placed on different information bits is desirable. For example, the sign bit and high order bits

of pulse coded modulation (PCM) data are more critical to system performance than the lower order bits [57]. In packet switched networks, the header information requires more error protection than the data;, and in multi-user environments, different users may require more error protection than others. In Adaptive Predictive Coding and Code-book Excited Linear Prediction, the filter coefficients and the codebook choice are more important than the residual information. Systems in which some information is non-essential enhancement information, e.g. embedded coding schemes and high definition television, are also potential application environments [6] [68]. Encoders which provide more than one level of error protection to information information bits are called linear unequal error protection (LUEP) codes. It is also possible to provide unequal error protection the channel bits, but that is not discussed in this work. The purpose of unequal error protection is to provide a higher degree of error protection for the more important bits, without increasing the associated increase in complexity/cost /bandwidth that would occur if the protection were increased for the entire information stream. The research discussed in this work studies the unequal error protection capabilities of convolutional codes.

The dissertation is organized as follows. Chapter 2 discusses error control coding, particularly block codes and convolutional codes. In addition to general convolutional codes, two specific types of convolutional codes, unit memory codes and double memory codes are presented. Some basic concepts that are used later in the dissertation are introduced. Chapter 3 discusses previous work on unequal error protection codes. Unequal error protection block codes and multi-level codes are briefly reviewed. Next, new work on the unequal error protection capabilities of convolutional codes is presented in Chapter 4. The effective free distance vector is defines as a performance parameter. A modified transfer function which allows analysis of unequal error pro-

tection convolutional codes is presented. Upper bounds on the effective free distances are derived. Also, results of code searches are presented and bit error rate simulations for specific encoders are discussed. Chapter 5 presents extensions of the results in Chapter 4 to trellis coded modulation. Chapter 6 contains conclusions and suggestions for further research.

CHAPTER 2

Error Control Coding

2.1 Introduction

The purpose of an error control code is to increase the probability that a message will be reliably transmitted over a noisy channel. This chapter begins by reviewing two common classes of error control codes: block codes and convolutional codes. Both types operate on bit streams emitted from information sources. It is assumed that the information stream is binary, i.e. consists only of 0's and 1's, but results may be generalized to an arbitrary alphabet. Block codes are discussed in Section 2.2. The distinction between a code and an encoder is made, and the minimum distance of a code is discussed. Section 2.3 describes convolutional codes. The section includes a general description of convolutional codes, as well as several examples. In addition, the minimum free distance of a convolutional code is defines. Section 2.4 describes Unit Memory Codes, a special class of convolutional codes. Another special class of convolutional codes, Double Memory Codes, are discussed in Section 2.5.

2.2 Blo ck Codes

A block encoder divides the message sequence into message blocks of k-bits, $\mathbf{u}$, and transmits associated codewords, $\mathbf{v}$, of length n. Figure 2.1 shows a general block codes. There is a one-to-one correspondence between each possible message block and its associated codeword. Because each message block consists of k bits, there are 2^k codewords. The (n, k) binary block code is the set of 2^k n-dimensional binary codewords. Each codeword depends only on the current input, so the system is memoryless. The rate of the code is defined as $R = k/n$. For block codes, the rate is generally expressed as a proper fraction. That is, a code with $k = 2$ and $n = 4$ is called a rate 1/2 code.

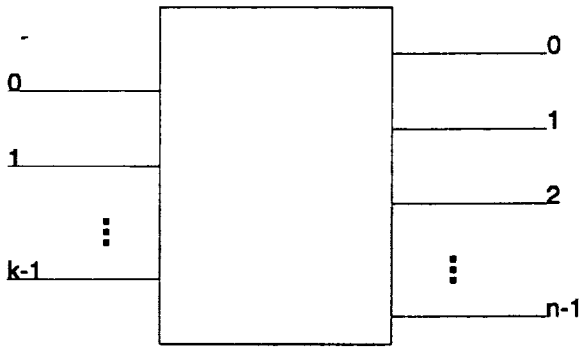


Figure 2.1: General Block Code Encoder

A distinction between a *code* and an *encoder* is made. An encoder is the rule that maps each possible k-bit input to a specific n-bit codeword. That is, an encoder divides an information sequence into blocks of length k, between each k-bit message block and n-bit codeword. An encoder realization of a specific code is not necessarily, and in fact is not normally, unique. For instance, consider the rate 1/2 code, $\mathcal{C} = \{0000, 1010, 0101, 1111\}$. The code is the set of four codewords listed. One possible

encoder realization of the code makes the following associations between the input messages and the codewords.

u	⟶	v
00		0000
01		0101
10		1010
11		1111

(2.1)

Another possible encoder makes the following different associations between the input messages and the codewords.

u	⟶	v
00		0000
01		1010
10		1111
11		0101

(2.2)

Although the encoders generate the same code, or set of n-tuples, the associations between input and outputs differ.

An encoder can be represented by a generator matrix $\mathbf{G}$, which spans the space of the codewords and shows the relationships between the message words and the codewords with the equation

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} \tag{2.3}$$

The first encoder example, given in (2.1) has the generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

while the encoder in (2.2) has the generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}. \tag{2.5}$$

The two encoders, and their corresponding generator matrices, are different realizations of the same code. To rephrase, a code is the set of n-dimensional vectors, while an encoder can be thought of as a set of ordered pairs of k-bit information blocks and n-bit codewords.

A useful performance parameter of a linear binary block code is the minimum distance, d_{min}, between any two codewords. The Hamming distance between two codewords $\mathbf{v}$ and $\mathbf{v}'$, $d_H(\mathbf{v}, \mathbf{v}')$ is the number of corresponding bits of $\mathbf{v}$ and $\mathbf{v}'$ that are different. The minimum distance of a code $\mathcal{C}$ is then defined as the minimum Hamming distance between any two codewords, i.e.

$$d_{min} = \min_{\mathbf{v} \neq \mathbf{v}'} [d_H(\mathbf{v}, \mathbf{v}') : \mathbf{v}, \mathbf{v}' \in \mathcal{C}]. \tag{2.6}$$

For linear block codes, an equivalent definition of d_{min} is the minimum Hamming weight of any codeword, where the Hamming weight of a codeword $\mathbf{v}$ is the total number of 1's in $\mathbf{v}$, and is denoted by $w_H(\mathbf{v})$.

Maximum likelihood decoding of binary block codes chooses the codeword that differs in the fewest number of bit positions from the received n-tuple, $\mathbf{r}$. That is, a maximum likelihood decoder chooses the codeword that is "closest" to the received n-tuple. When maximum likelihood decoding is used, a code with minimum distance d_{min} is guaranteed to detect $(d_{min} - 1)$ bit errors introduced by the transmission channel. The error detection capability stems from the fact that corruption of $(d_{min} - 1)$ or fewer bits of the transmitted codeword will results in an n-tuple that does not belong to the set of codewords. In that case, it is apparent to the receiver that the received n-tuple is corrupted. However, if d_{min} or more bits are changed, it is possible that the received n-tuple is itself a codeword, but not the codeword that was sent. The decoder has no way to tell that this is the case. Similarly, a code with minimum

distance d_{min} is guaranteed to correct $\lfloor (d_{min}-1)/2 \rfloor$ transmission errors, because an error pattern of $(d_{min}-1)$ or fewer errors will not move the received n-tuple to a point closer to a codeword different from the transmitted codeword.

2.3 Convolutional Codes

Elias [12] proposed convolutional codes as an alternative to block codes. Like block codes, convolutional codes separate the information sequence into k-bit message blocks and n-bit codewords. However, with convolutional codes, encoder output depends on both the current and previous message blocks. The k-bit message blocks can be viewed in (at least) two ways: a sequence of k consecutive bits that originated as a sequence from one information source, or one bit from each of k information sources. Either model is appropriate, although one or the other is sometimes more conducive to better understanding for specific applications. The distinction between *code* and *encoder* that was made in Section 2.2 is applicable to convolutional codes.

A general convolutional encoder is shown in Figure 2.2. The k-bit block entering the encoder at time t is $\mathbf{u}_t$, and the n-bit codeword leaving a convolutional encoder at time t is $\mathbf{v}_t$. Let $\mathbf{u}_{t_0,t_1}$ be the entire message sequence entering the encoder from times t_0 to t_1, i.e. $\mathbf{u}_{t_0,t_1} = (\mathbf{u}_{t_0}\mathbf{u}_{t_0+1}, \ldots \mathbf{u}_{t_1})$. Similarly, $\mathbf{v}_{t_0,t_1} = (\mathbf{v}_{t_0}\mathbf{v}_{t_0+1} \ldots \mathbf{v}_{t_1})$ denotes the entire code sequence leaving the encoder during times t_0 to t_1. An (n,k,m) binary convolutional code can be represented by the encoding equation

$$\mathbf{v}_t = \mathbf{u}_t\mathbf{G}_0 + \mathbf{u}_{t-1}\mathbf{G}_1 + \ldots + \mathbf{u}_{t-m}\mathbf{G}_m, \tag{2.7}$$

where the encoding matrices, $\mathbf{G}_i$, $i = 0, 1, \ldots, m$, are $(k \times n)$ binary matrices. The memory distribution vector $\mathbf{M} = (m_0, m_2, \ldots, m_{k-1})$ indicates the size of the shift register on each input line. For example, the first input line has m_0 memory units.

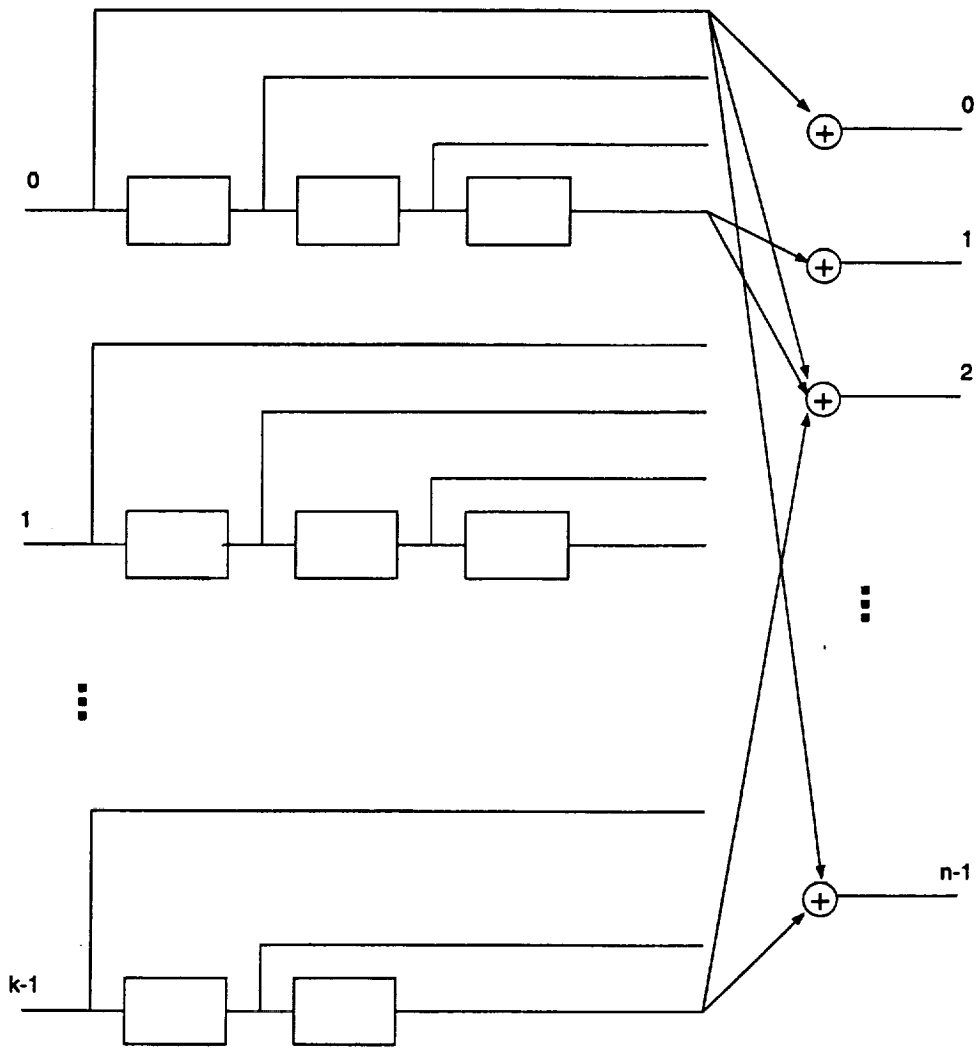


Figure 2.2: A General Convolutional Encoder

The maximum number of memory units on any one line is m. The state complexity, or memory order, of a convolutional code is defined to be the number of state variables, $K = \sum m_i$.

An alternative equation description of a convolutional encoder is

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G}, \tag{2.8}$$

where $\mathbf{u} = (\mathbf{u}_0\mathbf{u}_1 \ldots)$ is the semi-infinite sequence of message blocks, $\mathbf{v} = (\mathbf{v}_0\mathbf{v}_1 \ldots)$ is the semi-infinite sequence of codewords, and $\mathbf{G}$ is the semi-infinite generator matrix formed from the encoder matrices

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_m & \mathbf{0} & \cdots \\ \vdots & & \ddots & & & & \end{bmatrix} \tag{2.9}$$

Figure 2.3 shows a (3,2,2) convolutional encoder with $\mathbf{M} = (1, 2)$, $K = 3$, and encoding matrices

$$\mathbf{G}_0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{2.10}$$

The free distance, d_{free}, of a convolutional code, $\mathcal{C}$, is the minimum Hamming distance between all pairs of code sequences. Formally,

$$d_{free} = \min_{\mathbf{v}_{0,t} \neq \mathbf{v}'_{0,t}} [d_H(\mathbf{v}_{0,t}, \mathbf{v}'_{0,t}) : and\, \mathbf{v}_{0,t}, \mathbf{v}'_{0,t} \in \mathcal{C}]. \tag{2.11}$$

Due to the linearity of binary convolutional codes, the free distance is also the minimum Hamming weight of any non-zero code sequence,

$$d_{free} = \min[w_H(\mathbf{v}_{0,t} \neq \mathbf{0} \ : \ \mathbf{v}_{0,t} \in \mathcal{C})]. \tag{2.12}$$

It is assumed that the first non-zero input to the encoder arrives at time 0. The free distance can be difficult to determine because code sequences may be of infinite

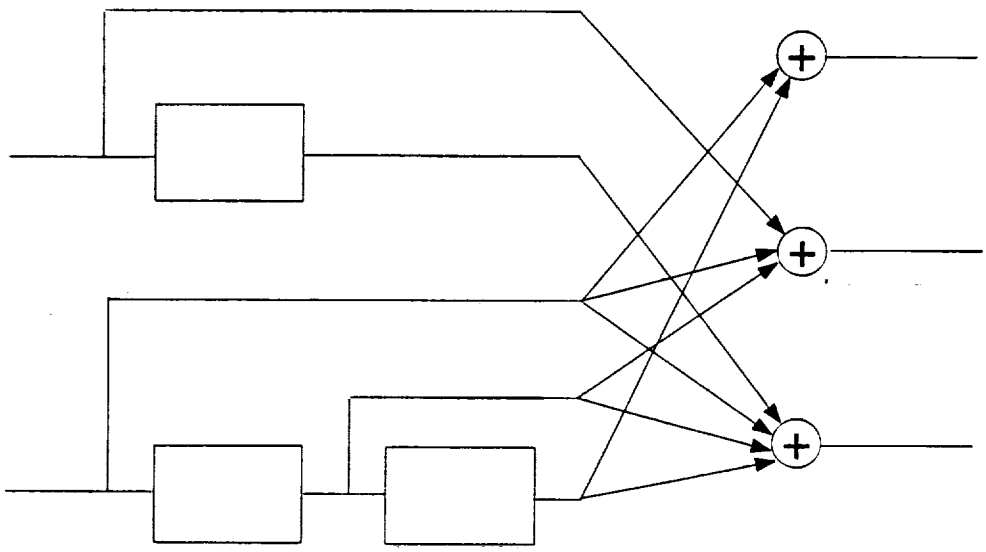


Figure 2.3: A Specific $(3, 2, 2)$ Convolutional encoder

length. Often, bounds are calculated for the maximum achievable free distance (upper bounds) [9, 30, 19, 17] so that the performance of a particular code can be compared to the best theoretical performance.

Convolutional encoders are occasionally described by a transfer function. The concept of the transfer function of a convolutional encoder is used later in the dissertation, and is reviewed here. It is assumed that the reader is familiar with the method of determining a transfer function from an augmented state diagram using Mason's gain formula [32], or some other algorithm [66], [54], and [8].

The two-variable transfer function has the form

$$T(X,Y) = \sum_{d=d_{free}}^{\infty} \sum_{b=1}^{\infty} A_{b,d}.X^d.Y^b, \tag{2.13}$$

where $A_{b,d}$ is the number of code sequences with Hamming weight d that have corresponding (input) message blocks with Hamming weight b. The average bit error

probability for a specific transfer function is bounded by

$$P_b(E) < \frac{1}{k} \cdot \sum_d B_d . P_d, \tag{2.14}$$

where $B_d = \sum_b b.A_{b,d}$ is the total number of non-zero information bits associated with all codewords of weight d, and $P_d = \left(\sqrt{4p(1-p)}\right)^d$. For the sake of simplicity, we have assumed a binary symmetric channel with crossover probability, p.

Examining the above equation, it is seen that the dominating term in the upper bound on the bit error probability is the term with the lowest value of d, which happens to be the free distance. This explains the use of the free distance as a performance measure for convolutional codes.

Typically, the Viterbi algorithm [32], [67], [38], [14], [15] is used to decode convolutional codes with relatively small memory orders. Viterbi decoding is a maximum likelihood decoding algorithm, i.e. it selects the code word that minimized the probability of decoding error, assuming all codewords are equally likely. The complexity of the Viterbi algorithm increases dramatically as the number of states in the trellis increase. For that reason, sequential decoding, [69] [13] [73] [22], is generally used for encoders with $K \geq 10$.

2.4 Unit Memory Codes

An interesting class of binary convolutional codes are unit memory codes [31] [29] [25]. A unit memory code (UMC) is a binary convolutional code with memory $m = 1$ and multiple input lines, i.e. $k > 1$. Therefore, the encoding equation of a UMC is $\mathbf{v}_t = \mathbf{u}_t\mathbf{G}_0 + \mathbf{u}_{t-1}\mathbf{G}_1$. An (n_o, k_o, m) convolutional code with $(k_o \times n_o)$ encoding matrices $\mathbf{g}_0, \mathbf{g}_1, \cdots, \mathbf{g}_m$, is equivalent to the $(n = mn_o, k = mk_o, 1)$ UMC which has the two $(mk_o \times mn_o)$ binary encoding matrices

$$\mathbf{G}_0 = \begin{bmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_{m-1} \\ 0 & \mathbf{g}_0 & & \mathbf{g}_{m-2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & \mathbf{g}_0 \end{bmatrix} \quad \mathbf{G}_1 = \begin{bmatrix} \mathbf{g}_m & 0 & \cdots & 0 \\ \mathbf{g}_{m-1} & \mathbf{g}_m & & 0 \\ \vdots & & \ddots & \vdots \\ \mathbf{g}_1 & \cdots & & \mathbf{g}_m \end{bmatrix} \tag{2.15}$$

The two encoders are equivalent in the sense that they generate identical outputs when operating on identical input sequences. This is easily verified by comparing the semi-infinite generator matrices of the two encoders. For the basic encoder, the semi-infinite generator matrix is

$$\mathbf{G}_{basic} = \begin{bmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \cdots & \mathbf{g}_{m-1} & 0 & \cdots & \cdots \\ 0 & \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \cdots & \mathbf{g}_{m-1} & 0 & \cdots \\ 0 & 0 & \ddots & & & & \ddots & \end{bmatrix}, \tag{2.16}$$

where each entry is a $(k_o \times n_o)$ sub-matrix. On the other hand, the semi-infinite generator matrix for the unit memory code is

$$\mathbf{G}_{UMC} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & 0 & \cdots & \\ 0 & \mathbf{G}_0 & \mathbf{G}_1 & 0 & \cdots \\ 0 & 0 & \ddots & \ddots & \end{bmatrix}, \tag{2.17}$$

with $(mk_o \times mn_o)$ sub-matrices. Replacing $\mathbf{G}_0$ and $\mathbf{G}_1$ with the expressions from 2.15,

$$\mathbf{G}_{UMC} = \begin{bmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_{m-1} & \mathbf{g}_m & 0 & \cdots & 0 & & & & & & \\ 0 & \mathbf{g}_0 & & \mathbf{g}_{m-2} & \mathbf{g}_{m-1} & \mathbf{g}_m & & 0 & & & 0 & & \cdots & \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & & & & & & \\ 0 & \cdots & & \mathbf{g}_0 & \mathbf{g}_1 & \cdots & & \mathbf{g}_m & & & & & & \\ & & & & \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_{m-1} & \mathbf{g}_m & 0 & \cdots & 0 & & \\ & & & & 0 & \mathbf{g}_0 & & \mathbf{g}_{m-2} & \mathbf{g}_{m-1} & \mathbf{g}_m & & 0 & & \\ & & 0 & & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & 0 & \cdots \\ & & & & 0 & \cdots & & \mathbf{g}_0 & \mathbf{g}_1 & \cdots & & \mathbf{g}_m & & \\ & & 0 & & & & 0 & & & & \ddots & & \ddots & \end{bmatrix}. \tag{2.18}$$

Comparing Equations (2.16) and (2.18) confirms Equation (2.15.

As an example, consider the (2,1,3) basic convolutional encoder with encoder matrices $\mathbf{g}_0 = [1\ 1]$, $\mathbf{g}_1 = [1\ 1]$, $\mathbf{g}_2 = [0\ 1]$, $and\, \mathbf{g}_3 = [1\ 1]$. When the input to the encoder is the message sequence $(1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ \cdots)$, the code sequence is

(11 11 10 11 01 01 10 00 11$\cdots$). The associated (6,3,1) unit memory encoder has encoding matrices

$$\mathbf{G}_0 = \begin{bmatrix} 11 & 11 & 01 \\ 00 & 11 & 11 \\ 00 & 00 & 11 \end{bmatrix} \quad and \quad \mathbf{G}_1 = \begin{bmatrix} 11 & 00 & 00 \\ 01 & 11 & 00 \\ 11 & 01 & 11 \end{bmatrix} \tag{2.19}$$

When the input to the unit memory encoder is the message sequence (101 110 010 $\cdots$), the code sequence is (111110 110101 100011$\cdots$).

Let n_o and k_o be relatively prime. The class of (n_o, k_o, m) convolutional codes will hereafter be called basic convolutional codes. The encoding matrices of a UMC that has been converted from a basic code must adhere to the form in Equation 2.15. However, the encoder matrices for a general UMC are not under the same restriction. That is, general UMC matrices are not block triangular matrices with constant diagonals. It follows that UMCs are a larger class of codes than basic convolutional codes. In addition, since every basic convolutional code can be converted to a UMC of the same rate and state complexity, the optimal UMC has a free distance at least as large as the free distance of the optimal basic code.

Lee developed an upper bound on the free distance of a $(n, k, 1)$ UMC, which is now presented. As previously stated, the free distance, d_{free}, of a convolutional code is the minimum Hamming distance between all pairs of code sequences that are associated with input sequences that differ in at least one message block. It can be assumed without loss of generality that one of the code sequences in the comparison is always the all-zero sequence, and that the first non-zero portion of the other code sequence in the comparison occurs at time 0. So, the free distance is the minimum Hamming weight among all code sequences generated by input sequences that are non-zero in the message block at time 0, i.e.

$$d_{free} = \min_{\mathbf{u}_0 \neq 0} [w_H(\mathbf{v}_{0,t}) \; for \; all \; t]. \tag{2.20}$$

When the *only* non-zero portion of the information sequence is $\mathbf{u}_0$, then the only non-zero output is occurs at times 0 and 1, and is $\mathbf{v}_{0,1} = \mathbf{u}_0[\mathbf{G}_0\mathbf{G}_1]$. The set of all such $\mathbf{u}_0$'s and $\mathbf{v}_{0,1}$'s forms a $(2n, k)$ block code. Therefore, at least one code sequence belonging to the UMC has a Hamming weight less than or equal to the minimum distance of the optimal $(2n, k)$ block code. The minimum distance of the optimal $(2n, k)$ block code will be denoted $d_{opt}(2n, k)$. It follows that the optimal free distance of the $(n, k, 1)$ UMC is upperbounded by $d_{opt}(2n, k)$. The optimal block code minimum distances are tabulated in [18, 65]. In several cases, the UMC upper bound is larger than the free distance attained by the optimal basic codes of the same rate and state complexity. Lee conducted an exhaustive search for the optimal UMCs and found several UMCs with a larger free distance than the optimal basic codes of the same rate and state complexity. Lee's results are shown in Table 2.4.

(n, k)	UMC upper bound	optimal UMC	optimal basic
(4,2)	5	5	5
(6,3)	6	6	6
(8,4)	8	8	7
(10,5)	9	9	8
(12,6)	10	10	10
(6,2)	8	8	8
(9,3)	10	10	10
(12,4)	12	12	12
(15,5)	15	15	13
(18,6)	16	16	15
(8,2)	10	10	10
(12,3)	13	13	13
(16,4)	16	16	16
(20,5)	20	20	18
(24,6)	24	24	20

Table 2.1: Optimal Unit Memory Codes

A drawback associated with UMCs is a possible increase in Viterbi decoding com-

plexity when compared to the complexity of basic convolutional codes. The increase in complexity is a result of the increased number of branches leaving each state of a trellis representation of the encoder. The number of branches leaving each state is called the branch complexity. A (n_o, k_o, m) basic convolutional code has a branch complexity of 2^{k_o}, while a $(mn_o, mk_o, 1)$ UMC has a branch complexity of 2^{mk_o}. Therefore, the cost of a (potential) increase in the free distance is an increase in the branch complexity. The state complexities of the (n_o, k_o, m) basic convolutional code and $(mn_o, mk_o, 1)$ UMC are both 2^{mk_o}. While state complexity is the primary measure of Viterbi decoding complexity, the branch complexity effects are not negligible. However, the branch complexity *per decoded information bit* of the basic encoder and UMC are identical.

The results obtained by Lee for UMCs led to the study of Double Memory Codes [35], which are discussed in the next section.

2.5 Double Memory Codes

This section describes double memory convolutional codes and presents an upper bound on their free distance [35]. It is shown that the free distance upper bound can be larger than the free distances previously attained by codes with relatively prime k and n. For certain rates, the bound is as large as the upper bound for unit memory codes. A double memory code which has a free distance larger than the free distance of the optimal basic code is briefly described. Double memory codes have lower branch complexities then the corresponding unit memory codes.

A double memory code (DMC) is a convolutional code with $m = 2$ that can be described by $\mathbf{v}_t = \mathbf{u}_t\mathbf{G}_0 + \mathbf{u}_{t-1}\mathbf{G}_1 + \mathbf{u}_{t-2}\mathbf{G}_2$. It is assumed unless explicitly stated otherwise that the memory allotted to every input line is 2. It can be seen that

any $(n_o, k_o, 2m)$ convolutional code with $(k_o \times n_o)$ encoding matrices $\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_{2m}$ is equivalent to the $(mn_o, mk_o, 2)$ DMC with $(mk_o \times mn_o)$ encoding matrices

$$\mathbf{G}_0 = \begin{bmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_{m-1} \\ 0 & \mathbf{g}_0 & & \mathbf{g}_{m-2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & \mathbf{g}_0 \end{bmatrix} \quad \mathbf{G}_1 = \begin{bmatrix} \mathbf{g}_m & \mathbf{g}_{m+1} & \cdots & \mathbf{g}_{2m-1} \\ \mathbf{g}_{m-1} & \mathbf{g}_m & & \mathbf{g}_{2m-2} \\ \vdots & & \ddots & \vdots \\ \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_m \end{bmatrix} \tag{2.21}$$

$$\mathbf{G}_2 = \begin{bmatrix} \mathbf{g}_{2m} & 0 & \cdots & 0 \\ \mathbf{g}_{2m-1} & \mathbf{g}_{2m} & & 0 \\ \vdots & & \ddots & \vdots \\ \mathbf{g}_{m+1} & \cdots & & \mathbf{g}_{2m} \end{bmatrix}.$$

It should be noted that the class of double memory codes, although smaller than the class of UMCs, is larger than the class of basic convolutional codes. This results from the fact that every basic code can be converted to an equivalent DMC, and every DMC can be converted to an equivalent UMC, but the reverse relationships (UMC to DMC, and DMC to basic code) do not hold for all UMCs or DMCs.

An upper bound on the free distance may be developed in the same way that the bound for UMCs was developed. The free distance is again recognized as the minimum weight vector resulting from an information sequence that is non-zero at time 0. For an $(n, k, 2)$ DMC, the set of such $\mathbf{u}_0$'s and their associated outputs can be considered as a $(3n, k)$ block code with $\mathbf{v}_{0,2} = \mathbf{u}_0[\mathbf{G}_0\mathbf{G}_1\mathbf{G}_2]$. Accordingly, the optimal d_{free} is upper bounded by the highest attainable minimum distance of a $(3n, k)$ block code.

The free distance bounds for $k > 1$ are shown in Table 2.5. As for UMCs, the free distance of the optimal basic code with the appropriate rate and state complexity

(n,k)	K	DMC upper bound	UMC upper bound	optimal UMC	optimal basic
(4,2)	4	8	8	8	7
(6,3)	6	10	10	10	10
(8,4)	8	12	14		12
(10,5)	10	15	16		14
(6,2)	4	12	12	12	12
(9,3)	6	15	16	16	15
(12,4)	8	18	22	22	18
(15,5)	10	22	27		22
(18,6)	12	26	32		24

Table 2.2: Upper Bounds for Double Memory Codes

provides a lower bound on the optimal DMC free distance. The table lists the k and n used for the DMC. The DMC at a given rate is compared to the optimal (n_o, k_o) basic convolutional code with the same memory order, K, and the optimal $(2n, 2k)$ UMC with the same memory order. In some cases, the block code upper bound for the DMC free distance is larger than the free distance achieved by the optimal basic code of the same rate and state complexity. In particular, consider the rate 2/4, 5/10, and 6/18 bounds. However, the existence of a DMC that attains the block code upper bound is not guaranteed. A rate 2/4 DMC encoder with free distance 8 was found. The encoding matrices for that encoder are:

$$\mathbf{G}_0 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \text{ and } \mathbf{G}_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{2.22}$$

Double memory codes are interesting for several reasons. First, the free distance performance of a DMC is at least as high as the free distance of the basic code of the same rate and state complexity. in some cases, the upper bound for DMCs are

equal to the upper bound for the UMC of the same states complexity. In addition, as with UMCs, the form of DMCs is byte-oriented, which is attractive for use in concatenated coding systems. Finally, partial double memory codes, in which fewer than two memory units are allocated to a portion of the input lines, show potential as unequal error protection codes, due to the unbalanced distribution of memory units.

2.6 Complexity Concerns

In general, when decoding a (n, k, m) convolutional code with state complexity K, a Viterbi decoder must have a $2^n \times 2^n$ lookup table, and performs $2^K \cdot 2^k$ additions and $2^K \cdot (2^k - 1)$ binary comparisons to decode k information bits. The Viterbi decoding complexities of a $(mn_o, mk_o, 1)$ UMC, $(\frac{m}{2}n_o, \frac{m}{2}k_o, 2)$ DMC, and (n_o, k_o, m) basic convolutional codes are compared in this section. In the example that follows, $n_o = 2$, $k_o = 1$, and $m = 4$. The free distance for the optimal $(2, 1, 4)$ convolutional code is 7, while the free distance of the optimal $(8, 4, 1)$ UMC and the $(4, 2, 2)$ DMC is 8.

To decode 4 information bits, the Viterbi decoding for the UMC requires a 256×256 lookup table, and performs $16 \cdot 16 = 256$ additions and $16 \cdot 15 = 240$ comparisons. On the other hand, the Viterbi decoder for the DMC requires a 16×16 lookup table, and $16 \cdot 4 = 64$ additions and $16 \cdot 3 = 48$ comparisons to decode 2 information bits. Finally, to decode 1 information bit, the Viterbi decoding for the basic convolutional code performs $16 \cdot 2 = 32$ additions and $16 \cdot 1 = 16$ comparisons, and uses a 4×4 lookup table.

The requirements *per decoded information bit* for the three codes are shown in Table 2.6. The complexity of the basic convolutional code is obviously the smallest, while the complexity of the UMC is the largest. For this example, the DMC provides

a larger free distance than the basic code, with a reduced decoding complexity when compared to the UMC. In general, the tradeoffs of performance vs. complexity are dependent on the specific codes of interest.

	UMC	*DMC*	*basic*
lookup table:	256×256	16×16	4×4
additions:	64	32	32
comparisons:	60	24	16

Table 2.3: Decoding Complexity Comparison

2.7 Summary

The increase in free distance exhibited by Unit Memory Codes when compared to the appropriate basic code led to the hope that, rather than increasing the free distance and error protection encountered by all information bits, the error protection for one input bit position could be significantly increased. That is, the additional capabilities of the unit memory code could be focus on a specific information bit position. The next chapter introduces unequal error protection coding, and reviews previous work done in the area. The subsequent chapter then discusses unequal error protection with convolutional codes.

CHAPTER 3

Unequal Error Protection Coding

3.1 Introduction

In certain environments, a discrepancy in the amount of error protection placed on different information bits is desirable. For example, the sign bit and high order bits of pulse coded modulation (PCM) data are more critical to system performance than the lower order bits [57]. In packet switched networks, the header information requires more error protection than the data, and in multi-user environments, different users may require more error protection than others. In Adaptive Predictive Coding and Code-book Excited Linear Prediction, the filter coefficients and the codebook choice are more important than the residual information. Systems in which some information is non-essential enhancement information, e.g. embedded coding schemes and high definition television, are also potential application environments. Encoders which provide more than one level of error protection to information information bits are called linear unequal error protection (UEP) codes. It is also possible to provide unequal error protection the channel bits, but that is not discussed in this work. The purpose of unequal error protection is to provide a higher degree of error protection for the more important bits, without increasing the associated increase in complexity/cost/bandwidth that would occur if the protection were increased for the

entire information stream.

This chapter briefly reviews unequal error protection approaches that have been published in the past. The bulk of unequal error protection (UEP) codes have been block codes. The general concepts of UEP block codes are presented in Section 3.2 Multi-level coding with unequal error protection is discussed in Section 3.3.

3.2 UEP Block Codes

Wolf and Manisck introduced unequal error protection block codes [34]. Since then new results have been presented for classes of codes that include nonsystematic cyclic UEP codes, codes derived form difference set, iterative and concatenated designs of UEP codes, cyclic code classes, and linear a UEP codes derived form shorter codes [33] [51] [63][11] [64].

An important concept in evaluating UEP block codes is the separation vector, first defined in [11]. For a linear (n, k) binary code $\mathcal{C}$, the *separation vector* $\mathbf{s}(\mathbf{G}) = [s_1(\mathbf{G}), \ldots, s_k(\mathbf{G})]$ with respect to encoder matrix $\mathbf{G}$ of $\mathcal{C}$, is defined as

$$s_i(\mathbf{G}) = \min w_H(\mathbf{uG}) \; for \; all \; \mathbf{u} \; such \; that \; u_i \neq 0, \tag{3.1}$$

where u_i is the i^{th} bit of the k-bit message u. The definition of the separation vector immediately leads to the following result. For a linear (n, k) code with encoding matrix G, complete maximum likelihood decoding guarantees correct decoding of the i^{th} information bit whenever the error pattern has a Hamming weight less than or equal to $\lfloor (s_i(\mathbf{G})-1)/2 \rfloor$. If a linear code $\mathcal{C}$ has an encoding matrix $\mathbf{G}$ with a separation vector for which components are not mutually equal, then the code is called a linear unequal error protection code. It is possible to order the separation vector so that the components are non-increasing, simply by reordering the rows of $\mathbf{G}$. Every code

has an optimal generator matrix $\mathbf{G}^*$, whose separation vector is componentwise larger than or equal to the separation vector of any other generator matrix of the code The separation vector of a linear code is defined as the separation vector of the optimal generator matrix of the code.

It is easily seen that the minimum distance of the code is equal to the smallest component of the separation vector, i.e. $d_{min} = \min_i [s_i]$. It should be noted that the separation vector is a measure associated with a particular encoder realization of a code

Van Gils [63] [64] defined the minimal length necessary to achieve a specific separation vector for a given rate as a basic parameter of UEP block codes. He developed several bounds on that parameter. He first defined $n(\mathbf{s})$ as the length of the shortest linear binary block code dimension k with a separation vector of at least $\mathbf{s}$.

An $(n(\mathbf{s}), k, \mathbf{s}]$ code is called optimal if an $(n(\mathbf{s}), k, \mathbf{t}]$ code with $\mathbf{t} \geq \mathbf{s}, \quad \mathbf{t} \neq \mathbf{s}$, does not exist. The inequality between the two vectors indicates a componentwise comparison. For example, if $\mathbf{s} = (2, 3, 4)$, $\mathbf{t} = (3, 3, 4)$, and $\mathbf{w} = (2, 5, 5)$, then $\mathbf{t} > \mathbf{s}$, and $\mathbf{w} > \mathbf{s}$. The relationship between $\mathbf{w}$ and $\mathbf{t}$ is best described by the equations $\mathbf{w} \neq \mathbf{t}$, $\mathbf{w} \not> \mathbf{t}$, and $\mathbf{w} \not< \mathbf{t}$.

An upper bound on $n(\mathbf{s})$ is

$$n(\mathbf{s}) \leq \sum_{i=1}^{k} s_i. \tag{3.2}$$

The proof of 3.2 is straightforward. For $l = 1, \cdots, k$, let $\mathbf{G}_l$ be the $(1 \times s_l)$ matrix $[1, 1, \cdots, 1]$. Then, each matrix $\mathbf{G}_l$ has minimum distance s_l. Let $\mathbf{G}$ be defined as the $(k \times n(\mathbf{s})$ matrix $\mathbf{G} = \mathrm{diag}[\mathbf{G}_1 \mathbf{G}_2 \cdots \mathbf{G}_k]$. Then $\mathbf{G}$ has separation vector $\mathbf{s} = (s_1, s_2, \cdots, s_k)$, and $n(\mathbf{s}) \leq \sum_{i=1}^{k} s_i$.

An interesting lower bound on $n(\mathbf{s})$ discussed in [63] is

$$n(\mathbf{s}) \geq \sum_{i=1}^{k} \lceil s_i/2^{i-1} \rceil, \tag{3.3}$$

where the components of $\mathbf{s}$ are ordered so that they are nonincreasing.

The proof of 3.3 follows. Let $\mathcal{C}$ be a linear $(n = n(\mathbf{s}), k, \mathbf{s})$ binary code, and let $\mathbf{G}$ be a minimal weight generator matrix for $\mathcal{C}$. It can be shown that the first row of $\mathbf{G}$, denoted by $\mathbf{r}_1$, has Hamming weight s_1. Without loss of generality, the first s_1 columns of $\mathbf{G}$ have a 1 in the first row. Deleting the first s_1 columns and the first row of $\mathbf{G}$ yields a $(k-1) \times (n-s_1)$ binary matrix, $\hat{\mathbf{G}}$, with rank $k-1$. Hence, $\hat{\mathbf{G}}$ is a generator matrix of an $(n-s_1, k-1)$ code with separation vector, $\hat{\mathbf{s}} = (\hat{s_2}, \cdots, \hat{s_k})$. Let $j \in \{2, \cdots, k\}$, and let $\mathbf{u}$ be the message block that is non-zero only in the j^{th} bit position. Then $\mathbf{c} = \mathbf{uG} = (\mathbf{c_1}|\mathbf{c_2})$, where $\mathbf{c_1}$ has length s_1 and $w_H(\mathbf{c_2}) = \hat{s}_j$. By definition of the separation vector,

$$w_H(\mathbf{c_1}) + \hat{s}_j \geq s_j. \tag{3.4}$$

Furthermore, at least $w_H(\mathbf{c_1})$ components of $\mathbf{c_1}$ equal 1, so

$$w_H(\mathrm{row}(i) - \mathbf{c}) \leq s_1 - w_H(\mathbf{c_1}) + \hat{s}_j. \tag{3.5}$$

However,

$$w_H(\mathrm{row}(i) - \mathbf{c}) \geq \max(s_1, s_j), \tag{3.6}$$

so

$$n(\mathbf{s})(s_1, \cdots, s_k) \geq s_1 + n(\lceil \frac{s_2}{2} \rceil, \cdots, \lceil \frac{s_k}{2} \rceil). \tag{3.7}$$

Repeating the process results in the bound given in 3.3.

Table 3.2 presents some of the separation vectors that were shown in [63]. The optimal minimum distance for each rate (n, k) is shown in the column labeled d_{opt}, and the achieved separation vectors are shown in the column labeled $\mathbf{s}$.

n	k	d_{opt}	s
4	2	2	(3,2)
5	2	3	(4,2)
5	3	3	(3,2,2)
6	2	4	(5,2)
6	3	3	(4,2,2)
8	2	5	(7,2), (6,4)
8	3	4	(6,2,2), (5,4,4)
8	4	4	(5,2,2,2)

Table 3.1: Selected UEP Block Code Results

In addition to, or, in some cases, is lieu of the separation vector, several authors use a mean distortion vector as a measure of LUEP block code performance [51]. The design criterion is the overall mean square error between the numerical representations of the decoded k-bit sequence and the original k-bit information sequence . However, mean square error is dependent on the method of numerical representation, and not as closely related to the bit error rate. The method is useful for specific applications but is not used in this dissertation.

3.3 Multi-level Coding

Multi-level coding is another method used to achieve UEP [28, 50, 6]. The information sequences that require more error protection are assigned to the more powerful subcoders in the multi-level coding system. Multi-level codes have the disadvantage of high decoder complexity. The advantage of the method lies in the ease of achieving large disparity in the protection provided.

The first technique proposed in [6] is a time sharing generalization in which the code specifies the multiplexing rule that is to be chosen. That is, two different code signal constellations are possible, and the choice of the constellation is dependent on the importance of the data. The second UEP technique proposed in that paper combined

multi-level coding and non-standard set partitioning. In a nonstandard partitions, less important data may be assigned to the points within a subset, thereby allowing the minimum intr-subset distance to be smaller than the standard set partitioning. The important data may then have an increased minimum Euclidean distance.

3.4 Summary

In this chapter, unequal error protection coding was introduced. In particular, binary linear UEP block codes were discussed. The separation vector was defined, and some bounds on the necessary code length for a given separation vector were presented. The next chapter discusses unequal error protection with convolutional codes.

CHAPTER 4

Unequal Error Protection with Convolutional Codes

4.1 Introduction

In this chapter, we examine the unequal error protection capabilities of convolutional codes by presenting classes of convolutional codes which satisfy the basic property of LUEP codes, that is, provide unequal error protection for each *input information digit.* The LUEP property is satisfied for certain rates $R = k/n$, where $k \geq 2$ and k and n are not necessarily relatively prime.

In contrast with the UEP block codes discussed in Chapter 3, the LUEP convolutional codes presented in this dissertation lack algebraic structure. For that reason, good codes are found by a search procedure. Optimal decoding for UEP convolutional codes remains the Viterbi decoding for short-constraint-length, or sequential decoding for long constraint-length.

This chapter is organized as follows. In Section 4.2, the effective free distance vector is presented. Section 4.3 presents a modified transfer function for convolutional encoders from which the unequal error protection capabilities of a code can be calculated. Several bounds on the unequal error protection capabilities of convolutional encoders are derived and discussed in Section 4.4. Finally, Section 4.5 presents the

results of searches for UEP codes.

4.2 The Effective Free Distance Vector

It is convenient to define an effective free distance vector, $\mathbf{d}$, as an alternative to the free distance as a primary performance parameter for UEP convolutional encoders. The effective free distance vector is similar to the separation vector concept of linear UEP block codes discussed in the previous chapter. Similar vectors have been proposed in [43], [40], [36], [37], [35], [49].

For a given (n, k), $k \neq 1$ convolutional encoder, $\mathbf{G}$, the effective free distance vector is defined as the k-dimensional vector

$$\mathbf{d} = (d_0, d_1, \cdots, d_{k-1}) \tag{4.1}$$

where d_j, the j^{th} effective free distance, is the lowest Hamming weight among all code sequences that are generated by input sequences with at least one "1 " in the j^{th} position, i.e.,

$$d_j = \min_{\mathbf{v}_{[0,t+m]}} \left\{ w_H(\mathbf{v}_{[0,t+m]}) : \mathbf{v}_{[0,t+m]} = \mathbf{u}_{[0,t]}.\mathbf{G}, \forall t \right\} \tag{4.2}$$

where the j^{th} bit of $\mathbf{u}_s$ is non-zero for some $s \in [0, t]$, and $\mathbf{u}_s = 0$ for $s > t$. If an effective free distance vector $\mathbf{d}$ corresponding to a convolutional encoder with generator matrix $\mathbf{G}$ is such that its components are not mutually equal, then we call this encoder a linear unequal error protection convolutional code. It is evident from the previous definitions that the free distance of the code equals

$$d_{free} = \min \{d_0, d_1, \cdots, d_{k-1}\} \tag{4.3}$$

Thus, $d_i \geq d_{free}$.

From the above definition, we have the following error-correcting capability of a convolutional code when used in a binary-input symmetric-output channel. An (n, k)

convolutional encoder, with generator matrix $\mathbf{G}$ and Viterbi or Sequential decoding algorithm, guarantees that the j^{th} input information digit is decoded correctly whenever the error pattern has Hamming distance less than or equal to $\lfloor (d_j - 1)/2 \rfloor$.

Note that for a given memory distribution vector $\mathbf{M} = (m_1, m_2, ..., m_k)$, and a given set of encoder matrices $\mathbf{G}_i$, each permutation of the k components of $\mathbf{M}$ and the corresponding rows of each $\mathbf{G}_i$ leads to effective free distance vectors with components that are permutations of the set $\{d_0, d_1, \cdots, d_{k-1}\}$. Let $\tilde{\mathbf{d}}(\mathcal{C}) = \left\{\tilde{d}_0, \tilde{d}_1, ..., \tilde{d}_{k-1}\right\}$ denote the vector formed by ordering the components of $\mathbf{d}$ in a nondecreasing order. An (n, k, m) UEP convolutional encoder, $\mathbf{G}$, with state complexity K and ordered effective free distance vector $\tilde{\mathbf{d}}(\mathbf{G})$ is optimum if there exists no other (n, k, m) UEP convolutional encoder,$\mathbf{G}'$ with state complexity K and ordered effective free distance vector $\tilde{\mathbf{d}}(\mathbf{G}')$ which is larger (componentwise) than $\tilde{\mathbf{d}}(\mathbf{G})$.

As an example, consider the $(3, 2)$ convolutional encoder with $\mathbf{M} = (1, 1)$, and submatrices

$$\mathbf{G}_0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}; \mathbf{G}_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

The encoder diagram and the associated trellis are shown in Figs. 4.1, and 4.2, respectively.

Examining the possible paths through the trellis reveals that the first effective free distance, d_0, is 3, and the second effective free distance, d_1, is 4. That is, $\mathbf{d} = (3, 4)$. The non-zero path through the trellis with weight 3 is shown by dotted lines. The non-zero paths with weight 4 are shown with dark lines. All other paths have weight greater than 4. It can be seen that the weight 3 path is created by an input vector

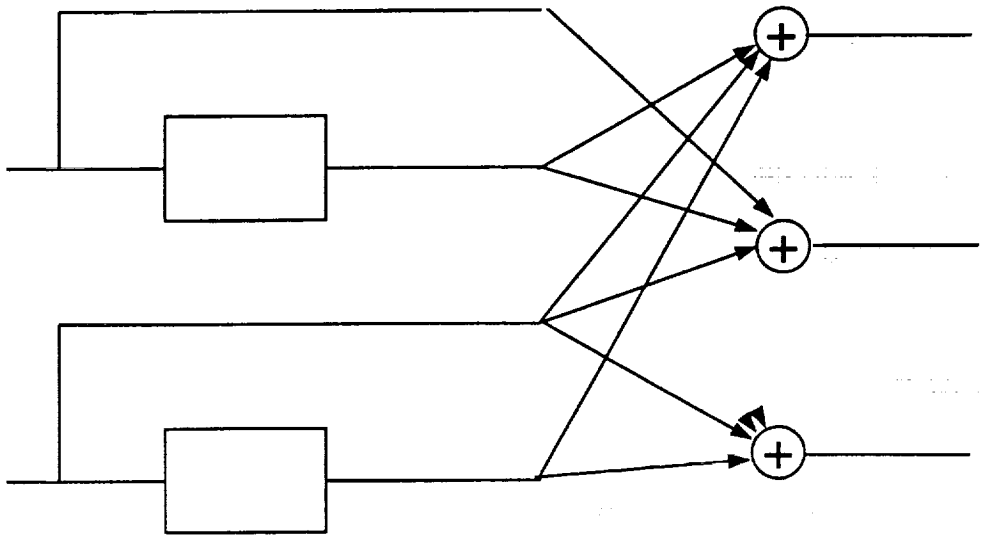


Figure 4.1: A specific $(3, 2)$ encoder

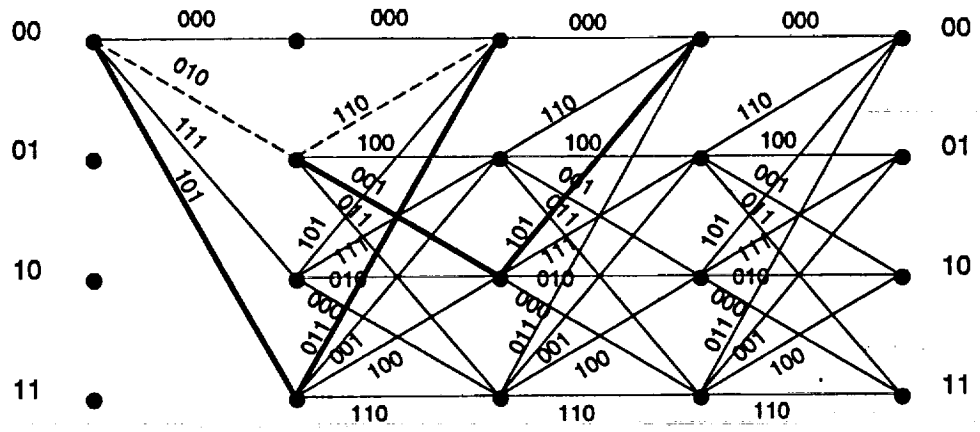


Figure 4.2: Trellis for a specific $(3, 2)$ encoder

sequence that is non-zero only in the first input bit position, (01, 00). When the input sequences are non-zero in the second position, the minimum weight of any path is 4.

It is important to recognize that the first "1" in the j^{th} position does not necessarily occur at time zero. For instance, the input sequence (10 01 00) is one of the sequences that must be considered when determining the second effective free distance, d_1, of an encoder with two input lines ($k = 2$), and $\mathbf{M} = (1, 1)$. On the other hand, the input sequence (10 10 00) does not affect d_1.

4.3 A Modified Transfer Function

State diagram analysis has long been used to determine the transfer functions of low complexity (n, k, m) convolutional encoders. The transfer function, in turn, is then manipulated to determine the free distance, event error probability, and bit error probability of the encoder. The bit error probability derived from the standard transfer function is the probability that an input bit is decoded incorrectly. However, the error probability which is relevant for unequal error protection codes is the probability of bit error at each specific bit position.

This section presents a modified transfer function analysis for time-invariant convolutional encoders that yields the individual bit error probability for any specified input bit position. First, standard transfer function analysis will be briefly reviewed. Then, the modified transfer function will be described and illustrated with an example. An upper bound on the average bit error probability for a specific input bit position is presented. Then the unequal error protection capabilities of several codes are presented and discussed.

Recall that the two-variable transfer function of a code has the form

$$T(X,Y) = \sum_{d=d_{free}}^{\infty} \sum_{b=1}^{\infty} A_{b,d}\, X^d\, Y^b, \tag{4.4}$$

where $A_{b,d}$ is the number of code sequences with Hamming weight d that have corresponding (input) message sequences with Hamming weight b. The average bit error probability for a specific transfer function is bounded by

$$P_b(E) < \frac{1}{k} \sum_{d} B_d\, P_d, \tag{4.5}$$

where $B_d = \sum_b b\, A_{b,d}$ is the total number of non-zero information bits associated with all codewords of weight d, and $P_d = \left(\sqrt{4p(1-p)}\right)^d$. For the sake of simplicity, we assume a binary symmetric channel with crossover probability p.

When the individual bit error probability is desired for each of the k input positions, then the split-state diagram must be modified before Mason's formula is applied. Each branch label has the new form

$$X^i Y_0^{j_0} Y_1^{j_1} \cdots Y_{k-1}^{j_{k-1}}, \tag{4.6}$$

where j_q is equal to the input bit in the q^{th} position, and i is the Hamming weight of the branch output. Obviously, the sum of the j_q's is the Hamming weight of the input message block. The modified transfer function is then calculated in the same way described in [32]. The resulting modified transfer function is

$$T(X, Y_0, Y_1, \cdots, Y_{k-1}) = \sum_{d=d_{free}}^{\infty} \sum_{j=0}^{j_d} C_{d,j} X^d Y_0^{b_{0,j}} Y_1^{b_{1,j}} \cdots Y_{k-1}^{b_{k-1,j}}, \tag{4.7}$$

where $C_{d,j}$ is the number of paths associated with the j^{th} input sequence distribution of 1's that generates code vectors of weight d, j_d is the number of distinct input sequence distributions that generate code vectors of weight d, and the entity $b_{0,j}, b_{1,j}, \ldots\ldots, b_{k-1,j}$ represents a particular input sequence distribution of 1's. The

probability that a decoding error occurs for a bit located in the i^{th} position of a message block in the message sequence is then

$$P_b^{(i)}(E) < \sum_d B_d^{(i)} P_d \; for \; 0 \leq i \leq k-1, \tag{4.8}$$

where $B_d^{(i)} = \sum_{j=0}^{j_d} b_{i,j} C_{d,j}$ is the total number of 1's in bit position i contained in all input vectors that generate code vectors of weight d. Note that the new parameters are related to the original parameters by the equations $B_d = \sum_i B_d^{(i)}$ and $P_b(E) = (1/k) \sum_{i=0}^{k-1} P_b^{(i)}(E)$. In addition, the smallest d in the bound for $P_b^{(i)}(E)$ for which $B_d^{(i)}$ is non-zero is the effective free distance of the i^{th} input position, d_i.

In Section 4.2, the effective free distance vector of the $(3,2)$ convolutional encoder with $\mathbf{M} = (1,1)$ and encoding matrices

$$\mathbf{G}_0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}; \mathbf{G}_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.9}$$

was shown to be (3,4). The modified state diagram for the encoder is shown in Figure 4.3. The modified transfer function is

$$\begin{aligned} T(X, Y_0, Y_1) = \; & X^3 Y_0 + \\ & X^4 (2 Y_0 Y_1 + Y_0^2 + Y_0^2 Y_1^2) + \\ & X^5 (Y_1 + 3Y_0^3 Y_1 + 3Y_0 Y_1^2 + Y_0^3 + 3Y_0^2 Y_1^3 + 2Y_0^3 Y_1^2 + Y_0^3 Y_1^4) + ... \end{aligned} \tag{4.10}$$

The transfer function indicates that there is one path of weight 3 through the trellis, and it is generated by the input sequence that has one 1 on line 0 and is zero on line 1.

The bound for the probability of a bit error in the first input position is then $P_b^{(0)}(E) < P_3 + 6P_4 + 30P_5 + ...$ Similarly, the bound for the probability of a bit error in the second input position is given by $P_b^{(1)}(E) < 4P_4 + 27P_5 + ...$

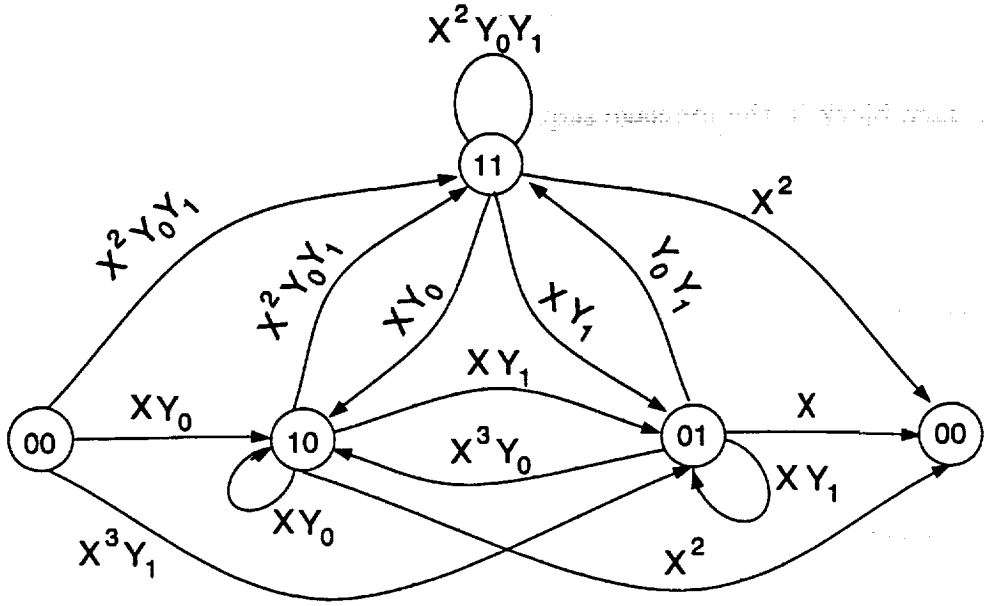


Figure 4.3: Modified state diagram for a specific $(3,2)$ encoder with $\mathbf{d} = (3,4)$

Consider the $(3,2)$ convolutional encoder with $\mathbf{M} = (1,1)$ and encoding matrices

$$\mathbf{G}_0 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \mathbf{G}_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.11}$$

The encoder representation is shown in Figure 4.4, and the modified state diagram for the encoder is shown in Figure 4.5. The modified transfer function is

$$\begin{aligned} T(X, Y_0, Y_1) = \; & X^3(Y_0Y_1) + \\ & X^4(Y_0 + Y_0Y_1^2 + 2Y_0^2Y_1^2) + \\ & X^5(Y_1 + 2Y_0Y_1 + 3Y_0^2Y_1 + Y_0Y_1^3 + 3Y_0^2Y_1^3 + 4Y_0^3Y_1^3) + \cdots \end{aligned} \tag{4.12}$$

and the effective free distance vector is (3,3).

The encoder represented in (4.11) is equivalent to the encoder in (4.9). The difference in the effective free distance vectors demonstrates the dependence of the effective free distance on the encoder realization of the code.

An obvious drawback of the state diagram analysis is the high level of complexity when the memory order and input vector dimension are not restricted to low val-

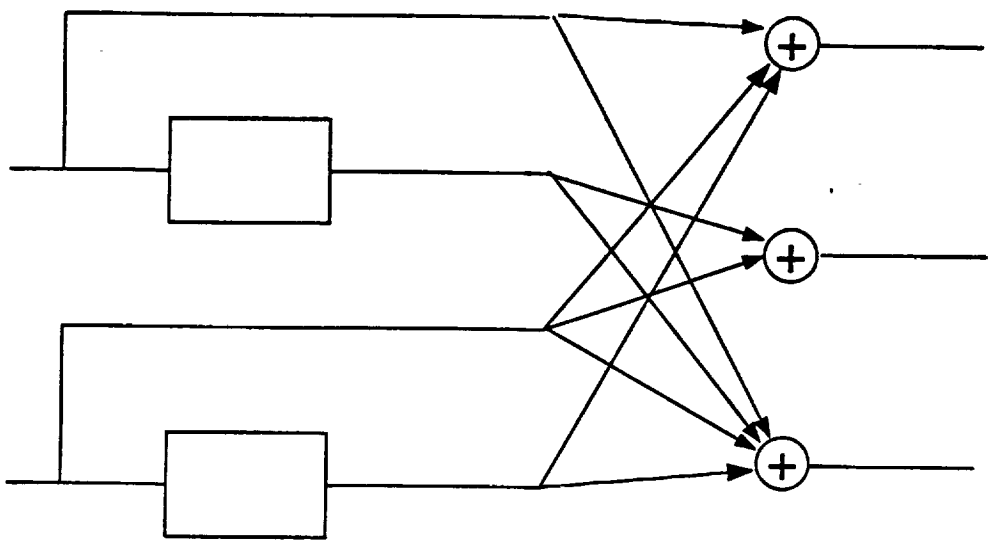


Figure 4.4: A specific $(3,2)$ encoder with $\mathbf{d} = (3,3)$

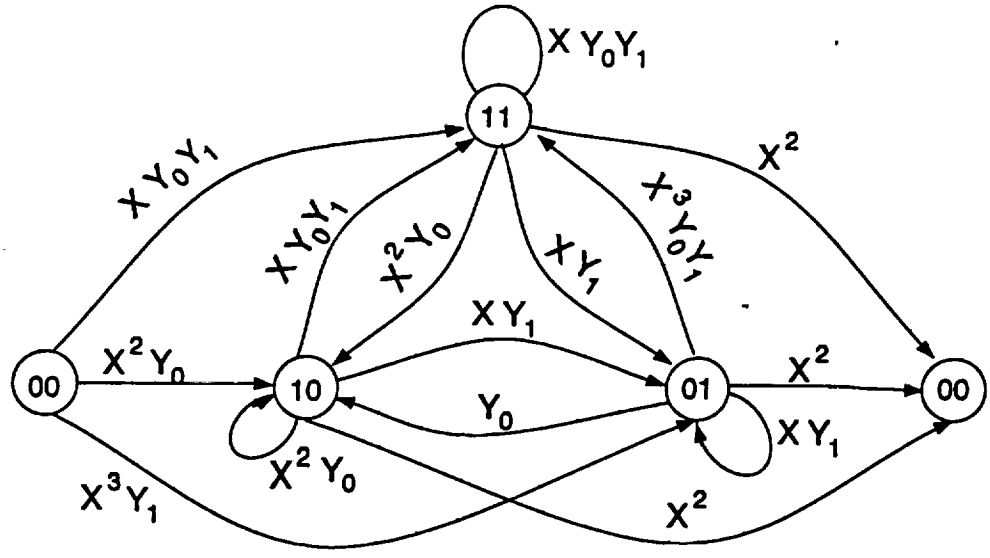


Figure 4.5: Modified state diagram for a specific $(3,2)$ encoder with $\mathbf{d} = (3,3)$

ues. As the total memory increases, the number of states increases exponentially. In addition, as the dimension of the input vector increases, the number of branches leaving each state increases exponentially. However, existing algorithms which attempt to reduce the computational complexity of calculating the transfer function can be modified to incorporate the additional information needed to determine the unequal error protection provided by the code. In addition, the same branch labels can be used in modifications of other algorithms [8] which were developed to determine the free distance vector of an encoder without calculating the transfer function.

4.4 Two-Way Bounds

In this section, a bound on the individual effective free distances is derived. Evaluating the bound is a useful tool in determining the unequal error protection capabilities of encoders of specified rate and memory distribution. In addition, it allows a comparison between the effective free distance of a specific encoder and the theoretically optimal effective free distance. First, a bound on the Hamming weight of the sum of two vectors with known Hamming weights is presented. Then this bound is applied to effective free distances and the implications are discussed.

Let $\mathbf{x}$ and $\mathbf{y}$ be n-bit binary vectors and let $\mathbf{z}$ be the modulo 2 sum of $\mathbf{x}$ and $\mathbf{y}$,

$$\begin{aligned} \mathbf{z} &= (z_1, z_2, ..., z_n) \\ &= x \oplus y \\ &= (x_1 \oplus y_1, x_2 \oplus y_2, ..., x_n \oplus y_n) \end{aligned} \tag{4.13}$$

Assume that the Hamming weights of $\mathbf{x}$ and $\mathbf{y}$ are known and are w_x and w_y, respectively. It can be shown that the Hamming weight of $\mathbf{z}$ is upper bounded by the following relationship

$$w_z \leq \min\{n - w_x, w_y\} + \min\{n - w_y, w_x\}. \tag{4.14}$$

The proof of the bound in (4.14) is given below.

There are two cases which result in $z_i = 1$ and which contribute to the Hamming weight of $\mathbf{z}$. *Case 1* occurs when x_i is 1 and y_i is 0; *Case 2* occurs when x_i is 0 and y_i is 1. The Hamming weight of $\mathbf{z}$ is equal to the total number of bit positions in which either of the two cases appears. Therefore, w_z can be upper bounded by the sum of the maximum number of occurrences of *Case 1* and the maximum number of occurrences of *Case 2*. The number of bit positions in which *Case 1* occurs can be no greater than the minimum of the number of 1 's in $\mathbf{x}$ and the number of 0's in $\mathbf{y}$. Similarly, the number of bit positions in which *Case 2* occurs can be no greater than the minimum of the number of 0's in $\mathbf{x}$ and the number of 1's in $\mathbf{y}$. Therefore,

$$w_z \leq \min\{n - w_x, w_y\} + \min\{n - w_y, w_x\}. \tag{4.15}$$

The bound in (4.14) can be applied to a convolutional encoder and provides the basis for a bound on the effective free distance of a particular input line as a function of the effective free distance of another line.

Recall that a rate $R = k/n$ convolutional encoder with input (message) sequence $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, ...)$, code sequence $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, ...)$, and memory distribution $\mathbf{M} = (m_0, m_1, ..., m_{k-1})$, with $m_0 \leq m_1 \leq ..... \leq m_{k-1}$, can be represented by the equation

$$\mathbf{v}_j = \mathbf{u}_j.\mathbf{G}_0 \oplus \mathbf{u}_{j-1}.\mathbf{G}_1 \oplus ..... \oplus \mathbf{u}_{j-m_{k-1}}.\mathbf{G}_{m_{k-1}}, \tag{4.16}$$

where $\mathbf{u}_i$ is a binary *k-tuple*, $\mathbf{v}_i$ is a binary *n-tuple*, and $\mathbf{G}_i$ is a $(k \times n)$ binary matrix.

We will denote the encoding matrix, $\mathbf{G}$, by the concatenation of the submatrices $\mathbf{G}_i$, $0 \leq i \leq m_{k-1}$, that is,

$$\mathbf{G} = [\mathbf{G}_0|\mathbf{G}_1|\cdots|\mathbf{G}_{\cdot m_{k-1}}] \tag{4.17}$$

Because the effective free distance of input line j, d_j, is the minimum Hamming weight of all codewords associated with input sequences that are non-zero on line j, the effective free distance of a particular input line is no larger than the Hamming weight, w_j, of the corresponding row in the encoding matrix $\mathbf{G}$. In addition, from the definition of d_j, we know that the Hamming weight of the binary sum of row i and row j is at least equal to d_j, for all $i \neq j$, so that

$$w(row(i) \oplus row(j)) \geq d_j. \tag{4.18}$$

Applying the bound of (4.14) to the bound of (4.18) yields an upper bound on d_j in terms of w_i and w_j, that is

$$\begin{aligned} d_j \leq\ & w(row(i) \oplus row(j)) \leq \min\{n(m_{k-1}+1) - w_i, w_j\} + \min\{n(m_{k-1}+1) - w_j, w_i\} \\ \leq\ & [n(m_{k-1}+1) - w_i] + [n(m_{k-1}+1) - w_j] \\ =\ & 2n(m_{k-1}+1) - w_i - w_j. \end{aligned} \tag{4.19}$$

The upper bound in (4.19) can be further manipulated to eliminate the dependence on a particular encoding matrix. The bound is loosened in the process, but it applies to any encoder with the same rate and memory distribution.

Equation (4.19) can be rewritten as

$$d_j + w_j \leq 2n(m_{k-1} + 1) - w_i. \tag{4.20}$$

Since $d_j \leq w_j$, the bound on d_j can be loosened to

$$2d_j \leq 2n(m_{k-1} + 1) - w_i \tag{4.21}$$

Similarly, since $d_i \leq w_i$,

$$2d_j \leq 2n(m_{k-1} + 1) - d_i \tag{4.22}$$

or

$$d_j \leq (1/2)\,[2n(m_{k-1} + 1) - d_i] \tag{4.23}$$

The bound in (4.19) requires knowledge of the Hamming weights of rows i and j of the encoding matrix, whereas the bound in (4.23) presents a relationship between two effective free distances, regardless of the encoder. In addition, the bound in (4.23) can be used to compute the maximum possible value of an individual effective free distance when the minimum value of d, i.e. d_{free}, is known.

To tighten the bound for encoders with unbalanced memory distributions, we consider input sequences constructed so that a periodic impulse of period $m_j + 1$ enters each input line j. The original encoder has the encoder matrix

$$\mathbf{G} = [\mathbf{G}_0|\mathbf{G}_1|\cdots|\mathbf{G}_{m_{k-1}}] \tag{4.24}$$

$$= \begin{bmatrix} \mathbf{r}_0 \\ \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{k-1} \end{bmatrix} \tag{4.25}$$

Using the appropriate periodic input sequences, we can form two vectors

$$\mathbf{r}'_i = \underbrace{[\mathbf{r}_i|\mathbf{r}_i|\ldots|\mathbf{r}_i]}_{b \text{ times}} \tag{4.26}$$

and

$$\mathbf{r}'_j = \underbrace{[\mathbf{r}_j|\mathbf{r}_j|\ldots|\mathbf{r}_j]}_{c \text{ times}}, \tag{4.27}$$

which are valid code sequences.

Note that $w'_i = w_H(\mathbf{r}'_i) = bw_i$ and $w'_j = w_H(\mathbf{r}'_j) = cw_j$. From the definition of the effective free distance,

$$d_j \leq w_H(\mathbf{r}'_i \oplus \mathbf{r}'_j). \tag{4.28}$$

Let $N = n \max[b(m_i+1), c(m_j+1)]$. From the vector bound in (4.14),

$$w_H(\mathbf{r}'_i \oplus \mathbf{r}'_j) \leq \min\left\{N - w'_i, w'_j\right\} \tag{4.29}$$

$$+ \min\left\{N - w'_j, w'_i\right\}. \tag{4.30}$$

So,

$$d_j \leq \min\left\{N - w'_i, w'_j\right\} \tag{4.31}$$

$$+ \min\left\{N - w'_j, w'_i\right\} \tag{4.32}$$

$$\leq [N - w'_i] + [N - w'_j] \tag{4.33}$$

$$= 2N - w'_i - w'_j. \tag{4.34}$$

Rewriting,

$$d_j + w'_j + w'_i \leq 2N \tag{4.35}$$

or

$$d_j + bw_i + cw_j \leq 2N. \tag{4.36}$$

Since $d_j \leq w_j$ and $d_i \leq w_i$, the bound can be loosened to

$$bd_i + (c+1)d_j \leq 2N = 2n \max\left[b(m_i+1), c(m_j+1)\right] \tag{4.37}$$

When the number of information lines, *i.e.* k, is larger than 2, it is possible to repeatedly apply the two-way bounds to the effective distances. For instance, if $k = 3$, then the following bounds hold

$$\begin{aligned} d_0 + 2d_1 &\leq 2n(m+1) \\ d_0 + 2d_2 &\leq 2n(m+1) \\ d_1 + 2d_2 &\leq 2n(m+1), \end{aligned} \tag{4.38}$$

where $m = \max\{m_0, m_1, m_2\}$. The bounds are tightest when d_0, d_1, and d_2 are ordered so that $d_0 \leq d_1 \leq d_2$. Assuming such an ordering, $d_0 = d_{free}$, the largest possible value of d_2 is $\frac{1}{2}(2n(m+1) - d_0)$, which occurs when $d_1 = d_0 = d_{free}$. If the effective distance of line 1 is to be increased to $d_0 + a$, then the maximum allowed value of d_2 is decreased to $\frac{1}{2}(2n(m+1) - d_0 - a)$. As an example, for a rate 3/4 encoder with $\mathbf{M} = (1,1,1)$, if d_0 is 2 and d_1 is 3, then the largest possible value of d_2 is 6. In fact, when d_2 is 6, the bound permits values of d_0 and d_1 up to 4.

4.5 k-Way Bounds

An alternative bound also applies to the effective free distances. The advantage of this bound is that is applies to more than two distances at one time. When $k = 2$, the following bound reduces to the bound in Equation (4.23).

Again, assume that $m_0 \leq \ldots \leq m_{k-1}$. No assumption on the relative sizes of the effective distances is necessary.

The generator matrix forms a rate $\frac{k}{n(m_{k-1}+1)}$ block code with $\mathbf{d} = (d_0, \ldots, d_{k-1})$.

It is obvious that

$$d_0 \leq n(m_0 + 1). \tag{4.39}$$

Because the generator matrix has at least one zero in each row, j, for which $m_j \neq 0$, the first equation of the k-way bound can be tightened slightly, to

$$d_0 \leq n(m_0 + 1) - 1. \tag{4.40}$$

Following the development of the Griesmer bound [17], rearrange the columns so that the first row has only ones in the first d_0 positions, and only zeros in the remaining $n(m_{k-1} + 1) - d_0$ positions. Several variables are useful in the bound development. We define y_j^0 as the number of 1's in the last $n(m_j + 1) - d_0$ positions of row j, s_j^0 as the number of 1's in the first d_0 positions of (row $j \oplus$ row 0), and l_j as the number of 1's in the first d_0 positions of row j.

Then for $j = 1, \ldots, k-1$, row j can have either:

a) $\geq \lceil \frac{d_0}{2} \rceil$ 1's or

b) $\geq \lceil \frac{d_0}{2} \rceil$ 0's

in the first d_0 positions.

Situation a) implies than $s_j^0 \leq d_0 - \lceil \frac{d_0}{2} \rceil$. Since $w_H(row j \oplus row 0) \geq d_j$ and $w_H(row j \oplus row 0) = s_j^0 + y_j^0$,

$$s_j^0 + y_j^0 \geq d_j. \tag{4.41}$$

So,

$$d_0 - \lceil \frac{d_0}{2} \rceil + y_j^0 \geq s_j^0 + y_j^0 \geq d_j, \tag{4.42}$$

or

$$y_j^0 \geq d_j - (d_0 - \lceil \frac{d_0}{2} \rceil). \tag{4.43}$$

Since y_j^0 is obviously upperbounded by $n(m_j+1)-d_0$,

$$d_j-(d_0-\lceil\frac{d_0}{2}\rceil)\leq n(m_j+1)-d_0 \tag{4.44}$$

or

$$d_j+\lceil\frac{d_0}{2}\rceil\leq n(m_j+1). \tag{4.45}$$

Allowing $j=1$ and defining $X_1=\lceil\frac{d_0}{2}\rceil$, we have

$$d_1+X_1\leq n(m_1+1) \tag{4.46}$$

for $j=1,\ldots,k-1$.

Because l_j is the number of 1's in the first d_0 positions of row j, and situation b) assumes that there are at least $\lceil\frac{d_0}{2}\rceil$ 0's in the first d_0 positions, this implies that $d_0-\lceil\frac{d_0}{2}\rceil\geq l_j\geq 0$. Also, because the Hmming weight of row j is no less than d_j,

$$l_j+y_j^0\geq d_j \tag{4.47}$$

or

$$y_j^0\geq d_j-(d_0-\lceil\frac{d_0}{2}\rceil). \tag{4.48}$$

It then follows that

$$d_j+\lceil\frac{d_0}{2}\rceil\leq n(m_j+1). \tag{4.49}$$

This is the same as the result when situation a) is assumed.

When the first row and the first d_0 columns of the encoding matrix are removed, we are left with a residual block code of rate $\frac{k-1}{n(m_{k-1}+1)-d_0}$ with $\mathbf{d}\geq(d_1-(d_0-X_1),d_2-(d_0-X_1),\ldots,d_{k-1}-(d_0-X_1))$. To continue, we follow the same procedure as we did for the original block code That is, first rearrange the columns so that the first row has only ones in the first $d_1-(d_0-X_1)$ positions, and only zeros in the remaining $n(m_{k-1})-d_0-(d_1-(d_0-X_1))=n(m_{k-1})-d_1-X_1$ positions. Then for

$j = 2, \ldots, k-1$, the portion of the original row j that belongs to the residual code can have either: a) $\geq \lceil \frac{d_1-(d_0-X_1)}{2} \rceil$ 1's or b) $\geq \lceil \frac{d_1-(d_0-X_1)}{2} \rceil$ 0's in the first $d_1 - (d_0 - X_1)$ positions.

Again, more variables are defined. Let y_j^1 be the number of 1's in the last $n(m_j + 1) - d_0 - (d_1 - (d_0 - X_1))$ positions of row j, s_j^1 be the number of 1's in the first $d_1 - (d_0 - X_1)$ positions of row $j \oplus$ row 1, and l_j be the number of 1's in the first $d_1 - (d_0 - X_1)$ positions of row j.

Situation a) implies than $s_j^1 \leq d_1 - (d_0 - X_1) - \lceil \frac{d_1-(d_0-X_1)}{2} \rceil$. Since $w_H(row j \oplus row 1) \geq d_j - (d_0 - X_1)$ and $w_H(row j \oplus row 1) = s_j^1 + y_j^1$,

$$s_j^1 + y_j^1 \geq d_j - (d_0 - X_1) \tag{4.50}$$

or

$$y_j^1 \geq d_j - (d_0 - X_1) - (d_1 - (d_0 - X_1) - \lceil \frac{d_1 - (d_0 - X_1)}{2} \rceil)) \tag{4.51}$$

or

$$y_j^1 \geq d_j - d_1 + \lceil \frac{d_1 - (d_0 - X_1)}{2} \rceil. \tag{4.52}$$

But $y_j^1 \leq n(m_j + 1) - d_0 - (d_1 - (d_0 - X_1))$, so

$$d_j - d_1 + \lceil \frac{d_1 - (d_0 - X_1)}{2} \rceil \leq n(m_j + 1) - d_0 - (d_1 - (d_0 - X_1)), \tag{4.53}$$

or

$$d_j + X_2 + X_1 \leq n(m_j + 1), \tag{4.54}$$

where $X_2 = \lceil \frac{d_1-(d_0-X_1)}{2} \rceil$,for $j = 2, \ldots, k-1$. If $j = 2$, then

$$d_2 + X_2 + X_1 \leq n(m_2 + 1). \tag{4.55}$$

Situation b) implies similar results.

The process can be repeated for the residual block codes up to rate $\frac{2}{n(m_{k-1}+1)-d_0-(d_1-(d_0-X_1))-\ldots-}$ resulting in a set of k equations:

$$d_0 \leq n(m_0+1)-1, \tag{4.56}$$

and

$$d_j + \sum_{l=1}^{j} X_l \leq n(m_j+1) \tag{4.57}$$

for $j = 1, \ldots, k-1$, where $X_1 = \lceil \frac{d_0}{2} \rceil$, and $X_l = \lceil \frac{d_{l-1}-d_{l-2}+X_{l-1}}{2} \rceil$ for $l = 2, \ldots k-1$.

4.6 Plotkin-Type Bound

The Plotkin bound for block codes states that a code of length n with M codewords has minimum distance

$$d_{min} \leq \frac{n}{2}\frac{1}{1-1/M}. \tag{4.58}$$

When the non-zero input to a convolutional encoder has length h, the encoder can be considered as a block code with 2^{hk} codewords of length $h(m_{k-1}+1)$. Therefore, the free distance of the code may be upperbounded [30] by

$$d_{free} \leq \frac{n(m_{k-1}+h)}{2}\frac{2^{hk}}{2^{hk}-1}, h = 1, 2, \cdots. \tag{4.59}$$

The effective distance for a specific input line, j, is the minimum Hamming weight among codewords that are associated with inputs that contain at least one 1 on that input line. A code generated by the inputs of length h which have at least one 1 on line j is called the restricted block code, C_j^h. The set of such inputs and outputs may be considered as a series of block codes, similar to the approach used for the Plotkin bound. We define C_j^h as the number of codewords in the restricted block code C_j^h. Then, C_j^h is equal to the total number of codewords in the unrestricted block code

with the same size input vectors minus the number of codewords that are all-zero on line j, or

$$C_j^h = 2^{hk} - 2^{h(k-1)} \tag{4.60}$$

or

$$C_j^h = 2^{h(k-1)}(2^h - 1). \tag{4.61}$$

Then, using the bound in (4.59), d_j is upper-bounded by

$$d_j \leq \frac{n(m_{k-1}+h)}{2} \frac{2^{h(k-1)}(2^h-1)}{2^{h(k-1)}(2^h-1)-1}, h = 1, 2, ... \tag{4.62}$$

for $j = 0, \cdots, k-1$. Note that each effective distance is subject to the same bounding value, i.e., the bound is independent of j.

4.7 Another bound

Another bound which applies to the effective free distance d_j is quickly seen from the fact that the allowable inputs include the set of inputs which are all-zero on all lines that are not the line of interest. Therefore,

$$d_j \leq d_{opt}(n, 1, m_j), \tag{4.63}$$

where $d_{opt}(n, 1, m_j)$ is the optimal achievable free distance of the rate $1/n$, memory m_j convolutional encoder. However, bound (4.63) is always looser than bound (4.62), making it useful only as a quick indication of the maximum possible effective distance vector. The bounds are applied to encoders of various rates and memory distributions in Tables 4.1 - 4.4. For each listed encoder configuration, the bounds on the effective distance vector for the given free distance are listed. The optimal (achieved) free distance, d_{opt} is provided for reference.

Rate=2/3					
M	d_{opt}	Bound (4.23)	Bound (4.37)	Bound (4.57)	Bound (4.62)
(1,1)	3	(3,4)	(3,4)	(3,4)	(3,4)
(1,2)	4	(4,7) (3,8)	(4,7) (3,8)	(4,7) (3,7)	(4,6) (3,6)
(2,2)	5	(5,6) (4,7) (3,7)	(5,6) (4,7) (3,7)	(5,6) (4,7) (3,7)	(5,6) (4,6) (3,6)
(1,3)	5	(5,9) (4,10) (3,10)	(5,7) (4,8) (3,9)	(5,9) (4,10) (3,10)	(5,8) (4,8) (3,8)
(1,4)	8	(5,12) (4,13) (3,13)	(5,10) (4,11) (3,12)	(5,12) (4,13) (3,13)	(5,9) (4,9) (3,9)
(2,3)	6	(6,9) (5,9) (4,10) (3,10)	(6,9) (5,9) (4,10) (3,10)	(6,9) (5,9) (4,10) (3,10)	(6,8) (5,8) (4,8) (3,8)

Table 4.1: UEP bounds for Rate 2/3 Convolutional Encoders

4.8 Results

A non-exhaustive search for codes that meet these bounds was conducted. The FAST algorithm presented in [8] was used with the branch labeling method presented in Section 4.3 to determine the effective free distances. The search method stepped through successive possible encoding matrices, and immediately rejected encoders which contained a row with a Hamming weight less than the specified desired minimum distance for the corresponding input line. In addition, the search algorithm rejected encoders which had taken "too many" (typically $5*m_{k-1}$) steps along a path without increasing the Hamming weight of the output. Theoretically, this criterion may reject encoders that are not catastrophic and that have the desired effective free distance vector, but it greatly reduced search time.

Tables 4.5 and 4.6 give the result for rate 2/3 and rate 2/4 encoders, respectively.

Rate=2/4					
M	d_{opt}	Bound (4.23)	Bound (4.37)	Bound (4.57)	Bound (4.62)
(1,1)	5	(5,5)	(5,5)	(5,5)	(5,6)
		(4,6)	(4,6)	(4,6)	(4,6)
(1,2)	6	(6,9)	(6,9)	(6,9)	(6,8)
		(5,9)	(5,9)	(5,9)	(5,8)
		(4,10)	(4,10)	(4,10)	(4,8)
(2,2)	7/8	(8,8)	(8,8)	(8,8)	(8,8)
		(7,8)	(7,8)	(7,8)	(7,8)
		(6,9)	(6,9)	(6,9)	(6,8)
		(5,9)	(5,9)	(5,9)	(5,8)
(1,3)	7/8	(7,12)	(7,9)	(7,12)	(7,10)
		(6,13)	(6,10)	(6,13)	(6,10)
		(5,13)	(5,11)	(5,13)	(5,10)
		(4,14)	(4,12)	(4,14)	(4,10)
		(3,14)	(3,13)	(3,14)	(3,10)
(1,4)	8	(7,16)	(7,13)	(7,16)	(7,13)
		(6,17)	(6,14)	(6,17)	(6,13)
		(5,17)	(5,15)	(5,17)	(5,13)
		(4,18)	(4,16)	(4,18)	(4,13)
		(3,18)	(3,17)	(3,18)	(3,13)
(2,3)	8	(8,12)	(8,12)	(8,12)	(8,10)
		(7,12)	(7,12)	(7,12)	(7,10)
		(6,13)	(6,13)	(6,13)	(6,10)
		(5,13)	(5,13)	(5,13)	(5,10)
		(4,14)	(4,14)	(4,14)	(4,10)
		(3,14)	(3,14)	(3,14)	(3,10)

Table 4.2: UEP bounds for Rate 2/4 Convolutional Encoders

Rate=2/5					
M	d_{opt}	Bound (4.23)	Bound (4.37)	Bound (4.57)	Bound (4.62)
(1,1)	6	(6,7)	(6,7)	(6,7)	(6,8)
		(5,7)	(5,7)	(5,7)	(5,8)
		(4,8)	(4,8)	(4,8)	(4,8)
(1,2)	9	(9,10)	(9,10)	(9,10)	(9,10)
		(8,11)	(8,11)	(8,11)	(8,10)
		(7,11)	(7,11)	(7,11)	(7,10)
		(6,12)	(6,12)	(6,12)	(6,10)
		(5,12)	(5,12)	(5,12)	(5,10)
(2,2)	9	(10,10)	(10,10)	(10,10)	(10,10)
		(9,10)	(9,10)	(9,10)	(9,10)
		(8,11)	(8,11)	(8,11)	(8,10)
		(7,11)	(7,11)	(7,11)	(7,10)
		(6,12)	(6,12)	(6,12)	(6,10)
		(5,12)		(5,12)	(5,10)
(1,3)	5	(9,15)	(9,11)	(9,15)	(9,13)
		(8,16)	(8,12)	(8,16)	(8,13)
		(7,16)	(7,13)	(7,16)	(7,13)
		(6,17)	(6,14)	(6,17)	(6,13)
		(5,17)	(5,15)	(5,17)	(5,13)
(1,4)	8	(9,20)	(9,16)	(9,20)	(9,16)
		(8,21)	(8,17)	(8,21)	(8,16)
		(7,21)	(7,18)	(7,21)	(7,16)
		(6,22)	(6,19)	(6,22)	(6,16)
		(5,22)	(5,20)	(5,22)	(5,16)
(2,3)	6	(13,13)	(13,13)	(13,13)	(13,13)
		(12,14)	(12,14)	(12,14)	(12,13)
		(11,14)	(11,14)	(11,14)	(11,13)
		(10,15)	(10,15)	(10,15)	(10,13)
		(9,15)	(9,15)	(9,15)	(9,13)
		(8,16)	(8,16)	(8,16)	(8,13)
		(7,16)	(7,16)	(7,16)	(7,13)
		(6,17)	(6,17)	(6,17)	(6,13)
		(5,17)	(5,17)	(5,17)	(5,13)

Table 4.3: UEP bounds for Rate 2/5 Convolutional Encoders

Rate=3/4					
M	d_{opt}	Bound (4.23)	Bound (4.37)	Bound (4.57)	Bound (4.62)
(1,1,1)	4	(5,5,5)	(5,5,5)	(4,4,5)	(4,5,5)
(1,1,2)	4	(5,5,9) (4,4,10) (4,6,9)	(5,5,9) (4,4,10) (4,6,9)	(5,5,7) (4,4,9) (4,6,8)	(5,8,8)
(1,2,2)	5	(6,6,9) (6,8,8) (4,4,10)	(6,6,9)	(6,6,7) (4,7,7) (4,6,8) (4,4,10)	(6,8,8)
(2,2,2)	6	(6,6,9) (6,8,8) (4,4,10)	(6,6,9) (6,8,8) (4,4,10)	(6,6,7) (5,6,8) (4,4,9) (4,7,7)	(6,8,8)
(1,1,3)	5	(5,5,13) (4,4,14) (4,6,13)	(5,5,11) (4,4,12) (4,6,10)	(5,6,11) (5,8,10) (4,4,13) (4,6,12)	(5,10,10)
(1,2,3)	6	(7,8,12) (6,9,11)	(7,8,9)	(7,8,9) (7,7,10) (6,7,11) (6,9,10)	(7,10,10)

Table 4.4: UEP bounds for Rate 3/4 Convolutional Encoders

A primary goal of the searches was to find encoders with at least one effective distance greater than the free distance of the optimal code of the same rate and memory order. A decrease in free distance is acceptable. In Table 4.5, there are four instances in which the higher effective free distance is larger than the optimal free distance for rate 2/3 encoders with the same state complexity. For $\mathbf{M} = (1,1)$, $\mathbf{d} = (3,4)$, and $d_{free} = 3$ for the time-invariant convolutional code, and $d_{free} = 4$ for the time-varying convolutional code as shown in [46]. Similarly, the optimal encoder for $\mathbf{M} = (1,1)$ [32] has an effective free distance vector $\mathbf{d} = (4,5)$. These two encoders are examples of encoders which provide unequal error protection while maintaining a free distance equal to the optimal free distance for the same rate and state complexity. Two encoders with state complexity 4 have one effective free distance vector $\mathbf{d} = (4,6)$. One has memory distribution $\mathbf{M} = (2,2)$, and the other has memory distribution $\mathbf{M} = (1,3)$. An encoder with $\mathbf{d} = (6,6)$ requires a state complexity of 5 [32], so allowing the protection of one bit to drop from an effective distance of 6 to an effective distance of 4 allows a reduction in the state complexity. The trellis associated with an encoder of state complexity 4 has $2^4 = 16$ states, while the trellis associated with an encoder of state complexity of 5 has $2^5 = 32$ states.

Table 4.6 shows several rate 2/4 encoders with one effective free distance that is larger than the optimal free distance for rate 1/2 encoders with the same rate and state complexity. Several are noted in the following discussion. For $\mathbf{M} = (1,2)$ the encoder with effective free distance $\mathbf{d} = (6,7)$ increases the protection provided to one of the input bits. In comparison, the rate 1/2, $M = 3$ optimal encoder provides a free distance of 6 for all input bits, and a state complexity of 4 is required for a rate 1/2 encoder to provide a free distance of 7 to all input bits. The encoder that provides error protection $\mathbf{d} = (4,8)$ with $\mathbf{M} = (1,2)$ increases the effective

free distance on the second bit position by two, with the cost being a corresponding decrease in the effective free distance on the first bit. Note that the optimal rate 1/2 encoder that provides a free distance of 8 requires a state complexity of 5. The encoder with $\mathbf{M} = (2,2)$ and $\mathbf{d} = (8,8)$ although not being a LUEP code, it is interesting because it achieves the free distance shown to be achievable by unit-memory codes [31], and both effective free distance exceed the free distance provided by the optimal rate 1/2, $M = 4$ encoder. The $\mathbf{M} = (1,3)$ encoder with effective free distance vector $\mathbf{d} = (4,9)$ provides protection to one of the bits which exceed the protection offered by even unit-memory codes of the same state complexity. The $\mathbf{M} = (2,3)$ encoder with effective free distance vector $\mathbf{d} = (8,9)$ is interesting because it provides unequal error protection while maintaining a free distance equal to the optimal value of 8. However, some encoders are obviously better choices for implementation. For instance, when the memory distribution is $\mathbf{M} = (1,2)$, the encoder with $\mathbf{d} = (4,8)$ is better than the encoder with $\mathbf{d} = (3,8)$.

The tightest bounds for the effective free distances are listed in the tables. It can be seen that the derived bounds are relatively tight when the memory distribution is balanced. The bounds appear to loosen as the memory increases and as the memory distribution becomes more unbalanced. The bounds also loosen as k increases. For example, the bound for rate 3/4, $\mathbf{M} = (1,1,1)$ encoders is looser than the bound for rate 2/4 $\mathbf{M} = (1,1)$ encoders.

Figures 4.6 - 4.33 show the bit error rate (BER) plots for the codes presented in Tables 4.5, 4.6, 4.9, using Viterbi decoding with soft decision decoding. Three sets of data points are shown in each plot. The data points described by the 'x' are the (simulated) bit error rate for input line 0. Similarly, the data points described by the 'o' are the (simulated) bit error rate for input line 1. The overall BERs are marked

Rate=2/3				
M	d_{opt}	d($bound$)	d	G
(1,1)	3	(3,4)	(3,4)	6 4 5 3
(1,2)	4	(4,6)	(4,5)	5 3 0 3 1 5
	4	(3,6)	(3,5)	2 3 0 5 7 5
(2,2)	5	(5,6)	(5,5)	5 4 6 3 5 3
	5	(4,6)	(4,6)	1 5 1 7 2 6
(1,3)	5	(5,7)	(5,5)	3 7 0 0 4 6 7 6
	5	(4,8)	(4,6)	6 3 0 0 1 3 7 5
(1,4)	6	(5,9)	(5,6)	7 5 0 0 0 5 1 2 4 7
	6	(4,9)	(4,6)	3 5 0 0 0 5 1 2 1 4
(2,3)	6	(6,7)	(6,6)	5 7 3 0 3 1 5 5
	6	(5,8)	(5,6)	1 5 6 0 3 2 1 3
	6	(4,8)	(4,6)	4 2 5 0 0 3 1 7

Table 4.5: Rate 2/3 UEP Convolutional Encoders

Rate=2/4				
M	d_{opt}	d($bound$)	d	G
(1,1)	5	(5,5)	(5,5)	15 14 03 07
(1,2)	6	(6,8)	(6,7)	17 06 00 06 12 15
	6	(5,8)	(5,7)	07 06 00 13 10 13
	6	(4,8)	(4,8)	14 05 00 07 17 13
(2,2)	$7/8^{UMC}$	(8,8)	(8,8)	16 11 16 05 17 05
		(7,8)	(7,7)	15 06 14 03 05 13
		(6,8)	(6,8)	01 07 03 14 13 16
(1,3)	$7/8^{UMC}$	(7,9)	(7,7)	07 17 00 00 10 11 15 03
		(6,10)	(6,8)	13 15 00 00 15 13 01 13
		(5,10)	(5,7)	14 07 00 00 17 02 01 16
		(4,10)	(4,9)	14 03 00 00 07 11 01 16
		(3,10)	(3,10)	02 06 00 00 15 13 01 13
(1,4)	8	(7,13)	(7,8)	17 07 00 00 00 06 13 01 14 12
	8	(6,13)	(6,10)	16 13 00 00 00 05 03 13 11 15
	8	(5,13)	(5,9)	11 07 00 00 00 16 05 14 04 13
	8	(4,13)	(4,10)	12 14 00 00 00 17 15 04 07 13
	8	(3,13)	(3,12)	10 06 00 00 00 17 15 04 07 13
(2,3)	8	(8,10)	(8,9)	05 17 05 00 13 14 15 16
	8	(7,10)	(7,8)	05 12 13 00 13 06 15 07
	8	(6,10)	(6,8)	01 03 07 00 14 03 10 16
	8	(5,10)	(5,9)	05 12 01 00 13 06 07 14

Table 4.6: Rate 2/4 UEP Convolutional Encoders

Rate=2/5				
M	d_{opt}	d($bound$)	d	G
(1,1)	6	(6,7)	(6,7)	13 26 34 35
		(5,7	(5,7)	03 32 37 15
(1,2)	9	(8,10)	(8,8)	07 37 00 31 21 23
		(7,10)	(7,8)	15 17 00 03 36 32
		(6,10)	(6,10)	03 27 00 35 32 24
(2,2)	9	(9,10)	(9,9)	33 01 17 11 36 32
		(8,10)	(8,10)	03 23 07 11 36 32
(1,3)	12	(8,11)	(8,9)	35 17 00 00 26 32 11 07
		(7,12)	(7,10)	36 25 00 00 33 04 15 13
		(6,13)	(6,11)	16 26 00 00 33 04 15 13

Table 4.7: Rate 2/5 UEP Convolutional Encoders

Rate=3/4				
M	d_{opt}	d($bound$)	d	G
(0,1,2)	4	(4,4,9)	(4,4,5)	17 00 00 05 06 00 03 04 03
		(3,3,9)	(3,3,5)	03 00 00 17 06 00 04 17 12
(1,2,2)	5	(5,6,7)	(5,5,5)	11 17 00 05 05 11 03 04 03
		(4,4,10)	(4,4,6)	01 13 00 03 03 14 16 04 12

Table 4.8: Rate 3/4 UEP Convolutional Encoders

by '*'s. For cases in which BER was lower than 10^{-7}, data points do not appear. For comparison, the BER plots for the optimal free distance codes listed in [32] are shown in Figures 4.34 - 4.42. The encoding matrices and effective free distance vectors for the optimal codes are listed in Table 4.9.

(n,k)	M	d	G
(3,2)	(1,1)	(3,3)	4 7 3 4
	(1,2)	(4,5)	5 3 0 3 1 5
	(2,2)	(5,5)	5 4 6 3 5 3
	(2,3)	(6,6)	5 7 3 0 3 1 5 5
	(3,3)	(7,7)	5 7 2 5 3 7 5 3
(2,1)	(2)	(5)	3 1 3
	(3)	(6)	3 3 1 3
	(4)	(7)	3 1 1 2 3
	(5)	(8)	3 2 1 3 1 3

Table 4.9: Optimal convolutional encoders

Some observations about the BER plots for the UEP convolutional encoders follow. Examining Figure 4.7, it is seen that for the rate 2/3, $\mathbf{M} = (1,2)$ encoder, which has $\mathbf{d} = (4,5)$, the BER for line 1 is lower for all SNRs than the BER for line 0, which was expected. The disparity in the protection offered to the two input positions increases as the SNR increases, similar to the manner in which the disparity in average BER for two codes with different free distances increases with increasing SNR. Every error probability for the rate 2/3, $\mathbf{d} = (4,5)$ encoder is lower than the error probability for the optimal encoder, shown in Figure 4.35. The average BERs in Figure 4.7 are lower than the average BERs in Figure 4.8, which is expected for encoders which have effective distance vectors $\mathbf{d} = (4,5)$ and $\mathbf{d} = (3,5)$, respectively. In addition, although line 1 for both encoders have the same effective free distance of 5, the BER of line 1 in

Figure 4.7 is lower than the BER of line 1 in Figure 4.8. This phenomemon is seen in comparisons of other encoders, also. The phenomenon is possibly due to an increase in the number of codewords at each distance, so that while the effective distances are equal, the differences in multiplicities are large enough to affect the individual error rates.

The rate 2/3, $\mathbf{M} = (2, 2)$ encoder with $\mathbf{d} = (4, 6)$ is an encoder which achieves an effective free distance on one line that is higher than the optimal free distance by reducing the effective free distance of the other line. Comparing the error rates in Figure 4.10 to the rates for the comparable optimal code in Figure 4.36, it is seen that, as expected, the optimal code has a lower average BER at every SNR. However, line 1 of Figure 4.10 has a significantly lower BER at 4 dB.

The encoder analyzed in Figure 4.16 was constructed from a basic $(2, 1, 2)$ encoder, which can provide no unequal error protection [43]. The slight differences in the individual BERs for line 0 and line 1 at 3 dB and 4 dB are probably due to the limited number of information bits that were encoded and decoded in the simulation.

The rate 2/4, $\mathbf{M} = (1, 2)$, $\mathbf{d} = (6, 7)$ encoder, when compared to the $(2, 1, 3)$, free distance 6 optimal encoder, has a slightly lower average BER at 3 dB and 5 dB, with a slightly higher average BER at 4 dB. At 5 dB, the individual bit error rate is significantly lower for line 0.

The results in Figure 4.18 are interesting. The protection on line 0 follows the standard curve shape, but the protection on line 1 has a non-standard from. The error rate on line 1 is lower than the average BER for the corresponding optimal code.

It is interesting that the differences in the effective distance vectors of the rate 2/4 encoder with $\mathbf{d} = (8, 8)$ and the corresponding optimal basic code with $d_{free} = 7$ do

not correspond to significant differences in the BER plots. The error rates in Figure 4.20 are lower, but the differences in the two plots do not indicate that the effective distance vectors are significantly different. This is an instance in which the number of low weight codewords is a significant factor, which can be seen from the transfer functions of the two encoders. The modified transfer function for the rate 2/4 encoder is

$$T(X, Y_0, Y_1) = X^8(Y_0+Y_1+3Y_0Y_1+3Y_0Y_1^2+Y_0Y_1^3+Y_1^2+2Y_0^2Y_1+Y_0^2Y_1^2+Y_0^2Y_1^4)+\cdots. \tag{4.64}$$

On the other hand, the transfer function for the rate 1/2 encoder is

$$T(X, Y) = X^7(Y + Y^3) + X^8(Y^2 + Y^4) + \cdots. \tag{4.65}$$

The rate 1/2 encoder has a total of four code sequences of weight 8 or less, while the rate 2/4 encoder has fourteen code sequences of weight 8.

The rate 2/4 encoder with $\mathbf{d} = (6, 8)$ has lower BERs than those of the optimal encoder with $d_{free} = 7$, which is the desired effect.

The BER plots illustrate the dependence of the probability of error on each input line on both the effective free distance and the number of codewords at that effective distance. The effective distance vector alone provides valid information about the disparity in unequal error protection, but the multiplicities provide additional useful information.

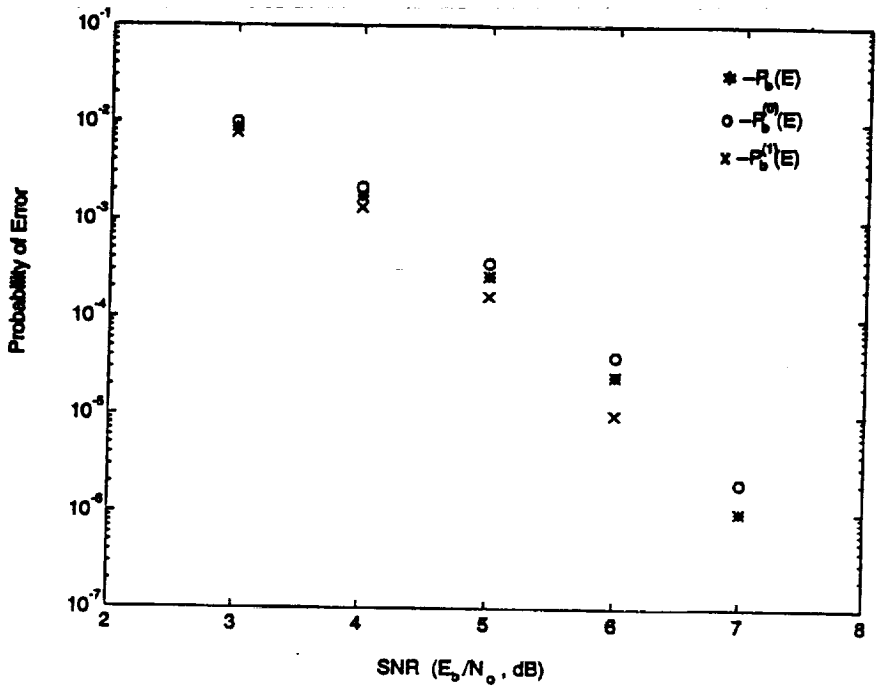


Figure 4.6: BER plot for $R = 2/3$, $\mathbf{M} = (1, 1)$ encoder with $\mathbf{d} = (3, 4)$

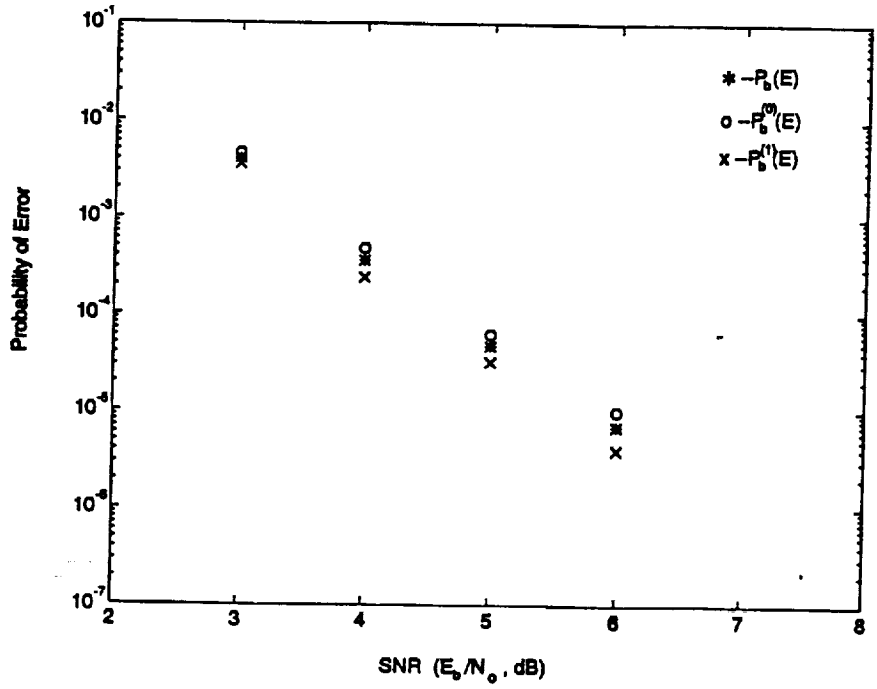


Figure 4.7: BER plot for $R = 2/3$, $\mathbf{M} = (1, 2)$ encoder with $\mathbf{d} = (4, 5)$

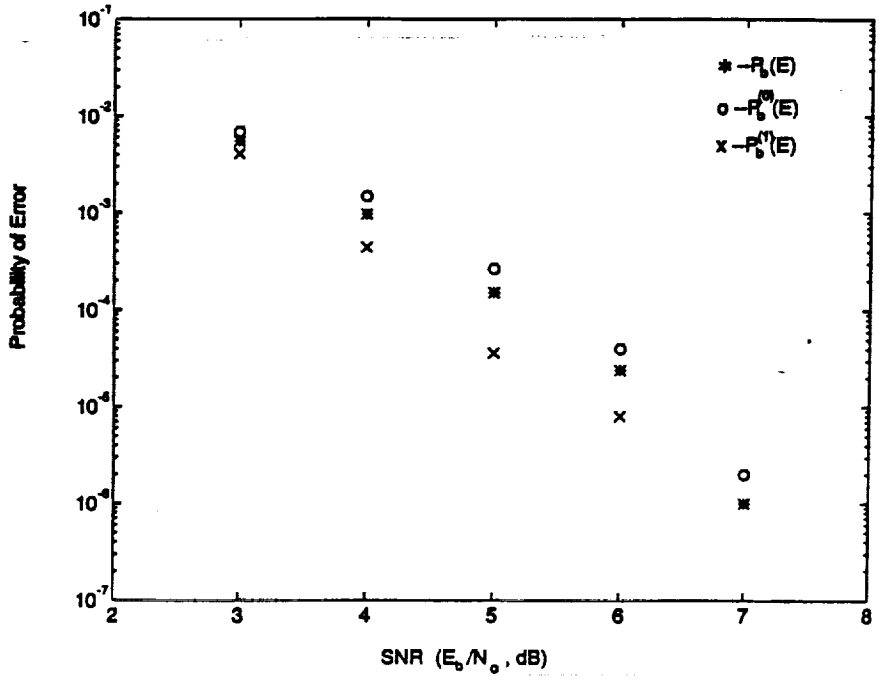


Figure 4.8: BER plot for $R = 2/3$, $\mathbf{M} = (1, 2)$ encoder with $\mathbf{d} = (3, 5)$

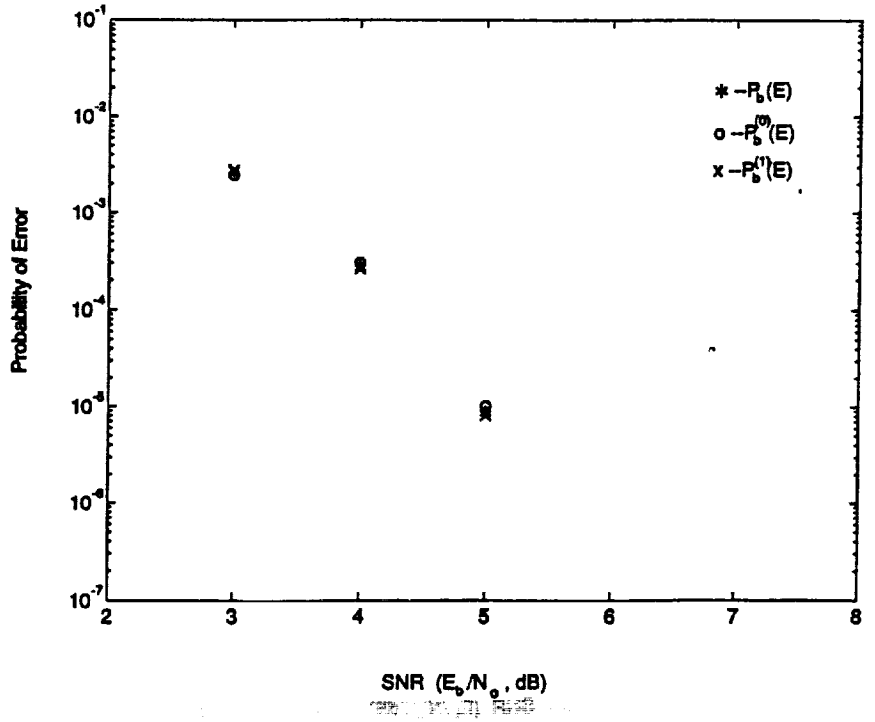


Figure 4.9: BER plot for $R = 2/3$, $\mathbf{M} = (2, 2)$ encoder with $\mathbf{d} = (5, 5)$

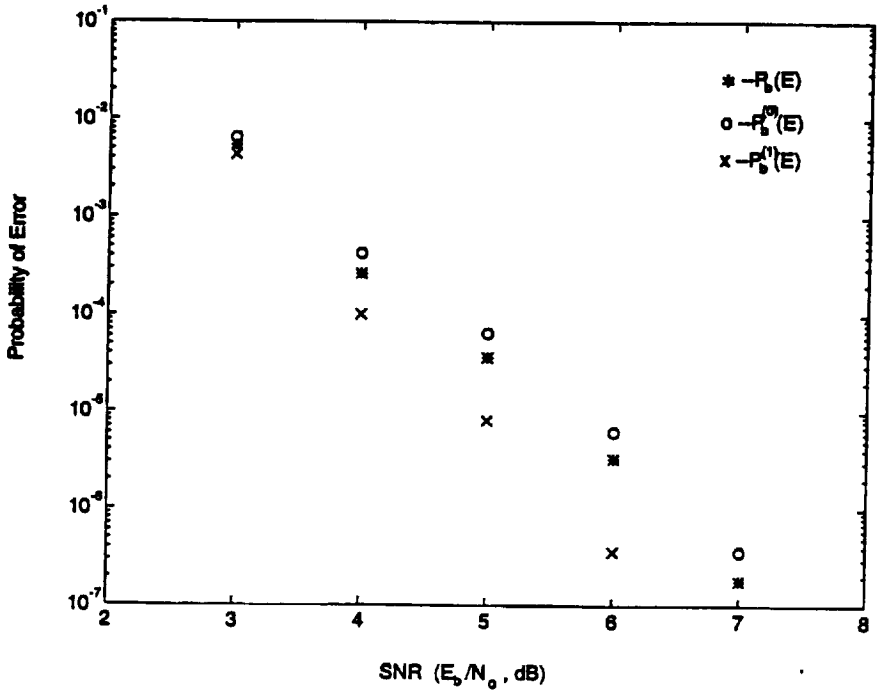


Figure 4.10: BER plot for $R = 2/3$, $\mathbf{M} = (2, 2)$ encoder with $\mathbf{d} = (4, 6)$

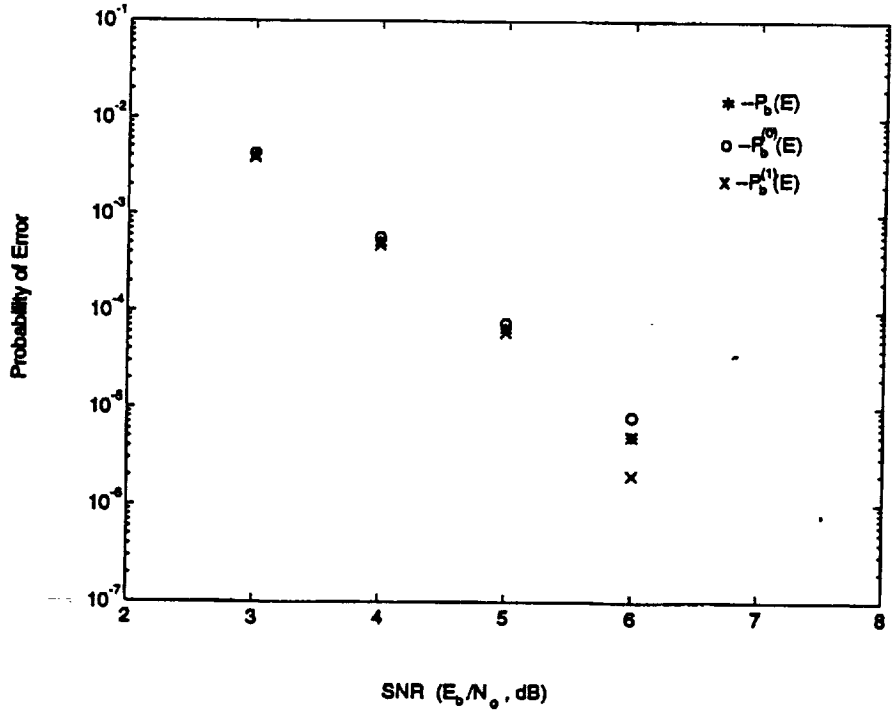


Figure 4.11: BER plot for $R = 2/3$, $\mathbf{M} = (1, 3)$ encoder with $\mathbf{d} = (5, 5)$

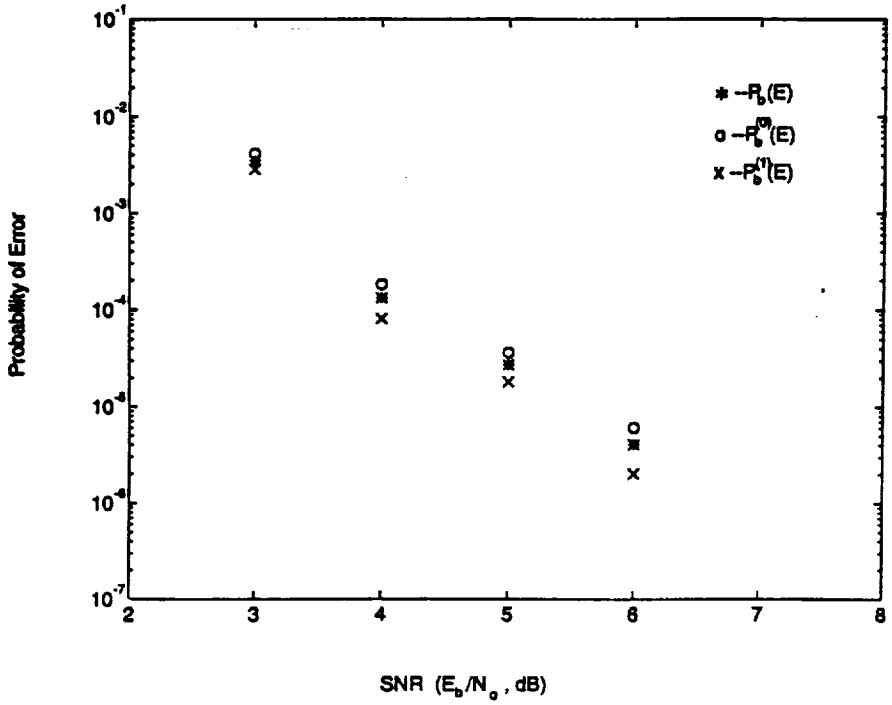


Figure 4.12: BER plot for $R = 2/3$, $\mathbf{M} = (1, 4)$ encoder with $\mathbf{d} = (4, 9)$

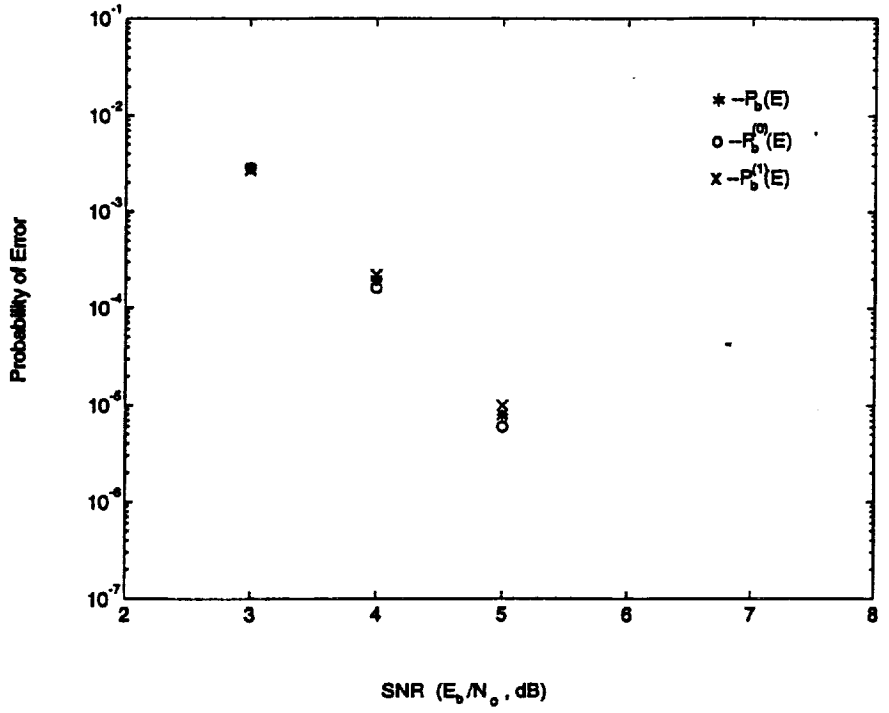


Figure 4.13: BER plot for $R = 2/3$, $\mathbf{M} = (2, 3)$ encoder with $\mathbf{d} = (6, 6)$

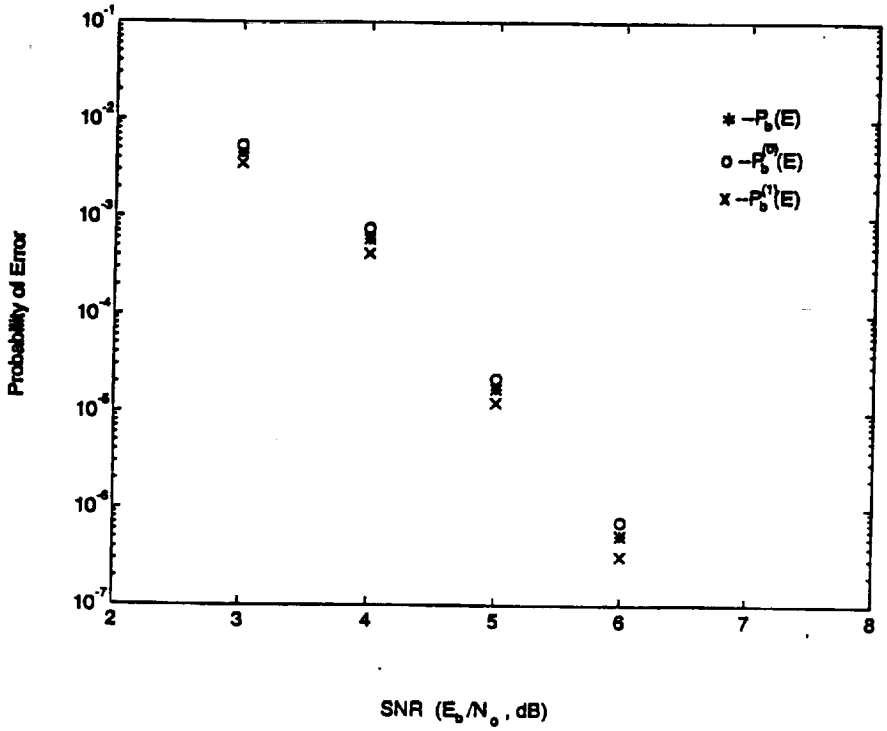


Figure 4.14: BER plot for $R = 2/3$, $\mathbf{M} = (2, 3)$ encoder with $\mathbf{d} = (5, 6)$

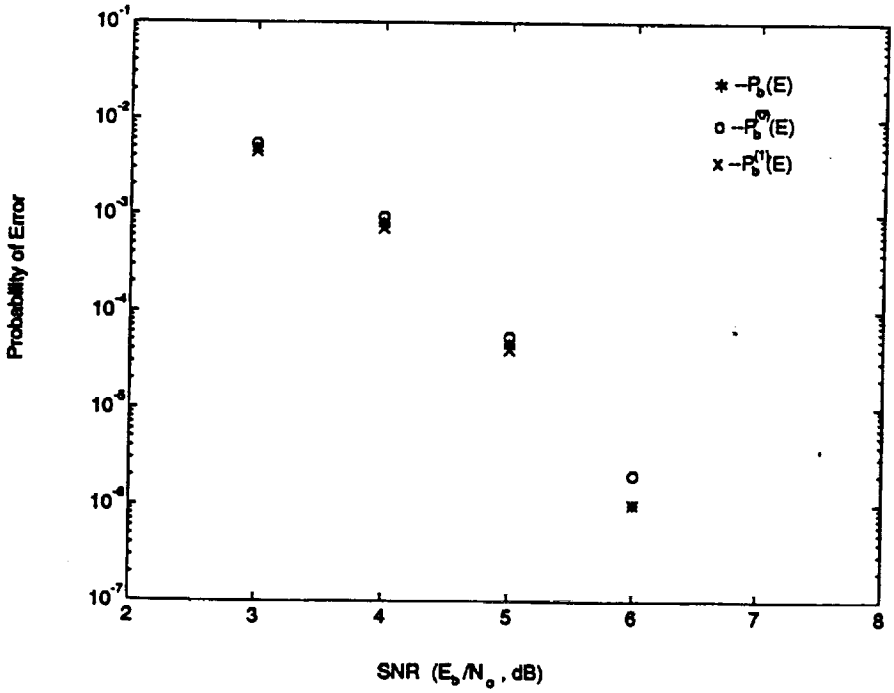


Figure 4.15: BER plot for $R = 2/3$, $\mathbf{M} = (2, 3)$ encoder with $\mathbf{d} = (4, 6)$

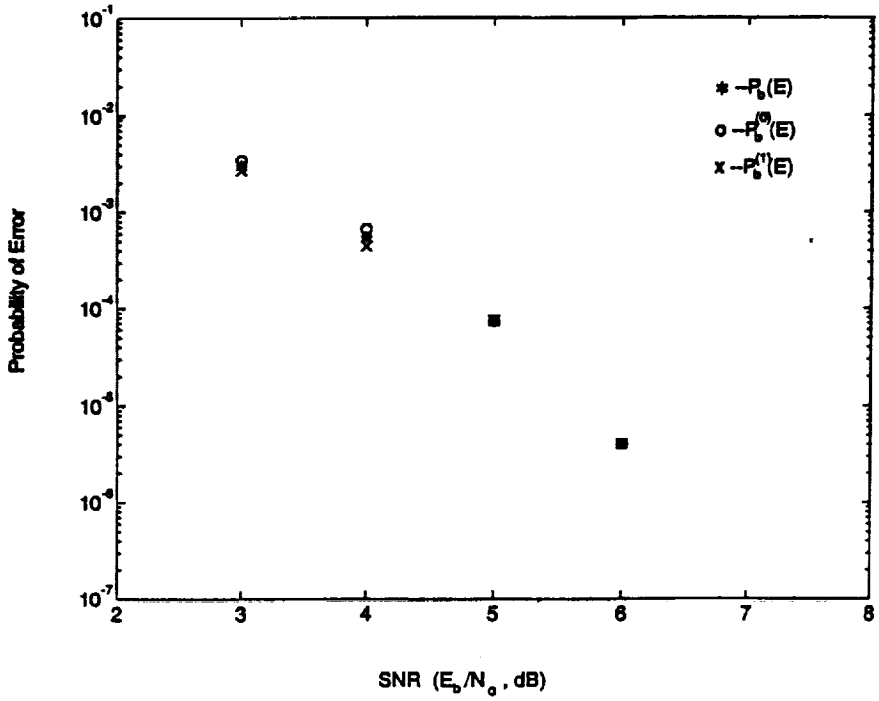


Figure 4.16: BER plot for $R = 2/4$, $\mathbf{M} = (1, 1)$ encoder with $\mathbf{d} = (5, 5)$

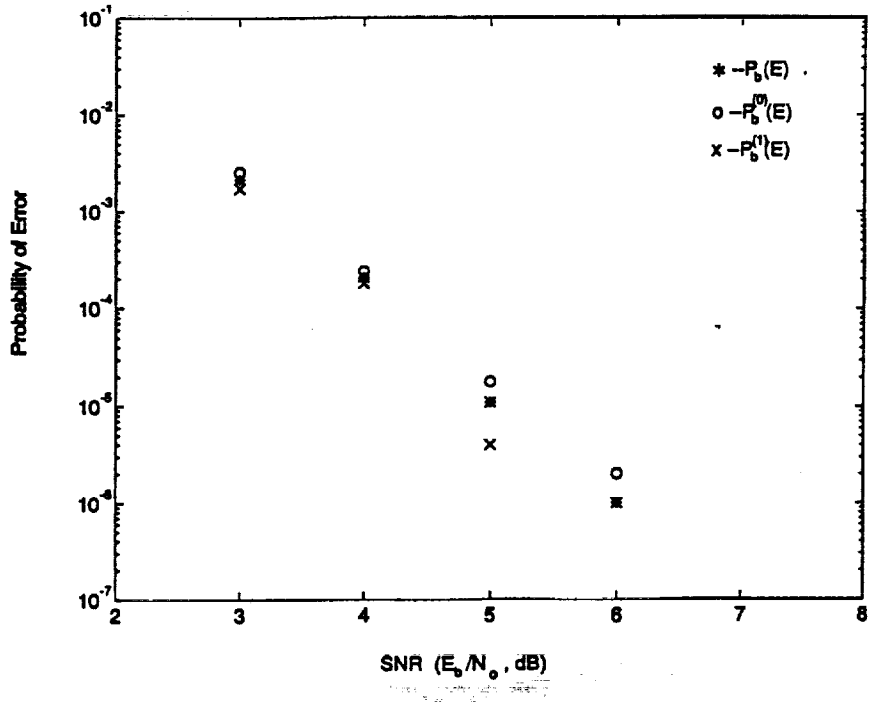


Figure 4.17: BER plot for $R = 2/4$, $\mathbf{M} = (1, 2)$ encoder with $\mathbf{d} = (6, 7)$

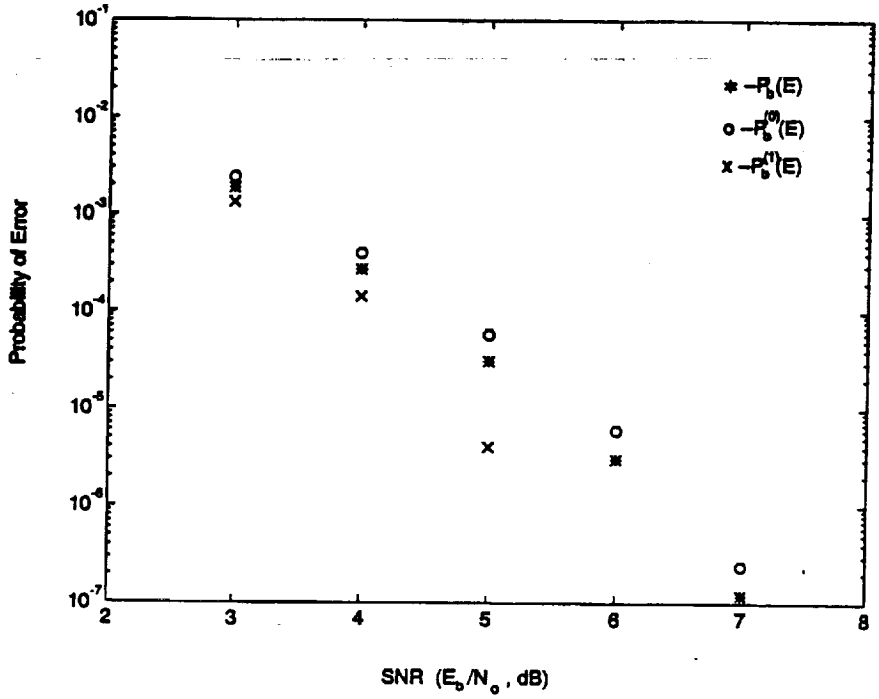


Figure 4.18: BER plot for $R = 2/4$, $\mathbf{M} = (1, 2)$ encoder with $\mathbf{d} = (5, 7)$

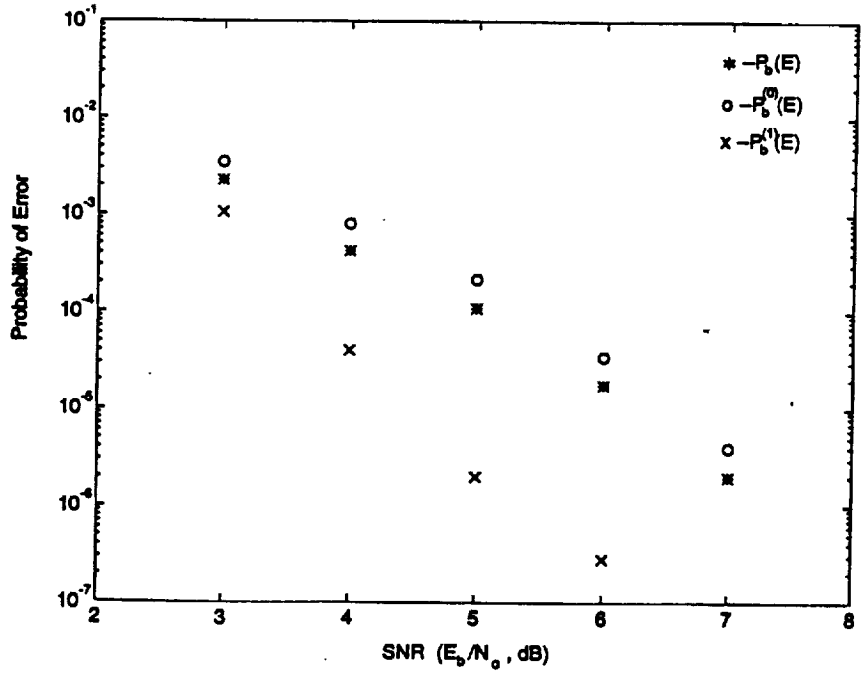


Figure 4.19: BER plot for $R = 2/4$, $\mathbf{M} = (1, 2)$ encoder with $\mathbf{d} = (4, 8)$

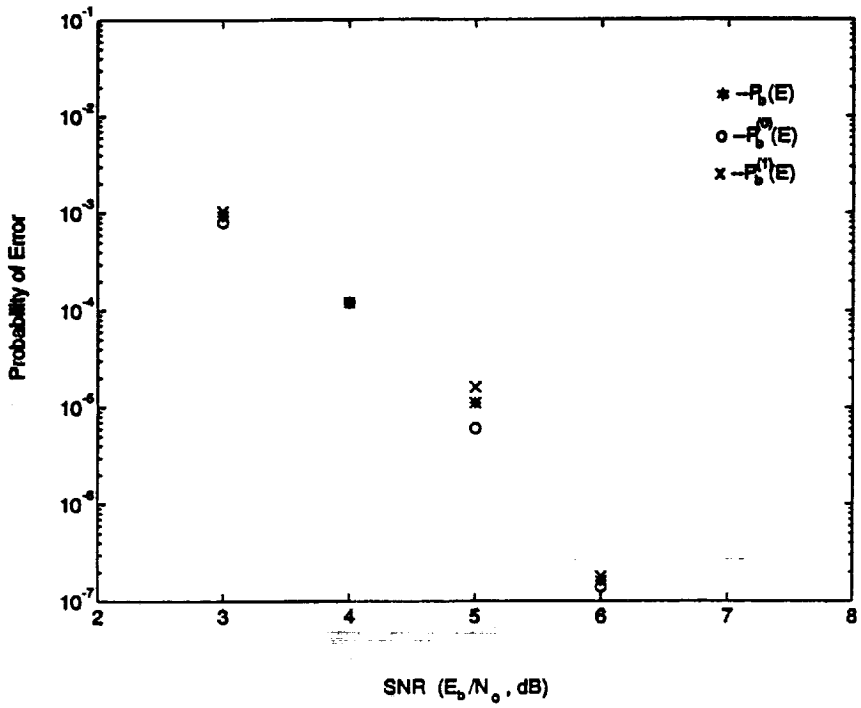


Figure 4.20: BER plot for $R = 2/4$, $\mathbf{M} = (2, 2)$ encoder with $\mathbf{d} = (8, 8)$

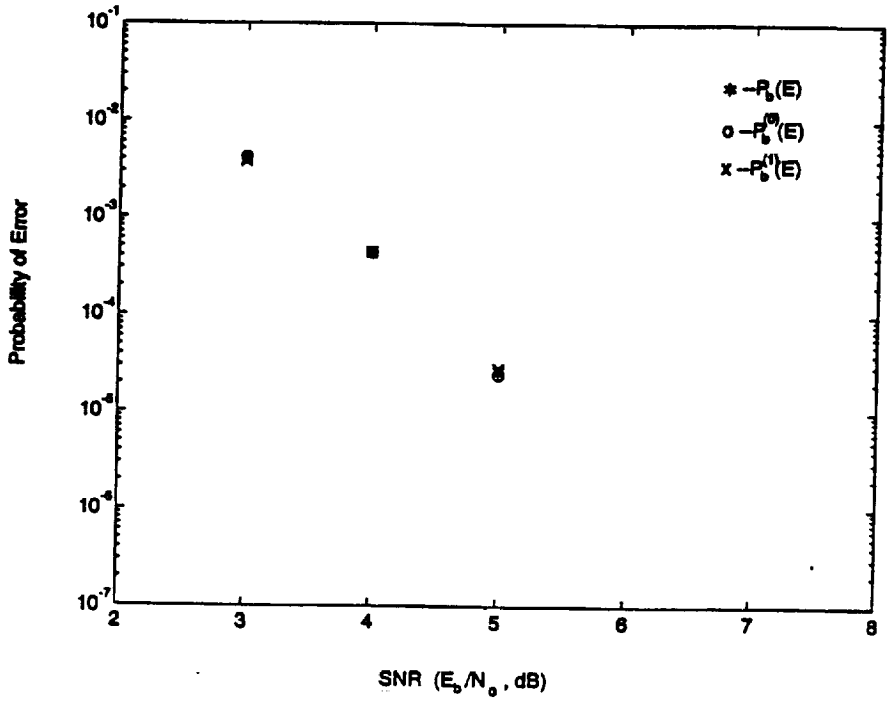


Figure 4.21: BER plot for $R = 2/4$, $\mathbf{M} = (2, 2)$ encoder with $\mathbf{d} = (7, 8)$

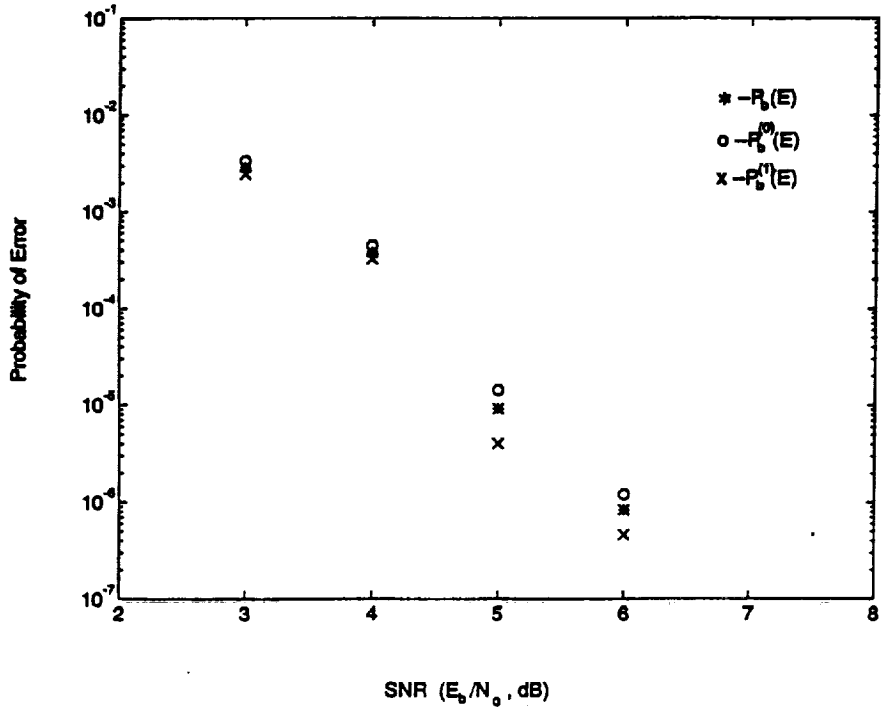


Figure 4.22: BER plot for $R = 2/4$, $\mathbf{M} = (1, 3)$ encoder with $\mathbf{d} = (6, 8)$

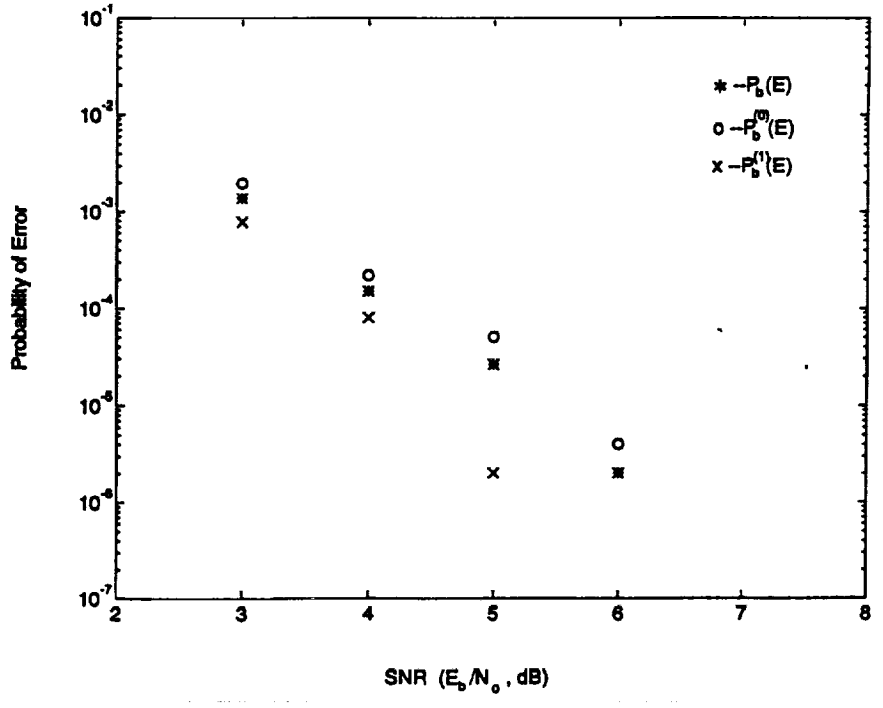


Figure 4.23: BER plot for $R = 2/4$, $\mathbf{M} = (1, 3)$ encoder with $\mathbf{d} = (5, 7)$

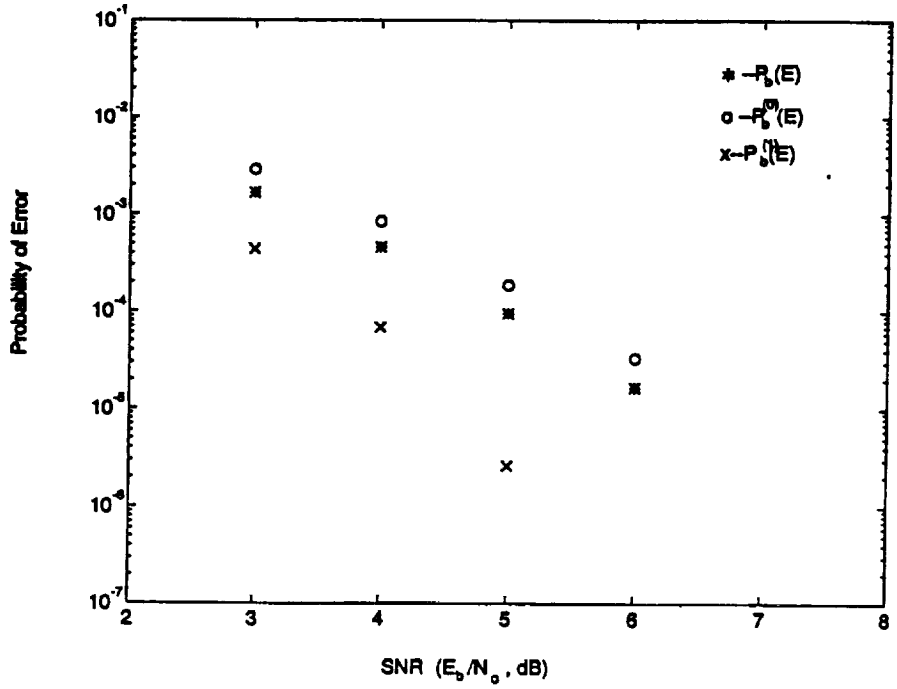


Figure 4.24: BER plot for $R = 2/4$, $\mathbf{M} = (1, 3)$ encoder with $\mathbf{d} = (4, 9)$

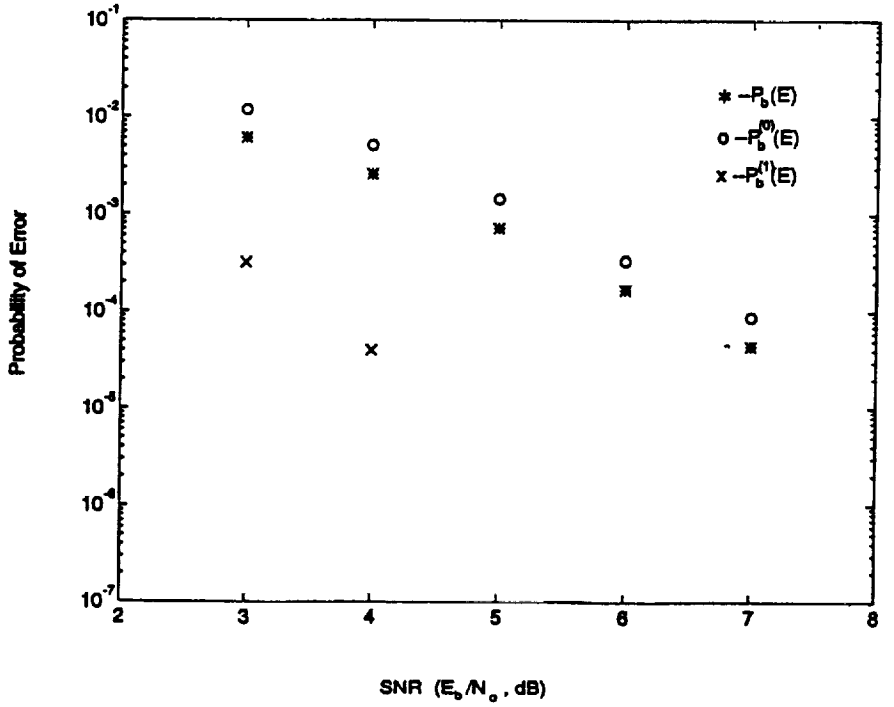


Figure 4.25: BER plot for $R = 2/4$, $\mathbf{M} = (1, 3)$ encoder with $\mathbf{d} = (3, 10)$

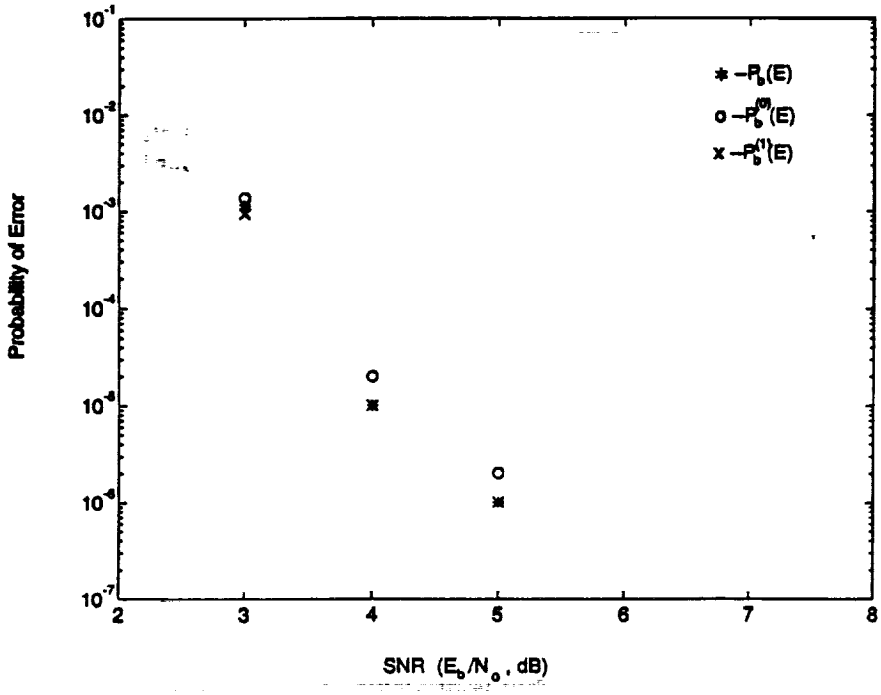


Figure 4.26: BER plot for $R = 2/4$, $\mathbf{M} = (1, 4)$ encoder with $\mathbf{d} = (7, 8)$

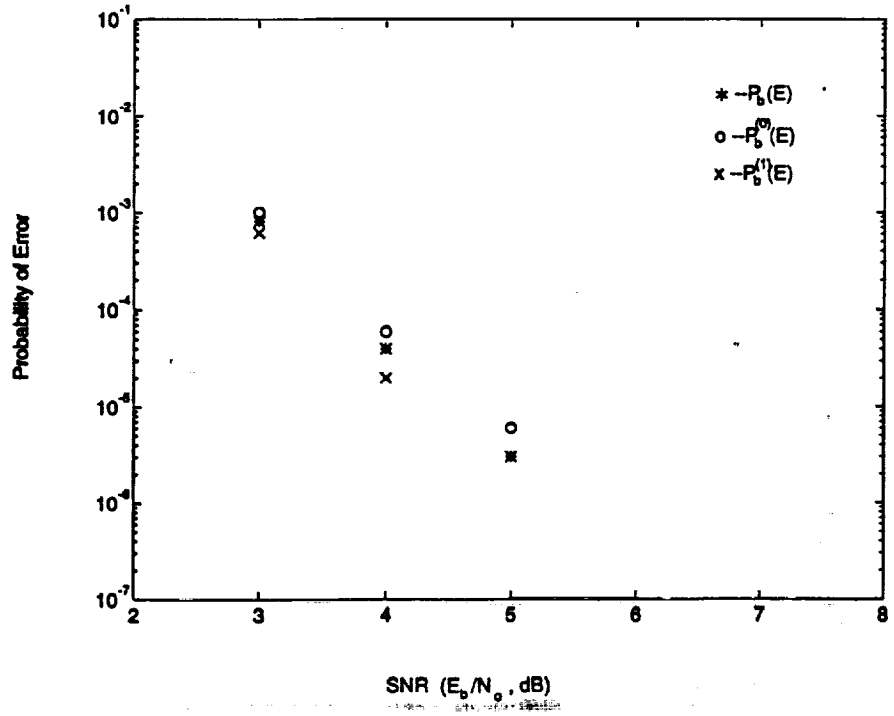


Figure 4.27: BER plot for $R = 2/4$, $\mathbf{M} = (1, 4)$ encoder with $\mathbf{d} = (6, 10)$

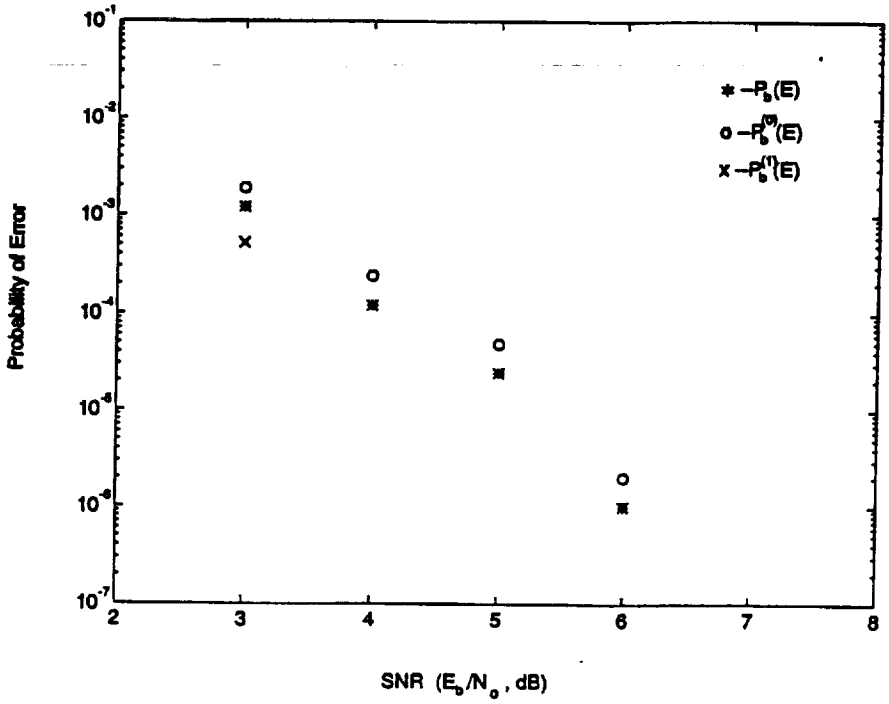


Figure 4.28: BER plot for $R = 2/4$, $\mathbf{M} = (1, 4)$ encoder with $\mathbf{d} = (5, 9)$

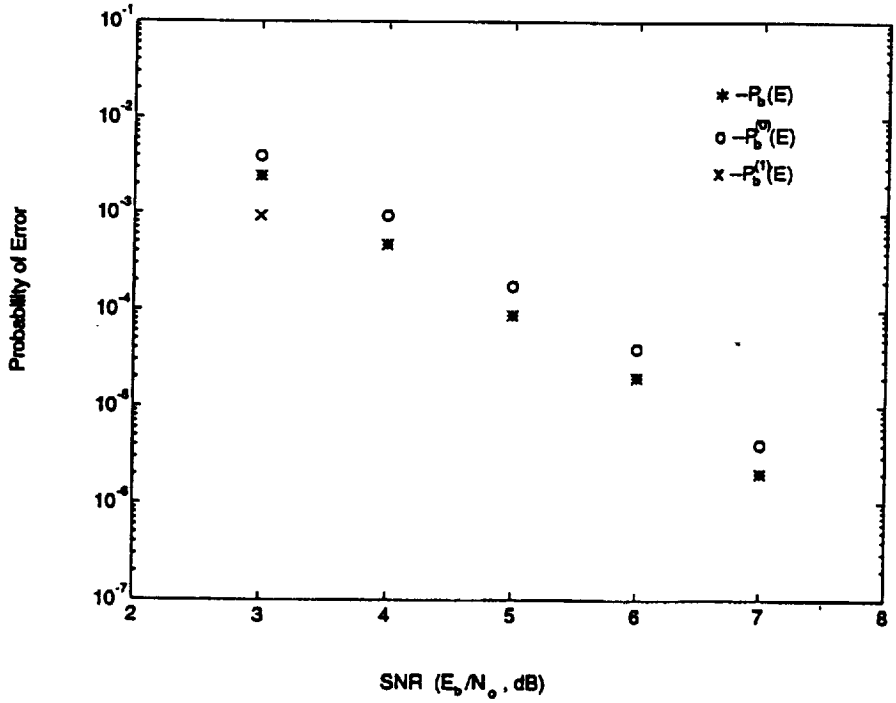


Figure 4.29: BER plot for $R = 2/4$, $\mathbf{M} = (1, 4)$ encoder with $\mathbf{d} = (4, 10)$

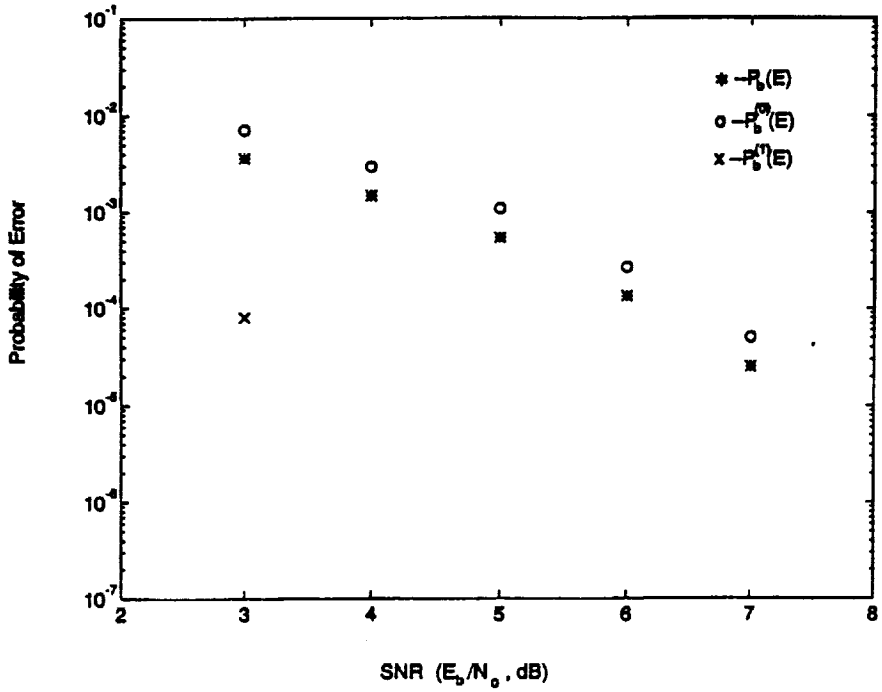


Figure 4.30: BER plot for $R = 2/4$, $\mathbf{M} = (1, 4)$ encoder with $\mathbf{d} = (3, 12)$

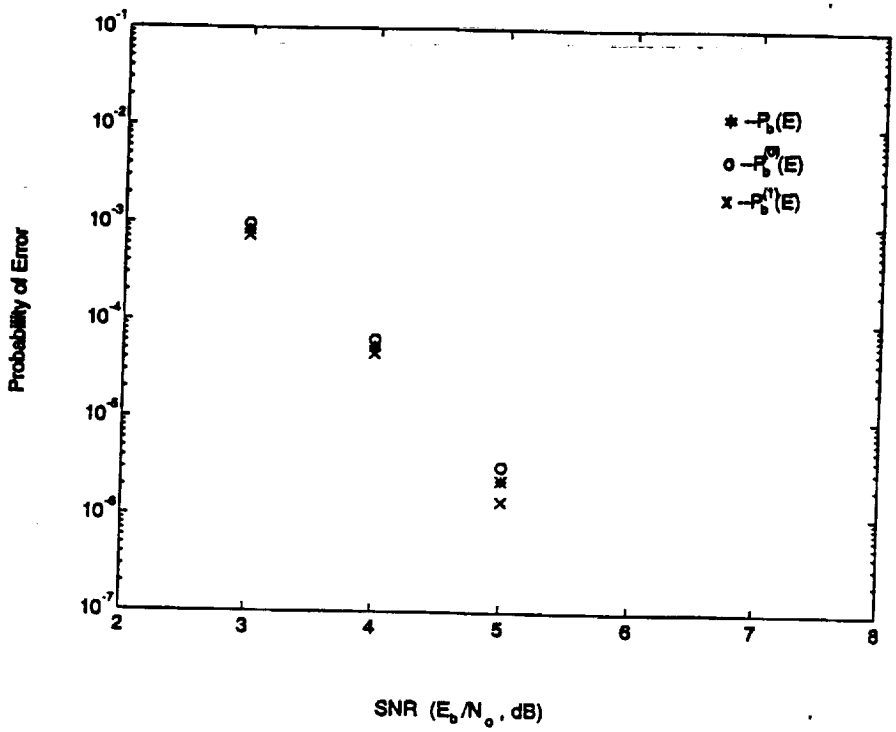


Figure 4.31: BER plot for $R = 2/4$, $\mathbf{M} = (2, 3)$ encoder with $\mathbf{d} = (8, 9)$

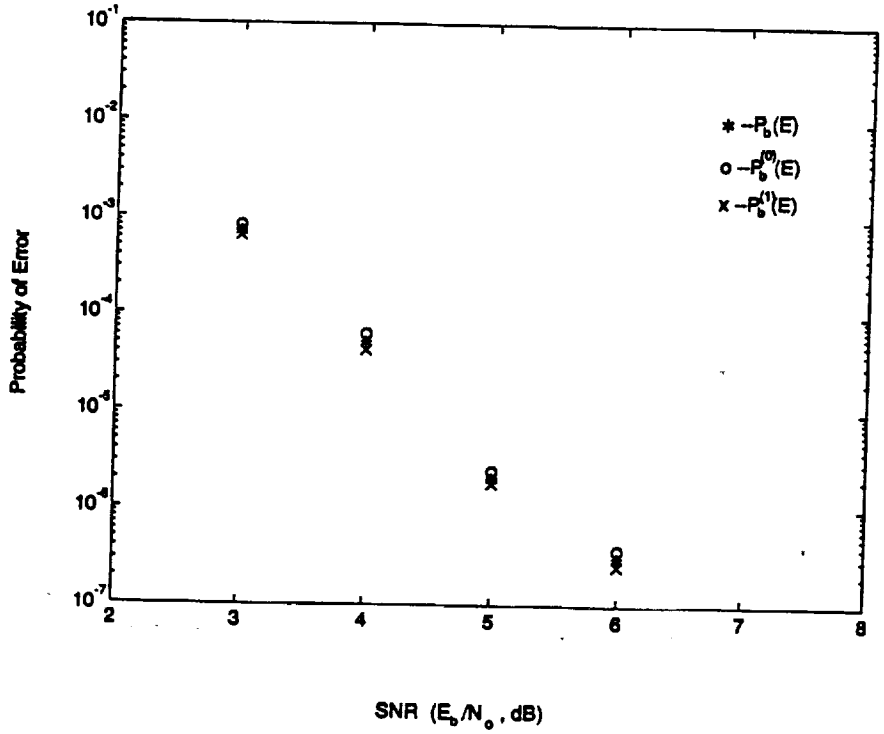


Figure 4.32: BER plot for $R = 2/4$, $\mathbf{M} = (2, 3)$ encoder with $\mathbf{d} = (7, 8)$

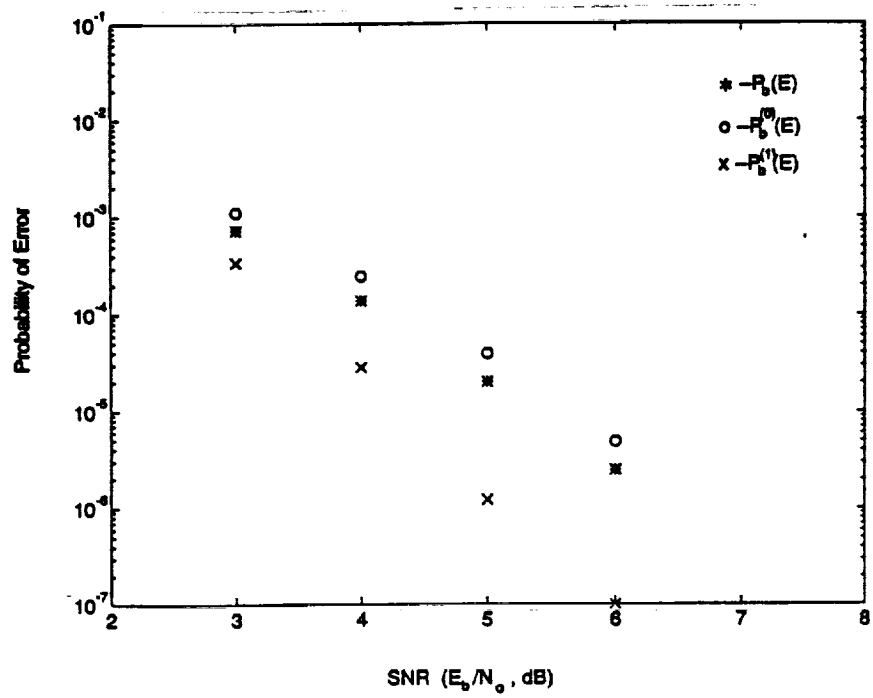


Figure 4.33: BER plot for $R = 2/4$, $\mathbf{M} = (2, 3)$ encoder with $\mathbf{d} = (5, 9)$

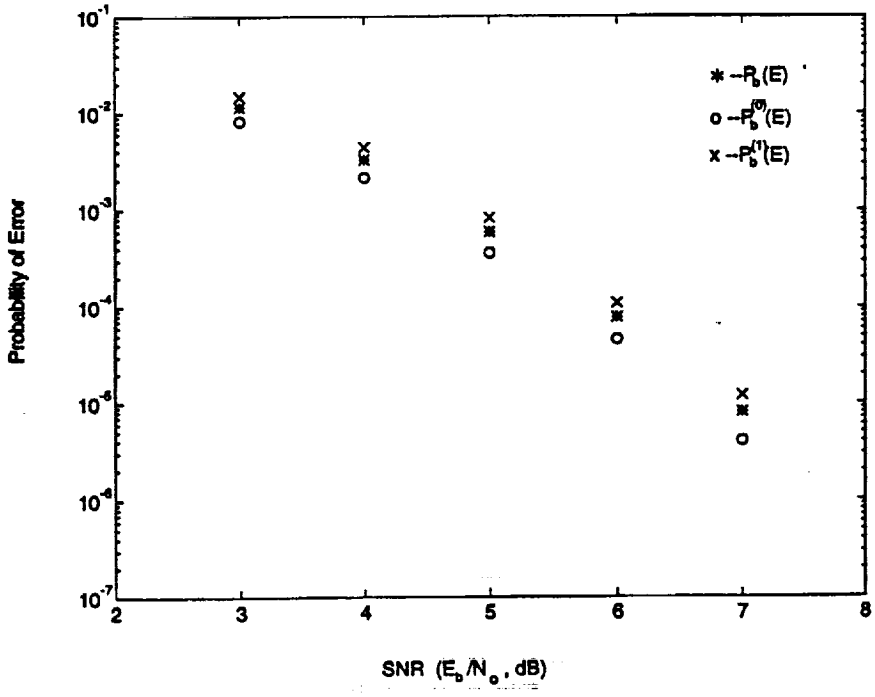


Figure 4.34: BER plot for optimal $R = 2/3$, $\mathbf{M} = (1, 1)$ encoder with $\mathbf{d} = (3)$

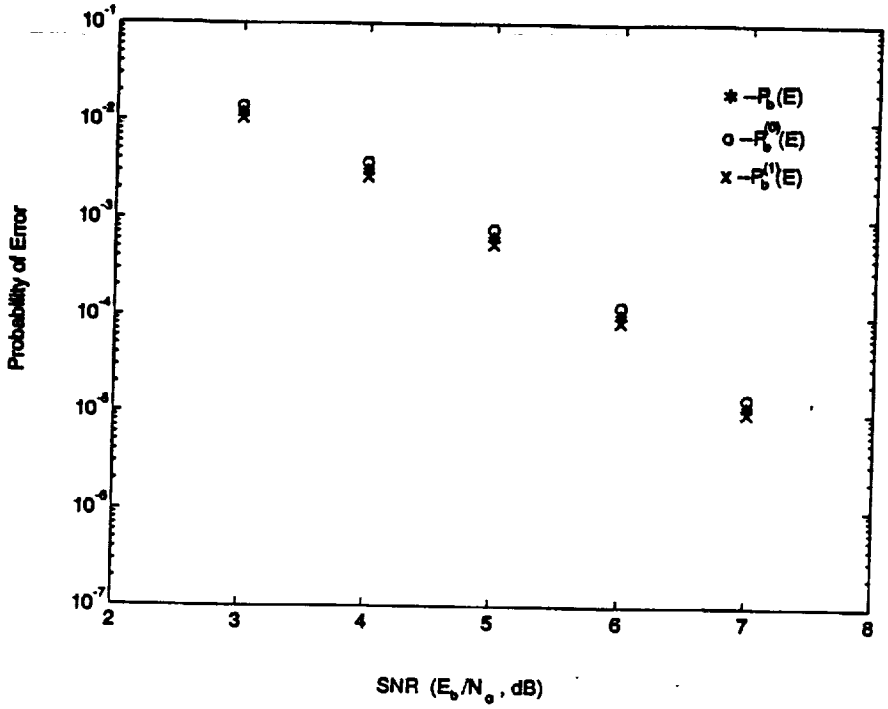


Figure 4.35: BER plot for optimal $R = 2/3$, $\mathbf{M} = (1, 2)$ encoder with $\mathbf{d} = (4)$

4.9 Summary

In this chapter, we examined the unequal error protection capabilities of convolutional codes. The effective free distance vector is presented and defined as a measure of the unequal error protection. Also, a modified transfer function for convolutional encoders from which the unequal error protection capabilities of a code can be calculated was defined. Several bounds on the unequal error protection capabilities of convolutional encoders were derived and discussed, and the results of searches for UEP codes were presented. It was shown that convolutional encoders can provide

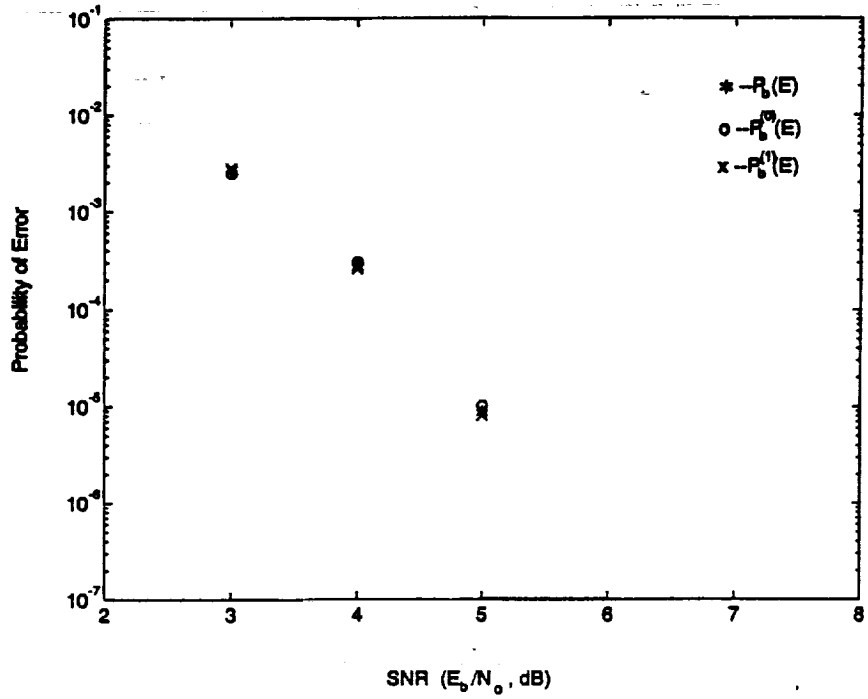


Figure 4.36: BER plot for optimal $R = 2/3$, $\mathbf{M} = (2, 2)$ encoder with $\mathbf{d} = (5)$

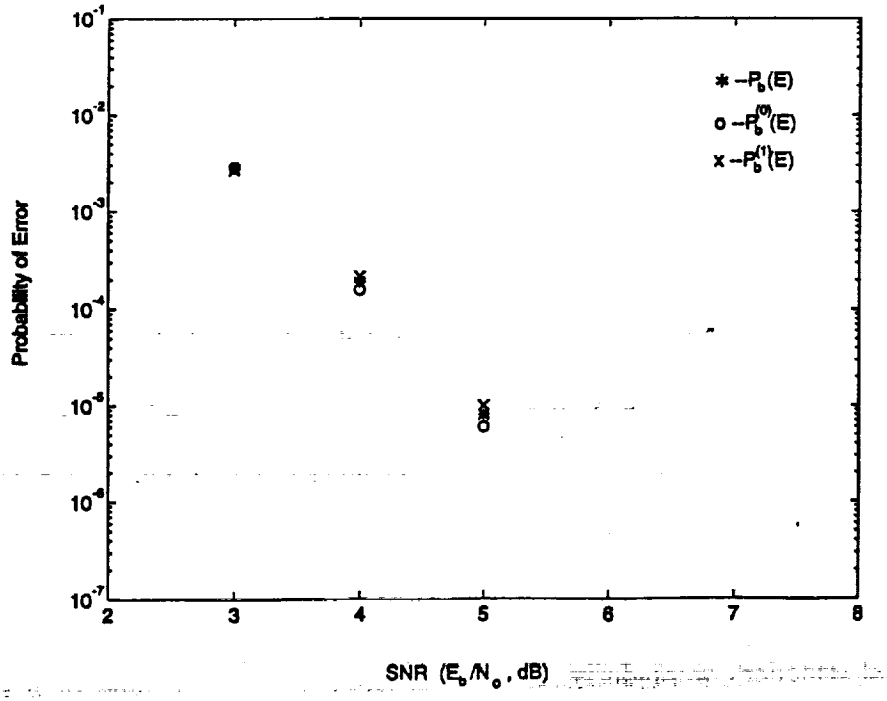


Figure 4.37: BER plot for optimal $R = 2/3$, $\mathbf{M} = (2, 3)$ encoder with $\mathbf{d} = (6)$

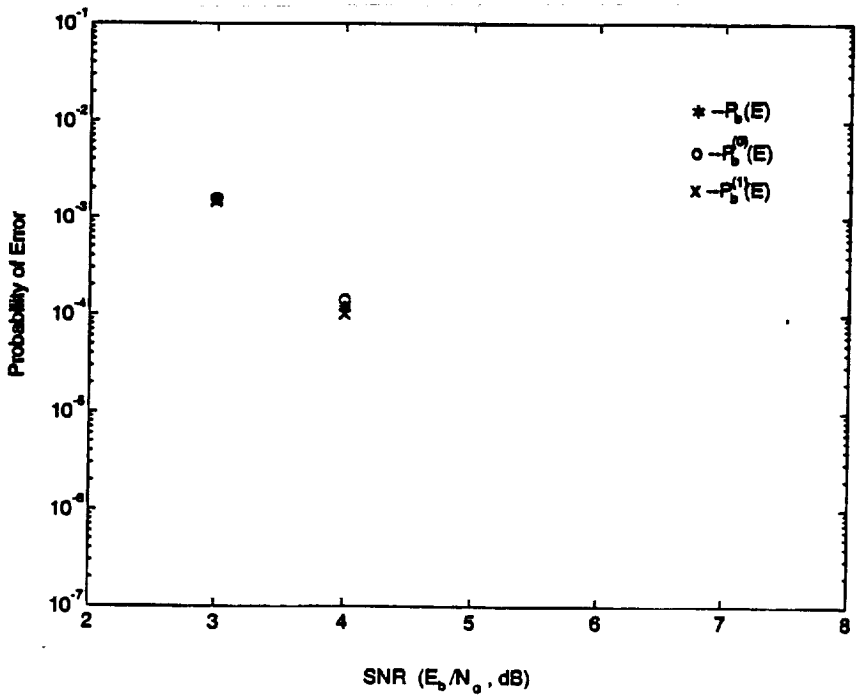


Figure 4.38: BER plots for optimal $R = 2/3$, $\mathbf{M} = (3, 3)$ encoder with $\mathbf{d} = (7)$

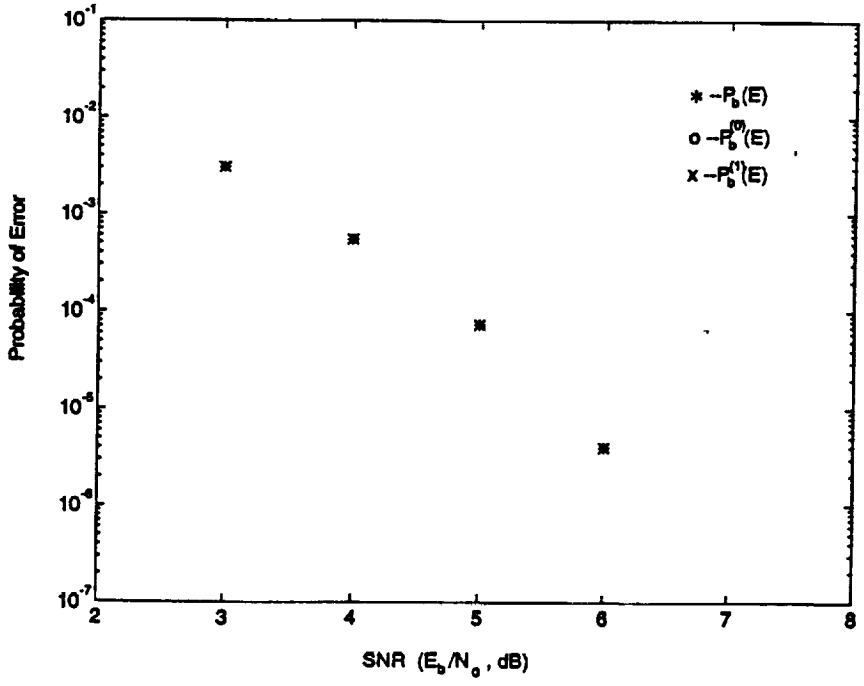


Figure 4.39: BER plots for optimal $R = 1/2$, $\mathbf{M} = (2)$ encoder with $\mathbf{d} = (5)$

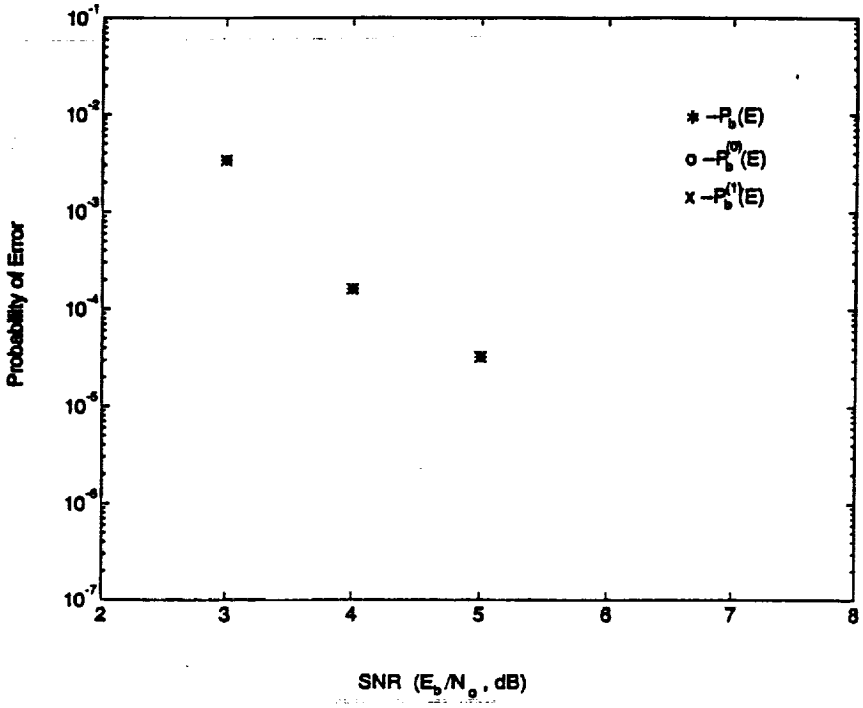


Figure 4.40: BER plots for optimal $R = 1/2$, M = (3) encoder with d = (6)

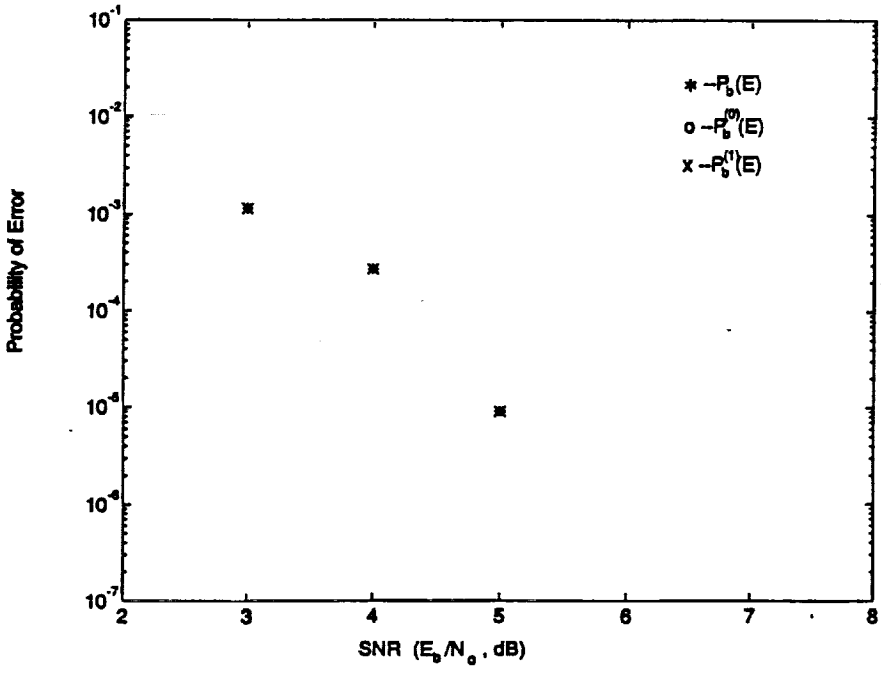


Figure 4.41: BER plots for optimal $R = 1/2$, M = (4) encoder with d = (7)

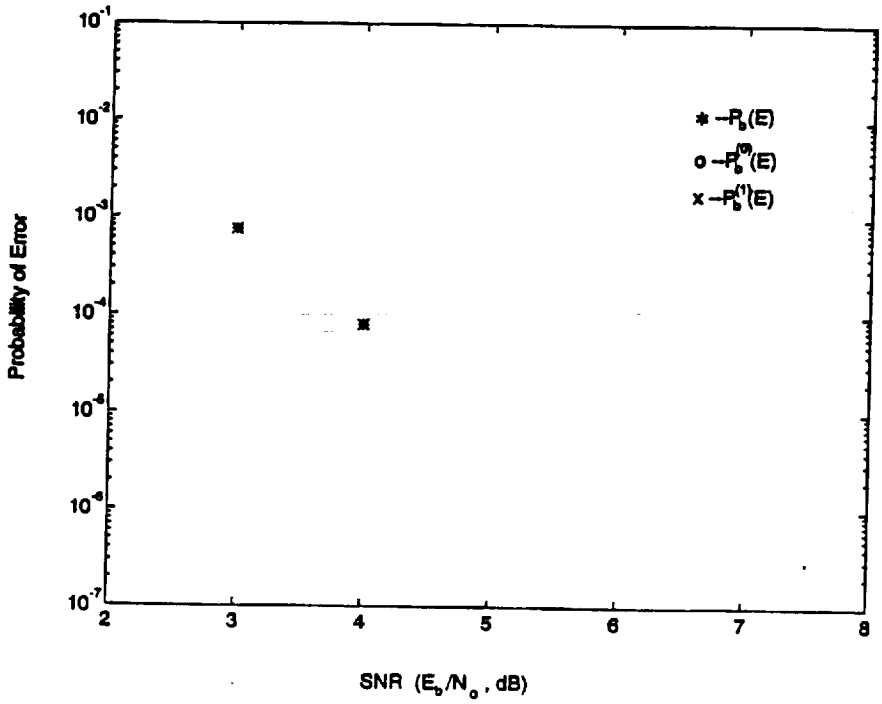


Figure 4.42: BER plots for optimal $R = 1/2$, $\mathbf{M} = (5)$ encoder with $\mathbf{d} = (8)$

unequal error protection. Several encoders which provided more protection to one information position than the protection offered by the optimal free distance encoder. The cost of the increased protection for a specific input line is typically a decrease in the protection offered to the other information bits.

C1-2

CHAPTER 5

Achieving Unequal Error Protection with Trellis Coded Modulation

5.1 Introduction

Traditional channel coding techniques achieve coding gain (i.e. reduce the required signal-to-noise ratio for a specified error probability) at the expense of the required bandwidth. For instance, a rate 1/2 code doubles the bandwidth relative to an uncoded transmission. Bandwidth expansion is often not possible on band-limited channels. Trellis coded modulation (TCM) was developed as a bandwidth efficient means to achieve coding gain [60].

Trellis coded modulation is a combined modulation and coding technique that can realize coding gains without increasing the required bandwidth. A finite state encoder, such as a convolutional encoder, determines the selection of the modulation signals and generates coded signal sequences. This chapter discusses the unequal error capabilities of TCM. Section 5.2 briefly review modulation techniques and describes trellis coded modulation. Search results for unequal error protection TCM codes are presented in Section 5.3.

5.2 Trellis Coded Modulation

Modulators convert a discrete signal into an analog waveform, for transmission over a channel. During each signaling interval, the modulator maps k bits into one of $M = 2^k$ possible channel signals. The demodulation receives a corrupted version of the transmitted channel signal, and estimates the m original bits by choosing the channel signal which was closest to the received signal and performing the inverse mapping.

Signal set representations for two amplitude modulation schemes are shown in Figures 5.1; two PSK signal constellations are shown in Figure 5.2. If the average signal energy is held constant, the signal points move closer together as the size of the signal set increases. Because a maximum likelihood demodulator chooses the channel signal closest to the received signal, more opportunities for errors occur for the larger signal set.

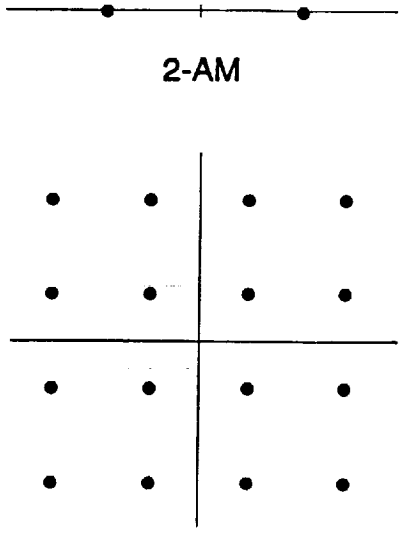
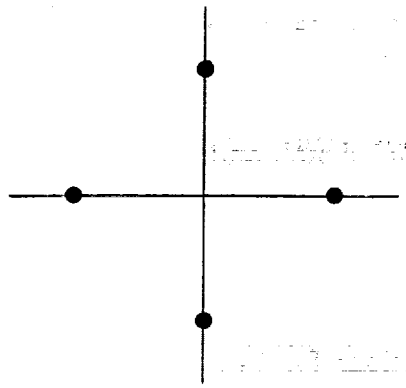
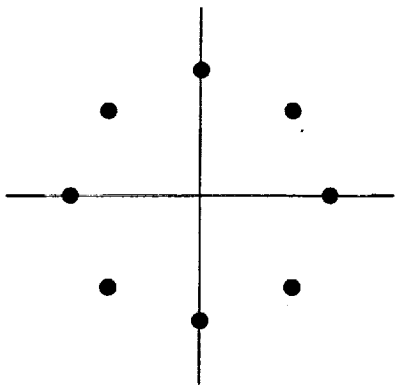


Figure 5.1: Amplitude Modulation Signal Sets



4-PSK



8-PSK

Figure 5.2: Phase Shift Keying Signal Sets

Modulation and error-correction coding are performed independently, the results are mediocre. Consider the example presented in [61]. Both the 4-PSK modulation system without coding and the 8-PSK system with independent rate 2/3 convolutional coding transmit two information bits per modulation interval. The 8PSK system has a BER exceeding 10^{-2} when it is operated at the signal-to-noise ratio (SNR) for which the 4-PSK system exhibits a BER of 10^{-5}. The increased error rate is due to closer signal points in the 8 PSK constellation. The rate 2/3 code requires a state complexity of $K = 6$ to reduce the error rate of the 8PSK system to 10^{-5}. The convolutional code require a 64-state Viterbi decoder, which is fairly complex. That is, the 8PSK coded system is more complex and transmits no more reliably than the simpler 4PSK uncoded system. The difficulty in developing a simple, reliable system with independent coding and modulation led to the development of TCM.

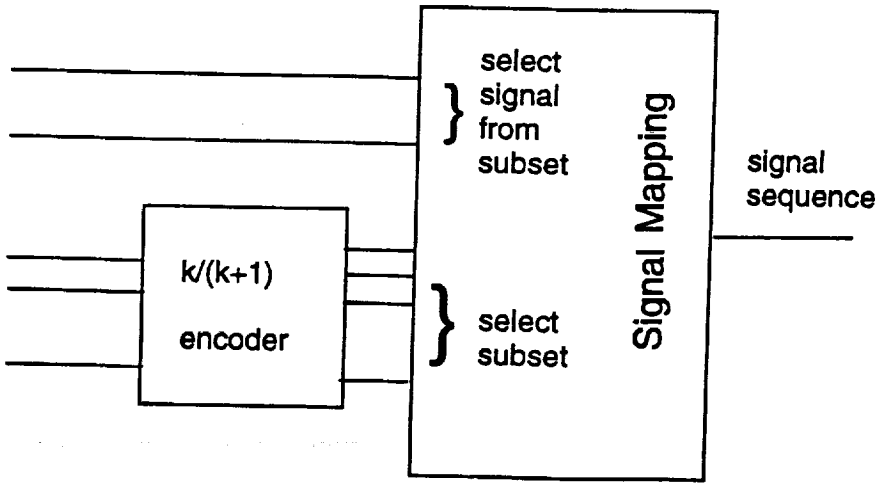


Figure 5.3: A General TCM System

Trellis coded modulation is a combined modulation and coding technique that can realize coding gains without increasing the required bandwidth. A finite state encoder, such as a convolutional encoder, determines the selection of the modulation signals and generates coded signal sequences. Figure 5.3 shows a general block diagram for a

TCM system which doubles the signal set. There are k information bits entering the system, of which $\tilde{k}$ bits enter the rate $\tilde{k}/(\tilde{k}+1)$ convolutional encoder, and $(k-\tilde{k})$ remain uncoded.

Consider the rate 2/3 TCM coded 8PSK system with a 4-state, rate 1/2 convolutional encoder, which will be compared to an uncoded 4 PSK system. The 4 PSK signal space is shown in Figure 5.2; the labelled 8PSK signal space is shown in Figure 5.4.

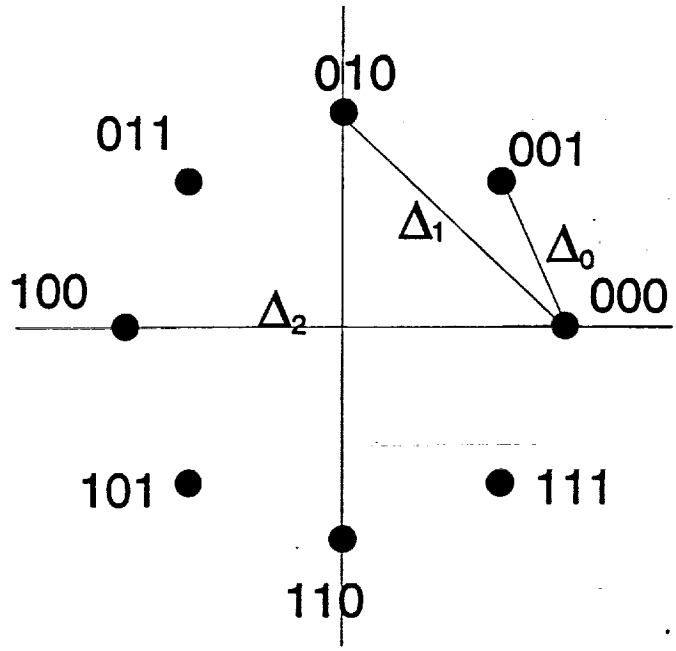


Figure 5.4: A Labeled 8PSK Signal Set

It is possible to partition the signal set, which involves repeatedly dividing the signal set (or subset) into two smaller subsets with a larger smallest intra-set distance, δ_i, where i is the number of times the partitioning has been conducted. . The partitioning for the 8PSK constellation is shown in Figure 5.5. The coded bits produced by the encoder select a subset, while the uncoded bit chooses a signal from the selected subset. The free Euclidean distance, d_E of a TCM code is defined as the

minimum Euclidean distance between any two valid signal sequences, or

$$d_E^2 = \min[\sum_n |y_n - y'_n|^2], \tag{5.1}$$

where y_n is the n^{th} channel signal in the sequence.

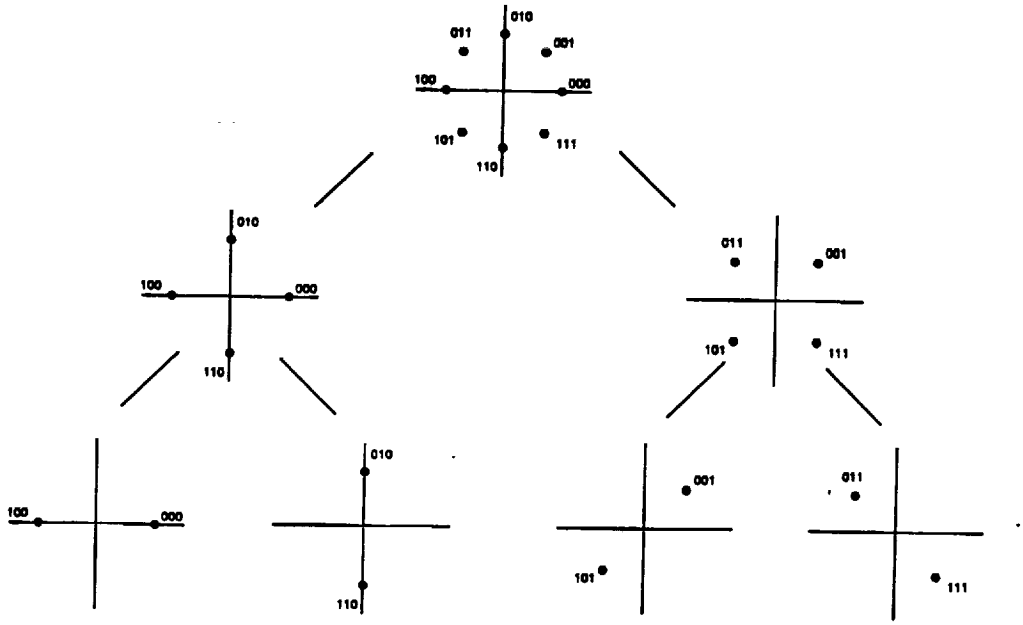


Figure 5.5: Partitioning an 8PSK Signal Set

Assume the convolutional encoder has the trellis diagram shown in Figure 5.6. Then the trellis for the TCM code is aas shown in Figure 5.7. Parallel transitions occur on the trellis due to the uncoded input bit, and limit the free distance of the code. To offset this limitation, the signals associated with parallel transitions are assigned to the signal points that are most distant. In fact, the branch labels of the trellis for the convolutional code are assigned as signal set labels according to the three rules:

a) Parallel transitions are associated with signals with the maximum Euclidean distance between them,

b) Transitions exiting or entering a node in the trellis are assigned to signal points with the next largest Euclidean distance between them, and

c) All signal points are used with equal frequency.

The signal labels in Figure 5.4 were assigned according to the above rules. The signals assigned to the parallel transitions have Euclidean distance 2. On the other hand, any two signal sequences that have diverging paths in the trellis have at least Euclidean distance $\sqrt{\Delta_0^2 + \Delta_1^2 + \Delta_0^2} = \sqrt{\Delta_0^2 + \Delta_2^2} = \sqrt{4.585}$. Therefore, the free Euclidean distance of the code is 2. Since the free distance of the uncoded system in this example is $\sqrt{2}$, the coded TCM system has a coding gain, in decibel (dB), of $10\log(\frac{2^2}{(\sqrt{2})^2}) = 3$ dB.

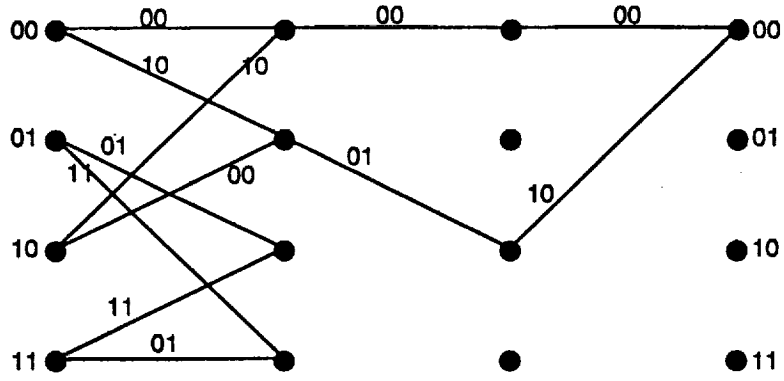


Figure 5.6: A 4-State Binary Trellis

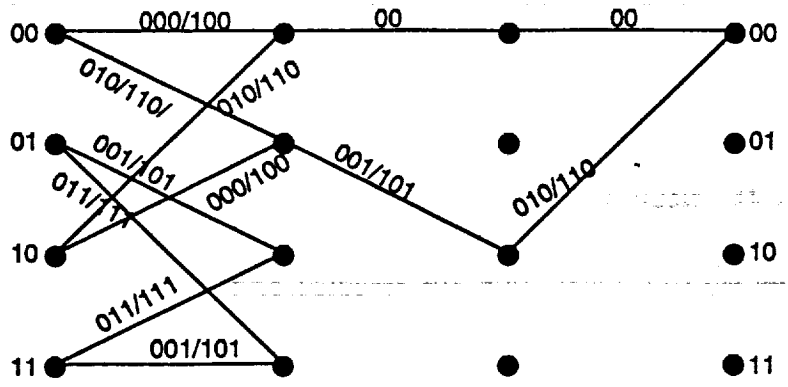


Figure 5.7: A 4-State TCM Trellis

In general, the free Euclidean distance of a TCM code is the minimum of the minimum distance between parallel transitions and the minimum distance between

non-parallel paths through the trellis. Because linearity does not hold for TCM code sequences, the Euclidean distance between every pair of TCM signal sequences must be computed to determine the free distance if an unmodified trellis is used for the calculations. However, the reference path may be the all zero path if each binary branch label on the trellis is replaced with the minimum squared distance between all pairs of signal points with binary labels which differ by the value of the original branch label. With the modification, algorithms which are used to evaluate binary linear trellises may be used to determine the free Euclidean distance of a TCM code. This modification significantly simplifies the analysis of a TCM code.

5.3 Unequal Error Protection with Trellis Coded Modulation

When considering UEP with TCM, the free Euclidean distance vector is similar to the effective free distance vector for binary codes. The free distance vector for TCM is $\mathbf{d}_E = (d_{E1}, \ldots, d_{E2})$, where d_{Ej} is the minimum Euclidean distance between any two code sequences with input sequences that differ in the j^{th} position. Combining the branch label modifications presented in Section 4.3 and Section 5.2, the free Euclidean distance vector of a specific encoder may be calculated with the same algorithms that calculate free distance and the effective free distance vector.

The results for binary encoders may be used as guidelines for expected results for the TCM. A TCM code is represented by a trellis that is topologically identical to the binary trellis at the same rate and memory distribution. If the branches of the TCM trellis are to be labelled with the minimum distances associated with the all signal pairs with labels that differ by the encoder branch output, then the only differences between the two trellises are the weights assigned to the branches.

Furthermore, a given signal constellation, with a specified labeling or mapping, has a unique ordered set of minimum distances that are placed on the trellis. For example, consider the 8PSK constellation with natural mapping, shown in Figure 5.4. The minimum squared distances associated with the branch labels of the trellis are:

label	minimum squared branch distance
000	0.000
001	0.586
010	0.586
011	2.000
100	0.586
101	2.000
110	4.000
111	3.414

Therefore, the ordered set of the minimum squared distances which replace the binary branch labels is {0.000, 0.586, 0.586, 0.586, 2.000, 3.414, 3.414, 4.000}.

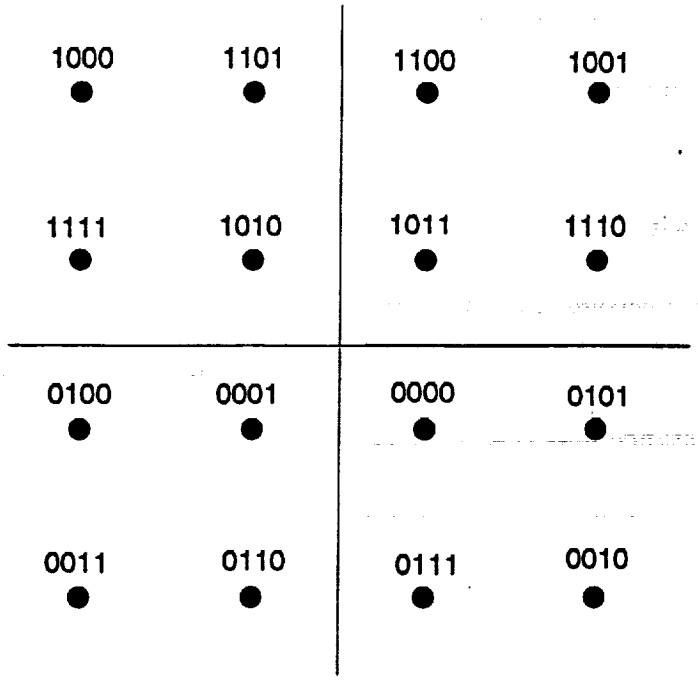


Figure 5.8: A Labeled 16QAM Signal Set

Similarly, the 16QAM with standard mapping, shown in Figure 5.8 has the mini-

mum squared branch distances

label	minimum squared branch distance
0000	0
0001	1
0010	2
0011	5
0100	4
0101	1
0110	2
0111	1
1000	8
1001	5
1010	2
1011	1
1100	4
1101	5
1110	2
1111	5

so the ordered set of branch distances is {0, 1, 1, 1, 1, 2, 2, 2, 2, 4, 4, 5, 5, 5, 5, 8}.

The ordered set of branch distances for a binary encoder is equal to the (ordered) Hamming weights of the binary branch labels. So then, the ordered set of distances for a rate $k/3$ binary encoder is $\{0, 1, 1, 1, 2, 2, 2, 3\}$, and the ordered set of branch distances for a rate $k/4$ binary encoder is $\{0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4\}$.

The maximum possible free Euclidean distance vector for a specific TCM code with a certain signal mapping and memory distribution may be calculated from the trellis of the optimal UEP binary convolutional encoder of the same configuration. The free Euclidean distance vector for a specific TCM is no larger than the effective distance vector of the optimal UEP binary encoder, evaluated with its branch distances replaced by the minimum mapping distances of the signal set that occupy the same relative place in the ordered set of distances. That is, the ordered branch distances for the labelled constellation are aligned with the ordered branch distances

for the binary encoder, and then are assigned as the branch distances on the the trellis. When the trellis for the optimal UEP binary encoder is evaluated with the new branch labels, the effective squared Euclidean distance of the system with the most advantageous branch labeling is found. The resulting free Euclidean distance vector is not necessarily achievable, nor is necessarily achievable with the encoder that is the optimal binary encoder. Instead, the resulting effective distance vector is the distance that would be achieved for a constellation mapping with the given set of squared Euclidean branches, mapped so that the smallest squared Euclidean distance is aligned with the smallest Hamming weight, etc. It is possible that with the same constellation mapping, a different encoder may be found that produces the same effective distance vector.

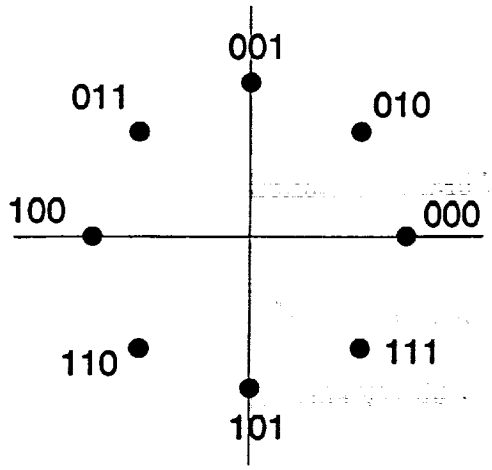


Figure 5.9: Alternative Labeling for an 8PSK Signal Set

It is hypothesized here that the natural mapping for the 8PSK signal constellation and the typical 16QAM mapping produce the largest ordered sets of branch distances among all possible mappings. That is, rearranging the mapping so that one or more of the values in the set is reduced is not accompanied by an increase in another value that offsets the original reduction. Conversely, increasing one value in the set

Rate=2/3, 8PSK		
M	uncoded bits	d_E^2
(0,1)	1	(2.586,2.586)
(1,1)	0	(3.172,3.172)
(0,2)	1	(4.0,4.586)
(0,3)	0	(4.0,5.172)
(1,2)	0	(4.586,4.586)
(1,2)	0	(4.0,5.172)
(2,2)	0	(5.172,5.172)

Table 5.1: Unequal Error Protection with 8PSK TCM

causes a decrease in another value that is at least as large in magnitude. As an example, consider the 8PSK constellation with the mapping shown in Figure 5.9. The minimum squared distance between all pairs of signals with labels that differ by 001 was increased from .586 to 2.0, when compared with the natural mapping shown in Figure 5.4. However, not only was the minimum squared distance between all pairs of signals with labels that differ by 010 decreased from 2.0 to .586, but the minimum squared distance between all pairs of signals with labels that differ by 011 was reduced to from 3.414 to .586. Non-exhaustive searches for signal mappings with a larger (componentwise) ordered set of minimum distances than the standard mappings for the 8PSK constellation or the 16QAM constellation were conducted. None were found, leading to the the conclusion that the standard mappings possess the largest ordered sets of branch distances.

The results of searches for 8PSK systems with unequal error protection are shown in Table 5.1. A criterion of the search was that at least one effective squared Euclidean distance must be larger than the corresponding effective free distance of the typical system. The components of the distance vectors listed in Table 5.1 are the squared values. Similarly the results of searches for 8PSK systems with unequal er-

Rate=2/4, 16QAM		
M	uncoded bits	d_E^2
(0,1)	0	(6,6)
	1	(5,6)
	1	(4,7)
(1,1)	0	(9,9)
	0	(8,10)
(0,2)	0	(8,10)
(1,2)	0	(11,11)
	0	(7,10)

Table 5.2: Unequal Error Protection with 16QAM TCM

ror protection are shown in Table 5.2. The components in the vectors listed for the 16QAM results are the unnormallized, squared distances.

5.4 Summary

Trellis coded modulation systems with 8PSK and 16QAM were examined as potential UEP coding systems. It was shown that the free Euclidean distance vector of a TCM system can be upper bounded if the signal mapping and the optimal UEP binary code of the appropriate configuration are known. However, the results were scarce, due to the limitations of the signal sets.

CHAPTER 6

Conclusions and Recommendations for Future Research

The unequal error protection capabilities of convolutional encoders were investigated. The effective free distance vector, **d**, was defined as an alternative to the free distance as a primary performance parameter for UEP convolutional encoders. It was shown that, although the free distance for a code is unique to the code and independent of the encoder realization, the effective distance vector is dependent on the encoder realization.

A modified transfer function, which provides a method to calculate **d**, was presented. The modified transfer function developed a new branch labeling method that allows standard algorithms that were originally developed to calculate the free distance of a code to calculate the effective distance vector.

Several upper bounds on **d** were derived. The bounds indicate that convolutional encoders with unbalanced memory distributions may provide more pronounced unequal error protection than encoders with balanced memory distributions. The bounds are evaluated and compared. The results of searches for good unequal error protection codes are presented. Both balanced and unbalanced memory allocation

configurations are examined. A primary goal of the searches was to find encoders with at least one effective distance greater than the free distance of the optimal code of the same rate and memory order. A decrease in free distance was acceptable. A number of binary convolutional encoders meeting the goal were listed. Bit error rate (BER) plots for the encoders were presented, and confirmed the effective distance as a measure of unequal error protection. At the same time, the BER plots show that the number of code sequences with Hamming weights equal to the individual effective distance is more important than expected.

Extensions to trellis coded modulation were examined. It was found that providing unequal error protection with TCM coding is difficult due to the limitations of the signal constellations. However, it was shown that the optimal binary UEP encoder provides information about the maximum possible free Euclidean distance for a TCM code with the same configuration.

Topics for future study are now identified. Because the effective distance is dependent on the encoder realization, and because $k > 1$ for UEP encoders, ways to reduce the search space would be extremely useful. For instance, tighter bounds on the effective distance vector would better identify encoder configurations with UEP potential. It is possible that the approach used to developed bound (4.37) may be applied to the k-way bound of (4.57), resulting in a tighter bound. Identifying features common to good UEP encoders would also help focus searches.

Extensions of the results to trellis coded modulation are limited for the 8PSK and 16QAM systems, due in part to the restrictive nature of the constellations themselves. It is possible that larger standard, as well as nonstandard, constellations will have enough flexibility to provide unequal error protection with TCM.

Similarly, multi-dimensional codes may have unequal error protection capabilities.

In particular, it is possible that altering the signal set partitioning developed in [48] will bring about unequal error protection to the information bits.

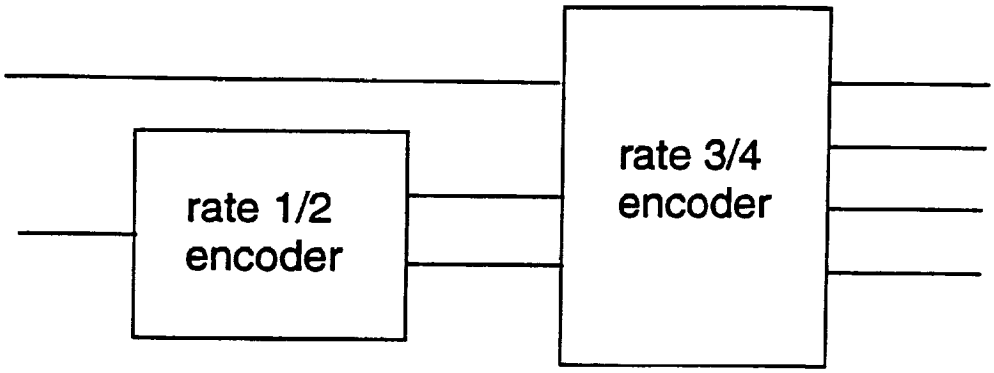


Figure 6.1: A Concatenated UEP System

An example of another configuration which holds promise as an UEP coding system is the multi-level concatenated system shown in Figure 6.1. Although specific rates are shown in the figure, the rates are unspecified for the general system. It is expected that the information bits which enter the first encoder will have larger effective distances than the information bits which pass through only one encoder. Furthermore, the disparity in error protection may be enhanced by choosing appropriate UEP binary encoders codes. A simplified decoding algorithm for a system similar to the one in Figure 6.1 would be important.

Providing unequal error protection with binary convolutional encoders and trellis coded modulation remains a viable area of research.

BIBLIOGRAPHY

[1] Khaled A. S. Abdel-Ghaffer, "On Unit Constraint-Length Convolutional Codes," *IEEE Trans. on Information Theory*, vol. 38, pp.200-206, January 1992.

[2] E. R. Berlekamp, "*Algebraic Coding Theory*," McGraw-Hill, 1968.

[3] I.M. Boyarinov and G.L. Katsman, "Linear Unequal Error Potection Codes," *IEEE Trans. on Information Theory*, IT-27:168–175, March 1981.

[4] L. Calabi, "On the Minimal Weight of Binary Group Codes," *IEEE Trans. on Information Theory*, vol. 10, pp. 385-387, October 1964.

[5] A. Robert Calderbank, J.E. Mazo, and Victor K. Wei, "Asymptotic Upper Bounds on the Minimum Distance of Trellis Codes," *IEEE Trans. on Communications*, vol. 33, pp. 305-309, April 1985.

[6] A. R. Calderbank and N.Seshadri, "Multilevel Codes for Unequal Error Protection," *Proceedings of the Int'l Symposium on Information Theory*, p. 183, January 1993.

[7] A. R. Calderbank and N.Seshadri, "Multilevel Codes for Unequal Error Protection," *IEEE Trans. on Information Theory*, vol. 39, pp. 1234-1248, July 1993.

[8] Mats Cedervall and Rolf Johannesson, "A FAST Algorithm for Computing Distance Spectrum of Convolutional Codes," *IEEE Trans. on Information Theory*, 35:1146–1159, November 1989.

[9] Daniel J. Costello, Jr., "Free Distance Bounds for Convolutional Codes," *IEEE Trans. on Information Theory*, IT-20:356–365, May 1974.

[10] David G. Daut, James W. Modestino, and Lee D. Wismer, "New Short Constraint Length Convolutional Code Constructions for Selected Rational Rates," *IEEE Trans. on Information Theory*, IT-28:794–800, September 1982.

[11] L.A. Dunning and W.E. Robbins, "Optimal Encodings of Linear Block Codes for Unequal Error Protection," *Inf. Control*, 37:150–177, 1978.

[12] P. Elias, "Coding for Noisy Channels," *IRE Conv. Record*, pp. 37-47, 1961.

[13] R. M. Fano, "A Heuristic Discussion of Probabiistic Decoding," *IEEE Trans. on Information Theory*, vol.IT-9, pp. 64-74, April 1963.

[14] G. D. Forney, Jr. "The Viterbi Decoding Algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268-278, March 1973.

[15] G. D. Forney, Jr. "Convolutional Codes II: Maximum Likelihood Decoding," *Inf. Control*, vol. 25, pp. 222-266, July 1974.

[16] E. N. Gilbert, "A Comparison of Signalling Alphabets," *Bell System Technical Journal*, 31:504–522, 1952.

[17] J. H. Griesmer, "A Bound for Error-Correcting Codes," *IBM J. Res. Develop.*, 4:532–542, 1960.

[18] Hermann J. Helgert and Russell D Stinaff, "Minimum-Distance Bounds for Binary Linear Codes," *IEEE Trans. on Information Theory*, vol. 19, pp. 344-356, May 1973.

[19] J. A. Heller, "Short Constraint Length Convolutional Codes," *JPL Summary 37-54*, vol. 3, pp. 171-177, 1968.

[20] M. E. Hellman, and J. Raviv, "Probability of Error, Equivocation, and the Chernoff Bound," *IEEE Trans. on Information Theory*, vol.IT-16, pp. 368-372, July 1970.

[21] I. M. Jacobs, "Probability of Error Bounds for Binary Transmission on the Slowly Fading Rician Channel," *IEEE Trans. on Information Theory*, vol.IT-12, pp. 431-441, Oct. 1966.

[22] F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack," *IBM J. Res. and Dev.*, vol. 13, pp. 675-685, 1969.

[23] J. Jensen, "Bounds on the Free Distance of Systematic Convolutional Codes," *IEEE Trans. on Information Theory*, IT-34:586–589, May 1988.

[24] J. Justesen, "Bounded Distance Decoding of Unit Memory Codes," *Report IT-121 at the Technical University of Denmark*, September 1990.

[25] J. Justesen, E. Paaske, and M. Ballan, "Quasi-Cyclic Unit Memory Convolutional Codes," *IEEE Trans. on Information Theory*, vol. IT-36, pp. 540-547, May 1990.

[26] C.C. Kilgus and W.C. Gore, "Cyclic Codes with Unequal Error Protection," *IEEE Trans. on Information Theory*, IT-17:214–215, March 1971.

[27] C.C. Kilgus and W.C. Gore, "A Class of Cyclic Unequal Error Protection Codes," *IEEE Trans. on Information Theory*, IT-18:687–690, September 1972.

[28] Y. Kofman, E. Zehavi, and S. Shamai (Shitz), "Analysis of a Multilevel Coded Modulation System," submitted to *IEEE Trans. on Communications*, 1991.

[29] Gregory S. Lauer, "Some Optimal Partial Unit Memory Codes," *IEEE Trans. on Information Theory*, vol. IT-25, pp. 240-243, March 1979.

[30] J. Layland and R. McEliece, "An Upper Bound on the Free Distance of a Tree Code," *JPL Technical Report*, JPL SPS 37-62, vol. 3, pp. 63-64, 1970.

[31] L. N. Lee, "Short Unit-Memory Byte-Oriented Binary Convolutional Codes Having Maximal Free Distance," *IEEE Trans. on Information Theory*, vol. IT-22, pp. 349-352, May 1976.

[32] Shu Lin and Daniel J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., 1983.

[33] D. Mandelbaum, "Unequal Error Protection Codes Derived from Difference Sets," *IEEE Trans. on Information Theory*, vo. IT-18, pp. 686-687, September 1972.

[34] B. Manisck and J.K. Wolf, "On Linear Unequal Error Protection Codes," *IEEE Trans. on Information Theory*, vo. IT-13, pp. 600-607, October 1967.

[35] Diane G. Mills and Daniel J. Costello, Jr., "An Upper Bound on the Free Distance of Double Memory Codes," *Proceedings of the 1992 Allerton Conference on Communications, Controls, and Computing*, p. 20, October 1992.

[36] Diane G. Mills and Daniel J. Costello, Jr., "Using a Modified Transfer Function to Calculate Unequal Error Protection Capabilities of Convolutional Codes," *Proceedings of the 1993 Int'l Symposium on Information Theory*, pp. 144, San Antonio, Texas, January 1993.

[37] D. G. Mills, and D. J. Costello, Jr., "A Bound on the Unequal Error Protection Capabilities of Rate k/n Convolutional Codes," *Proceedings of the 1994 Intl. Symp. on Inform. Theory.*, p. 274, Trondheim, Norway, June 1994.

[38] J. K. Omura, "On the Viterbi Decoding Algorithm," *IEEE Trans. on Information Theory*, vol.IT-15, pp. 177-179, Jan. 1967.

[39] J. K. Omura, and M. K. Simon, "Generalized Transfer Function Techniques," *Jet Propulsion Laboratories Technical Report,* 1981.

[40] R. Palazzo, Jr., "Linear Unequal Error Protection Convolutional Codes," *IEEE Intl. Symp. on Information Theory,* pp. 88-89, Brighton, England, 1985.

[41] R. Palazzo, Jr., and R. C. F. Cruz, "A New Search Algorithm for Good Unequal Error Potection Convolutional Codes," Proceedings of the *Intl. Symp. on Signals, Systems, and Electronics - URSI,* pp. 653-656, Germany, 1989.

[42] R. Palazzo, Jr., "An Upper Bound on the Average Distortion Measure of Cyclostationary Sequences via Dynamic Programming," *Proceedings of the Intl. Symp. on Operational Research, Porto Alegre, Brazil, 1986.*

[43] Reginaldo Palazzo, Jr., "On the Linear Unequal Error Protection Convolutional Codes," *GLOBECOM '86 Proceedings*, pp. 38.6.1-38.6.5, 1986.

[44] R. Palazzo, Jr., and K. V. O. Fonseca, "Periodically Time-Varying Trellis Coded Modulation," *Proceedings of the Intl. Symp. on Inform. and Coding Theory*, pp. 81-89, Campinas, Brazil, 1987.

[45] R. Palazzo, Jr., and K. V. O. Fonseca, "Unequal Error Protection of Superlinear Time-Varying Trellis Coded Modulation," *Proceedings of the 4-th Joint Sweden-USSR Intl Workshop on Inform. Theory*, pp. 332-337, Visby, Sweden, 1989.

[46] R. Palazzo, Jr., "A Time-Varying Convolutional Encoder Better than the Best Time-Invariant Encoder," *IEEE Trans. on Information Theory*, vol.IT-30, pp. 1109-1110, May 1993.

[47] R. Palazzo, Jr., "A Network Flow Approach to Convolutional Codes," submitted to *IEEE Trans. on Communnication*, 1993.

[48] S. S. Pietrobon, R.H. Deng, A. Lafanechere, G. Ungerboeck, D. J. Costello, Jr, "Trellis-Coded Multidimensional Phase Modulation," *IEEE Trans. on Information Theory*, vol. 36, pp.63-89, January, 1990.

[49] Ph. Piret, *Convolutional Codes*, MIT Press, 1988.

[50] Gregory J. Pottie and Desmond P. Taylor, "Multilevel Codes Based on Partitioning," *IEEE Trans. on Information Theory*, 35:87–98, January 1989.

[51] G. Robert Redinbo and Wai Yuen Cheung, "The Design and Implementation of Unequal Error-Correcting Coding Systems," *IEEE Trans. on Communications*, COM-30:1125–1135, May 1982.

[52] William Joseph Rosenberg, "Structural Properties of Convolutional Codes," *Ph.D. Dissertation, UCLA*, May 1971.

[53] Marc Rouanne and Daniel J. Costello, Jr., "A Lower Bound on the Minimum Euclidean Distance of Trellis-Coded Modulation Schemes," *IEEE Trans. on Information Theory*, 34:1011–1020, September 1988.

[54] M. Rouanne, and D. J. Costello, Jr., "An Algorithm for Computing the Distance Spectrum of Trellis Codes," *IEEE Journal on Selected Areas in Commun.*, vol.SAC-7, pp. 929-940, Aug. 1989.

[55] N. Seshadri and C-E. W. Sundberg, "Multi-Level Block Coded Modulation with Unequal Error Protection for the Rayleigh Fading Channel," *submission*, 1992.

[56] C.E. Shannon, *A Mathematical Theory of Communication*, University of Illinois Press, Urbana, Illinois, 1962.

[57] Akira Shiozaki, "Unequal Error Protection of PCM Signals by Self-orthogonal Convolutional Codes," *IEEE Trans. on Communications*, vol. COM-37, pp. 289-290, March 1989.

[58] R. Michael Tanner, "Convolutional Codes from Quasi-Cyclic Codes: A Link between the Theories of Block and Convolutional Codes," in *Computer Research Labaratory, Technical Report, UCSC-CRL-87-21*, November 1987.

[59] Christian Thommesen and Jorn Justesen, "Bounds on Distances and Error Exponents of Unit Memory Codes," *IEEE Trans. on Information Theory*, vol. IT-29, pp. 637-649, September 1983.

[60] Gottfried Ungerboeck, "Channel Coding with Miltilevel/Phase Signals," *IEEE Trans. on Information Theory*, vol. 28, pp.55-67, January, 1982.

[61] Gottfried Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part I: Introduction," *IEEE Communications Magazine*, 25:5–11, February 1987.

[62] Gottfried Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part II: State of the Art," *IEEE Communications Magazine*, 25:12–21, February 1987.

[63] Wil J. van Gils, "Linear Unequal Error Protection Codes from Shorter Codes," *IEEE Trans. on Information Theory*, IT-30:544–545, May 1984.

[64] Wil J. Van Gils, "Two Topics on Linear Unequal Error Protection Codes: Bounds on their Length and Cyclic Code Classes," *IEEE Trans. on Information Theory*, IT-29:866–876, November 1986.

[65] T. Verhoeff, "An Updated Table of Minimum-Distance Bounds for Binary Linear Codes," *IEEE Trans. on Information Theory*, IT-33:665–680, September 1987.

[66] A. J. Viterbi, and J. K. Omura, *Principles of Digital Communications and Coding,* MacGraw-Hill, 1979.

[67] A. J. Viterbi, "Error Bounds for Convlutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, vol.IT-13, pp. 260-269, April 1967.

[68] L.-F. Wei, "Coded Modulation with Unequal Error Protection," *IEEE Trans. Comm.*, vol. 41, pp. 1439-1449, October 1993.

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

[70] J. M. Wozencraft, and I. M. Jacobs, *Principles of Communication Engineering*, John Wiley & Sons, 1965.

[71] Kazuhiko Yamaguchi and Hideki Imai, "Multilevel Coded Modulation using Unequal Channel-Error Protection Code," *SITA 91*, December 1991.

[72] Kazuhiko Yamaguchi, Ryuji Kohno, and Hideki Imai, "Coded Modulation Based on Nonuniformity in Encoding and Mapping," *CSIM 91-14*, September 1991.

[73] K. Zigangirov, "Some Sequential Decoding Procedures," *Probl. Peredachi Inf.*, vol. 2, pp. 13-25, 1966.