Order Number 9315483

Bounds for convolutional Projection codes

Li, Dong, Ph.D.

City University of New York, 1993

U·M·I

300 N. Zeeb Rd.
Ann Arbor, MI 48106

BOUNDS FOR CONVOLUTIONAL PROJECTION CODES

BY

DONG LI

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree for the Doctor of Philosophy, The City University of New York.

1993

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

January 16, 1993
Date

Chair of Examining Committee

1/26/93
Date

Executive Officer

Joseph Barba

Tarek Saadawi

David Manela

Radimir Bozowic
Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

BOUNDS FOR CONVOLUTIONAL PROJECTION CODES

BY

DONG LI

Adviser: Professor D.L. Schilling

A new circuit implementation of convolutional Projection codes is introduced. The structure of the convolutional P1, P2 and P3 codes are studied, and their generating functions obtained. For a small constraint length convolutional code, one can use the state-diagram technique to obtain the generating function. However, a typical convolutional Projection code has a constraint length of 56. Hence, the state-diagram technique cannot be used realistically to obtain the generating function because of the extremely large number of computations involved. A new technique called the state equation technique has been developed to handle such large, constraint length, convolutional Projection codes. Several examples using the new technique are presented. The results of the examples are compared to the results obtained by the state-diagram technique and the simulation.

ACKNOWLEDGEMENT

I wish to express my deep gratitude to Professor D.L. Schilling for his valuable guidance and encouragement during the period of this research. The many technical discussions and his constructive suggestions are most appreciated.

I would like to thank all the members of my doctoral committee; Professor J. Barba, Professor T. Saadawi, Professor D. Manela, and Professor R. Bozovic, for the time and effort each has taken to read and constructively criticize this dissertation.

Special thanks to my parents and my wife for their support and encouragement.

Table of Contents

Lists of Tables

Lists of Figures

1 STATEMENT OF THE PROBLEM

1.1 INTRODUCTION

Modern communication systems often must be designed to transmit very high data rates-sometimes many millions of bits per second. To protect such systems from error, codes are often used.

Rectangular codes were introduced by Patel and Hong in 1974 for high density magnetic tapes [1] and Prusinkiewicz and Budkowski later introduced a double-track, error-correcting, rectangular code [2]. These codes have only two parity check slopes. In 1987, Schilling and Manela introduced a generalized version of the rectangular code called the Projection code [3]. This code can have more than two parity check slopes and a more sophisticated parity check algorithm. Characteristically it is highly structured, with random and burst error correcting capability.

In recent years, much research has been done with the Projection code. In particular, a lot of simulations

have been performed. The simulation results show that it is one of the best codes available [4]. Still, theoretical bounds are required to prove the error rate performance. To that end, upper and lower bounds of the bit error rate for the convolutional Projection codes have been calculated by employing the state equation technique developed by this author.

1.2 PROJECTION CODE

There are two versions of the Projection code. The first version is called the block Projection code. The block Projection code can be designed to have convenient block sizes with desired code rates. The second version is called the convolutional Projection code. The convolutional Projection code is a special class of convolutional code with respect to the design of the generating function and the rates with choice. In the next section, we will review basic concepts in convolutional code.

The two versions of the Projection code make the code more attractive than other codes for a wide variety of applications. In the convolutional Projection code,

the inherent geometric properties dictate the minimum path distance, or the free distance, and the generating function. These attributes provide a code designer with a convenient tool for designing a convolutional code with a known free distance and other properties of the generating function.

The block Projection code is constructed by taking blocks of data, then adding parity check blocks to the data and drawing parity check lines with different slopes from the data block to the parity check block and requiring the summation of every bit on a line to be 0 mod-2 for binary code, and the summation of every symbol on a line to be 0 mod-M for non-binary M-ary code. $M = 2^n$ (n is the number of bits in a symbol). The above procedure constitutes the encoding process of a block Projection code. This is illustrated in a binary block Projection encoder shown in Fig.1.2.1.

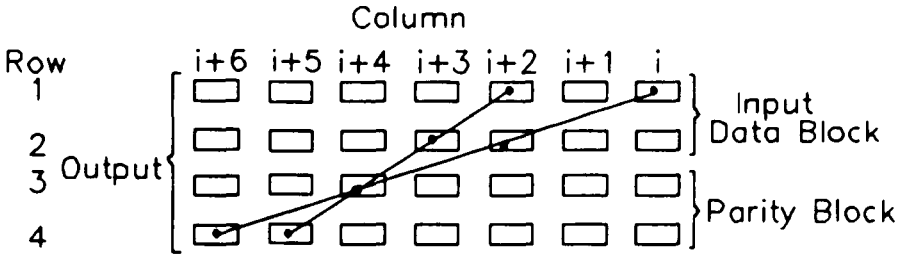

Figure 1.2.1 A binary block Projection encoder.

In this binary block Projection encoder, each box represents a shift register. There are two parity check lines shown. The encoding rule for this code in the two parity check lines can be summarized in the following two equations:

$$d_{1,i} + d_{2,i+2} + c_{3,i+4} + c_{4,i+6} = 0 \mod 2 \qquad (1.2.1a)$$

$$d_{1,i+2} + d_{2,i+3} + c_{3,i+4} + c_{4,i+5} = 0 \mod 2 \qquad (1.2.1b)$$

where d denotes a data bit and c denotes a parity check bit.

Every block Projection code can also be realized by a convolutional code. We can use the encoding process of a block Projection code as a systematic way to

construct a convolutional code and the code obtained is called a convolutional Projection code. This is illustrated in Chapter two.

The minimum Hamming distance for a block Projection code or the free distance for a convolutional Projection code, is designed with choice, and the terms of the generating function for a convolutional Projection code, except for the coefficients, are designed with choice. This gives a code designer a great amount of flexibility. In a low error rate input system, if the output error rate requirement is not that high, one can use a simple Projection code. In a high error rate input system, where the output error rate requirement is high, one can choose a more complex Projection code to meet the system requirements.

A very simple procedure which is easy to implement for the convolutional Projection code will be illustrated in Chapter two. The implementation of the encoder of the block Projection code and the error rate analysis can be found in references [3] and [5].

The Projection code can be further classified into three types. The first is called the P1 Projection

code or basic SM code. The free distance, d_{free}, for a convolutional Projection code and the minimum Hamming weight, d_{min}, for a block Projection code are both r+1, where r is the number of parity check lines in the codes. The second type is called P2 Projection code or PASM (partial autoconcatenation SM). It's free distance for a convolutional Projection code and the minimum Hamming weight for a block Projection code are both less than or equal to 2^r. By choosing the right set of slopes, the maximum d_{free} and d_{min} are achieved. The last type is called P3 Projection code or TASM (total autoconcatenation SM). The d_{free} and the d_{min} are both 2^r for any choice of a set of r distinct slopes.

A detailed analysis of the encoders of a block Projection code and a convolutional Projection code can be found in Chapter two.

1.3 CONVOLUTIONAL CODE

1.3.1 CONCEPTS OF CONVOLUTIONAL CODE

A convolutional code can be defined by a shift-register encoder configuration shown in Fig.1.3.1.1.

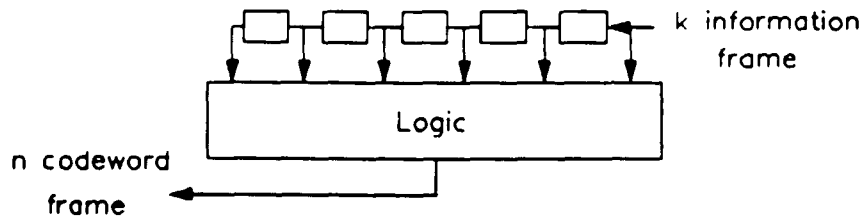

Figure 1.3.1.1 A rate k/n shift-register convolutional encoder.

The encoder consists of shift registers and exclusive-or logical summers which are not shown in the figure. The internal connections of the encoder are not shown. Most of the basic concepts are illustrated by this encoder. The input sequence is shifted in, beginning at time zero, and continues indefinitely. The input sequence is broken into segments, or frames, each of k symbols. A segment may be as short as one symbol. The encoder can store S segments. During each unit of time, a new segment of the input sequence is shifted into the encoder, and the earliest segment stored in the S-stage shift register is shifted out. The encoder computes a single codeword segment of length n symbols, and shifts it out as the next input sequence segment is

shifted in.

The rate, R, of the code is given by:

$$R = \frac{k}{n}$$

The constraint length, L, of the convolutional code is defined as L=kS. The convolutional encoder shown in Fig.1.3.1.1 has L=5 for each input sequence segmented into one symbol, and S=5.

There are several ways to represent a simple convolutional code graphically. One is the so-called code tree [6] that describes the encoding rule and the set of conditions in the code space. For a Rate-1/n code, if the first input data bit is 0, the corresponding n output bits are those on the first upper branch, while if the bit is 1 the output bits are on the first lower branch. In the same way, if the second bit is 0, an upper branch is followed and the next n output bits are read. For k input bits, there are corresponding k levels of the code tree and 2^k possible output sequences.

For the code tree considered above, after the first n+1 branches, the tree structure becomes repetitive, i.e. the sequence of output bits from a pair of nodes of the tree are identical, and thus those nodes can be joined together. This is referred to as a trellis diagram [7]. The completely repetitive structure of the trellis diagram suggests a further reduction in representation of the code to the so-called state diagram [8]. By considering the encoder as a finite-state machine, the state diagram can be drawn directly.

Optimum Decoding on BSC

On a Binary Symmetric Channel (BSC) errors that transform a code bit 0 to 1 or 1 to 0 are assumed to occur independently from bit to bit with probability p. If all input sequences are equally likely, the optimum decoder that minimizes the overall error probability for convolutional code is one that examines the error-corrupted received sequence and

chooses the data sequence corresponding to the transmitted sequence, which is closest to the received sequence in the sense of Hamming distance.

Generating Function

A major characteristic of a convolutional code is that it is a group code [9]. Thus the weight, or distance, from any specific codeword, or a path, to any other codeword is equal to the weight of the resultant codeword. In connection to the distances among paths, one with the minimum distances compared to the all-zero path is the free distance. The number of paths and the distance for each path are determined by means of the generating function [10], or transfer function, from the code's state diagram by labeling the branches of the state diagram D^l, D^{l-1}, $\ldots D^2$, D, $D^0 = 1$, where the exponent corresponds to the distance of the particular branch from the corresponding branch of the all-zero path. The generating function, T(D), can be obtained from the collection of all the paths due to the distinct

input sequences, which can be expressed as

$$T(D) = \sum_{j=1}^{\infty} c_j D^{n(j)} \qquad (1.3.1.1)$$

where c_j denotes the number of different paths with path distance n(j).

1.3.2 PERFORMANCE BOUND FOR OPTIMUM DECODING

For a convolutional code, which is a linear code, the performance of the optimum decoding algorithm, or the Viterbi algorithm, on the BSC is analyzed by assuming that the all-zero sequence is being transmitted. The received channel sequence is corrupted with noise. The decoder attempts to identify the correct path, or sequence, through the trellis and the all-zero path is eliminated for the first time when the first error is made. The decoder chooses an incorrect path. An incorrect path is one that diverged from the all-zero path at some step prior to the jth step, and is merging with the all-zero path again for the first time at the jth step. If the path distance is n(j), an error is being made if

the BSC caused (n(j) + 1)/2 or more errors in the n(j) number of non-zero bits. Hence, the probability of an error in the path at jth step, compared to all-zero path, is given by:

If n(j) is odd.

$$P_{n(j)} = \sum_{e=\frac{n(j)+1}{2}}^{n(j)} \binom{n(j)}{e} p^e (1-p)^{n(j)-e} \qquad (1.3.2.1)$$

If n(j) is even.

$$P_{n(j)} = \frac{1}{2}\binom{n(j)}{\frac{n(j)}{2}} (p(1-p))^{\frac{n(j)}{2}} + \sum_{e=\frac{n(j)}{2}+1}^{n(j)} \binom{n(j)}{e} p^e (1-p)^{n(j)-e} \qquad (1.3.2.2)$$

where p is the transition probability on the BSC. $P_{n(j)}$ is called the first-event error probability [8] for the path at the jth step with distance n(j).

Considering all the paths of the generating function given in Eg.(1.3.1.1), an union bound of the first-event error probability, P_E, can be expressed as

$$P_E < \sum_{j=1}^{\infty} c_j P_{n(j)} \qquad (1.3.2.3)$$

where c_j is the number of distinct paths with distance n(j). Note that some of the coefficients are zero, which are dictated by the structure of the code. C_1 is the number of distinct paths with distance d_{free}. The above union bound can be further upper bounded [11] when $P_{n(j)}$ is

$$P_{n(j)} < (4p(1-p))^{\frac{n(j)}{2}} \qquad (1.3.2.4)$$

Then,

$$P_E < \sum_{j=1}^{\infty} c_j (4p(1-p))^{\frac{n(j)}{2}} \qquad (1.3.2.5)$$

A loose upper bound of the bit error rate can be obtained by weighting the expression Eq.(1.3.2.5) by n(j)/n, the largest number of bits in error caused by BSC for each step j in a path. The over-estimation of the number of bits in error in a path corresponding to the jth step justifies the bit error being upper bounded. Thus, the loose upper bound of the bit error rate, $P_{b(loose\text{-}upper)}$, for the BSC is

$$P_{b(loose\text{-}upper)} = \sum_{j=1}^{\infty} c_j \frac{n(j)}{n} (4p(1-p))^{\frac{a(j)}{2}} \qquad (1.3.2.6)$$

If instead of transmitting a single bit into the encoder at each time, k bits are transmitted into the encoder, the above expression must be divided by k.

1.4 COMPARISON OF THE STATE EQUATION TECHNIQUE WITH PREVIOUS TECHNIQUES

For a small constraint length convolutional code, one can use the Viterbi state-diagram technique to obtain the generating function. However, for a typical

convolutional Projection code the constraint length is typically of the order of 56. The state-diagram technique cannot realistically be used to obtain the generating function due to the extreme large constraint length. Since 1971, using simulation technique to find optimum convolution codes is limited by constraint length no larger than 20. The state equation technique is developed to compute the generating function of large constraint length convolutional Projection codes. The first 4 coefficients of the generating function of a L=56 convolutional Projection code is calculated in Chapter five by employing the state equation technique.

The error rate of a powerful convolutional code or convolutional Projection code is dominated by the free distance; that is, it is sufficient to know the first few coefficients of the generating function to bound the error rate of the code.

2 BLOCK AND CONVOLUTIONAL PROJECTION ENCODERS

2.1 BINARY P1 ENCODER

An encoder for the block and the convolutional Projection code can be implemented as shown in the following examples:

Block P1 encoder for 2 data lines and 2 parity check lines with slopes of 1 and 1/2:

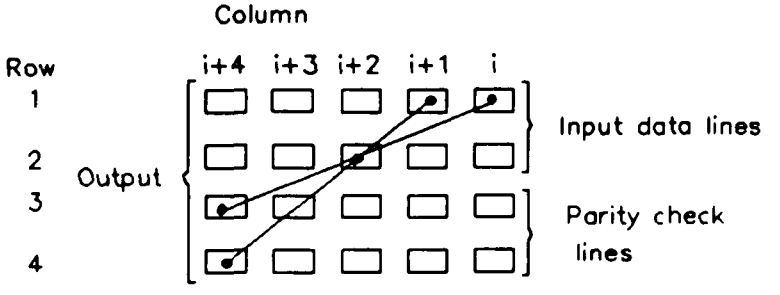

Figure 2.1.1 A Rate-1/2 block P1 encoder with slopes of 1 and 1/2.

In this P1 block Projection code, the top two rows comprise the data block and the bottom two rows comprise the parity check block. There are two parity check

lines shown. In each line, those bits represented by dots are summed to zero mod-2. Note that a parity bit along a check line checks only data bits; it does not check other parity bits. It is clear that the parity check bits in the two parity check lines are directly obtained from the input data by following the rule:

$$d_{1,1} + d_{2,1+2} + c_{3,1+4} = 0 \mod 2 \qquad (2.1.1a)$$

$$d_{1,1+1} + d_{2,1+2} + c_{4,1+4} = 0 \mod 2 \qquad (2.1.1b)$$

The length of the rectangular block shown in Fig.2.1.1 can be varied, and the number of slopes is equal to the number of parity check rows. However, the slopes must be chosen to be distinct. Further, the leftmost column and rightmost column can be joined together so as to form a cylinder for the purpose of encoding. After encoding, it can be re-opened to form a rectangular block again. In addition to using straight check lines, check lines can be chosen to be curves. However, the author has shown that choosing curved parity check lines does not represent an improvement of the code. Consider the fact that every bit in the block is being checked

the same number of times. Instead of a bit being checked in the immediate neighborhood of bits, as in straight line checking, curves checking are made uses a different partitioning of bits in the block. This does not alter the Hamming distance of the code which denotes the code's performance in the presence of random errors. It does, however, represent an interleaving and hence does affect the response of the code to burst errors.

The minimum Hamming weight for this code is 3. For r parity check lines, the P1 block Projection code's minimum Hamming weight is r+1. The proof is given in Theorem 1.

Convolutional P1 encoder for 2 data lines and 2 parity check lines with slopes of 1 and 1/2:

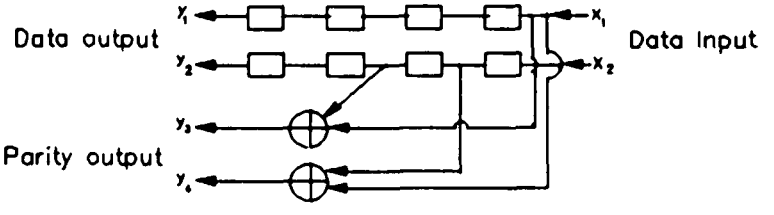

Figure 2.1.2 A convolutional Rate-1/2 P1 encoder with slopes of 1 and 1/2.

In this P1 convolutional Projection encoder the small rectangles represent shift registers. The outputs from the mod-2 adders are the parity check bits. Data are shifted into the encoder two bits at a time at input ports x_1 and x_2. The parity check lines can be seen to be a direct copy of the parity lines of the P1 block code for the same set of slopes.

The convolutional Projection code can be constructed with the same degree of freedom as the block Projection code. Because of this feature, an engineer has an advantage in designing a convolutional code with pre-determined free distance and other required properties.

The free distance for this code is 3. The proof is given in Theorem 2. In general, for a P1 convolutional Projection code, the free distance is

$$d_{free} = r + 1 \qquad (2.1.2)$$

The generating function for a P1 convolutional Projection code of Rate-1/1+r with r distinct slopes is

$$T(D) = D^{r+1} + \sum_{j=2}^{\infty} c_j D^{(r+1)+(j-1)(r+1)} \qquad (2.1.3)$$

where c_j is the coefficient of the generating function. Note that $c_1 = 1$. The other coefficients, c_j, can be obtained by employing the state equation technique. The distance between any adjacent terms in the generating function is r+1. It is proven in Theorem 2.

The generating function for a Rate-k/k+r ($k \neq 1$) binary P1 code can be obtained by employing the state equation technique.

Theorem 1

For a binary P1 block Projection code, the minimum Hamming weight is r+1 for r distinct parity check lines. The Hamming weights for a Rate-1/1+r code are

(r+1)+i(r+1), where i takes values of 0, 1, 2, ...

Proof:

In a Rate-1/1+r code, there is only one data line and r parity check lines. According to the encoding rule, an input data bit of 1 being shifted into the encoder will induce r bits of 1 on parity check shift registers for r distinct slopes. If some of the slopes are the same, then there are less than r parity bits of 1 on the parity check shift registers. This is because of the fact that any number of 1s in one bit position counts as one 1 in that position. Next, another 1 is shifted into the encoder. No parity bits will overlap with the parity bits due to the previous input. The Hamming weight is (r+1)+(r+1). By considering all other distinct input sequences to the encoder, it is clear that the minimum Hamming weight is r+1 and the Hamming weights for all other codewords are (r+1)+i(r+1), where i takes values of 1, 2, 3,... .

Theorem 2

For a Rate-1/1+r binary convolutional P1 code, the free distance is r+1. The distance between two adjacent terms in the generating function is r+1. The generating function is

$$T(D) = D^{r+1} + \sum_{j=2}^{\infty} c_j D^{(r+1)+(j-1)(r+1)}$$

Proof:

Binary convolutional P1 code is directly constructed from the block P1 code by one-to-one identity mapping. The blocklength of the block code is infinity or at some finite value where every bit in the block is checked exactly the same number of times. A coded block of bits in the constructed convolutional Projection code are exactly the same as the corresponding block of bits in the block code. Thus the path distances in the convolutional Projection code are the same as the Hamming weights for the corresponding block Projection code. Therefore, the free distance for P1 code

is r+1 and the distance between two adjacent terms in the generating function is r+1. Furthermore, the number of distinct free distance paths is one for a Rate-1/1+r convolutional P1 code because of the fact that there is only one input data line in the encoder and a pattern caused by input data shifted in time counts only once, i.e. it does not count two distinct patterns. Thus, the generating function for a Rate-1/1+r convolutional Projection code is

$$T(D) = D^{r+1} + \sum_{j=2}^{\infty} c_j D^{(r+1)+(j-1)(r+1)}$$

The encoder in Fig.2.1.2 can be simplified to an equivalent simplified form with only one shift register as shown in Fig.2.1.3. Consider the fact that all the shift registers in the first output line y_1 and the shift registers next to output line y_2 in the original encoder shown in Fig. 2.1.2 are redundant in obtaining the generating function of the code. These shift registers only serve the purpose of delaying the data outputs y_1 and y_2. They do not alter the total number

of bits of 1 in the output. Therefore, the generating function of the simplified encoder is the same as the original encoder. The simplified encoder requires less computation to obtain the generating function than for the original encoder.

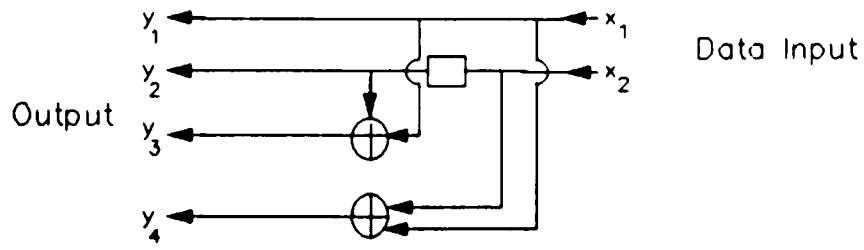

Figure 2.1.3 The simplified version of the encoder shown in Fig. 2.1.2

2.2 BINARY P2 ENCODER

An encoder for the block and convolutional P2 code can be implemented as shown in Fig.2.2.1 and Fig.2.2.2 respectively.

Block P2 encoder for 2 data lines and 2 parity check lines with slopes of 1 and 1/2:

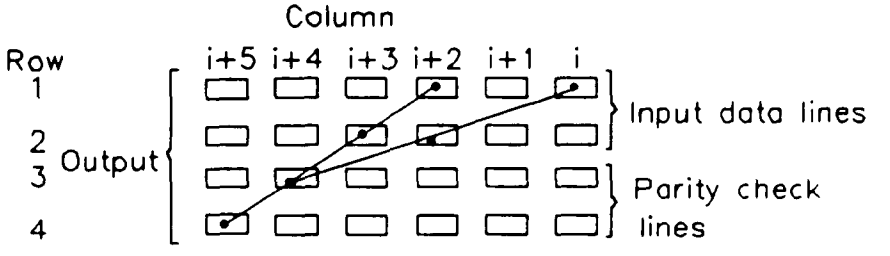

Figure 2.2.1 A Rate-1/2 block P2 encoder with slopes of 1 and 1/2.

In this P2 code, the encoder design is almost the same as in the P1 code except that the parity bits also check the parity bits above them. Thus,

$$d_{1,i} + d_{2,i+2} + c_{3,i+4} = 0 \mod 2 \qquad (2.2.1a)$$

$$d_{1,i+2} + d_{2,i+3} + c_{3,i+4} + c_{4,i+5} = 0 \mod 2 \qquad (2.2.1b)$$

Note that the extra feature of the parity check bits along a line checking the data bits and parity bits above them improves the minimum Hamming weight of the code.

The minimum Hamming weight for this code is 4. In general, a P2 block Projection code has $d_{min} \leq 2^r$, where r is the number of distinct slopes. The maximum d_{min} ($d_{min.max}$) is

$$d_{min.max} = 2^r \qquad (2.2.2)$$

The maximum d_{min} is achieved by choosing a set of prime numbers as the slopes [12].

Convolutional P2 encoder for 2 data lines and 2 parity check lines with slopes of 1 and 1/2:

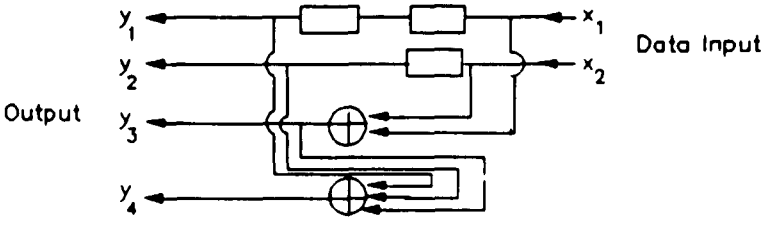

Figure 2.2.2 A Rate-1/2 convolutional P2 encoder with slopes of 1 and 1/2.

The encoder design for convolutional P2 code differs from the convolutional P1 code. In the convolutional

P2 encoder there are feedback paths, but in the convolutional P1 encoder there is no feedback. This alters the output order of parity check lines in the P2 encoder. The parity check line bits output must follow the order of the parity line with smallest slope first, the second smallest slope next and the largest slope last. In Fig.2.2.2, y_3 must output before y_4.

The free distance for this code is 4. In general, $d_{free} \leq 2^r$. The maximum d_{free}, denoted by $d_{free,max}$, is

$$d_{free,max} = 2^r \tag{2.2.3}$$

The d_{free} of the code can be maximized by choosing the same set of slopes of the corresponding optimal block P2 code.

In general, the generating function for an optimal convolutional P2 code can be expressed as

Rate-1/1+r code:

$$T(D) = D^{2^r} + \sum_{j=2}^{\infty} c_j D^{2^r+(j-1)2^{r-1}} \tag{2.2.4}$$

Rate-k/k+r code:

$$T(D) = \sum_{j=1}^{\infty} c_j D^{2^r \cdot (j-1)2} \qquad (2.2.5)$$

Equations (2.2.4) and (2.2.5) are proven in Theorem 3.

Theorem 3

In an optimal convolutional P2 code, the generating function for a Rate-1/1+r code is

$$T(D) = D^{2^r} + \sum_{j=2}^{\infty} c_j D^{2^r \cdot (j-1)2^{r-1}}$$

and the generating function for a Rate-k/k+r code is

$$T(D) = \sum_{j=1}^{\infty} c_j D^{2^r \cdot (j-1)2}$$

Proof:

For a Rate-1/1+r block P2 code, a 1 in the input data line induces $2^r - 1$ 1s in the parity check block. The pattern in the code consists of 2^r bits that are 1 by assuming initially the rest of bits in the code block

are 0. Next, a 1 in the data line is shifted into the encoder, which induces another pattern that overlaps with the previous pattern in one quarter of the 2^r bits, or positions. The overlapped bits in each pattern then are appended to 0. The total number of 1s in the two patterns is then $2\left(2^r - \frac{1}{4} * 2^r\right)$. Clearly, the Hamming weight due to two consecutive 1s in the input is $2^r + 2^{r-1}$. Considering all possible distinct input sequences, it is clear that the Hamming weights for the code are $2^r + (i-1)2^{r-1}$, where i=1,2,3... .

For a corresponding convolutional P2 code that is constructed by one-to-one identity mapping from the block P2 code, a codeword with minimum Hamming weight in the block code, corresponds to a path with the free distance in the convolutional code. The collection of all the other Hamming weights in the block code corresponds to all the other distances in the convolutional code. Clearly, the distances of distinct paths in the code are $2^r + (i-1)(2^{r-1})$, where i=1,2,3,... . Note that the number of distinct free distance paths is one because of the fact that there is only one input data line and a pattern shifted in time counts only once.

For a Rate-k/k+r ($k \neq 1$) code, each 1 on a data line induces $2^r - 1$ 1s on the parity block. Such a pattern satisfies the P2 encoding procedure. Any combination of such patterns with points that satisfy the P2 encoding condition would have an even number of 1s, because, for each overlapped position, the bit in that position is appended to 0. Since the generating function of the corresponding convolutional code is the collection of all the distinct patterns satisfying the encoding conditions in the block code, it is the collection of the possible even distances starting at 2^r,

$$T(D) = \sum_{j=1}^{\infty} c_j D^{2^r \cdot (j-1)2}$$

2.3 BINARY P3 ENCODER

An encoder for the P3 block and convolutional Projection code can be implemented as shown in Fig.2.3.1 and Fig.2.3.2.

Block P3 encoder for 2 data lines and 2 parity check lines with slopes of 1 and 1/2:

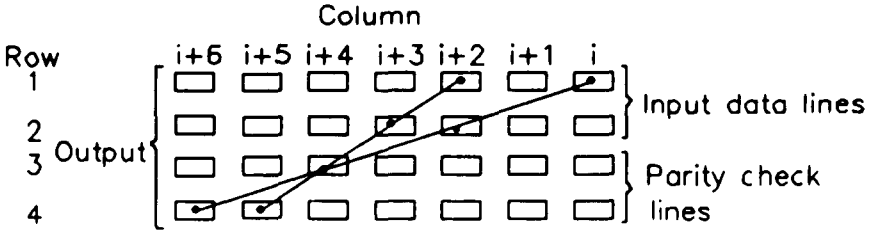

Figure 2.3.1 A Rate-1/2 block P3 encoder with slopes of 1 and 1/2.

The encoder design for this P3 code is almost the same as the P2 code, except that every parity bit checks the data bits, and the parity bits along the line. Thus

$$d_{1,i} + d_{2,i+2} + c_{3,i+4} + c_{4,i+6} = 0 \mod 2 \qquad (2.3.1a)$$

$$d_{1,i+2} + d_{2,i+3} + c_{3,i+4} + c_{4,i+5} = 0 \mod 2 \qquad (2.3.1b)$$

This feature guarantees that for any set of r distinct slopes, the minimum Hamming weight is 2^r; the proof is given in Theorem 3. The minimum Hamming weight for this code is 4.

Convolutional P3 encoder for 2 data lines and 2 parity check lines with slopes of 1 and 1/2:

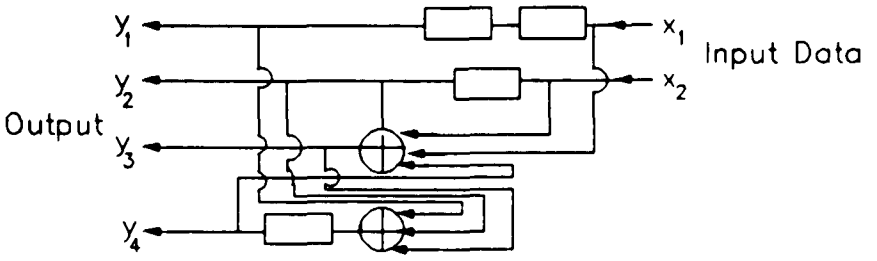

Figure 2.3.2 A Rate-1/2 convolutional P3 encoder with slopes of 1 and 1/2.

The encoder design for a Rate-1/2 P3 convolutional Projection code with slopes of 1 and 1/2 is the same as for the P2 convolutional Projection code with the added feature of the corresponding Rate-1/2 block P3 code with slopes of 1 and 1/2 added to it. For the P3 code, the free distance is 2^r for any set of r distinct slopes. Like the P2 code, the P3 code also has feedback paths. This alters the output order of parity check lines in the P3 code. In the encoder shown in Fig.2.3.2, y_3 must output before y_4. The generating function of

the code is

$$T(D) = \sum_{j=1}^{\infty} c_j D^{8\cdot(j-1)2}$$

The distance between any two adjacent terms of the generating function is 2.

In general, the generating function for a convolutional P3 code can be expressed as

Rate-1/1+r code:

$$T(D) = D^{2^r} + \sum_{j=2}^{\infty} c_j D^{2^r\cdot(j-1)2^{r-1}} \qquad (2.3.2)$$

Rate-k/k+r code:

$$T(D) = \sum_{j=1}^{\infty} c_j D^{2^r\cdot(j-1)2} \qquad (2.3.3)$$

Equations (2.3.2) and (2.3.3) are proven in Theorem 4.

Theorem 4

In a block P3 code with r distinct parity check lines, the minimum Hamming weight is 2^r . For the corresponding convolutional P3 code that is constructed by one-to-one identity mapping of the block code, the free distance is 2^r, and the generating function for a

Rate-1/1+r code is

$$T(D) = D^{2^r} + \sum_{j=2}^{\infty} c_j D^{2^r+(j-1)2^{r-1}}$$

and the generating function for a Rate-k/k+r code is

$$T(D) = \sum_{j=1}^{\infty} c_j D^{2^r+(j-1)2}$$

Proof:

Consider the fact that any pattern having M points, or M bits, that are 1, satisfies the parity check conditions of the block P3 code with r distinct parity check lines in the coded block is a codeword of Hamming weight of M, assuming that the rest of the bits in the coded block are 0. A codeword with minimum Hamming weight is a pattern in the code space where the number of points is minimum. In the pattern, each point is checked, or intersected, by the same set of r distinct lines. A candidate for such a pattern is a r-dimensional solid parallelogram projected onto the code plane. Each vertex is intersected by the same set of r distinct lines. There are 2^r vertices. Further, there is no structure with less than 2^r vertices in which each vertex is intersected by r lines in r-dimensional space. Therefore, the codeword, represented by a projected

r-dimensional solid parallelogram, is a codeword with the minimum Hamming weight. Thus the minimum Hamming weight of the code is 2^r.

For a Rate-1/1+r block P3 code, a 1 in the input data line induces $2^r - 1$ 1s in the parity check block. This pattern in the code block consists of 2^r bits that are 1 by assuming initially the rest of bits in the code block are 0. Next, a 1 in the input data line is shifted into the encoder, which forms another pattern that overlaps with the previous pattern in one quarter of the 2^r bits, or positions. The overlapped bits in each pattern then are appended to 0. The total number of 1s in the two patterns is then $2\left(2^r - \frac{1}{4} * 2^r\right)$. Clearly, the Hamming weight due to two consecutive 1s in the input is $2^r + 2^{r-1}$. Considering all possible distinct input sequences, it is clear that the Hamming weights for the code are $2^r + (i-1)2^{r-1}$, where i=1,2,3... .

For a corresponding convolutional P3 code that is constructed by one-to-one identity mapping from the block P3 code, a codeword with minimum Hamming weight in the block code, corresponds to a path with the free distance in the convolutional code. The collection of

all the other Hamming weights in the block code corresponds to all the other distances in the convolutional code. Clearly, the distances of distinct paths in the code are $2^r+(i-1)(2^{r-1})$, where i=1,2,3,... . Note that the number of distinct free distance paths is one because of the fact that there is only one input data line and a pattern shifted in time counts only once.

For a Rate-k/k+r code, each 1 on a data line induces 2^r-1 1s on the parity block. Such a pattern satisfies the P3 encoding procedure. Any combination of such patterns with points that satisfy the P3 encoding condition would have an even number of 1s because, for each overlapped position, the bit in that position is appended to 0. Since the generating function of the corresponding convolutional code is the collection of all the distinct patterns satisfying the encoding conditions in the block code, it is the collection of the possible even distances starting at 2^r.

3 THE STATE EQUATION TECHNIQUE IN OBTAINING T(D)

3.1 INTRODUCTION TO THE STATE EQUATION TECHNIQUE

The state equation technique utilizes the state equations of a convolutional encoder instead of drawing a state diagram of the encoder and uses a state flow diagram to obtain the generating function for the code by the analytical method developed by Viterbi. It computes terms of the generating function.

In chapter two, we defined the convolutional Projection encoder. The generating function of P1, P2 and P3 convolutional Projection codes have been put forward with coefficient c_j to be determined. In this chapter we will explore the state equation technique in obtaining T(D).

3.2 THE STATE EQUATION TECHNIQUE

For any binary convolutional Projection code, the encoder can be generally drawn as shown in Fig. 3.2.1.

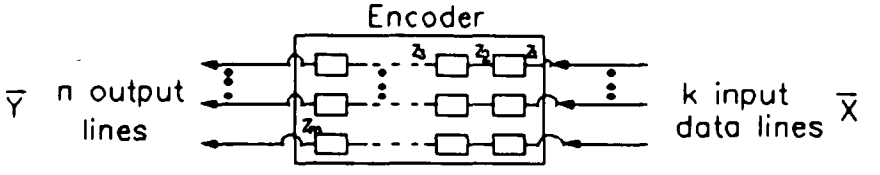

Figure 3.2.1 A general binary convolutional Projection encoder.

In Fig. 3.2.1, there are k inputs and n outputs, where n=k+r. The rate of the code is k/n. The basic building blocks of the encoder are shift registers and mod-2 adders which are not shown in the figure. The internal connections of the encoder and the input and output connections are also not shown. If one labels each of the connection points in the encoder as a state variable, $Z_1, Z_2, Z_3, \ldots Z_m$, some of them can be written as a function of the inputs to the encoder while others cannot. In addition, some of the state variables are recursive. Therefore one can implicitly list all the state equations resulting:

$$Z_i = f_i \quad , \qquad i = 1, 2, \ldots m$$

where f_i is the function relating the state variable Z_i

to either a linear combination of some of the input data or a linear combination of some of the input data and some of the state variables. The inputs can be viewed as a K-tuple vector $\vec{X}$ and the outputs as a N-tuple vector $\vec{Y}$, where

$$\vec{X} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_k \end{pmatrix} \quad , \quad \vec{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{pmatrix}$$

Denoting $\vec{Z}$ as a M-tuple state variable vector, where

$$\vec{Z} = \begin{pmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_m \end{pmatrix}$$

then $\vec{y}$ can be expressed as

$$\vec{Y} = A\vec{Z} \tag{3.2.1}$$

where A is a coefficient matrix with binary elements, and the multiplication and addition operations of the state equations are defined in a binary field for binary convolutional Projection code. For the non-binary convolutional Projection code, the operations are defined in a Galois extension field of characteristic 2. Now we define the path distance function, $|\vec{Y}(K)|$, as

$$|\vec{Y}(K)| = \sum_{i=0}^{k-1} \sum_{j=1}^{n} y_j(i) \tag{3.2.2}$$

which satisfies the following conditions:

$$\vec{Z}(j) = 0 \quad , \quad for\ j = K \tag{3.2.3.a}$$

$$\vec{Z}(j) \neq 0 \quad , \quad for\ j = 0, 1, 2, 3, \ldots K-1 \tag{3.2.3.b}$$

where $\vec{X}$ and $\vec{Z}$ are zero for negative discrete time.

K is a step index which increments by 1 as the computation of T(D) progresses. Equation (3.2.3.a) and Equation (3.2.3.b) are the conditions which must be satisfied to have valid solutions in matrix Equation (3.2.1) which can be used to compute the distances of paths guaranteed by the matrix equation. As K takes values of 1, 2, 3, ... ∞ , the collection of all the valid solutions resulting from the computation is the generating function. Then the generating function of the convolutional Projection code can be expressed as

$$T(D) = b_\tau D^{|\vec{Y}(\tau)|} + b_{\tau+1} D^{|\vec{Y}(\tau+1)|} + \ldots$$

$$= \sum_{i=\tau}^{\infty} b_i D^{|\vec{Y}(i)|} \qquad (3.2.4)$$

where τ is the first value of K for which a valid solution of the state equations is obtained. $b_i D^{|\vec{Y}(i)|}$ indicates that there are b_i number of different paths with distance of $|\vec{Y}(i)|$. Some of the coefficients are zero which is dictated by the structure of the code. The coefficient of d_{free} which is the first non-zero term of T(D), can be determined as the number of steps in

the computation equals 2L+1 by assuming that input sequence starts at zero in discrete time. L is the constraint length of the code. Consider the fact that the (2L+2)th bit of the input sequence does not influence the first bit of the input sequence in the code plane for a general convolutional Projection code with feedback. Thus, the first 2L+1 bits in all the input sequences determine all the different paths having distance d_{free}. The group of 2L+1 bits, starting at the (2L+2)th bit position of the input sequences will give the same set of paths because of the fact that if a path is shifted in the distance plane, it is not counted twice for a convolutional code.

The technique described above for computing T(D) is based on direct computation. For each input non-zero vector sequence, the output vector sequence is non-zero. For a non-catastrophic convolutional Projection code this corresponds to a non-zero path which originates from the all-zero state, i.e. all-zero path and then deviates from it and eventually returns to it.

The generating function given in Eq. (3.2.4) can be rearranged and renamed so the index number, i, starts

at one. The resulting T(D) given in Eq. (3.2.5) below is identical to the generating function given in Eq.(1.3.1.1).

$$T(D) = \sum_{j=1}^{\infty} c_j D^{n(j)} \qquad (3.2.5)$$

where c_j denotes the number of distinct paths with distance $n(j)$.

3.3 AN EXAMPLE IN OBTAINING T(D) BY THE STATE EQUATION TECHNIQUE

Example

Given a Rate-1/2 convolutional code shown in Fig.3.3.1,

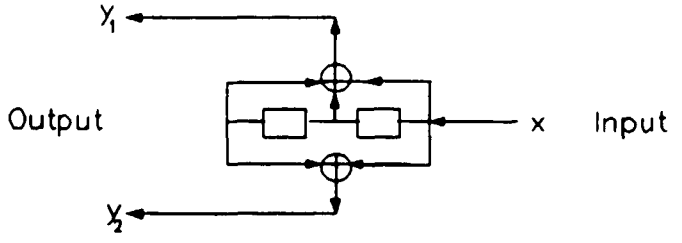

Figure 3.3.1 A Rate-1/2 convolutional code.

the generating function of the code can be obtained analytically by the state-diagram technique [13] as

$$T(D) = \frac{D^5}{1 - 2D}$$

Expanding T(D), yields:

$$T(D) = D^5 + 2D^6 + 4D^7 + \ldots + 2^i D^{i+5} + \ldots$$

To use the state equation technique in computing T(D), we first label the state variables for the code as shown in Fig. 3.3.2.

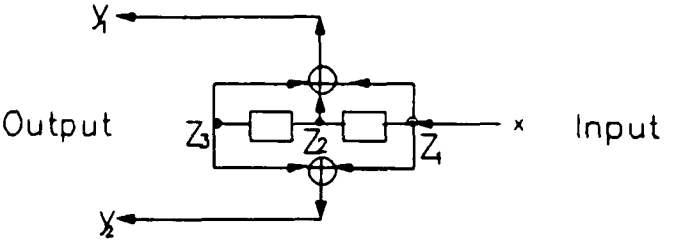

Figure 3.3.2 The encoder of Fig. 3.3.1 assigned with the state variables.

Then

$$\vec{Y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

The state equations are then the following:

$$Z_1(i) = X(i)$$

$$Z_2(i) = X(i-1)$$

$$Z_3(i) = X(i-2)$$

and

$$\vec{Z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}$$

The outputs are

$$y_1(i) = z_1(i) + z_3(i) + z_2(i) \mod 2$$

$$y_2(i) = z_1(i) + z_3(i) \mod 2$$

For k=1, setting

$$\vec{Z}(1) = 0$$

which implies

$$Z_1(1) = 0$$

$$Z_2(1) = 0$$

$$Z_3(1) = 0$$

that implies

$$x(1) = 0$$

$$x(0) = 0$$

Considering that the condition in Eq. (3.2.3) is not satisfied because we have

$$\vec{Z}(0) = \begin{pmatrix} x(0) \\ x(-1) \\ x(-2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

clearly then, for k=1, there is no solution.

For k=2, setting

$$\vec{Z}(2) = 0$$

implies

$$x(2) = 0$$

$$x(1) = 0$$

$$x(0) = 0$$

which requires

$$\vec{Z}(1) \neq 0$$

$$\vec{Z}(0) \neq 0$$

but

$$\vec{Z}(1) = \begin{pmatrix} x(1) \\ x(0) \\ x(-1) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\vec{Z}(0) = \begin{pmatrix} x(0) \\ x(-1) \\ x(-2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Hence, the condition is still not satisfied for k=2.

For k=3, setting

$$\vec{Z}(3) = 0$$

and requiring

$$\vec{Z}(2) \neq 0$$

$$\vec{Z}(1) \neq 0$$

$$\vec{Z}(0) \neq 0$$

implies

$$x(3) = 0$$

$$x(2) = 0$$

$$x(1) = 0$$

which leaves x(0) indeterminate.

Now, for $\vec{Z}(2) \neq 0$ requires that

$$x(0) = 1$$

$\vec{Z}(1) \neq 0$ and $\vec{Z}(0) \neq 0$ are satisfied for x(0)=1. So, for k=3, the only solution to the state equations is x(0)=1. The distance of the only path guaranteed from this solution is computed as follows:

$$|\vec{Y}(3)| = \sum_{i=0}^{2} \sum_{j=1}^{2} y_j(i) = y_1(0) + y_2(0) + y_1(1) + y_2(1) + y_1(2) + y_2(2)$$

where,

$$y_1(0) = z_1(0) + z_2(0) + z_3(0) = x(0) + x(-1) + x(-2) = 1 \mod 2$$

$$y_2(0) = z_1(0) + z_3(0) = x(0) + x(-2) = 1 \mod 2$$

$$y_1(1) = z_1(1) + z_2(1) + z_3(1) = x(1) + x(0) + x(-1) = 1 \mod 2$$

$$y_2(1) = z_1(1) + z_3(1) = x(1) + x(-1) = 0 \mod 2$$

$$y_1(2) = z_1(2) + z_2(2) + z_3(2) = x(2) + x(1) + x(0) = 1 \mod 2$$

$$y_2(2) = z_1(2) + z_3(2) = x(2) + x(0) = 1 \mod 2$$

Thus,

$$|\vec{Y}(3)| = 5$$

Using the procedure described above, a few terms of the generating function have been calculated:

For K=1, 2, Eq.(3.2.2) has no solution, because Eq.(3.2.3) is not satisfied.

For K=3, there is one solution that corresponds to a path with distance 5 denoted by D^5.

For K=4, there is D^6.

For K=5, there are D^6 and D^7.

For K=6, there are $2D^7$, D^8 and D^{10}.

For K=7, there are D^7, $3D^8$, D^9, and $2D^{11}$.

For K=8, there are $3D^8$, $4D^9$, D^{10}, $2D^{11}$ and $3D^{12}$.

There are eight steps in the calculation of the above example. The generating function obtained in the eight steps of calculation is the sum of all the solutions in Eq.(3.2.2) for k=1 to k=8. Thus,

$$T(D) = D^5 + 2D^6 + 4D^7 + \ldots$$

Comparison of the first three terms of T(D), from the above computation, with the first three terms of T(D), from the analytical solution, indicates an exact correspondence. The rest of the terms of T(D) would also match if the steps of calculation were continued.

The free distance can be obtained with confidence when the number of step calculation equals $L+1$, where L is the constraint length. The next term of the generating function can be obtained loosely by $2L+1$ steps. If

the encoder has feedback, the number of steps equals the maximum number of encoder outputs that can be affected by a single information bit. The closed form of the generating function can be obtained if enough terms are calculated.

4 BOUNDS OF THE BIT ERROR RATE FOR THE CONVOLUTIONAL PROJECTION CODE

4.1 LOWER BOUND OF T(D)

The state equation technique can be used to compute terms of the generating function, but it cannot be used to compute all the terms for an infinitely long input sequence. For a given convolutional code, the terms of T(D), in general, do not follow a pattern. That makes obtaining T(D) more difficult. However, for the purpose of finding the error bounds, it is helpful if T(D) can be bounded.

The lower bound of T(D) for a convolutional code or a convolutional Projection code is clearly the first term of the generating function,

$$T(D)_{lower} = c_1 D^{d_{free}} \qquad (4.1.1)$$

A tighter lower bound for T(D) can be obtained by adding more terms to the generating function. C_1 can be calculated by employing the state equation technique.

For a convolutional code without a feedback, I+1 number of steps are required to calculate c_1. For a convolutional Projection code with feedback paths, at most 2L+1 steps are required to calculate c_1, where L is the constraint length of the code.

In practice, the error rate of a powerful convolutional code is dominated by the free distance term of the generating function and the error rate of a powerful block code is dominated by the number of codewords with minimum Hamming weight. This is also true for a powerful convolutional Projection code. Estimation of the lower bound of T(D) enables one to find a lower bound of the bit error rate which in turn is very close to the bit error rate of the code.

4.2 LOWER BOUND OF THE BIT ERROR RATE FOR CONVOLUTIONAL PROJECTION CODE

The lower bound of the bit error rate for the convolutional Projection code can be calculated by choosing any path of the c_1 paths in the generating function. The first-event error probability, P_E, is then a lower bound for the code, assuming that the

probability of a path which deviates and merges repeatedly more than once from the all-zero path is very small. Thus, the first-event error probability lower bound for the BSC is

$$P_E = P_{d_{free}} \tag{4.2.1}$$

Therefore, by weighting P_E by $\frac{B(1)}{k}$, the lower bound of the bit error rate is

$$P_b = \frac{1}{n} d_{free} P_{d_{free}} \tag{4.2.2}$$

If d_{free} *is odd,*

$$P_b = \frac{1}{n} d_{free} \sum_{e=\frac{d_{free}+1}{2}}^{d_{free}} \binom{d_{free}}{e} p^e (1-p)^{d_{free}-e} \tag{4.2.3}$$

If d_{free} *is even,*

$$P_b = \frac{1}{2n} d_{free} \binom{d_{free}}{\frac{d_{free}}{2}} (p(1-p))^{\frac{d_{free}}{2}}$$

$$+ \frac{1}{n} d_{free} \sum_{e=\frac{d_{free}}{2}+1}^{d_{free}} \binom{d_{free}}{e} p^e (1-p)^{d_{free}-e} \tag{4.2.4}$$

4.2.1 LOWER BOUND OF THE BIT ERROR RATE FOR P1 CONVOLUTIONAL PROJECTION CODE

For a Rate-1/(1+r), P1, convolutional Projection code, from Eq.(2.1.2), we have

$$d_{free} = r + 1$$

Combining Eq.(4.2.3), Eq.(4.2.4) and Eq.(2.1.3) with the above equation, one obtains a lower bound of the bit error rate:

If $r+1$ *is odd,*

$$P_{b(lower)} = \sum_{e=\frac{(r+1)-1}{2}}^{r+1} \binom{r+1}{e} p^e (1-p)^{r+1-e} \tag{4.2.1.1}$$

If $r+1$ *is even,*

$$P_{b(lower)} = \frac{1}{2}\binom{r+1}{\frac{r+1}{2}} (p(1-p))^{\frac{r+1}{2}} + \sum_{e=\frac{r+1}{2}+1}^{r+1} \binom{r+1}{e} p^e (1-p)^{r+1-e} \tag{4.2.1.2}$$

For a Rate-k/k+r, P1, convolutional Projection code, combining Eq.(4.2.3) and Eq.(4.2.4) with Eq.(2.1.2), one obtains a lower bound of the bit error rate:

If $r+1$ *is odd,*

$$P_{b(lower)} = \frac{r+1}{k+r} \sum_{e=\frac{(r+1)+1}{2}}^{r+1} \binom{r+1}{e} p^e (1-p)^{r+1-e} \quad (4.2.1.3)$$

If $r+1$ *is even,*

$$P_{b(lower)} = \frac{r+1}{2(k+r)} \binom{r+1}{\frac{r+1}{2}} (p(1-p))^{\frac{r+1}{2}} + \frac{r+1}{k+r} \sum_{e=\frac{r+1}{2}+1}^{r+1} \binom{r+1}{e} p^e (1-p)^{r+1-e} \quad (4.2.1.4)$$

where p is the input bit error rate. The coefficient c_1 is the number of distinct paths with the minimum distance and it can be calculated by employing the state equation technique.

4.2.2 LOWER BOUND OF THE BIT ERROR RATE FOR OPTIMAL P2 CONVOLUTIONAL PROJECTION CODE

The lower bound of the bit error rate for an

optimal P2 convolutional Projection code is calculated below. For a Rate-1/(1+r) code, from Eq.(2.2.3), we have:

$$d_{free} = 2^r$$

Combining Eq.(4.2.4) and Eq.(2.2.4) with the above equation, one obtains a lower bound of the bit error rate:

$$P_{b(lower)} = \frac{2^{r-1}}{r+1}\binom{2^r}{2^{r-1}}(p(1-p))^{2^{r-1}} + \frac{2^r}{r+1}\sum_{e=2^{r-1}+1}^{2^r}\binom{2^r}{e}p^e(1-p)^{2^r-e} \quad (4.2.2.1)$$

For a Rate-k/k+r optimal P2 code, combining Eq.(4.2.3) with Eq.(2.2.5), one obtains a lower bound of the bit error rate:

$$P_{b(lower)} = \frac{1}{k+r}2^{r-1}\binom{2^r}{2^{r-1}}(p(1-p))^{2^{r-1}} + \frac{1}{k+r}2^r\sum_{e=2^{r-1}+1}^{2^r}\binom{2^r}{e}p^e(1-p)^{2^r-e} \quad (4.2.2.2)$$

4.2.3 LOWER BOUND OF THE BIT ERROR RATE FOR P3 CONVOLUTIONAL PROJECTION CODE

The lower bound of the bit error rate for a P3 convolutional Projection code is calculated below.

For a Rate-1/(1+r) code, combining Eq.(4.2.4) with Eq.(2.3.2), one obtains a lower bound of the bit error rate:

$$P_{b(lower)} = \frac{2^{r-1}}{r+1}\binom{2^r}{2^{r-1}}(p(1-p))^{2^{r-1}} + \frac{2^r}{r+1}\sum_{e=2^{r-1}+1}^{2^r}\binom{2^r}{e}p^e(1-p)^{2^r-e} \quad (4.2.3.1)$$

For a Rate-k/k+r P3 code, combining Eq.(4.2.4) with Eq.(2.3.3), one obtains a lower bound of the bit error rate:

$$P_{b(lower)} = \frac{1}{k+r}2^{r-1}\binom{2^r}{2^{r-1}}(p(1-p))^{2^{r-1}} + \frac{1}{k+r}2^r\sum_{e=2^{r-1}+1}^{2^r}\binom{2^r}{e}p^e(1-p)^{2^r-e} \quad (4.2.3.2)$$

4.3 UPPER BOUND OF THE BIT ERROR RATE FOR THE CONVOLUTIONAL PROJECTION CODE

4.3.1 UPPER BOUND OF THE BIT ERROR RATE FOR P1 CONVOLUTIONAL PROJECTION CODE

For a Rate-1/1+r, P1, convolutional Projection code, from Eq.(2.1.3), we have the generating function:

$$T(D) = D^{r+1} + \sum_{j=2}^{\infty} c_j D^{j(r+1)}$$

Combining the above equation with Eq.(1.3.2.6), we have a loose upper bound of the bit error rate for the BSC:

$$P_{b(\text{loose-upper})} = (4p(1-p))^{\frac{(r+1)}{2}} + \sum_{j=2}^{\infty} j c_j (4p(1-p))^{\frac{j(r+1)}{2}} \quad (4.3.1.1)$$

For a Rate-k/k+r, P1, convolutional Projection code, an upper bound can be obtained by finding the generating function.

4.3.2 UPPER BOUND OF THE BIT ERROR RATE FOR OPTIMAL P2 CONVOLUTIONAL PROJECTION CODE

For a Rate-1/1+r, optimal P2, convolutional Projection code, from Eq.(2.2.4), we have the generating function:

$$T(D) = D^{2^r} + \sum_{j=2}^{\infty} c_j D^{2^r+(j-1)2^{r-1}}$$

Combining the above equation with Eq.(1.3.2.6), we have a loose upper bound of the bit error rate for the BSC:

$$P_{b(loose\text{-}upper)} = \frac{2^r}{r+1}(4p(1-p))^{2^{r-1}} + \sum_{j=2}^{\infty} \frac{c_j}{r+1}(j+1)2^{r-1}(4p(1-p))^{(j+1)2^{r-2}} \qquad (4.3.2.1)$$

For a Rate-k/k+r, optimal P2, convolutional Projection code, from Eq.(2.2.5), we have the generating function:

$$T(D) = \sum_{j=1}^{\infty} c_j D^{2^r \cdot (j-1)2}$$

Combining the above equation with Eq.(1.3.2.6), we have a loose upper bound of the bit error rate for the BSC:

$$P_{b(loose\text{-}upper)} = \sum_{j=1}^{\infty} \frac{c_j}{k+r} (2(j-1)+2^r)(4p(1-p))^{(j-1+2^{r-1})} \qquad (4.3.2.2)$$

4.3.3 UPPER BOUND OF THE BIT ERROR RATE FOR P3 CONVOLUTIONAL PROJECTION CODE

For a Rate-1/1+r, P3, convolutional Projection code, from Eq.(2.3.2), we have the generating function:

$$T(D) = D^{2^r} + \sum_{j=2}^{\infty} c_j D^{2^r \cdot (j-1)2^{r-1}}$$

Combining the above equation with Eq.(1.3.2.6), we have a loose upper bound of the bit error rate for the BSC:

$$P_{b(loose\text{-}upper)} = \frac{2^r}{r+1}(4p(1-p))^{2^{r-1}} + \sum_{j=2}^{\infty} \frac{c_j}{r+1}(j+1)2^{r-1}(4p(1-p))^{(j+1)2^{r-2}} \qquad (4.3.3.1)$$

For a Rate-k/k+r, P3, convolutional Projection code, from Eq.(2.3.3), we have the generating function:

$$T(D) = \sum_{j=1}^{\infty} c_j D^{2^r+(j-1)2}$$

Combining the above equation with Eq.(1.3.2.6), we have a loose upper bound of the bit error rate for the BSC:

$$P_{b(loose\text{-}upper)} = \sum_{j=1}^{\infty} \frac{c_j}{k+r}(2(j-1)+2^r)(4p(1-p))^{(j-1+2^{r-1})} \qquad (4.3.3.2)$$

5 RESULTS

5.1 COMPUTATION OF THE UPPER AND LOWER BOUNDS FOR A RATE-1/2, P3, CONVOLUTIONAL PROJECTION CODE

A Rate-1/2, P3, convolutional Projection with slopes of 1, 1/2 and 1/7 is encoded by the procedure in Chapter 2 and the upper and lower bounds of the bit error rate are obtained. The encoder is implemented as shown in Fig. 5.1.1.

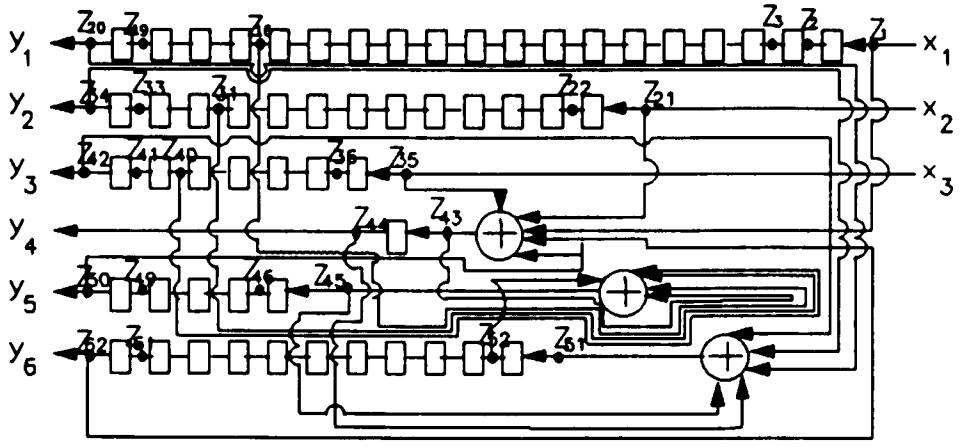

Figure 5.1.1 The encoder of a Rate-1/2, P3, convolutional Projection code with slopes of 1, 1/2 and 1/7.

Labeling each connecting point on the encoder by the state variable in order, from top to bottom and right to left results in

the state equations:

$$Z_1(i) = X_1(i)$$
$$Z_2(i) = X_2(i-1)$$
$$\vdots$$
$$Z_{20}(i) = X_1(i-19)$$

$$Z_{21}(i) = X_2(i)$$
$$Z_{22}(i) = X_2(i-1)$$
$$\vdots$$
$$Z_{34}(i) = X_2(i-13)$$

$$Z_{35}(i) = X_3(i)$$

$$Z_{36}(i) = X_3(i-1)$$

.

.

.

$$Z_{42}(i) = X_3(i-7)$$

$$Z_{44}(i) = Z_{43}(i-1)$$

$$Z_{46}(i) = Z_{45}(i-1)$$

$$Z_{47}(i) = Z_{45}(i-2)$$

.

.

.

$$Z_{50}(i) = Z_{45}(i-5)$$

$$Z_{52}(i) = Z_{51}(i-1)$$

$$Z_{53}(i) = Z_{51}(i-2)$$

.

.

.

$$Z_{62}(i) = Z_{51}(i-11)$$

There are three recursive equations:

$$Z_{45}(i) = x_1(i-15) + x_2(i-10) + x_3(i-5) + Z_{43}(i) + Z_{51}(i-1) \mod 2$$

$$Z_{43}(i) = x_1(i) + x_2(i) + x_3(i) + x_1(i-20) + x_2(i-15) + x_3(i-10) + Z_{51}(i-11) + Z_{51}(i-6) + Z_{43}(i-5) \mod 2$$

$$Z_{51}(i) = x_1(i-19) + x_2(i-13) + x_3(i-7) + x_1(i-15) + x_2(i-10) + x_3(i-5) + Z_{43}(i) + Z_{43}(i-1) + Z_{51}(i-1) \mod 2$$

The output equations are:

$$y_1(i) = Z_{20}(i)$$

$$y_2(i) = Z_{34}(i)$$

$$y_3(i) = Z_{42}(i)$$

$$y_4(i) = Z_{44}(i)$$

$$y_5(i) = Z_{50}(i)$$

$$y_6(i) = Z_{62}(i)$$

Up to the 11th step of calculation, there is no solution because the conditions of the state equation technique are not satisfied. Starting at $K = 12$, there

are solutions as follows

For K = 12, $|\vec{Y}(12)| = 8$

For K = 13, $|\vec{Y}(13)| = 12$

The calculation of the coefficients is carried out by a computer program devised specially for the P3 convolutional Projection code and it is found that $c_1 = 15$ $c_2 = 0$, $c_3 = 285$, and $c_4 = 949$. The generating function, T(D), of the code can be expressed as

$$T(D) = 15D^8 + 285D^{12} + 949D^{14} + \ldots$$

Using Eq.(4.2.3.2) and Eq.(4.3.3.2), the lower and upper bounds of the bit error rate for the BSC are given by,

$$P_{b(lower)} = \frac{2}{3}\binom{8}{4}(p(1-p))^4 + \frac{4}{3}\sum_{e=5}^{8}\binom{8}{e}p^e(1-p)^{8-e}$$

$$P_{b(upper)} = 10\binom{8}{4}(p(1-p))^4 + 20\sum_{e=5}^{8}\binom{8}{e}p^e(1-p)^{8-e}$$
$$+ 285\binom{12}{6}(p(1-p))^6 + 570\sum_{e=7}^{12}\binom{12}{e}p^e(1-p)^{12-e}$$
$$+ 1107\binom{14}{7}(p(1-p))^7 + 2214\sum_{e=8}^{14}\binom{14}{e}p^e(1-p)^{14-e} + \ldots$$

$$P_{b(loose\text{-}upper)} = 20(4p(1-p))^4 + 570(4p(1-p))^6$$
$$+ 2214(4p(1-p))^7) + \ldots$$

5.2 COMPARISON OF THE BOUNDS WITH SIMULATION RESULTS

The lower and upper bounds of the bit error rate calculated in section 5.1 are tabulated together with simulation results in Table 5.2.1 , Table 5.2.2 and Table 5.2.3. The simulation in the tables was performed using the ad hoc decoding approach of Schilling & Lomp [14], while the lower and upper bounds were obtained for optimal decoding.

Table 5.2.1 Comparison of the lower and upper bounds with the simulation result for a Rate-1/2, P3, code of blocklength 31 with slopes 1, 1/2, and 1/7.

$p_{(input)}$	$P_{b(simulation)}$	$P_{b(upper)}$	$P_{b(lower)}$	$P_{b(loose-upper)}$
$1.00*10^{-2}$	$5.00*10^{-5}$	$7.12*10^{-6}$	$4.56*10^{-7}$	$5.17*10^{-5}$
$2.00*10^{-2}$	$1.00*10^{-3}$	$1.26*10^{-4}$	$7.11*10^{-6}$	$9.28*10^{-4}$
$3.00*10^{-2}$	$4.00*10^{-3}$	$7.66*10^{-4}$	$3.51*10^{-5}$	$5.70*10^{-3}$
$4.00*10^{-2}$	$1.40*10^{-2}$	$3.10*10^{-3}$	$1.08*10^{-4}$	$2.31*10^{-2}$
$5.00*10^{-2}$	$3.00*10^{-2}$	$9.50*10^{-3}$	$2.58*10^{-4}$	$7.27*10^{-2}$

Table 5.2.2 Comparison of the lower and upper bounds with the simulation result for a Rate-1/2, P3, code of blocklength 139 with slopes 1, 1/2, and 1/7.

$P_{(input)}$	$P_{b(simulation)}$	$P_{b(upper)}$	$P_{b(lower)}$	$P_{b(loose-upper)}$
$1.00*10^{-2}$	$4.10*10^{-5}$	$7.12*10^{-6}$	$4.56*10^{-7}$	$5.17*10^{-5}$
$2.00*10^{-2}$	$7.00*10^{-4}$	$1.26*10^{-4}$	$7.11*10^{-6}$	$9.28*10^{-4}$
$3.00*10^{-2}$	$3.00*10^{-3}$	$7.66*10^{-4}$	$3.51*10^{-5}$	$5.70*10^{-3}$
$4.00*10^{-2}$	$1.10*10^{-2}$	$3.10*10^{-3}$	$1.08*10^{-4}$	$2.31*10^{-2}$
$5.00*10^{-2}$	$2.60*10^{-2}$	$9.50*10^{-3}$	$2.58*10^{-4}$	$7.27*10^{-2}$

Table 5.2.3 Comparison of the lower and upper bounds with the simulation result for a Rate-1/2, P3, convolutional Projection code.

$P_{(input)}$	$P_{b(simulation)}$	$P_{b(upper)}$	$P_{b(lower)}$	$P_{b(loose-upper)}$
$8.00*10^{-3}$	$3.00*10^{-5}$	$2.88*10^{-6}$	$1.87*10^{-7}$	$2.10*10^{-5}$
$1.00*10^{-2}$	$1.30*10^{-4}$	$7.12*10^{-6}$	$4.56*10^{-7}$	$5.17*10^{-5}$
$2.00*10^{-2}$	$1.90*10^{-3}$	$1.26*10^{-4}$	$7.11*10^{-6}$	$9.28*10^{-4}$

Further, the data in Table 5.2.1, Table 5.2.2 and Table 5.2.3 were plotted as shown in Fig.5.2.1, Fig.5.2.2 and Fig.5.2.3 respectively.

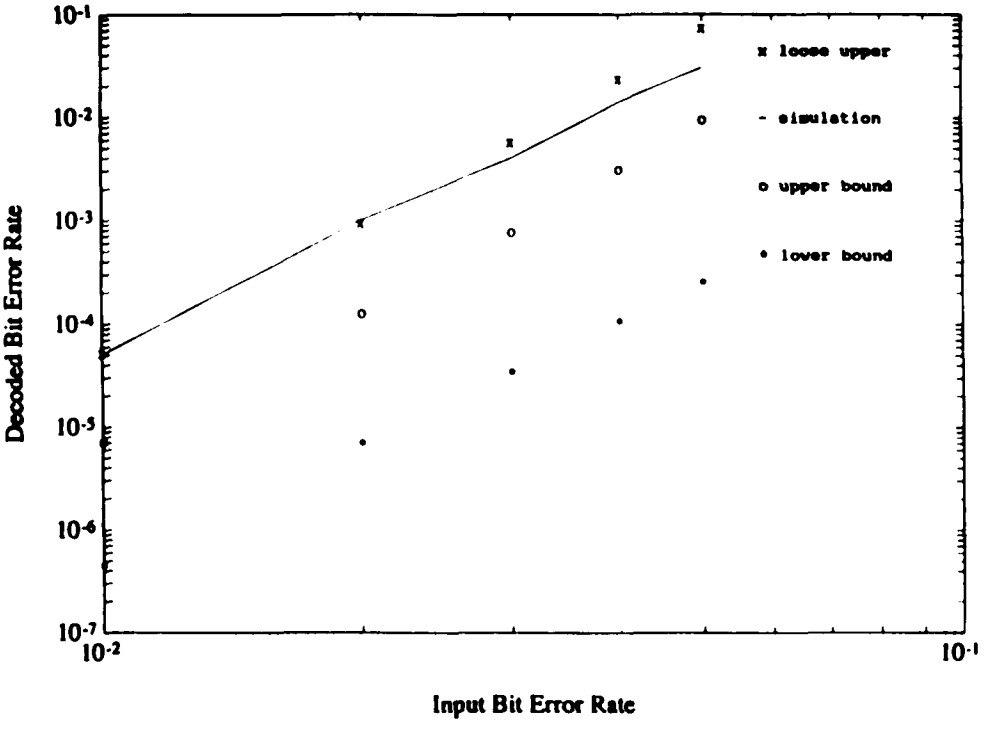

Figure 5.2.1 Comparison of the bounds with simulation for a Rate-1/2, P3, code with blocklength 31.

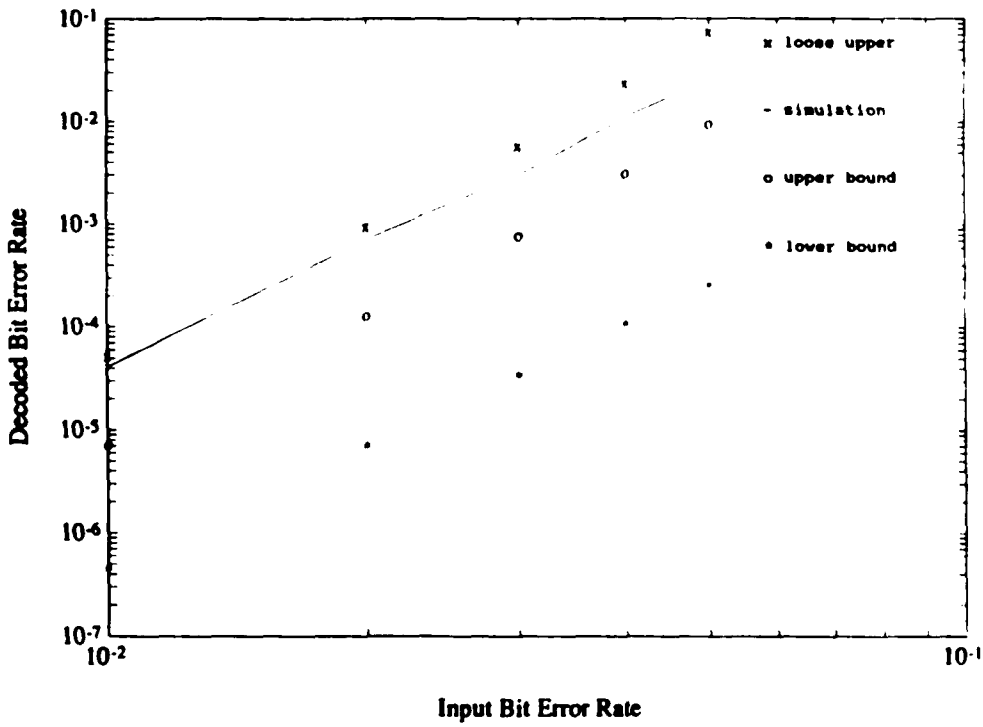

Figure 5.2.2 Comparison of the bounds with simulation for a Rate-1/2, P3, code with blocklength 139.

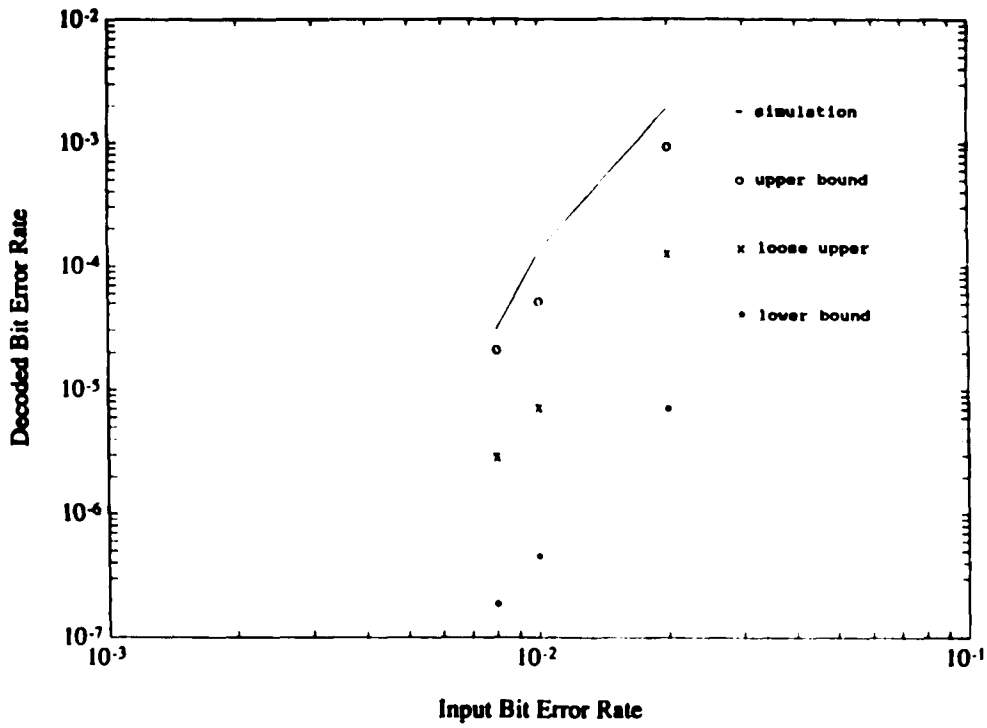

Figure 5.2.3 Comparison of the bounds with simulation for a Rate-1/2, P3, convolutional code.

6 CONCLUSIONS

The convolutional Projection codes have better structure than most convolutional codes in the areas of error rate performance design and simple encoding procedure. In addition, the encoding technique for the convolutional Projection code has an advantage for a code designer for finding an optimal convolutional code with respect to a large constraint length without computer simulations. All these advantages give the Projection code superior performance than most existing codes.

The structures of the convolutional P1, P2 and P3 codes have been studied, and their generating functions derived. Furthermore, lower and upper bounds on the bit error rate were obtained for the Binary Symmetric Channel.

The state equation technique derived is much simpler to use than the state-diagram technique in obtaining the generating function for a convolutional code. It also makes it possible to obtain the generating function for convolutional Projection codes, while the state-diagram technique fails to handle such codes with very large constraint length.

7 REFERENCES

[1] A. M. Patel and S. J. Hong, " Optimal Rectangular Codes for High Density Magnetic Tapes" IBM J. Res. Dev., Vol. 18, pp. 579-588, November 1974.

[2] P. Prusinkiewicz and S. Budkowski, " A Double Track Error-Correction Code for Magnetic Tape ", IEEE Tran. On Computers, pp. 642-645, June 1976.

[3] D. Manela, "A new class of forward error correcting codes for burst and random errors", Doctoral dissertation of the City University of New York, 1987.

[4] G. R. Lomp and D. L. Schilling, " A New Burst and Random Error Correcting Code: The Projection Code", IEEE Int'l. Symposium on Inf. Theory, January 1990.

[5] G. Yang, "Bounds on the block Projection code", Doctoral dissertation of the City University of New York, 1993.

[6] H. Taub, D.L. Schilling, "Principles of Communication System", 2nd ed., pp564-566, McGraw-Hill, New York, 1986.

[7] H. Taub, D.L. Schilling, "Principles of Communication System", 2nd ed., pp569-571, McGraw-Hill, New York, 1986.

[8] A.J. Viterbi, J.K. Omura, "Principles of Digital Communication", McGraw-Hill, New York, 1979.

[9] R.E. Blahut, "Theory and practice of error control codes", Addison Wesley, New York, 1983.

[10] A.J. Viterbi, "Convolutional Codes and their Performance in Communication Systems" IEEE Tran. Comm. Tech., pp752-756, Oct. 1971.

[11] A.J. Viterbi, "Convolutional Codes and their Performance in Communication Systems" IEEE Tran. Comm. Tech., pp759-760, Oct. 1971.

[12] D. Manela, "A new class of forward error correcting codes for burst and random errors", Doctoral dissertation of the City University of New York, pp108-110, 1987.

[13] A.J. Viterbi, "Convolutional Codes and their Performance in Communication Systems" IEEE Tran. Comm. Tech., pp755, Oct. 1971.

[14] Gary R. Lomp, "A new burst and random error correcting code: Non-binary Projection code", A final report for NSF, SCS Telecom, pp34, July 1989.