This dissertation has been microfilmed exactly as received

LIN, Shu, 1936-
AN INVESTIGATION OF CODES AND DECODING PROCEDURES FOR SEQUENTIAL DECODING SYSTEMS.

Rice University, Ph.D., 1965
Engineering, electrical

University Microfilms, Inc., Ann Arbor, Michigan
RICE UNIVERSITY

AN INVESTIGATION OF CODES AND DECODING PROCEDURES

FOR SEQUENTIAL DECODING SYSTEMS

by

Shu Lin

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

Thesis Director's signature:

Paul E. Pfeiffer

Houston, Texas
March, 1965
This research can be divided into three parts:

In part one, important algebraic and distance properties of convolutional tree codes are studied, with a view to developing good codes for sequential decoding systems. A fundamental distance property is developed; this makes it possible to introduce a related threshold function for paths of various length. This function specifies an error-correcting capability for the three codes, analogous to that for block codes.

In part two, algorithms for efficient probabilistic decoding on a sequential basis have been proposed. In particular, strategies have been devised to utilize the full error-correcting capability of codes and to deal with the difficult problem of error propagation, seemingly inherent in sequential decoding systems. Use of a multiple-threshold test based on distance properties of the code enables the decoder to adjust its operation to the noise conditions in such a way that reasonably efficient operation is carried out for each noise situation. Although estimates of probability of error and numbers of decoding operations have been based on the binary symmetric channel model, the procedures have been designed to deal with burst errors within the error-correcting capability of the code.

In part three, the concepts of threshold block decoding and an easy way to evaluate reasonable thresholds are described. In a modified manner, these thresholds are applied to Wozencraft's sequential decoding scheme; and it is shown that the average number of computations required to process the incorrect subset is bounded by a quantity which is independent of
the constraint length $n_t$ for $R_t < R_{comp}$. It is also shown that
the probability of error decreases exponentially with $n_t$. The results
obtained in this part are different from the results obtained by
Wozencraft and Reiffen. All the analysis in this part is based on random
coding arguments.
ACKNOWLEDGEMENTS

The author would like to express his deepest appreciation to Dr. Paul E. Pfeiffer, Director of Research, for his constant inspiration, valuable ideas, good suggestions and patient guidance of this research. None of this research would have been possible without his assistance in these years.

Many thanks are due to his fellow graduate student, H. W. Lyne, for a number of suggestions and criticisms.

The author wishes also to thank the Texaco, Inc., whose fellowship made this research possible. This research was part of a project supported by the U. S. Army Research Office - Durham.

In particular, the author is indebted to his beloved wife, Ivy, for her constant encouragement and patience.
TABLE OF CONTENTS

ABSTRACT

ACKNOWLEDGEMENTS

Chapter I. INTRODUCTION

1-1. The Coding Problem

1-2. The Binary Symmetric Channel

Chapter II. CONVOLUTIONAL TREE CODES

2-1. Infinite Binary Tree Codes with Transmission Rate \( R_t = \frac{1}{n_o} \)

2-2. Truncated Binary Trees

2-3. Binary Tree Codes with Transmission Rate \( R_t = \frac{k_o}{n_o} \)

2-4. Canonical Form

2-5. Some Properties of the Initial Truncated Tree Codes

2-6. Construction of Good Generator Sequences

2-7. Non-binary Convolutional Tree Codes

Chapter III. A SEQUENTIAL DECODING SYSTEM UTILIZING DISTANCE

PROPERTIES OF CONVOLUTIONAL TREE CODES

3-1. Fundamental Distance Property for a Convolutional Tree Code

3-2. Probabilistic Decoding

3-3. Decoding Procedure

The Threshold Test for Incorrect Paths

Multiple-threshold Test

Error-correcting Strategy

A Search Procedure

3-4. Basic Starting Procedure

3-5. Periodic Restart of Decoding

3-6. Average Number of Decoding Operations

3-7. Strategies for Repeat or Store and Reprocess
3-8. Adaptive Operation
3-9. Synchronization
3-10. Implementation

Chapter IV. A NEW ANALYSIS FOR WOZENCRAFT'S SEQUENTIAL DECODING SCHEME

4-1. Threshold Block Decoding
4-2. Sequential Decoding (by Wozencraft)
   Basic Concept
   Decoding Algorithm
   Average Number of Computations in Processing the Incorrect Subset
   Probability of Error

Chapter V. CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

5-1. Conclusion
5-2. Recommendation for Future Research

APPENDICES

A. BOUNDS ON SUMS OF RANDOM VARIABLES

B. A STUDY OF THE RANDOM VARIABLES OF A BINARY SYMMETRIC CHANNEL

REFERENCES
CHAPTER I. INTRODUCTION

1-1. The Coding Problem

Since the fundamental work of C. E. Shannon on mathematical theory of information in 1948, great progress has been made in the field of transmission of information. In the course of its development, information theory has been divided into two branches. The first one is concerned with the general properties of various channels and with showing their capability for transmission of information. The second branch is concerned with combating the random noise which disturbs the channel during the transmission of information in order to attain reliable communication; this branch is called coding theory. At the present stage, the first branch is further developed than the latter one.

In respect to the coding of a noisy channel for achieving reliable data transmission, two tasks have to be accomplished. The first one is the transformation of a raw message into a suitable signal to be transmitted over the noisy channel. This is called the encoding operation, which is equivalent to the construction of a code. The second task is concerned with the reproduction of the original signal upon receiving the disturbed signal at the channel output. This is called the decoding operation. Classically, the coding problem has been approached by means of block coding. Even though the theoretical effectiveness of block codes has been proved, the practical implementation of coding is still a problem to be solved. The major problem in the block coding is that the optimum decoding procedure requires either an amount of storage which increases exponentially with the block length $n$, or an amount of computations which also increases exponentially with $n$. This implies
that the complexity of decoding equipment grows exponentially with \( n \). In order to achieve reliable communication, the block length \( n \) has to be large. This large block length \( n \) will then cause the decoding to be impractical.

In 1957, Wozencraft of M.I.T. proposed an interesting coding scheme by which the decoding computational complexity increases only algebraically with \( n \) and the average probability of error over a defined ensemble decreases exponentially with \( n \). This decoding scheme is the well-known sequential decoding scheme. The introduction of the concept of sequential decoding represents an important breakthrough in the attempts to make practical application of information theory to communication system. Since 1957, Wozencraft's work has been extended by B. Reiffen, and other different sequential decoding schemes have been proposed by Ziv and Fano. The common feature of this previous work is that the analysis and the determination of discarding thresholds are based on the random coding argument which was introduced by Shannon.

In this research, a new sequential type decoding scheme for binary channels has been proposed. This decoding scheme is also probabilistic in nature, but it intends to utilize the algebraic properties of the codes; the determination of discarding thresholds is based only on the distance property of codes. It seems that this decoding scheme is more realistic and conceptually simpler than the other schemes. Simulation studies of this decoding scheme are currently being performed by H. Lyne.
1-2. The Binary Symmetric Channel

In the following discussion, we are concerned with the encoding and decoding of messages for transmission through a binary symmetric channel, which is the simplest abstraction of physical channels. A binary symmetric channel (abbreviated BSC) is defined by the transition probability diagram of Fig. 1-1. The channel accepts binary symbols, for example \( \{0,1\} \), at its input, and produces binary symbols at its output. Each input symbol has a probability \( p_0 < \frac{1}{2} \) of being received incorrectly, and a probability \( q_0 = 1 - p_0 \) of being received correctly. The transition probability is constant and independent of the value of the symbol being transmitted -- i.e., the channel has no memory --; the channel is as likely to change a "1" into an "0" as to change an "0" into a "1".

The communication system we are concerned with may be symbolically represented as shown in Fig. 1-2. The message source generates a sequence \( m \) of binary symbols "0" and "1". We will call this sequence \( m \) the information sequence or message. The encoder transforms this information sequence \( m \) into a longer sequence \( u \) suitable for transmission through the BSC. The channel then disturbs the input sequence, and delivers the noisy version \( v \) to the decoder. The task of the decoder -- knowing the channel transition probability \( p_0 \), the noisy received sequence \( v \), and the encoding rules for transforming \( m \) into \( u \) -- is to reproduce the information sequence or message \( m \). The noise is a random sequence of binary symbols, synchronous with the input sequence.

Given such a binary symmetric channel, the coding problem is to determine an encoding scheme whereby any information sequence \( m \) is encoded into a sequence \( u \) such that the decoder can uniquely and
with arbitrarily high probability reproduce $m$ in spite of the channel disturbances. Therefore, we are interested in not only the encoding problem but also the decoding problem.
Fig. 1-1.

The Binary Symmetric Channel

Fig. 1-2.

Communication over a Binary Symmetric Channel
CHAPTER II. CONVOLUTIONAL TREE CODES

In this chapter, a fairly complete theory of tree codes (or convolutional codes) is developed and an algorithm for step-by-step construction of tree code generator sequences is proposed. The generators constructed in this way will generate trees with a fairly large portion of minimum distance in the first few stages. This quickly diverging distance property will be exploited in decoding. Although most of the work in the chapter is on the binary case, the theory developed can be extended easily to the non-binary case with little modification.
2-1. Infinite Binary Tree Codes with Transmission Rate $R_t = \frac{1}{n_o}$

For simplicity, we consider first the case for $R_t = 1/n_o$ (i.e., each message digit will be encoded into $n_o$ channel input digits). Let $\bar{g}$ be an infinite sequence of binary digits. In this sequence $\bar{g}$, only the first $n_t = \gamma n_o$ digits may differ from zero, the remaining digits are all zeros

$$\bar{g} = (g_1 g_2 g_3 \ldots \ldots g_{n_t} 000\ldots)$$

(2-1)

We also require that $g_1 = 1$.

Let us define $D$ as a shift operator such that

$$D g = \overbrace{(00\ldots0 \ g_1 g_2 g_3 \ldots \ldots g_{n_t} 000\ldots)}^{n_o}$$

(2-2)

That is, when $D$ operates on $\bar{g}$, it shifts every digit of $\bar{g}$ to the right by $n_o$ places. $D^2$ shifts every digit of $\bar{g}$ to the right by $2n_o$ places,

$$D^2 g = \overbrace{(00\ldots0 \ g_1 g_2 g_3 \ldots \ldots g_{n_t} 000\ldots)}^{2n_o}$$

(2-3)

Thus, the operator $D^k$ shifts $\bar{g}$ to the right by $kn_o$ places, when it is applied to $\bar{g}$.

Now we are interested in the following set (infinite) of sequences (infinite), each of which is of the form $D^k g$, where $k = 0, 1, 2, 3, \ldots$

$$\bar{g} = (g_1 g_2 g_3 \ldots \ldots g_{n_t} 000\ldots)$$

$$D g = \overbrace{(00\ldots0 \ g_1 g_2 g_3 \ldots \ldots g_{n_t} 000\ldots)}^{n_o}$$

$$D^2 g = \overbrace{(00\ldots0 \ g_1 g_2 g_3 \ldots \ldots g_{n_t} 000\ldots)}^{2n_o}$$

$$\vdots$$

$$D^k g = \overbrace{(00\ldots0 \ g_1 g_2 g_3 \ldots \ldots g_{n_t} 000\ldots)}^{kn_o}$$
For convenience, we will treat each of these sequences as a vector. From the nature of the shift operation, it is obvious that this forms a set of linearly independent vectors over the binary field. If we use the infinite set \( \{ \vec{g}, D\vec{g}, D^2\vec{g}, \ldots \} \) as a base or \( G_\infty \) shown in Fig. 2-1 as a generator matrix, this will generate a vector space \( U_\infty \) of infinite dimension over the binary field. Each element of \( U_\infty \) can be expressed as

\[
\vec{u} = m_1 \vec{g} \oplus m_2 D\vec{g} \oplus m_3 D^2\vec{g} \oplus \ldots = \sum_{i=0}^{\infty} m_{i+1} D^i\vec{g}
\]

where \( m_i \in GF(2) \), and all the operations are modulo-2.

For convenience of discussion, we define the first \( n_0 \) digits of \( \vec{g} \) as the first segment of \( \vec{g} \) and denote it with

\[ g(1) = g_1 g_2 \ldots g_{n_0} \]

and the second \( n_0 \) digits of \( \vec{g} \) as the second segment of \( \vec{g} \) and denote it with

\[ g(2) = g_{n_0+1} g_{n_0+2} \ldots g_{2n_0} \]

Thus, we have

\[
\begin{align*}
g(3) &= g_{2n_0+1} g_{2n_0+2} \ldots g_{3n_0} \\
g(v) &= g_{(v-1)n_0+1} g_{(v-1)n_0+2} \ldots g_{vn_0} \\
g(v+1) &= 00 \ldots 0 \\
g(v+2) &= 00 \ldots 0 \\
\end{align*}
\]

where \( v = n_t / n_0 \).
We will denote any $n_o$ zero digits with $[0]$. 

Then, we obtain

$$g(k) = g(k-1)n_o + 1 g(k-2)n_o + 2 \cdots g(k-n_o)$$  
$$g(k) = [0]$$  
$$k > v$$  

and

$$d^i_g = [0][0][\ldots][0] g(1) g(2) \cdots g(v)[0][0][\ldots]$$

The generator matrix $G_\infty$ is shown in Fig. 2-2.

The vector space $U_\infty$ generated by $G_\infty$ has a very interesting structure. The set of elements may be partitioned into two equal subsets. From Eq. 2-5 and Fig. 2-2, we can easily see that one subset of the elements corresponds to $m_1 = 0$, and these elements have the same prefix $[0]$ consisting of $n_o$ zeros; the other subset of elements corresponds to $m_1 = 1$, and these elements have the same prefix $g(1)$ consisting of $n_o$ digits. This partitioning is diagrammed in Fig. 2-3. We can also partition the upper half into two equal subsets, one subset corresponds to $m_1 = 0$ and $m_2 = 0$, and all the elements will have the prefix $[0][0]$; the second subset corresponds to $m_1 = 0$ and $m_1 = 1$, and the elements in this subset will have the prefix $[0]g(1)$. The lower half in Fig. 2-3 can also be divided into two equal subsets. One corresponds to $m_1 = 1$ and $m_2 = 0$, all the elements in this subset have the same prefix $g(1)$ $g(2)$; the other subset corresponds to $m_1 = 1$ and $m_2 = 1$, all the elements in it have the same prefix $g(1)$ $g(1) \oplus g(2)$. This is shown in Fig. 2-4. The partitioning process can be carried on indefinitely. Thus, the vector space $U_\infty$ has an infinite tree structure with nodes spaced $n_o$ digits apart and two branches stemming from each node. Each branch corresponds
to a particular binary message digit. A topological representation of this infinite tree is shown in Fig. 2-5.

In the coding of a channel with binary input, the vector space \( U_m \) can be used as a code. If the message sequence consists of a sequence of binary digits \( \bar{m} = (m_1 m_2 m_3 \ldots) \) then the output of the channel encoder will be

\[
\bar{u} = m_1 \bar{g} \oplus m_2 \bar{Dg} \oplus m_3 \bar{D^2g} \oplus \ldots.
\]

(2-9)

which is a path in the infinite tree. In continuous communication, the message sequence \( \bar{m} \) is infinitely long, so that the encoder output (or channel input) \( \bar{u} \) will also be infinitely long. The encoding operation can be viewed as the process in which the encoder traces a particular path through the tree under the instructions given by the message digits. Each message digit gives instructions as follows: Let \( m_k \) be the \( k \)th digit of the message sequence \( \bar{m} \). For \( m_k = 0 \), the encoder selects the upper branch stemming from the \( k-1 \)th order node. For \( m_k = 1 \), the encoder selects the lower branch. The sequence of selections at consecutive nodes determines the encoder output (or channel input) uniquely.

Example: Let \( n_t = 9 \), \( n_o = 3 \) and \( \bar{g} = (111 \ 010 \ 001 \ 000 \ldots) \)

The tree will be that shown in Fig. 2-6.

If \( \bar{m} = (11 \ 00 \ldots) \)

then \( \bar{u} = (111 \ 101 \ 011 \ 011 \ldots) \)

Codes of this class are called **convolutional tree codes**; the corresponding encoding operation is called **convolutional encoding**. These codes were proposed by P. Elias [15] and J. M. Wozencraft [1]. When a convolutional code is used for coding, the encoding circuitry is very simple, consisting of only a shift register with \( v = n_t/n_o \) stages and \( n_o \) modulo-2
adders. Figure 2-7 shows a suitable circuit configuration and illustrates a step in the generation of a code.

The connections of the \( n_0 \) adders to each stage of the shift register are as follows: Divide the first \( n_t \) digits of \( \bar{g} \) into \( \nu \) successive blocks of \( n_0 \) digits, the \( k \)th segment

\[
g(k) = g(k-1)n_0 + 1 \cdot g(k-1)n_0 + 2 \cdots g(k)n_0\]

will determine the connections between the \( k \)th stage of the register and the \( n_0 \) adders: For \( g(k)n_0 + 1 = 1 \), there is a connection from the \( k \) stage to first adder. For \( g(k)n_0 + 1 = 0 \), there is no connection. The connection between the \( k \)th stage and the \( i \)th adder depends on \( g(k-1)n_0 + i \), with \( i \leq n_0 \).

The general encoder circuit is shown in Fig. 2-7a, and the circuit for previous example is shown in Fig. 2-7b. Every time the message sequence \( \bar{m} \) shifts through the register by one digit, there will be a block of \( n_0 \) digits output at the \( n_0 \) adders.

From Eq. 2-5, we can see that the \( (k+1) \)-th message digit \( m_{k+1} \) determines whether or not \( D^k_{\bar{g}} \) is to be included in the formation of the channel input. Since \( D^k_{\bar{g}} \) has an effective length of \( n_t \) digits (i.e., the non-zero digits can exist only from \( (kn_0 + 1) \)th place to \( (kn_0 + n_t) \)th place), it is clear that each message digit \( m_k \) only affects the channel input over a span \( n_t \) digits, which we will call the constraint length. In the sequential decoding scheme proposed by Wozencraft [1], the decoder decodes the message digits one at a time; therefore, the decoder is restricted to examine no more than \( n_t \) digits of the received sequence \( \bar{v} \).
2-2. **Truncated Binary Trees**

Let us denote by \([D^i_g] \) the truncated sequence which contains only the first \(n_t\) digits of \(D^i_g\). Thus, \([D^i_g]\) consists of \(n_t\) zero digits if \(i \geq \nu = n_t/n_0\).

Now, consider the following set of truncated sequences

\[
[[\bar{g}], [D_g], ... [D^{\nu-1}_g]]
\]

\[
[\bar{g}] = g_1g_2g_3 \ldots g_{n_t}
\]

\[
[D_g] = (00 \ldots 0 g_1g_2g_3 \ldots g_{n_t-n_0}^{2n_0})
\]

\[
[D^2_g] = (00 \ldots 0 g_1g_2g_3 \ldots g_{n_t-2n_0})
\]

\[
\vdots
\]

\[
[D^{\nu-1}_g] = (00 \ldots 0 g_1g_2 \ldots g_{n_0})
\]

(2-10)

As we know, these \(\nu\) truncated sequences are linearly independent over the binary field \(\text{GF}(2)\) (with \(g_1 = 1\)). If we use this set as a base, or use the matrix shown in Fig. 2-8 as the generator matrix, it will generate a vector space \(U\) of dimension \(\nu = n_t/n_0\) with a finite tree structure.

Each path has the form

\[
[\bar{u}] = m_1[\bar{g}] \oplus m_2[D_g] \oplus \ldots \oplus m_\nu[D^{\nu-1}_g]
\]

(2-11)

\([\bar{u}]\) is actually the truncated sequence of the infinite channel input sequence \(\bar{u}\) of Eq. 2-5,

\[
[\bar{u}] = [m_1\bar{g} \oplus m_2D_g \oplus \ldots \oplus m_\nu D^{\nu-1}_g \oplus m_{\nu+1} D^{\nu}_g \oplus \ldots]
\]

\[
= m_1[\bar{g}] \oplus m_2[D_g] \oplus \ldots \oplus m_\nu[D^{\nu-1}_g] \oplus m_{\nu+1}[D^{\nu}_g] \oplus \ldots
\]

\[
= m_1[\bar{g}] \oplus m_2[D_g] \oplus \ldots \oplus m_\nu[D^{\nu-1}_g]
\]

where \([D^i_g] = (000 \ldots)\) for \(i \geq \nu\).

From this fact, the finite tree generated by \(G\) in Fig. 2-8, or the set of \(\nu\) truncated sequences \([[\bar{g}],[D_g],...[D^{\nu-1}_g]]\) is the truncated
tree of the infinite tree generated by $G_\infty$ in Fig. 2-1. We will call this tree the initial truncated tree.

We may also define a finite tree having paths $\nu = n_t/n_0$ branches long, starting from any node of the infinite tree as a truncated tree. The truncated tree emerging from any kth order node will be called a kth-order truncated tree. The zero-order truncated tree is the initial truncated tree. Since there are $2^k$ kth-order nodes, there are $2^k$ truncated trees of kth order. Each of these $2^k$ truncated trees depends on the path of $k$ branches preceding it.

Next, we are going to examine the truncated tree more carefully and prove a theorem which is very important in the decoding problem. For convenience, we will call $\tilde{g}$ or its truncated sequence $[^{\tilde{g}}]$ the generator sequence for the tree code.

Consider the generator sequence $\tilde{g}$

$$\tilde{g} = g_1 g_2 g_3 \ldots \ldots g_n 000 = g(1) g(2) \ldots g(\nu) [0][0][0] \ldots$$

Let us define the subsequence of $\tilde{g}$ starting at any segment as the tail sequence of $\tilde{g}$ and denote it with

$$T^i g = g(i) g(i+1) g(i+2) \ldots$$

(2-11a)

Thus

$$T^i g = [0][0][0] \ldots \quad \text{for } i > \nu$$

(2-11b)

We also define the first $\nu$ segments of a tail sequence as the truncated tail sequence and denote it with

$$[T^i \tilde{g}] = g(i) g(i+1) \ldots \ldots g(i+\nu-1)$$

(2-12)

For $i > \nu$

$$[T^i \tilde{g}] = [0][0][0] \ldots [0]$$

(2-13)

Consider the set of truncated paths of $\nu$ branches long cut between
the \( i^{th} \) order node and \((i + \nu)\)th order node. There are \(2^{i+\nu}\) of these paths. These \(2^{i+\nu}\) truncated paths are generated by the truncated matrix beginning from \((i + 1)\)th column to \((i + \nu)\)th column of \(G_\infty\) as shown in Fig. 2-9. They can be partitioned into \(2^i\) subsets; each subset emerges from one of the \(2^i\) \(i^{th}\)-order nodes. Thus, each subset is an \(i^{th}\) order truncated tree.

It is easy to see that the first \(i\) rows of the truncated matrix enclosed by the solid lines in Fig. 2-9, are truncated tail sequences of \(g\). The next \(\nu\) rows below the dashed line form the matrix \(G\) which generates the initial truncated tree. Each of the \(i^{th}\) order truncated trees is formed by adding a particular linear combination of the first \(i\) rows of the truncated matrix to each of the paths generated by the last \(\nu\) rows of the truncated matrix. This particular linear combination of the first \(i\) rows depends on the path consisting of the previous \(i\) branches leading to the truncated tree. If the path corresponds to the message sequence \(m_1m_2\ldots m_i\), then the particular linear combination of the first \(i\) rows being used is

\[
\tau = m_1[T^{i+\nu}g] \oplus m_2[T^{i-\nu}g] \oplus \ldots m_i[T^{i-2\nu}g] \\
= \sum_{j = 1}^{i} m_j [T^{i-j+2\nu}g] 
\]  

(2-14)

Then, we have the following theorem.
Theorem 2-2-1: The truncated trees emerging from two nodes of the same order, greater than or equal to $\nu$, are exactly alike if the paths leading to these two nodes correspond to the same message digits in the last $\nu-1$ places.

Proof: Let the two $i^{th}$ order truncated trees be $S(i)$ and $S'(i)$ and the two paths leading to the starting nodes of $S(i)$ and $S'(i)$ be $\bar{p}$ and $\bar{p}'$ respectively.

Let the two message sequences corresponding to $\bar{p}$ and $\bar{p}'$ be $(m_1 m_2 m_3 \ldots m_i)$ and $(m_1' m_2' \ldots m_i')$.

The sets of paths in $S(i)$ are formed by adding

$$\tilde{\tau} = m_1 [T^{i+1}_g] \oplus m_2 [T^{i+1}_g] \oplus \ldots \oplus m_i [T^2_g]$$

to each of the truncated paths of the initial truncated tree $S(0)$.

The set of paths in $S'(i)$ are formed by adding

$$\tilde{\tau}' = m_1' [T^i_g] \oplus m_2' [T^i_g] \oplus \ldots \oplus m_i' [T^2_g]$$

to each of the truncated paths of the initial truncated tree $S(0)$.

Since

$$[T^i_g] = [0][0] \ldots [0]$$

for $i > \nu$

Thus

$$\tilde{\tau} = m_{i-\nu+2} [T^i_g] \oplus m_{i-\nu+3} [T^{i-1}_g] \oplus \ldots \oplus m_i [T^2_g]$$

and

$$\tilde{\tau}' = m_{i-\nu+2} [T^i_g] \oplus m_{i-\nu+3} [T^{i-1}_g] \oplus \ldots \oplus m_i' [T^2_g]$$

By assumption

$$m_{i-\nu+2} = m_{i-\nu+2} \ldots m_i = m_i$$

Therefore, $\tilde{\tau} = \tilde{\tau}'$

and $S(i)$ and $S'(i)$ are alike. Q. E. D.
2-3. Binary Tree Codes with Transmission Rate $R_t = \frac{k_o}{n_o}$

Consider a set of $k_o$ infinite sequences of binary digits

\[ g = (g_1g_2g_3 \ldots g_{n_t} 000 \ldots) \]
\[ h = (h_1h_2h_3 \ldots h_{n_t} 000 \ldots) \]
\[ \vdots \]
\[ p = (p_1p_2p_3 \ldots p_{n_t} 000 \ldots) \]  

(2-15)

In each of these sequences, only the first $n_t = n_o$ digits may differ from zero. We also require $g_1 = h_2 = \cdots = p_{k_o} = 1$. All other digits among the first $k_o$ digits in each sequence are zero. It is obvious that this set of $k_o$ infinite sequences are linearly independent over the binary field.

As in Sec. 2-1, $D^k g, D^k h, \ldots, D^k p$ are the shifts of $g, h, \ldots, p$ to the right by $k n_o$ places, with $k = 0, 1, 2, \ldots$. Let us use

\[ \{g, h, \ldots, p, D^k g, D^k h, \ldots, D^k p, \ldots D^k g, D^k h, \ldots, D^k p, \ldots, \} \]  

(2-16)

as a base or the matrix $G_\infty$ shown in Fig. 2-10 as a generator matrix. It will generate a vector space $U_\infty$ of infinite dimension over the binary field. Each element can be expressed as

\[ \bar{u} = \bigoplus_{i = 0}^{\infty} m_{k_o+1} D^i g \oplus \bigoplus_{i = 0}^{\infty} m_{k_o+2} D^i h \oplus \cdots \oplus \bigoplus_{i = 0}^{\infty} m_{(i+1)k_o} D^i p \]  

(2-17)

where $M_t \in GF(2)$.

By the same argument as in Sec. 2-1, it is easy to see that this vector space also has an infinite tree structure with nodes spaced $n_o$ digits apart and $2^{k_o}$ branches stemming from each node. Each branch corresponds to a particular block of $k_o$ message digits.

Example: Let $k_o = 2$, $n_o = 3$, $n_t = 9$ and

\[ g = (101 001 010 000 \ldots) \]
\[ h = (011 010 001 000 \ldots) \]

Then the generator matrix is shown in Fig. 2-11 and the first initial truncated tree is shown in Fig. 2-12.
When the infinite tree with \( R_t = k_o/n_o \) is used as a code, the encoding operation will be the same as stated in Sec. 2-1. The encoder traces a path through the tree under the instructions given by the message sequence. Each time the instruction is given by a block of \( k_o \) message digits. There are 2 \( k_o \) branches emerging from each node of the tree. Thus, at each node the block of \( k_o \) message digits will instruct the encoder to trace a particular branch. The general encoder is shown in Fig. 2-13. Each unit time, \( k_o \) message digits enter the encoder input, and \( n_o \) channel input digits leave the encoder. The encoder circuit consists of \( k_o \) shift registers of \( v = \frac{n_t}{n_o} \) stages and \( n_o \) modulo-2 adders. The connections between each of the \( k_o \) shift registers and the \( n_o \) modulo-2 adders depend on one of the \( k_o \) generators. The connection instructions are the same as in Section 2-1. The general circuit is shown in Fig. 2-14 and the circuit for the last example is shown in Fig. 2-15.
Now consider the set of truncated sequences of \( \tilde{g}, \tilde{h}, \ldots, \tilde{p} \) and their shifts.

\[
[\tilde{g}] = (g_1g_2 \ldots \ldots g_{n_t})
\]
\[
[\tilde{h}] = (h_1h_2h_3 \ldots \ldots h_{n_t})
\]
\[
\vdots
\]
\[
[\tilde{p}] = (p_1p_2p_3 \ldots \ldots p_{n_t})
\]

\[
[D\tilde{g}] = (0 \ldots 0 \tilde{g}_1 \tilde{g}_2 \ldots \tilde{g}_{n_t-n_0})
\]
\[
[D\tilde{h}] = (0 \ldots 0 \tilde{h}_1 \tilde{h}_2 \ldots \tilde{h}_{n_t-n_0})
\]
\[
[D\tilde{p}] = (0 \ldots 0 \tilde{p}_1 \tilde{p}_2 \ldots \tilde{p}_{n_t-n_0})
\]

\[
[D^{\nu-1}\tilde{g}] = (00 \ldots 0 \tilde{g}_1 \tilde{g}_2 \ldots \tilde{g}_{n_0})
\]

\[
[D^{\nu-1}\tilde{h}] = (00 \ldots 0 \tilde{h}_1 \tilde{h}_2 \ldots \tilde{h}_{n_0})
\]

\[
[D^{\nu-1}\tilde{p}] = (00 \ldots 0 \tilde{p}_1 \tilde{p}_2 \ldots \tilde{p}_{n_0})
\]

The vector space generated by this set or the generator matrix \([G]\) shown in Fig. 2-16 has a finite dimension of \( k_0\nu \) and has a finite tree structure. Each path of the tree is of the form

\[
[\tilde{u}] = \bigoplus_{i=0}^{\nu-1} m_{ik_{i+1}} [D^{i-1}\tilde{g}] \bigoplus_{i=0}^{\nu-1} m_{ik_{i+2}} [D^{i-1}\tilde{h}] \oplus \ldots
\]

\[
[\tilde{u}] = \bigoplus_{i=0}^{\nu-1} m_{i(k_{i+1})} [D^{i-1}\tilde{p}]
\]
which is easily shown to be the truncated $\bar{u}$ of Eq. 2-17. Therefore, this finite tree is the initial truncated tree of the infinite tree generated by the matrix $G_d$ shown in Fig. 2-10.

As in Section 2-2, we may also call a finite tree having paths $\nu = n_t/n_o$ branches long, starting from any node of the infinite tree (with $R_t = k_o/n_o$) a truncated tree. The truncated tree emerging from any $i$th-order node will be called an $i$th-order truncated tree. Since there are $2^{i k_o}$ $i$th-order nodes, there will be $2^{i k_o}$ $i$th-order truncated trees. Each of these $2^{i k_o}$ truncated trees depends on the path of $i$ branches preceding it. By the same argument as Section 2-2, we can generalize Theorem 2-1 for $R_t = k_o/n_o$.

**Theorem 2-3-1** The truncated trees ($R_t = k_o/n_o$) emerging from two nodes of the same order, greater than or equal to $\nu$, are exactly alike if the paths leading to these two nodes correspond to the same message digits in last $(\nu-1)k_o$ places.
2-4. Canonical Form

First, consider the simple case with $R_t = 1/n_0$. The generator sequence is

$$[\bar{g}] = (g_1 g_2 \cdots g_{n_0} g_{n_0+1} g_{n_0+2} \cdots g_{2n_0} \cdots g_{(\nu-1)n_0+1} \cdots g_{\nu n_0})$$

where $\nu = n_t/n_0$ (2-20)

If

$$\begin{cases} g_{in_0+1} = 0 \\ g_1 = 1 \end{cases} \text{ for } 1 \leq i \leq \nu - 1$$

(2-21)

then we have

$$[\bar{g}] = (1 \ g_2 g_3 \cdots g_{n_0} 0 \ g_{n_0+2} \cdots g_{2n_0} 0 \ g_{2n_0+2} \cdots g_{3n_0} \cdots 0 \ g_{(\nu-1)n_0+2} \cdots g_{\nu n_0})$$

(2-22)

The tree generated by this generator sequence has the property that the first digit on each branch of the tree is the corresponding message digit. We then say this generator sequence is in canonical form.

For $R_t = k_0/n_0$, there are $k_0$ generator sequences

$$[\bar{g}] = (g_1 g_2 \cdots g_{n_0} g_{n_0+1} g_{n_0+2} \cdots g_{2n_0} \cdots g_{(\nu-1)n_0+1} \cdots g_{\nu n_0})$$

$$[\bar{h}] = (h_1 h_2 \cdots h_{n_0} h_{n_0+1} h_{n_0+2} \cdots h_{2n_0} \cdots h_{(\nu-1)n_0+1} \cdots h_{\nu n_0})$$

$$[\bar{p}] = (p_1 p_2 \cdots p_{n_0} p_{n_0+1} p_{n_0+2} \cdots p_{2n_0} \cdots p_{(\nu-1)n_0+1} \cdots p_{\nu n_0})$$

(2-23)

If

$$\begin{cases} g_1 = 1, & h_1 = \cdots = p_1 = 0 \\ h_2 = 1, & g_2 = \cdots = p_2 = 0 \end{cases} \text{ for } 1 \leq i \leq \nu - 1$$

(2-24)

and

$$\begin{cases} g_{in_0+1} = \cdots = g_{in_0+k_0} = 0 \\ h_{in_0+1} = \cdots = h_{in_0+k_0} = 0 \end{cases} \text{ for } 1 \leq i \leq \nu - 1$$

(2-25)

then the set of $k_0$ generator sequence have the form:
\[
\begin{align*}
\mathbf{g} &= 100\ldots0 \ g_{k_0+1} \ g_{k_0+2} \ldots g_{n_0} \ 00\ldots0 \ g_{n_0+k_0+k} \ldots g_{2n_0} \ldots \\
\mathbf{h} &= 010\ldots0 \ h_{k_0+1} \ h_{k_0+2} \ldots h_{n_0} \ 00\ldots0 \ h_{n_0+k_0+1} \ldots h_{2n_0} \ldots \\
\mathbf{p} &= 000\ldots1 \ p_{k_0+1} \ p_{k_0+2} \ldots p_{n_0} \ 00\ldots0 \ p_{n_0+k_0+1} \ldots p_{2n_0} \ldots \\
\end{align*}
\]

This set is in canonical form. The tree generated by this set of generators will have the property that the first \( k_0 \) digits on each branch of the tree are the corresponding \( k_0 \) message digits.
2-5. **Some Properties of the Initial Truncated Tree Codes**

In this section, we shall examine the structure of the initial truncated trees more carefully and develop some interesting properties which will be useful in decoding. In Secs. 2-2 and 2-3, we have shown that any order truncated tree is formed by adding a particular sequence of \( n_t \) digits to every path of the initial truncated tree. The addition of a particular sequence to every element of the initial truncated tree will not destroy the properties of the initial truncated tree. This will be shown in the course of developing the properties. Therefore, we shall be concerned only with the code consisting of the words in the initial truncated tree.

For convenience, we shall call the **initial truncated tree code** the **ITT code** and denote it with \( S \).

Each path of \( S \) is a sequence of \( n_t \) binary digits which will be called a **codeword**. If the transmission rate is \( R_t = k_o / n_o \), there will be \( 2^{n_t R_t} \) paths in \( S \) (or \( 2^k \) paths, where \( k = n_t / n_o \)).

Let us use \( n \) to indicate the **digit position** (or **component position**)
in a codeword (or path), where \( 1 \leq n \leq n_t \).

**Definition:** At any \( n \leq n_t \), if all the codewords in \( S \) contain only zero digits at that position, then we say the ITT code has a **null position** at \( n \).

Consider the set of \( k_o \) generator sequences

\[
\left[ \mathbf{g} \right] = (g_{1o} g_{2o} \cdots g_{no}, g_{no+1} g_{2no}, \ldots, g_{(v-1)n_o+1} g_{vn_o})
\]

\[
\left[ \mathbf{h} \right] = (h_{1o} h_{2o} \cdots h_{no}, h_{no+1} h_{2no}, \ldots, h_{(v-1)n_o+1} h_{vn_o})
\]

\[
\left[ \mathbf{p} \right] = (p_{1o} p_{2o} \cdots p_{no}, p_{no+1} p_{2no}, \ldots, p_{(v-1)n_o+1} p_{vn_o})
\]
Lemma 2-5-1: If at least one of the digits \( \{ g_i, h_i, \ldots, p_i \} \) is not zero, for each \( 1 \leq i \leq n_0 \), then there will be no null position in the ITT code.

Proof: As we know, the generator matrix of the ITT code is formed by shifting the set of \( k_0 \) generator sequences to the right by \( n_0 \) digits each time. By the assumption, at least one of the digits \( g_i, h_i, \ldots, p_i \) is not zero for \( 1 \leq i \leq n_0 \); therefore, each column of the generator matrix contains at least one non-zero digit. In other words, at any position \( n \), there is at least one row which contains a non-zero digit at \( n \). Since the codewords of the tree code are the possible linear combinations of the rows of the generator matrix, there must be at least one codeword which contains a non-zero digit at \( n \). This proves the theorem. Q.E.D.

If an initial truncated tree code does not have any null position, then it will have the following property.

Lemma 2-5-2: If the tree code does not have a null position at \( n \), then the number of codewords which have digit "1" at \( n \) is equal to the number of codewords which have digit "0" at \( n \).

Proof: As we know, the tree code is a group. Thus, adding any codeword that has digit "1" at \( n \) to all of the codewords would reproduce the original set of codewords in different order. In this way, the codewords which have digit "1" at \( n \) will be changed into codewords which have digit "0" at \( n \). Since the code is still the same, therefore, the number of codewords which contain digit "1" at \( n \) must be equal to the number of codewords which contain digit "0" at \( n \). Q.E.D.
A direct result of Lemma 2-5-2 is the following

**Lemma 2-5-3:** If there is no null position in the initial truncated tree code, then the total number of digits "1" is equal to the total number of digits "0" and is equal to $\frac{1}{2} \ln M$ (where $M$ is the number of codewords in the tree code).

Let $\tilde{u}$ be the code-word transmitted over a noisy channel and $\tilde{v}$ be the received sequence at the channel output. Then we define the set of **test error sequences** as the set of sequences obtained by adding the received sequence to each of the codewords of the tree code.

If the actual error (noise) sequence is $\tilde{e}_o$ then

$$\tilde{v} = \tilde{u} \oplus \tilde{e}_o \quad (2-28)$$

Let $\{\tilde{u}_i : 1 \leq i \leq M\}$ be the set of codewords; the test error sequence is

$$\tilde{e}_i = \tilde{v} \oplus \tilde{u}_i \quad l \leq i \leq M \quad (2-29)$$

and the set of test error sequences will be

$$\{\tilde{e}_i : 1 \leq i \leq M\}.$$

**Theorem 2-5-1:** The total number of digits "1" in the set of test error sequences is equal to the total number of digits "0" in that set.

Proof: $\tilde{e}_i = \tilde{u}_i \oplus \tilde{v}$

but $\tilde{v} = \tilde{u} \oplus \tilde{e}_o$

thus $\tilde{e}_i = (\tilde{u}_i \oplus \tilde{u}) \oplus \tilde{e}_o \quad l \leq i \leq M$

Since the tree code $S$ is a group, $\tilde{u}$ in $S$ and $\tilde{u}_i$ in $S$ imply $\tilde{u}_i \oplus \tilde{u}$ also in $S$. As $i$ runs through 1 to $M$, the set $\{\tilde{e}_i : 1 \leq i \leq M\}$ is just the set obtained by adding $\tilde{e}_o$ to all the codewords in $S$. By Lemma 2-5-2, the number of digits "1" of the tree code at any $1 \leq n \leq n_t$
is equal to the number of digits "0" and \( n \); thus an error digit "1"
at \( n \) will not change the equality. And adding \( \bar{c}_o \) to the set\
\[ \{ \bar{u}_i : 1 \leq i \leq M \} \] will not change the equality of total number of "1"
in the code and the total number of "0" in the code. Q.E.D.

For transmission rate \( R_t = k_o/n_o \), there are \( 2^{k_o} \) branches, each \( n_o \) digits long, stemming from each node of the initial truncated tree
code. We thus can partition this truncated tree code into \( 2^{k_o} \) disjoint subsets. Each of these subsets \( S_j \) is a subtree, of length \( v-1 \) branches, stemming from one of the first-order nodes. Hence,

\[
S_j = \{ \bar{u} ; \bar{u} = m_1^j [\bar{s}] \oplus m_2^j [\bar{h}] \oplus \ldots \oplus m_k^j [\bar{p}] \}
\]

\[
= \bigoplus_{i=1}^{v-1} \frac{m_{i+1}^j}{k_o}[D_i^o] \oplus \ldots \oplus \bigoplus_{i=1}^{v-1} \frac{m_{i+1}^j}{k_o}[D_i^p]
\]

for \( 0 \leq j \leq 2^{k_o}-1 \) \hspace{1cm} (2-30)

where \( m_1^j, m_2^j, \ldots, m_k^j \) are fixed for each \( j \), and \( m_t \) \((t > k_o)\) varies
over the binary field. Thus \( S_j \) contains \( 2^{(v-1)k_o} \) elements which have
the same initial prefix of \( n_o \) digits. The subset \( S_0 \) which corresponds
to \( m_1^0 = m_2^0 = \ldots = m_k^0 = 0 \) contains the zero path of the tree \( S \). For
convenience, we will call \( S_0 \) the \textit{zero subset} of \( S \).

Let us define the distance between any two distinct subsets \( S_i\)
and \( S_j \) as

\[
\Delta (S_i, S_j) = \min d(\bar{u}, \bar{v})
\]

\[
\bar{u} \in S_i
\]

\[
\bar{v} \in S_j
\]

\[
i \neq j
\]

(2-31)

where \( d(\bar{u}, \bar{v}) \) is the \textit{Hamming distance} between two paths \( \bar{u} \) and \( \bar{v} \)
(or the number of places where \( \bar{u} \) and \( \bar{v} \) differ).
We also define
\[ D_{\min} = \min_{i,j} \Delta(S_i, S_j) \]
\[ i,j \]
\[ i \neq j \] (2-32)

as the **minimum distance** of a tree code \( S \).

For convenience of the proof of the following theorems, we assume that the set of first \( k_0 \) digits of the \( k_0 \) generators are as follows:
\[ s_1 = 1, \quad s_2 = s_3 = \ldots = s_{k_0} = 0 \]
\[ h_2 = 1, \quad h_1 = h_3 = \ldots = h_{k_0} = 0 \]
\[ p_{k_0} = 1, \quad p_1 = p_2 = \ldots = p_{k_0-1} = 0 \] (2-33)

By this assumption, it is obvious that the \( k_0 \) generators are linearly independent over the binary field. And also the prefixes of the \( 2^{k_0} \) subsets defined in Eq. (2-30) are distinct.

**Theorem 2-5-2**: If \( S_i \) and \( S_j \) are two distinct subsets of tree code \( S \) and if \( \bar{u} \) and \( \bar{u}' \) are any two paths in \( S_i \), then the set of Hamming distances between \( \bar{u} \) and every path \( \bar{v} \) of the subset \( S_j \) is exactly the same as the set of Hamming distances between \( \bar{u}' \) and every path \( \bar{v} \) of the subset \( S_j \).

**Proof**: Let the prefix of the paths of \( S_i \) be
\[ 1, 2, \ldots, u_{n_0} \]
and the prefix of the paths of \( S_j \) be
\[ 1, 2, \ldots, v_{n_0} \]

Since \( \bar{u} \) and \( \bar{u}' \) are in the same subset \( S_i \), they must have the same prefix \( 1, 2, \ldots, u_{n_0} \).

If we add \( \bar{u} \) to every element \( \bar{v} \) in \( S_j \), we obtain the subset \( S_k \).
of elements in the tree code with the prefix \( u^1 \oplus v^1, u^2 \oplus v^2, \ldots u^n \oplus v^n \). This is due to the group property of tree code. Now, we add \( \tilde{u}' \) to every element \( \tilde{v} \) in \( S_j \), we obtain another subset \( S'_k \) of elements in the tree code with prefix \( u^1 \oplus v^1, u^2 \oplus v^2, \ldots u^n \oplus v^n \).

Thus, both \( S_k \) and \( S'_k \) have the same prefix and have \( 2^{(v-1)k_0} \) elements in the tree code with the same prefix \( u^1 \oplus v^1, u^2 \oplus v^2, \ldots u^n \oplus v^n \). \( S_k \) and \( S'_k \) must be the same set of elements. The set of weights of the elements in \( S_k \) is the set of Hamming distances between \( \tilde{u} \) and every element \( \tilde{v} \) of \( S_j \), and the set of weights of the elements in \( S'_k \) is the set of Hamming distances between \( \tilde{u}' \) and every element \( \tilde{v} \) of \( S_j \). Therefore, these two sets of distances are the same.

By theorem 2-5-2, we then have

\[
\Delta (S'_k, S_j) = \min \{ d(\tilde{u}', \tilde{v}) \mid \tilde{v} \in S_j \} \tag{2-34}
\]

where \( \tilde{u} \) is any element of \( S_i \).

**Lemma 2-5-4:** If \( \tilde{u} \) is any path in \( S_i \) and \( \tilde{v} \) is any path in \( S_j \), then the set of Hamming distances between \( \tilde{u} \) and every path in \( S_j \) is the same as the set of Hamming distance between \( \tilde{v} \) and every path in \( S_i \).

**Proof:** This lemma can be proved by using the same argument as in theorem 2-5-2.

Let \( S_i \) be any subset of the tree code \( S \). We denote by \( S_i^c \) its complement. \( S_i^c \) contains all the elements of \( S \) except those elements in \( S_i \).

**Lemma 2-5-5:** If \( \tilde{u} \) and \( \tilde{u}' \) are any two paths in \( S_i \), then the set of Hamming distances between \( \tilde{u} \) and every element in the set \( S_i^c \) is exactly the same as the set of Hamming distances between \( \tilde{u}' \) and every element in the set \( S_i^c \).
Proof: This is a direct result of Theorem 2-5-2.

**Lemma 2-5-6:** Let \( S \) and \( S_j \) be two distinct subsets of \( S \), and \( \tilde{u} \in S_i, \tilde{v} \in S_j \). Then the set of Hamming distances between \( \tilde{u} \) and every element in \( S_i^c \) is the same as the set of Hamming distances between \( \tilde{v} \) and every element in \( S_j^c \).

Proof: Consider the set \( S^* \) of elements obtained by adding \( \tilde{u} \) to every element in \( S_i^c \). Because of the fact that all \( 2^k \) initial prefixes of the tree code \( S \) are distinct and the modulo two addition of any two distinct prefixes will not be the prefix of \( n \) zero digits, \( S^* \) is thus the set \( S_0^c \) which contains all the elements of the tree code \( S \) except those elements in \( S_0 \), where \( S_0 \) contains the elements in \( S \) which have the prefix of \( n \) zero digits.

By the same argument, we can also show that the set \( S^{**} \) of elements formed by adding \( \tilde{v} \) to every element of \( S_j^c \) is the set \( S_0^c \).

The set of weights of the elements of \( S_0^c \) is the Hamming distances between \( \tilde{u} \) and every element of \( S_i^c \), also it is the set of Hamming distances between \( \tilde{v} \) and every element of \( S_j^c \). This proves the Lemma.
By Theorem 2-5-2, Lemma 2-5-4, Lemma 2-5-5, and Lemma 2-5-6, we obtain

\[
D_{\min} = \min_{i,j} \Delta(S_i, S_j)
\]
\[
= \min_{i \neq j} \Delta(S_i, S_j)
\]
\[
= \min_{i \neq j} \Delta(S_i, S_j) \quad \text{for any } i
\]
\[
= \min_{i \neq j} \Delta(S_i, S_j) \quad \text{for any } j
\]
\[
= \min_{i \neq j} \min_{\bar{v} \in S_j} d(\bar{u}, \bar{v}) \quad \text{for any } i
\]
\[
= \min_{i \neq j} \min_{\bar{u} \in S_i} d(\bar{u}, \bar{v}) \quad \text{for any } j
\]

(2-35)

**Definition:** The path with the smallest weight which is not in the subset \( S_o \) is called the **minimum weight path**.

**Theorem 2-5-3:** The minimum distance of the tree is equal to the weight of the minimum weight path which is not in \( S_o \).

**Proof:**

From Eq. 2-34 we have

\[
D_{\min} = \min_{i \neq j} \min_{\bar{v} \in S_j} d(\bar{u}, \bar{v})
\]

for any \( i \) and any \( \bar{u} \in S_i \).

If we let \( i = 0 \) and \( \bar{u} = 0 \) the zero path in \( S_o \), then

\[
D_{\min} = \min_{j \neq 0} \min_{\bar{v} \in S_j} d(\bar{0}, \bar{v})
\]

(2-36)

But \( d(\bar{0}, \bar{v}) \) is the Hamming distance between the path \( \bar{v} \) and the zero path \( \bar{0} \). Therefore, \( d(\bar{0}, \bar{v}) \) is just equal to the weight \( w(\bar{v}) \) of \( v \).
Thus

$$D = \min_{j} \min_{\forall \in S_j} \min_{j \neq 0} w(\bar{v}) \tag{2-37}$$

This proves the theorem.

**Definition:** Two tree codes are said to be completely distinct if their intersection contains only the zero path.

**Theorem 2-5-4:** If any one of the $k_o$ generators, say $\bar{g}$, is replaced by a path in any of those subsets (defined in Eq. 2-30) which are formed by including $[\bar{g}]$ (i.e., $m^j_1 = 1$ for all $j$), the new set of $k_o$ generators will generate the same tree $S$.

**Proof:** As we know, the path can be written as

$$[\bar{u}] = [\bar{g}] = \bigoplus_{i=1}^{v-1} m_{i k_o + 1} [D^i \bar{g}] \bigoplus \bigcap_{i=0}^{v-1} m_{i k_o + 2} [D^i \bar{h}]$$

$$\bigoplus \ldots \bigoplus \bigcap_{i=0}^{v-1} m_{(i+1)k_o} [D^- i \bar{p}] \tag{2-38}$$

$$= [\bar{g}] \bigoplus [\bar{f}]$$

where

$$[\bar{f}] = \bigoplus_{i=1}^{v-1} m_{i k_o + 1} [D^i \bar{g}] \bigoplus \bigcap_{i=0}^{v-1} m_{i k_o + 2} [D^i \bar{h}]$$

$$\bigoplus \ldots \bigoplus \bigcap_{i=0}^{v-1} m_{(i+1)k_o} [D^- i \bar{p}] \tag{2-39}$$

which is a linear combination of the set

$$[[h], \ldots, [\bar{p}], [D^\bar{g}], [D^\bar{h}], \ldots, [D^\bar{p}], \ldots]$$

When $[\bar{g}]$ is replaced by $[\bar{u}]$, the $k_o$ generators will be $[\bar{u}], [\bar{h}], \ldots, [\bar{p}]$ and the new generator matrix will be
\[
G' = \begin{pmatrix}
[\tilde{u}] \\
[\tilde{h}] \\
\vdots \\
[\tilde{p}] \\
[D\tilde{u}] \\
[D\tilde{h}] \\
\vdots \\
[D\tilde{p}]
\end{pmatrix} = \begin{pmatrix}
[\tilde{g}] \oplus [\tilde{f}] \\
[\tilde{h}] \\
\vdots \\
[\tilde{p}]
\end{pmatrix}
\]

It is obvious that \([G']\) is obtained from matrix \(G\) in Fig. 2-16 by row operations. Therefore, the two spaces generated by \(G\) and \(G'\) are the same. The theorem is thus proved.

**Lemma 2-5-7:** The maximum number of initial truncated tree codes that contain a particular \(n_t\)-tuple is

\[
Q = 2^{(n_t-k^2)(k_o-1)} n_{R-1}^t 2^{t-1}
\]

Proof: By the assumption made in Eq. 2-33, each of the \(k_o\) generators can have \(2^{n_t-k^2}\) possible choices. Now if we fix one of them, say \([\tilde{g}]\), there will be \((2^{n_t-k^2})_{o-1}\) possible combinations of \(k_o\) generators which contain the specified \([\tilde{g}]\). And by Theorem 2-5-4, each of these combinations with \([\tilde{g}]\) replaced by a path in any of those subsets which are formed by including \([\tilde{g}]\) will generate the same tree. There are \(2^{n_t-k^2-1}\) of those subsets. Each subset contains \(\frac{1}{2^{k_o^2}} n_{R-1}^t\) paths which can be used to replace \([\tilde{g}]\) without changing the tree code \(S\).
Thus, the maximum number of tree codes which contain \([g]\) is

\[
Q = (2^{n_t-k_0})k_0-1 2^{n_tR_t-1}
= 2^{(n_t-k_0)(k_0-1)} 2^{n_tR_t-1}
\]

(2-40)

**Theorem 2-5-5:** With transmission \(R_t = k_0/n_o\), and constraint length \(n_t\), there exists at least one convolutional tree code for which the minimum distance \(D_{\text{min}}\) exceeds \(D\), where \(D\) is the largest integer satisfying the inequality

\[
\sum_{r=1}^{k_0} \sum_{j=0}^{k_0-n_t+r} C_r C_j < \frac{1}{2^{k_0-1}} 2^{n_t}(1-R_t)
\]

(2-41)

Proof: By the assumption made in Eq. 2-33, any path which is not in the zero subset \(S_o\) will have at least one non-zero digit in its first \(k_0\) positions. Thus, the total number of \(n_t\)-tuples which have weight less than or equal to \(D\) and have at least one non-zero digit in the first \(k_0\) positions is

\[
N_D = \sum_{r=1}^{k_0} \sum_{j=0}^{k_0-n_t+r} C_r C_j
\]

(2-42)

For each of these \(N_D\) sequences, there are at most

\[
Q = 2^{(n_t-k_0)(k_0-1)} 2^{n_tR_t-1}
\]

codes which contain that sequence. Therefore, there are at most \(N_Q\) codes with minimum distance less than or equal to \(D\).

By the assumption made in Eq. 2-33, there are

\[
N = (2^{n_t-k_0})k_0
\]

possible choices for the set of \(k_0\) generators. In other words, there are \(N\) possible tree codes.
If

\[ N_D g < N, \]  \hspace{1cm} (2-43)

there must be at least one code among the \( N \) for which the minimum distance \( D_{\text{min}} \) is greater than \( D \).

Eq. 4-43 may be rearranged to give

\[ N_D g < \frac{N}{Q} \]

which means

\[ \sum_{r=1}^{k_0} \sum_{j=0}^{n_t - k_0} C_r C_j < \frac{1}{2^{k_0 - 1}} \]

\[ n_t (1 - R_t) \]  \hspace{1cm} (2-44)

The bound of the last theorem is called the Gilbert bound for tree codes. This bound was first shown to hold for binary tree codes with \( R_t = 1/n_o \) by Wozencraft [2]. The minimum distance of a tree code is equivalent to the minimum distance of a block code. When a tree code is used for coding, the larger the minimum distance it has, the greater the error correction capability it possesses. Theorem 2-5-5 just shows the existence of a tree code which has minimum distance satisfying the Gilbert bound; it does not tell us how to construct this tree code. Thus, a systematic way is needed to construct good tree codes (or good sets of generator sequences). By good tree codes, we mean the tree codes with minimum distance satisfying the Gilbert bound.

Besides the Gilbert bound, we can also obtain an upper bound on the minimum distance of tree codes.

**Theorem 2-5-6.** The minimum distance \( D_{\text{min}} \) of a binary tree code satisfies the following inequality

\[ D_{\text{min}} \leq \frac{1}{2} \left( n_t + \frac{n_o}{2^{k_0 - 1}} \right) \]  \hspace{1cm} (2-45)
Proof: By Lemma 2-5-3, the total number of digits "1" in an initial truncated tree code is \( \frac{1}{2} n_t M \), where \( M = 2^{n_t R_t} \). And we know the subset \( S_o \) with prefix of \( n_o \) zero digits is a subgroup of tree code \( S \). By using Lemma 2-5-2, we can show that the number of digits "1" in \( S_o \) is \( \frac{1}{2} (n_t - n_o) \frac{M}{2^{k_o}} \). Thus the total number of digits "1" in the subset \( S_o^c \) is

\[
\frac{1}{2} n_t M - \frac{1}{2} (n_t - n_o) \frac{M}{2^{k_o}}
\]  

(2-46)

where \( S_o^c \) is the subset which contains all the codewords in tree code \( S \) except those in the subset \( S_o \).

The total number of codewords in \( S_o^c \) is \( M - \frac{1}{2^{k_o}} M \). Thus the average weight of a codeword in \( S_o^c \) is

\[
D_{\text{ave}} = \frac{\frac{1}{2} n_t M - \frac{1}{2} (n_t - n_o) \frac{M}{2^{k_o}}}{M - \frac{1}{2^{k_o}} M} = \frac{1}{2} (n_t + \frac{n_o}{2^{k_o - 1}})
\]  

(2-47)

Since the weight of minimum weight path in \( S_o^c \) is less than the average weight, we thus prove the theorem.

According to Lemma 2-5-1, a tree code with transmission rate \( R_t = 1/n_o \) will have no dummy position if the first \( n_o \) digit of the generator sequence \( \tilde{g} \) are all "1". For this case, the two branches emerging from any node of the tree will be complements of each other (i.e., the modulo-2 sum of these two branches will be all digit "1").

The tree has two subsets, one denoted with \( S_o \) corresponds to \( m_1 = 0 \), the other denoted with \( S_1 \) corresponds to \( m_1 = 1 \). Because of the complement property, the average weight of any branch beside the first order branch in \( S_1 \) is \( \frac{n_o}{2} \). Since each path in the tree code has \( \frac{n_t}{n_o} \) branches, so the average weight of a path in \( S_1 \) is
\[
\text{d}_{\text{ave}} = n_o + \left( \frac{n_t}{n_o} - 1 \right) \frac{n_o}{2} \\
= \frac{1}{2} \left( n_t + n_o \right) 
\]  
(2-48)

Thus we have

**Theorem 2-5-7:** The minimum distance \( D_{\text{min}} \) of a tree code with \( R_t = 1/n_o \) satisfies

\[
D_{\text{min}} \leq \frac{1}{2} \left( n_t + n_o \right) 
\]  
(2-49)

If \( n_o \) is odd, a tighter bound on \( D_{\text{min}} \) can be obtained. Besides the first-order branch, one of the two branches stemming from any node in the tree \( S \) will have weight smaller than the other by at least one (due to the complementary property). If at any node we select the branch with smaller weight, then there exists a path in \( S_t^{-1} \) with weight less than

\[
n_o + \left( \frac{n_t}{n_o} - 1 \right) \cdot \frac{n-o-1}{2} \\
= \frac{1}{2} \left[ n_t (1-R_t) + n_o + 1 \right] 
\]  
(2-50)

Thus the weight of the minimum path in \( S_t^{-1} \) will be less than

\[
D_{\text{min}} \leq \frac{1}{2} \left[ n_t (1-R_t) + n_o + 1 \right] 
\]  
(2-51)

for \( n_o \) is odd.
2-6. **Construction of Good Generator Sequences**

Of all the algebraic properties of a tree code which we developed in the last section, the minimum distance property is the most important one and will play an important role in the decoding. We shall see, in the sequential type decoding, the minimum distance of a tree code gives the error-correction capability. The larger the minimum distance the tree code has, the greater the error-correction capability it possesses. In all of the decoding schemes of sequential type proposed for the tree codes\(^1,3\) this minimum distance property has been either used implicitly or explicitly. No matter what decoding scheme is used, a decoding error is likely to be made once the number of transmission errors in the span \(n_t\) is greater than or equal to half the minimum distance. Because of its importance, we would like to construct generators which will generate a tree code with as large a minimum distance as possible.

In the last section, we have shown the existence of a tree code with minimum distance satisfying the Gilbert bound. Unfortunately, no procedure has been found to construct the set of \(k_o\) generators which will generate a tree with maximum minimum distance or even with minimum distance satisfying the Gilbert bound. Therefore, all we can do is to test all possible sets of \(k_o\) generators and select the set which gives the maximum minimum distance. For large \(n_t\) and \(R_t = k_o/n_o\) this one by one test will require tremendous labor and take a large amount of time. Some other effective procedures are thus needed.

Before we describe a searching procedure, it would be better for us to review the generator matrix and the structure of a tree.
For $R_t = k_o / n_o$, there are $k_o$ generators

\[
\begin{align*}
\tilde{g} &= (g_1 g_2 \ldots g_{n_o}, g_{n_o}+1 g_{n_o}+2 \ldots g_{2n_o}, \ldots, g_{(v-1)n_o+1} \ldots g_{vn_o}) \\
\tilde{h} &= (h_1 h_2 \ldots h_{n_o}, h_{n_o}+1 h_{n_o}+2 \ldots h_{2n_o}, \ldots, h_{(v-1)n_o+1} \ldots h_{vn_o}) \\
\vdots \\
\tilde{p} &= (p_1 p_2 \ldots p_{n_o}, p_{n_o}+1 p_{n_o}+2 \ldots p_{2n_o}, \ldots, p_{(v-1)n_o+1} \ldots p_{vn_o})
\end{align*}
\]

where $v = n_L / n_o$. \hfill (2-52)

We have defined the $i$th segment of a generator sequence, say $\tilde{g}$, as

\[
\tilde{g}_i = g(i-1)n_o+1 g(i-1)n_o+2 \ldots g_{in_o}
\] \hfill (2-53)

Each generator has $v$ segments. Then we have

\[
\begin{align*}
\tilde{g}_i &= g(1) g(2) g(3) \ldots g(v) \\
\tilde{h}_i &= h(1) h(2) h(3) \ldots h(v) \\
\vdots \\
\tilde{p}_i &= p(1) p(2) p(3) \ldots p(v).
\end{align*}
\hfill (2-54)

We will call \{$g(i), h(i) \ldots p(i)$\} the set of $i$th segments of the generators.

The generator matrix for the tree code is shown in Fig. 2-17. We have indicated that the $i$th column of the generator matrix generates the $i$th order branches of the tree.

Now, we are going to describe a reasonable procedure for constructing the set of $k_o$ good generator sequences. This procedure involves constructing the set of generators, segment by segment. We start by selecting the set of first segments (i.e., $g(1), h(1) \ldots p(1)$). Once we make a selection, we are going to select the set of second segments (i.e., $g(2) \ldots p(2)$). Having made the selection of the set of second segments, we are going to construct the set of third segments. We continue this step-by-step construction process until each of the $k_o$ generator sequences reaches the desired length of $v$ segments. The rules for constructing each set of generator segments is summarized as follows:

The selection of the set of $k$th segments must
(1) Maximize the minimum distance of the tree of \( k \) branches in length (or maximize the weight of the minimum-weight path in the subset \( S_o^c \)).

(2) Minimize the number of the minimum-weight paths in \( S_o^c \). In \( S_o^c \), there may be several paths with the same smallest weight. Keeping the number of these paths small not only reduces the difficulty of selecting the next set of segments, but also reduces the probability of error in the sequential type decoding scheme.

(3) Keep the weights of all paths in \( S_o^c \) as close to one another as possible. This is desirable since the minimum distance will be largest if all the paths in \( S_o^c \) have the same weight.

(4) Avoid the periodic appearance of the sets of generator segments.

Now we use the above algorithm to construct a tree code with \( R_t = 1/3 \).

There is only one generator sequence. Each segment of the generator sequence is 3 digits long. Therefore, each segment has \( 2^3 \) possible choices:

- 000 100
- 001 101
- 010 110
- 011 111

According to the above algorithm, the first segment of the generator sequence must be 111, because this choice gives the maximum minimum distance for the tree one branch long as shown in Fig. 2-18a.

When we extend the tree to 2 branches, it has the form as shown in Fig. 2-18b.
In order to complete the tree, we have to make a selection of the second segment of the generator sequence. Let us consider all the 8 possible choices. Since the first segment consists of all "1"'s, the two branches stemming from any node of the tree are complements of each other. The choice of any 3-tuple or its complement as the second segment will generate the same tree. Therefore, there are actually 4 possible choices for the second segment.

(1) The choice of 000 (or 111) will generate the tree as shown in Fig. 2-18c. The minimum distance is 3.

(2) The choice of 001 (or 110) will generate the tree as shown in Fig. 2-18d. The minimum distance is 4.

(3) The choice of 010 (or 101) will generate the tree as shown in Fig. 2-18e. The minimum distance is 4.

(4) The choice of 100 (or 011) will generate the tree as shown in Fig. 2-18f. The minimum distance is 4.

According to the rules, the first choice 000 (or 111) is a bad choice. The second, third and fourth choices are all equally good. They all give maximum minimum distance 4.

Let us use 001 as the second segment of the generator sequence. Now we are going to construct the third segment. The tree of three branches long is of the form as shown in Fig. 2-19a. In order to complete the tree, we have to select g(3). Now since the minimum weight path in S1 is 111 001, we would pick g(3) which will increase the weight of this path when it is extended by one branch. It is easy to see that g(3) = 000 is a bad choice, since it does not increase the weight of the minimum weight path. If g(3) = 001, then the tree will be that shown in Fig. 2-19b. The weight of minimum weight path has been increased by one. But there are two minimum weight paths in the tree indicated by circles.
If we let \( g(3) = 010 \), the tree will be that shown in Fig. 2-19c. This choice \( g(3) = 010 \) also increases the weight of the minimum weight path by one. But there is only one minimum weight path. According to the algorithm, the third choice \( g(3) = 010 \) is better than the other two choices. We can show \( g(3) = 100 \) is also a good choice for the third segment of the generator sequence.

The construction of the fourth segment \( g(4) \) is determined similarly: \( g(4) = 010 \) is a good choice which increases the minimum distance by one and still keeps the number of minimum weight paths at one.

It is not too hard to continue this step-by-step method of constructing the generator sequence \( \tilde{g} \) up to 12 segments by hand. The labor will be tedious for more than 12 segments. Computer searching is thus needed for large \( n_{R_1} \). The author was able to construct a good generator sequence \( \tilde{g} \) with \( R_1 = 1/3 \) up to 12 segments which gives a minimum distance \( D_{\text{min}} = 13 \). This generator sequence is
\[
\tilde{g} = 111 001 010 010 001 011 010 000 000 000 010 \ldots
\]
An extensive computer search for good generator sequences with large \( n_{R_1} \) is being carried by H. Lyne of Rice University.

By use of the above designing method, it is possible to build a tree code with a quick separation between \( S_o \) and \( S^c_o \) at the very beginning. This quick separation property is very desirable in the sequential type decoding to be discussed in the following chapters.
2-7. Non-binary Convolutional Tree Codes

In the previous sections, we have developed a fairly complete theory for binary convolutional tree codes. In the present section, we shall generalize the previous work to the non-binary case.

We assume that the channel input alphabet is a finite field \( \text{GF}(q) \) of \( q \) elements and the source alphabet is a subfield \( \text{F}_s \) of \( \text{GF}(q) \) which consists of \( r \) elements. Thus the message sequence is a string of letters from \( \text{F}_s \) and the channel input sequence is a string of letters from \( \text{GF}(q) \). Now we are concerned with the convolutional encoding of the message sequence into the channel input sequence.

Consider the general case that the transmission rate is \( R_t = k_o/n_o \) (i.e., a block of \( k_o \) message symbols is mapped into a block of \( n_o \) channel input symbols for each unit time). Then the set of \( k_o \) infinite generator sequences will be

\[
\hat{g} = (g_1 g_2 g_3 \ldots \ldots g_{n_t} 000 \ldots) \\
\hat{h} = (h_1 h_2 h_3 \ldots \ldots h_{n_t} 000 \ldots) \\
\vdots \\
\hat{p} = (p_1 p_2 p_3 \ldots \ldots p_{n_t} 000 \ldots)
\]

(2-52)

where the first \( n_t \) components of each infinite generator sequence are from the finite field \( \text{GF}(q) \) and the remaining components are the zero element of \( \text{GF}(q) \). Let us assume that

\[
g_1 \neq 0, g_2 = g_3 = \ldots = g_{k_o} = 0 \\
h_2 \neq 0, h_1 = h_3 = \ldots = h_{k_o} = 0 \\
\vdots \\
p_{k_o} \neq 0, p_1 = p_2 = \ldots = p_{k_o-1} = 0
\]

(2-53)

Then it is obvious that the set of \( k_o \) infinite generator sequences are linearly independent over the subfield \( \text{F}_s \) (or the field \( \text{GF}(q) \)).
As in Sec. 2-1, we define the shift operator $D$ by

$$D^i g = (00...0 \overset{i}{\underline{g}} 000...)$$

$$D^i h = (00...0 \overset{i}{\underline{h}} 000...)$$

$$\vdots$$

$$D^i p = (00...0 \overset{i}{\underline{p}} 000...)$$

where $i = 0, 1, 2, 3, \ldots$

It is obvious that $g, h, \ldots, p$ and their shifts form a linearly independent set (infinite). Let us use this set as a base or

$$G = \left\{ \begin{array}{c}
g \\
h \\
\vdots \\
p \\
Dg \\
Dh \\
\vdots \\
P \\ \vdots \\
\end{array} \right\}$$

as a generator matrix. Consider the set of linear combinations of the form

$$u = m_1 g + m_2 h + \ldots + m_{k_0} p + m_{k_0+1} Dg + m_{k_0+2} Dh + \ldots + m_{2k_0} Dp + \ldots$$

$$u = \bigoplus_{i=0}^{\infty} m_{i+k_0} D^i g + \bigoplus_{i=0}^{\infty} m_{i+k_0+2} D^i h + \ldots + \bigoplus_{i=0}^{\infty} m_{(i+1)k_0} D^i p$$

where $m_j \in F_s$ and the operations are carried out in the field $GF(q)$. This set will be a vector space (of infinite dimensions) over the subfield $F_s$. By the same argument as in Sec. 2-1, it is easy to show that this vector space can be arranged as an infinite tree with nodes spaced $n_0$ digits apart and with $k_0$ branches emerging from each node. Each branch of the tree corresponds to a particular block of
$k_0$ message digits from $F_s$ (see Fig. 2-18).

When this kind of vector space is used as a code in the coding of a discrete channel, the encoding operation will be equivalent to tracing a particular path in the tree under the instructions of the message sequence $m_1m_2m_3\cdots$. Each instruction is given by a block of $k_0$ message digits.

As in the previous sections, we are concerned with the initial truncated tree $S$ of $\nu = n_t/n_0$ branches of length. It is generated by the set of truncated sequences

$$[\vec{g}] = (g_1g_2\cdots\cdots g_{n_t})$$

$$[\vec{h}] = (h_1h_2\cdots\cdots h_{n_t})$$

$$\vdots$$

$$[\vec{p}] = (p_1p_2\cdots\cdots p_{n_t})$$

$$[D\vec{g}] = (00\ldots0 g_1g_2\cdots\cdots g_{n_t-n_0})$$

$$[D\vec{h}] = (00\ldots0 h_1h_2\cdots\cdots h_{n_t-n_0})$$

$$\vdots$$

$$[D\vec{p}] = (00\ldots0 p_1p_2\cdots\cdots p_{n_t-n_0})$$

$$\nu=1n_0$$

$$[D^{\nu-1}\vec{g}] = (00\ldots0 g_1g_2\cdots\cdots g_{n_0})$$

$$\vdots$$

$$[D^{\nu-1}\vec{p}] = (00\ldots0 p_1p_2\cdots\cdots p_{n_0})$$

Each path in $S$ will be of the form

$$[\vec{u}] = \bigcup_{i=0}^{m_{ik_0+1}} [D^{\nu-1}_i\vec{g}] + \bigcup_{i=0}^{m_{ik_0+2}} [D^{\nu-1}_i\vec{h}] + \ldots$$

$$+ \bigcup_{i=0}^{m(i+1)k_0} [D^{\nu-1}_i\vec{p}]$$

$$\nu=1$$

$$(2-58)$$
Since there are \( r^0 \) branches emerging from the initial node, we can partition \( S \) into \( r^0 \) disjoint subsets
\[
S_j = \left[\tilde{u}^{j}[\tilde{u}^{j}] = m_1^{j}[\tilde{g}^j] + m_2^{j}[\tilde{h}^j] + \ldots + m_k^{j}[\tilde{p}^j] + \sum_{i=1}^{\nu-1} m_{\nu k_0 + i}^{j}[D_i^{j}] \right]
\]
\[
+ \sum_{i=1}^{\nu-1} m_{\nu k_0 + i}^{j}[D_i^{j}] + \ldots + \sum_{i=1}^{\nu-1} m_{(i+1)k_0}^{j}[D_i^{j}] \]  
(2-59)

for \( 0 \leq j \leq r^0 - 1 \)

where \( m_1^{j} m_2^{j} \ldots m_k^{j} \) is a fixed \( k_0 \)-tuple for each \( j \) and \( m_1^{j} \) varies over \( \mathbb{F}_s \) for \( i > k_0 \). Each path in any one of the above subsets has the same prefix of \( n_0 \) digits long. The subset \( S_0 \) which corresponds to \( m_1^{0} = m_2^{0} = m_3^{0} = \ldots = m_{k_0}^{0} = 0 \) has zero prefix and contains the zero path of the truncated tree \( S \). We then obtain
\[
S = \biguplus_{j=0}^{r^0-1} S_j \]  
(2-60)

where \( \biguplus \) indicates the disjoint union.

As in Sec. 2-5, we define the distance between two distinct subsets \( S_i, S_j \) as
\[
\Delta (S_i, S_j) = \min_{\tilde{u} \in S_i, \tilde{v} \in S_j} d(\tilde{u}, \tilde{v}) \]  
(2-61)

where \( d(\tilde{u}, \tilde{v}) \) is the conventional Hamming distance (i.e., the number of places where \( \tilde{u} \) and \( \tilde{v} \) are different).

The minimum distance of the tree \( S \) is defined as
\[
D_{\min} = \min_{i,j} \Delta (S_i, S_j) \]  
(2-62)
All the theorems developed for the binary tree in Sec. 2-5 can also be applied to non-binary tree with little modification. Most theorems follow the same arguments as for the binary case. We thus state them without proof.

Beside the assumption we made about the set of \( k_0 \) generators in Eq. 2-53, we also assume that at least one of the \( k_0 \) digits, \( g_i, h_i, \ldots, p_i \) is not zero for \( k_0 + 1 \leq i \leq n_0 \). Then it is obvious that the tree \( S \) does not have a null position.

**Lemma 2-7-1:** At any component position \( n \) \((1 \leq n \leq n_0)\) in the non-binary truncated tree, at least \( r \) distinct elements of the field \( \text{GF}(q) \) will appear. \( r \) is the number of elements in the subfield \( F_s \).

**Proof:** consider the set of first segments of the \( k_0 \) generators

\[
S_1 S_2 S_3 \ldots S_{n_0} \\
h_1 h_2 h_3 \ldots h_{n_0} \\
\vdots \\
p_1 p_2 p_3 \ldots p_{n_0}
\]

(2-63)

The set of first order branches of the tree \( S \) is the set of all linear combinations of this set of first segments of the \( k_0 \) generators over the subfield \( F_s \). Suppose \( h_i \) is not zero, then \( f_1 h_i, f_2 h_i, \ldots, f_r h_i \) will appear on \( r \) distinct branches at \( i \)th position \((1 \leq i \leq n_0)\), where \( f_1, f_2, \ldots, f_r \) are \( r \) distinct elements of the subfield \( F_s \). Since \( f_1 h_i, f_2 h_i, \ldots, f_r h_i \) are \( r \) distinct field elements, therefore, at least \( r \) distinct field elements will appear at the \( i \)th position. Because the set of first-order branches will reappear at the top of any higher-order branches, the Lemma is then proved. Q.E.D.

**Lemma 2-7-2:** At any component position \( n \) in the tree \( S \), the number of appearances of any non-zero field element is equal to the number of appearances of the zero element of the field \( \text{GF}(q) \). Thus distinct
elements at component position $n$ will appear in equal numbers of paths (or codewords).

Proof: Follow the same argument as in Lemma 2-5-2.

Since the number of elements in $GF(q)$ is $q$, therefore, at most $q$ distinct elements will appear at component position $n(l - n_{o})$. Let $M = r^{K_{o}n_{t}}$ be the total number of paths (or codewords) in $S$. By the two above lemmas, we then can say that at most $\frac{1}{r}$ codewords in $S$ contain the zero element of the field $GF(q)$ at component position $n$, or at most $\frac{q-1}{q}$ codewords do not contain the zero of $GF(q)$ at $n$. Therefore, there are at most $\frac{q-1}{q}r^{K_{o}n_{t}}$ non-zero positions in the tree $S$ and there are at least $\frac{r-1}{r}(n_{t} - n_{o})M$ non-zero positions in the zero subset $S_{o}$. Thus, there are at most

$$N = \frac{q-1}{q}r^{K_{o}n_{t}} - \frac{r-1}{r}(n_{t} - n_{o})M$$

(2-64)

non-zero positions in the subset $S_{o}^{c}$, where $S_{o}^{c} = \bigcup_{i=1}^{K_{o}-1} S_{i}$. The total number of paths in $S_{o}^{c}$ is $r^{K_{o}}(r^{K_{o}-1})$. We then obtain that the average number of non-zero positions in a path of $S_{o}^{c}$ is

$$d_{ave} \leq \frac{N}{r^{K_{o}}(r^{K_{o}-1})} \leq \left(\frac{q-1}{q} + \frac{q-r}{qr(K_{o}-1)}\right)r_{t} + \frac{r-1}{r}n_{o}$$

(2-65)

The equality holds if and only if $r$ is equal to $q$ (or the subfield $F_{s}$ is equal to the field $GF(q)$). In that case, we have

$$d_{ave} = \frac{q-1}{q}\left\{r_{t} + \frac{n_{o}}{K_{o}-1}\right\}$$

(2-66)

In the non-binary case, we define the weight $W(\bar{u})$ of a path $\bar{u}$ in $S$ as the number of non-zero components in the path $\bar{u}$. Therefore,
The average weight of a path \( \tilde{u} \) in \( S_o^c \) is

\[
d_{\text{ave}} \approx \left( \frac{q-1}{q} \frac{q-r}{qr(r^{k_o-1})} \right)^{n_t} + \frac{r-1}{r} \frac{n_o}{r^{k_o-1}} \tag{2-67}
\]

**Theorem 2-7-1:** If \( S_i \) and \( S_j \) are two distinct disjoint subsets of \( S \) defined in Eq. 2-59, and \( \tilde{u} \) and \( \tilde{u}' \) are any two paths in \( S_i \), then the set of Hamming distances between \( \tilde{u} \) and every path \( \tilde{v} \) of the subset \( S_j \) is exactly the same as the set of Hamming distances between \( \tilde{u}' \) and every path \( \tilde{v} \) of \( S_j \).

**Proof:** See the proof of Theorem 2-5-2.

By the above theorem, we obtain

\[
\Lambda(S_i, S_j) = \min_{\tilde{v} \in S_j} d(\tilde{u}, \tilde{v}) \tag{2-68}
\]

where \( \tilde{u} \) is any path in \( S_j \).

**Lemma 2-7-3:** If \( \tilde{u} \) and \( \tilde{u}' \) are any two paths in the subset \( S_i \), then the set of Hamming distances \( \{d(\tilde{u}, \tilde{w}) ; \tilde{w} \in S_i^c \} \) is exactly the same as the set of Hamming distances \( \{d(\tilde{u}', \tilde{w}) ; \tilde{w} \in S_i^c \} \).

**Proof:** This is the direct result of theorem 2-7-1.

**Lemma 2-7-4:** Let \( S_i \) and \( S_j \) be the two subsets of \( S \) defined in Eq. 2-59, and \( \tilde{u} \in S_i^c, \tilde{v} \in S_j \). Then the set of Hamming distances between \( \tilde{u} \) and every path in \( S_i^c \) is the same as the set of Hamming distances between \( \tilde{v} \) and every path in \( S_j^c \).

**Proof:** See the proof of Lemma 2-5-5.
By Theorem 2-7-1, Lemma 2-7-3 and Lemma 2-7-4, we obtain

\[
\Delta_{\min} = \min_{i,j} \Delta(S_i, S_j) \\
= \min_{i} \Delta(S_i, S_j) \quad \text{for any } i \\
= \min_{i} \Delta(S_i, S_j) \quad \text{for any } j \\
= \min_{i} \min_{\bar{u}, \bar{v}} d(\bar{u}, \bar{v}) \quad \text{for any } i \text{ and any } \bar{u} \in S_i \\
= \min_{i} \min_{\bar{u}, \bar{v}} d(\bar{u}, \bar{v}) \quad \text{for any } j \text{ and any } \bar{v} \in S_j
\] (2-69)

\textbf{Definition:} The path with the smallest weight which is not in subset \( S_o \) is called the minimum weight path.

\textbf{Theorem 2-7-2:} The minimum distance of the tree code \( S \) is equal to the weight of the minimum weight path.

\textbf{Proof:} See the proof of Theorem 2-5-3.

\textbf{Theorem 2-7-3:} If any one of the \( k_o \) generators, say \( \bar{g} \), is replaced by a path in any of those subsets (defined in Eq. 2-59) which are formed by including \( [\bar{g}] \) (i.e., \( m_i^j \neq 0 \) for all \( j \)), the new set of \( k_o \) generators will generate the same tree \( S \).

\textbf{Proof:} See Theorem 2-5-4.

\textbf{Lemma 2-7-5:} The maximum number of non-binary tree codes that contain a particular \( n_t \)-tuples is

\[
Q^*_t = \frac{r-1}{r} r^{k_o} \left[ (q-1)^{n_t-k_o} \right]^{k-1}
\] (2-70)

where \( \nu = \frac{n_t}{n_o} \).
Proof: By the assumption made in Eq. 2-53, each of the \( k_0 \)
generators can have \((q-1)q^{(n-k_0)}\) possible choices. Now, if we fixed one
of them, say \( [\bar{g}] \), there will be \( \left[ \left( q-1 \right) q^{(n-k_0)} \right] \)
possible combinations of \( k_0 \) generators which contain the fixed \( [\bar{g}] \). Each of these
combinations will generate a tree code containing \( [\bar{g}] \). By Theorem 2-7-3,
each of these combinations with \( [\bar{g}] \) replaced by a path in any of those
subsets which are formed by including the generator \( [\bar{g}] \) will generate
the same tree. Since there are \( (r-1)q^{(n-k_0)} \) subsets formed by including
\( [\bar{g}] \) and each of these subsets contains \( r \) paths, therefore,
there are
\[
\frac{(r-1)q^{(n-k_0)}}{r} = \frac{(r-1)q^{(n-k_0)}}{r}
\]
paths can be used to replace \( [\bar{g}] \) without changing the tree \( S \). Hence,
the total number of tree codes which will contain \( [\bar{g}] \) is
\[
Q^* = \frac{r-1}{r} q^{(n-k_0)} \left[ \left( q-1 \right) q^{(n-k_0)} \right] \left( q^{(n-k_0)} \right)
\tag{2-71}
\]

Theorem 2-7-4: With transmission rate \( R_t = \frac{k}{n} \) and constraint
length \( n_t \), there exists at least a tree code for which the minimum
distance \( D_{\text{min}} \) exceeds \( D \), where \( D \) is the largest integer satisfying
the inequality
\[
\sum_{i=1}^{k_0} \frac{k_0}{C_i (q-1)} \frac{D-i}{\sum_{j=0}^{n_t-k_0} (q-1)^j} \frac{n_t}{q} \frac{k_0}{v_k} \frac{q}{r_k} \leq 1 \tag{2-72}
\]

Proof: Because of the assumption in Eq. 2-53, any path which is not
in the zero subset \( S_0 \) will have at least one non-zero digit in its first
\( k_0 \) positions. Thus, the total number of \( n_t \)-tuples which have weight
less than or equal to \( D \) and have at least one non-zero digit in the
first \( k_0 \) positions is

\[
N^*_D = \sum_{i=1}^{k_0} C_i^k (q-1)^i \sum_{j=0}^{D-i} C_j^n t^{-k_0} (q-1)^j
\]

For each of these \( N^*_D \) sequences, there are at most

\[
Q^* = \frac{r-1}{r} \frac{k_0^k}{r} \frac{n_t^{-k_0} k_0^{-l}}{(q-1)q^{t-1}}
\]

codes which contain that sequence. And since the source alphabet has \( r \) elements, there will be at least \( r-1 \) paths of the same weight in each non-binary tree code. Therefore, there are at most

\[
\frac{1}{r-1} N^*_D Q^*
\]

codes with minimum distance \( D_{\min} \) less than or equal to \( D \).

By the assumption made in Eq. 2-53, there are

\[
N^* = \sum_{i=0}^{n_t} \sum_{i=0}^{k_0} (q-1)^i
\]

possible choices for the set of \( k_0 \) generators. In other words, there are \( N \) possible tree codes.

If

\[
\frac{1}{r-1} N^*_D Q^* < N^*
\]

(2-74)

there must be at least one code among \( N^* \) for which the minimum distance \( D_{\min} \) is greater than \( D \).

Eq. 2-74 may be rearranged to give

\[
N^*_D < (r-1) \frac{N^*_D}{Q^*}
\]

which means

\[
\sum_{i=1}^{k_0} C_i^k (q-1)^i \sum_{j=0}^{D-i} C_j^n t^{-k_0} (q-1)^j < \frac{r(q-1)}{k_0} \frac{n_t}{q^{\frac{k_0}{r}}}
\]

(2-75)

where \( D \) is the largest integer satisfying the inequality.
This is the Gilbert bound for non-binary tree code. In case that \( q = r = 2 \), Eq. 2-72 will reduce to Eq. 2-44 which is the Gilbert bound for binary tree codes.

In Eq. 2-67, we have shown that the average weight of a path in \( S^C_o \) is

\[
d_{\text{ave}} \leq \left( \frac{q-1}{q} + \frac{q-r}{qr(r^k-1)} \right) \left( nt + \frac{r-1}{r} \frac{n_o}{r^k-1} \right)
\]

(2-67)

Since the weight of the minimum weight path in \( S^C_o \) is less than the average weight of a path in \( S^C_o \), therefore, we obtain the following theorem.

**Theorem 2-7-5:** The minimum distance \( D_{\text{min}} \) of a non-binary tree satisfies the following inequality.

\[
D_{\text{min}} \leq \left( \frac{q-1}{q} + \frac{q-r}{qr(r^k-1)} \right) \left( nt + \frac{r-1}{r} \frac{n_o}{r^k-1} \right)
\]

(2-76)
Fig. 2-2.
### Fig. 2-3.

<table>
<thead>
<tr>
<th>0</th>
<th>[0]</th>
<th>Half of the elements in $U_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>$g(1)$</td>
<td>Half of the elements in $U_\infty$</td>
</tr>
</tbody>
</table>

### Fig. 2-4.

<table>
<thead>
<tr>
<th>0</th>
<th>[0]</th>
<th>$\frac{1}{4}$ of $U_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_2$</td>
<td>$g(1)$</td>
<td>$\frac{1}{4}$ of $U_\infty$</td>
</tr>
<tr>
<td>$m_1$</td>
<td>$g(2)$</td>
<td>$\frac{1}{4}$ of $U_\infty$</td>
</tr>
<tr>
<td>1</td>
<td>$g(1) \oplus g(2)$</td>
<td>$\frac{1}{4}$ of $U_\infty$</td>
</tr>
</tbody>
</table>
Fig. 2-5a. Table showing the development of a single-generator tree code.
Fig. 2-5b. An infinite tree for transmission rate $R_t = \frac{1}{n_o}$.
Fig. 2-6. First few branches for a tree code with $n_o = 3$, $k_o = 1$. 
Fig. 2-7a. General circuit diagram. The long dash lines are used to indicate that the connections may or may not exist.
Fig. 2-7b. The connections for previous example

with \( n_o = 3, \quad \tilde{g} = 111 \ 010 \ 001 \).
\[ G = \begin{pmatrix}
  g(1) & g(2) & g(3) & \cdots & g(\nu) \\
  [0] & g(1) & g(2) & \cdots & g(\nu-1) \\
  [0] & [0] & g(1) & \cdots & g(\nu-2) \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  [0] & \cdots & \cdots & \cdots & g(1)
\end{pmatrix} \]

Fig. 2-8
Fig. 2-9.
Figure 2-11.
<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>000</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>011</td>
<td>011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>101</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2-12.
Fig. 2-13.
Fig. 2-14.
Fig. 2-15. The encoding circuit for
\[ \tilde{g} = (101 \ 001 \ 010) \]
\[ \tilde{h} = (011 \ 010 \ 001) \]

with \( k_o = 2 \), \( n_o = 3 \).
[G] =

\[
\begin{pmatrix}
g(1) & g(2) & g(3) & \ldots & g(\gamma) \\
h(1) & h(2) & h(3) & \ldots & h(\gamma) \\
. & . & . & \ldots & . \\
. & . & . & \ldots & . \\
p(1) & p(2) & p(3) & \ldots & p(\gamma) \\
g(1) & g(2) & \ldots & g(\gamma-1) \\
h(1) & h(2) & \ldots & h(\gamma-1) \\
. & . & \ldots & . \\
. & . & \ldots & . \\
p(1) & p(2) & \ldots & p(\gamma-1) \\
g(1) & . & . & . & . \\
h(1) & . & . & . & . \\
. & . & . & . & . \\
p(1) & . & . & . & . \\
g(1) & . & . & . & . \\
h(1) & . & . & . & . \\
. & . & . & . & . \\
p(1) & . & . & . & . 
\end{pmatrix}
\]

Fig. 2-17.
Fig. 2-18. Search for the second segment of the generator sequence.
<table>
<thead>
<tr>
<th>0 0 0</th>
<th>0 0 0</th>
<th>0 0 0</th>
<th>0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 1 1</td>
<td></td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>0 0 1</td>
<td></td>
<td>0 0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 1 0</td>
<td>1 1 1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>g(3)</td>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td></td>
<td>1 1 1 ⊕ g(3)</td>
<td></td>
<td>1 1 1 ⊕ g(3)</td>
</tr>
<tr>
<td></td>
<td>0 0 1 ⊕ g(3)</td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 0</td>
<td>1 1 0</td>
<td>1 1 0</td>
</tr>
</tbody>
</table>

(a) \( g(1) = 1 1 1 \)
\( g(2) = 0 0 1 \)

(b) \( g(1) = 1 1 1 \)
\( g(2) = 0 0 1 \)
\( g(3) = 0 0 1 \)

<table>
<thead>
<tr>
<th>0 0 0</th>
<th>0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>0 0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

(c) \( g(1) = 1 1 1 \)
\( g(2) = 0 0 1 \)

Fig. 2-19. Search for the third segment of the generator sequence.
Fig. 2-20. An infinite non-binary tree
CHAPTER 3. A SEQUENTIAL DECODING SYSTEM UTILIZING DISTANCE PROPERTIES OF CONVOLUTIONAL TREE CODE

In this chapter we present a sequential decoding scheme designed to exploit the class of convolutional tree codes studied in Chapter 2. This decoding scheme is similar in some ways to the known existing schemes proposed by Wozencraft [1] and Ziv [4] but is different in many other ways. The most important features of this new scheme can be summarized as follows:

(1) Strategies have been devised to utilize the full error-correcting capability of the codes and to deal with the difficult problem of error propagation, seemingly inherent in sequential decoding system.

(2) A sensitive way, called multiple-threshold test, has been provided to aid in early detection and approximate location of decoding errors. This test is especially effective when the tree codes have quick separation property.

(3) Although estimates of the probability of error and of numbers of decoding operations are based on the oversimplified binary symmetric channel model, the procedures have been designed to deal with "burst errors" within the error-correcting capability of the codes.
3.1. Fundamental Distance Property for a Convolutional Tree Code

Before we discuss the decoding strategy it would be advisable to summarize briefly the more important properties of the code and, in particular, to formulate a fundamental distance property which provides the error-correcting capability of a code. For simplicity, we shall limit our consideration to the tree codes generated by a single generator sequence (i.e., the tree codes correspond to the transmission rate $R_t = 1/n_o$). The constraint length is $n_C = n_o \cdot \nu$ digits or $\nu$ branches. In this case, there are two branches emerging from each node of the tree, and each branch is $n_o$ digits long, as shown in Fig. 3-1.

For many purposes, it is desirable to consider paths of various finite lengths. If $k$ is any positive integer, it is convenient to call the set of all paths of length $k$ branches or $kn_o$ digits stemming from a given node the $k$-unit stemming from that node. Fig. 3-2 shows a 2-unit in the code represented on the tree of Fig. 3-1. Just as we defined the minimum distance for a truncated tree (a $\nu$-unit in our present terminology), similarly we define a minimum distance for any $k$-unit.

As shown in Chapter 2, any $k$-unit may be derived by adding to the $k$-unit stemming from the initial node an appropriate sequence of $kn_o$ digits. This modification leaves the minimum distance for the $k$-unit unchanged. We may therefore state the

Fundamental distance property for a convolutional code. The minimum distance of any $k$-unit in the tree depends only upon the length $k$ and does not depend upon the node from which the $k$-unit stems.

This means that we can introduce a distance function $d(\cdot)$ for $k$-units which depends only on $k$. Thus, $d(k)$ is the minimum distance for every
k-unit in the code tree. As we shall see, one of the most important characteristics of a convolutional code for sequential decoding is the structure of the distance function $d'$. The codes constructed according to the rules outlined in Chapter 2 have the property that $d(k)$ increases relatively rapidly with $k$ for small values of $k$, and has smaller rates of increase at larger values of $k$. 
3-2. Probabilistic Decoding

Throughout this chapter we shall assume that the technical problem of synchronizing the operation of the encoder at the transmitting end and the operation of the decoder at the receiving end is solved. Some attention to possible synchronization checks is given in a later section, but the problem of synchronization is considered primarily a technical problem, not dealt with in the encoding-decoding strategy.

We shall assume that the decoder stores \( n_t = n_o \) channel output digits and operates to compare this sequence of digits with possible paths in the code tree. An attempt is made to identify the corresponding path traced in the code tree by the encoder, under the control of the original message. In the process of decoding, a tentative choice of a path means a tentative decoding of the channel output digits in the decoder. If there were no noise in the transmission channel to perturb the transmitted signal, the sequence in the decoder would correspond exactly with a path in the tree, and this path could be identified with certainty. In the presence of noise, however, the received sequence may correspond exactly with no path in the tree. In this case, a choice must be made as to which path is most probable.

It is always difficult, and usually impossible, to calculate the probabilities upon which such a choice may be made. The practical approach is to make a decision on the basis of the Hamming distance -- that is, on the basis of the number of places in which the two sequences under consideration differ. It is assumed that a smaller perturbation of the received sequence away from the actual transmitted sequence is more probable than a larger one. This is demonstrably true for the binary symmetric channel;
and it is reasonable to assume that it is true for practical binary channels.

When a choice has been made as to the most probable path, the decoder shifts out $n_0$ channel digits and prints out the corresponding decoded message digit. This means that a final decoding commitment has been made on this message digit. At the same time, the decoder shifts the next $n_0$ channel digits into the decoder. The addition of this new information in the decoder calls for a new decision regarding the path, and the process of making a choice is repeated.

The decoding scheme described below is a systematic method of comparing paths, in a manner designed to exploit the properties of the code. When perturbations due to noise are infrequent, the search for a path proceeds on a branch-by-branch basis, with a minimum number of decoding operations. Distance properties of the code provide very sensitive tests for an error in the choice of path, which amounts to providing tests for an error in decoding. When noise causes a temporary choice of an incorrect path, algorithms for a more sophisticated search are provided. We begin by describing the simplest running mode and the tests for an incorrect path. Then we consider some algorithms for search when an incorrect path is indicated.
3-3. Decoding Procedure

We suppose the decoder has found the correct path at the full length of \( v \) branches. One segment of \( n_0 \) channel digits is shifted out of the decoder and a new segment is shifted in. The decoder selects the new path correctly if it selects the single new branch correctly. It has one of two choices. This choice is made by comparing the new segment of \( n_0 \) received digits with the two possible branches in the tree and selecting the nearest one, which is the "most probable" one.

Since the code has the property that all 1-units have the same distance \( d(1) \) between the two branches in the unit, only one comparison operation is necessary to make the choice. Good codes are designed with \( d(1) = n_0 \), which is the maximum possible separation. This means that the two branches stemming from a single node are the 1's complements of each other. A comparison of the received segment with the upper branch shows whether the distance is less than \( n_0/2 \). If it is less than \( n_0/2 \), this upper branch is chosen; if not, the lower branch is chosen. The operation of generating the lower branch is a simple one of complementing the upper branch. It is apparent that the mechanization of this operation in terms of computer type operations is quite easy.

Once the correct path is found for \( v \) segments, the search can continue on a branch-by-branch basis, so long as the noise perturbation in any one segment does not exceed \( n_0/2 = d(1)/2 \). In low noise situations, this condition may prevail for comparatively long periods of time. During such periods, decoding is simple and rapid. Problems arise, however, when the density of errors increases for a period of time. If the number of channel errors in a single segment exceeds \( n_0/2 \), the branch-by-branch
search results in the choice of an incorrect path. Once on an incorrect path, branch-by-branch search cannot possibly result in a return to the correct path. To be successful, the decoding scheme must solve two problems:

(1) It must be able to detect the fact that a wrong path has been chosen.

(2) It must provide strategies for returning to the correct path once the fact of an incorrect choice is detected.

The Threshold Test for Incorrect Paths

Once an incorrect choice of a branch is made, the subsequent branches must be incorrect. As successive branches are chosen, the fundamental distance property of the code drives the selected (but incorrect) path away from the correct path. The distance increases with the number of branches beyond the node of separation -- that is, beyond the node at which the first incorrect choice is made. For \( k \) such branches, the distance between the correct path and the incorrect one must be at least \( d(k) \). To see this, we may argue as follows. Let \( \tilde{u} \) be the correctly transmitted sequence, \( \tilde{w} \) the tentatively decoded sequence corresponding to the path chosen, and \( \tilde{v} \) the sequence received at the channel output. We suppose that the first \( v-k \) branches in the decoder are correct and the last \( k \) are incorrect. The distance between \( \tilde{u} \) and \( \tilde{w} \) will then be due to the distance in the last \( k \) branches. As a matter of fact, \( \tilde{u} \) and \( \tilde{w} \) will lie in opposite (upper and lower) halves of the k-unit stemming from the node of separation. We must therefore have

\[
d(\tilde{u}, \tilde{w}) \geq d(k)
\]
We may utilize this property to obtain an effective test for detecting the fact of an incorrect choice before the first incorrect branch is shifted out of the decoder. A fundamental property of distance functions (the triangle inequality) enables us to assert

\[ d(\tilde{u}, \tilde{v}) + d(\tilde{v}, \tilde{w}) \geq d(\tilde{u}, \tilde{w}) \]

\[ d(\tilde{v}, \tilde{w}) \geq d(\tilde{u}, \tilde{w}) - d(\tilde{u}, \tilde{v}) \]

Utilizing the separation property derived above, we have the relation

\[ d(\tilde{v}, \tilde{w}) \geq d(k) - d(\tilde{u}, \tilde{v}) \]

The distance between the received sequence \( \tilde{v} \) and the tentatively decoded sequence \( \tilde{w} \) is at least as great as the distance \( d(k) \), reduced by the amount of the distance between the received sequence \( \tilde{v} \) and the correct (transmitted) sequence \( \tilde{u} \). Note that \( d(k) \) is a known property of the code and \( d(\tilde{v}, \tilde{w}) \) is an observable quantity in the decoder.

To simplify the statement of the test, we introduce a threshold function \( T(\cdot) \), derived from the distance function \( d(\cdot) \), as follows:

**Definition:** For each integer \( k \), the value \( T(k) \) of the threshold function \( T(\cdot) \) is that integer satisfying the relationship

\[ T(k) < d(k)/2 \leq T(k) + 1. \]

It may be noted that \( T(v) \) is the quantity known as the error-correcting capability of a block code of length \( n = v \cdot n_0 \) digits with minimum distance \( d(v) \). The quantity \( T(v) \) plays a similar role in the sequential decoding scheme described in this paper, and we shall frequently refer to it as the error-correcting capability of the convolutional code.

Suppose the distance \( d(\tilde{u}, \tilde{v}) \) between the correct sequence \( \tilde{u} \) and the received sequence \( \tilde{v} \) is no greater than \( T(v) \). In well-designed codes this is true with very high probability. Then if \( k = v \) we must have
\[ d(\tilde{v}, \tilde{w}) > d(\nu) - T(\nu) \quad \text{and} \quad d(\nu) > 2T(\nu) \]

so that

\[ d(\tilde{v}, \tilde{w}) > T(\nu) \]

We could, of course, have the last relationship for fewer than \( \nu \) incorrect branches. Also, by the triangle inequality we may state that

\[ d(\tilde{u}, \tilde{w}) \geq |d(\tilde{v}, \tilde{w}) - d(\tilde{u}, \tilde{v})| \]

The conditions \( d(\tilde{v}, \tilde{w}) > T(\nu) \) and \( d(\tilde{u}, \tilde{v}) \leq T(\nu) \) thus imply \( d(\tilde{u}, \tilde{w}) > 0 \), which means that at least one branch is incorrect. We may thus state the following theorem, which provides an important

**Threshold Test:** Suppose the distance between the transmitted sequence \( \tilde{u} \) and the received sequence \( \tilde{v} \) at length \( \nu \) segments is less than the error-correcting capability \( T(\nu) \) of the code; then

(i) if the distance \( d(\tilde{w}, \tilde{v}) \) between the received sequence \( \tilde{v} \) and the tentatively decoded sequence \( \tilde{w} \) in the decoder is greater than \( T(\nu) \), there must be at least one incorrect segment in \( \tilde{w} \).

(ii) if the first branch in the decoder (i.e., the one to be shifted out next) is incorrect, then \( d(\tilde{w}, \tilde{v}) \) exceeds \( T(\nu) \) for each possible sequence \( \tilde{w} \).

We cannot count on this test if the number of channel errors exceeds \( T(\nu) \). Because of the low probability of the occurrence of the latter condition, we bet that it does not occur. If it does, we are in trouble anyhow, because the error-correcting capability of the code has been exceeded, and we must be prepared to accept a possible decoding error.

This threshold test provides a highly reliable basic test for decoding errors within the error-correcting capabilities of the code. It has the disadvantage, however, that the node of separation may be
shifted a relatively large number of segments into the decoder before the error is detected. As the discussion in a later section shows, this situation requires a rather large number of decoding operations to correct the mistake. It would be desirable to be able to detect decoding errors as soon as possible after they occur. Also, it would be desirable to be able to locate the node of separation, where the error first occurs. In fact, if the node of separation were known precisely, the error at that point could be corrected immediately in single-generator, binary codes, since there are only two possible branches stemming from that node.

**Multiple-threshold Test**

The basic threshold test provides a test for detecting an incorrect decoding choice within a \( v \)-unit, provided the error-correcting capability \( T(v) \) of the code is not exceeded in the \( v \) segments spanned by that unit. The test is applied to the tentatively decoded paths of full length \( v \) segments in the decoder.

An extension of this testing procedure, to aid in early detection and approximate location of decoding errors, may be made by considering sections of the path of shorter length. Each path in the decoder consists of a sequence of digits. These are to be compared with the corresponding digits in the received sequence. In dealing with these sequences, it is convenient to write them in their usual order, with the latest digits in time to the right. To maintain this convention in our graphical and verbal representations, we suppose the digits are located physically in a decoder register in this order. Digits are shifted to the left, entering the decoder register at its right-hand end and leaving at its left-hand end. The earliest digits are thus farthest to the left in the decoder register and the latest digits are farthest to the right.
Suppose a tentatively decoded path $\tilde{w}$ is correct for the first $v$-b segments in the decoder, and the threshold test is applied to the last $b$ segments. Because of the fundamental distance property, we may deal with this last $b$ segments as if we had a code with constraint length $b$ segments and error-correcting capability $T(b)$. Thus, if the error-correcting capability $T(b)$ is not exceeded by the noise in the last $b$ branches, a distance greater than $T(b)$ in the last $b$ segments between $\tilde{v}$ and $\tilde{w}$ indicates a decoding error within that b-unit. And if, under the assumed conditions, an error occurs, it will be indicated before the node of separation is shifted out of the last $b$ positions in the decoder.

The test is inconclusive if an error in decoding occurs before the last $b$ segments, or if the channel noise exceeds $T(b)$ in the $b$ segments under consideration. Because of the rapid early separation properties of the code, the error-correcting capability $T(b)$ takes on substantial values for relatively small $b$. The probability of channel noise exceeding $T(b)$ may then be rather small. Values for a binary, single-generator code with rate $R = 1/3$ and a binary symmetric channel with transition probability $p_0 = 0.02$ are shown in the table below.
### PROBABILITY THAT CHANNEL NOISE EXCEEDS $T(b)$

- **$R = 1/3$**
- **BSC with $p_o = 0.02$**

<table>
<thead>
<tr>
<th>$b$</th>
<th>$d(b)$</th>
<th>$T(b)$</th>
<th>$P$</th>
<th>$2^2$</th>
<th>$2^{b+1} - 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>$1.2 \times 10^{-3}$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>$5.7 \times 10^{-3}$</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>$6.1 \times 10^{-4}$</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
<td>$1.5 \times 10^{-3}$</td>
<td>16</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>$1.8 \times 10^{-4}$</td>
<td>32</td>
<td>62</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>3</td>
<td>$3.9 \times 10^{-4}$</td>
<td>64</td>
<td>126</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>4</td>
<td>$5.0 \times 10^{-5}$</td>
<td>128</td>
<td>254</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>4</td>
<td>$9.9 \times 10^{-5}$</td>
<td>256</td>
<td>510</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>4</td>
<td>$1.8 \times 10^{-4}$</td>
<td>512</td>
<td>1022</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>5</td>
<td>$2.5 \times 10^{-5}$</td>
<td>1024</td>
<td>2046</td>
</tr>
</tbody>
</table>

Suppose $b$ is taken to be seven; then, if no decoding error is made before the last seven segments (21 channel digits) in the path, and if no more than four channel errors occur among the last 21 channel digits, the received signal $\bar{v}$ and the tentatively decoded path $\bar{w}$ differ in more than four of the last 21 places only if there is a decoding error in the last seven branches. This statement does not depend upon the channel being a binary symmetric channel, although the estimates of probabilities in the table above were calculated on that basis. If a short burst of two, three, or four consecutive or very closely spaced errors should occur, but no further ones, in 21 successive digits, branch-by-branch decoding could result in the choice of an incorrect path. The threshold test applied to the last seven segments in the path would discover the error before it is shifted out of the 7-unit under consideration. The discussion in a later section shows that the
decoding error can be corrected before it is shifted out of this unit.

We may utilize the facts discussed and illustrated above to achieve early detection and approximate location of decoding errors. To do so, we utilize the following:

MULTIPLE-THRESHOLD TEST. Consider a sequence of paths of increasing length, consisting successively of the last \( b_1, b_2, \ldots, b_m, \nu \) segments of the tentatively decoded path in the decoder. Apply the threshold test to each of these partial paths, using the corresponding threshold \( T(b_i) \) for the path of length \( b_i \) segments. Suppose \( b_r \) is the largest number of segments for which the corresponding threshold \( T(b_r) \) is exceeded. We may assert there is a decoding error in the last \( b_r \) segments of the tentatively decoded path if the following conditions hold:

1. The channel noise does not exceed the appropriate thresholds at lengths of \( b_r, b_{r+1}, \ldots, b_m, \nu \) segments, respectively.
2. There is no decoding error in the first \( \nu - b_r \) segments.

The situation is represented diagrammatically in Fig. 3-3.

The usefulness of the multiple-threshold test rests upon the fact that the two conditions hold with very high probability. To illustrate, suppose we let \( b_1 = 7 \) and \( b_2 = 14 \). Values of \( T(7) = 4 \) and \( T(14) = 6 \) have been achieved in binary codes with rate \( R = 1/3 \). To estimate probabilities, we assume a binary symmetric channel with \( p_o = 0.02 \). Let us start with a correct path in the decoder. The first error that is made will be discovered (and corrected) before it has been shifted out of the last seven segments if the noise in these last seven segments
(21 channel digits) does not exceed $T(7)$. The probability that this value is exceeded is $P_7 = 5 \times 10^{-5}$. If the noise is too severe, the error may escape detection or correction in these seven positions and be shifted further into the decoder. It will be discovered in the last 14 segments if the noise in these segments does not exceed $T(14)$. The probability that this value is exceeded is $P_{14} = 1.9 \times 10^{-5}$. Similar statements can be made for larger values of $b_1$, with increasingly smaller probabilities that the error will escape detection and correction.

We apply the multiple-threshold test (MTT), bet on the appropriate conditions being met, and act accordingly. In the unlikely event that the conditions are not met at a given position, they are met with increased probability at a later stage of decoding, unless the full error-correcting capability of the code $T(v)$ is exceeded by the channel noise in the full length of $v$ segments in the decoder. One of the features of decoding procedures based on the MTT is that the further a tentatively decoded segment moves into the decoder, the greater the probability that it is correct. A segment is shifted out of the decoder and its corresponding decoded digit is printed out precisely at the stage at which we are most certain of its correctness -- as certain as we can be within the error-correcting capability of the code.

The numbers $b_1$ appear as design parameters of the system. One natural approach is to determine $b_1 = b$ and then let $b_k = kb$. It is desirable to choose $b$ so that the probability that channel error exceeds $T(b)$ in $n_0 b$ channel digits is reasonably small and yet the number of paths in the $b$-unit is not too large. Since $T(b)$ is not strictly increasing, the probability in question is not quite a decreasing function of $b$. Examination of the table above shows that $b = 7$ is a
good choice for this code, resulting in a good compromise between small probability and large number of paths. The design choice is obviously dependent upon the distance function \( d(\cdot) \) or, equivalently, upon the threshold function \( T(\cdot) \) for the particular code used.

**Error-correcting Strategy**

No single strategy is logically implied by the adoption of the MTT. The following strategy is designed to probe systematically, in a manner that exploits the MTT and the code properties. A number of variations are readily apparent, and some of these are worth experimental investigation. We describe what seems to be the most promising strategy.

When the MTT has indicated an error in a tentatively decoded path, we suppose, as before, that \( b_r \) is the largest number of segments for which the corresponding threshold \( T(b_r) \) is exceeded. We accept the first \( \nu - b_r \) segments of the path in the decoder and search the \( b_r \)-unit at the end of this partial path for a new tail consisting of \( b_r \) segments. A procedure for carrying out this search is discussed in the next paragraph. We suppose the search produces a new tail which lies at least as close to the corresponding tail of the received sequence \( \bar{v} \) as does the original tail (to be replaced). Apply the MTT to the new tail; this is equivalent to applying the MTT to the full-length path consisting of the \( \nu - b_r \) original segments and the \( b_r \) new segments. Suppose \( b_s \) is the greatest number of segments in the new path for which the corresponding threshold \( T(b_s) \) is exceeded. Obviously, \( b_s \leq b_r \). We accept the modified full-length path for the first \( \nu - b_s \) segments and search the \( b_s \)-unit at the end of this partial path for a new tail of \( b_s \) segments. This process is repeated until a path is found which
satisfies all thresholds in the MTT. When such a path is found, it is treated as if it were the correct path; branch-by-branch decoding is resumed until the MTT gives a new indication of a decoding error. If no such path is found, consider the path with the smallest $b_s$; use the $v$-segments previously accepted segments of that path and complete with the best $b_s$-segment tail produced by the search procedure. The resulting path is treated as if it were correct, and branch-by-branch decoding is resumed -- at least for the next branch. It is assumed that any decoding errors accepted will be rediscovered by the MTT at a later stage, before the branch following the node of separation is shifted out of the decoder.

Let us illustrate with the case $b_r = b_2 = 2b$ and $b_1 = b$. We accept the first $v$-2b segments of the tentatively decoded path in the decoder. We search the 2b-unit at the end of this partial path for a new 2b-segment tail which lies at least as close to the tail of the received sequence $\bar{v}$ as the tail to be replaced. The new tail is subjected to the MTT. Each of the two thresholds $T(b)$ and $T(2b)$ will either be exceeded or not. We tabulate the various possibilities that may occur and describe the subsequent procedure in each case. We let a 1 indicate that a particular threshold is exceeded and a 0 indicate that it has not been exceeded.
T(2b) T(b)

1. 0 0 Accept the whole path. Proceed with decoding on a branch-by-branch basis, as if the path in the decoder were correct, until the MTT gives a new indication of an error.

2. 0 1 Accept the first b segments of the new section of the path; search the last b-unit for a tail which satisfies T(b).
   (a) If any such tail is found, accept the whole path and proceed as in case 1.
   (b) If no such path is found after a complete search, accept the best possible tail produced by the search to complete the path. Having accepted the whole path, proceed as in case 1.

3. 1 0 Search out the 2b-unit for a new tail of 2b segments; apply the MTT to this tail. If case 1 or 2 now holds, proceed accordingly; if not, continue to search for a satisfactory tail. If a search through the 2b-unit does not produce a satisfactory tail, accept the best possible tail produced by the search to complete the path. Proceed as in case 1.

4. 1 1 Treat exactly as in case 3.

A Search Procedure

The problem is to try to find a path within the b_r-unit which satisfies the MTT. In particular, we seek within the b_r-unit considered above a path which lies within a distance T(b_r) of the b_r-segment tail of the received sequence v.
Instead of making a branch-by-branch search, we make a branch-by-branch choice based on a b-unit search. The b-unit stemming from the same node as the $b_r$-unit is searched by generating the various paths, one branch at a time. Each partial path is compared with the corresponding portion of the received sequence $\tilde{v}$. If at any point in the development of a path the distance exceeds $T(b_r)$, this path is discarded and no further branches are generated on any path stemming from the last node considered. We may refer to this procedure as search with discard [at distance $T(b_r)$]. This procedure is represented schematically in Fig. 4.

If one or more paths survive the search and discard procedure to the full length of $b$ segments, the most probably of these is selected. The first segment of this path (lying in the $b_r$-unit) is accepted, and a new search with discard is begin in the b-unit stemming from the accepted branch. Distances are measured over all segments lying within the $b_r$-unit. The process is repeated in the manner represented diagrammatically in Fig. 5. By repetitive b-unit searching and accepting of one branch, a path is traced to a point where only $b$ segments remain to be determined. These last $b$ segments are taken to be the most probable path in the last b-unit. If this procedure is successful, a path is found in the $b_r$-unit for which the $T(b_r)$ threshold is satisfied.

If at any point in the process it is impossible to trace a path through the b-unit being searched with discard, a larger $(b+1)$-unit is searched. This is done by backing up one segment, considering the alternate branch, and running through the b-unit stemming therefrom. This process may be visualized with the aid of Fig. 6. If, in turn, this process results in no paths through the $(b+1)$-unit which satisfy the
threshold $T(b_r)$, a $(b+2)$-unit is searched with discard. Whenever a path is found which penetrates to the end of the unit being searched, say a $(b+k)$-unit, accept $k+1$ segments and proceed with the branch-by-branch choice based on a $b$-unit search.

Should it be necessary to back up to the first node in the $b_r$-unit and no path satisfying the threshold $T(b_r)$ exists, several possibilities for determining the best possible path may be considered. One could continue the search with a relaxed threshold value, greater than $T(b_r)$. An alternative which is desirable to prevent an excessive number of decoding operations is to take the original path as the best possible choice. Since a high noise condition must exist in the case considered, the strictly "most probable" choice could very well be incorrect. Thus, it seems reasonable to take the original path, which could, with reasonably large probability, be the correct one. The choice between alternatives could depend upon the manner of implementing the decoding procedures. At any rate, the machine would be programmed to make a specific choice to be considered as the "best possible path".
3-4. **Basic Starting Procedure**

The decoding procedure described above presumes there is a tentatively decoded path in the decoder. The MTT is applied to this sequence, and the indicated action is taken. The problem remains how to obtain most efficiently a tentatively decoded path for the first \( v \) message digits or the first \( v \) segments of the received sequence. We suppose that it is known where the encoder starts in the tree when the first message digits are encoded. In view of the operations already required in the decoding procedure described above, two possibilities suggest themselves: either a branch-by-branch search or a branch-by-branch choice based on a b-unit search.

Several factors point to the desirability of beginning with a branch-by-branch search to establish the first trial path in the decoder. Unless the noise is momentarily severe, the branch-by-branch search has a high probability of continuing without error for a considerable number of operations. When a path of length \( v \) segments has been determined on this basis, the MTT is applied. Then if the MTT is violated, one begins at the indicated point with the process of branch-by-branch choice based on a b-unit search. Each b-unit search may require more operations than the full branch-by-branch search required to establish a trial path. Because of the very high probability that one or more b-unit searches are saved by first making the branch-by-branch search, it is clearly desirable to proceed on this basis.
3-5. Periodic Restart of Decoding

One of the serious problems of sequential decoding is its tendency to propagate decoding errors. Once a decoding error has been allowed to work through the decoder, it may not be possible by ordinary search procedures to rediscover the correct path. Thus, once a final commitment has been made on the incorrect decoding of a message digit, the tests and procedures for decoding become highly unreliable, and subsequent errors are highly probable. To limit the extent of such error propagation, we may use a fundamental property of the convolutional tree codes stated in Theorem 2-1.

The first step in the procedure is to inject a known sequence of $\nu-1$ digits into the message sequence at prescribed intervals -- say after every $10^\nu$ regular message digits. The result is to separate the message into what we may call sections. The block of $\nu-1$ known digits we shall call the restart sequence, because of the manner in which we utilize it. Periodic injection of the restart sequence reduces the effective transmission rate $R_\tau$ slightly -- about 10 percent if it is introduced after every 10 regular message digits.

Since we assume synchronous action, the decoder "knows" where the encoded segments corresponding to the restart sequence is in the transmitted sequence. When the corresponding segments in the received sequence are fed into the decoder, the decoder automatically decodes correctly by making the appropriate choice of branch at each node. This choice is determined by the pattern of the digits in the restart sequence injected into the sequence of message digits. If the decoding of the last segment of the message section is correct, the path in the decoder produced according to the pattern of the restart sequence will be the correct one. If
one or more of the last message digits are decoded incorrectly, then then path produced will correspond to an incorrect portion of the code tree and the path produced will not be the correct path. However, regardless of the correctness or incorrectness of this path, the decoder will arrive at a branch such that the tree stemming therefrom will correspond exactly to that entered next by the encoder. If the decoder makes subsequent correct choices of the branches, the path thus generated will correspond exactly to the correct one.

As the segments of the received sequence corresponding to the restart sequence are fed into the decoder, decoding continues normally with one exception. Whenever the decoder is operating on any segment corresponding to a digit in the restart sequence, it is constrained to the choice of branch determined by the pattern prescribed by the restart sequence. Once all \( \vee \) segments corresponding to the restart sequence are in the decoder, no corrective search is possible. If the MTT is satisfied, decoding can continue into the next message section on a branch-by-branch search basis. Should the MTT indicate that an error has occurred in decoding (in the digits of the previous message section) or excessive channel noise has occurred in the \( \vee \) segments under consideration, the decoder must make a fresh start. It simply allows \( \vee \) segments of the next message section to enter the decoder, then utilizes the basic starting procedure discussed in the previous paragraphs.
3-6. **Average Number of Decoding Operations**

The problem of estimating the number of decoding operations is complicated, and exact analysis is not feasible. Fundamentally, we are forced to resort to experiment to determine requirements. Some crude estimates based on the binary symmetric channel model are useful, however.

We may approach the problem by estimating the number of operations required to make a kb-unit search, and then determine the probabilities (or bounds on the probabilities) of making searches of various kinds. We shall consider one decoding operation to be the generation of one branch or segment of the decoded sequence in the decoder plus applying the threshold test to the appropriate path.

The maximum number of operations in a kb-unit search is the number of branches in such a unit. This is given by

\[ 2 + 2^2 + 2^3 + \ldots + 2^{kb} = 2^{kb+1} - 2 \leq 2^{kb+1} = N_k \]

If branch-by-branch choice on the basis of a b-unit search goes through, then there are \((k-1)b + 1\) repetitions of the b-unit search. An upper bound on the number of decoding operations for this process is

\[ n_k = [(k-1)b + 1]2^{b+1} \]

If we let \(B_k\) be the event of a kb-unit search, then occurrence of the event \(B_k\) implies that channel noise is greater than \(T[(k-1)b]\) in the last \((k-1)b\) segments of the path. We have as two estimates of the average number of operations due to kb-unit searches the values \(n_k P(B_k)\) and \(N_k P(B_k)\). In the table following we list values obtainable for a code with \(n_o = 3\), \(b = 7\), and binary symmetric channel transition probability \(p_o = 0.02\).
ESTIMATES OF AVERAGE NUMBER OF DECODING OPERATIONS CONTRIBUTED BY kb-UNIT SEARCHES

\( b = 7 \quad n_o = 3 \quad \text{BSC with} \quad p_o = 0.02 \)

<table>
<thead>
<tr>
<th>k</th>
<th>( \text{kb} )</th>
<th>( n_k )</th>
<th>( N_k )</th>
<th>( P(B_k) )</th>
<th>( n_k' P(B_k) )</th>
<th>( N_k P(B_k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>256</td>
<td>2.56 ( \times 10^2 )</td>
<td>1.18 ( \times 10^{-3} )</td>
<td>0.302</td>
<td>0.302</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>2048</td>
<td>3.28 ( \times 10^4 )</td>
<td>4.98 ( \times 10^{-5} )</td>
<td>0.104</td>
<td>1.62</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>3840</td>
<td>4.19 ( \times 10^6 )</td>
<td>1.87 ( \times 10^{-5} )</td>
<td>0.072</td>
<td>78.3</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>5632</td>
<td>5.37 ( \times 10^8 )</td>
<td>4.57 ( \times 10^{-6} )</td>
<td>0.026</td>
<td>2450</td>
</tr>
</tbody>
</table>

It should be apparent that the estimate \( N_k P(B_k) \) is extremely pessimistic, for it assumes that any time a kb-unit search is carried out every branch in the unit must be generated. This is obviously unrealistic. One really should not expect to have anything approaching a full kb-unit generated unless, in fact, the threshold \( T(kb) \) is exceeded. For \( k \geq 3 \), it is extremely unlikely that more than a series of \( (b+2) \)-unit searches should be carried out to find a path through the kb-unit, provided the threshold \( T(kb) \) is not exceeded. Since each \( (b+2) \)-unit search results in the acceptance of 3 segments of the path, we may estimate the number of operations by

\[
 n_k' = \frac{1}{3} [(k-1)b-2]^{b+3}
\]

The average number of operations required by kb-unit searches is then estimated by

\[
 N^* = n_k' P(B_k) + N_k P(B_{k+1})
\]

For the system described above, typical values are:

ALTERNATE ESTIMATE OF THE AVERAGE NUMBER OF DECODING OPERATIONS CONTRIBUTED BY kb-UNIT SEARCHES

\( b = 7 \quad n_o = 3 \quad \text{BSC with} \quad p_o = 0.02 \)

<table>
<thead>
<tr>
<th>k</th>
<th>( n_k' )</th>
<th>( N_k )</th>
<th>( n_k' P(B_k) )</th>
<th>( N_k P(B_{k+1}) )</th>
<th>( N^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4.10 ( \times 10^3 )</td>
<td>4.19 ( \times 10^6 )</td>
<td>0.07</td>
<td>2.08</td>
<td>2.15</td>
</tr>
<tr>
<td>4</td>
<td>6.50 ( \times 10^3 )</td>
<td>5.37 ( \times 10^8 )</td>
<td>0.0003</td>
<td>70.0</td>
<td>70.0</td>
</tr>
</tbody>
</table>
3-7. Strategies for Repeat or Store and Reprocess

The figures in the previous section indicate that if the noise is not too severe, the average amount of decoding operation does not increase very markedly over that required for minimal noise. However, there is a considerable range in the numbers of decoding operations required for various digits. And the number rises to quite large values under severe noise disturbance. This means that if the decoder is designed to handle the most severe cases within the error-correcting capability of a long code, it will be operating for a very large fraction of the time at far less than its capacity.

A variety of attackes may be made on this problem, depending upon the operating situation. In the case a return channel is available for a repeat request, the decoder may simply request a repeat of a section of signal whenever its decoding capabilities are taxed. In such a case, the full error-correcting capability of the code is not used for actual error correction. It is used, however, to provide high reliability for the threshold tests.

If noise conditions are not too great and no return channel is available, the system may be designed to handle the fluctuations in numbers of decoding operations by providing sufficient buffer storage -- say in the form of tape recording. This generally is not completely satisfactory, however, because almost any amount of storage is liable to overflow.

The sectionalizing of the message sequence and the associated restart procedure described earlier make possible another solution of the problem, in many situations. If the capability of the decoder is overrun, it is possible to dump a section of the received signal into storage for
reprocessing at a later time -- perhaps by a larger decoding machine. Since the sectionalization described earlier achieves the condition of starting at the same place in the tree for the beginning of each new section, the reprocessing is greatly simplified. This solution is particularly applicable to the case of a data center serving many communication links. A single large, fast decoder could reprocess the troublesome sections for several channels. Each channel would have a simpler decoder for on-line operation. When the number of decoding operations per segment exceeds a certain design level, the section is stored for reprocessing, and the on-line decoder proceeds with the next message section. Relatively simple switching operations can provide the necessary flexibility.

Another advantage of the store and reprocess procedure is that it may allow the recovery of the transmitted signal even in some cases in which the channel noise exceeds the threshold. The threshold is based on minimum distances; many paths are at considerably greater distance from the opposite half of the \( \nu \)-unit, so that they could be recovered by a maximum-likelihood decoding procedure. The decoding operation is simply carried out with relaxed (larger) threshold values and maximum-likelihood (minimum-distance) paths are chosen. In general, the number of operations will be high, and such procedures are justified only when the section of message processed may have particularly valuable data. Such might be the case for scientific data telemetered from a remote station, under conditions for which repeat strategies are not feasible.
3-8. Adaptive Operation

There is a sense in which the basic decoding scheme described in this paper could be considered "adaptive". Use of the threshold tests enables the decoder to adjust its operation to the noise conditions in such a way that reasonably efficient operation is carried out for each noise situation. If a return link is available, the adaptation could be carried a step further. When the threshold tests and the numbers of decoding operations continue to indicate high noise conditions, the decoder could call for a change in rate of transmission of digits or perhaps a change in the code to one of lower rate. Similarly, if low noise conditions are indicated, the rate of transmission could be adjusted upward.
3-9. **Synchronization**

As pointed out earlier, the crucial task of synchronizing the decoder operation with the received signal has been considered a technical one not dealt with directly in the encoding-decoding process. The use of the restart sequence to sectionalize the message sequence suggests strongly that problems of synchronization should be considered in the design of this restart sequence. In so far as the restart operation is concerned, all that is necessary is that the restart sequence be a sufficiently long \((\nu - 1\) or more message digits), known sequence. Thus, the designer is free to design a restart sequence which provides efficient synchronization checks and resynchronization procedures.
3-10. Implementation

This discussion has been programmatic, in the sense that it has described basic strategies and procedures which must be implemented in terms of actual hardware before they are meaningful. Thus it points to a program of development and empirical testing of alternative procedures. It is quite clear that the operations called for are digital operations closely related to those utilized in ordinary digital computers. Some work toward programming the system on a general purpose computer has been initiated at Rice University by H. Lyne [11]. Certain features of the available machine have made it particularly suitable for this system, but at the expense of somewhat less efficient programming the same results could be obtained on commercially available general purpose computers.

Because of the unusually long word length desirable -- probably 100 to 150 binary digits -- and the somewhat special nature of some of the operations, it is anticipated that decoding could be done more efficiently by special purpose computers designed specifically for the task. Some existing special purpose machines may well be adapted to the task by slight modifications.
Fig. 3-1. Tree representation of a convolutional code.
Fig. 3-2. A 2-unit in the code of Fig. 1.
Fig. 3-3. Threshold values and partial paths used in the multiple-threshold test.
Fig. 3-4. Representation of the b-unit search with discard. Symbol x indicates the point at which the threshold is exceeded.
Fig. 3-5. Branch-by-branch choice based on the b-unit search.
Fig. 3-6. Larger unit search when the b-unit search fails.
CHAPTER 4. A NEW ANALYSIS FOR WOZENCRAFT'S SEQUENTIAL DECODING SCHEME

In this chapter, we introduce the concepts of threshold block decoding and an easy way to evaluate reasonable thresholds. In a modified manner, these thresholds are applied to Wozencraft's sequential decoding scheme; and we are able to show that the average number of computations required to process the incorrect subset is bounded by a quantity which is independent of the constraint length $n_t$ for $R_t < R_{\text{comp}}$. We also show that the probability of error decreases exponentially with $n_t$. The results obtained in this chapter are different from the results obtained by Wozencraft and Reiffen [2]. All the analysis in this chapter is based on random coding arguments. Another purpose of this chapter is to give a comparison between the decoding scheme presented in Chapter 3 and Wozencraft's scheme.
4-1. Threshold Block Decoding

Let \( \tilde{v} \) be the received sequence. In maximum-likelihood decoding, the decoder will identify the codeword \( \tilde{u}_k \) as the transmitted sequence if the conditional probability \( p_{V|U}(\tilde{v}|\tilde{u}_k) \) is the largest. In the case that the channel noise is not severe, we will have the following condition

\[
p_{V|U}(\tilde{v}|\tilde{u}_k) \geq \bigwedge_{i \in I \atop i \neq k} p_{V|U}(\tilde{v}|\tilde{u}_i)
\]

(4-1)

where \( I \) is the index set \( I = \{i: \tilde{u}_i \in S\} \), and \( S \) is the code used.

Let \( N(S) = 2^R \) be the total number of codewords in the code \( S \).

In the following analysis, we assume that all the channel inputs (or codewords) are equiprobable, i.e.,

\[
p_{U}(\tilde{u}_i) = \frac{1}{N(S)} = 2^{-R} \quad \text{for } i \in I
\]

(4-2)

Adding both sides of Eq. 4-1 by \( p_{V|U}(\tilde{v}|\tilde{u}_k) \), we obtain

\[
2p_{V|U}(\tilde{v}|\tilde{u}_k) \geq \bigwedge_{i \in I} p_{V|U}(\tilde{v}|\tilde{u}_i)
\]

(4-3)

Since

\[
p_{V}(\tilde{v}) = \bigwedge_{i \in I} p_{V|U}(\tilde{v}|\tilde{u}_i) p_{U}(\tilde{u}_i)
\]

(4-4)

we have

\[
p_{V}(\tilde{v}) = 2^{-R} \bigwedge_{i \in I} p_{V|U}(\tilde{v}|\tilde{u}_i)
\]

(4-5)

so that

\[
\bigwedge_{i \in I} p_{V|U}(\tilde{v}|\tilde{u}_i) = 2^{R} p_{V}(\tilde{v})
\]

(4-6)

Substituting Eq. 4-6 into Eq. 4-3, we have

\[
p_{V|U}(\tilde{v}|\tilde{u}_k) \geq 2^{-R} p_{V}(\tilde{v})
\]

(4-7)

Or

\[
\frac{p_{V|U}(\tilde{v}|\tilde{u}_k)}{p_{V}(\tilde{v})} \geq 2^{R}
\]

(4-8)
Taking logarithms on both sides of Eq. 4-8, we finally obtain

$$\log_2 \frac{P_{V|U}(\bar{v}|\bar{u}_k)}{P_V(\bar{v})} \geq nR_t - 1$$

(4-9)

The left side of Eq. 4-9 is the mutual information between the codeword \(\bar{u}_k\) and the received sequence \(\bar{v}\). Thus, we have

$$I(\bar{u}_k : \bar{v}) \geq nR_t - 1$$

(4-10)

On the other hand, we have

$$I(\bar{u}_k : \bar{v}) \leq I(\bar{u}_k) = nR_t$$

(4-11)

The equality of Eq. 4-11 holds for \(P_{U|V}(\bar{u}_k|\bar{v}) = 1\) (i.e., when \(\bar{v}\) uniquely specifies \(\bar{u}_k\)).

By combining Eq. 4-10 and Eq. 4-11, we obtain the following relation

$$nR_t - 1 \leq I(\bar{u}_k : \bar{v}) \leq nR_t$$

(4-12)

Hence, when the largest conditional probability \(P_{V|U}(\bar{v}|\bar{u}_k)\) satisfies the condition in Eq. 4-1, the mutual information between the codeword \(\bar{u}_k\) and \(\bar{v}\) will satisfy the inequality of Eq. 4-12.

Therefore, it seems reasonable to take

$$I_T = nR_t - 1$$

(4-13)

as the appropriate threshold. In the decoding, the decoder searches for a codeword \(\bar{u}_k\) such that \(I(\bar{u}_k : \bar{v}) \geq I_T\), and then identifies \(\bar{u}_k\) as the transmitted sequence. An error occurs whenever Eq. 4-10 is satisfied for any codeword \(\bar{u}_i\) other than the correct one \(\bar{u}_k\). This event is equivalent to the event that there exists an incorrect codeword \(\bar{u}_i\) such that \(P_{V|U}(\bar{v}|\bar{u}_i)\) is the largest and condition in Eq. 4-1 is satisfied.

In case the channel noise is so severe that the condition stated in Eq. 4-1 does not hold, even if \(P_{V|U}(\bar{v}|\bar{u}_k)\) is the largest conditional probability, we can consider that there exists a highly probable subset \(S_H\) of codewords which contains \(\bar{u}_k\) and such that the following
inequality holds
\[ \sum_{j \in J} p_{V \mid U}(\vec{v} \mid \vec{u}_j) \geq \sum_{i \in I} p_{V \mid U}(\vec{v} \mid \vec{u}_i) \] (4-14)

where \( J = \{ j : \vec{u}_j \in S^*_H \} \) is the index set of \( S^*_H \).

It is important to indicate that \( S^*_H \) is the smallest subset of \( S \) such that Eq. 4-14 holds.

The inequality of Eq. 4-14 can be rewritten as
\[ 2 \sum_{j \in J} p_{V \mid U}(\vec{v} \mid \vec{u}_j) \geq \sum_{i \in I} p_{V \mid U}(\vec{v} \mid \vec{u}_i) \] (4-15)

Let \( N(S^*_H) = N_H \) be the number of codewords in \( S^*_H \).

Since \( \vec{u}_k \in S^*_H \) and \( p_{V \mid U}(\vec{v} \mid \vec{u}_k) \) is the largest conditional probability, we can replace each term in the left hand side of Eq. 4-15 by \( p_{V \mid U}(\vec{v} \mid \vec{u}_k) \); we then obtain
\[ 2N_H p_{V \mid U}(\vec{v} \mid \vec{u}_k) > \sum_{i \in I} p_{V \mid U}(\vec{v} \mid \vec{u}_i) \] (4-16)

Substituting Eq. 4-6 into Eq. 4-16, we obtain
\[ 2N_H p_{V \mid U}(\vec{v} \mid \vec{u}_k) \geq 2^{nR_t} p_{V}(\vec{v}) \] (4-17)

or
\[ \frac{p_{V \mid U}(\vec{v} \mid \vec{u}_k)}{p_{V}(\vec{v})} \geq \frac{2^{nR_t}}{2N_H} \] (4-18)

Taking logarithms on both sides of Eq. 4-18 yields
\[ \log_2 \frac{p_{V \mid U}(\vec{v} \mid \vec{u}_k)}{p_{V}(\vec{v})} \geq nR_t - \log_2 2N_H \] (4-19)

or
\[ \delta(\vec{u}_k; \vec{v}) \geq nR_t - \log_2 2N_H \] (4-20)

combining Eq. 4-11 and Eq. 4-20, we finally obtain
\[ nR_t - \log 2N_H \leq \delta(\vec{u}_k; \vec{v}) \leq nR_t \] (4-21)
Therefore, in threshold block decoding, the decoder starts from searching a codeword \( \tilde{u}_k \) such that \( \delta(\tilde{u}_k; \tilde{v}) \geq nR_t - 1 \) and identifies \( \tilde{u}_k \) as the transmitted codeword. If the noise is so severe that there is no such \( \tilde{u}_k \) that \( \delta(\tilde{u}_k; \tilde{v}) \geq nR_t - 1 \), then the decoder reduces the threshold \( \delta_t = nR_t - 1 \) by a quantity \( \log_2 N_H \); and the decoder searches again to find a codeword \( \tilde{u}_k \) such that \( \delta(\tilde{u}_k; \tilde{v}) \geq nR_t - \log 2N_H \). Whenever \( \tilde{u}_k \) is found, the decoder will identify \( \tilde{u}_k \) as the transmitted codeword. If the decoder still fails to find a codeword \( \tilde{u}_k \) that \( \delta(\tilde{u}_k; \tilde{v}) \geq nR_t - \log 2N_H \), the threshold will be further reduced until a \( \tilde{u}_k \) is found such that \( \delta(\tilde{u}_k; \tilde{v}) \) satisfies the appropriate threshold. The choice of \( N_H \) depends on the judgment of the channel noise. An appropriate choice of \( N_H \) will reduce the number of calculations.

Let \( d_k \) be the number of places where \( \tilde{u}_k \) and \( \tilde{v} \) differ, which is sometimes called the Hamming distance between \( \tilde{u}_k \) and \( \tilde{v} \).

Since the binary symmetric channel is constant, it is possible to rewrite Eq. 4-7 as

\[
\begin{align*}
d_k &\leq \frac{n-d_k}{nR_t - 1} \\
p_o q_o &\geq 2^p_{uv} \\
(4-22)
\end{align*}
\]

where \( p_o \) is the transition probability of the binary symmetric channel.

Taking logarithms on both sides, we obtain

\[
(4-23)
\]

\[
d_k \log p_o + (n-d_k) \log q_o \geq (nR_t - 1) + \log p_{uv} \]

From Eq. 4-23, we then have

\[
(4-24)
\]

\[
d_k \leq \frac{1}{\log p_o} \left[ 1 - n(R_t - \log q_o) - \log p_{uv} \right]
\]

Let

\[
(4-25)
\]

\[
\delta = \frac{1}{\log p_o} \left[ 1 - n(R_t - \log q_o) - \log p_{uv} \right]
\]

which we will call the distance threshold.
Therefore, in threshold block decoding, the searching for a codeword \( \tilde{u}_k \) such that \( d(\tilde{u}_k;\tilde{v}) \geq nR - 1 \) is equivalent to searching for a codeword \( \tilde{u}_k \) such that the Hamming distance \( d_k = d(\tilde{u}_k, \tilde{v}) \) between \( \tilde{u}_k \) and \( \tilde{v} \) is less than \( \delta \).

In case that the noise is severe, \( \delta \) will be determined by Eq. 4-17. Thus, we obtain

\[
\delta = \frac{1}{\log_{q_o} \left[ \log_2 2N_H - n(R - log q_o) - \log_2 P_V(\tilde{v}) \right]}
\]  

(4-26)

If random coding is used, then

\[ P_V(\tilde{v}) = 2^{-n} \]  

(4-27)

and Eq. 4-26 becomes

\[
\delta_r = \frac{n}{\log_{q_o} \left[ 1 - R + log q_o \right]} + \frac{1}{\log_{q_o} P_0} \log 2N_H
\]  

(4-28)

Eq. 4-28 can be rewritten as

\[
\delta_r = n \left\{ \tau(C - R) + P_o \right\} + \tau \log 2N_H
\]  

(4-29)

where \( C \) is the **channel capacity** and

\[
\tau = \frac{1}{\log_{q_o} P_0}
\]  

(4-30)

Therefore, in random coding, the decoder starts by searching for a codeword \( \tilde{u}_k \) in \( S \) such that the Hamming distance \( d(\tilde{u}_k;\tilde{v}) \) between \( \tilde{u}_k \) and \( \tilde{v} \) is less than \( \delta_r = n[\tau(C - R) + P_o] + \tau \); and then identifies \( \tilde{u}_k \) as the transmitted sequence. If the noise is so severe that no such \( \tilde{u}_k \) with \( d(\tilde{u}_k;\tilde{v}) < n \{ \tau(C - R) + P_o \} + \tau \), then the distance threshold \( \delta_r \) is relaxed by a quantity \( \tau \log 2N_H \). The decoder searches again to find a codeword \( \tilde{u}_k \) such that \( d(\tilde{u}_k;\tilde{v}) \) is less than the relaxed threshold. If the decoder still fails to find a \( \tilde{u}_k \) satisfying the relaxed threshold, \( \delta_r \) will be further relaxed by increasing \( N_H \) until a \( \tilde{u}_k \) is found such that \( d(\tilde{u}_k;\tilde{v}) \) satisfies the appropriate threshold.
Actually, the threshold block decoding provides no advantage over
the usual block decoding in terms of the probability of error or in
terms of the number of computations. But the above analysis provides
reasonable thresholds for Wozencraft's sequential decoding scheme.
4-2. **Sequential Decoding (by Wozencraft).**

**Basic Concept:**

In Sec. 4-1, we indicate that it is always possible to find a highly probably subset $S_H$ of codewords for a given received sequence $\tilde{v}$ such that

$$\sum_{j \in J} p_{v|u}(\tilde{v} | \tilde{u}_j) \geq \sum_{i \in I \& i \neq j} p_{v|u}(\tilde{v} | \tilde{u}_i)$$  \hspace{1cm} (4-31)

where $J$ and $I$ are the index sets $J = \{ j: \tilde{u}_j \in S_H \}$ and $I = \{ i: \tilde{u}_i \in S \}$ respectively.

In general, the size of $S_H$ is small. For small channel noise, we may even have

$$p_{v|u}(\tilde{v} | \tilde{u}_k) \geq \sum_{i \in I \& i \neq k} p_{v|u}(\tilde{v} | \tilde{u}_i)$$  \hspace{1cm} (4-32)

i.e., $S_H$ consists only a single codeword $\tilde{u}_k$.

Let $S_L$ be the set of codewords which are not in $S_H$. We will call $S_L$ as the **improbable subset** for given $\tilde{v}$, and all the codewords in $S_L$ as the **improbable codewords** for given $\tilde{v}$. Thus, we have

$$S = S_H \uplus S_I$$

where $\uplus$ indicates the disjoint union.

It is obvious that the Hamming distance $d_i = d(\tilde{u}_i, \tilde{v})$ between the received sequence $\tilde{v}$ and any codeword in the improbable subset $S_L$ is greater than the distance threshold

$$\delta = n[\tau(C-R_L) + p_o] + \tau \log 2N_H$$  \hspace{1cm} (4-33)

where $N_H = N(S_H)$ is the number of codewords in $S_H$.

In general, the size $N_H$ of the subset $S_H$ is small. For some class of codes with specific structure, even though it may be difficult to find the codeword $\tilde{u}_k$ such that $d(\tilde{u}_k, \tilde{v})$ is smallest (i.e., $p_{v|u}(\tilde{v} | \tilde{u}_k)$
is the largest), it should be relatively easy to eliminate the improbable subset \( S_L \) from detailed consideration. Therefore, concentration is not on selecting the single \( \tilde{u}_k \) such that \( d_k = d(\tilde{u}_k, \tilde{v}) \) is the smallest, but rather on discarding all the improbable codewords \( \tilde{u}_i \) in \( S_L \) such that \( d_i > \delta \). This discarding strategy is most effective when convolutional tree codes are used. Wozencraft's sequential decoding scheme for tree codes is based on this discarding concept.

Let us consider the tree code with constraint length \( n_t \) and transmission rate \( R_t = 1/n_o \). Because of the tree structure, the code has \( 2^{nR_t} \) possible paths at length \( n \), where \( l \leq n \leq n_t \) and \( 1 \leq A \leq 2^{(n_o-1)R_t} \). \( A \) is equal to one when \( n \) is divisible by \( n_o \).

For convenience of discussion, we let \( S(n) \) be the tree code at length \( n \).

Wozencraft's decoding strategy can be stated as follows. First of all, a distance threshold

\[
\delta(n) = n \left( C - R_t \right) + p_o + \tau \log 2N_H
\]

is set at any \( n \), where \( l \leq n \leq n_t \). The decoder starts to generate the paths of the tree sequentially. As the decoder traces a path \( \tilde{u}_i(n) \) in the tree, it counts the number of disagreement \( d_i(n) \) between the path \( \tilde{u}_i(n) \) and the received sequence \( \tilde{v}(n) \) in length \( n \). If \( d_i(n) < \delta(n) \), the decoder follows \( \tilde{u}_i(n) \) from \( n \) to \( n+1 \); if \( d_i(n) > \delta(n) \), the decoder discards \( \tilde{u}_i(n) \) at \( n \) and traces the next path of the tree. When the decoder discards any path \( \tilde{u}_i(n) \) at length \( n \), it discards a subset of paths which have \( \tilde{u}_i(n) \) as the prefix. The size of this subset is \( 2^{(n_t-n)R_t} \). Therefore, we would like to discard all the improbable paths at small \( n \). By this discarding process, the decoder finally eliminates
all the improbable paths and only a small number of paths in the tree are retained through \( n_t \). Then the decoder will make a decoding on the basis of the retained subset of paths.

**Decoding Algorithm:**

For simplicity, we let the transmission rate \( R_t = 1/n_o \). As we indicated in Chapter 2, the tree code can be partitioned into two subsets; one subset \( S_o \) corresponds to having the first message digit \( m_1 = 0 \), the other subset \( S_1 \) corresponds to having \( m_1 = 1 \). For convenience, we define the subset \( S_o \) or \( S_1 \) which contains the transmitted path as the **correct subset**, and all the paths in it will be referred as the **correct paths**. The subset which does not contain the transmitted path will be referred to as the **incorrect subset**. The task of the decoder is to decide which subset (\( S_o \) or \( S_1 \)) is the correct subset; the decoder then decodes the first message digit \( m_1 \) accordingly.

For convenience of the following analysis, we rewrite the general form of distance threshold at \( n \)

\[
\delta(n) = n\{\tau(C-R_c) + p_o\} + \tau \log 2N_H
\]  

(4-34)

into the form

\[
\delta^j(n) = n\{\tau(C-R_c) + p_o\} + \tau(j-1)\alpha
\]  

(4-35)

where \( \log 2N_H = (j-1)\alpha \), \( j \) is an integer, and \( \alpha \) is a constant.

Now let us define a sequence of distance criteria \( \delta^1(n), \delta^2(n), \ldots, \delta^j(n), \ldots \) at \( n \) where

\[
\delta^1(n) = n\{\tau(C-R_c) + p_o\}
\]
\[
\delta^2(n) = n\{\tau(C-R_c) + p_o\} + \tau\alpha
\]
\[
\delta^3(n) = n\{\tau(C-R_c) + p_o\} + 2\tau\alpha
\]  

(4-36)

\[
\delta^j(n) = n\{\tau(C-R_c) + p_o\} + \tau(j-1)\alpha
\]
Then Wozencraft's sequential decoding algorithm can be stated as follows:

(i) The decoder starts with the smallest distance criterion $\delta^1(n)$, and begins to generate sequentially the paths of the tree $S$. As the decoder proceeds, it discards any path of length that differs from the received sequence $\bar{v}(n)$ in $\delta^1(n)$ or more places.

(ii) As soon as the decoder discovers any path in $S$ that is retained through the constraint length $n_c$, it prints out the corresponding first message digit.

(iii) If the entire tree $S$ is discarded the decoder adopts the next larger distance criterion $\delta^2(n)$, and continues the same searching process until some path in $S$ is retained through $n_c$. Then the decoder prints the corresponding first message digit $m_1$.

(iv) The decoder continues this search process until some path in $S$ is retained through $n_c$ for an appropriate distance criterion $\delta^j(n)$.

By these rules, the decoder never uses the $j$th distance criterion $\delta^j(n)$ unless the entire $S$ is discarded for $(j-1)$th distance criterion $\delta^{j-1}(n)$. The probability that the entire tree $S$ is discarded depends only on the channel noise. Only when the channel is especially noisy will we require a larger distance criterion before accepting some path in $S$. In general we will be able to discover a sequence in $S$ retained through $n_c$ with a small distance criterion.
Average Number of Computations in Processing the Incorrect Subset:

The following analysis is based on the random coding argument. We assume that each binary digit in the entire tree code is selected from the binary field, independently at random, with \( p(0) = p(1) = \frac{1}{2} \). Thus, the probability that the received sequence \( \tilde{v} \) and any incorrect input path agree in any position is \( \frac{1}{2} \) (See Appendix B). It can be shown that this could be accomplished by selecting the code generator sequence equiprobably at random from the ensemble of \( 2^{n_t} \) possible \( n_t \)-tuples.

For convenience, we assume \( S_0 \) is the correct subset (i.e., \( S_0 \) contains the transmitted path).

For the purpose of the following discussion, we introduce some random processes as follows

\[ U(\cdot, n), \quad V(\cdot, n), \quad E(\cdot, n) \quad \text{and} \quad D(\cdot, n) \]

are such that for any particular \( \xi \), we have

\[ U(\xi, n) = \tilde{u}(n) \quad \text{the input path at length} \quad n, \]
\[ V(\xi, n) = \tilde{v}(n) \quad \text{the received sequence at length} \quad n, \]
\[ E(\xi, n) = \tilde{e}(n) \quad \text{the channel noise sequence at length} \quad n \]
\[ D(\xi, n) = d(n) \quad \text{the weight of channel noise sequence at length} \quad n \]

where \( E(\xi, n) = U(\xi, n) \oplus V(\xi, n) \) and \( D(\xi, n) = |E(\xi, n)| \).

Now, we are interested in the probability that any incorrect path \( \tilde{u}_1(n) \) in \( S_1 \) (incorrect subset) will be retained at length \( n \).

Let \( E_1(\xi, n) = V(\xi, n) \oplus \tilde{u}_1(n) \) and \( D_1(\xi, n) = |E_1(\xi, n)| \). Then we have \( P(E_1^k = 1) = P(E_1^k = 0) = \frac{1}{2} \), where \( E_1^k \) is the kth component of \( E_1(\xi, n) \), or \( E_1^k(\xi) = V^k(\xi) \oplus u_1^k \) (see Appendix B).

Let us define \( \pi^j(n) \) to be the event that any particular incorrect path \( \tilde{u}_1(n) \) will be retained at length \( n \) when the jth distance criterion \( \delta^j(n) \) is used in the searching. Then we have
\[ n^j(n) = \bigcap_{t=1}^{n} \{ \xi : D_1(\xi, t) \leq \delta^j(t) \} \]  

(4-37)

where \( \{ \xi : D_1(\xi, t) : \delta^j(t) \} \) is the event that the distance between any incorrect path \( \bar{u}_1(n) \) and the received sequence \( \bar{v}(n) \) is less than or equal to the distance threshold \( \delta^j(t) \) at length \( t \).

Certainly, we may say

\[ \pi^j(n) \subset \{ \xi : D_1(\xi, n) \leq \delta^j(n) \} \]  

(4-38)

and

\[ P[\pi^j(n)] \leq P(\{ D_1(\xi, n) \leq \delta^j(n) \}) \]  

(4-39)

Set \( \delta^j(n) = np^j_n \), then

\[ p^j_n = (\tau(C-R_t^c) + p_0) + \frac{1}{n} \tau(j-1)\alpha \]  

(4-40)

which is a function of \( n \) and \( p_0 \).

By use of the Chernoff bound, we obtain

\[ P[\pi^j(n)] \leq P[D_1(\xi, n) \leq \delta^j(n)] \]

\[ = P[D_1(\xi, n) \leq np^j_n] \]

\[ \leq 2^{-n[1-H(p^j_n)]} \quad \text{for } p^j_n \leq \frac{1}{2} \]  

(4-41)

A sufficient accurate bound on \( P[\pi^j(n)] \) for \( p^j_n > \frac{1}{2} \) is unity.

From Eq. 4-40, we see that there is a largest value of \( n \), which we call \( n_1 \), such that

\[ p^j_{n_1+1} < \frac{1}{2} \leq p^j_{n_1} \]  

(4-42)

or

\[ \tau(C-R_t^c) + p_0 + \frac{\tau}{n_1+1} (j-1)\alpha < \frac{1}{2} \leq \tau(C-R_t^c) + p_0 + \frac{\tau}{n_1} (j-1)\alpha \]  

(4-43)

Solving Eq. 4-43 for \( n_1 \), we obtain

\[ n_1 \leq \frac{\tau(j-1)\alpha}{\frac{1}{2} - \tau(C-R_t^c) - p_0} < n_1+1 \]  

(4-44)

Thus

\[ P[\pi^j(n)] \leq \begin{cases} 1 & n \leq n_1 \\ 2^{-n[1-H(p^j_n)]} & n > n_1 \end{cases} \]  

(4-45)

For any length \( n \), there are \( N(S_1) = \frac{1}{2} A^2 n^R t \) possible incorrect paths, and for each of these the probability of retentions is \( P[\pi^j(n)] \).
Assuming no previous discarding, therefore, the average number $M_j(n)$ of incorrect paths retained at length $n$ is bounded by

$$M_j(n) \leq N(S_1) \mathbb{P}[\pi^j(n)]$$

$$\leq \begin{cases} \frac{1}{2} A_2 n R_t & n \leq n_1 \\ \frac{1}{2} A_2 n [H(p_n^j) - H(p_t)] & n > n_1 \end{cases}$$ \hspace{1cm} (4-46)$$

where $R_t = 1 - H(p_t)$.

Assume that the decoder is at $n$th digit of a path $\tilde{u}_j$; we define "one computation" as the process of moving the decoder from the $n$th digit to the $(n+1)$th digit and comparing $d_j(n+1)$ against $\delta_j(n+1)$.

Let $N_j(n+1)$ be the average number of computations at length $n+1$ necessary to check all the paths in the incorrect subset $S_1$ which have been accepted out to length $n$. Thus

$$N_j(n+1) = \Delta(n) M_j(n)$$ \hspace{1cm} (4-47)$$

where $\Delta(n) = 1$ or 2 depending on whether or not $n$ is divisible by $n_o$. If $N_j$ is the average number of computations required to process the incorrect subset $S_1$ for the $j$th distance criterion, we have

$$N_j = \bigcup_{n=1}^{n_t} N_j(n)$$ \hspace{1cm} (4-48)$$

Set $\Delta(n) = 2$, then

$$N_j \leq A \bigcup_{n=1}^{n_1} n R_t + A \bigcup_{n=n_1+1}^{n_t} 2 n [H(p_n^j) - H(p_t)]$$ \hspace{1cm} (4-49)$$

In order to evaluate the second summation of Eq. 4-49, we will make the following approximation. Consider the binary entropy curve $H(p)$.

We construct two tangents $T_o(p)$ and $T_c(p)$ to the $H(p)$ curve at
\( p = p_0 \) and \( p = p_c \) respectively, where \( p_c \) is defined as

\[
\frac{p_c}{q_c} = \sqrt{\frac{p_0}{q_0}} \quad (4-50)
\]

So

\[
T_0(p) = -p \log p_0 - q \log q_0 \quad (4-51)
\]

\[
T_c(p) = -p \log p_c - q \log q_c \quad (4-52)
\]

Now let us define \( p_2 \) by

\[
T_c(p_2) = 1. \quad (5-53)
\]

It is obvious that

\[
H(p_n^j) \leq \begin{cases} 
1 & \text{for } p_n^j > p_2 \\
T_c(p_n^j) & \text{for } p_n^j \leq p_2
\end{cases} \quad (4-54)
\]

Let us define \( n_2^j \) to be the largest value of \( n \) such that \( p_n^j \geq p_2 \) for \( n \leq n_2^j \) and \( p_n^j < p_2 \) for \( n > n_2^j \). Thus we have

\[
T_c(p_{n_2^j+1}^j) < T_c(p_2) = 1 \leq T_c(p_n^j) \quad (4-55)
\]

\( T_c(p_n^j) \) can be expressed as

\[
T_c(p_n^j) = \frac{1}{2} (1 - R_c) + \frac{1}{2n} (j-1) \alpha + \frac{1}{2} \log \frac{p_0}{p_c} \quad (4-56)
\]

If we define

\[
R_{\text{comp}} = 1 - \log \frac{p_0}{2} = 1 - \log \frac{q_0}{2} \quad (4-57)
\]

Eq. 4-56 can be written as

\[
T_c(p_n^j) = 1 - \frac{1}{2} (R_{\text{comp}} + R_c) + \frac{1}{2n} (j-1) \alpha \quad (4-58)
\]

Substituting Eq. 4-58 into Eq. 4-55, we obtain

\[
1 - \frac{1}{2} (R_{\text{comp}} + R_c) + \frac{(j-1) \alpha}{2(n_2^j+1)} \alpha < 1 \leq 1 - \frac{1}{2} (R_{\text{comp}} + R_c) + \frac{(j-1) \alpha}{2n_2^j} \quad (4-59)
\]
For Eq. 4-59, we thus have
\[ n_2^j \leq \frac{(j-1)\alpha}{R_{\text{comp}} + R_t} < n_2^j + 1 \] (4-60)

Therefore, Eq. 4-54 can be rewritten as
\[ H(p_n^j) \leq \begin{cases} 
1 & \text{for } n < n_2^j \\
T_c(p_n^j) & \text{for } n \geq n_2^j
\end{cases} \] (4-61)

Substituting Eq. 4-61 into Eq. 4-49, we obtain
\[ N_j \leq A \sum_{n=1}^{n_2^j+1} \left( \frac{n^j}{2} n R_t - A \frac{n_t}{2} R_{\text{comp}} + \frac{1}{2} \left( \ln n - \ln n_2^j \right) \right) \] (4-62)

Since
\[ T_c(p_n^j) - H(p_t^j) = -\frac{1}{2} \frac{(R_{\text{comp}} - R_t)}{R_{\text{comp}}} + \frac{1}{2n} (j-1)\alpha \] (4-63)

then
\[ N_j \leq A \sum_{n=1}^{n_2^j+1} \left( \frac{n^j}{2} n R_t - A \frac{n_t}{2} R_{\text{comp}} + \frac{1}{2} (j-1)\alpha \right) \] (4-64)

The first summation can be bounded as follows
\[ \sum_{n=1}^{n_2^j} \left( \frac{n^j}{2} n R_t < \frac{1}{2} \frac{n^j R_t}{1 - 2 - R_t} \right) \] (4-65)

By Eq. 4-60, we have
\[ n_2^j < \frac{R_t}{1 - 2 - R_t} \frac{1}{R + R_{\text{comp}}} (j-1)\alpha \] (4-66)

For \( R_t < R_{\text{comp}} \), the second summation can be bounded by
\[
\frac{1}{2}(j-1) \alpha \leq \frac{n_t}{2} - \frac{1}{2} n [R_{\text{comp}} - R_t] \\
\leq n = n_2^{j+1} - \frac{1}{2} (n_2^2 + 1) (R_{\text{comp}} - R_t) \\
< 2^{\frac{1}{2}}(j-1) \alpha \leq 2 - \frac{1}{2} (R_{\text{comp}} - R_t) \\
1 - 2 \frac{1}{2} (R_{\text{comp}} - R_t) \\
(4-67)
\]

By Eq. 4-60, we have
\[
\frac{1}{2}(j-1) \alpha \leq \frac{n_t}{2} - \frac{1}{2} n [R_{\text{comp}} - R_t] \\
\leq n = n_2^{j+1} - \frac{R_{\text{comp}} - R_t}{2(R_{\text{comp}} + R_t)} (j-1) \alpha \\
< 2^{\frac{1}{2}}(j-1) \alpha \leq 2 - \frac{1}{2} (R_{\text{comp}} - R_t) \\
1 - 2 \frac{R_{t}}{R_{\text{comp}} + R_t} \\
(4-68)
\]

Substituting Eq. 4-66 and Eq. 4-68 into Eq. 4-64, we finally obtain
\[
\bar{N}_j \leq A_1 2^{\frac{1}{2} (j-1) \alpha} \quad \text{for} \quad R_t < R_{\text{comp}} \\
(4-69)
\]

where
\[
A_1 = A \left( \frac{1}{1 - R_t} + \frac{1}{1 - 2 \frac{1}{2}(R_{\text{comp}} - R_t)} \right) \\
(4-70)
\]

Up to now, we have obtained a bound on the average number of computations \(N_j\) required to process the incorrect subset when \(j\)th distance criterion is used.

Let us define \(C_j\) to be the event that the \(j\)th distance criterion is used. Then, we want to know what is the probability that \(\delta^j(n)\) will be used.
Assume \( \tilde{u}_o \) is the transmitted path and \( \tilde{u}_o \subseteq S_o \). We define the following events

\[
\begin{align*}
\omega_o &= \text{the event that } \tilde{u}_o \text{ will be discarded when the (j-1)th distance criterion } \delta^{j-1}(n) \text{ is used.} \\
\overline{\omega}_o &= \text{the event that the correct subset } S_o \text{ will be discarded when the (j-1)th distance criterion } \delta^{j-1}(n) \text{ is used.} \\
\Omega_o &= \text{the event that the entire tree will be discarded when } \delta^{j-1}(n) \text{ is used.}
\end{align*}
\]

Then, we will have the following relation

\[
\Omega_o \subseteq \overline{\omega}_o \subseteq \omega_o \tag{4-71}
\]

Let \( E_o(\xi, n) = V(\xi, n) \ominus \tilde{u}_o(n) \) be the actual noise sequence and \( d_o(n) = |E_o(\xi, n)| \) the weight of actual noise sequence of length \( n \).

We know that the transmitted path will be discarded at (j-1)th distance criterion if and only if \( d_o(n) \geq \delta^{j-1}(n) \) at any \( n \). Thus

\[
\begin{align*}
\omega_o &\subseteq \bigcup_{n=1}^{n_t} \{ \xi : D_o(\xi, n) \geq \delta^{j-1}(n) \} \\
\overline{\omega}_o &\subseteq \bigcup_{n=1}^{n_t} \{ d_o(n) \geq \delta^{j-1}(n) \} \\
\Omega_o &\subseteq \bigcup_{n=1}^{n_t} \{ d_o(n) \geq \delta^{j-1}(n) \}
\end{align*}
\]

And we may say

\[
\begin{align*}
P(\omega_o) &\leq \sum_{n=1}^{n_t} P[\bigcup_{n=1}^{n_t} \{ d_o(n) \geq \delta^{j-1}(n) \}] \\
P(\overline{\omega}_o) &\leq \sum_{n=1}^{n_t} P[d_o(n) \geq \delta^{j-1}(n)] \\
P(\Omega_o) &\leq \sum_{n=1}^{n_t} P[d_o(n) \geq \delta^{j-1}(n)]
\end{align*}
\]

By relation 4-71, we may say

\[
P(\Omega_o) \leq P(\overline{\omega}_o) \leq P(\omega_o)
\]

\[
P(\Omega_o) \leq P(\omega_o)
\]

\[
P(\Omega_o) \leq P(\overline{\omega}_o) \leq P(\omega_o)
\]
Since the decoder will never use the $j$th distance criterion unless the entire tree $S$ is discarded for $(j-1)$th distance criterion. Therefore, we obtain

$$P(C_j) = P(\Omega_o)$$
$$\leq P(W_o)$$
$$\leq P(w_o)$$

$$\frac{n_t}{\sum_{n=1}^{\delta_{j-1}(n)}}$$

Set $\delta_{j-1}(n) = n p_n^{j-1}$.

By use of Chernoff bound, we then obtain

$$P[d_o(n) \geq \delta_{j-1}(n)] = \sum_{k=n}^{n} C^n_k p_o^k q_o^{n-k} p_n^{j-1}$$

$$\leq 2^{-n[T_o(p_n^{j-1}) - H(p_n^{j-1})]}$$

(4-77)

and

$$P(C_j) \leq \sum_{n=1}^{\delta_{j-1}(n)} 2^{-n[T_o(p_n^{j-1}) - H(p_n^{j-1})]}$$

(4-78)

From Eq. 4-61 and Eq. 4-60, we have

$$H(p_n^{j-1}) \leq \begin{cases} 1 & \text{for } n < n_2^{j-1} \\ T_c(p_n^{j-1}) & \text{for } n \geq n_2^{j-1} \end{cases}$$

(4-79)

where

$$n_2^{j-1} \leq \frac{(j-2)c}{R_t + R_{\text{comp}}} \leq n_2^{j-1} + 1$$

(4-80)
Substituting Eq. 4-79 into Eq. 4-78 we obtain
\[ P(C_j) \leq \sum_{n=1}^{n_2^{j-1}} \frac{n}{2} - n \left[ T_o(p_n^{j-1}) - 1 \right] \]
\[ + \sum_{n=n_2^{j-1}+1}^{n_t} \frac{n}{2} - n \left[ T_o(p_n^{j-1}) - T_c(p_n^{j-1}) \right] \]  
(4-81)

\[ T_o(p_n^{j-1}) - 1 \] can be expressed as
\[ T_o(p_n^{j-1}) - 1 = \frac{1}{n} (j-2) \alpha - R_t \]  
(4-82)

and \[ T_o(p_n^{j-1}) - T_c(p_n^{j-1}) \] can be expressed as
\[ T_o(p_n^{j-1}) - T_c(p_n^{j-1}) = \frac{1}{2} (R_{comp} - R_t) + \frac{1}{2n} (j-2) \alpha \]  
(4-83)

Substituting Eq. 4-82 and Eq. 4-83 into Eq. 4-81
\[ P(C_j) \leq \sum_{n=1}^{n_2^{j-1}} \frac{n}{2} n R_t - (j-2) \alpha \]
\[ + \sum_{n=n_2^{j-1}+1}^{n_t} \frac{n}{2} - \frac{1}{2} n [R_{comp} - R_t] - \frac{1}{2} (j-2) \alpha \]
\[ - (j-2) \alpha \sum_{n=1}^{n_2^{j-1}} \frac{n}{2} n R_t + 2 - \frac{1}{2} (j-2) \alpha \sum_{n=n_2^{j-1}+1}^{n_t} - \frac{1}{2} n [R_{comp} - R_t] \]  
(4-84)

The first summation of Eq. 4-84 can be bounded as
\[ \sum_{n=1}^{n_2^{j-1}} \frac{n}{2} n R_t \leq 2 (j-2) \alpha \sum_{n=1}^{n_2^{j-1}R_t} \frac{1}{2} - R_t \]  
(4-85)
By use of Eq. 4-80, Eq. 4-85 becomes

\[
\frac{R_t}{R_{\text{comp}} + R_t} < \left( \frac{j-2}{n} \right) ^{n_R} \leq 2 \frac{R_t R_{\text{comp}}}{(j-2) \alpha R_t - R_{\text{comp}} + R_t} - \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} (j-2) \alpha 
\]

\[
\leq 2 \frac{-R_t}{1 - 2 R_t}
\]

For \( R_t < R_{\text{comp}} \), the second summation of Eq. 4-84 can be bounded as

\[
\frac{1}{2} \left( \sum_{n=n_2}^{n_1} \frac{1}{1 - 2 R_t} \right) \leq \frac{1}{2} \left( \frac{n_2^{j-1} + 1}{j-2} \right) \left( R_{\text{comp}} - R_t \right) \leq 2 \frac{\frac{1}{2} (n_2^{j-1} + 1)(R_{\text{comp}} - R_t)}{1 - 2} - \frac{1}{2} \left( R_{\text{comp}} - R_t \right)
\]

\[
\leq 2 \frac{\frac{1}{2} (n_2^{j-1} + 1)(R_{\text{comp}} - R_t)}{1 - 2} - \frac{1}{2} \left( R_{\text{comp}} - R_t \right)
\]

By use of Eq. 4-80 we obtain

\[
\frac{1}{2} \left( \sum_{n=n_2}^{n_1} \frac{1}{1 - 2 R_t} \right) \leq \frac{1}{2} \left( \frac{n_2^{j-1} + 1}{j-2} \right) \left( R_{\text{comp}} - R_t \right) \leq 2 \frac{\frac{1}{2} (n_2^{j-1} + 1)(R_{\text{comp}} - R_t)}{1 - 2} - \frac{1}{2} \left( R_{\text{comp}} - R_t \right)
\]

\[
\leq 2 \frac{\frac{1}{2} (n_2^{j-1} + 1)(R_{\text{comp}} - R_t)}{1 - 2} - \frac{1}{2} \left( R_{\text{comp}} - R_t \right)
\]

\[
\leq 2 \frac{\frac{1}{2} (n_2^{j-1} + 1)(R_{\text{comp}} - R_t)}{1 - 2} - \frac{1}{2} \left( R_{\text{comp}} - R_t \right)
\]
Substituting Eq. 4-86 and Eq. 4-88 into Eq. 4-84 we finally obtain

\[
P(C_j) \leq A_2 \frac{R_{comp}}{R_{comp} + R_t} (j-2) \alpha
\]  

(4-89)

where

\[
A_2 = \left(\frac{1}{1 - 2} - \frac{1}{R_t - R_{comp}} \right) \frac{1}{2} (R_{comp} - R_t)
\]  

(4-90)

Thus, we have

\[
P(C_j) \begin{cases} = 1 & \text{for } j = 1 \\ \leq A_2 \frac{R_{comp}}{R_{comp} + R_t} (j-2) \alpha & \text{for } j > 1 \end{cases}
\]  

(4-91)

Now, we let \( \bar{N} \) be the total average number of computations required to process the incorrect subset \( S_1 \). Then

\[
\bar{N} = \sum_j P(C_j) N_j = N_1 + \sum_{j=1}^{\infty} P(C_j) N_j
\]  

(4-92)

Substituting Eq. 4-69 and Eq. 4-91 into Eq. 4-92, it yields

\[
\bar{N} \leq A_1 + \sum_{j=2}^{\infty} A_2 \frac{R_{comp}}{R_{comp} + R_t} (j-2) \alpha
\]  

\[
\leq A_1 + A_1 A_2 \frac{R_{comp}}{R_{comp} + R_t} \frac{R_t}{R_{comp} + R_t} \frac{R_{comp}}{R_{comp} + R_t} (j-1) \alpha
\]  

(4-93)

For \( R_t < R_{comp} \)

\[
\bar{N} \leq A_1 + A_1 A_2 \frac{R_{comp}}{R_{comp} + R_t} (j-2) \alpha
\]  

\[
\leq A_1 + A_1 A_2 \frac{R_{comp}}{R_{comp} + R_t} \frac{R_t}{R_{comp} + R_t} \frac{R_{comp}}{R_{comp} + R_t} \frac{R_{comp}}{R_{comp} + R_t} \frac{R_t}{R_{comp} + R_t} (j-1) \alpha
\]  

(4-94)
Let \( g = \frac{R_t}{R_{\text{comp}}} \), Eq. 4-94 becomes

\[
\bar{N} \leq A_1 + A_2 \frac{2}{1+\beta} \frac{\alpha}{1-\beta} \frac{\beta}{1+\beta} \frac{1}{1-\beta} \frac{1}{\alpha} \quad (4-95)
\]

for \( R_t < R_{\text{comp}} \)

We desire to select \( \alpha \) to minimize the bound on \( \bar{N} \). Setting \( \frac{d\bar{N}}{d\alpha} = 0 \), we obtain

\[
\alpha = \frac{1+\beta}{1-\beta} \log \frac{1}{\beta} = \frac{\beta+1}{\beta-1} \log \beta \quad (4-96)
\]

Substituting Eq. 4-96 into Eq. 4-95, we obtain

\[
\bar{N} \leq A_1 + A_2 \frac{\beta}{1-\beta} \frac{\beta}{1-\beta} = A_1 \left( 1 + A_2 \frac{\beta}{1-\beta} \right) \quad \text{for} \ \beta < 1 \quad (4-97)
\]

Eq. 4-97 shows that the average number of computations \( \bar{N} \) required to process the incorrect subset is bounded by a quantity which is independent of the constraint length \( n_t \) for \( R_t < R_{\text{comp}} \). This result is different from the result obtained by Wozencraft and Reiffen [2]; in their results, \( \bar{N} \) is bounded by a quantity which is proportional to \( n_t^\beta \) for \( R_t < R_{\text{comp}} \).
Probability of Error

The first message digit \( m_1 \) is decoded as soon as the decoder discovers in the searching procedure a path \( \tilde{u} \) in the tree \( S \) such that the Hamming distance \( d(n) \) between \( \tilde{u}(n) \) and \( \tilde{v}(n) \) satisfies

\[
d(n) < \delta_i(n) \quad \text{for} \quad 1 \leq n \leq n_t \tag{4-98}
\]

An error occurs if and only if this path \( \tilde{u} \) is in the incorrect subset \( S_1 \).

Let \( I = \{ i : \tilde{u}_i \in S_1 \} \) be the index set for the incorrect subset \( S_1 \), and \( D_i(\xi, n_t) \) be the Hamming distance between any incorrect path \( \tilde{u}_i(n_t) \) and the received sequence of length \( n_t \).

Let us define \( \epsilon_j \) to be the event that an error occurs when the \( j \)th distance criterion is used. Then, we have

\[
\epsilon_j \subset \bigcup_{i \in I} \{ \xi : D_i(\xi, n_t) \leq \delta_i(n_t) \} \tag{4-99}
\]

where \( \bigcup_{i \in I} \{ \xi : D_i(\xi, n_t) \leq \delta_i(n_t) \} \) is the event that the Hamming distance \( D_i(\xi, n_t) \) between any incorrect path \( \tilde{u}_i(n_t) \) and the received sequence is less than or equal to \( \delta_i(n_t) \). We then may say

\[
P(\epsilon_j) \leq P\left[ \bigcup_{i \in I} \{ \xi : D_i(\xi, n_t) \leq \delta_i(n_t) \} \right] \leq \sum_{i \in I} P[\{ \xi : D_i(\xi, n_t) \leq \delta_i(n_t) \}] \leq N(S_1) P[D_i(\xi, n_t) \leq \delta_i(n_t)] \tag{4-100}
\]

where \( N(S_1) \) is the number of paths in the incorrect subset \( S_1 \).

Set \( \delta_i(n_t) = n_t p^j \). For random coding we have

\[
P[D_i(\xi, n_t) \leq n_t p^j] \leq 2^{-n_t[1-H(p^j)]} \quad \text{for} \quad p^j \leq \frac{1}{2} \tag{4-101}
\]
Substituting Eq. 4-101 into Eq. 4-100, we obtain

\[ P(\varepsilon_j) \leq \frac{1}{2} \frac{n_t R_t}{2} - n_t [1-H(p^j)] \]

\[ \leq \frac{1}{2} \frac{n_t [H(p^j) - H(p^j)]}{2} \quad \text{for} \quad p_t \geq p^j \quad (4-102) \]

There exists an integer \( j_c \) such that

\[ p^j \leq p_t < p^{j_c+1} \quad (4-103) \]

For \( p_t < p^j \), an appropriate upper bound for \( P(\varepsilon_j) \) is unity. Thus, we have

\[
\begin{align*}
P(\varepsilon_j) & \leq \\
&= \begin{cases} 
1 & \text{for } j > j_c \\
\frac{1}{2} \frac{n_t [H(p_t) - H(p^j)]}{2} & \text{for } j \leq j_c 
\end{cases} 
\end{align*} \quad (4-104) \]

From Eq. 4-103 we have

\[
\frac{j_c(n_t)}{n_t} \leq p_t < \frac{j_c+1}{n_t} \quad (4-105) \]

or

\[
\{\tau(C-R_t) + p_o\} + \frac{1}{n_t} \tau(j_c+1) \leq p_t < \{\tau(C-R_t) + p_o\} + \frac{1}{n_t} \tau j_c \quad (4-106) \]

Solving Eq. 4-106 for \( j_c \), we obtain

\[
\frac{p_t - p_o - \tau(C-R_t)}{\tau \alpha} n_t < j_c \leq 1 + \frac{p_t - p_o - \tau(C-R)}{\tau \alpha} n_t \quad (4-107) \]

Noting \( \tau = \frac{1}{\log \frac{q_o}{q_o}} \), Eq. 4-107 can be reduced to

\[
\frac{n_t}{\alpha} [T_o(p_t) - H(p^j)] \leq j_c \leq 1 + \frac{n_t}{\alpha} [T_o(p_t) - H(p^j)] \quad (4-108) \]

where \( T_o(p_t) = -p_t \log p_o - q_t \log q_o \)

Let us define \( \varepsilon \) to be the event of making an error in the decoding.
Then
\[ P(e) = \sum_{j} P(e_j) P(C_j) \] \hspace{1cm} (4-109)

Breaking the summation into two parts, we have
\[ \sum_{j=1}^{j_c} P(e_j) P(C_j) + \sum_{j>j_c} P(e_j) P(C_j) \] \hspace{1cm} (4-110)

Since \( P(e_j) \leq 1 \) for \( j > j_c \), then
\[ \sum_{j=1}^{j_c} P(C_j) + \sum_{j>j_c} P(C_j) \] \hspace{1cm} (4-111)

The second summation can be overbounded as follows:
\[ \sum_{j>j_c} P(C_j) \leq A_2^2 \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} (j-2) \alpha \hspace{1cm} (4-112) \]

From Eq. 4-107, we know
\[ j_c - 1 > \frac{n_t}{\alpha} [T_o(p_t) - H(p_t)] \] \hspace{1cm} (4-113)

Substituting Eq. 4-113 into Eq. 4-112, we have
\[ \sum_{j>j_c} P(C_j) \leq A_2^2 \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} \frac{n_t [T_o(p_t) - H(p_t)]}{\alpha} \] \hspace{1cm} (4-114)
Use $\alpha = \frac{3+1}{3-1} \log \beta$, where $\beta = \frac{R}{R_{\text{comp}}}$, Eq. 4-114 becomes

$$P(C_j) \leq A_2 \frac{1}{\beta^{1-\beta}} \frac{1}{1-\beta} \frac{1}{2} \frac{1}{2} \frac{n_t}{2} \left[H(p_t) - H(p_j^i)\right]$$

(4-115)

The first summation can be bounded as follows

$$\sum_{j=1}^{j_c} P(e_j) P(C_j) \leq A_2 \frac{1}{2} \frac{1}{2} \frac{1}{2} \frac{n_t}{2} \left[H(p_t) - H(p_j^i)\right] - \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} \frac{(j-2)\alpha}{(j-2)\alpha}$$

(4-116)

Noting $H(p_j^i) \leq T_c(p_j^i)$, then

$$\sum_{j=1}^{j_c} P(e_j) P(C_j) \leq A_2 \frac{1}{2} \frac{1}{2} \frac{1}{2} \frac{n_t}{2} \left[H(p_t) - T_c(p_j^i)\right] - \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} \frac{(j-2)\alpha}{(j-2)\alpha}$$

(4-117)

By Eq. 4-58, we have

$$H(p_t) - T_c(p_j^i) = \frac{1}{2} (R_{\text{comp}} - R_t) - \frac{1}{2} n_t (j-1)\alpha$$

(4-118)

Substituting Eq. 4-118 into Eq. 4-117, we have

$$\sum_{j=1}^{j_c} \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} \alpha \leq A_2 \frac{1}{2} \frac{1}{2} \frac{1}{2} \frac{n_t}{2} (R_{\text{comp}} - R_t) \frac{j_c}{2} - \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} \frac{(j-2)\alpha}{(j-2)\alpha}$$

$$\sum_{j=1}^{j_c} \frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} \alpha \leq A_2 \frac{1}{2} \frac{1}{2} \frac{1}{2} \frac{n_t}{2} (R_{\text{comp}} - R_t) \frac{j_c}{2} - \frac{R_{\text{comp}} - R_t}{2(R_{\text{comp}} + R_t)} (j-1)\alpha$$

(4-119)
\[
\frac{R_{\text{comp}}}{R_{\text{comp}} + R_t} \leq A_2 \frac{n_t (R_{\text{comp}} - R_t)}{2} - \frac{1}{2} n_t [R_{\text{comp}} - R_t] \\
\text{for } R_t < R_{\text{comp}}
\]

Substituting \( \gamma = \frac{3 + 1}{2} \log 3 \), and \( \alpha = \frac{R_t}{R_{\text{comp}}} \) into Eq. 4-119, we finally obtain

\[
P(e_j) P(C_j) \leq A_2 \frac{1}{3^{3-1}} \frac{1}{2} - \frac{1}{2} n_t [R_{\text{comp}} - R_t] \\
\text{for } R_t < R_{\text{comp}}
\]

Substituting Eq. 4-115 and Eq. 4-120, we obtain

\[
P(e) \leq A_2 \frac{1}{3^{3-1}} \frac{1}{2} - \frac{1}{2} n_t [R_{\text{comp}} - R_t] \\
+ A_2 \frac{1}{3^{3-1}} \frac{1}{2} - \frac{1}{1+\alpha} n_t [T_{\alpha} (p_t) - H(p_t)] \\
\text{for } R_t < R_{\text{comp}}
\]

Now, let us define

\[
R_c = 1 - H(p_c) \\
\text{where } p_c \text{ is defined as}
\]

\[
p_c = \sqrt{\frac{p_c}{q_c}} \\
q_c = \sqrt{q_o}
\]

For \( R_t \leq R_c \), we can show that

\[
R_{\text{comp}} - R_t < T_{\alpha} (p_t) - H(p_t)
\]
Thus, we may have

\[
P(\epsilon) \approx A_2^3 \frac{1}{3-1} \left\{ \frac{1}{2} \frac{1}{1-\delta^2} \right\} \frac{n_t[R_{\text{comp}} - R_t]}{2} \tag{4-125}
\]

For \( R_c < R_t < R_{\text{comp}} \), we can show that

\[
R_c - R_t > T_o(p_t) - H(p_t) \tag{4-126}
\]

Thus, we may have

\[
P(\epsilon) \leq A_3^3 \frac{1}{3-1} \left\{ \frac{1}{2} \frac{1}{1-\delta^2} \right\} \frac{n_t[T_o(p_t) - H(p_t)]}{2} \tag{4-127}
\]

Combining Eq. 4-125 and Eq. 4-127, we finally obtain

\[
P(\epsilon) \leq \begin{cases} 
A_2^2 \frac{n_t[R_{\text{comp}} - R_t]}{2} & \text{for } R_t \leq R_c \\
A_3^2 \frac{n_t[T_o(p_t) - H(p_t)]}{2} & \text{for } R_c < R_t < R_{\text{comp}}
\end{cases} \tag{4-128}
\]

where

\[
A_3 = A_2^3 \frac{1}{3-1} \left\{ \frac{1}{2} \frac{1}{1-\delta^2} \right\} \frac{1}{1-\delta^2} \tag{4-129}
\]

The bounds on the ensemble average probability of error developed in this analysis are different from those obtained by Wozencraft and Reiffen [2]. By comparison, we can see that these bounds still decrease exponentially with constraint length \( n_t \), but the exponent have been reduced by a factor of \( \frac{1}{2} \); we also notice that the coefficients in our results are independent of \( n_t \), but the coefficients in Wozencraft's results are proportional to \( n_t^2 \).
CHAPTER 5. CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

5-1. Conclusion

In Chapter 2 of this research report, the convolutional tree codes have been studied in detail. All the known properties of this class of codes have been carefully re-examined and generalized; and some new properties have been developed and proved. Most important features of this chapter are that we have given some considerations to the construction of good tree codes, and rules for step-by-step construction of good generators have been proposed. These rules enable us to construct tree codes with large portion of the minimum distance at the very beginning of the trees (i.e., the sub-tree emerging from any node separate very quickly at the beginning). This quick separation property has been utilized in the decoding presented in the Chapter 3. By hand calculation, the author is able to construct good generators up to \( \nu = 12 \). An extensive computer searching program for good generators has been carried out by Mr. Henry Lyne, at Rice, who intends to construct good generators up to \( \nu = 30-40 \) or longer.

In Chapter 3, a new sequential decoding scheme for the binary symmetric channel has been proposed. The most important features of this scheme are that:

1. the determination of the thresholds is simple, and they only depend on the distance property of the tree code. But the selection of the thresholds in any other sequential decoding scheme is based upon the channel transition probability \( p_0 \) which is assumed to be known beforehand by the decoder. In a practical situation, \( p_0 \) cannot be well determined; therefore, the set
of thresholds cannot be evaluated properly; this, in turn, will hurt the performance of the decoding scheme.

(ii) In the searching process of this scheme, the decoder always attempts to find a path in the tree which differs from the received sequence in less than $T(v)$ places, where $T(v)$ is the error correcting capability of the tree code. If the channel noise disturbs the transmitted path less than $T(v)$ places in $n_t$ digits, the decoder always decodes the message digits correctly. Thus, the decoding scheme utilizes the error-correcting capability of a tree code fully.

(iii) A sensitive test, called the multiple-threshold test, has been provided to aid in early detection and approximate location of decoding errors. This test is especially effective when the tree codes have the property of quick separation.

(iv) Although estimates of the probability of error and numbers of decoding operations are based on the oversimplified binary symmetric channel model, the procedures have been designed to deal with "burst errors" within the error-correcting capability of codes.

(v) The decoding scheme consists of two modes, the running mode of decoding is designed to be used as often as possible, the complicated searching process will not be used unless the channel noise is too severe.

(vi) Periodic restart process is used to stop the error propagation inherent in the sequential decoding.
This decoding scheme is considered to be conceptually simpler than any other known sequential decoding schemes. The simulation studies of this decoding scheme are currently being performed on computer by Mr. Henry Lyne; experimental results will be obtained soon.

In Chapter 4, we have introduced the concept of threshold block decoding which provides an easy and reasonable way to evaluate thresholds. When these thresholds are applied to Wozencraft's sequential decoding scheme, it is possible to show that the average number of computations required to process the incorrect subset is bounded by a quantity which is independent of the constraint length; and the probability of error decreases exponentially with $n_t$. If the modified scheme suggested by Gallager is used, it is easy to show that the total average number of computations is bounded by a quantity which is proportional to $n_t^{2+\beta}$, where $\beta < 1$. This suggests that the use of appropriate thresholds will reduce the number of operations.
5-2. **Recommendations for Future Research**

In the following, we suggest certain areas in which further research should be worthwhile:

(i) Further investigation on the algebraic properties of the convolutional tree codes.

In the new sequential decoding scheme presented in this research, we intend to exploit the known algebraic properties of the tree codes as much as possible. But the only properties being exploited are the tree structure, the minimum distance, and the reentry property stated in Theorem 2-1. It seems that these properties may be even utilized in a more effective manner and other properties may also be applicable to the decoding. Besides, more new properties may be developed and proved.

(ii) Construction of good generators for tree codes.

A more effective algorithm and easy way for constructing good generators of tree codes should be developed, in order to provide a large error correcting capability.

(iii) Investigation of the block group codes with tree structures.

Besides the convolutional tree codes, there are some known block group codes which also have the tree-like structure; the Reed-Muller codes are examples. Because of their tree structure and the large distance between any pair of codewords, it seems possible to develop a simple probabilistic decoding scheme for these kinds of block codes.
(iv) Sequential-like block decoding for block group codes with tree structure.

Because of their tree structure and large distance between any pair of codewords, the following decoding scheme seems to be promising. The sequential search process is used to discard most of the improbable codewords from detailed consideration, and then select the most probable one from those retained codewords. It is obvious that this decoding is the optimum one in the sense of probability of error (i.e., maximum likelihood decoding). Because the operations performed in the sequential search process and the size of the retained subset of codewords are usually both small, it is expected that the total number of computations for making a decoding will also be small. Since it has the nature of block decoding, the error propagation does not exist.
APPENDIX A. Bounds on Sums of Random Variables

1. Chernoff Bounds

Assume \( \{Z_i(\cdot)\}_{i=1}^n \) is a sequence of independent random variables, each of which has the same distribution. Let their sum be

\[
D_n(\xi) = \sum_{i=1}^n Z_i(\xi)
\]

(A-1)

The moment generating function of \( Z_i(\cdot) \), denoted by \( M_i(t) \), is defined for every real number \( t \) by

\[
M_i(t) = \mathbb{E}[e^{tZ_i}] = \int_{-\infty}^{\infty} e^{tz} dF_{Z_i}(z)
\]

(A-2)

where \( F_{Z_i}(z) \) is the distribution function of \( Z_i(\cdot) \). It is clear that all \( M_i(t) \) are the same, since all \( Z_i(z) \) have the same distribution. Let the common moment generating function be \( M(t) \). The moment generating function of \( D_n(\xi) \) is given by

\[
M(t) = \mathbb{E}[e^{tD_n}] = \mathbb{E}[e^{t \sum_{i=1}^n Z_i}] = \prod_{i=1}^n M_i(t)
\]

(A-3)

In order to evaluate the probability that \( D_n(\xi) \) is greater than or equal to some value \( k \), we define an event

\[
A_k = \{\xi: D_n(\xi) \geq k\}
\]

(A-4)

For \( t > 0 \)

\[
A_k = \{\xi: e^{tD_n(\xi)} \geq e^{tk}\}
\]

(A-5)

Since

\[
e^{tD_n(\xi)} \geq e^{tk} I_{A_k}(\xi)
\]

(A-6)

where

\[
I_{A_k}(\xi) = \begin{cases} 1 & \xi \in A_k \\ 0 & \xi \in A_k^c
\end{cases}
\]

(A-7)

it follows that

\[
\mathbb{E}[e^{tD_n}] \geq \mathbb{E}[e^{tk} I_{A_k}(\xi)] = e^{tk} P(A_k)
\]

(A-8)

From Eq. A-8 we obtain
\[ P(A_k) \leq e^{tk} M n(t) \]

\[ P(A_k)^N = P[D_n \geq k] \leq e^{-tk} n(t) \]  \hspace{1cm} (A-9)

Now, put \[ u(t) \overset{D}{=} \log e M(t) \]  \hspace{1cm} (A-10)

and note that \[ u(0) = 0. \] We thus have

\[ P(D_n \geq k) \leq e^{-tk + nu(t)} \] \hspace{1cm} t \geq 0 \hspace{1cm} (A-11)

In order to obtain the best possible bound, we maximize with respect to the parameter \( t \). Setting the derivative of the exponent equal to zero, we get

\[ \frac{d}{dt} [-tk + nu(t)] = -k + nu'(t) = 0, \text{ which yields } k = nu'(t) \]  \hspace{1cm} (A-12)

A maximum, if it exists, must be a root of \[ u'(t) = \frac{k}{n}. \] Finally we have

\[ P(D_n \geq nu'(t)) \leq e^{-n[t u'(t) - u(t)]} \] \hspace{1cm} t \geq 0 \hspace{1cm} (A-13)

By a similar argument we can obtain

\[ P(D_n \leq n u'(t)) \leq e^{-n[t u'(t) - u(t)]} \] \hspace{1cm} t \leq 0 \hspace{1cm} (A-14)

Equations A-13 and A-14 are called Chernoff bounds and are valid for any finite, discrete random variable.
2. **Sums of Binomials**

Suppose each $Z_i(*)$ is a binary random variable with two possible values "0" or "1". And $P(Z_i = 1) = p_0$ and $P(Z_i = 0) = 1 - p_0 = q_0$

Then, from Eqs. A-2 and A-10

$$M(t) = p_0 e^t + q_0$$  \hspace{1cm} (A-15)

$$u(t) = \log_e(p_0 e^t + q_0)$$  \hspace{1cm} (A-16)

and

$$u'(t) = \frac{p_0 e^t}{q_0 + p_0 e^t}$$  \hspace{1cm} (A-17)

If we let $k = np = n u'(t)$ then $p = \frac{p_0 e^t}{q_0 + p_0 e^t} \geq p_0$ for $t \geq 0$

or

$$p q_0 + p p_0 e^t = p_0 e^t$$

$$p_0 (1-p) e^t = p q_0$$

$$e^t = \frac{p q_0}{p_0 q}$$ where $q = 1 - p$

$$t = \log_e \frac{p q_0}{p_0 q}$$  \hspace{1cm} (A-18)

Substituting Eq. A-18 into A-15, we get

$$u(t) = \log_2 \left( q_0 + \frac{q_0 \rho}{q} \right)$$  \hspace{1cm} (A-19)

Substituting A-19 into A-13, we obtain

$$P(D_n \geq np) \leq e^{-n[p \log_e \frac{q_0 \rho}{p_0 q} - \log_e \left( q_0 + \frac{p q_0}{q} \right)]} \text{ for } p \geq p_0 > 0$$  \hspace{1cm} (A-20)

Since $e^k \log_e x = 2^k \log_2 x$ Eq. A-20 becomes

$$P(D_n \geq np) \leq 2^{-n[p \log_2 \frac{q_0 \rho}{p_0 q} - \log_2 \left( q_0 + \frac{p q_0}{q} \right)]}$$  \hspace{1cm} (A-21)

By a little algebra, the term in brackets may be written

$$T_0(p) - H(p) = [-p \log_2 p_0 - q \log_2 q_0]$$

$$-[-p \log_2 p - q \log_2 q]$$
where
\[ T_0(p) = -p \log_2 p_0 - q \log_2 q_0 \] (A-22a)
\[ H(p) = -p \log_2 p - q \log_2 q \] (A-22b)

Then Eq. A-21 becomes
\[ P(D_n \geq np) \leq 2^{-n[H_0(p) - H(p) + (p-p_0) \log_2 \frac{q_0}{p_0}]} \quad p \geq p_0 \neq 0 \] (A-22)

By a similar argument, we obtain
\[ P(D_n \leq np) \leq 2^{-n[H(p_0) - H(p) + (p-p_0) \log_2 \frac{q_0}{p_0}]} \quad \text{for } p \leq p_0 \] (A-23)

Of particular interest is the case where \( p_0 = \frac{1}{2} \). Since \( H(\frac{1}{2}) = 1 \), we have
\[ P[D_n \leq np] \leq 2^{-n[1 - h(p)]} \quad p \leq \frac{1}{2} \] (A-24)
APPENDIX B. A Study of the Random Variables for a Binary Symmetric Channel

Let $B^n_1$ be the channel input space; i.e., $B^n_1$ is the set of possible sequences consisting of $n$ binary digits. There are $N = 2^n$ such sequences, which may be numbered $\overline{u}_1, \overline{u}_2, ..., \overline{u}_n$. Consider the random variable $U(\cdot)$. For each choice of the variable $\xi$, an input sequence is determined.

$$U(\xi) = \overline{u} = (u^1, u^2, u^3, ..., u^n) \in B^n_1$$

$u^k$ is the $k$th component of $\overline{u}$. We may also write $U(\xi) = \{U^1(\xi), U^2(\xi), ..., U^n(\xi)\}$

$$u^k = U^k(\xi) \in B = \{0, 1\}$$

Similarly, let $B^n_2$ be the channel output space consisting of $N = 2^n$ possible sequence of $n$ binary digits. Let the random variable $V(\cdot)$ be such that $V(\xi) = \overline{v} = (v^1, v^2, v^3, ..., v^n) \in B^n_2$. For each choice of $\xi$, $v^k$ is the $k$th component of $\overline{v}$. We may also write $V(\xi) = \{V^1(\xi), V^2(\xi), ..., V^n(\xi)\}$

$$v^k = V^k(\xi) \in B = \{0, 1\}$$

Let $B^n_3$ be the channel noise space consisting of $N = 2^n$ possible noise sequences of $n$ binary digits. It is convenient to define the random variable $E(\cdot)$ such that $E(\xi) = \overline{e} = (e^1, e^2, ..., e^n) = U(\xi) + V(\xi)$. We may also write $E(\xi) = \{E^1(\xi), E^2(\xi), ..., E^n(\xi)\}$, where $e^k = E^k(\xi)$. For the binary symmetric channel (BSC), $\{E^k(\cdot) : 1 \leq k \leq n\}$ is an independent class with $P(\{\xi : E^k(\xi) = 1\}) = p_0$ and $P(\{\xi : E^k(\xi) = 0\}) = q_0 = 1 - p_0$. The random variables $U(\cdot)$ and $E(\cdot)$ are independent.

For random coding, $U(\xi) = \overline{u}_1$ is constructed by selecting its $n$ digits independently at random with equal probability. This is assumed to make $\{U^j(\cdot) : 1 \leq j \leq n\}$ an independent class with $P(\{\xi : U^k(\xi) = 1\}) = P(\{\xi : U^k(\xi) = 0\}) = \frac{1}{2}$. Also $\{U^j(\cdot), E^k(\cdot) : 1 \leq j \leq n, 1 \leq k \leq n\}$ is an independent class.
Each time a sequence \( U(\xi) = \overline{u}_{1} \) is picked randomly to be transmitted, we also select \( 2^{\nu} - 1 \) codewords from \( B_{1}^{n} \) with replacement. We call this set of \( 2^{\nu} - 1 \) codewords \( S_{k}^{*} \). The distinct sequences in \( S_{k}^{*} \). The distinct sequences in \( S_{k}^{*} \cup \overline{u}_{1} \) forms a codeword list \( S_{k} \) which must be sent to the decoder for decoding.

Let \( S^{*}(\cdot) \) be a random variable such that \( S^{*}(\xi) = S_{k}^{*} \) for each choice of \( \xi \). We suppose \( U(\cdot), E(\cdot), S^{*}(\cdot) \) are independent random variables. Let

\[
V(\xi) = U(\xi) \oplus E(\xi) \quad \text{so that} \quad (B-1)
\]

\[
v^{k}(\xi) = U^{k}(\xi) \oplus E^{k}(\xi) \quad (B-2)
\]

Let

\[
A_{k} = \{ \xi: U^{k}(\xi) = 1 \}
\]

\[
B_{k} = \{ \xi: E^{k}(\xi) = 1 \}
\]

\[
C_{k} = \{ \xi: v^{k}(\xi) = 1 \}
\]

Then

\[
C_{k} = A_{k} B_{k} C_{k} + A_{k} C_{k} B_{k} = f_{k}(A_{k} B_{k}) \quad \text{(A Boolean function)}
\]

By a known theorem on independence \( \{A_{k} B_{j}: 1 \leq j \leq n, \ 1 \leq k \leq n\} \) is an independent class. This implies the fact that \( \{C_{k}: 1 \leq k \leq n\} \) is also an independent class [13]. Write

\[
U^{k}(\xi) = I_{A_{k}}(\xi) \quad (B-3)
\]

\[
E^{k}(\xi) = I_{B_{k}}(\xi) \quad (B-4)
\]

\[
v^{k}(\xi) = I_{C_{k}}(\xi) = I_{A_{k}} B_{k} C_{k} + A_{k} B_{k} C_{k} = I_{f_{k}(A_{k} B_{k})}(\xi) \quad (B-5)
\]

\( \{v^{k}(\xi): 1 \leq k \leq n\} \) is an independent class iffi \( \{f_{k}(A_{k} B_{k}): 1 \leq k \leq n\} \) or \( \{C_{k}: 1 \leq k \leq n\} \) or \( \{C_{k}: 1 \leq k \leq n\} \) is an independent class. Since we have already shown \( \{C_{k}: 1 \leq k \leq n\} \) is an independent class, it follows that \( \{v^{k}(\xi): 1 \leq k \leq n\} \) is an independent class.

Consider \( E_{i}(\xi) \overset{D}{=} V(\xi) \oplus \overline{u}_{i} \) (B-6)

where \( V(\xi) \) is the received sequence and \( \overline{u}_{i} \in S_{k}^{*} \). The sequence \( \overline{u}_{i} \) is called the test signal. The random variable \( E_{i}(\xi) \) is called test noise.
sequence. Now, $E_i(\xi) = \{E_i^1(\xi), E_i^2(\xi), \ldots, E_i^n(\xi)\}$, where $E_i^k(\xi) = V^k(\xi) \oplus u_i^k$. Since $\{V^k(\xi) \mid 1 \leq k \leq n\}$ is an independent class and $u_i^k$ is a constant, it is easy to show that $\{E_i^k(\xi) \mid 1 \leq k \leq n\}$ is also an independent class.

Since $V^k(\xi) = U^k(\xi) \oplus E^k(\xi)$, it follows that

$$P(V^k = 1) = P(U^k = 0, E^k = 1) + P(U^k = 1, E^k = 0)$$

$$= P(U^k = 0)P(E^k = 1) + P(U^k = 1)P(E^k = 0)$$

$$= \frac{1}{2} P_0 + \frac{1}{2} q_0 = \frac{1}{2} \quad (B-7)$$

By the same argument $P(V^k = 0) = \frac{1}{2} \quad (B-8)$

Since $E_i^k(\xi) = V^k(\xi) \oplus u_i^k \quad (B-6)$

and $V^k(\xi) = U^k(\xi) \oplus E^k(\xi) \quad (B-2)$

Substituting (B-2) into (B-6), we obtain

$$E_i^k(\xi) = U^k(\xi) \oplus E^k(\xi) \oplus u_i^k \quad (B-9a)$$

Let $U_i^k(\xi) = U^k(\xi) \oplus u_i^k$. Then

$$E_i^k(\xi) = E^k(\xi) \oplus U_i^k(\xi) \quad (B-9b)$$

and $\{\xi : E_i^k(\xi) = 1\} = \{E^k(\xi) = 1, U_i^k(\xi) = 0\} \cup \{E^k(\xi) = 0, U_i^k(\xi) = 1\} \quad (B-10)$

Therefore,

$$P(E_i^k = 1) = P(E^k = 1, U_i^k = 0) + P(E^k = 0, U_i^k = 1)$$

$$= \frac{1}{2} P_0 + \frac{1}{2} q_0$$

$$= \frac{1}{2} \quad (B-11a)$$

By the same argument, we obtain

$$P(E_i^k = 0) = \frac{1}{2} \quad (B-11b)$$

Equations (B-11a) and (B-11b) show the probability that any test signal $\bar{u}_i$ agrees with the received signal $V(\xi)$ at any digit is $\frac{1}{2}$. This is an important fact for random coding. We have used this fact in deriving the upper bound on the average probability of decoding error.
Next we want to show the independence of random variables $\mathcal{E}_k(\cdot)$ and $\mathcal{E}_k^*(\cdot)$. From (B-9b), we have
\[
[\xi: \mathcal{E}_k(\xi) = 1, \mathcal{E}_k^*(\xi) = 1] = [\xi: u_k(\xi) = 0, \mathcal{E}_k(\xi) = 1]
\] (B-13)

Then
\[
P(\mathcal{E}_k = 1, \mathcal{E}_k^* = 1) = \frac{1}{2} p_0
\] (B-14)

\[
P(\mathcal{E}_k = 1 | \mathcal{E}_k^* = 1) = \frac{P(\mathcal{E}_k = 1, \mathcal{E}_k^* = 1)}{P(\mathcal{E}_k^* = 1)} = \frac{1}{2} \frac{p_0}{p_0} = \frac{1}{2}
\] (B-15)

Since $P(\mathcal{E}_k = 1) = \frac{1}{2}$, we have
\[
P(\mathcal{E}_k = 1 | \mathcal{E}_k^* = 1) = P(\mathcal{E}_k = 1) = \frac{1}{2}
\] (B-16)

By the same argument we can prove that
\[
P(\mathcal{E}_k = 1 | \mathcal{E}_k^* = 0) = P(\mathcal{E}_k = 1) = \frac{1}{2}
\]
\[
P(\mathcal{E}_k = 0 | \mathcal{E}_k^* = 0) = P(\mathcal{E}_k = 0) = \frac{1}{2}
\]
\[
P(\mathcal{E}_k = 0 | \mathcal{E}_k^* = 1) = P(\mathcal{E}_k = 0) = \frac{1}{2}
\]

Therefore,

$\mathcal{E}_k(\cdot)$ and $\mathcal{E}_k^*(\cdot)$ are independent, and $\{\mathcal{E}_j(\cdot), \mathcal{E}_k^*(\cdot): 1 \leq j \leq n, 1 \leq k \leq n\}$ is an independent class. Consider $D_1(\xi) = |\mathcal{E}_1(\xi)|$, the weight of a test noise sequence and $D(\xi) = |\mathcal{E}(\xi)|$ weight of a noise sequence. As functions of independent random variables, $D_1(\cdot)$, $D(\cdot)$ are also independent.
References


