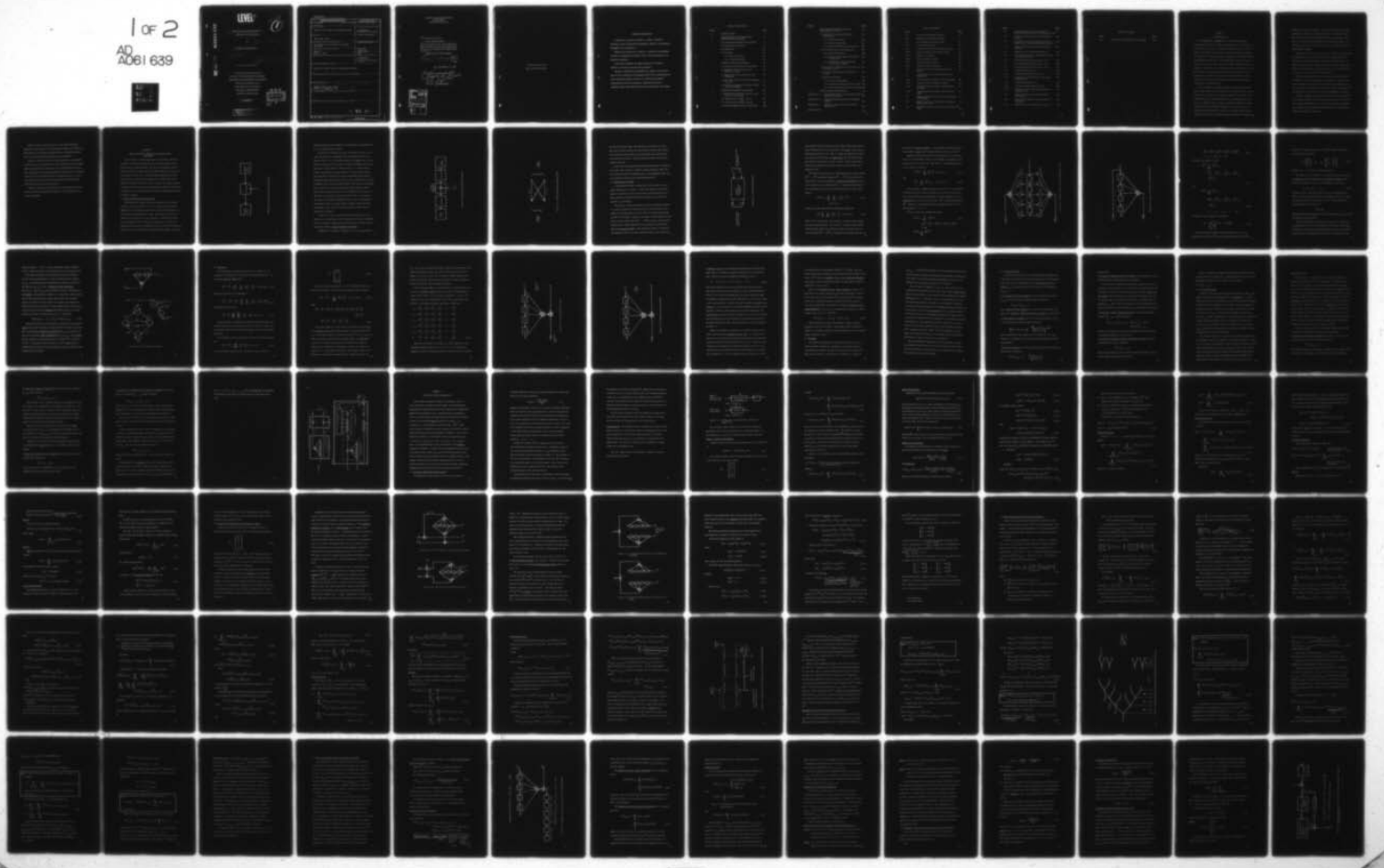
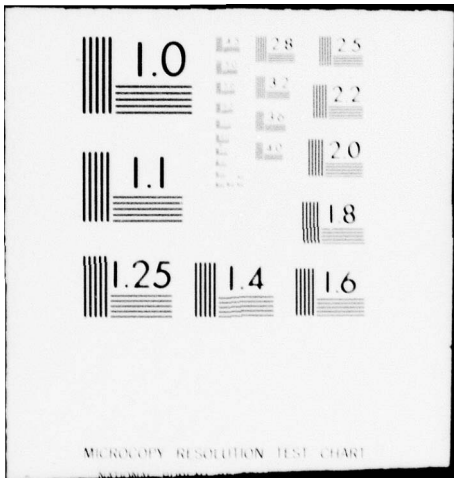
AD-A061 639 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/4
MAP ERROR BIT DECODING OF CONVOLUTIONAL CODES.(U)
AUG 77 J T KINDLE
UNCLASSIFIED AFIT-CI-79-13T NL

1 OF 2
AD
AD61 639




1.0
1.1
1.25
1.4
1.6
2.8
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART

MAP ERROR BIT DECODING OF CONVOLUTIONAL CODES

by

James Thomas Kindle

Doctoral thesis,

186 p.

AFIT-CI-79-13T

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Electrical Engineering)

August 1977

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CI 79-13T	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Map Error Bit Decoding of Convolutional Codes		5. TYPE OF REPORT & PERIOD COVERED Dissertation
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James Thomas Kindle		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT Student at the University of Southern California		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/CI WPAFB OH 45433		12. REPORT DATE August 1977
		13. NUMBER OF PAGES 176
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release, Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE AFR 190-17.

JOSEPH P. HIPPS, Major, USAF
Director of Information, AFIT

NOV 9 1978

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

11 21 030

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNIVERSITY OF SOUTHERN CALIFORNIA
THE GRADUATE SCHOOL
UNIVERSITY PARK
LOS ANGELES, CALIFORNIA 90007

This dissertation, written by

James Thomas Kindle

under the direction of his Dissertation Committee, and approved by all its members, has been presented to and accepted by The Graduate School, in partial fulfillment of requirements of the degree of

DOCTOR OF PHILOSOPHY

Harold W. Lee

Dean

Date September 2, 1977

DISSERTATION COMMITTEE

Chairman

To my wife Joan, and

sons James and Michael

ACKNOWLEDGEMENT

I would like to thank Dr. Charles L. Weber, committee chairman, for his continued encouragement, guidance, and optimism throughout this investigation.

Thanks are also due to Dr. Robert A. Scholtz for his excellent courses in coding theory at USC as well as his participation on my doctoral committee.

I also wish to express my appreciation to Dr. Russell A. Johnson for serving on my doctoral committee.

Finally I would like to acknowledge the support received from the Air Force Institute of Technology (AFIT) of the United States Air Force during my studies at USC as an AFIT doctoral student. Naturally the views and conclusions contained in this work represent those of the author and not necessarily those of the USAF.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Since Shannon [1] showed that it is theoretically possible through coding to achieve reliable communication at rates below channel capacity, much work in the field of coding theory was centered on finding and applying various suitable coding and decoding techniques for memoryless channels. One such technique is convolutional coding, first discussed by Elias [2] in 1955 and since shown to be quite cost effective in terms of error correcting capability provided for a given equipment complexity level. The space shuttle communications system [3] and commercial communication satellites [4] are but two of many current examples of the use of convolutional coding to obtain performance gain in a practical digital communication link having restrictions on available power.

The convolutional encoding operation is deterministic and thus quite easily implemented. The decoding of convolutional codes, on the other hand, must provide correction of random errors made in transmission and thus is a design problem with an underlying statistical nature. To approach such a problem one ideally would like to apply statistical estimation theory techniques directly and thus derive an optimal estimator (decoder). In practice a feasible decoder such as that due to Viterbi [5] may be developed first and only subsequently

shown to be an optimal estimator, in this case the optimal maximum likelihood data sequence estimator (Forney [6]). However Massey [7] effectively looked at a Maximum Aposteriori Probability (MAP) error bit estimator and showed how for the orthogonal subset of convolutional codes this could be specialized to the implementable threshold decoding technique.

In this work an attempt is made to first consider the convolutional decoding problem from an optimal estimation viewpoint and then specialize to possibly suboptimal but implementable decoding techniques. While much work on data estimation has appeared subsequent to the Viterbi algorithm, here error estimation is stressed and thus is more akin to the original work of Massey [7].

Chapter 2 provides a brief review of convolutional coding and decoding for memoryless channels with emphasis on only those elements needed later to discuss error estimation and syndrome decoding. Provided is a qualitative breakout of the various possible decoding approaches for data and errors from a probabilistic viewpoint.

Chapter 3 displays general optimal error bit and error sequence estimation techniques both with and without the practical restriction on time delay allowable before an error estimate must be produced. The complementary relationship between MAP error decoding and MAP data decoding is also noted.

Chapter 4 then concentrates on one of the optimal decoding approaches from Chapter 3, namely Optimal Finite Delay MAP Error Bit decoding. Two suboptimal approximations, ROMMAP and SUB-MAP, are derived and their implementations discussed.

Chapter 5 illustrates how the actual performance of a ROMMAP decoder may be computed and that of the SUBMAP decoder both upper and lower bounded. Also indicated is the relationship between SUB-MAP decoding and conventional feedback decoding. Lastly, simulation results of practical SUBMAP decoder implementations for several convolutional codes are presented to quantitatively illustrate and confirm previous theoretical predictions.

Chapter 6 qualitatively summarizes the conclusions that may be drawn from this investigation and suggests some continuations that may be worthwhile.

Chapter 2

CONVOLUTIONAL CODES AND PROBABILISTIC DECODING

In this chapter a brief introduction to convolutional coding and decoding is provided stressing only those areas necessary to later derivations and implementation. The basic communication system problem model is first presented to provide motivation for the use of convolutional coding. Next, various decoding approaches for both data and error estimation are qualitatively reviewed from a probabilistic standpoint to show the range of options that could be pursued for implementation of the crucial decoding step. Chapter 3 will further consider the error estimation options and derive appropriate optimal decoding techniques.

2.1 Basic Communication System Model

The underlying communication system problem can be simply stated by referring to Figure 2.1. Here a Data Source generates digital data to be reliably transferred to a Data User through a real communication channel which is noisy. The output of the source in this work is assumed to be a data sequence of symbols from GF(q) as is the input to the user. The noisy channel causes errors to be introduced into the data as presented to the user with such a frequency (error rate) so as to make the received data unacceptable. The noisy

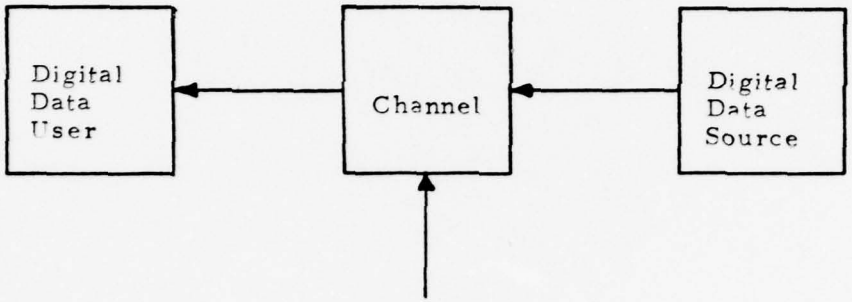


Figure 2.1 Communication System Problem

channel includes all the modulation, transmission, and demodulation of a real communications system.

To reduce the frequency of errors presented to the user, an encoder and decoder are appended to the communication system as shown in Figure 2.2. To simplify analysis, the binary case of GF(q) = GF(2) is assumed and the channel is modeled as an equivalent Binary Symmetric Channel (BSC) as shown in Figure 2.3. This is both a simple and appropriate channel model for a system which employs the commonly used 180 degree Binary Phase Shift Keying (BPSK) modulation with hard decisions in the receiver and which is corrupted by Additive White Gaussian Noise (AWGN). The specific relationship between the BSC crossover probability p and the Signal-to-Noise Ratio (SNR) encountered at the receiver will be given in Section 5.2. The BSC is memoryless in that the symbol output at each discrete time instant is statistically dependent only on the corresponding input symbol and not on any other input or output symbol preceding or following. Thus the BSC is also described as a DMC (Discrete Memoryless Channel).

In this work it also is assumed that the input data rate to the channel is so far below the channel capacity that sufficient bandwidth exists to enable the coding gains predicted in later sections. This is commonly called an infinite bandwidth assumption.

Included in the "solution" of Figure 2.2 are as yet unspecified

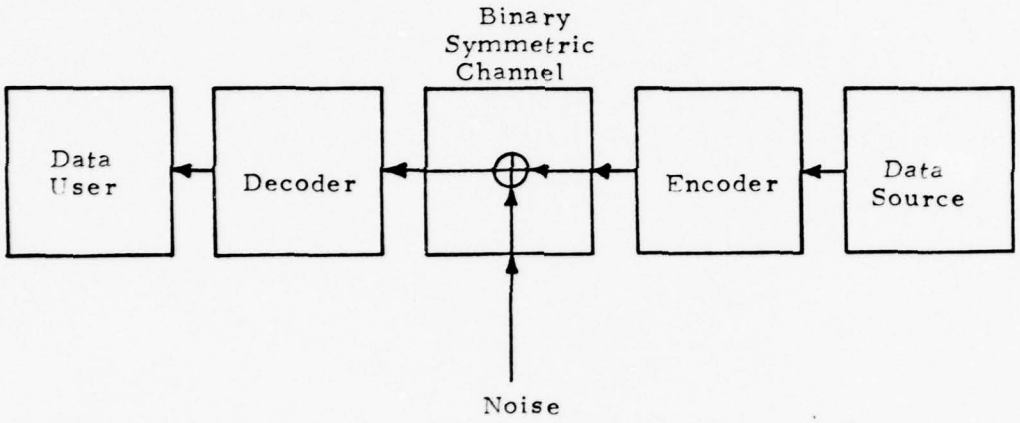


Figure 2.2 Communication System Solution

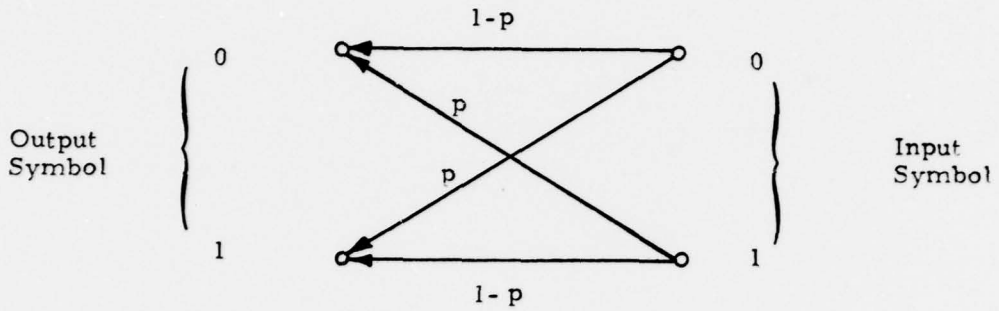


Figure 2.3. Binary Symmetric Channel (BSC)

encoder and decoder units. The function of the encoder is to map data source output symbols into code symbols such that BSC channel errors made in transmitting such code symbols can sometimes be corrected by the decoder - thus decreasing the data error rate presented to the user.

In this work convolutional encoding will be assumed to implement the encoder unit. This is a relatively simple encoding method. The more difficult problem of implementing a corresponding decoder will be investigated using several estimation approaches.

2.2 Convolutional Encoding

A convolutional encoder is defined as a linear finite-state machine employing one or more ν - stage shift registers and having k inputs at each discrete time instant with n outputs determined by n linear algebraic function generators operating on the present and possible ν prior inputs.

The idea of a simple convolutional encoder is depicted in Figure 2.4. where each block of k binary bits from the Data Source is encoded into an output block of n ($n > k$) bits to be transferred through the BSC to the decoder. However, the output n-tuple depends not only on the current input k-tuple $(I_t^{(1)}, \ldots, I_t^{(k)})$ but also is constrained by certain prior k-tuples whose effect is still present in the encoder. Thus the constraint length K of the particular encoder is defined as the maximum number of n-tuple output bits which can be influenced

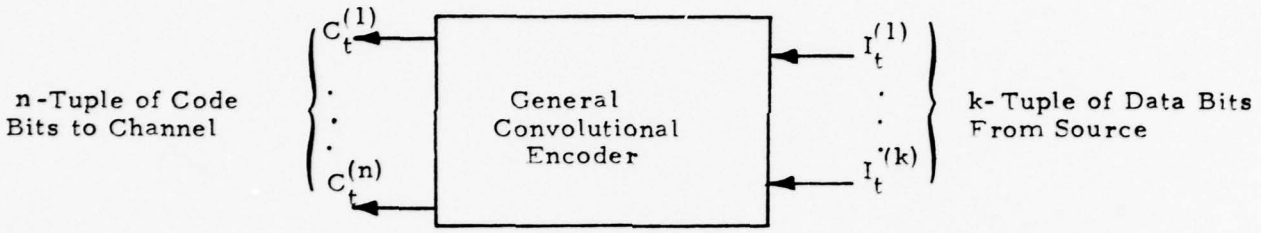


Figure 2.4. Convolutional Encoder

by an input k-tuple bit. Each bit of the n-tuple is thus generated by a linear combination of K or less data bits. The rate R of the code can be approximated by R = k/n. That is, though the channel is transferring n bits per time period, the information rate is only k bits per time period. The presence of these "excess" n-k bits represents a feature to be exploited later by the decoder to correct BSC transmission errors.

The linear function generator relationship between input k-tuples $(I_t^{(1)}, \ldots, I_t^{(k)})$ and output n-tuples $(c_t^{(1)}, \ldots, c_t^{(n)})$ can in general be expressed in terms of coefficients of generator polynomials in arbitrary variables y and z (Berlekamp [8]). Using notation similar to that of Berlekamp, the generator polynomial set for k inputs and n outputs

$$G^{(j)}(y, z) = \sum_{\ell=1}^{k} \sum_{h=0}^{\nu} G_{\ell, h}^{(j)} \, y^{\ell} \, z^{h} \qquad (2.1)$$

$$\text{for } 1 \leq j \leq n$$

yields the n-tuple output digits (code digits) computed by

$$C_t^{(j)} = \sum_{\ell=1}^{k} \sum_{h=0}^{\nu} G_{\ell, h}^{(j)} \, I_{t-h}^{\ell} \quad \text{for } 1 \leq j \leq n. \qquad (2.2)$$

Here t is the current time. The variable y is used to index through the input k-tuple positions. The variable z is used to index through information bits within the constraint length of the code. Since the current k-tuple $(I_t^{(1)}, \ldots, I_t^{(k)})$ is available at the encoder input but not

yet stored, the memory length ν of the encoder specifies the maximum number of prior input k-tuples which may affect a given $c_t^{(j)}$.

Although the above notation is sufficiently general to be used by all rate k/n convolutional codes, it is worthwhile to specialize to the systematic rate 1/n code case. Thus with k = 1 and D substituted for z, (2.1) and (2.2) become:

$$G^{(j)}(D) = \sum_{h=0}^{\nu} G_h^{(j)} D^h \quad \text{for } 1 \le j \le n \tag{2.3}$$

and

$$C_t^{(j)} = \sum_{h=0}^{\nu} G_h^{(j)} I_{t-h} \quad \text{for } 1 \le j \le n. \tag{2.4}$$

As an example, consider the binary, rate $R = k/n = \frac{1}{2}$ systematic convolutional encoder shown in Figure 2.5a in terms of binary unit delay (storage) devices and polynomial coefficient weights. The term systematic here means that one code bit, $C_t^{(1)}$, is always identical to $I_t^{(1)}$. The binary logic implementation is shown in Figure 2.5b. In this example $K = \nu + 1 = 5$.

The set of generator polynomials is thus:

$$G^{(1)}(D) = \sum_{h=0}^{\nu} G_h^{(1)} D^h \tag{2.5}$$

$$= G_0^{(1)} + G_1^{(1)} D + G_2^{(1)} D^2 + G_3^{(1)} D^3 + G_4^{(1)} D^4$$

$$= 1 \text{ and}$$

$$G^{(2)}D = \sum_{h=0}^{\nu} G_h^{(2)} D^h$$

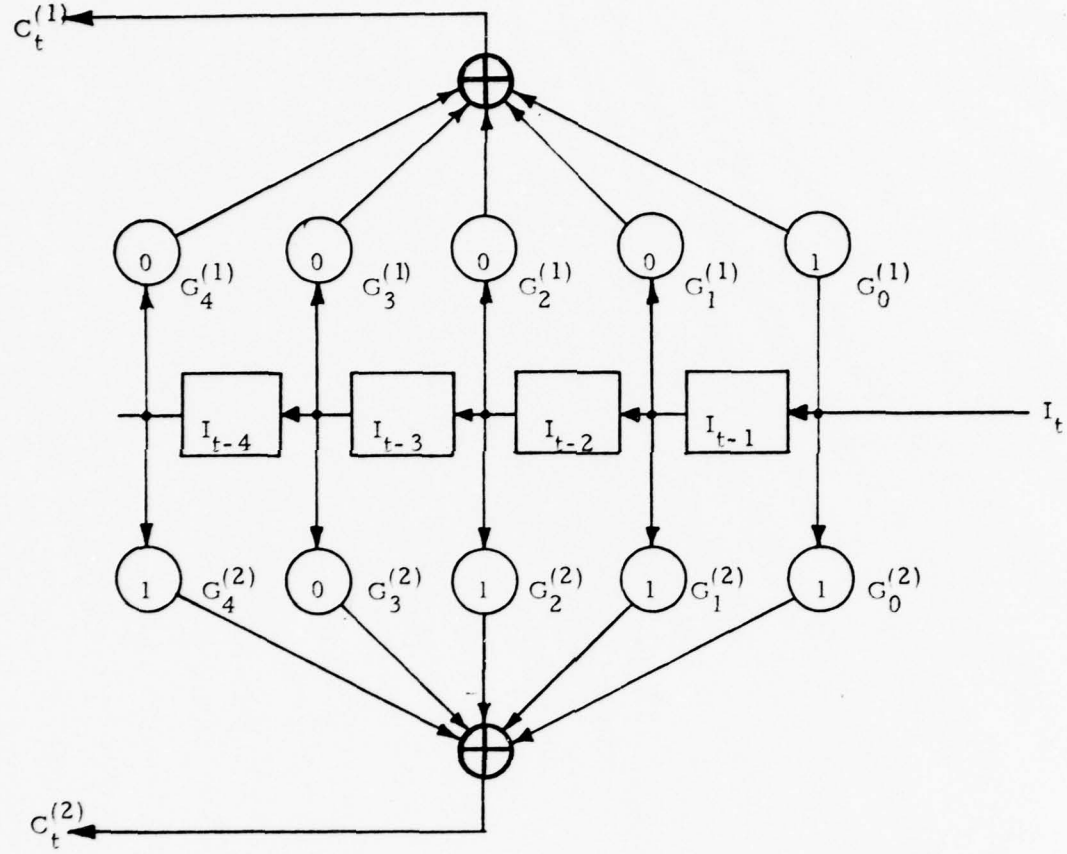


Figure 2.5a Encoder for K=5 Rate 1/2 Systematic Code

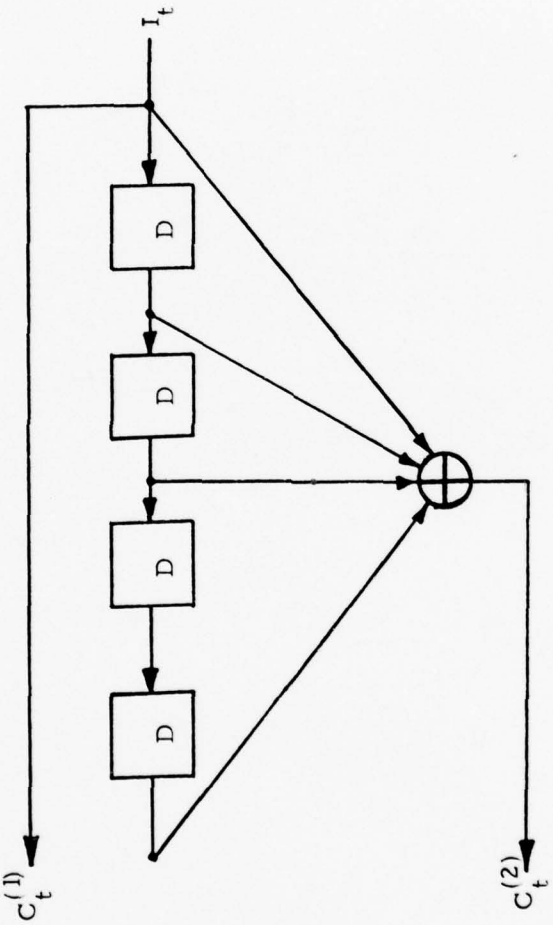


Figure 2.5b Logic of K=5 Rate 1/2 Systematic Code

$$= G_0^{(2)} + G_1^{(2)} D + G_2^{(2)} D^2 + G_3^{(2)} D^3 + G_4^{(2)} D^4 \qquad (2.6)$$

$$= 1 + D + D^2 + 0 + D^4 .$$

The output code digits are then:

$$C_t^{(1)} = \sum_{h=0}^{\nu} G_h^{(1)} I_{t-h}$$

$$= G_0^{(1)} I_t + G_1^{(1)} I_{t-1} + G_2^{(1)} I_{t-2} + G_3^{(1)} I_{t-3} + G_4^{(1)} I_{t-4}$$

$$= I_t \qquad (2.7)$$

and

$$C_t^{(2)} = \sum_{h=0}^{\nu} G_h^{(1)} I_{t-h}$$

$$= G_0^{(2)} I_t + G_1^{(2)} I_{t-1} + G_2^{(2)} I_{t-2} + G_3^{(2)} I_{t-3} + G_4^{(2)} I_{t-4} \qquad (2.8)$$

$$= I_t + I_{t-1} + I_{t-2} + 0 + I_{t-4} .$$

In terms of vector notation one would have

$$\underline{C}_t = \begin{bmatrix} C_t^{(1)} \\ C_t^{(2)} \end{bmatrix} \quad \text{and} \quad \underline{I}_t = [I_t] . \qquad (2.9)$$

For this example consider the transmission of the vector $\underline{C}_t$ through the BSC. Each $C_t^{(i)}$ will be independently affected by noise

in the form of additive binary bit errors $e_t^{(i)}$. Thus at the decoder the received vector will be:

$$\underline{R}_t = \begin{bmatrix} r_t^{(1)} \\ r_t^{(2)} \end{bmatrix} = \underline{C}_t + \underline{E}_t = \begin{bmatrix} C_t^{(1)} + e_t^{(1)} \\ C_t^{(2)} + e_t^{(2)} \end{bmatrix} \tag{2.10}$$

Thus $\underline{E}_t = \underline{R}_t - \underline{C}_t = \underline{R}_t \oplus \underline{C}_t$ in this binary case.

It is instructive at this point to very briefly mention the concepts of encoder state and the encoder state transition probability. The state of the encoder at time t, $\mathscr{S}_t$, is defined for a rate 1/n by

$$\mathscr{S}_t = (I_{t-1}, I_{t-2}, \ldots, I_{t-\nu}) \tag{2.11}$$

systematic code. Thus the encoder state can be represented by a binary number composed of the values residing in each of the ν physical storage elements of the encoder at time t. The encoder state transition probability at time t is

$$P(\mathscr{S}_{t+1} / \mathscr{S}_t),$$

the probability that the next state of the encoder will be $\mathscr{S}_{t+1}$ given that the encoder is now in state $\mathscr{S}_t$.

With such definitions, the operation of the rate 1/n encoder can be considered in terms of a state diagram of 2^ν states. The encoder resides in a current state $\mathscr{S}_t$ and will transition to one of two allowed successor states depending on the value of I_t. For illustrative pur-

poses the simple $\nu = 2, K = 3$ rate $\frac{1}{2}$ systematic encoder of Figure 2.6a is considered from a state diagram point of view in Figure 2.6b.

The output code digits are deterministic functions of both the current state $\mathscr{S}_t$ and the input I_t. Since I_t becomes the first element of $\mathscr{S}_{t+1}$, the output digits are equivalently deterministic functions of the state pair $\mathscr{S}_t$, $\mathscr{S}_{t+1}$. Thus a <u>conditional output probability</u> $P(\underline{C}_t / \mathscr{S}_t, I_t) = P(\underline{C}_t / \mathscr{S}_t, \mathscr{S}_{t+1})$ can be defined for any $\underline{C}_t$. Obviously for the <u>encoder</u> this probability is either 0 or 1. However in later discussions it will be shown that for another linear finite-state machine, namely the syndrome former portion of a decoder, the output probability will be more a function of certain random inputs. For now it is only worth noting that the <u>conditional</u> probability $P(\underline{C}_t / \mathscr{S}_t, I_t)$ is independent of all prior encoder states $\mathscr{S}_j$ for $j < t$. Therefore,

$$P(\underline{C}_t / \mathscr{S}_0, \mathscr{S}_1, \ldots, \mathscr{S}_{t-1}, \mathscr{S}_t, \mathscr{S}_{t+1}) = P(\underline{C}_t / \mathscr{S}_t, \mathscr{S}_{t+1}). \qquad (2.12)$$

Thus the encoder can be considered as a Markov source having 2^{ν} <u>states</u> whose state transitions are determined by the value of the input data I_t through <u>state transition probabilities</u> $P(\mathscr{S}_{t+1} / \mathscr{S}_t)$ and whose outputs are specified by <u>output probabilities</u> $P(\underline{C}_t / \mathscr{S}_t, \mathscr{S}_{t+1})$. This <u>encoder</u> Markov property was used productively by McAdam [9] and Bahl et al. [10] in analyzing encoder operation. It will be applied and expanded in later sections of this investigation to analyze a slightly more complex decoder problem.

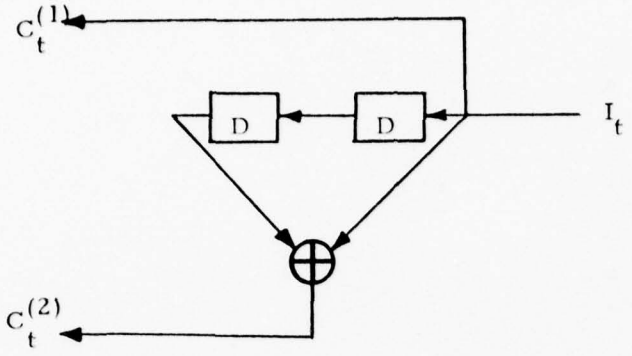


Figure 2.6a Simple $\nu = 2$ Encoder

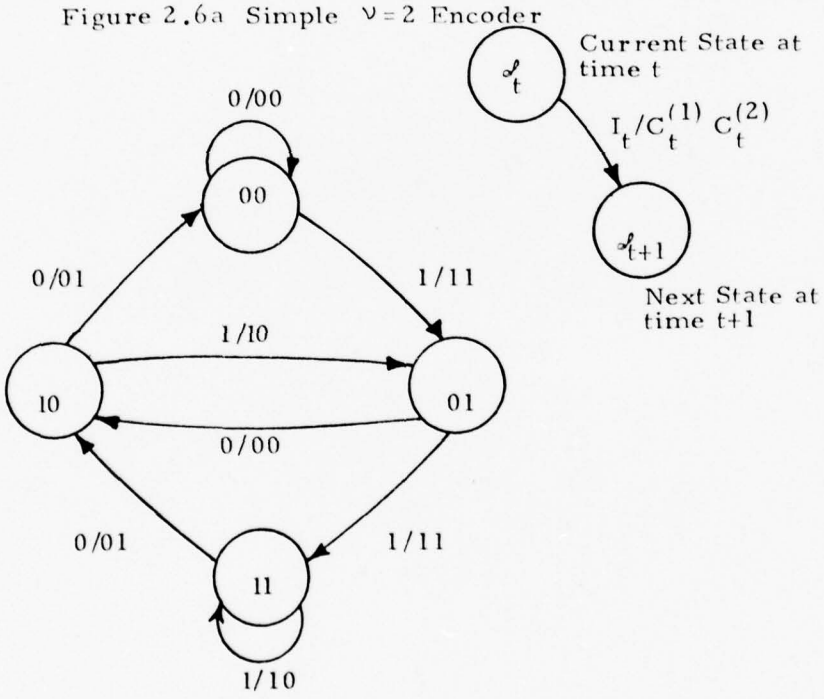


Figure 2.6b Equivalent State Diagram

2.3 Syndromes

For the moment reconsider the general k, n encoder case. In accordance with Massey [7] (Theorem 9) and Berlekamp [8] one can define syndrome digits $Z_t^{(j)}$ as

$$Z_t^{(j)} = r_t^{(j)} - \sum_{\ell=1}^{k} \sum_{h=0}^{\nu} G_{\ell,h}^{(j)} \; r_{t-h}^{(\ell)} \quad \text{for } k < j \leq n. \tag{2.13}$$

Using Eq. (2.10) this can be expanded to:

$$Z_t^{(j)} = C_t^{(j)} + e_t^{(j)} - \sum_{\ell=1}^{k} \sum_{h=0}^{\nu} G_{\ell,h}^{(j)} \; (C_{t-h}^{(\ell)} + e_{t-h}^{(\ell)}) \tag{2.14}$$

and reduced by Eq. (2.2) to

$$Z_t^{(j)} = e_t^{(j)} - \sum_{\ell=1}^{k} \sum_{h=0}^{\nu} G_{\ell,h}^{(j)} \; e_{t-h}^{(\ell)} \quad \text{for } k < j \leq n. \tag{2.15}$$

In general usage, syndromes are quantities computed <u>from</u> noisy received data but are functions <u>of</u> errors only. Thus it is evident that the syndrome digits above are functions <u>only</u> of the errors introduced by the BSC.

For the binary rate 1/n systematic code case the syndrome digits are

$$Z_t^{(j)} = e_t^{(j)} + \sum_{h=0}^{\nu} G_h^{(j)} \; e_{t-h}^{(1)} \quad \text{for } k < j \leq n. \tag{2.16}$$

A vector notation common in later sections for rate 1/n codes is

$$\underline{Z}_t = \begin{bmatrix} Z_t^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ Z_t^{(n)} \end{bmatrix} \qquad (2.17)$$

For the previously mentioned K = 5 rate $\frac{1}{2}$ example of Eqs. (2.5) and (2.6) and Figure 2.5 there is only one syndrome digit equation:

$$Z_t^{(j)} = e_t^{(j)} + \sum_{h=0}^{\nu} G_h^{(j)} e_{t-h}^{(1)} \quad \text{(i.e. j=2 only).} \qquad (2.18)$$

Thus,

$$Z_t^{(2)} = e_t^{(2)} + G_0^{(2)} e_t^{(1)} + G_1^{(2)} e_{t-1}^{(1)} + G_2^{(2)} e_{t-2}^{(1)} + G_3^{(2)} e_{t-3}^{(1)} + G_4^{(2)} e_{t-4}^{(1)}$$

$$= e_t^{(2)} + e_t^{(1)} + e_{t-1}^{(1)} + e_{t-2}^{(1)} + e_{t-4}^{(1)} \qquad (2.19)$$

The actual computation of Eq. (2.18) or Eq. (2.19) is accomplished by a syndrome former whose physical inputs are received digits $r_t^{(j)}$ $1 \le j \le n$ and whose outputs are syndrome digits $Z_t^{(j)}$ $k < j \le n$. However, since Eq. (2.15) has shown that the syndrome digits are independent of the actual code bits transmitted, the analysis of the syndrome former can be simplified by considering its inputs to be only error bits $e_t^{(j)}$, $1 \le j \le n$ introduced by the BSC. For example, Figure 2.7a shows the actual physical syndrome former implementation for the

K = 5 rate $\frac{1}{2}$ code mentioned previously. Figure 2.7b on the other hand shows the syndrome former that will be analyzed in lieu of that in Figure 2.7a. One can also refer to Figure 2.8 which indicates just where the syndrome former is used in relation to other components of the convolutional code communication system.

Since Eq.(2.16) applies at a particular time t at the decoder, one can write a syndrome equation set for any code by considering all t values in some span, $t_1 \leq t \leq t_2$. For the example K = 5 code, if one considers time values including the current time t and 5 prior time instants, the following syndrome equation set is obtained:

$$
\begin{aligned}
Z_{t-5} &= e^{(2)}_{t-5} + e^{(1)}_{t-5} + e^{(1)}_{t-6} + e^{(1)}_{t-7} + 0 + e^{(1)}_{t-9} \\
Z_{t-4} &= e^{(2)}_{t-4} + e^{(1)}_{t-4} + e^{(1)}_{t-5} + e^{(1)}_{t-6} + 0 + e^{(1)}_{t-8} \\
Z_{t-3} &= e^{(2)}_{t-3} + e^{(1)}_{t-3} + e^{(1)}_{t-4} + e^{(1)}_{t-5} + 0 + e^{(1)}_{t-7} \\
Z_{t-2} &= e^{(2)}_{t-2} + e^{(1)}_{t-2} + e^{(1)}_{t-3} + e^{(1)}_{t-4} + 0 + e^{(1)}_{t-6} \\
Z_{t-1} &= e^{(2)}_{t-1} + e^{(1)}_{t-1} + e^{(1)}_{t-2} + e^{(1)}_{t-3} + 0 + e^{(1)}_{t-5} \\
Z_{t} &= e^{(2)}_{t} + e^{(1)}_{t} + e^{(1)}_{t-1} + e^{(1)}_{t-2} + 0 + e^{(1)}_{t-4}
\end{aligned}
\tag{2.20}
$$

Note how a particular error digit (e.g. $e^{(1)}_{t-5}$) influences a span of syndrome digits (in this case Z_i for $t-1 \leq i \leq t-5$). This will be significant in later decoding steps where it will be convenient to define

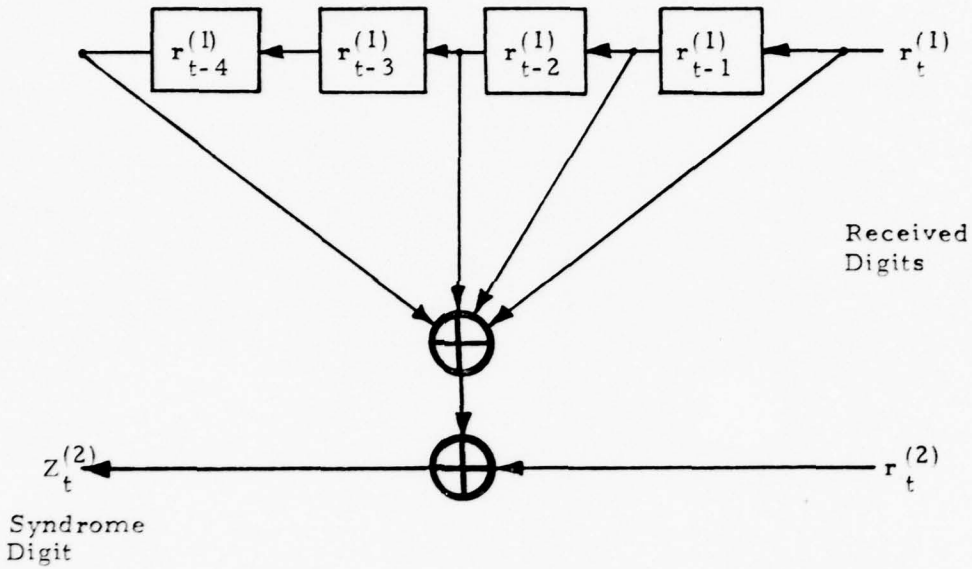


Figure 2.7a Actual Syndrome Former

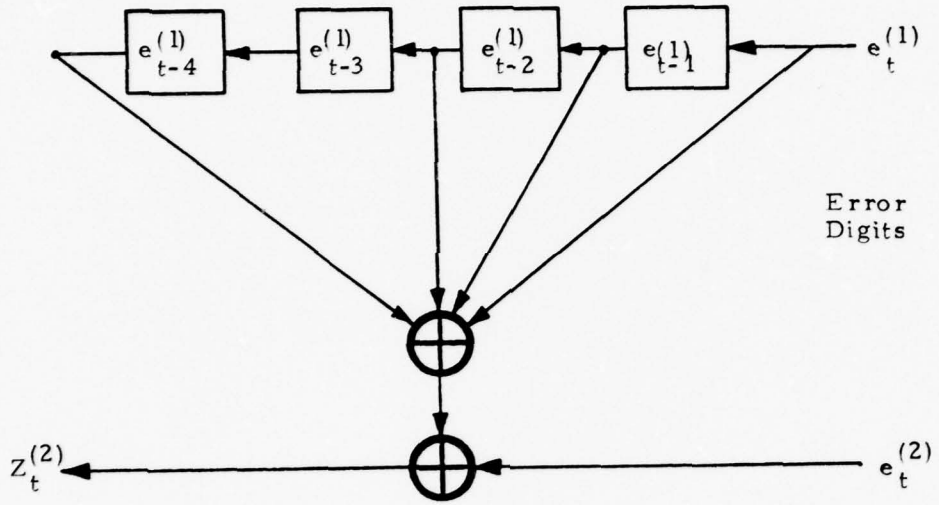


Figure 2.7b Syndrome Former Analyzed

a syndrome vector $\underline{Z}$ as consisting of all syndromes on the left hand side of the set of syndrome equations delimited by a specified time span. Thus for the above example the syndrome vector is

$$\underline{Z} = (Z_{t-5}, Z_{t-4}, \ldots, Z_t) = Z_{(t-5,\,t)} = Z_{t-5}^{t} \qquad (2.21)$$

where four equivalent notations which will appear later are given. The number of bits in a given syndrome vector $\underline{Z}$ will be denoted as either $|Z|$ or LSYN in following work. Note that a particular span of syndrome equations such as Eq. (2.20) specifies a set of error digits on the right hand side. It will be convenient to define an overall error vector, $\underline{E}'$, as the total set of error digits appearing in such a span of equations. When later discussing single error bit estimation, the error vector, $\underline{E}$, will be used to represent the set of such error digits less the particular bit to be estimated. The number of error bits in $\underline{E}$ will be identified as LEN. Thus the number of error bits in $\underline{E}'$ will be LEN + 1.

While on the subject of syndromes, it will prove useful to later work to here define the syndrome former state, S_t, as being the binary vector represented by the logical values present in the physical storage elements of the syndrome former at time t. Thus one could represent the operation of the syndrome former in terms of a syndrome former state diagram similar to that defined for the encoder. For example the K = 5 rate $\frac{1}{2}$ syndrome former of Figure 2.7 could

be represented as a state diagram having $2^{\nu} = 2^4$ states. The syndrome former state transitions are determined in this example by the value of $e_t^{(1)}$. Thus the syndrome former state transition probability $P(S_{t+1}/S_t)$ for the two allowable transitions is just equal to $P(e_t^{(1)} = 0)$ and $P(e_t^{(1)} = 1)$ respectively.

However the syndrome former output probability is more complex than in the encoder case since the output Z_t depends not only on the current state S_t and the input $e_t^{(1)}$ but also upon $e_t^{(2)}$ (see Figure 2.7b). This complication will be considered in detail in Chapter 3. For now it is sufficient to note that the conditional syndrome former output probability, $P(Z_t/S_t, S_{t+1})$ is independent of all prior syndrome former states S_j for $j < t$. Therefore,

$$P(Z_t/S_0, S_1, \ldots, S_t, S_{t+1}) = P(Z_t/S_t, S_{t+1}) \qquad (2.22)$$

Thus the syndrome former also displays a Markov property analogous to that of the encoder. This property will be exploited in Chapter 3 to simplify analysis of optimal decoding by error estimation using a syndrome former.

2.4 Decoding

Convolutional encoding has been shown in Section 2.2 to be relatively simple to implement. Decoding of convolutional codes is somewhat more difficult. The goal of a decoder is to provide to the data user an estimate of the data source output bit I_t or sequence

$\underline{I}_{(t_1, t_2)}$. Probabilistic decoding is a method of doing so by processing information available at the decoder and outputing as an estimate that value $\hat{I}_t$ (or $\hat{\underline{I}}$) with the highest probability of occurrence conditioned on the information available to the decoder.

Since the noise is additive (i.e. $\underline{R}_t = \underline{C}_t + \underline{E}_t$), there are two complementary approaches to such estimation. In direct data estimation the most probable I_t (or $\underline{I}$) is determined conditioned on available information (e.g. a received vector $\underline{R}$). An equivalent form of data estimation first produces code digit (or code sequence) estimates then deterministically estimates corresponding databits (or sequences) by solving Eq. (2.4). In error estimation the most probable additive transmission errors are estimated conditioned on available information and then subtracted from the appropriate received data to provide code digit C_t or code sequence $\underline{C}$ estimates. With such error estimates, Eq. (2.4) then provides a means of determining actual data bit or sequence estimates. Figure 2.8 is an example of a case in which the available information to the decoder consists of syndrome bits which are used to form the error estimate $\hat{e}_t$.

Several probabilistic approaches to bit and sequence decoding are qualitatively discussed in the next two sections to show the distinction between data estimation and error estimation. Error approaches explored in this work are noted.

2.4.1 Data Estimation

A number of researchers have investigated optimal approaches to decoding convolutional codes by data estimation. In (1) through (4) below their goal was to estimate the encoded information bit I_t or sequence $I_{(1,N)}$ by processing the entire finite received sequence $R_{(1,N+\nu)}$.

(1) Conventional Viterbi Decoding selects the information sequence $\hat{I}_{(1,N)}$ to be the information sequence which maximizes

$$P(I_{(1,N)} / R_{(1,N+\nu)})$$

where equilikely input sequences are assumed. Consequently, this approach is Maximum Likelihood Sequence Decoding. (Viterbi [5]).

(2) Map Sequence Decoding selects the information sequence $\hat{I}_{(1,N)}$ to be the information sequence which maximizes

$$P\left(I_{(1,N)} / R_{(1,N+\nu)}\right) = \frac{P\left(I_{(1,N)}, R_{(1,N+\nu)}\right)}{P\left(R_{(1,N+\nu)}\right)}$$

where nonequilikely input sequences are considered. If the input sequences are equilikely, conventional Viterbi decoding is obtained (McAdam [9]).

(3) Map Bit Decoding selects the information bit $\hat{I}_t$ to be that information bit which maximizes

$$P(I_t / R_{(1,N+\nu)}) = \frac{P(I_t, R_{(1,N+\nu)})}{P_{(R(1,N+\nu))}}$$

(McAdam [9]).

(4) Maximum Likelihood (ML) Bit Decoding is the special case of map bit decoding when information bits are equilikely.

Since it may not be possible or desirable in practice to process the entire received sequence, as required for the optimal approaches (1) through (4) above, some work has appeared in which a finite storage length of the possibly infinite length received sequence was used to estimate either an information bit $\hat{I}_t$ (or $\hat{I}_{t-\Delta}$) or information sequence $\hat{I}_{(1,t)}$. This "real time constraint" on the information bit/sequence estimation has been considered by the following two approaches.

(5) Real Time Viterbi Sequence Decoding seeks that information sequence $\hat{I}_{(1,\ t+\Delta)}$ which maximizes

$$P(I_{(1,\ t+\Delta)} / R_{(1,\ t+\Delta)})$$

and sets $\hat{I}_t$ = t-th bit of sequence $\hat{I}_{(1,\ t+\Delta)}$

This is a finite delay version of the conventional Viterbi Algorithm for equilikely input sources (Lee [11]).

(6) Real Time Minimum Bit Error (RTMBEP) Decoding seeks that information bit $\hat{I}_t$ which maximizes

$$P(I_t / R_{(1,\ t+\Delta)}).$$

This is a finite delay version of McAdam Map Bit Decoding, specialized to equilikely input bits. (Lee [11]).

These six approaches sought a particular type of estimate of the information bit $\hat{I}_t$ or sequence $\hat{I}_{(1,\ t)}$ derived based on the receipt and availability of some received vector or vector segment (e.g. $R_{(1,\ t+\Delta)}$).

2.4.2 Error Estimation

Instead of estimating the information bit/sequence as above, one might consider estimating error bits $\hat{e}_t$ or sequences $\hat{e}_{(1,\ t)}$ and obtaining an estimate of the information bits by subtracting the estimated error bits from the appropriate received bits. In such an approach, a Syndrome Former could be used to produce syndrome bits Z_t from the received vector $\underline{R}_t$, thereby making available to the decoder a vector $\underline{Z}_{(1,\ t)}$ which is independent of the information bits or sequences actually sent and dependent only on the channel errors which have occurred. Then an optimal estimate of the error bit $\hat{e}_t$ or sequence $\hat{e}_{(1,\ t)}$ could be derived analogous to the approaches taken in the case of information bit/sequence estimation.

One complicating factor in such an error estimation approach is that channel errors are not equilikely (otherwise it would be senseless to try to transmit information over such a channel). Therefore, Maximum Likelihood (ML) estimation must not be used. Rather, a Maximum A Posteriori Probability (MAP) estimate should be implemented which takes into account the nonequilikely apriori probabilities of channel error values which are related to the E_s/N_o level in the

applicable channel.

Another source of concern in the error estimation approach would be the number of error bits or sequences to (potentially) be estimated. For example, in conventional information bit decoding of a rate 1/3 convolutional code, only an estimate of a single information bit (e.g. I_t) is required. In error estimation for this code, on the surface it might seem necessary to consider estimating as many as 3 error bits. It will be seen that it is not always necessary to estimate all error bits. Specifically, in a rate 1/3 systematic code it is obvious that only a single error bit estimate is required. The error estimation approaches in this work have been derived for rate 1/n systematic codes. However, the extension of such probabilistic decoding to non-systematic codes is relatively straightforward although this requires estimation of more than a single error bit.

In this investigation the error estimation approach has been pursued and the following four estimates have been derived.

(1) Map Error Bit Estimate Decoding which selects as the error bit estimate $\hat{e}_t$ that error bit value which maximizes

$$P(e_t / Z_{(1,\, N+\nu)}) .$$

This decoding approach is basically an adaptation of the approach used by McAdam [9] and Bahl et al [10] for MAP data bit estimation.

(2) Map Error Sequence Decoding which selects that error sequence $\hat{e}_{(1, N)}$ which maximizes

$$P(e_{(1, N)} / Z_{(1, N+\nu)}) .$$

This approach can be viewed in general as an adaptation of Viterbi decoding to the nonequilikely error sequence decoding case. Also pursued is a conversion of a specific rate 1/3 systematic code error sequence decoding problem to a rate 1/2 non-systematic decoding configuration in which the minimum Hamming weight error sequence approach of Schalkwijk and Vinck [13] is useful.

Approaches (1) and (2) assume the availability of the entire received sequence $R_{(1, N+\nu)}$ by assuming the availability of the entire syndrome vector $\underline{Z}_{(1, N+\nu)}$ in (1) and of essentially infinite storage in (2). These are developed in detail in Chapter 3.

Approaches (3) and (4) deal with the potentially more practical case of finite delay-finite storage (i.e. "real time") MAP error decoding.

(3) Real Time MAP Error Bit Decoding which selects that error bit value $\hat{e}_t$ which maximizes

$$P(e_t / Z_{(1, t+\Delta)}).$$

This is essentially an adaptation for error estimation of Lee's RTMBEP Decoding of information bits [11].

(4) Real Time Viterbi MAP Error Sequence Decoding which selects that error sequence $\hat{e}_{(1,\ t+\Delta)}$ which maximizes

$$P(e_{(1,\ t+\Delta)} / Z_{(1,\ t+\Delta)})$$

and selects $\hat{e}_t$ as the t-th bit of this sequence. This is basically an adaptation for error estimation of Lee's Real Time Viterbi Decoding [11] Approaches (3) and (4) are developed in detail in Chapter 3.

Note that all of the above error estimation approaches have been MAP estimates. This is due to the nonequilikely nature of the channel errors.

After considering the above "optimal" approaches to MAP error estimation, a suboptimal method of MAP error estimation has been derived in Chapter 4 which eliminates most active computations during decoding. This ROMMAP approach selects as the error bit estimate $\hat{e}_{t-\Delta}$ that error bit value which maximizes

$$P(e_{t-\Delta} / Z_{(t-\Delta,\ t)}) .$$

(Basically, this is a finite delay (Δ) - finite storage (L) MAP estimate).

The ROMMAP is a suboptimal attempt at implementing finite delay-finite storage ("real time") MAP Error Bit Decoding (see (3) above) since, to obtain a readily computable ROMMAP look-up table, that portion of the total syndrome vector $Z_{(1,\ t)}$ from initialization to

time t - L - 1 (i.e. $Z_{(1,\ t - L - 1)}$ has been neglected. Consequently, the ROMMAP performance should be inferior to the optimal real time.

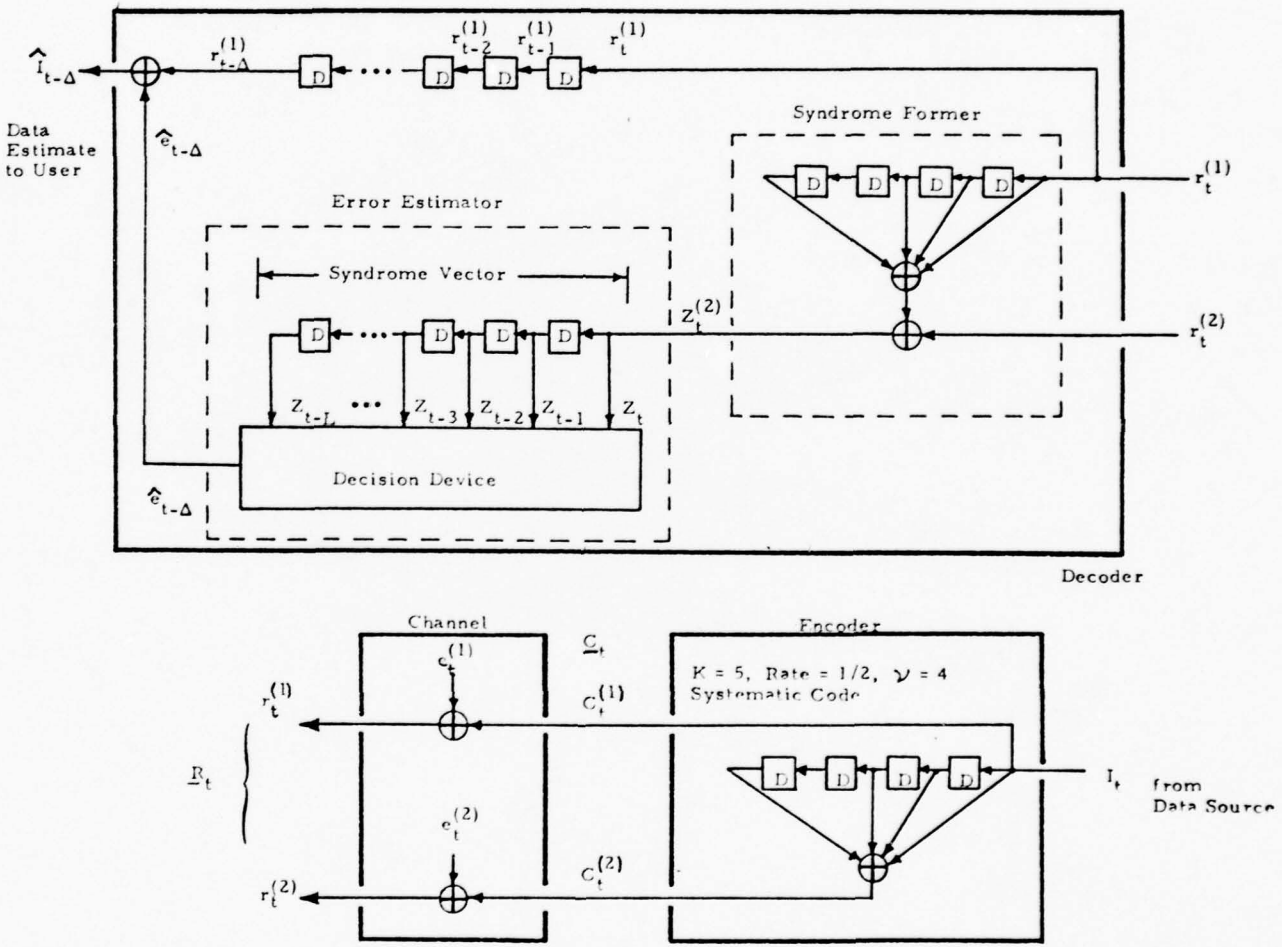


Figure 2.8. Convolutionally Encoded Communication System

Chapter 3

OPTIMAL ERROR ESTIMATION

In this chapter optimal decoding of convolutional codes via error estimation is explored in some depth. Several estimation techniques previously used only for data estimation are adapted and applied to the problem of error estimation based on syndromes. First a recursive MAP error bit estimation technique is developed for the case in which the entire syndrome vector $\underline{Z}_{(1,\tau)}$ is available (i.e. no finite delay constraint exists). Next, a MAP error sequence estimator with some analogy to the Viterbi maximum likelihood sequence decoding algorithm is analyzed. Then a somewhat more realistic case of optimal MAP error bit decoding with a finite delay constraint is derived in detail. For completeness, a corresponding finite delay Viterbi MAP error sequence technique is exhibited. Finally, the relationship between MAP error bit decoding using a syndrome $\underline{Z}$ and MAP data bit decoding using the received vector $\underline{R}$ is looked at from a conceptual viewpoint. This chapter contains the theoretically optimal approaches to error estimation which serve as the base from which feasible suboptimal implementations of error decoders are obtained in Chapter 4.

3.1 Recursive MAP Error Bit Estimation

To implement a MAP estimate of an error bit e_t given an

available syndrome vector $\underline{Z}$ one would normally seek to select that value $\hat{e}_t$ of e_t so as to maximize

$$P(e_t/\underline{Z}) = \frac{P(\underline{Z}/e_t)P(e_t)}{P(\underline{Z})} \qquad (3.1)$$

Suppose one attempts to do this via a recursive estimate of $e_t^{(j)}$ given the total syndrome vector $\underline{Z}_1^{\tau}$ produced as a result of syndrome bit computation based on the total vector of received bits $\underline{R}$ representing an input information stream of length T followed by ν zeros to drive the convolutional encoder back to the all zero state. We can do so by using the fact that the syndrome former is a linear sequential circuit whose output Z_t depends only on the current syndrome former physical state S_t and the incoming error bits at time t, namely E_t. Here $\tau = T + \nu$.

Unlike the encoder itself, the syndrome former does not necessarily store all input error bits in E_t. Therefore in general one cannot say that the pair of states S_t, S_{t+1} defined by actual physical storage elements determine an output Z_t. For example, in the rate 1/2 systematic code example the error bit $e_t^{(2)}$ contributes to the syndrome output Z_t but does not appear in any component of the physical state S_t represented by the actual contents of the storage elements in the syndrome former.

Both McAdam [9] and Bahl et al. [10] derived MAP estimates of information bits given the total received vector $\underline{R}$. In the following,

the approach and notation of Bahl will be applied to the derivation of a MAP estimate of error bits $e_t^{(j)}$ given the total computed syndrome vector $\underline{Z}_1^{\tau}$. As usually done (e.g. Schalkwijk and Vinck [13]) we consider the syndrome former inputs to be error bits only since the syndrome former output is a function of these errors only and not of the code bits which also enter in R_t.

For the sake of clarity we define the following assumption and later show how although it was true in Bahl's situation it does not have to be true for the syndrome case at critical steps.

Assumption #1: The syndrome former has physical storage elements (whose contents we call S_t) which contain all components of R_t. With this assumption we can treat the syndrome former as a Markov source whose state pairs S_t, S_{t+1} determine Z_t and proceed as in Bahl's work. We will note if and when this assumption is actually required in the syndrome case.

The next figure shows a block diagram comparison between Bahl's problem and ours:

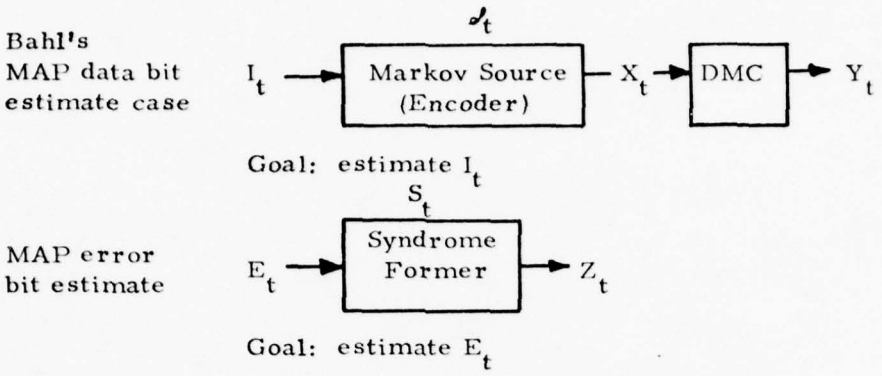


Figure 3.1. Comparison of Data and Error Estimation Approaches.

We now follow Bahl's notation and estimate the a posteriori probability of the states and transitions of a syndrome former. In the following, m is one of the 2^{ν} possible binary values of a state.

Source Transition Probabilities

Syndrome former state transition probabilities may be expressed as:

$$p_t(m/m') = Pr(S_t=m/S_{t-1}=m') \tag{3.2}$$

Our syndrome former state transition probabilities are governed by the statistics of the input n-tuple,

$$E(t) = \begin{bmatrix} e_t^{(1)} \\ e_t^{(2)} \\ \vdots \\ e_t^{(n)} \end{bmatrix} . \tag{3.3}$$

through

$$\Pr(S_t=m/S_{t-1}=m') = \sum_{E_t} \Pr(S_t=m, E_t/S_{t-1}=m')$$

$$= \sum_{E_t} \Pr(S_t=m/E_t, S_{t-1}=m')\Pr(E_t/S_{t-1}=m') \qquad (3.4)$$

Since E_t is *not* a function of $S_{t-1}=m'$ we have

$$\Pr(S_t=m/S_{t-1}=m') = \sum_{E_t} \Pr(S_t=m/E_t, S_{t-1}=m')\Pr(E_t) \qquad (3.5)$$

Since the channel is memoryless, the individual bits in E_t are bit-to-bit independent and one could let $\Pr(E_t) = \prod_{j=1}^{n} \Pr(e_t^{(j)})$. The first factor on the right is just 0 or 1 depending on whether the transition is possible and that the particular E_t being considered in the sum is that E_t which causes such a transition (or one of the several E_t which can cause the same transition in the case of some don't care elements $e_t^{(j)}$ in E_t).

The don't care elements can be eliminated from the above sum by defining

$E_t^* \triangleq$ (those $e_t^{(j)}$ which actually enter into physical syndrome state storage elements)

yielding:

$$\Pr(S_t=m/S_{t-1}=m') = \sum_{E_t^*} \Pr(S_t=m/E_t^*, S_{t-1}=m')\Pr(E_t^*) . \qquad (3.6)$$

Output Probabilities

Syndrome former output probabilities may be written as:

$$q_t(Z/m', m) = Pr(Z_t = Z/S_{t-1} = m', S_t = m). \qquad (3.7)$$

Note that $q_t(Z/m', m) = 0$ or 1 when Assumption #1 is true since S_{t-1}, S_t together contain exactly those error bits needed to determine Z_t. If Assumption #1 is not true, then $q_t(Z/m', m)$ would not necessarily be 0 or 1 but would be dependent on other error bits which never reside in the physical state. However in such a case one can average over these other error bits to still obtain:

$$q_t(Z/m', m) = \sum_{\delta_t} Pr(Z_t = Z/S_{t-1} = m', S_t = m, \delta_t = \delta) Pr(\delta_t = \delta) \qquad (3.8)$$

where $\delta_t \triangleq$ all those error bits in E_t which affect the syndrome Z_t but which are not present in the actual physical state S_{t-1}, S_t.

Objective of the Decoder

The objective of the decoder is to examine Z_1^τ and estimate the a posteriori probability of the syndrome former states:

$$Pr(S_t = m/Z_1^\tau) = \frac{Pr(S_t = m, Z_1^\tau}{Pr(Z_1^\tau)} \triangleq \frac{\lambda_t(m)}{Pr(Z_1^\tau)} \qquad (3.9)$$

and transitions

$$Pr(S_{t-1} = m', S_t = m/Z_1^\tau) = \frac{Pr(S_{t-1} = m', S_t = m, Z_1^\tau)}{Pr(Z_1^\tau)} \triangleq \frac{\sigma_t(m', m)}{Pr(Z_1^\tau)}. \qquad (3.10)$$

Now derive the joint probabilities defined above, namely,

$$\lambda_t(m) \triangleq \Pr(S_t=m, Z_1^\tau) \text{ and} \tag{3.11a}$$

$$\sigma_t(m',m) = \Pr(S_{t-1}=m', S_t=m, Z_1^\tau). \tag{3.11b}$$

To simplify notation, define

$$\alpha_t(m) \triangleq \Pr(S_t=m, Z_1^t) \tag{3.12a}$$

$$\beta_t(m) \triangleq \Pr(Z_{t+1}^\tau / S_t=m) \tag{3.12b}$$

$$\gamma_t(m',m) \triangleq \Pr(S_t=m, Z_t / S_{t-1}=m') \tag{3.12c}$$

Now

$$\lambda_t(m) = \Pr(S_t=m, Z_1^\tau) = \Pr(S_t=m, Z_1^t, Z_{t+1}^\tau)$$

$$= \Pr(S_t=m, Z_1^t)\Pr(Z_{t+1}^\tau / S_t=m, Z_1^t) \tag{3.13}$$

Note that the statistics of Z_{t+1}^τ are not dependent on past syndrome bits when the condition on S_t is applied. For example Z_{t+1}^τ is a function of S_t and E_{t+1} but $\mathbf{Z}_1^t$ doesn't provide any additional information. Therefore,

$$\lambda_t(m) = \Pr(S_t=m, Z_1^t)\Pr(Z_{t+1}^\tau / S_t=m)$$

$$\lambda_t(m) = \alpha_t(m)\beta_t(m) \tag{3.14}$$

Similarly,

$$\sigma_t(m',m) = \Pr(S_{t-1}=m', S_t=m, Z_1^\tau) = \Pr(S_{t-1}=m', S_t=m, Z_1^{t-1}, Z_t, Z_{t+1}^\tau)$$

$$= \Pr(S_{t-1}=m', Z_1^{t-1})\Pr(S_t=m, Z_t / S_{t-1}=m', Z_1^{t-1})$$

$$\cdot \Pr(Z_{t+1}^\tau / S_{t-1}=m', Z_1^{t-1}, S_t=m, Z_t) \tag{3.15}$$

Here use has been made of the fact that:

(1) $S_t=m$, Z_t are dependent on $S_{t-1}=m'$ but <u>not</u> on Z_1^{t-1} (past syndrome bits) when conditioned on $S_{t-1}=m'$; and

(2) Z_{t+1}^{τ} when conditioned on $S_t=m$ is not statistically dependent on $S_{t-1}=m'$ (past state), Z_1^{t-1} (past syndrome bits), or Z_t (immeidately past syndrome bits).

Therefore we can write the joint probability as

$$\sigma_t(m', m) = \Pr(S_{t-1}=m', Z_1^{t-1})\Pr(S_t=m, Z_t/S_{t-1}=m')\Pr(Z_{t+1}^{\tau}/S_t=m)$$

$$\sigma_t(m', m) = \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m) \qquad (3.16)$$

<u>Forward Recursion</u>

Now for $t = 1, 2, 3, \ldots, \tau$ a forward recursion can be derived as follows.

$$\alpha_t(m) = \Pr(S_t=m, Z_1^t) = \sum_{\text{all } S_{t-1}=m'} \Pr(S_{t-1}=m', S_t=m, Z_1^t)$$

$$= \sum_{\text{all } S_{t-1}=m'} \Pr(S_{t-1}=m', S_t=m, Z_1^{t-1}, Z_t)$$

$$= \sum_{\text{all } S_{t-1}} \Pr(S_{t-1}=m', Z_1^{t-1})\Pr(S_t=m, Z_t/S_{t-1}=m', Z_1^{t-1}) \qquad (3.17)$$

Using fact (1) above one obtains:

$$\alpha_t(m) = \sum_{\text{all } S_{t-1}} \Pr(S_{t-1}=m', Z_1^{t-1}) \Pr(S_t=m, Z_t/S_{t-1}=m')$$

$$\alpha_t(m) = \sum_{\text{all } S_{t-1}} \alpha_{t-1}(m') \gamma_t(m', m) \qquad (3.18)$$

At t=0 we have boundary conditions: $\alpha_0(0) = 1$, $\alpha_0(m) = 0$ for all $m \neq 0$ since the syndrome former S_0 state is known to be 0.

Backward Recursion

In a similar manner the backward recursion may be obtained as follows:

$$\beta_t(m) = \Pr(Z_{t+1}^{\tau}/S_t=m) = \sum_{\text{all } S_{t+1}} \Pr(S_{t+1}=m', Z_{t+1}^{\tau}/S_t=m)$$

$$= \sum_{\text{all } S_{t+1}} \Pr(S_{t+1}=m', Z_{t+1}, Z_{t+2}^{\tau}/S_t=m)$$

$$= \sum_{\text{all } S_{t+1}} \Pr(S_{t+1}=m', Z_{t+1}/S_t=m) \Pr(Z_{t+2}^{\tau}/S_{t+1}=m', Z_{t+1}, S_t=m) \qquad (3.19)$$

Again we can state that Z_{t+2}^{τ} when conditioned on $S_{t+1}=m'$ is statistically independent of Z_{t+1} (past syndrome bits) and of $S_t=m$ (past syndrome former state).

Therefore,

$$\beta_t(m) = \sum_{\text{all } S_{t+1}} \beta_{t+1}(m') \gamma_{t+1}(m, m') \qquad (3.20)$$

with boundary conditions $\beta_\tau(0) = 1$, $\beta_\tau(m) = 0$ for all $m \neq 0$ since we have assumed that we drive the syndrome former to $\underline{0}$ state once we reach the point in time when it is known that only the tail of ν zero bits are being encoded at the transmitter. Now

$$\gamma_t(m', m) \triangleq \Pr(S_t=m, Z_t / S_{t-1}=m')$$

$$= \Pr(S_t=m / S_{t-1}=m') \Pr(Z_t / S_{t-1}=m', S_t=m)$$

$$\gamma_t(m', m) = p_t(m/m') q_t(Z_t / m', m) \tag{3.21}$$

Note that we do not have to average over any intermediate X since in the syndrome case we have no X_t as was present in Bahl's problem situation.

Operation of Decoder

The decoder must compute the following.

$$\lambda_t(m) = \Pr(S_t=m, Z_1^\tau) = \alpha_t(m)\beta_t(m)$$

Syndrome Former State Joint Probability (3.22)

$$\sigma_t(m', m) = \Pr(S_{t-1}=m', S_t=m, Z_1^\tau) = \sum_{S_{t-1}=m'} \alpha_{t-1}(m')\gamma_t(m', m)$$

State Transition Joint Probability (3.23)

These computations may be accomplished in the following steps:

Step #1:

$\alpha_0(m), \beta_\tau(m)$ $m = 0, 1, \ldots, M-1$ are initialized $(M = 2^\nu)$.

$$\left.\begin{aligned} &\alpha_0(0)=1, \alpha_0(m)=0 \text{ for all } m\neq 0 \\ &\beta_\tau(0)=1, \beta_\tau(m)=0 \text{ for all } m\neq 0 \end{aligned}\right\} \text{ Assuming Initial and Final States } \underline{\text{Known}} \tag{3.24}$$

Step #2:

As soon as each Z_t is obtained compute

$$\gamma_t(m', m) = p_t(m/m')q_t(Z_t/m', m) = \Pr(S_t=m, Z_t/S_{t-1}=m') \tag{3.25}$$

Next compute

$$\alpha_t(m) = \sum_{S_{t-1}=m'} \alpha_{t-1}(m')\gamma_t(m', m) \tag{3.26}$$

Step #3:

After the complete sequence Z_1^τ *has been received, recursively* compute

$$\beta_t(m) = \sum_{m'} \beta_{t+1}(m')\gamma_{t+1}(m, m') \tag{3.27}$$

and multiply by the stored $\alpha_t(m)$ to obtain

$$\lambda_t(m) = \alpha_t(m)\beta_t(m) \tag{3.28}$$

and by $\gamma_t(m', m)$ and $\alpha_{t-1}(m')$ to obtain

$$\sigma_t(m', m) = \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m). \tag{3.29}$$

Error Bit Estimation

Having determined $\lambda_t(m)$ and $\sigma_t(m', m)$ for all $0 \le t \le \tau$, now estimate the required error bits $\{e_t^{(j)}\}$. Assume that the error

bits required to appear explicitly in the syndrome former physical state.

Let $A_t^{(j)}$ be the set of syndrome former states S_t such that $s_t^{(j)} = 0$ where $s_t^{(j)}$ is the value of $e_t^{(j)}$ after it is shifted into the associated syndrome former storage element.

Then $s_t^{(j)} = e_t^{(j)}$ for $e_t^{(j)} \epsilon E^*$ where $E^* \triangleq$ [those $e_t^{(j)}$ which must be estimated and appear explicitly in syndrome former state]. This implies

$$\Pr(e_t^{(j)} = 0, Z_1^{\tau}) = \sum_{S_t \epsilon A_t^{(j)}} \lambda_t(m) . \tag{3.30}$$

Normalizing by

$$\Pr(Z_1^{\tau}) = \lambda_{\tau}(0) , \tag{3.31}$$

the conditional probability,

$$\Pr(e_t^{(j)}=0/Z_1^{\tau}) = \frac{1}{\lambda_{\tau}(0)} \sum_{S_t \epsilon A_t^{(j)}} \lambda_t(m), \tag{3.32}$$

is obtained. Now <u>decode the error bit</u> by the rule:

$$\hat{e}_t^{(j)} = 0 \text{ if } \Pr(e_t^{(j)}=0/Z_1^{\tau}) \geq 0.5 \tag{3.33}$$

$$\hat{e}_t^{(j)} = 1 \quad \text{otherwise.}$$

Thus the MAP estimate of a desired error bit $e_t^{(j)}$ given the entire syndrome vector Z_1^{τ} has been obtained on a recursive basis.

It is seen that Assumption #1 is not needed for this error bit estimation as long as the error bit to be estimated does appear in the syndrome former physical state.

3.2 Syndrome Decoding of Rate 1/3 Systematic Codes

In the case of rate 1/n systematic convolutional codes the syndrome based decoder does not have to estimate all error bits in

$$E_t = \begin{bmatrix} e_t^{(1)} \\ e_t^{(2)} \\ \vdots \\ e_t^{(n)} \end{bmatrix} \tag{3.34}$$

The other error bits $e_t^{(2)}, e_t^{(3)}, \ldots, e_t^{(n)}$ can be viewed as nuisance parameters which in MAP estimation should be averaged out in the computation of $\hat{e}_t^{(1)}$. This was the approach followed in the previous work on recursive MAP error bit estimation.

A different approach to the decoding problem exists in which estimation of the most probable error sequence is the goal of the decoder. Recall that classical Viterbi Maximum Likelihood data sequence decoding assumes equilikely input data sequences. Since errors are very nonequilikely, Appendix A has been provided to show in general how the basic Viterbi sequence decoding metrics would have to be modified to account for nonequilikely inputs such as error sequences.

Schalkwijk and Vinck [13] have derived and demonstrated a recursive algorithm like Viterbi's which (for rate 1/2 non-systematic codes) finds the error sequence of minimum Hamming weight which could have caused the observed syndrome sequence. This minimum weight error sequence is the most probable error sequence given the observed syndrome sequence. Considering the work that has been done [13] in obtaining minimum weight error sequences, the discussion that follows merely shows how a specific rate 1/3 systematic code error sequence decoding problem may be converted to a rate 1/2 non-systematic error sequence decoding problem suitable for solution by the method of Schalkwijk and Vinck. The rate 1/3 systematic code of this example was selected from Bussgang's table of optimum subgenerators of a canonical subgenerator (Fig. 7 in [19]).

Consider the syndrome former for a rate 1/3 systematic code shown in Figure 3.2a. The goal here is to estimate the error sequence $(e_0^{(1)}, e_1^{(1)}, \ldots, e_{N-1}^{(1)})$ based on the syndrome sequences $s_{(0,N-1)}^{(2)}$ and $s_{(0,N-1)}^{(3)}$ where N is the length of the entire output sequence from the data source. Figure 3.2a can be redrawn as in Figure 3.2b to more clearly isolate the "nuisance" errors $e_t^{(2)}$ and $e_t^{(3)}$. One can thus observe that the problem of estimating the desired error sequence $e_0^{(1)}, \ldots, e_{N-1}^{(1)}$ is equivalent to estimating the "data" input sequence to a rate 1/2 convolutional "error encoder" of

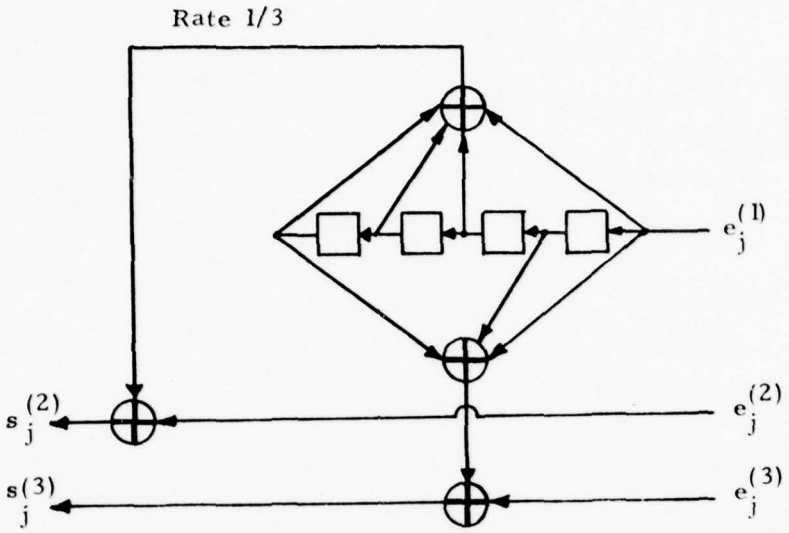


Figure 3.2a "First Syndrome Former" with Error Inputs

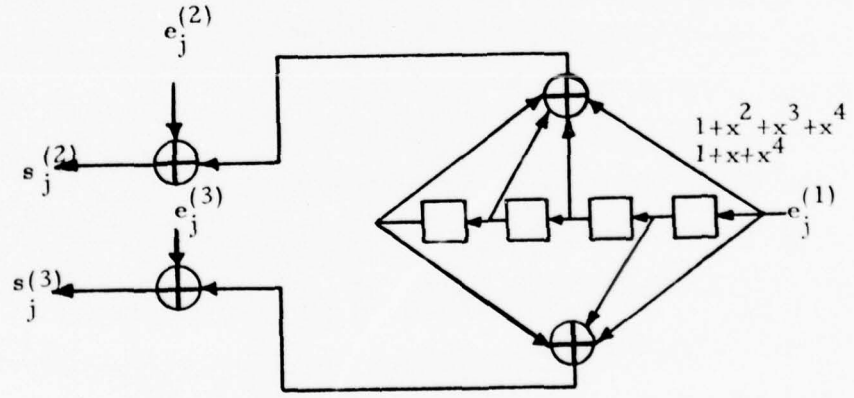


Figure 3.2b Alternate View of Syndrome Former

Figure 3.2b. Although this appears to have reduced the rate 1/3 problem to a possibly more manageable rate 1/2 one, normal Viterbi sequence decoding cannot be used to estimate the error input. The classical Viterbi algorithm is a maximum likelihood decoder for equilikely input sequences while here the input is a very non-equilikely sequence of error bits $e_t^{(1)}$.

Now instead of pursuing a modified Viterbi algorithm with its more complicated metric calculations for nonequilikely error input sequences (see Appendix A), suppose error sequence decoding using the method of Schalkwijk and Vinck [13] is followed for the case shown in Figure 3.2b.

To keep things straight, call the units of Fig. 3.2a and 3.2b the First Syndrome Former (error encoder). For this error encoder there exists a corresponding Second Syndrome Former shown in Fig. 3.3a.

The operation of this Second Syndrome Former on input syndrome bits $s_t^{(2)}$ and $s_t^{(3)}$ can be viewed as the equivalent operation on error bits $e_t^{(2)}$ and $e_t^{(3)}$ as shown in Fig. 3.3b. The rate 1/3 problem has thus been converted to a rate 1/2 problem exactly like that of Schalkwijk and Vinck [13]. The minimum weight $(e^{(2)}(x), e^{(3)}(x))$ sequence can be found by their technique and a non-probabilistic method used to obtain the error sequence estimate for $e_t^{(1)}$. The Schalkwijk and Vinck method is not affected by the equi-

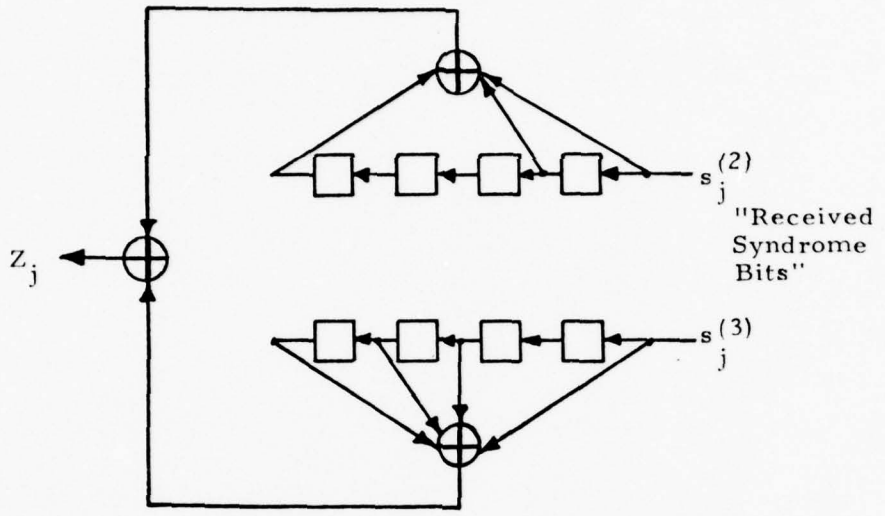


Figure 3.3a "Second Syndrome Former" with Syndrome Bit Inputs

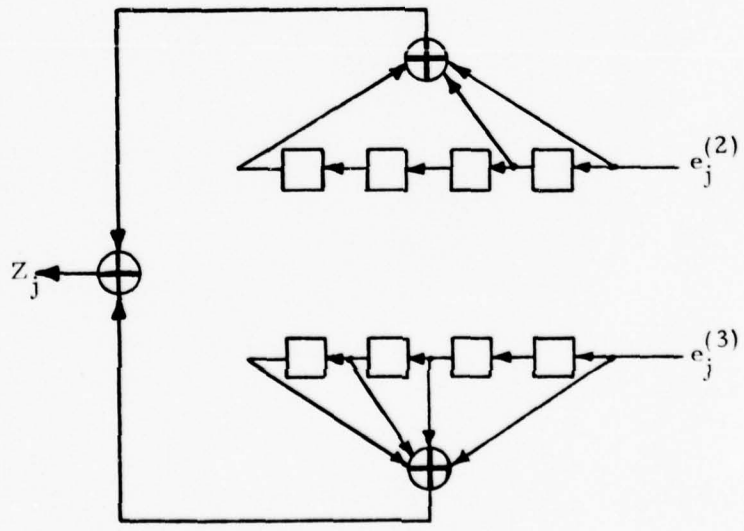


Figure 3.3b "Second Syndrome Former" with Error Bit Inputs

probable or nonequiprobable nature of the encoder input $e_t^{(1)}$ since their method determines the minimum Hamming weight noise sequence input which due to our assumption of a BSC is the most probable sequence.

The following parallels Schalkwijk and Vinck's explanation of their approach as modified for the rate 1/3 to rate 1/2 case.

The Second Syndrome Former Output is:

$$Z(x) = C_2(x)e^{(3)}(x) + C_3(x)e^{(2)}(x) \tag{3.35}$$

where

$$C_2(x) = 1+x^2+x^3+x^4 \tag{3.36a}$$

$$C_3(x) = 1+x+x^4 \tag{3.36b}$$

Here x plays the role of the delay operator.

By Euclid's algorithm there exists $D_2(x)$ and $D_3(x)$ such that

$$D_2(x)C_2(x) + D_3(x)C_3(x) = 1 \tag{3.37}$$

yielding

$$D_2(x) = x + 1 \tag{3.38a}$$

$$D_3(x) = x. \tag{3.38b}$$

Received are

$$s^{(2)}(x) = C_2(x)e^{(1)}(x) + e^{(2)}(x) \tag{3.39a}$$

$$s^{(3)}(x) = C_3(x)e^{(1)}(x) + e^{(3)}(x). \tag{3.39b}$$

The error sequence estimate is formed as

$$\hat{e}^{(1)}(x) = D_2(x)(s^{(2)}(x) + \hat{e}^{(2)}(x)) + D_3(x)(s^{(3)}(x)+\hat{e}^{(3)}(x)). \quad (3.40)$$

If the noise sequence estimate $(\hat{e}^{(2)}(x), \hat{e}^{(3)}(x))$ is correct then

$$\begin{aligned}\hat{e}^{(1)}(x) &= D_2(x)(C_2(x)e^{(1)}(x) + e^{(2)}(x) + \hat{e}^{(2)}(x)) \\ &\quad + D_3(x)(C_3(x)e^{(1)}(x) + e^{(3)}(x) + \hat{e}^{(3)}(x)) \\ &= (D_2(x)C_2(x) + D_3(x)C_3(x))e^{(1)}(x) = e^{(1)}(x). \end{aligned} \quad (3.41)$$

Equation (3.40) can be written as

$$\hat{e}^{(1)}(x) = (D_2(x)s^{(2)}(x) + D_3(x)s^{(3)}(x)) + \underbrace{D_2(x)\hat{e}^{(2)}(x)+D_3(x)\hat{e}^{(3)}(x)}_{\hat{w}(x)} \quad (3.42)$$

where with

$$w(x) = D_2(x)e^{(2)}(x) + D_3(x)e^{(3)}(x) \quad (3.43a)$$

$$\hat{w}(x) = D_2(x)\hat{e}^{(2)}(x) + D_3(x)\hat{e}^{(3)}(x) \quad (3.43b)$$

one obtains the following:

$$\hat{e}^{(1)}(x) = \underbrace{(D_2(x)s^{(2)}(x) + D_3(x)s^{(3)}(x))}_{\text{Computed easily from the received } s^{(2)}(x) \text{ and } s^{(3)}(x)} + \underbrace{\hat{w}(x)}_{\text{Determined via Viterbi-like algorithm}} \quad (3.44)$$

Since $\hat{w}(x)$ can be determined based only on the minimum Hamming weight noise $(\hat{e}^{(2)}(x), \hat{e}^{(3)}(x))$ inputs to the Second Syndrome Former and not the data inputs $e_t^{(1)}$ to the First Syndrome Former, this approach does not have to be adjusted for nonequiprobable $e_t^{(1)}$ inputs. Once

the $\{e_t^{(1)}\}$ sequence has been determined we only have to mod 2 add it to the $\{r_t^{(1)}\}$ sequence to determine $\{I_t\}$.

It is interesting to note that the rate 1/3 systematic code with

$$\begin{bmatrix} G_1 = 10000 \\ G_2 = 10111 \\ G_3 = 11001 \end{bmatrix}^*$$

converts by the above Second Syndrome Former technique to the rate 1/2 nonsystematic code $\begin{bmatrix} G_2 = 10111 \\ G_3 = 11001 \end{bmatrix}$ given in Schalkwijk and Vinck Table III, "Codes Realized." (The code is listed as code 4 with $\begin{bmatrix} g_2 = 10011 \\ g_3 = 11101 \end{bmatrix}^{**}$). [13].

From the discussion in Schalkwijk and Vinck regarding Table III is appears that had a rate 1/3 code been used with

$$\begin{bmatrix} G_1 = 10000 \\ G_2 = 11001 \\ G_3 = 11101 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} G_1 = 10000 \\ G_2 = 11001 \\ G_3 = 11011 \end{bmatrix}$$

the decoding problem could have "converted" to a rate 1/2 code (Codes 3 and 2 respectively in Table III [13]) whose decoding would have required far less complexity than the Code 4 obtained from the rate 1/3 code of this example.

*0-th order left.

**0-th order right.

3.3 Optimal Real Time MAP Error Bit Estimation

In this section a recursive method for the computation of the optimum MAP error bit estimate of $e_t^{(1)}$ is derived assuming a finite delay Δ. The notation used throughout this section parallels that of Fritchman and Mixsell [20] in their work on optimal real time data bit estimation.

Consider a rate 1/n systematic code. The task is to estimate one error bit (e.g. $e_{t-\Delta}^{(1)}$) given the output of the syndrome former from the start of the reception (i.e. t=1) to the present time t=t allowing only a finite delay Δ. That is, find the maximum a posteriori estimate of $e_{t-\Delta}^{(1)}$ given $\underline{Z}_{(1,t)}$. This can be written as $\hat{e}_{t-\Delta}^{(1)}$ where $\hat{e}_{t-\Delta}^{(1)}$ is that value of $e_{t-\Delta}^{(1)}$ that produces the

$$\underset{\substack{\text{over all} \\ e_{t-\Delta}^{(1)} \text{ values}}}{\text{maximum}} \left\{ \Pr(e_{t-\Delta}^{(1)} / \underline{Z}_{(1,t)}) \right\} = \underset{\substack{\text{over all} \\ e_{t-\Delta}^{(1)} \text{ values}}}{\text{maximum}} \left\{ \frac{\Pr(e_{t-\Delta}, \underline{Z}_{(1,t)})}{\Pr(\underline{Z}_{(1,t)})} \right\} \tag{3.45}$$

where

(1) $e_{t-\Delta}^{(1)}$ is the error bit to be estimated;

(2) $\{e_t^{(1)}\}$ are error bits which do enter the syndrome former physical state;

(3) $\underline{Z}_{(1,t)}$ is the syndrome vector from startup to present (containing all the information possibly available to a syndrome type decoder); and

(4) Δ is the "delay" permitted before an error bit decision must be output from the decoder.

One should note that in (3.45) the estimate will be made based on the entire vector $\underline{Z}_{(1,t)}$ via a recursive technique. This is in contrast to the ROMMAP approach to be discussed later in Section 4.1, in which the estimate will be based on only the finite length syndrome segment $\underline{Z}_{(t-\Delta,t)}$ namely,

$$\underset{\substack{\text{over all} \\ e^{(1)}_{t-\Delta} \text{ values}}}{\text{maximum}} \left\{ \Pr(e^{(1)}_{t-\Delta} / \underline{Z}_{(t-\Delta,t)}) \right\} = \underset{\substack{\text{over all} \\ e^{(1)}_{t-\Delta} \text{ values}}}{\text{maximum}} \left\{ \frac{\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(t-\Delta,t)})}{\Pr(\underline{Z}_{(t-\Delta,t)})} \right\} \tag{3.46}$$

For now, the problem presented by Eq. (3.45) will be considered. There exist several methods of expressing the numerator joint probability $\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)})$ which have been used by various authors [11], [20] in estimating information bits.

For example,

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}) = \sum_{e^{(1)}_{t-\Delta+1}} \cdots \sum_{e^{(1)}_{t}} \Pr(e^{(1)}_{(t-\Delta,t)}, \underline{Z}_{(1,t)}) \tag{3.47}$$

with $e^{(1)}_{(t-\Delta,t)} = (e^{(1)}_{t-\Delta}, e^{(1)}_{t-\Delta+1}, \ldots, e^{(1)}_{t-1}, e^{(1)}_{t})$. Here averages are taken over all possible paths (error bit sequences) of length Δ after the particular error bit to be estimated.

By conditioning the numerator on the syndrome former state S_{t+1} and averaging over all possible values of S_{t+1} one obtains an

values of $e^{(1)}_{t-\Delta}$, one can also determine a measure of decision quality by summing (in the binary case) the two numerator terms and using:

$$\left\{\begin{matrix}\text{Decision} \\ \text{quality on} \\ \hat{e}^{(1)}_{t-\Delta}\end{matrix}\right\} = \frac{\Pr(\hat{e}^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)})}{\Pr(\hat{e}^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}) + \Pr(\bar{e}^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)})} \tag{3.50}$$

Therefore since either approach can be useful in soft decision decoding, one can somewhat arbitrarily use (1) above (similar to Fritchman and Mixsell [20]) since this entails slightly less computations and is somewhat easier to derive than the approach (2) analogous to Lee [11] and [12].

In the following, the case of $\Delta > \nu$ is considered. This would appear to be of most applicability since one would typically use a delay Δ greater than a constraint length $(\nu+1)$ in order to obtain as much coding gain as possible with a particular convolutional code.

For completeness, the case $\Delta \leq \nu$ is considered in Appendix C where a decision directed approach is also described based on incorporation of prior estimates of $\hat{e}^{(1)}_{t-\Delta-1}$. This is similar to Abend and Fritchman [21] equation 21b.

Here let us consider the numerator of (3.3-1)

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}) = \sum_{S_{t+1}=s} \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}, S_{t+1}=s) \tag{3.51}$$

Since $S_t = (e^{(1)}_{t-1}, e^{(1)}_{t-2}, \ldots, e^{(1)}_{t-\nu})$ and $S_{t+1} = (e^{(1)}_{t}, e^{(1)}_{t-1}, \ldots, e^{(1)}_{t-\nu+1})$ for this rate 1/n systematic code (where $\{e^{(1)}_t\}$ are those error bits entering into the syndrome former physical state), we can also write:

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}) = \sum_{e^{(1)}_{t-\nu+1}} \cdots \sum_{e^{(1)}_{t}} \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}, e^{(1)}_{t-\nu+1}, \ldots, e^{(1)}_{t}) \tag{3.52}$$

or

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}) = \sum_{e^{(1)}_{t-\nu+1}} \cdots \sum_{e^{(1)}_{t}} \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}, e^{(1)}_{(t-\nu+1,t)}) \tag{3.53}$$

Now consider the $\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}, e^{(1)}_{(t-\nu+1,t)})$ terms of the numerator summation conditioned on $e^{(1)}_{t-\nu}$ and averaged over $e^{(1)}_{t-\nu}$.

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}, e^{(1)}_{t-\nu+1,t}) = \sum_{e^{(1)}_{t-\nu}} \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, \underline{Z}_t, e^{(1)}_{(t-\nu,t)})$$

$$= \sum_{e^{(1)}_{t-\nu}} \Pr(\underline{Z}_t / e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t)}) \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t)}) \tag{3.54}$$

Since the statistics of $\underline{Z}_t$, when conditioned on $e^{(1)}_{(t-\nu,t)}$, are <u>not</u> a function of prior values of $\underline{Z}_{(1,t-1)}$ and since the statistics of $\underline{Z}_t$ when conditioned on $e^{(1)}_{(t-\nu,t)}$ are not a function of $e^{(1)}_{t-\Delta}$, one can write

$$\Pr(\underline{Z}_t / e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t)}) = \Pr(\underline{Z}_t / e^{(1)}_{(t-\nu,t)}) . \tag{3.55}$$

Also, due to the bit-by-bit independence of the errors, one can write

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t)}) =$$

$$\Pr(e^{(1)}_{t}) \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t-1)} / e^{(1)}_{t}) \qquad (3.56)$$

But the statistics of $\underline{Z}_{(1,t-1)}$ and of $e^{(1)}_{(t-\nu,t-1)}$ are not a function of a "future" $e^{(1)}_{t}$. Therefore

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t)}) = \Pr(e^{(1)}_{r}) \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t-1)}) \qquad (3.57)$$

and one can thus write

$$\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}, e^{(1)}_{(t-\nu+1,t)} =$$

$$\Pr(e^{(1)}_{t}) \Pr(\underline{Z}_{t} / e^{(1)}_{(t-\nu,t)} \Pr(e^{(1)}_{t-\Delta}, Z_{(1,t-1)} e^{(1)}_{(t-\nu,t-1)}) \qquad (3.58)$$

$e^{(1)}_{t-\nu}$

This is a computable recursion since:

(1) $\Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t-1)}, e^{(1)}_{(t-\nu,t-1)})$ is the pri or numerator term for estimating $e^{(1)}_{t-\Delta}$ from $\underline{Z}_{(1,t-1)}$;

(2) $\Pr(e^{(1)}_{t})$ is known for the binary symmetric channel; and

(3) $\Pr(\underline{Z}_{t} / e^{(1)}_{(t-\nu,t)}) = \Pr(\underline{Z}_{t} / S_{t}, S_{t+1}) =$ syndrome former output probability.

Since the syndrome former output $\underline{Z}_{t}$ is a function of <u>both</u> the state pair (S_{t}, S_{t+1}) and the other "don't care" error bit(s) (i.e. those that do not enter into the syndrome former storage elements),

one can obtain the syndrome former output probabilities by conditioning on $\mathcal{S}_t$ and averaging over $\mathcal{S}_t$ where

$$\mathcal{S}_t = \left\{ \begin{array}{l} \text{All those error bits } e_t^{(i)} \text{ in } \underline{E}_t \text{ which affect the syndrome} \\ \text{output } \underline{Z}_t \text{ but which are not present in the actual storage} \\ \text{elements representing } (S_t, S_{t+1}). \end{array} \right\} .$$

Therefore

$$\Pr(\underline{Z}_t / e^{(1)}_{(t-\nu, t)}) = \Pr(\underline{Z}_t / S_t, S_{t+1}) = \sum_{\mathcal{S}_t} \Pr(Z_t / S_t, S_{t+1}, \mathcal{S}_t) \Pr(\mathcal{S}_t). \tag{3.59}$$

The expression (3.53) for the numerator now becomes

$$\Pr(e^{(1)}_{(t-\Delta}, \underline{Z}_{(1,t)}) = \sum_{e^{(1)}_{t-\nu+1}} \cdots \sum_{e^{(1)}_t} \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,t)}, e^{(1)}_{(t-\nu+1,t)})$$

$$= \sum_{e^{(1)}_{t-\nu+1}} \cdots \sum_{e^{(1)}_t} \sum_{e^{(1)}_{t-1}} \Pr(e^{(1)}_t) \Pr(\underline{Z}_t / e^{(1)}_{(t-\nu, t)}) \cdot$$

$$\cdot \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1, t-1)}, e^{(1)}_{(t-\nu, t-1)}) . \tag{3.60}$$

For notational convenience in the recursion explanation g_i is defined as

$$g_i \triangleq \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1, t-\Delta+i)}, e^{(1)}_{(t-\Delta-\nu+1+i, t-\Delta+i)}) \tag{3.61}$$

By the conditioning and averaging discussed above, this is equivalent to

$$g_i = \sum_{e^{(1)}_{t-\Delta+i-\nu}} \Pr(e^{(1)}_{t-\Delta+i}) \Pr(\underline{Z}_{t-\Delta+i} / e^{(1)}_{(t-\Delta-\nu+i,\, t-\Delta+i)})$$

$$\cdot \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,\, t-\Delta+i-1)}, e^{(1)}_{(t-\Delta-\nu+i,\, t-\Delta+i-1)}) \qquad (3.62)$$

where

$$g_0 = \Pr(e^{(1)}_{t-\Delta}, \underline{Z}_{(1,\, t-\Delta)}, e^{(1)}_{(t-\Delta-\nu+1,\, t-\Delta)}) \qquad (3.63)$$

$$= \Pr(\underline{Z}_{(1,\, t-\Delta)}, e^{(1)}_{(t-\Delta-\nu+1,\, t-\Delta)})$$

since $e^{(1)}_{t-\Delta}$ is contained in $e^{(1)}_{(t-\Delta-\nu+1,\, t-\Delta)}$.

Now

$$g_\Delta = \Pr(e^{(1)}_{t-\Delta}, Z_{(1,\, t-\Delta+\Delta)}, e^{(1)}_{(t-\Delta-\nu+1+\Delta,\, t-\Delta+\Delta)})$$

$$= \Pr(e^{(1)}_{t-\Delta}, Z_{(1,\, t)}, e^{(1)}_{(t-\nu+1,\, t)}) \qquad (3.64)$$

is exactly the term needed in the numerator summation (3.54) to decide on $\hat{e}^{(1)}_{t-\Delta}$.

Now following the modification of Fritchman and Mixsell [20] to Lee's algorithm to use the minimum delay one can define

$$Q_i \triangleq g_{i+\nu} = \Pr(e^{(1)}_{t-\Delta}, Z_{(1,\, t-\Delta+i+\nu)}, e^{(1)}_{(t-\Delta+i+1,\, t-\Delta+i+\nu)}) \qquad (3.65)$$

where

$$Q_0 = g_\nu = \Pr(e^{(1)}_{t-\Delta}, Z_{(1,\, t-\Delta+\nu)}, e^{(1)}_{(t-\Delta+1,\, t-\Delta+\nu)})$$

$$= \Pr(Z_{1,\, t-\Delta+\nu)}, e^{(1)}_{(t-\Delta,\, t-\Delta+\nu)}) \qquad (3.66)$$

and

$$Q_{\Delta-\nu} = g_{\Delta} = Pr(e_{t-\Delta}, Z_{(1,t)}, e_{(t-\nu+1,t)}) \tag{3.67}$$

which is exactly the probability needed in (3.53) to decide on $e^{(1)}_{t-\Delta}$. That is the numerator (3.45) equals

$$Pr(e^{(1)}_{t-\Delta}=i, Z_{(1,t)}) = \sum_{e^{(1)}_{t-\nu+1}} \cdots \sum_{e^{(1)}_{t}} Pr(e^{(1)}_{t-\Delta}=i, Z_{(1,t)}, e^{(1)}_{(t-\nu+1,t)}) \tag{3.68}$$

which is expressible as

$$Pr(e^{(1)}_{t-\Delta}=j, Z_{(1,t)}) = \sum_{e^{(1)}_{t-\nu+1}} \cdots \sum_{e^{(1)}_{t}} Q_{\Delta-\nu} \tag{3.69}$$

for j = 0, 1, in this binary case.

Recursion for Q_i

Given Q_0 for $e^{(1)}_{t-\Delta} = j$, j=0, 1, it is possible to recursively compute $Q_1, Q_2, \ldots, Q_{\Delta-\nu}$ as shown below. (In the following the superscript of $e^{(1)}_t$ will be dropped to ease notation, i.e. $e^{(1)}_t = e_t$.)

$$Q_i = \sum_{e_{t-\Delta+i}} P(e_{t-\Delta}=j, Z_{(1,t-\Delta+i+\nu)}, e_{(t-\Delta+i,t-\Delta+i+\nu)})$$

$$= \sum_{e_{t-\Delta+i}} P(Z_{t-\Delta+i+\nu}/e_{t-\Delta}=j, Z_{(1,t-\Delta+i+\nu-1)}, e_{t-\Delta+i,t-\Delta+i+\nu)}$$

$$\cdot P(e_{t-\Delta}=j, Z_{(1,t-\Delta+i+\nu-1)}, e_{(t-\Delta+i,t-\Delta+i+\nu)})$$

$$= \sum_{e_{t-\Delta+i}} P(Z_{t-\Delta+i+\nu}/e_{(t-\Delta+i,t-\Delta+i+\nu)}) P(e_{t-\Delta}=j, Z_{(1,t-\Delta+i+\nu-1)},$$

$$e_{(t-\Delta+i,t-\Delta+i+\nu)})$$

$$= \sum_{e_{t-\Delta+i}} P(e_{t-\Delta+i+\nu}) \overbrace{P(e_{t-\Delta}=j, Z_{(1,\, t-\Delta+i+\nu-1)}, e_{(t-\Delta+i,\, t-\Delta+i+\nu-1)})}^{Q_{i-1}}$$

$$\cdot P(Z_{t-\Delta+i+\nu} / e_{(t-\Delta+i,\, t-\Delta+i+\nu)}) \cdot \tag{3.70}$$

Therefore,

$$\boxed{Q_i = \sum_{e_{t-\Delta+i}} P(e_{t-\Delta+i+\nu}) P(Z_{t-\Delta+i+\nu} / e_{(t-\Delta+i,\, t-\Delta+i+\nu)}) Q_{i-1}} \tag{3.71}$$

Note that since both numerators (one for $e_{t-\Delta}=0$ and one for $e_{t-\Delta}=1$) must be computed, at each step in the recursion there are two Q_i's.

Decision

Once the two parallel recursions are complete (yielding $Q_{\Delta-\nu}$ for $e_{t-\Delta} = 0$ and $e_{t-\Delta} = 1$) one can then decide on $\hat{e}_{t-\Delta}$ by comparing the numerator for $e_{t-\Delta} = 0$,

$$P(e_{t-\Delta}=0, Z_{(1,\, t)}) = \sum_{e_{t-\nu+1}} \cdots \sum_{e_t} P(e_{t-\Delta}=0, Z_{(1,\, t)}, e_{(t-\nu+1,\, t)})$$

$$= \sum_{e_{t-\nu+1}} \cdots \sum_{e_t} Q_{\Delta-\nu} \text{ (with } e_{t-\Delta}=0) \tag{3.72}$$

with the numerator for $e_{t-\Delta}=1$,

$$P(e_{t-\Delta}=1, Z_{(1,\, t)}) = \sum_{e_{t-\nu+1}} \cdots \sum_{e_t} P(e_{t-\Delta}=1, Z_{(1,\, t)}, e_{(t-\nu+1,\, t)})$$

$$= \sum_{e_{t-\nu+1}} \cdots \sum_{e_t} Q_{\Delta-\nu} \quad \text{(with } e_{t-\Delta}=1) \tag{3.73}$$

Determination of Q_0

The two starting probabilities $Q_0(e_{t-\Delta}=0)$ and $Q_0(e_{t-\Delta}=1)$ are needed to initiate the above Q-recursion. These may be obtained as follows:

Define

$$q(e_{(t-\Delta,\, t-\Delta+\nu)}) = Q_0 = P(Z_{(1,\, t-\Delta+\nu)}, e_{(t-\Delta,\, t-\Delta+\nu)}) \quad (3.74)$$

and in general

$$q(e_{(t-\nu+i,\, t+i)}) = P(Z_{(1,\, t+i)}, e_{(t-\nu+i,\, t+i)}) . \quad (3.75)$$

This q function provides the necessary starting probabilities Q_0 in a manner similar to Lee [11] in which a denominator recursion was used to provide the start for each numerator recursion.

By using conditioning and averaging one can break up the joint probability for general i and write

$$q(e_{(t-\nu+i,\, t+i)}) = P(e_{t+i}) P(Z_{t+i}/e_{(t-\nu+i,\, t+i)}) \sum_{e_{t-\nu+i-1}} q(e_{(t-\nu+i-1,\, t+i-1)}) \quad (3.76)$$

Below is an illustration of the computation of a specific case, namely $i = \nu-\Delta$, of the determination of Q_0.

$$Q_0 = q(e_{(t-\Delta,\, t-\Delta+\nu)}) = P(e_{(t-\Delta,\, t-\Delta+\nu)}, Z_{(1,\, t-\Delta+\nu)})$$

$$= P(Z_{t-\Delta+\nu}/Z_{(1,\, t-\Delta+\nu-1)}, e_{(t-\Delta,\, t-\Delta+\nu)}) P(Z_{(1,\, t-\Delta+\nu-1)}, e_{(t-\Delta,\, t-\Delta+\nu)})$$

$$= P(Z_{t-\Delta+\nu}/e_{(t-\Delta,\, t-\Delta+\nu)}) P(Z_{(1,\, t-\Delta+\nu-1)}, e_{(t-\Delta,\, t-\Delta+\nu)})$$

$$= P(Z_{t-\Delta+\nu}/e_{(t-\Delta,\, t-\Delta+\nu)})P(e_{t-\Delta+\nu})P(Z_{(1,\, t-\Delta+\nu-1)}, e_{(t-\Delta,\, t-\Delta+\nu-1)}/e_{t-\Delta+\nu})$$

$$= P(e_{t-\Delta+\nu})P(Z_{t-\Delta+\nu}/e_{(t-\Delta,\, t-\Delta+\nu)})P(Z_{(1,\, t-\Delta+\nu-1)}, e_{(t-\Delta,\, t-\Delta+\nu-1)})$$

$$= P(e_{t-\Delta+\nu})P(Z_{t-\Delta+\nu}/e_{(t-\Delta,\, t-\Delta+\nu)}) \sum_{e_{t-\Delta-1}} \underbrace{P(Z_{(1,\, t-\Delta+\nu-1)}, e_{(t-\Delta-1,\, t-\Delta+\nu-1)})}_{q(e_{(t-\Delta-1,\, t-\Delta+\nu-1)})} \tag{3.77}$$

Thus it is feasible to obtain Q_0 from the previous value of $q(e_{(t-\Delta-1,\, t-\Delta+\nu-1)})$. One should note that $P(e_{t-\Delta+\nu})$ in the above is known for the binary symmetric channel. The second probability, $P(Z_{t-\Delta+\nu}/e_{(t-\Delta,\, t-\Delta+\nu)})$ must be averaged over the "don't care" error(s) namely,

$$P(Z_{t-\Delta+\nu}/e_{(t-\Delta,\, t-\Delta+\nu)}) = \sum_{\delta_{t-\Delta+\nu}} P(Z_{t-\Delta+\nu}/e_{(t-\Delta,\, t-\Delta+\nu)}, \delta_{t-\Delta+\nu}) \cdot P(\delta_{t-\Delta+\nu}) \tag{3.78}$$

where $\delta_{t-\Delta+\nu}$ represents those error bits which affect the output of the syndrome former at time $t-\Delta+\nu$ but do not appear in the syndrome former physical state (e.g. $e^{(2)}_{t-\Delta+\nu}$ in the rate $\frac{1}{2}$ example). This type of computation of the syndrome former output probabilities was to be expected since this analysis is done for a multiple input syndrome former. Some inputs (e.g. $e^{(1)}_t$) needed for decoding must be estimated while other inputs (e.g. $e^{(2)}_t$) are nuisance inputs which must be averaged out.

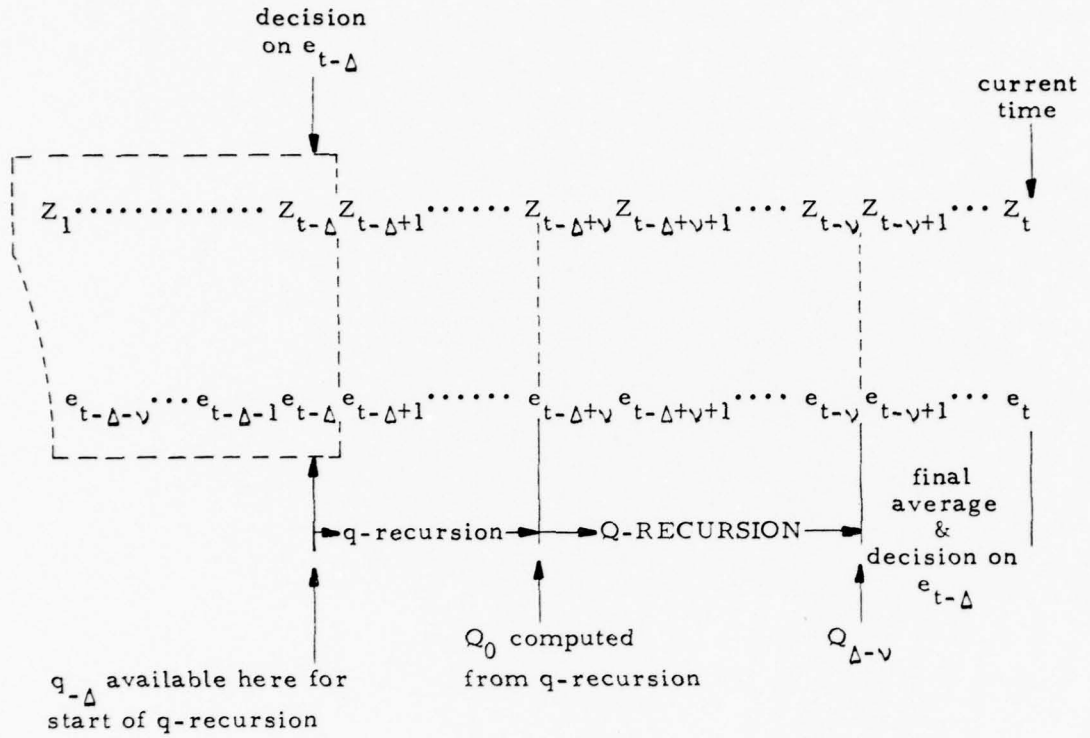


Figure 3.4. Optimal Real Time MAP Error Bit Recursion Relationships

does not have to be adjusted for nonequiprobable $e_t^{(1)}$ inputs. Once

It is best to note that $Q_0 = q(e_{(t-\Delta, t-\Delta+\nu)})$ is actually a set of $2^{\nu+1}$ numbers representing the joint probabilities $P(Z_{(1, t-\Delta+\nu)}, e_{(t-\Delta, t-\Delta+\nu)})$. One can consider each of these numbers to be labeled by the associated binary valued vectors $(e_{t-\Delta}, e_{t-\Delta+1}, \ldots, e_{t-\Delta+\nu})$. Thus the initial Q_0 starting probabilities for the Q recursion to $Q_{\Delta-\nu}$ is a substantial set of $2^{\nu+1}$ numbers labeled by $\nu+1$ binary error values.

However in the algorithm to follow, one never has to process more than $2^{\nu+1}$ numbers since in the Q recursion the contribution of a new error value is included while that of an old error value is averaged out. Thus the necessary storage for $2^{\nu+1}$ numbers is maintained throughout the recursion to $Q_{\Delta-\nu}$ ($\Delta-\nu$ steps). The final ν averaging steps following the Q recursion reduce the $2^{\nu+1}$ numbers down to two upon which the decision on $e_{t-\Delta}$ is made. One does, however, need to still retain the $2^{\nu+1}$ values for Q_0 until subsequent steps decide $\hat{e}_{t-\Delta}$ since these values are needed to compute the new set of Q_0 when t is increased by one to estimate $\hat{e}_{t-\Delta+1}$ in the next pass. Thus the maximum storage required is the sum $2^{\nu+1}+2^{\nu+1} = 2^{\nu+2}$.

Algorithm for Real Time MAP Error Bit Estimate

Below is given the optimal real time MAP error bit estimation algorithm assuming a delay of Δ. The statements which are boxed represent the actual algorithm with the remainder being additional

clarifications.

Step 0: Set $q(e_{(1,0)}) = 1$ for $t = 0$.

For $t = 1, 2, \ldots, \nu+1$ calculate

$$q(e_{(1,t)}) = Pr(e_t)Pr(Z_t/e_{(1,t)})q(e_{(1,t-1)})$$

Note this is the case of Eq. (3.76) with i=0 and $t \leq \nu+1$. Thus no summation is required until t is $> \nu+1$. That is

$$q(e_{(t-\nu+i,\,t+i)}) = Pr(e_{t+i})Pr(Z_{t+i}/e_{(t-\nu+i,\,t+i)}) \sum_{e_{t-\nu+i-1}} q(e_{(t-\nu+i-1,\,t+i-1)})$$

With i=0 one has

$$q(e_{(t-\nu,\,t)}) = Pr(e_t)Pr(Z_t/e_{(t-\nu,\,t)}) \sum_{e_{t-\nu-1}} q(e_{(t-\nu-1,\,t-1)}) \qquad (3.79)$$

and for $t \leq \nu+1$, $q(e_{(t-\nu,\,t)}) = q(e_{(1,t)})$. Thus

$$q(e_{(1,t)}) = Pr(e_t)Pr(Z_t/e_{(1,t)})q(e_{(1,t-1)}) \text{ for } t \leq \nu+1 .$$

Considering the above, the values of the initial start up q terms can be tabulated as below:

i=0, t=0: $q(e_{(1,0)}) = 1$ (one term) .

i=0, t=1: $q(e_{(1,1)}) = P(e_1)P(Z_1/e_{(1,1)})q(e_{(1,0)})$ (two terms)

this yields:

$$q(e_{(1,1)}=0) = P(e_1=0)P(Z_1/e_1=0)\ (1) = Pr(e_1=0, Z_1)$$

$$q(e_{(1,1)}=1) = P(e_1=1)P(Z_1/e_1=1)\ (1) = Pr(e_1=1, Z_1)\ .$$

$i=0, t=2$: $q(e_{(1,2)}) = P(e_2)P(Z_2/e_{(1,2)})q(e_{(1,1)})$ (four terms)

this yields:

$$q(e_{(1,2)}=00) = P(e_2=0)P(Z_2/e_{(1,2)}=00)q(e_{(1,1)}=0)$$

$$q(e_{(1,2)}=01) = P(e_2=1)P(Z_2/e_{(1,2)}=01)q(e_{(1,1)}=0)$$

$$q(e_{(1,2)}=10) = P(e_2=0)P(Z_2/e_{(1,2)}=10)q(e_{(1,1)}=1)$$

$$q(e_{(1,2)}=11) = P(e_2=1)P(Z_2/e_{(1,2)}=11)q(e_{(1,1)}=1)\ .$$

$$\vdots \qquad\qquad \vdots$$

$i=0, t=\nu+1$: $q(e_{(1,\nu+1)}) = P(e_{\nu+1})P(Z_{\nu+1}/e_{(1,\nu+1)})q(e_{(1,\nu)})$ ($2^{\nu+1}$ terms)

(3.80)

The above computations form a tree that is $\nu+1$ branches long having $2^{\nu+1}$ terminal nodes. This "initial error tree" is sketched in Figure 3.5.

Step 1: Set $P(e_{t-\nu}, Z_{(1,t)}, e_{(t-\nu+1,t)}) = q(e_{(t-\nu,t)})$

If $\Delta \leq \nu$ go to Step 4 (see Appendix C).

For illustration, consider the vase of $t = \nu+1$ which is the first time Step 1 is followed. One thus has,

$$\underbrace{P(e_1, Z_{(1,\nu+1)}, e_{(2,\nu+1)})}_{2^{\nu+1}\text{ terms}} = \underbrace{q(e_{(1,\nu+1)})}_{2^{\nu+1}\text{ terms}} = P(Z_{(1,\nu+1)}, e_{(1,\nu+1)}) \qquad (3.81)$$

S_{t+1} and averaging over all possible values of S_{t+1} one obtains an

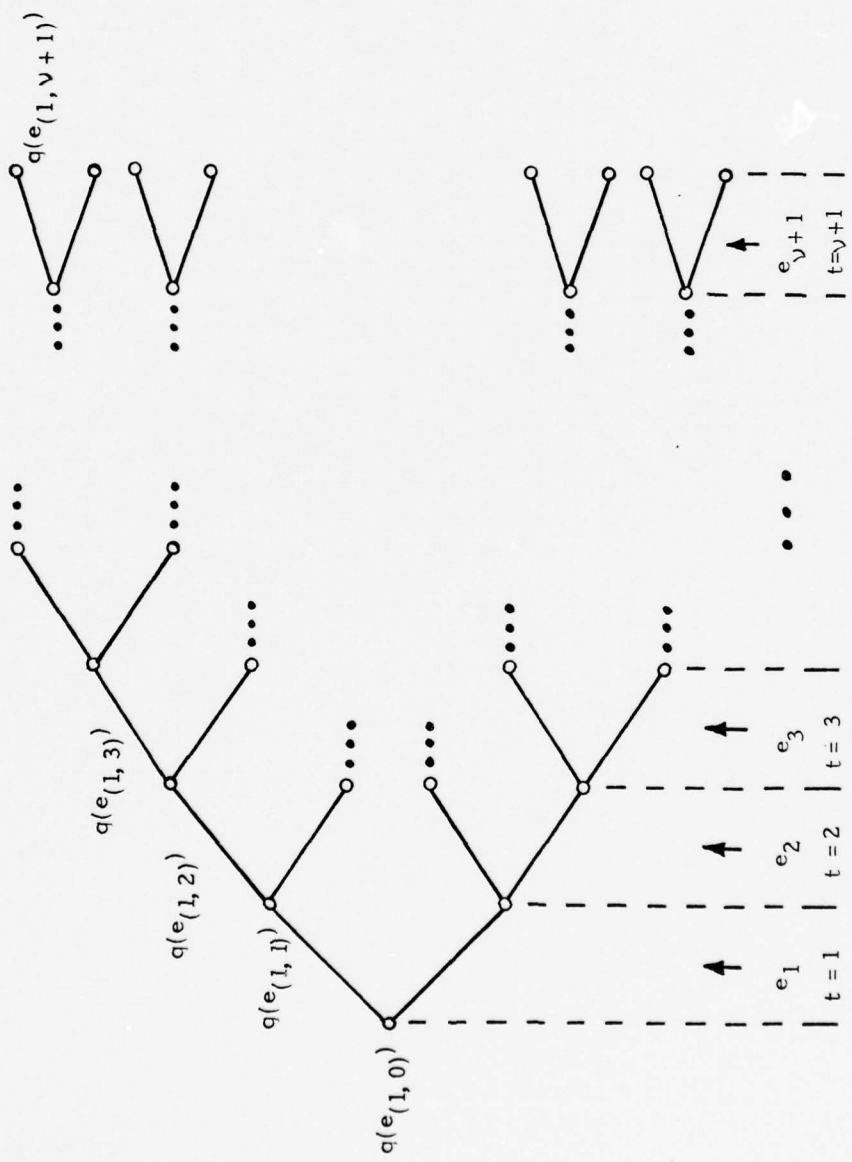


Figure 3.5 Optimal Real Time MAP Error Bit Estimate Initial Error Tree

Step 2: (The Q-recursion for $\Delta > \nu$ case.) For $i=1,2,\ldots,\Delta-\nu$ calculate

$$P(e_{t-\nu}, Z_{(1,t+i)}, e_{(t-\nu+1+i,\,t+i)})$$

$$= \sum_{e_{t-\nu+i}} P(e_{t+i})P(Z_{t+i}/e_{(t-\nu+i,\,t+i)}) \cdot P(e_{t-\nu}, Z_{(1,t+i-1)}, e_{(t-\nu+i,\,t+i-1)})$$

Step 2 basically increments by the new error bit and averages over an old error bit. Specifically for $t = \nu+1$ this becomes:

$i=1, Q_i$:

$$P(e_1, Z_{(1,\nu+2)}, e_{(3,\nu+2)})$$

$$= \sum_{e_2} P(e_{\nu+2})P(Z_{\nu+2}/e_{(2,\nu+2)})P(e_1, Z_{(1,\nu+1)}, e_{(2,\nu+1)})$$

$$= \sum_{e_2} P(e_{\nu+2})P(Z_{\nu+2}/e_{(2,\nu+2)})\underbrace{q(e_{(1,\nu+1)})}_{2^{\nu+1} \text{ of these terms}} \qquad (3.82)$$

It is instructive to consider just what the above Q_1 calculation actually involves. Recall that $q(e_{(1,\nu+1)})$ is a set of $2^{\nu+1}$ numbers (probabilities) each labeled by values of $(e_1, e_2, \ldots, e_{\nu+1})$ as indicated in the initial error tree of Figure 3.5 . The step 2 for Q_1 requires averaging over e_2. Therefore one sets $e_{\nu+2}=0$. Next one

takes the pair of numbers $q(e_{(1, e_2=0, 3, \ldots, \nu+1)})$ and $q(e_{(1, e_2=1, 3, \ldots, \nu+1)})$ and multiplies each by $P(e_{\nu+2}=0) \cdot P(Z_{\nu+2}/e_{(2=0, 3, \ldots, \nu+2=0)})$, and $P(e_{\nu+2}=0)P(Z_{\nu+2}/e_{(2=1, 3, \ldots, \nu+2=0)})$ respectively and adds.

When all pairs of numbers $q(e_{(1, \ldots, \nu+1)})$ have been so processed one has 2^{ν} new probabilities corresponding to $e_{\nu+2}=0$.

Now the above process is repeated with $e_{\nu+2}=1$ to yield 2^{ν} new probabilities incorporating the influence of $e_{\nu+2} = 1$.

Thus at the Q_1 step there must be stored $2^{\nu}+2^{\nu} = 2^{\nu+1}$ numbers to be used to compute Q_2. These are of the form: $P(e_1, Z_{(1, \nu+2)}, e_{(3, \nu+2)})$. By "incrementing" by $e_{\nu+2}$ and averaging over e_2, the required storage is maintained at $2^{\nu+1}$ as it can be also for the rest of the Q recursion to $Q_{\Delta-\nu}$. However the old set of $q(e_{(1, \nu+1)})$ must be kept in storage for use in Step 5 so the actual storage necessary is $2^{\nu+2}$.

Continuing the Q-recursion gives at $i = \Delta-\nu$, $Q_{\Delta-\nu}$:

$$P(e_1, Z_{(1, 1+\Delta)}, e_{(2+\Delta-\nu, 1+\Delta)})$$

$$= \sum_{e_{1+\Delta-\nu}} P(e_{1+\Delta})P(Z_{1+\Delta}/e_{(1+\Delta-\nu, 1+\Delta)})\underbrace{P(e_1, Z_{(1, \Delta)}, e_{(1+\Delta-\nu, \Delta)})}_{Q_{\Delta-\nu-1}} \tag{3.83}$$

Thus following the $Q_{\Delta-\nu}$ step of the Q-recursion, the decoder has stored $2^{\nu+1}$ probabilities labeled by the binary vectors

$(e_1, e_{\Delta-\nu+2}, \ldots, e_{1+\Delta})$. These probabilities are

$$p(e_1, Z_{(1,\,1+\Delta)}, e_{(\Delta-\nu+2,\,\Delta+1)}) .$$

Now Step 3 is entered to average out $e_{\Delta-\nu+2}$ through $e_{\Delta+1}$.

Step 3: (Final average and decision for $\Delta > \nu$ case.) For i=0, 1 calculate

$$\Lambda_i = \sum_{e_{t+\Delta-2\nu+1}} \cdots \sum_{e_{t+\Delta-\nu}} P(e_{t-\nu}=i, Z_{(1,\,t+\Delta-\nu)}, e_{(t+\Delta-2\nu+1,\,t+\Delta-\nu)})$$

If $\Lambda_0 > \Lambda_1$ decide $\hat{e}_{t-\nu}=0$, otherwise $\hat{e}_{t-\nu}=1$. Go to Step 5 to "increment" q.

For estimating $\hat{e}_1$ with $t = \nu+1$ these quantities are

$$\Lambda_0 = \sum_{e_{\Delta-\nu+2}} \cdots \sum_{e_{\Delta+1}} P(e_1=0, Z_{(1,\,\Delta+1)}, e_{(\Delta-\nu+2,\,\Delta+1)}) \tag{3.84}$$

$$\Lambda_1 = \sum_{e_{\Delta-\nu+2}} \cdots \sum_{e_{\Delta+1}} P(e_1=1, Z_{(1,\,\Delta+1)}, e_{(\Delta-\nu+2,\,\Delta+1)})$$

Again it is instructive to state just what is being done here. For Λ_0 select those numbers of the set $Q_{\Delta-\nu} = \{P(e_1, Z_{(1,\,\Delta+1)}, e_{(\Delta-\nu+2,\,\Delta+1)})\}$ labeled by $e_1=0$. Call this subset $Q_{\Delta-\nu}(e_1=0)$. It contains 2^ν numbers. At the first summation over $e_{\Delta-\nu+2}$ select those numbers in $Q_{\Delta-\nu}(e_1=0)$ whose labels match in all but the $e_{\Delta-\nu+2}$ position

$$P(e_1=0, Z_{(1,\Delta+1)}, e_{(\Delta-\nu+2=0,\ldots,\Delta+1)}),$$

$$P(e_1=0, Z_{(1,\Delta+1)}, e_{(\Delta-\nu+2=1,\ldots,\Delta+1)})$$

and add such pairs equally weighted to obtain $2^{\nu-1}$ elements in the set of new probabilities. Repeat this process a total of ν times resulting in

$$\Lambda_0 = P(e_1=0, Z_{(1,\Delta+1)})$$

$$\Lambda_1 = P(e_1=1, Z_{(1,\Delta+1)})$$

If $\Lambda_0 > \Lambda_1$ decide $\hat{e}_1=0$, otherwise $\hat{e}_1=1$.

Step 5: (Increment q_*) Increase t by 1 and calculate

$$q(e_{(t-\nu,t)}) = P(e_t)P(Z_t/e_{(t-\nu,t)}) \sum_{e_{t-\nu-1}} q(e_{(t-\nu-1,t-1)})$$

Go to Step 1.

Here for example $t = \nu+1$ would increase to $t = \nu+2$ to give

$$q(e_{(2,\nu+2)}) = P(e_{\nu+2})P(Z_{\nu+2}/e_{(2,\nu+2)}) \sum_{e_1} q(e_{(1,\nu+1)}) \qquad (3.85)$$

Here for each of the $2^{\nu+1}$ numbers $q(e_{(1,\nu+1)})$ in the set from the initial error tree, find the pair $q(e_1=0, e_2, \ldots, e_{\nu+1})$, $q(e_1=1, e_2, \ldots, e_{\nu+1})$ whose labels differ only in the e_1 position and add. This yields 2^{ν} new numbers. Next produce $2^{\nu+1}$ numbers by

multiplying each by $P(e_{\nu+2}=0)P(Z_{\nu+2}/e_{(2,\ldots,\nu+1,\nu+2=0)})$ and by $P(e_{\nu+2}=1)P(Z_{\nu+2}/e_{(2,\ldots,\nu+1,\nu+2=1)})$. This gives a new effective error tree of $2^{\nu+1}$ values of $q(e_{(2,\nu+2)})$ for use in deciding $\hat{e}_2$ during the next pass. Now go to Step 1 to process for $\hat{e}_2$, etc.

Thus the complete algorithm for estimating the error bit $e_{t-\Delta}$ at time t has been stated. This algorithm represents the optimal approach since the complete history of syndromes, namely $Z_{(1,t)}$ has been used to make the MAP error bit estimate with a finite delay Δ. This approach should provide performance the same as that of a MAP estimation of information bits. However the actual implementation is complicated by the non-equilikely nature of the error bits. This requires use of actual probabilities or logarithms of probabilities in the error bit case as opposed to the simpler Hamming distance metric useable in a Viterbi decoder for information bit sequences. With such a drawback due to the associated computation overhead, one is tempted to investigate possible suboptimal alternatives to the above using a finite length syndrome vector such as $Z_{(t-\Delta,t)}$. This will be done in Chapter 4.

The algorithm given above is complete for the case $\Delta > \nu$. For the case $\Delta \leq \nu$ the reader is referred to Appendix C which gives the missing Step 4 needed if $\Delta \leq \nu$.

3.4 Real Time Viterbi MAP Error Sequence Decoding

Conventional Viterbi data sequence decoding of convolutional codes has been shown by Viterbi [14] and Heller and Jacobs [17] to be a very attractive approach for the equilikely data situation. In such a case, the Viterbi algorithm has been shown by Viterbi [14] to choose $\hat{I}_{(1,N)}$ to be the information (data) sequence which maximizes the conditional probability $Pr(I_{(1,N)}/\underline{R}_{(1,N+\nu)})$. When all input data sequences $I_{(1,N)}$ are equilikely, significant simplification of the metric computations is possible. Indeed MAP and Maximum Likelihood estimators have the same form.

As another example of how conventional decoding algorithms adapt to error decoding, the conventional Viterbi approach will now be extended to error sequence estimation. This will require incorporation of the non-equilikely nature of the error inputs to the syndrome former and will assume a decoding delay requirement of Δ. Recall that for a BSC, higher weight error sequences are much less probable than low weight error sequences. Another complication is that, unlike the information sequence case, not all error bits enter into the syndrome former physical state. Some (e.g. $e_t^{(2)}$ in the rate $\frac{1}{2}$ systematic case) modify the syndrome former output Z_t but never actually residue in a syndrome former storage element. Making these two adjustments, the Real-Time Viterbi Decoding Algorithm of Lee [11] is rederived below for error sequences.

For a syndrome decoder with delay Δ, the Viterbi Decoding Rule for Error Sequences would be:

Choose $\hat{e}_t^{(1)}$ as the first bit of the syndrome former state S_{t+1} in the state sequence $S_{(1,t+\Delta+1)}$ which maximizes the conditional probability

$$\Pr(S_{(1,t+\Delta+1)}/Z_{(1,t+\Delta)}) = \frac{\Pr(S_{(1,t+\Delta+1)}, Z_{(1,t+\Delta)})}{\Pr(Z_{(1,t+\Delta)})} \qquad (3.86)$$

As in previous work the denominator is the same for all candidate state sequences. Thus the necessary maximization can be done on the joint probability in the numerator.

The syndrome former decoder for the K=5 code with finite delay is shown again in part in Figure 3.6 to clarify state and error bit relationships.

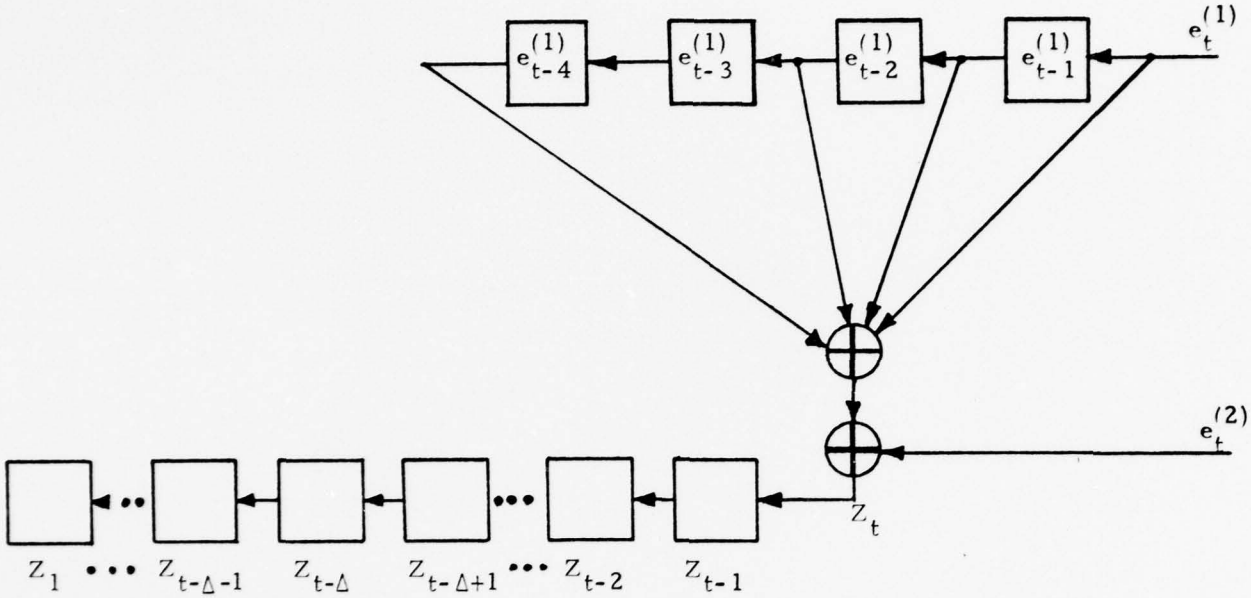
Recursion for Joint Probability

The numerator joint probability may be converted to a recursive form as follows:

$$\begin{aligned}
\Pr(S_{(1,t+1)}, Z_{(1,t)}) &= \Pr(S_{(1,t)}, S_{t+1}, Z_{(1,t-1)}, Z_t) \\
&= \Pr(S_{(1,t)}, Z_{(1,t-1)})\Pr(S_{t+1}, Z_t/S_{(1,t)}, Z_{(1,t-1)}) \\
&= \Pr(S_{(1,t)}, Z_{(1,t-1)})\Pr(S_{t+1}, Z_t/S_t)
\end{aligned}$$

$$\underbrace{\Pr(S_{(1,t+1)}, Z_{(1,t)})}_{\text{New Joint Probability}} = \underbrace{\Pr(S_{(1,t)}, Z_{(1,t-1)})}_{\text{Old Joint Probability}}\underbrace{\Pr(S_{t+1}/S_t)}_{\text{State Transition Probability}}\underbrace{\Pr(Z_t/S_t, S_{t+1})}_{\text{Syndrome Former Output Probability}} \qquad (3.87)$$

Syndrome Former State = $S_t = [e^{(1)}_{t-1}, e^{(1)}_{t-2}, \ldots, e^{(1)}_{t-4}]$



Syndrome Vector = $Z_{(1,t)} = [Z_t, Z_{t-1}, \ldots, Z_{t-\Delta}, \ldots, Z_1]$

Figure 3.6 Syndrome Former and Vector for K=5 (G2=11101) Code

In the above, the fact that the joint probability of S_{t+1} and Z_t, when conditioned on S_t, is not a function of either $Z_{(1,t-1)}$ or $S_{(1,t-1)}$ has been applied.

The Syndrome Former Output Probability may be computed by using:

$$\Pr(Z_t/S_t, S_{t+1}) = \sum_{\delta_t} \Pr(Z_t, \delta_t/S_t, S_{t+1})$$

$$= \sum_{\delta_t} \Pr(Z_t/S_t, S_{t+1}, \delta_t)\Pr(\delta_t) \qquad (3.88)$$

where δ_t = (those error bits in $\underline{E}_t$ which affect the syndrome Z_t but which do not appear in the syndrome former physical state (e.g. $e_t^{(2)}$ in K=5 example).)

The Syndrome Former Transition Probability may be computed by using:

$$\Pr(S_{t+1}/S_t) = \sum_{E_t^*} \Pr(S_{t+1}, E_t^*/S_t)$$

$$= \sum_{E_t^*} \Pr(S_{t+1}/E_t^*, S_t)\Pr(E_t^*) \qquad (3.89)$$

where E_t^* = (those error bits in $\underline{E}_t$ which actually enter into the syndrome former physical state (e.g. $e_t^{(1)}$ in the K=5 example).

Note that, unlike the conventional Viterbi data decoding case, syndrome former transition probabilities are not limited to 0 or $\frac{1}{2}$.

Rather they are determined by the channel error probability as represented by E_t^*.

Initial Conditions

If it is assumed that the syndrome former physical state at t=1 is $\underline{0}$, then the initial conditions for the above recursion are:

$$\Pr(S_{(1,2)}, Z_{(1,1)}) = \begin{cases} (1)\ \Pr(S_2/\underline{0})\Pr(Z_1/\underline{0}, S_2) \\ \text{if } \underline{0} \in \mathcal{Q}(S_2) \text{ and} \\ 0 \text{ if } \underline{0} \notin \mathcal{Q}(S_2) \end{cases} \tag{3.90}$$

Here

$$\Pr(S_2/\underline{0}) = \sum_{E_1^*} \Pr(S_2/E_1^*, \underline{0})\Pr(E_1^*)\,, \tag{3.91}$$

$\mathcal{Q}(S_i)$ = the allowed predecessor states which could transition to S_i,

and

$$\Pr(Z_1/\underline{0}, S_2) = \sum_{\delta_t} \Pr(Z_1/\underline{0}, S_2, \delta_t)\Pr(\delta_t)\,. \tag{3.92}$$

The best sequence $S_{(1,t+1)}$ for maximizing $\Pr(S_{(1,t+1)}, Z_{(1,t)})$ at time t+1 with $S_{t+1} = s$ must be the extension of one of the best sequences $S_{(1,t)}$ found at time t with S_t being one of the predecessors of s (i.e. $S_t \in \mathcal{Q}(s)$). In this error sequence decoding case each state s may have a number of predecessors. This condition gives rise to the more complex syndrome former state and trellis diagrams as noted in Schalkwijk and Vinck [13]. Thus in a Viterbi Error Sequence

Algorithm at time t+1, each syndrome former state s must have corresponding storage for only the best sequence to state s and the associated joint probability $\Pr(S_{(1,t+1)}, Z_{(1,t)})$.

The above adaptation has considered the entire state sequence $S_{(1,t)}$. For real time (i.e. finite delay = Δ) Viterbi error sequence decoding, only Δ elements of the best sequence for each state must be stored along with $\Pr(S_{(1,t)}, Z_{(1,t-1)})$.

Adaptation of Lee's Viterbi Algorithm

With the above clarifications, Lee's Real Time Viterbi Decoding Algorithm can be easily modified into the Real Time Viterbi Error Sequence Decoding Algorithm. Following the notation of Lee [11], the necessary metrics and sequences can be defined as: $h(s) = \Pr(S^{*}_{(1,t+1)}, Z_{(1,t)})$ for all syndrome former states where $S^{*}_{(1,t+1)}$ is the best path $S_{(1,t+1)}$ with $S_{t+1} = s$ and $B_i(s)$ = first digit of state S_{t+1-i}, in the path $S^{*}_{(1,t+1)}$. Note that $B_i(s) = \hat{e}^{(1)}_{t-i}$ in the example case of a rate 1/n systematic code.

Thus the following algorithm can be given.

Real Time Viterbi Error Sequence Decoding Algorithm ($\Delta > \nu$)

Step 0: Set t=1. Set $h(s) = \Pr(S^{*}_{(1,2)}, Z_{(1,1)}) = \Pr(s/\underline{0})\Pr(Z_1/\underline{0}, s)$ for all states having $\underline{0}$ as a predecessor. Set $h(s) = 0$ otherwise. Set $B_i(s) = 0$ for $1 \le i \le \Delta$ and for all states s.

Step 1: If $t \le \Delta$ go to Step 2. Otherwise set the error estimate $\hat{e}^{(1)}_{t-\Delta} = B_{\Delta}(s)$ where s is the state for which h(s) is a maximum.

Step 2: Increase t by 1. Replace $B_{\Delta-i+1}(s) \leftarrow B_{\Delta-i}(s)$ for $i = 1, 2, \ldots,$ $\Delta-1$ for each state s.

Step 3: For every s make the replacement $h(s) \leftarrow Pr(s/s')Pr(Z_t/s, s') \cdot h(s)$ where s' is the state maximizing the quantity on the right. Set $B_1(s)$ = 1st digit of s' for each s. Return to Step 1.

It has not been difficult in principle to adapt a finite delay Viterbi data sequence decoding algorithm to the decoding of error sequences. In practice, implementation of a Viterbi error sequence decoder would be more complex than a conventional Viterbi data sequence decoder. Unlike data sequences, which can be assumed equilikely, error sequences are very non-equiprobable. Thus the metrics involved in a Viterbi approach must be actual probabilities (or logarithms of probabilities) not merely simple functions of Hamming distances between received and candidate transmitted sequences. Appendix A, "Viterbi Algorithm for MAP Estimation," has additional details of the adaptation of the Viterbi Algorithm necessary with non-equilikely inputs.

3.5 Relation of Syndrome Decoding to MAP Data Bit Decoding

In any kind of MAP decoding the goal is to estimate some quantity A given some other quantity B by finding that value of A denoted by $\hat{A}$ which maximizes P(A/B).

Since

$$P(A/B) = \frac{P(A,B)}{P(B)} = \frac{P(B/A)P(A)}{P(B)} \qquad (3.93)$$

one can either

(1) determine a computable form of the expression for P(A/B) directly and then find the $\hat{A}$ which causes P(A/B) to be maximum; or

(2) determine a computable form of the expression P(B/A)P(A)/P(B) and determine $\hat{A}$ which causes P(B/A)P(A)/P(B) to be maximum.

With either approach one basic assumption is that the quantities A and B are related by some, usually probabilistic, relation or function.

Suppose we now apply the above to the problem of decoding convolutional codes. Here the fundamental problem is given a received vector $\underline{R}$ determine the best estimate of a particular information bit I_t. If we desire to use a MAP estimate then (3.93) becomes:

$$P(I_t/\underline{R}) = \frac{P(\underline{R}/I_t)P(I_t)}{P(\underline{R})} \qquad (3.94)$$

and one tries to determine that value of I_t (call it $\hat{I}_t$) which causes either side of (3.94) to be maximum. For example, this is the approach taken by McAdam [9] in deriving the MAP estimate of I_t given $\underline{R}$ by expressing the right hand side of (3.94) in terms of the encoder state sequences.

Usefulness of Syndromes

Now suppose one wants to investigate the estimation of I_t using a computed syndrome ($\underline{Z}$). One is tempted to write (3.93) as

$$P(I_t/\underline{Z}) = \frac{P(\underline{Z}/I_t)P(I_t)}{P(\underline{Z})} \tag{3.95}$$

and try to determine that value of I_t which maximizes either side of (3.95). However this would be wrong since the basic assumption that I_t is somehow related to $\underline{Z}$ is not true. Since the Syndrome $\underline{Z}$ is only related to the error bits $\{e_j^{(i)}\}$ and is independent of I_t, no matter how intricately one expands (3.95) even in terms of averaging over received vectors $\underline{R}$, the final result is equivalent to

$$P(I_t/\underline{Z}) = P(I_t) \tag{3.96}$$

Therefore we cannot estimate I_t given Z only.

Actually trying to estimate I_t given $\underline{Z}$ is the Data Processing Theorem at its worst. By the Data Processing Theorem, we can do no better than estimating I_t based on $P(I_t/R)$ since $\underline{R}$ is our received "raw" data. Any "data processor" (e.g. syndrome former) operating on the "raw data" $\underline{R}$ to yield some output (e.g. $\underline{Z}$) will tend to cause some information loss. In our syndrome case, this loss is all the information regarding I_t. Since the errors (e_t) are independent of the data (I_t) and the syndrome former output $\underline{Z}$ has no information regarding I_t, $\underline{Z}$ still has all the information regarding the errors $\{e_t^{(i)}\}$.

Consequently, we can just as well estimate e_t using $\underline{Z}$ by maximizing $P(e_t/\underline{Z})$ as we can estimate e_t using $\underline{R}$ by maximizing $P(e_t/\underline{R})$. If we obtain an estimate $\hat{e}_t$ by either method we must still go back to the raw data $\underline{R}$ to estimate the desired $\hat{I}_t$.

Since finite delay Δ and a discrete memoryless AWGN channel have been assumed, the error estimate $\hat{e}_{t-\Delta}^{(i)}$ can be subtracted from the appropriate $r_{t-\Delta}^{(i)}$ to yield $\hat{C}_{t-\Delta}^{(i)}$:

$$\hat{C}_{t-\Delta}^{(i)} = \underbrace{r_{t-\Delta}^{(i)}}_{\text{known}} - \underbrace{\hat{e}_{t-\Delta}^{(i)}}_{\text{estimated}} \tag{3.97}$$

With $\hat{C}_t^{(i)}$ estimated available, one can compute $\hat{I}_t$ deterministically. For the binary systematic rate 1/2 case sketched in Figure 3.7 , this is especially simple since $\hat{C}_t^{(1)} = \hat{I}_t$.

In the binary estimate case shown

$$I_t = C_t^{(1)} = r_t^{(1)} \oplus e_t^{(1)} . \tag{3.98}$$

A MAP estimator for I_t would use the rule:

Rule 1:

$$P(I_t = 0/\underline{R}) \underset{\text{decide } \hat{I}_t=1}{\overset{\text{decide } \hat{I}_t=0}{\gtrless}} P(I_t=1/\underline{R}) \tag{3.99}$$

If this rule is rewritten using the fact that $I_t = r_t^{(1)} \oplus e_t^{(1)}$ and the fact that $r_t^{(1)} \in \underline{R}$, one obtains

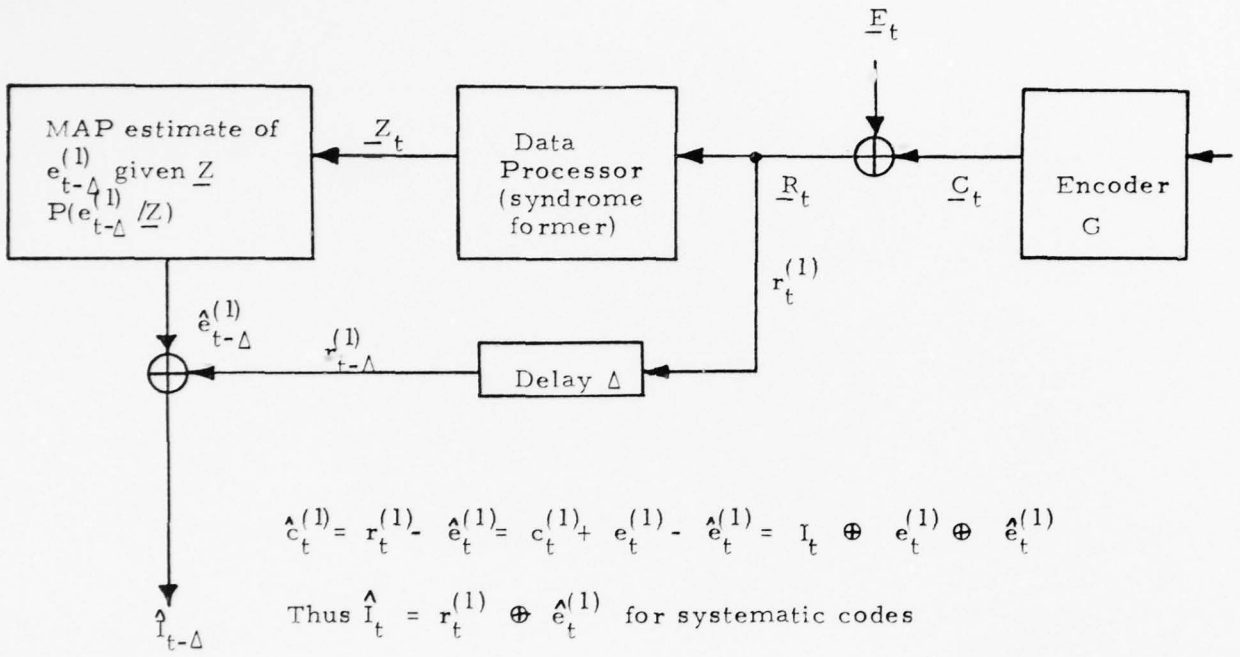
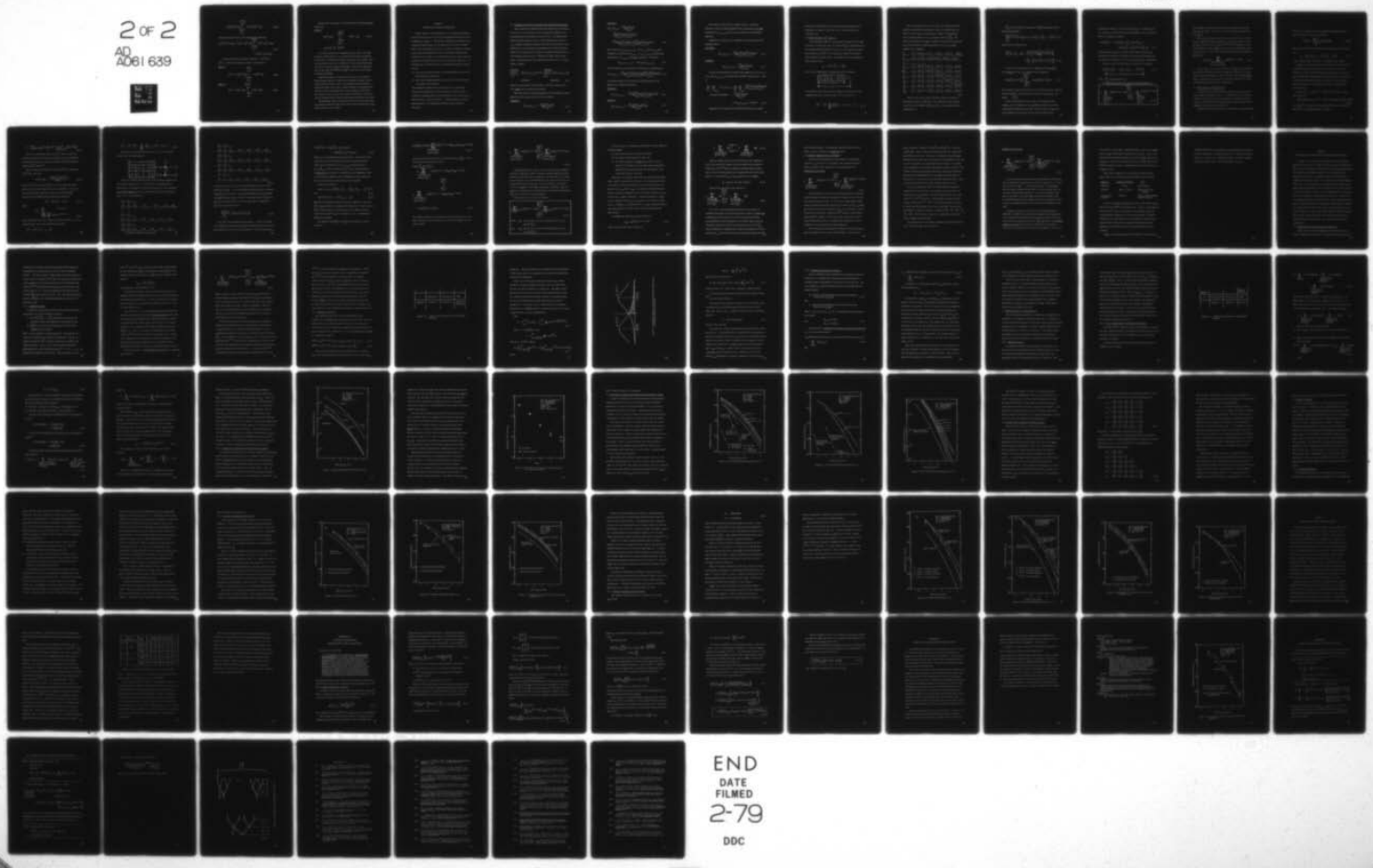


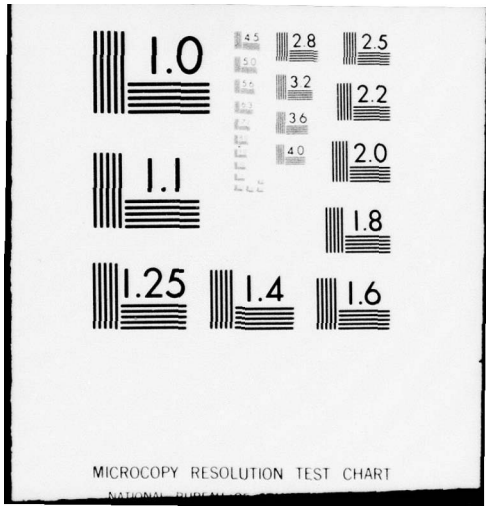
Figure 3.7 Simplified Sketch of Error Decoding of Binary Systematic Rate 1/2 Convolutional Code

AD-A061 639 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/4
MAP ERROR BIT DECODING OF CONVOLUTIONAL CODES.(U)
AUG 77 J T KINDLE
UNCLASSIFIED AFIT-CI-79-13T NL

2 OF 2
AD
A061 639



END
DATE
FILMED
2-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

$$P(r_t^{(1)} \oplus e_t^{(1)}=0/\underline{R}) \underset{\text{decide } \hat{I}_t=1}{\overset{\text{decide } \hat{I}_t=0}{\gtrless}} P(r_t^{(1)} \oplus e_t^{(1)}=1/\underline{R}) \quad . \tag{3.100}$$

The binary (mod 2) sums can be split into the following:

$$P(r_t^{(1)}=0, e_t^{(1)}=0/\underline{R}) + P(r_t^{(1)}=1, e_t^{(1)}=1/\underline{R}) \underset{\text{decide } \hat{I}_t=1}{\overset{\text{decide } \hat{I}_t=0}{\gtrless}} P(r_t^{(1)}=0, e_t^{(1)}=1/\underline{R}) + P(r_t^{(1)}=1, e_t^{(1)}=0/\underline{R}) \quad . \tag{3.101}$$

Consider the only two possible situations: (1) $r_t^{(1)}=0$; or $r_t^{(1)}=1$. *In these two situations,* Rule 1 *is equivalent to*

Rule 1.a

$$r_t^{(1)} = 0: \; P(e_t^{(1)}=0/\underline{R}) \underset{\text{decide } \hat{I}_t=0}{\overset{\text{decide } \hat{I}_t=0}{\gtrless}} P(e_t^{(1)}=1/\underline{R}) \tag{3.102}$$

Rule 1.b

$$r_t^{(1)}=1: \; P(e_t^{(1)}=1/\underline{R}) \underset{\text{decide } \hat{I}_t=1}{\overset{\text{decide } \hat{I}_t=0}{\gtrless}} P(e_t^{(1)}=0/\underline{R}) \tag{3.103}$$

Instead of the above rule, one could employ the following error-based rule:

RULE 2:

$$P(e_t^{(1)}=0/\underline{R}) \quad \begin{matrix} \text{decide } \hat{e}_t^{(1)}=0 \\ > \\ < \\ \text{decide } \hat{e}_t^{(1)}=1 \end{matrix} \quad P(e_t^{(1)}=1/\underline{R}) \qquad (3.104)$$

and set $\hat{I}_t = r_t^{(1)} \oplus \hat{e}_t^{(1)}$.

It is easily seen that for all possible cases, Rule 1 (as implemented by either Rules 1.a or 1.b) will give the same estimate $\hat{I}_t$ as will Rule 2. Thus both methods of estimating I_t are equally valid in the systematic code case considered. That is, selecting $\hat{I}_t$ to be that value of I_t which maximizes $P(I_t/\underline{R})$ is equivalent to setting $\hat{I}_t$ equal to the value $r_t^{(1)} \oplus \hat{e}_t^{(1)}$ where $\hat{e}_t^{(1)}$ is that value of $e_t^{(1)}$ which maximizes $P(e_t^{(1)}/\underline{R})$.

Basically the practical reason for trying to use a syndrome based decoder instead of a MAP data bit decoder using $\underline{R}$ is that the implementation might be easier. The syndrome-based estimator portion estimates errors with a single syndrome $\underline{Z}$ containing information on errors only followed by a mod 2 addition to obtain $\hat{I}_t$ in the systematic case. On the other hand the MAP Data bit estimator must work with $\underline{R}$ which contains information on both data and errors.

The following chapter will pursue MAP error bit decoding based on syndromes under the dual restrictions of finite decoding delay and finite syndrome vector length.

Chapter 4

SUBOPTIMAL ERROR ESTIMATION

In this chapter certain suboptimal error estimation techniques are derived which may be more easily implemented than the optimal techniques of Chapter 3. Specifically, ROMMAP ans SUBMAP, two suboptimal approximations to the optimal real time (fini'e delay) MAP error bit estimator of Section 3.3 will be considered. Both use a precomputed look-up table of error bit estimates addressed by syndrome vector values. It is felt that such a finite delay estimation problem best represents the case to be encountered in practice. In the following, the restrictions on the ROMMAP/SUBMAP technique will be:

(1) only a specified finite delay Δ is permitted before an error bit decision must be made; and

(2) only a finite length syndrome vector $\underline{Z}$ (or received vector $\underline{R}$) is available to the decoder.

The ROMMAP technique will be first shown to be a particular average of the optimal finite delay estimator. Next the actual methods of computing MAP error bit estimates given either $\underline{R}$ or $\underline{Z}$ will be derived. Lastly, SUBMAP, a computationally convenient approximation to the suboptimal ROMMAP approach will be discussed.

4.1 ROMMAP Derived from Optimal Finite Delay Error Decoding

First consider the optimal finite delay MAP error bit decoding of Section 3.3, and compare it to the suboptimal ROMMAP error decoding approach mentioned in Section 2.4.2. The optimal finite delay decoding attempts to make a MAP estimate of an error bit $e_{t-\Delta}$ using the computed syndrome bits up to the current time t but allowing only a finite storage of syndrome bits in such a decision. However it incorporates the influence of syndrome bits prior to t-L in making this MAP decision. ROMMAP decoding, on the other hand, essentially ignores the syndrome bits prior to t-L. In other words, compare

$$\underset{\substack{\text{over all}\\ \text{values of}\\ e_{t-\Delta}}}{\text{maximum}} \left\{ P(e_{t-\Delta}/z_{(1,t)}) \right\} \text{ versus } \underset{\substack{\text{over all}\\ \text{values of}\\ e_{t-\Delta}}}{\text{maximum}} \left\{ P(e_{t-\Delta}/z_{(t-\Delta,t)}) \right\} \tag{4.1}$$

OPTIMAL ROMMAP

where for now we have taken the delay Δ from the current time t to be equal to the storage delay length L.

Note first that the required probability for the ROMMAP and the optimal can (via Bayes rule) be written as follows.

ROMMAP:

$$P(e_{t-\Delta}/z_{(t-\Delta,t)}) = \frac{P(e_{t-\Delta}, z_{(t-\Delta,t)})}{P(z_{(t-\Delta,t)})} \tag{4.2}$$

OPTIMAL:

$$P(e_{t-\Delta}/z_{(1,t)}) = \frac{P(e_{t-\Delta}, z_{(1,t)})}{P(z_{(1,t)})}$$

$$= \frac{P(e_{t-\Delta}, z_{(1,t-\Delta-1)}, z_{(t-\Delta,t)})}{P(z_{(1,t-\Delta-1)}, z_{(t-\Delta,t)})}$$

$$= \frac{P(z_{(t-\Delta,t)}/e_{t-\Delta}, z_{(1,t-\Delta-1)})P(e_{t-\Delta}, z_{(1,t-\Delta-1)})}{P(z_{(t-\Delta,t)}/z_{(1,t-\Delta-1)})P(z_{(1,t-\Delta-1)})} \qquad (4.3)$$

Now consider $P(e_{t-\Delta}, z_{(1,t-\Delta-1)})$. Since $e_{t-\Delta}$ affects $z_{t-\Delta}$ and ν subsequent syndrome bits $z_{t-\Delta+1}, \ldots, z_{t-\Delta+\nu}$, $e_{t-\Delta}$ is statistically independent of $z_{(1,t-\Delta-1)}$ (the <u>past</u> syndromes). Therefore,

$$P(e_{t-\Delta}, z_{(1,t-\Delta-1)}) = P(e_{t-\Delta})P(z_{(1,t-\Delta-1)}) \qquad (4.4)$$

and the Optimal case probability of interest can be expressed as:

$$P(e_{t-\Delta}/z_{(1,t)}) = \frac{P(z_{(t-\Delta,t)}/e_{t-\Delta}, z_{(1,t-\Delta-1)})P(e_{t-\Delta})P(z_{(1,t-\Delta-1)})}{P(z_{(t-\Delta,t)}/z_{(1,t-\Delta-1)})P(z_{(1,t-\Delta-1)})} \qquad (4.5)$$

Cancelling equal terms of numerator and denominator in the optimal case yields the following.

OPTIMAL:

$$P(e_{t-\Delta}/z_{(t-\Delta,t)}) = \frac{P(e_{t-\Delta})P(z_{(t-\Delta,t)}/e_{t-\Delta}, z_{(1,t-\Delta-1)})}{P(z_{(t-\Delta,t)}/z_{(1,t-\Delta-1)})} \qquad (4.6)$$

ROMMAP:

$$P(e_{t-\Delta}/z_{(t-\Delta,t)}) = \frac{P(e_{t-\Delta})P(z_{t-\Delta,t)}/e_{t-\Delta})}{P(z_{(t-\Delta,t)})} \qquad (4.7)$$

Note that (4.6) and (4.7) are similar in form. The basic difference is that the optimal quantities are conditioned on the past syndrome bits $z_{(1, t-\Delta-1)}$ which are not available to the ROMMAP approach.

Equations (4.6) and (4.7) can simply be rearranged to give the following forms.

OPTIMAL:

$$P(e_{t-\Delta}/z_{(1,t)}) = \frac{P(e_{t-\Delta}, z_{(t-\Delta, t)}/z_{(1, t-\Delta-1)})}{P(z_{(t-\Delta, t)}/z_{(1, t-\Delta-1)})} \qquad (4.8)$$

ROMMAP:

$$P(e_{t-\Delta}/z_{(t-\Delta, t)}) = \frac{P(e_{t-\Delta}, z_{(t-\Delta, t)})}{P(z_{(t-\Delta, t)})} \qquad (4.9)$$

Now by taking the formal average of the Optimal Eq. (4.8) or Eq. (4.3) over $z_{(1, t-\Delta-1)}$, one obtains the ROMMAP equation (4.9). That is

$$\sum_{z_1} \cdots \sum_{z_{t-\Delta-1}} P(e_{t-\Delta}/z_{(1,t)}) = \sum_{z_1} \cdots \sum_{z_{t-\Delta-1}} \frac{P(e_{t-\Delta}, z_{(1, t-\Delta-1)}, z_{(t-\Delta, t)})}{P(z_{(1, t-\Delta-1)}, z_{(t-\Delta, t)})}$$

$$\text{Average of OPTIMAL} = \frac{P(e_{t-\Delta}, z_{(t-\Delta, t)})}{P(z_{(t-\Delta, t)})}$$

$$= P(e_{t-\Delta}/z_{(t-\Delta, t)}) \quad \text{ROMMAP} \qquad (4.10)$$

Equation (4.10) shows that the ROMMAP technique is actually

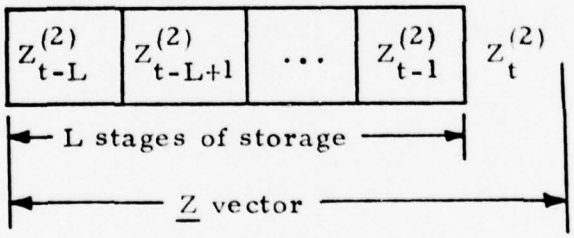
just what one would obtain from the optimal approach if one was required to average over all "past" (i.e. times less than $t-\Delta$) syndrome bits.

4.2 MAP Estimate of $e^{(1)}_{t-\Delta}$ given Z

In the ROMMAP approach to syndrome decoding of the example K-5 rate 1/2 systematic code, an estimate $\hat{e}^{(1)}_{t-\Delta}$ is obtained based on the computed finite syndrome vector $\underline{Z}$. It is assumed that sucessive syndrome bits $\{Z^{(i)}_t\}$ are computed and stored in associated shift registers of length L bits. The syndrome vector available in this example is thus:

$$\underline{Z} = (Z^{(2)}_t, Z^{(2)}_{t-1}, \ldots, Z^{(2)}_{t-L})$$

This vector would appear in storage elements as follows.



It is useful to write out part of the syndrome equation set for this example using the convention of Eq. (2.18) of Section 2.3.

$$Z^{(2)}_i = e^{(2)}_i + \sum_{h=0}^{\nu} G^{(2)}_{1,h} e^{(1)}_{i-h} \quad , \quad t-L \le i \le t, \quad \nu=4 \qquad (4.11)$$

This is done below in Eq. (4. 12) for $L > \Delta$ and can be used to gauge the complexity of a particular choice of L and the effect of feedback on this same set of equations. Shown is the particular set for decoding delay $\Delta = 5$ (i.e. estimate error bit $e^{(1)}_{t-5}$). The dotted line designates the upper "triangle" of error bits which would have been previously estimated and fed back if error estimate feedback were employed.

$$
\underline{Z}\begin{cases}
Z_{t-L} = e^{(2)}_{t-L} + G^{(2)}_0 e^{(1)}_{t-L} + G^{(2)}_1 e^{(1)}_{t-L-1} + G^{(2)}_2 e^{(1)}_{t-L-2} + G^{(2)}_3 e^{(1)}_{t-L-3} + G^{(2)}_4 e^{(1)}_{t-L-4} \\
Z_{t-L+1} = e^{(2)}_{t-L+1} + G^{(2)}_0 e^{(1)}_{t-L+1} + G^{(2)}_1 e^{(1)}_{t-L} + G^{(2)}_2 e^{(1)}_{t-L-1} + G^{(2)}_3 e^{(1)}_{t-L-2} + G^{(2)}_4 e^{(1)}_{t-L-3} \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
Z_{t-5} = e^{(2)}_{t-5} + G^{(2)}_0 e^{(1)}_{t-5} + G^{(2)}_1 e^{(1)}_{t-6} + G^{(2)}_2 e^{(1)}_{t-7} + G^{(2)}_3 e^{(1)}_{t-8} + G^{(2)}_4 e^{(1)}_{t-9} \\
Z_{t-4} = e^{(2)}_{t-4} + G^{(2)}_0 e^{(1)}_{t-4} + G^{(2)}_1 e^{(1)}_{t-5} + G^{(2)}_2 e^{(1)}_{t-6} + G^{(2)}_3 e^{(1)}_{t-7} + G^{(2)}_4 e^{(1)}_{t-8} \\
Z_{t-3} = e^{(2)}_{t-3} + G^{(2)}_0 e^{(1)}_{t-3} + G^{(2)}_1 e^{(1)}_{t-4} + G^{(2)}_2 e^{(1)}_{t-5} + G^{(2)}_3 e^{(1)}_{t-6} + G^{(2)}_4 e^{(1)}_{t-7} \\
Z_{t-2} = e^{(2)}_{t-2} + G^{(2)}_0 e^{(1)}_{t-2} + G^{(2)}_1 e^{(1)}_{t-3} + G^{(2)}_2 e^{(1)}_{t-4} + G^{(2)}_3 e^{(1)}_{t-5} + G^{(2)}_4 e^{(1)}_{t-6} \\
Z_{t-1} = e^{(2)}_{t-1} + G^{(2)}_0 e^{(1)}_{t-1} + G^{(2)}_1 e^{(1)}_{t-2} + G^{(2)}_2 e^{(1)}_{t-3} + G^{(2)}_3 e^{(1)}_{t-4} + G^{(2)}_4 e^{(1)}_{t-5} \\
Z_t = e^{(2)}_t + G^{(2)}_0 e^{(1)}_t + G^{(2)}_1 e^{(1)}_{t-1} + G^{(2)}_2 e^{(1)}_{t-2} + G^{(2)}_3 e^{(1)}_{t-3} + G^{(2)}_4 e^{(1)}_{t-4}
\end{cases}
\tag{4.12}
$$

Given an $\underline{Z}$ vector we want to find the maximum a posteriori probability estimate of $e^{(1)}_{t-\Delta}$ which in the systematic code example can then be mod 2 added to $r^{(1)}_{t-\Delta}$ to give $\hat{I}_{t-\Delta}$, the estimated $(t-\Delta)$th information bit.

$\hat{e}^{(1)}_{t-\Delta}$ is thus set equal to that value of $e^{(1)}_{t-\Delta}$ which maximizes the following probability:

$$\max_{\substack{\text{over all values v of} \\ e^{(1)}_{t-\Delta}}} \left\{ \Pr(e^{(1)}_{t-\Delta}=v/\underline{Z}) = \Pr(e^{(1)}_{t-\Delta}=v/(Z^{(2)}_{t-L}, \ldots, Z^{(2)}_{t}) \right\} \quad (4.13)$$

But this can be expressed as:

$$\max_{\text{over } v} \left\{ \Pr(e^{(1)}_{t-\Delta}=v/\underline{Z}) \right\} = \max_{\text{over } v} \left\{ \frac{\Pr(\underline{Z}/e^{(1)}_{t-\Delta}=v)\Pr(e^{(1)}_{t-\Delta}=v)}{\Pr(\underline{Z})} \right\}$$

$$= \max_{\text{over } v} \left\{ \Pr(\underline{Z}/e^{(1)}_{t-\Delta}=v)\Pr(e^{(1)}_{t-\Delta}=v) \right\} \quad (4.14)$$

This decision rule requires the computation of $\Pr(\underline{Z}/e^{(1)}_{t-\Delta}=v)$.

In the binary case the rule simplifies to the following:

$$\Pr(\underline{Z}/e^{(1)}_{t-\Delta}=1)\Pr(e^{(1)}_{t-\Delta}=1) \underset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=0}{\overset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=1}{\gtrless}} \Pr(\underline{Z}/e^{(1)}_{t-\Delta}=0)\Pr(e^{(1)}_{t-\Delta}=0) \quad (4.15)$$

For example, if the left side of Eq. (4.15) is greater than the right side, then decide $\hat{e}^{(1)}_{t-\Delta}=1$; if the left side is less than the right side, then decide $\hat{e}^{(1)}_{t-\Delta}=0$.

Since no assumption has been made of an orthogonal code, $\Pr(\underline{Z}/e_{t-\Delta}=1)$ for example can be expanded in terms of each of the conditional probabilities of the other error bits and averaged over the probability of occurrence of each error bit value for the

memoryless channel. For the BSC, the $Pr(e_k=1) = p$ and $Pr(e_k=0) = q = 1-p$ where p is the BSC crossover probability. That is, the first step in the expansion would be:

$$Pr(\underline{Z}/e^{(1)}_{t-\Delta}=1) = Pr(\underline{Z}/e^{(1)}_{t-\Delta}=1, e^{(1)}_{t-\Delta+1}=1)Pr(e^{(1)}_{t-\Delta+1}=1) + Pr(\underline{Z}/e^{(1)}_{t-\Delta}=1, e^{(1)}_{t-\Delta+1}=0)Pr(e^{(1)}_{t-\Delta+1}=0) \quad (4.16)$$

This expansion would be continued for both sides of the decision rule until all conditional probabilities for all error bits appearing in syndrome bit equation set for the specified $|\underline{Z}|$ were considered.

An error vector can be defined as in Section 2.3 as follows.

$$\underline{E} \triangleq (e^{(2)}_{t-L}, e^{(1)}_{t-L}, \ldots, e^{(1)}_{t-\Delta-1}, e^{(1)}_{t-\Delta+1}, \ldots, e^{(2)}_{t}, e^{(1)}_{t}) \quad (4.17)$$

(2(L+1)-1 error bits in $\underline{E}$)

(Note: $e^{(1)}_{t-\Delta}$ is not included in $\underline{E}$.)

The Maximum A posteriori Error Bit Decision Rule then is

$$p \sum_{\substack{\text{all } \underline{E} \text{ satisfying this } \underline{Z} \text{ for } e^{(1)}_{t-\Delta}=1}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \underset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=0}{\overset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=1}{\gtrless}} q \sum_{\substack{\text{all } \underline{E} \text{ satisfying this } \underline{Z} \text{ for } e^{(1)}_{t-\Delta}=0}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \quad (4.18)$$

For example, if the left side of Eq. (4.18) is greater than the right side, decide $e_{t-\Delta}^{(1)} = 1$; if the left side is less than the right side, decide $e_{t-\Delta}^{(1)} = 0$.

For a given E_b/N_0 (and therefore a given BSC crossover probability ($Pr(e_k^{(i)} = 1)$) the above computation can be carried out for each possible $\underline{Z}$ vector in the set $\{\underline{Z}\}$ of $2^{(L+1)} = 2^{LSYN}$ syndrome vectors. The resultant decisions $e_{t-\Delta}^{(1)}(\underline{Z})$ can be stored in a ROM addressed by its associated $\underline{Z}$. Also stored could be the actual decision confidence $p_1(\underline{Z})$ where

$$p_1(\underline{Z}) \triangleq p \sum_{\substack{\text{all } \underline{E} \text{ satisfying} \\ \text{this } \underline{Z} \text{ with } e_{t-\Delta}^{(1)}=1}} q^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} . \qquad (4.19)$$

This storage of the set $\{\underline{Z}\}$ each with its $e_{t-\Delta}^{(1)}(\underline{Z})$ is the ROMMAP - the Read Only Memory Map from a computed syndrome vector $\underline{Z}$ to its corresponding Maximum A posteriori Error Bit Estimate $e_{t-\Delta}^{(1)}(\underline{Z})$. Such a look-up table incorporated in a decoder enables error correction with essentially no decoder computations.

4.3 MAP Estimate of $e_{t-\Delta}^{(1)}$ Given $\underline{R}$

It is instructive to now temporarily examine just what it would take to make a ROMMAP-like (i.e., a look-up table) error estimator using the actual received vector $\underline{R}$ instead of a computed syndrome vector $\underline{Z}$. This diversion will demonstrate why the syndrome based approach of Section 4.2 is so attractive.

In basic MAP data bit decoding of a systematic convolutional code,

a decision or estimate $\hat{I}_t$ of the most probable information bit I_t is made by finding that value of I_t such that

$$\Pr(\hat{I}_t/\underline{R}) = \max_{\substack{\text{over all} \\ \text{values of} \\ I_t}} \left\{\Pr(I_t/\underline{R})\right\} \tag{4.20}$$

Here $\underline{R}$ is some selected segment of received bits (a finite number) e.g.,

$$\underline{R} = (r_{t-L}^{(1)}, r_{t-L+1}^{(1)}, \ldots, r_t^{(1)}, r_{t-L}^{(2)}, \ldots, r_t^{(2)}) \tag{4.21}$$

in our standard binary systematic convolutional code example.

McAdam [9] found a forward and backward recursion method for determining $\hat{I}_t$ given the total received vector. It might be useful to see what a ROMMAP approach to finding the most likely I_t given a finite segment $\underline{R}$ would give for a binary rate 1/n systematic code when equilikely data inputs are assumed. That is, assume a finite decoding delay Δ and storage for a finite segment $\underline{R}$ of the received vector.

Recall that $I_t = r_t^{(1)} - e_t^{(1)} = r_t^{(1)} \oplus e_t^{(1)}$ (over GF(2)). Then from Section 3.5

$$(\text{most probable } I_t/\text{given } r_t^{(1)}) = r_t^{(1)} - (\text{most probable } e_t^{(1)}/\text{given } r_t^{(1)}). \tag{4.22}$$

Note that $r_t^{(1)}$ is available and known while $\hat{e}_t^{(1)}$ must be estimated. Also $r_t^{(1)}$ is one component of $\underline{R}$. Therefore

$$\hat{I}_t = \underset{\text{maximizing}}{\text{that } I_t} \left\{ Pr(I_t/\underline{R}) \right\} = r_t^{(1)} - \underset{\text{maximizing}}{\text{that } e_t^{(1)}} \left\{ Pr(e_t^{(1)}/\underline{R}) \right\} \tag{4.23}$$

Since in the systematic code case $r_t^{(1)}$ is known, finding the maximum a posteriori estimate of I_t given $\underline{R}$ can be accomplished by finding the maximum a posteriori estimate of $e_t^{(1)}$ given $\underline{R}$ and subtracting $\hat{e}_t^{(1)}$ from $r_t^{(1)}$.

Now concentrate on finding that value of $e_t^{(1)}$ which maximizes $Pr(e_t^{(1)}/\underline{R})$. However,

$$Pr(e_t^{(1)}=v/\underline{R}) = \frac{Pr(\underline{R}/e_t^{(1)}=v)Pr(e_t^{(1)}=v)}{Pr(\underline{R})} \tag{4.24}$$

where $Pr(e_t^{(1)}=1) = p$ and $Pr(e_t^{(1)}=0) = 1-p = q$ for the BSC. $\underline{R}$ is given as received and $Pr(\underline{R}/e_t^{(1)})$ must be computed. This can be done using the same approach as for the case of $Pr(\underline{Z}/e_{t-\Delta}^{(1)})$ discussed in the ROMMAP determination by noting that

$$r_t^{(j)} = c_t^{(j)} + e_t^{(j)} \qquad 1 \le j \le n \tag{4.25}$$

where

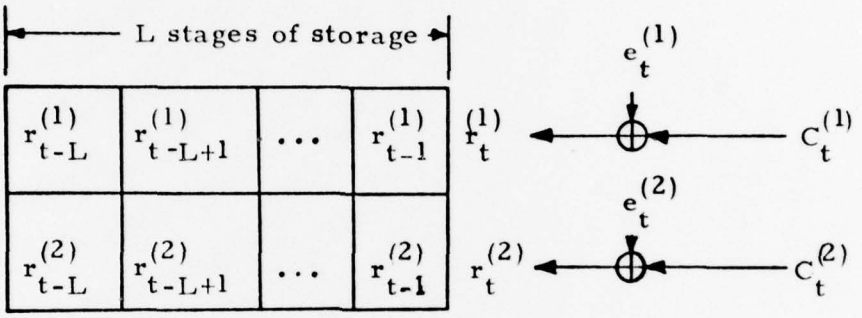
$$C_t^{(j)} = \begin{cases} I_t & \text{if } 1 \le j \le k \\ \sum_{\ell=1}^{k} \sum_{h=0}^{\nu} G_{\ell h} I_{t-h} & \text{if } k < j \le n . \end{cases} \tag{4.26}$$

Or, in the specific case of k=1 (only one info bit per block), ν=4, rate 1/2 systematic code of the example this becomes:

$$r_i^{(1)} = C_i^{(1)} + e_i^{(1)} = I_i + e_i^{(1)}$$

$$r_i^{(2)} = C_i^{(2)} + e_i^{(2)} = \sum_{h=0}^{\nu} G_{1h}^{(2)} I_{i-h} + e_i^{(2)}, \quad t-L \le i \le t . \tag{4.27}$$

If we have L bits of storage for the (j) r_i, $1 \le j \le 2$, received bit streams, our available $\underline{R}$ is:



To show a comparison between the set of syndrome equations of the ROMMAP approach (Eq. 4.12) with the corresponding set of equations for estimating error bits from an $\underline{R}$ vector, the comparable received bit equations for the example rate 1/2 code is given below (Eq. (4.28) for arbitrary L.

$$r_{t-L}^{(1)} = e_{t-L}^{(1)} + I_{t-L}$$

$$r_{t-L}^{(2)} = e_{t-L}^{(2)} + G_0^{(2)} I_{t-L} + G_1^{(2)} I_{t-L-1} + G_2^{(2)} I_{t-L-2} + G_3^{(2)} I_{t-L-3} + G_4^{(2)} I_{t-L-4}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$r_{t-5}^{(1)} = e_{t-5}^{(1)} + I_{t-5}$$

$$r_{t-5}^{(2)} = e_{t-5}^{(2)} + G_0^{(2)} I_{t-5} + G_1^{(2)} I_{t-6} + G_2^{(2)} I_{t-7} + G_3^{(2)} I_{t-8} + G_4^{(2)} I_{t-9}$$

$$r_{t-4}^{(1)} = e_{t-4}^{(1)} + I_{t-4}$$

$$r_{t-4}^{(2)} = e_{t-4}^{(2)} + G_0^{(2)} I_{t-4} + G_1^{(2)} I_{t-5} + G_2^{(2)} I_{t-6} + G_3^{(2)} I_{t-7} + G_4^{(2)} I_{t-8}$$

($\underline{R}$ equations continued on next page)

$$
\begin{aligned}
r_{t-3}^{(1)} &= e_{t-3}^{(1)} + I_{t-3} \\
r_{t-3}^{(2)} &= e_{t-3}^{(2)} + G_0^{(2)} I_{t-3} + G_1^{(2)} I_{t-4} + G_2^{(2)} I_{t-5} + G_3^{(2)} I_{t-6} + G_4^{(2)} I_{t-7} \\
r_{t-2}^{(1)} &= e_{t-2}^{(1)} + I_{t-2} \\
r_{t-2}^{(2)} &= e_{t-2}^{(2)} + G_0^{(2)} I_{t-2} + G_1^{(2)} I_{t-3} + G_2^{(2)} I_{t-4} + G_3^{(2)} I_{t-5} + G_4^{(2)} I_{t-6} \\
r_{t-1}^{(1)} &= e_{t-1}^{(1)} + I_{t-1} \\
r_{t-1}^{(2)} &= e_{t-1}^{(2)} + G_0^{(2)} I_{t-1} + G_1^{(2)} I_{t-2} + G_2^{(2)} I_{t-3} + G_3^{(2)} I_{t-4} + G_4^{(2)} I_{t-5} \\
r_{t}^{(1)} &= e_{t}^{(1)} + I_{t} \\
r_{t}^{(2)} &= e_{t}^{(2)} + G_0^{(2)} I_{t} + G_1^{(2)} I_{t-1} + G_2^{(2)} I_{t-2} + G_3^{(2)} I_{t-3} + G_4^{(2)} I_{t-4}
\end{aligned}
\tag{4.28}
$$

A total of $n(L+1)$ bits are received and held in $\underline{R}$ where $n=2$ in this particular example. As in the ROMMAP computations we can expand $Pr(\underline{R}/e_{t-\Delta}^{(1)})$ in terms of the conditional probabilities for all available error and information bits. That is we want that value of $e_{t-\Delta}^{(1)}$ which maximizes equation (4.24). But this is equivalent to finding that $e_{t-\Delta}^{(1)}$ value which produces the

$$
\begin{matrix} \text{max over} \\ \text{all values} \\ e_{t-\Delta}^{(1)} \end{matrix} \left\{ Pr(\underline{R}/e_{t-\Delta}^{(1)}) Pr(e_{t-\Delta}^{(1)}) \right\} \tag{4.29}
$$

This value can be computed from equation (4.24) by expanding $Pr(\underline{R}/e_{t-\Delta}^{(1)})$ repeatedly until all conditional probabilities for both error and information bits have been incorporated. The first step would be:

$$\Pr(\underline{R}/e^{(1)}_{t-\Delta}=v) = \Pr(\underline{R}/e^{(1)}_{t-\Delta}=v, x_k=1)\Pr(x_k=1)$$

$$+ \Pr(\underline{R}/e^{(1)}_{t-\Delta}=v, x_k=0)\Pr(x_k=0) \qquad (4.30)$$

where x_k is some particular info or error bit. At each step in the expansion the new bit introduced (x_k) takes on another identity - either some information bit (e.g. I_{t-L+k}) or some error bit (e.g. $e^{(i)}_{t-L+k}$). When working with $\underline{R}$ it is here that the presence of information bits must be accounted for in the computation; while in the syndrome ROMMAP case, all dependence upon information bits was removed when syndrome bits were computed.

Using the definitions,

$$\text{Error vector} = \underline{E} \triangleq \left[e^{(2)}_{t-L}, e^{(1)}_{t-L}, \ldots, e^{(1)}_{t-\Delta-1}, e^{(1)}_{t-\Delta+1}, \ldots, e^{(2)}_{t}, e^{(1)}_{t}\right] \qquad (4.31)$$

and

$$\text{Information vector} = \underline{I} \triangleq \left[I_{t-L-4}, \ldots, I_t\right] \qquad (4.32)$$

provides a notationally simpler form for the expansion. Note that $e^{(1)}_{t-\Delta}$ is not included as a component of the $\underline{E}$ vector. Also if x_k is an error bit then $\Pr(x_k=1) = p$ and $\Pr(x_k=0) = 1-p$, while if x_k is an information bit then $\Pr(x_k=1) = \Pr(x_k=0) = \frac{1}{2}$ (i.e., assuming equi-likely source inputs).

The quantity $\Pr(\underline{R}/e^{(1)}_{t-\Delta}=1)\Pr(e^{(1)}_{j-\Delta}=1)$ for example can now be written as

$$\Pr(\underline{R}/e_{t-\Delta}^{(1)}=1)\Pr(e_{t-\Delta}^{(1)}=1)=p\sum_{\substack{\text{all } \underline{EI} \\ \text{satisfying this} \\ \underline{R} \text{ with } e_{t-\Delta}^{(1)}=1}} p^{w(\underline{E})}q^{2(L+1)-1-w(\underline{E})}P(I=1)^{w(\underline{I})}P(I=0)^{(L+5-w(\underline{I}))}. \tag{4.33}$$

But with equilikely inputs $P(I=1) = P(I=0) = \frac{1}{2}$ and since $e_{t-\Delta}^{(1)}$ is 1 or 0, the new form of the decision rule is:

$$\Pr(\underline{R}/e_{t-\Delta}^{(1)}=1)\Pr(e_{t-\Delta}^{(1)}=1)$$

$$= p \sum_{\substack{\text{all } \underline{EI} \\ \text{satisfying this} \\ \underline{R} \text{ with } e_{t-\Delta}^{(1)}=1}} p^{w(\underline{E})}q^{2(L+1)-1-w(\underline{E})}\left(\tfrac{1}{2}\right)^{w(\underline{I})}\left(\tfrac{1}{2}\right)^{(L+5)-w(\underline{I})}$$

$$\begin{matrix}\text{decide} \\ \hat{e}_{t-\Delta}^{(1)}=1 \\ > \\ < \\ \hat{e}_{t-\Delta}^{(1)}=0\end{matrix}$$

$$q \sum_{\substack{\text{all } \underline{EI} \\ \text{satisfying this} \\ \underline{R} \text{ with } e_{t-\Delta}^{(1)}=1}} p^{w(\underline{E})}q^{2(L+1)-1-w(\underline{E})}\left(\tfrac{1}{2}\right)^{w(\underline{I})}\left(\tfrac{1}{2}\right)^{(L+5)-w(\underline{I})}$$

$$= \Pr(\underline{R}/e_{j-\Delta}^{(1)}=0)\Pr(e_{j-\Delta}^{(1)}=0)\ . \tag{4.34}$$

For equilikely inputs the contribution of the information bits to each side of the above rule is the same, so they can be dropped from the rule to obtain

$$p \sum_{\substack{\text{all } \underline{EI} \\ \text{satisfying this} \\ \underline{R} \text{ with } e^{(1)}_{t-\Delta}=1}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \underset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=0}{\overset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=1}{>\!<}} q \sum_{\substack{\text{all } \underline{EI} \\ \text{satisfying this} \\ \underline{R} \text{ with } e^{(1)}_{t-\Delta}=0}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} .$$

(4.35)

Comparing the above rule Eq. (4.35) with that for the ROMMAP estimate of $e^{(1)}_{t-\Delta}$ (Eq. (4.18)), one observes that the quantities being summed are the same but the conditions over which the sum is taken are not identical. In the ROMMAP case, the sum is over Error Vectors $\underline{E}$ satisfying the current $\underline{Z}$. In Eq. (4.35) above, the sum is over Error and Info vectors $\underline{EI}$ satisfying the current $\underline{R}$. Again the problem of considering the information bits exists when one estimates $e^{(1)}_{t-\Delta}$ based on the set of received bits $\underline{R}$.

This decision rule can also be written as

$$p \sum_{\text{all } \underline{E}} N_{E1} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \underset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=0}{\overset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=1}{>\!<}} q \sum_{\text{all } \underline{E}} N_{E0} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \qquad (4.36)$$

where N_{E1} = number of $\underline{I}$ vectors such that $\underline{EI}$ satisfy current $\underline{R}$ with $e^{(1)}_{t-\Delta}=1$

and N_{E0} = number of $\underline{I}$ vectors such that $\underline{EI}$ satisfy current $\underline{R}$ with $e^{(1)}_{t-\Delta}=0$.

At each step in the computation of the above sum over all $\underline{E}$, the following applies:

(1) $\underline{R}$ is known from the received bit streams,

(2) $\underline{E}$ is known for this particular step, and

(3) we need to compute the number of $\underline{I}$ vectors (out of a total of 2^{L+5} possible $\underline{I}$ vectors) which when substituted along with this particular $\underline{E}$ into the "$\underline{R}$ equations" will satisfy them for the current $\underline{R}$.

Since $\underline{R}$ is given and $\underline{E}$ is known for this particular step, $\underline{R}$-$\underline{E}$ either equals a code word or it does not. It is assumed here that L (the number of stages of storage for each received bit stream) is greater than L_{min} where L_{min} is a constraint length-determined number sufficient to guarantee that each possible input bit stream is encoded into a unique "codeword." With this assumption, if $\underline{R}$-$\underline{E}$ is not a codeword then N_{E1} (or N_{E0} as the case may be) is equal to zero. If $\underline{R}$-$\underline{E}$ is a codeword then it is a unique codeword and only one of the possible $\underline{I}$ vectors satisfies the $\underline{R}$ equations for this particular $\underline{E}$.

To simplify the following make the definition:

$$f(\underline{E}) \triangleq p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})}. \tag{4.37}$$

The decision rule then can be written as:

$$p \sum_{\substack{\text{all } \underline{EI} \text{ satisfying this} \\ \underline{R} \text{ with } e^{(1)}_{t-\Delta}=1}} f(\underline{E}) \underset{\text{decide "0"}}{\overset{\text{decide "1"}}{\gtrless}} q \sum_{\substack{\text{all } \underline{EI} \text{ satisfying this} \\ \underline{R} \text{ with } e^{(1)}_{t-\Delta}=0}} f(\underline{E}) \cdot \quad (4.38)$$

Again to simplify notation let us define the set of "codewords" $\{C_k\}$ to be those $2(L+1)$-tuples which are output by the encoder in rssponse to the 2^{L+5} possible input data streams under consideration. Also, for those $\underline{E}$ which are considered by the decision rule sums (i.e. the $\underline{E}$ whose N_{E1} or N_{E0} equal one) we can apply

$$\underline{E} = \underline{R} \oplus C_k \text{ for some } C_k \in \{C_k\}. \quad (4.39)$$

The decision rule then can be written as:

$$p \sum_{\substack{\text{all } \underline{E}=C_k \oplus \underline{R} \\ \text{for all } C_k \in \{C_k\} \\ \text{with } e^{(1)}_{t-\Delta}=1}} f(\underline{E}) \underset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=0}{\overset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=1}{\gtrless}} q \sum_{\substack{\text{all } \underline{E}=C_k \oplus \underline{R} \\ \text{for all } C_k \in \{C_k\} \\ \text{with } e^{(1)}_{t-\Delta}=0}} f(\underline{E}) \cdot \quad (4.40)$$

Eq. (4.40) above then equivalently directs the decoder to consider all possible error patterns $\underline{E}$ which could have mapped <u>any</u> codeword $\underline{C}_k$ into the actual received vector $\underline{R}$ and weigh them proportionately by their probability of occurrence $f(\underline{E})$ to make the maximum a posteriori decision on this particular error bit $e^{(1)}_{t-\Delta}$. Thus the additional complexity that arises by attempting to estimate error bits $e_{t-\Delta}$ from the received vector $\underline{R}$ without using syndromes

has been demonstrated. Consequently, the bulk of this work considers error bit estimation on syndromes only.

4.4 SUBMAP Approximation to ROMMAP

In Section 4.2 the ROMMAP decision rule for estimating the error bit $e_{t-\Delta}$ for K=5 rate $\frac{1}{2}$ systematic code example with finite delay Δ and only a finite syndrome vector $\underline{Z}$ was shown to be:

ROMMAP Decision Rule

$$p \sum_{\substack{\text{all } \underline{E} \\ \text{satisfying this} \\ \underline{Z} \text{ with } e_{t-\Delta}^{(1)}=1}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \underset{\text{decide } \hat{e}_{t-\Delta}^{(1)}=0}{\overset{\text{decide } \hat{e}_{t-\Delta}^{(1)}=1}{>\!<}} q \sum_{\substack{\text{all } \underline{E} \\ \text{satisfying this} \\ \underline{Z} \text{ with } e_{t-\Delta}^{(1)}=0}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \tag{4.41}$$

This ROMMAP decision rule can be implemented by a read only memory addressed by $\underline{Z}$ and containing the single error bit estimate for each $\underline{Z}$ given by the rule. To create such a look-up table would require considering all error patterns $\{\underline{E}'\}$ where $\underline{E}' \triangleq (\underline{E}, e_{t-\Delta})$ (thus $\underline{E}'$ contains all error bits including the one to be estimated). In such a binary case this means considering $2^{|\underline{E}'|}$ separate error patterns where $|\underline{E}'| \triangleq$ the number of error bits in this overall error vector. Such an exponential growth in computational complexity is a basic drawback to any look-up table decoding approach.

This drawback can be mitigated somewhat by taking advantage of the non-equilikely nature of the error bit values. On a practical

binary symmetric channel, the $Pr(e_t^{(i)}=1) << Pr(e_t^{(i)}=0)$. Thus only "lightweight" error vectors (those having few 1's) have a significant probability of occurring. Therefore, if instead of considering all possible error vectors $\{\underline{E}'\}$ as dictated by the ROMMAP decision rule, one was to consider only a lightweight subset of $\{\underline{E}'\}$, the computational complexity of creating the look-up table could be reduced. The resultant SUBMAP look-up table would not be significantly different than that of the full ROMMAP since the neglected contributions to the sums due to "heavy" $\underline{E}$ in 4.41 would be extremely small. Thus the decisions found for either table would be almost the same.

To see the complexity reduction in look-up table computation, consider the K=5 rate $\frac{1}{2}$ code. If a syndrome vector of LSYN = 15 syndrome bits is considered, the look-up table in either case would contain $2^{15} = 2^{LSYN}$ locations. The ROMMAP computation would require considering $2^{33+1} = 2^{34} = 17,179,869,210$ error vectors $\underline{E}'$. If instead one only includes error vectors of weight 5 or less, then the SUBMAP computation requires considering $\binom{34}{0} + \binom{34}{1} + \binom{34}{2} + \binom{34}{3} + \binom{34}{4} + \binom{34}{5} = 331,212$ error vectors for a complexity reduction by a factor of approximately 51,870.

For completeness the SUBMAP computation rule for the previous rate $\frac{1}{2}$ binary example may be written as:

SUBMAP Decision Rule

$$p \sum_{\substack{\text{all } \underline{E} \text{ satisfying this} \\ \underline{Z} \text{ such that } w(\underline{E}) \leq 4 \\ \text{with } e_{t-\Delta}^{(1)}}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})} \underset{\text{decide } e_{t-\Delta}^{(1)}=0}{\overset{\text{decide } e_{t-\Delta}^{(1)}=1}{>}} q \sum_{\substack{\text{satistying this} \\ \underline{Z} \text{ such that} \\ w(\underline{E}) \leq 5}} p^{w(\underline{E})} q^{2(L+1)-1-w(\underline{E})}. \tag{4.42}$$

Due to the rapid exponential growth in computation, in this work the full ROMMAP computation was carried out only for a K = 5 rate $\frac{1}{2}$ code for the case $|\underline{Z}| = 6$ = LSYN syndrome bits. All subsequent work for longer constraint length codes or more syndrome bits concentrated on the SUBMAP approsch. It is shown in Chapter 5 that there is available a very tight upper bound on performance using the SUBMAP approach which enables the accurate prediction of performance simultaneously with the creation of the SUBMAP look-up table.

A specific comment is in order on the relationship between ROMMAP, SUBMAP, and conventional feedback decoder approaches. The conventional feedback decoder as described by Heller [16] uses as the error bit estimate that value of $e_{t-\Delta}^{(1)}$ appearing in the minimum weight error vector that could have caused the given syndrome vector $\underline{Z}$. This means that the many-to-one mapping of $\{E\}$ vectors

into syndrome vectors $\{\underline{Z}\}$ is approximated by a one-to-one mapping using only one of the lightest weight error vectors $\underline{E}$. This is contrasted with the ROMMAP approach in which the entire many-to-one mapping is considered in making the error bit estimate and with the SUBMAP approach in which a significant portion of the many-to-one mapping is used.

Thus the three approaches can be ranked by how close they approximate the true finite delay-finite syndrome MAP estimate:

Approach	Mapping Considered	$\hat{e}$ is:
ROMMAP	Entire	MAP estimate
SUBMAP	Subset of entire	Approximate MAP estimate
Conventional Feedback	Single $\underline{E}$	Grosser approximation to MAP estimate

Note that in all three approaches the size of the decoder look-up table is the same ($2^{|\underline{Z}|}$ in the binary case). Thus one is logically led to assume that, given a specification on the maximum $|\underline{Z}|$ (and consequently the maximum size of the look-up table), it would be advisable to use look-up table entries computed by considering as large a subset of $\{\underline{E}\}$ vectors as possible thus approaching the ROMMAP or true Finite Delay-Finite Syndrome MAP error bit estimate.

Chapter 5 demonstrates that this is indeed the case since both

computed performance and simulation results show that the SUBMAP decoders outperform conventional feedback (i.e. minimum weight $\underline{E}$) decoders. In many cases a SUBMAP decoder even without feedback will perform better than a conventional feedback decoder.

Chapter 5

PERFORMANCE PREDICTION AND SIMULATION RESULTS

In Chapter 3 several theoretically optimal error bit decoding approaches were derived. Chapter 4 took one of these, optimal finite delay error bit decoding, and exhibited two suboptimal approximations, the ROMMAP and SUBMAP decoders. In this chapter decoder performance prediction is analyzed for these two decoders when implemented for typical codes. It is shown that an actual probability of bit error can be computed for the ROMMAP decoder while a tight upper bound on performance of the SUBMAP decoder can be obtained as the look-up table is created. A lower bound based on an idealized first error probability is developed and shown to give a lower limit to continued performance improvement for a given code and decoding delay. Also the performance improvement possible both by changing to a longer constraint length code and by increasing $|\underline{Z}|$ for a given code is displayed. Lastly, Monte Carlo SUBMAP simulation results for two codes using various $|\underline{Z}|$ values with and without feedback are presented and compared with conventional feedback decoder performance.

5.1 ROMMAP Creation and Predicted Performance

In this section the actual computation of a ROMMAP look-up table to implement a binary MAP finite delay-finite syndrome error

decoding rule is qualitatively described along with the manner of computing the actual probability of bit error for the ROMMAP decoder. This look-up table is addressed by the current syndrome vector value $\underline{Z}$. Defining LSYN $\triangleq |\underline{Z}|$ = the number of syndrome bits in the syndrome vector means that the size of the ROMMAP table must be 2^{LSYN}. Each addressed storage location in the table must contain either a 0 or 1 value corresponding to the MAP error bit estimate $\hat{e}_{t-\Delta}^{(1)}$ determined by the rule. This value will sometimes be denoted as $\hat{e}_{t-\Delta}^{(1)}(\underline{Z})$ in the following explanation for a binary rate $\frac{1}{2}$ systematic code.

5.1.1 ROMMAP Creation

It should be specifically noted that the two entities of interest to our decoder for a rate $\frac{1}{2}$ systematic code are:

(1) $\underline{Z} = (Z_t^{(2)}, Z_{t-1}^{(2)}, \ldots, Z_{t-L}^{(2)}$ (provided in real time by the syndrome former and syndrome bit shift register); and

(2) $\hat{e}_{t-\Delta}^{(1)}$ (the error bit to be estimated by reference to the ROMMAP look-up table).

Here L is the "syndrome bit delay parameter" indicating that L+1 consecutive syndrome bits comprise the syndrome vector $\underline{Z}$. It is always assumed that $Z_t^{(2)}$ is the latest syndrome bit output by the syndrome former. Also Δ is the "error bit delay parameter" indicating that Δ+1 noisy information bits $r_j^{(1)}$, $j = t, t-1, \ldots, t-\Delta$, are delayed and available to be corrected. This means that at time t,

when $r_t^{(1)}$ and $Z_t^{(2)}$ become available to the decoder, the ROMMAP decoder will produce $\hat{e}_{t-\Delta}^{(1)}$ by "looking up" the value $\hat{e}_{t-\Delta}^{(1)}(\underline{Z})$ in the ROMMAP table and mod 2 add it to $r_{t-\Delta}^{(1)}$ to give the information bit estimate

$$\hat{I}_{t-\Delta} = r_{t-\Delta}^{(1)} + \hat{e}_{t-\Delta}^{(1)}(\underline{Z})$$

Consequently the "decoding delay" is Δ.

The distinction between Δ and L is worth making clear since, although syndrome decoding schemes such as conventional feedback decoding usually implement $L \approx \Delta$, there is no inherent reason why the decoding delay Δ must equal the syndrome bit delay L. In the work that follows $L \geq \Delta$.

For convenience, the actual implementations discussed in this and later sections were restricted to binary systematic codes. For such codes the total ROMMAP table approach may be mechanized in a hardwired device which only has to determine if the current syndrome vector $\underline{Z}$ belongs to the subset of syndromes such that $\hat{e}_{t-\Delta}^{(1)}(\underline{Z}) = 1$. That is, the contents of a ROMMAP table actually map the 2^{LSYN} syndrome vectors into two points, (0, 1). A study of the redundancy in the syndromes-to-decision mapping was not pursued in the work but would seem to be a possible means of speeding decisions when compared to a slow look-up table search.

For later reference, the ROMMAP decision rule for a systematic rate $\frac{1}{2}$ code is:

$$p \sum_{\substack{\text{all } \underline{E} \\ \text{satisfying this} \\ \underline{Z} \text{ with } e^{(1)}_{t-\Delta}=1}} p^{w(\underline{E})}(1-p)^{LEN-w(\underline{E})} \underset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=0}{\overset{\text{decide } \hat{e}^{(1)}_{t-\Delta}=1}{>\!<}} (1-p) \sum_{\substack{\text{all } \underline{E} \\ \text{satisfying this} \\ \underline{Z} \text{ with } e^{(1)}_{t-\Delta}=0}} p^{w(\underline{E})}(1-p)^{LEN-w(\underline{E})} \tag{5.2}$$

where: $\underline{E}$ is the vector of all error bits entering into the set of LSYN syndrome equations other than the bit $e^{(1)}_{t-\Delta}$ to be estimated; LEN is the length of $\underline{E}$; $\underline{Z}$ is the current syndrome vector of length $|\underline{Z}| = LSYN$; and p is the BSC crossover probability. Note that Eq. (5.2) is merely the decision rule resulting from considering only the numerator of the ROMMAP probability in Eq. 4.14 and specializing to a rate $\frac{1}{2}$ binary case.

For the relatively short constraint length binary examples of ROMMAP and SUBMAP decoders considered here, the first step in implementation is the creation of the look-up table, i.e. the mapping from syndrome vectors $\underline{Z}$ to a particular error bit estimate $\hat{e}^{(1)}_{t-\Delta}$. In the ROMMAP case this mapping is done by deterministically generating all 2^{LEN} error patterns, substituting each into the right hand side of the appropriate set of syndrome equations with the error bit to be estimated set to 0 and then to 1. For each of the 2^{LEN} error patterns this gives two syndromes (equivalent to each of the

2^{LEN+1} overall error patterns mapping to one syndrome). A data structure consisting of a double vector of probabilities indexed by the 2^{LSYN} possible syndrome values was used to accumulate contributions to $P(\underline{Z}/e^{(1)}_{t-\Delta}=1)$ and $P(\underline{Z}/e^{(1)}_{t-\Delta}=0)$ as all error patterns were processed. Once all error patterns had been processed, the data structure entries are multiplied by p and 1-p respectively to yield the terms of the decision rule (5.2), for each $\underline{Z}$. By cycling through all $\underline{Z}$ locations in the data structure the MAP error bit decision $\hat{e}^{(1)}_{t-\Delta}(\underline{Z})$ at each $\underline{Z}$ can be made based on these terms and recorded in a third level of the data structure to give a triple vector. *This third level is the ROMMAP look-up table as shown in Figure 5.1.*

5.1.2 Signal to Noise Ratio

One should note here that since the contributions to the $P(e^{(1)}_{t-\Delta}=1/\underline{Z})$ and $P(e^{(1)}_{t-\Delta}=0/\underline{Z})$ numerators are based on the crossover probability p of the binary symmetric channel, it is necessary to consider how this p is related to the signal to noise (SNR) level of the received signal. For a particular $\underline{E}$ vector of weight $w(\underline{E})$ for example, the contribution would be

$$p^{w(\underline{E})}(1-p)^{LEN+1-w(\underline{E})} \text{ for the syndrome when } e^{(1)}_{t-\Delta} = 0 \text{ and} \qquad (5\text{-}3a)$$

$$p^{w(\underline{E})+1}(1-p)^{LEN-w(\underline{E})} \text{ for the syndrome when } e^{(1)}_{t-\Delta} = 1 \qquad (5\text{-}3b)$$

The crossover probability p can be related back to a signal to noise ratio assuming additive white Gaussian noise and a particular

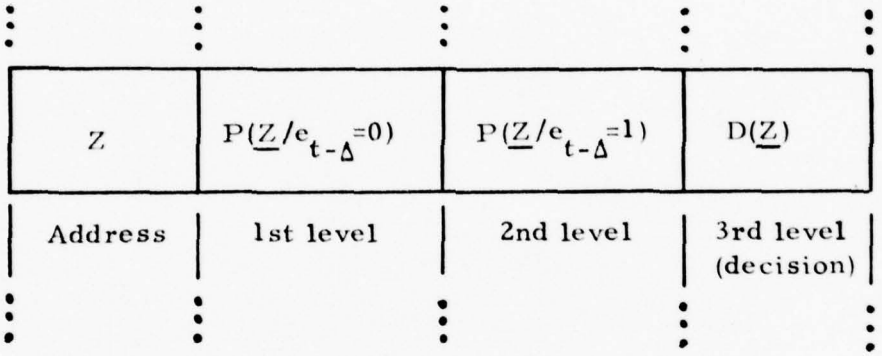


Figure 5.1. Triple Level Data Structure for ROMMAP Creation.

modulation. Thus any ROMMAP look-up table has associated with it a "SNR design value" corresponding to the p used in the probability contribution computations.

In the case of 180 degree binary phase shift keying (BPSK) modulation assumed throughout this work, one can easily relate the BSC crossover probability p to SNR level. The additive white Gaussian noise causes the received channel symbol level v to have the conditional distributions shown in Figure 5.2 assuming proper receiver gain adjustment to provide unit variance as noted by McAdam [9]. Note that in Fig. 5.2 the mean of the conditional distributions is $\pm\sqrt{2E_s/N_0}$ and the cross-hatched area can be used to compute the BSC crossover probability p.

Thus

$$p = \int_{v=0}^{v=+\infty} P(v/0)dv = \int_{v=0}^{v=+\infty} \frac{1}{\sqrt{2\pi}} \exp^{-\frac{1}{2}(v+\sqrt{2E_s/N_0})^2} dv. \tag{5.4}$$

With $x = v + \sqrt{2E_s/N_0}$ one has

$$p = \int_{x=+\sqrt{2E_s/N_0}}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp^{-\frac{1}{2}x^2} dx$$

and with $u = x/\sqrt{2}$ one obtains

$$p = \frac{1}{\sqrt{\pi}} \int_{u=\sqrt{E_s/N_0}}^{\infty} \exp^{-u^2} du = \frac{1}{2}\left(\frac{2}{\sqrt{\pi}} \int_{u=\sqrt{E_s/N_0}}^{+\infty} e^{-u^2} du = \frac{1}{2} \operatorname{erfc}(\sqrt{E_s/N_0})\right. \tag{5.5}$$

where

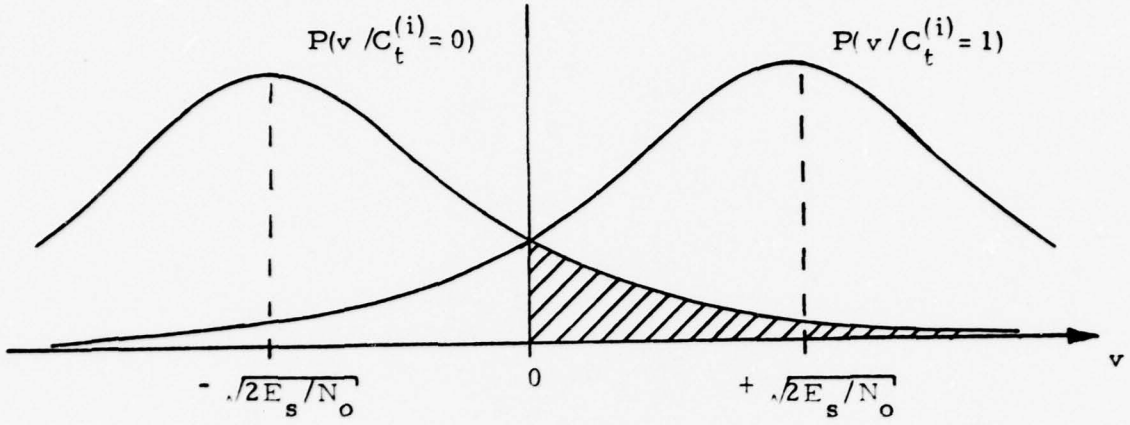


Figure 5.2 Conditional Densities of Received Channel Symbol Level for BPSK Modulation and AWGN

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \exp^{-t^2} dt.$$

This can also be expressed as

$$p = \tfrac{1}{2}[1-\text{erf}(E_s/N_0)] \text{ where } \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp^{-t^2} dt \qquad (5.6)$$

as noted in Wu [15]. In the above it should be emphasized that

$E_s \triangleq$ average signal energy per transmitted bit (symbol energy)

and

N_0 = noise power density.

If one desires p for BPSK to be expressed in terms of the more conventional E_b, the uncoded signal energy per bit for a rate R code, then, using $E_s/N_0 = (E_b/N_0)R$, the BSC crossover probability becomes:

$$p = \tfrac{1}{2}[1-\text{erf}(E_b R/N_0)] \qquad (5.7)$$

where R = the code rate.

Since all codes actually implemented and compared in this work were rate $R = \frac{1}{2}$, both predicted and simulated data for probability of bit error are plotted in terms of E_s/N_0 in db, the transmitted symbol signal to noise ratio. This is consistent with the approach of other authors (e.g. Heller [16], Schalkwijk and Vinck [13]) who tend to plot "P_{out} versus P_{in}." However some caution must be used in comparing results of other work (e.g. Heller and Jacobs [17]) where P_{out} (equivalent to P_e herein) is plotted in terms of E_b/N_0.

5.1.3 ROMMAP Probability of Bit Error

Once the ROMMAP look-up table has been computed, sufficient information is available in the aforementioned data structure to actually compute a probability of error for the error decision. The set of addresses $\{\underline{Z}\}$ of the ROMMAP table can be divided into two disjoint subsets:

$$Z_0 = \{\text{the set of all syndrome vectors for which look-up table entries } \hat{e}_{t-\Delta}(\underline{Z})=0\} \tag{5.8}$$

and

$$Z_1 = \{\text{the set of all syndrome vectors for which look-up table entries } \hat{e}_{t-\Delta}(\underline{Z})=1\}$$

Thus $Z = Z_0 \cup Z_1$ and $Z_0 \cap Z_1 = \emptyset$. The ROMMAP decoder operates by deciding:

and

$$\begin{aligned} \hat{e}_{t-\Delta} &= 0 \text{ if } \underline{Z} \epsilon Z_0 \\ \hat{e}_{t-\Delta} &= 1 \text{ if } \underline{Z} \epsilon Z_1 \ . \end{aligned} \tag{5.9}$$

To determine the probability of making a wrong error bit decision P_e, it is convenient to define:

$$P_{e,0} \triangleq \{\text{probability of making a wrong error bit decision when } e_{t-\Delta}=0\}$$

$$= \sum_{\underline{Z} \epsilon Z_1} P(\underline{Z}/e_{t-\Delta}=0) \tag{5.10}$$

and

$$P_{e,1} \triangleq \{\text{probability of making a wrong error bit decision when } e_{t-\Delta}=1\}$$

$$= \sum_{\underline{Z} \epsilon Z_0} P(\underline{Z}/e_{t-\Delta}=1) \qquad (5.11)$$

Averaging over both possible values of $e_{t-\Delta}$ gives the average error probability P_e:

$$P_e = P(e_{t-\Delta}=0)P_{e,0} + P(e_{t-\Delta}=1)P_{e,1} \qquad (5.12)$$

The quantities $P(\underline{Z}/e_{t-\Delta}=0)$ and $P(\underline{Z}/e_{t-\Delta}=1)$ appearing in (5.10) and (5.11) are the same as the quantities computed in creating the ROMMAP look-up table. Thus (5.10) and (5.11) can be evaluated for the particular code and particular LSYN value by summing over those data structure addresses $\underline{Z}$ for which $\hat{e}_{t-\Delta}(\underline{Z})=1$ and $\hat{e}_{t-\Delta}(\underline{Z})=0$ respectively. Which $\underline{Z}$ is in which subset is already flagged by the 0, 1 entries at each address $\underline{Z}$. This implies that, for a particular systematic binary code, the computation necessary for the creation of a ROMMAP table also provides a theoretical value for probability of decoding error P_e at the design SNR. For example Figure 5.4 shows ROMMAP P_e as computed for a K=5 code versus design SNR (lower curve).

Since the actual ROMMAP performance can be calculated, the use of any Chernoff or Simple Bound approach as used in Gallagher [18] for performance prediction is not warranted. Indeed when one pursues either approach it is found that the same quantities

$P(\underline{Z}/e_{t-\Delta}=0)$ and $P(\underline{Z}/e_{t-\Delta}=1)$ are needed for either bound evaluation so one might just as well compute P_e directly as described above.

Thus is has been shown that if a look-up table is created for a ROMMAP decoder, an actual probability of bit decoding error is readily available. However, as noted in Section 4.4, the number of error vectors $\underline{E}$ that must be considered for the full ROMMAP grows exponentially with syndrome vector length, LSYN. Thus one is led to consider the SUBMAP look-up table approach and a method of predicting or bounding P_e for SUBMAP decoding. This bound is considered in the following section.

5.2 SUBMAP Creation and Performance

Due to the exponential growth in computational complexity with syndrome vector length for the ROMMAP decoder, it is feasible to use the full ROMMAP approach only for quite short constraint lengths. Consequently it is worthwhile to consider creating a SUBMAP look-up table decoder as discussed in Section 4.4. This will be done in the following sections along with consideration of performance prediction and the relationship to conventional feedback decoding.

5.2.1 SUBMAP Creation

In computing a SUBMAP look-up table, the same three level data structure described above for the ROMMAP approach is used. It is loaded with probability contributions as before but only a lightweight subset of all possible $\underline{E}'$vectors is considered. Thus

contributions from the subset of $\underline{E}'$ vectors have been entered, the data structure entries at each $\underline{Z}$ address are $P^*(\underline{Z}/e_{t-\Delta})$, $P^*(\underline{Z}/e_{t-\Delta})$, and $D(\underline{Z}) = \hat{e}^{(1)}_{t-\Delta}(\underline{Z})$. The "*" is used to differentiate these quantities from the corresponding ROMMAP quantities since they will be only approximately the same. The SUBMAP entries will be less than or equal to the ROMMAP entries. That is $P(\underline{Z}/e_{t-\Delta}) = P^*(\underline{Z}/e_{t-\Delta}) + \delta$ where δ is the difference due to the neglected "heavyweight" vectors If the maximum weight of error vectors (MAX) in the lightweight subset is increased toward LEN+1 the SUBMAP and ROMMAP tables will become identical. Just how similar the performances of the two approaches will be is explained in the following section. For reference purposes the three level SUBMAP data structure can be visualized as depicted in Figure 5.3.

5.2.2 A Tight Upper Bound On SUBMAP Performance

Since the SUBMAP approach creates a look-up table by considering only a lightweight subset of the total set of possible error vectors, the effect on performance of the excluded error vectors must be measured or bounded.

Consider expressing the average error probability P_e for the SUBMAP case as follows:

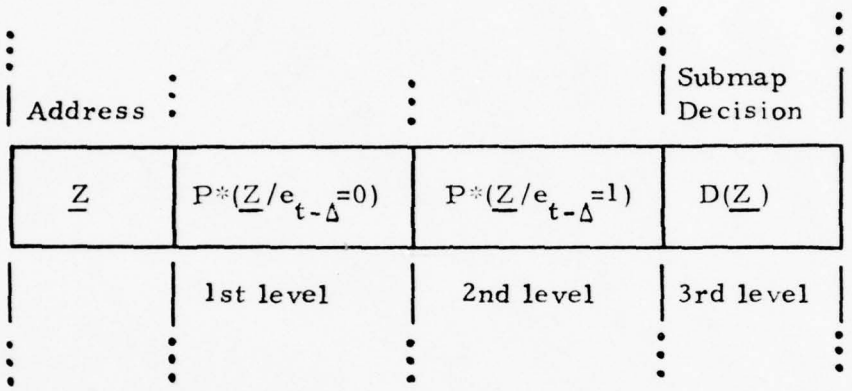


Figure 5.3. Triple-level Data Structure for SUBMAP creation.

$$P_e = \sum_{\substack{\text{all } \underline{E}' \\ \text{vectors}}} P(\text{error}/\underline{E}')P(\underline{E}') = \sum_{\substack{\text{all } \underline{E}' \text{vectors} \\ \text{considered in} \\ \text{SUBMAP}}} P(\text{error}/\underline{E}')P(\underline{E}')$$

$$+ \sum_{\substack{\text{all } \underline{E}' \text{ vectors} \\ \text{excluded from} \\ \text{SUBMAP}}} P(\text{error}/\underline{E}')P(\underline{E}') \qquad (5.13)$$

In the second summation above, the quantity $P(\text{error}/\underline{E}') \leq 1$. To upper bound P_e, set $P(\text{error}/\underline{E}') = 1$ only in the second term. In other words assume a decoding error is made with probability equal to one whenever an excluded $\underline{E}'$ occurs. This yields the bound:

$$P_e \leq \sum_{\substack{\text{all } \underline{E}' \text{vectors} \\ \text{considered in} \\ \text{SUBMAP}}} P(\text{error}/\underline{E}')P(\underline{E}') + \sum_{\substack{\text{all } \underline{E}' \text{vectors} \\ \text{excluded from} \\ \text{SUBMAP}}} (1)P(\underline{E}) \qquad (5.14)$$

Now for notational convenience define:

(1) $D(\underline{Z}) = \hat{e}_{t-\Delta}(\underline{Z})$ = the SUBMAP table 0, 1 decision addressed by Z;

(2) $\underline{Z}(\underline{E}')$ = the syndrome vector which corresponds to a particular error vector $\underline{E}'$;

(3) $\hat{P} = \sum_{\substack{\text{all } \underline{E}' \text{vectors} \\ \text{considered in} \\ \text{SUBMAP}}} P(\text{error}/\underline{E}')P(\underline{E}')$; and (4) $P_{skip} = \sum_{\substack{\text{all } \underline{E}' \text{vectors} \\ \text{excluded from} \\ \text{SUBMAP}}} (1)P(\underline{E}')$. (5.15)

Thus

$$P_e \leq \hat{P} + P_{skip} \tag{5.16}$$

Now consider $\hat{P}$. A decision mistake can be made in the following two ways when an $\underline{E}'$ vector contained in the subset considered in deriving the SUBMAP table is present:

(1) the error vector $\underline{E}'$ is such that $e_{t-\Delta}=1$ but $D(\underline{Z}(\underline{E}))=0$; or

(2) the error vector $\underline{E}'$ is such that $e_{t-\Delta}=0$ but $D(\underline{Z}(E))=1$.

In case (1) the contribution to the sum due to such an $\underline{E}'$ having $e_{t-\Delta}=1$ is

$$\begin{aligned} P(\text{error}/\underline{E}')P(\underline{E}') &= P(D(\underline{Z}(\underline{E}'))=0)P(\underline{E}') \\ &= (1-D(\underline{Z}(\underline{E}'))P(\underline{E}') . \end{aligned} \tag{5.17}$$

In case (2) the contribution to the sum due to such an $\underline{E}'$ having $e_{t-\Delta}=0$ is

$$\begin{aligned} P(\text{error}/\underline{E}')P(\underline{E}') &= P(D(\underline{Z}(\underline{E}'))=1)P(\underline{E}') \\ &= D(\underline{Z}(\underline{E}'))P(\underline{E}') \end{aligned} \tag{5.18}$$

By the previously described method of loading the look-up table data structure:

$$P^*(\underline{Z}/e_{t-\Delta}=1) = \sum_{\substack{\text{all } \underline{E}' \text{ in SUBMAP} \\ (\text{with } e_{t-\Delta}=1) \text{ mapping} \\ \text{to this } \underline{Z}}} P(\underline{E}) \quad \text{and} \quad P^*(\underline{Z}/e_{t-\Delta}=0) = \sum_{\substack{\text{all } \underline{E}' \text{ in SUBMAP} \\ (\text{with } e_{t-\Delta}=0) \\ \text{mapping to this } \underline{Z}}} P(\underline{E}) \tag{5.19}$$

Therefore,

$$\hat{P} = \sum_{\text{all } \underline{Z}} (1-D(\underline{Z}))P^*(\underline{Z}/e_{t-\Delta}=1) + \sum_{\text{all } \underline{Z}} D(\underline{Z})P^*(\underline{Z}/e_{t-\Delta}=0) \quad (5.20)$$

and $\hat{P}$ can be easily calculated from the three level SUBMAP data structure entries.

Now consider P_{skip}. This is merely the probability that an error vector $\underline{E}'$ occurs which has not been considered in creating the SUBMAP. Such an error vector must have weight greater than MAX, the maximum weight of error vectors considered in the SUBMAP. For notational convenience let LNER = LEN+1 = the number of bits in the overall error vector $\underline{E}'$ (including the bit to be estimated). Also note that the probability of having present an error vector weight w of length LNER is

$$P_w = \binom{\text{LNER}}{w} p^w (1-p)^{\text{LNER}-w} \quad (5.21)$$

Since all $\underline{E}'$ vectors having more than MAX ones have been excluded,

$$P_{skip} = \sum_{\substack{\text{all } \underline{E}' \\ \text{excluded} \\ \text{from SUBMAP}}} P(\underline{E}') = \sum_{i=\text{MAX}+1}^{\text{LNER}} P_i = 1 - \sum_{i=0}^{\text{MAX}} P_i \quad (5.22)$$

Thus it is a straightforward matter to compute both (5.20) and (5.22) to yield an upper bound on the actual P_e using the quantities

already available from the SUBMAP look-up table computation.

This bound will be quite tight especially at higher SNR levels where p is small thus reducing the magnitude of P_{skip}. Figure 5.4 shows the tightness of this upper bound as MAX is increased from 1 to 5. For the K=5, LSYN=6 decoder case shown, one notices that the bound is looser at low SNR and tighter at high SNR. Also for $MAX \geq d_{min} = 4$ there is little difference (≤ 0.1 db) between the theoretical ROMMAP performance and the upper bound on SUBMAP performance. However since the SUBMAP creation takes significantly less computations than the ROMMAP approach, it represents a much more feasible way to gauge performance. With the proper choice of a MAX weight considering the available minimum distance d_{min} of the code, one can closely approach ROMMAP performance with a SUBMAP decoder. The predicted performance shown by the bounds in Figure 5.4 will be confirmed by simulation in Section 5.5.

5.2.3 Performance Comparison with Conventional Feedback

The SUBMAP look-up table is created by considering a lightweight subset of E'vectors specified by MAX. Included in this subset is at least one minimum weight E'vector mapping to each syndrome Z. Such a single E'vector is commonly used to make the look-up table for conventional feedback decoders (see Heller [16]). Thus, for comparison purposes, as the three level data structure is loaded to generate the SUBMAP table, one can also generate a fourth level

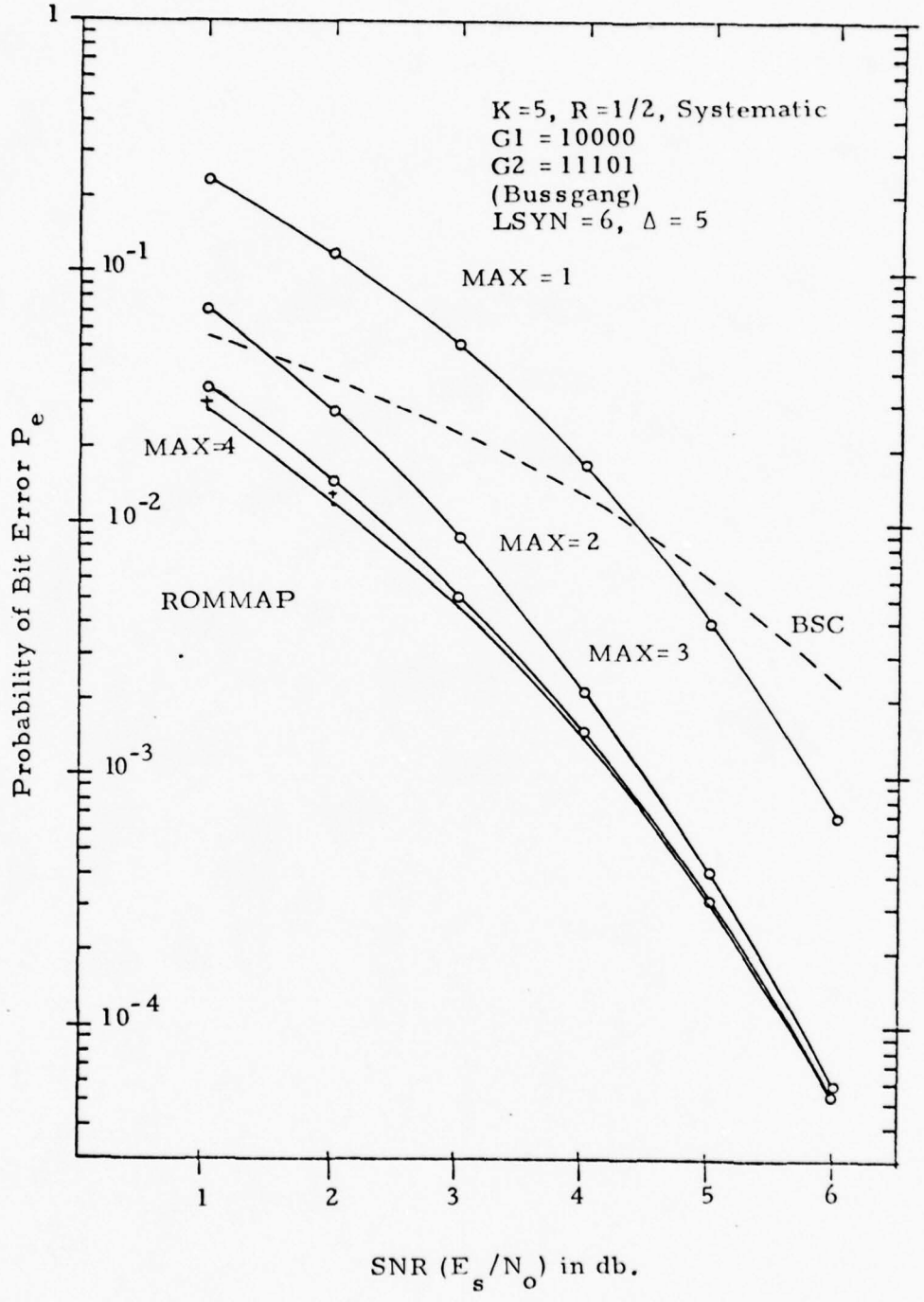


Figure 5.4 Upper Bound on SUBMAP Performance

containing a "minimum weight" look-up table obtained by associating with each $\underline{Z}$ the error bit $e_{t-\Delta}^{(1)}$ found in a minimum weight $\underline{E}'$ mapping to this $\underline{Z}$. This was done in the creation of look-up tables for the code and LSYN combinations simulated in Section 5.5 to enable a side by side comparison of actual SUBMAP versus actual conventional feedback performance.

The same upper bounding arguments described in Section 5.2.2 for the SUBMAP table can be applied to the minimum weight error vector table (conventional feedback) to give an upper bound on the performance of a decoder using the conventional feedback table without feedback. Figure 5.5 shows the predicted performance bounds for both the SUBMAP and conventional minimum weight tables for LSYN = 13 with a K = 11 code at a constant SNR level of 2 db (E_s/N_0). As expected, the SUBMAP upper bound remains lower than the minimum weight (without feedback) upper bound as MAX is increased to consider a larger subset of lightweight error vectors.

Both predicted SUBMAP and minimum weight bounds do not account for the potential effect of feeding back error estimates to correct syndromes. These predicted performance bounds give an upper limit on error probability when the results of prior decisions are not considered. One thus is led to hope that employing feedback may improve SUBMAP performance as it is known to improve minimum weight performance. The extent to which this is

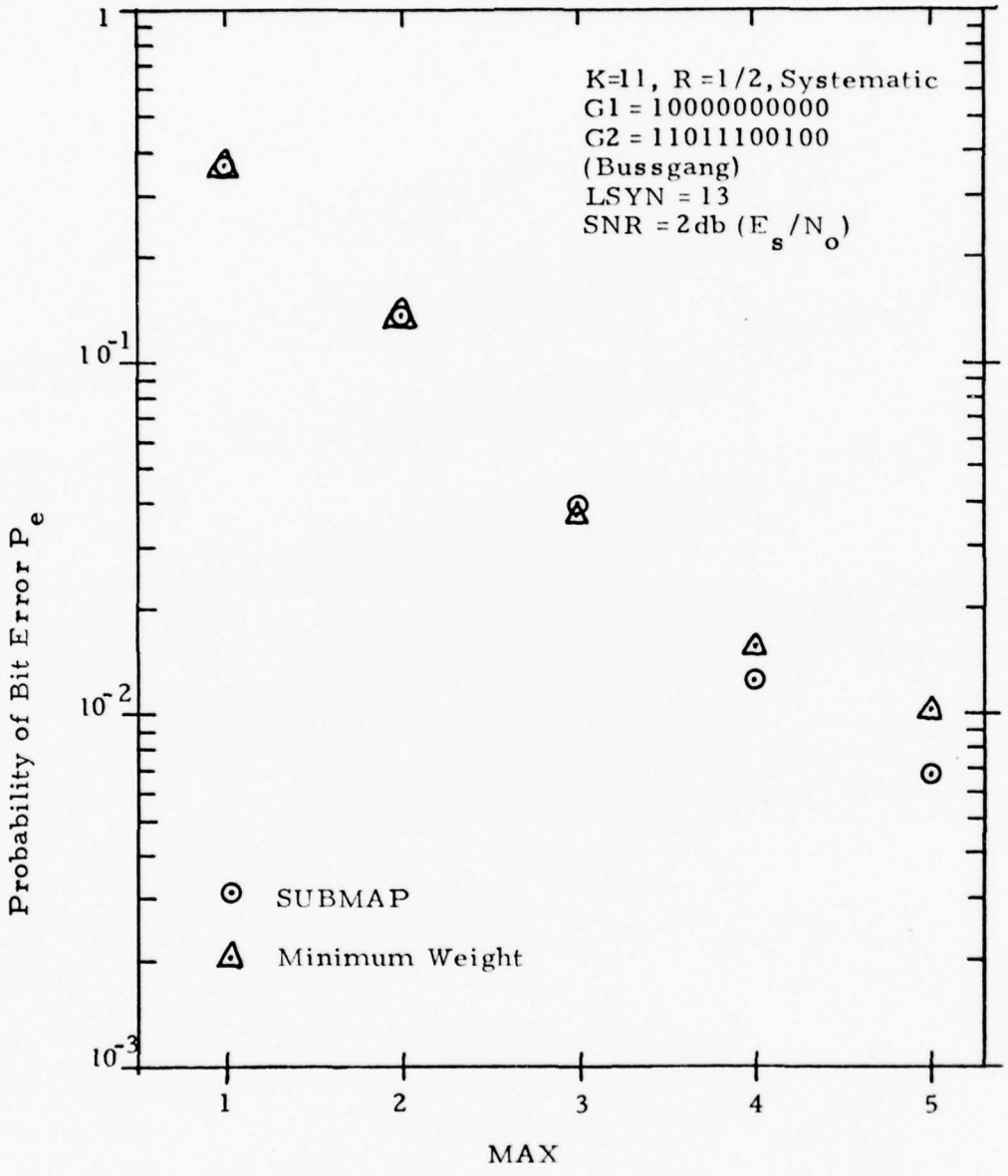


Figure 5.5 SUBMAP and Minimum Weight Performance as a Function of MAX

true is shown in Section 5.5 simulation.

5.3 Performance Improvement with Increasing Syndrome Length

In order to demonstrate the beneficial effect of using syndrome lengths greater than one constraint length, predicted SUBMAP performance for codes of constraint length K = 5, 7, and 11 were computed for several LSYN values. These predictions are given in Figures 5.6, 5.7, and 5.8 respectively and show how performance should improve as the number of syndrome bits used is increased.

From these predictions one can also observe that, for a given LSYN, substituting a code with a longer constraint length and minimum distance for a shorter constraint length code should provide better performance. LSYN is the parameter that determines the size of the SUBMAP look-up table and thus effectively the complexity of the decoder. Thus in a practical design one is led to allow the permitted level of complexity (look-up table size) determine a corresponding LSYN value and use a code having a constraint length as near to this LSYN as possible.

Specifically, using a longer constraint length code (e.g. K=11) provides better performance than using a shorter constraint length code (e.g. K=5) for the same number of syndrome bits (e.g. LSYN=11). This is to be expected due to the greater minimum distance (d_{min}=7) of the K=11 code versus d_{min}=4 for the K=5 code.

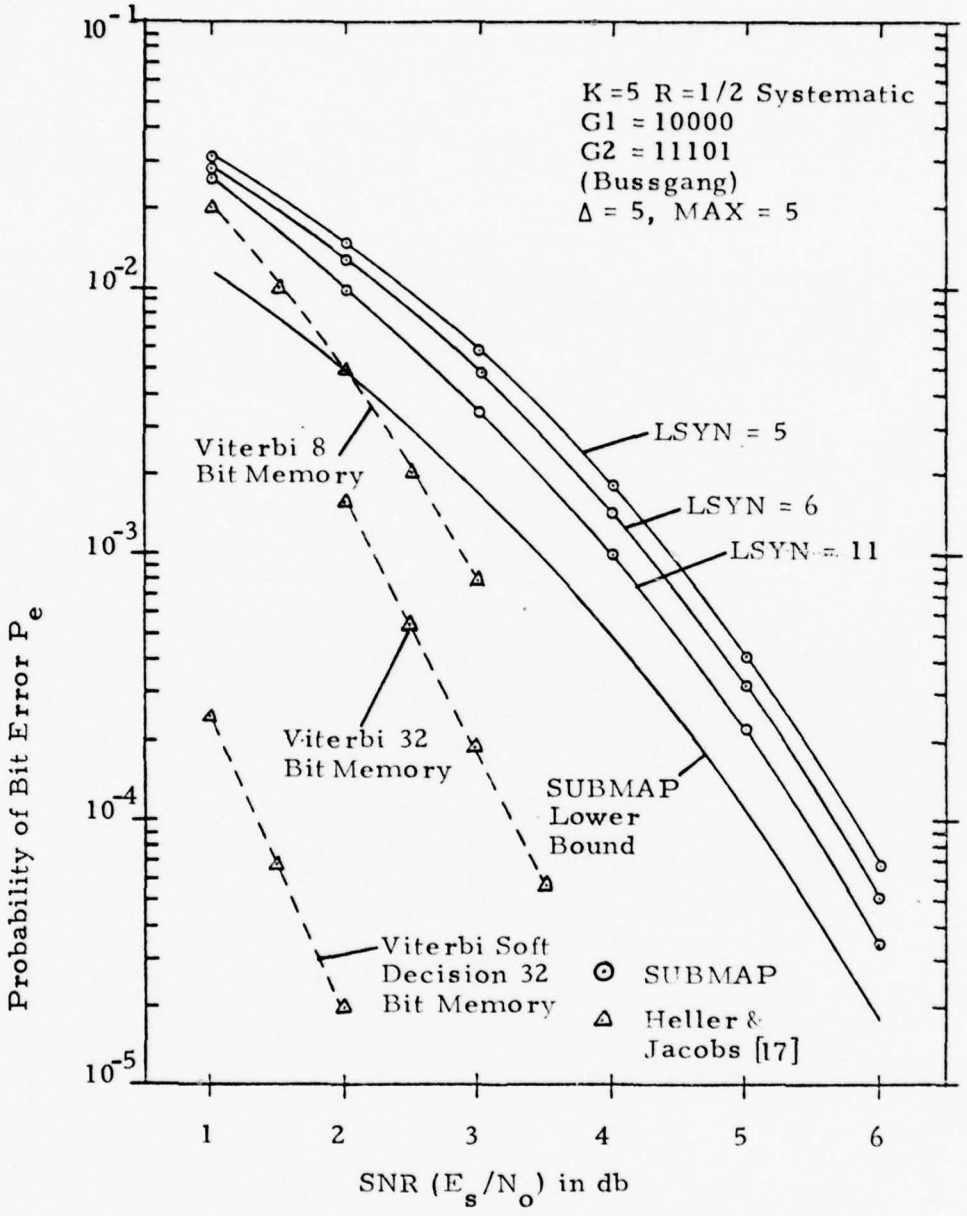


Figure 5.6 SUBMAP Predicted Performance, K = 5

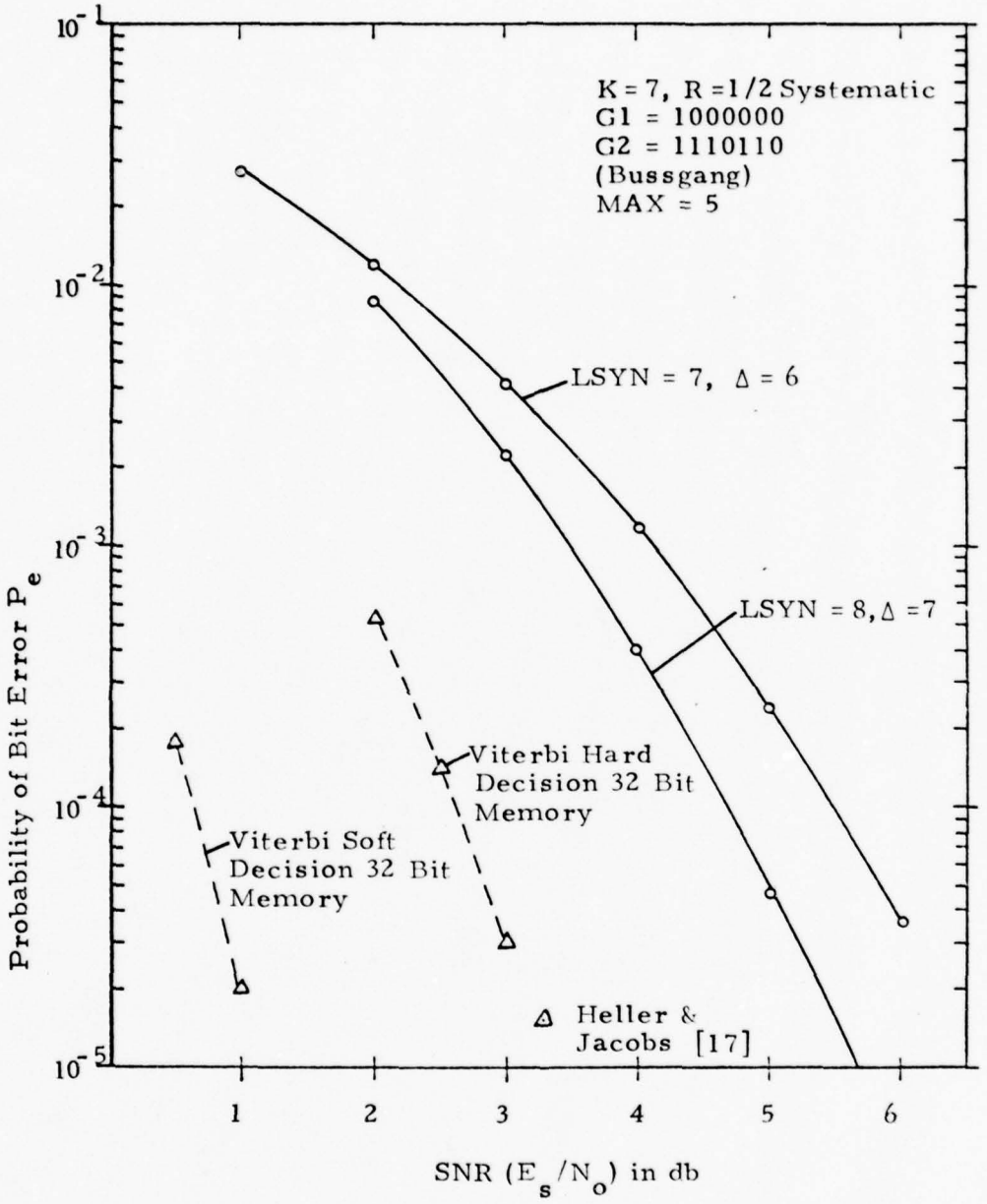


Figure 5.7 SUBMAP Predicted Performance, K=7

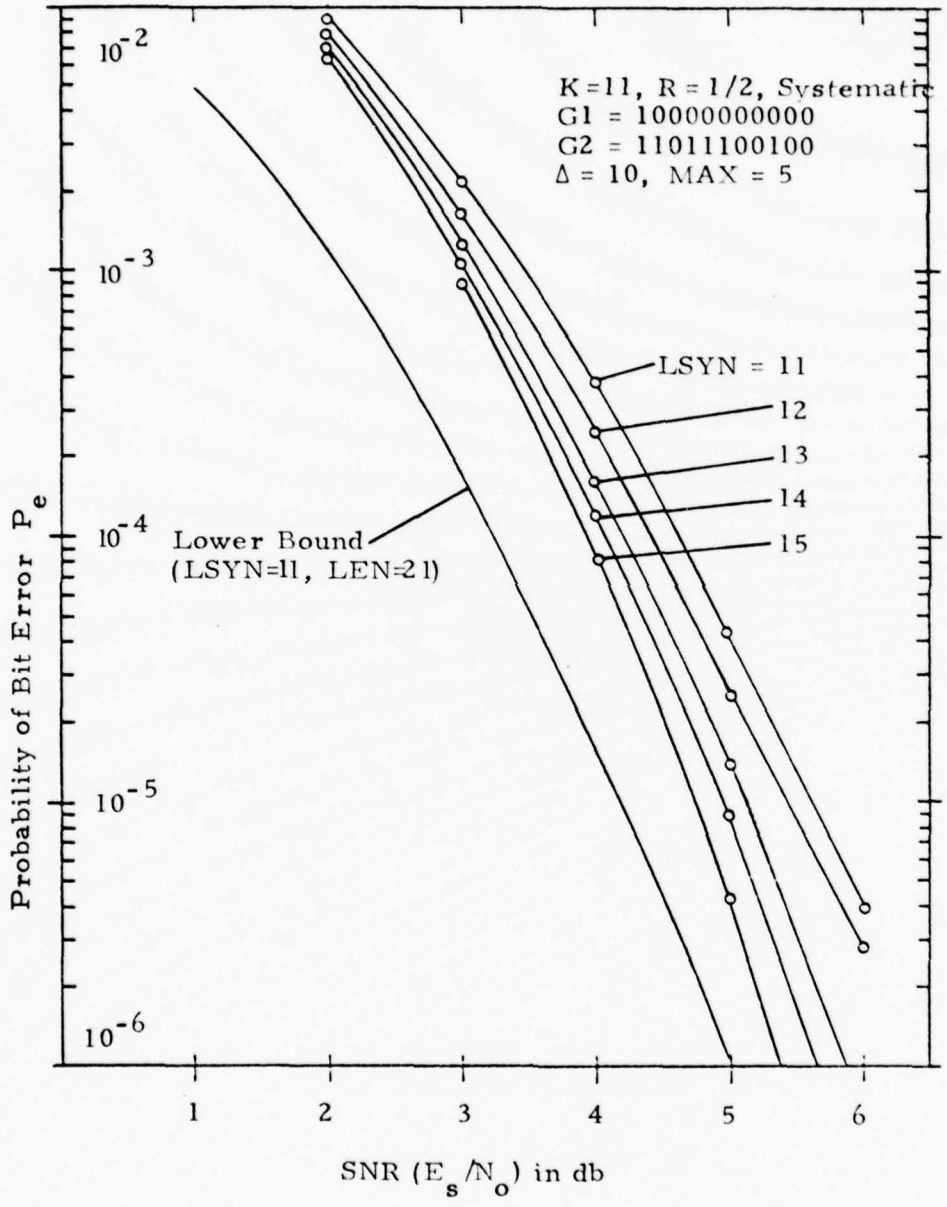


Figure 5.8 SUBMAP Predicted Performance, K=11

Also plotted in Figures 5.6 and 5.7 are extracted Viterbi decoding results as published by Heller and Jacobs [17] for K = 5 and K =7 respectively. The side-by-side comparison of Viterbi decoding with SUBMAP decoding indicates the greater range of coding gain achievable by the more complex Viterbi approach. This point will be touched on again in Section 5.4 where a loose lower bound on SUBMAP performance (for a fixed decoding delay Δ) will be seen to lie above the published Viterbi performance.

5.4 A Loose Lower Bound on Probability of Error

The previous sections have shown that for a given code and delay Δ one can achieve improved SUBMAP performance by using longer syndrome vectors as shown in Figures 5.6 through 5.8. It would be useful, when comparing the suboptimal SUBMAP decoder performance with other decoder approaches to have some lower bound on SUBMAP probability of error. This would enable one to state a performance improvement limit which even continued effort at increasing the SUBMAP look-up table size would not achieve. A computational method of specifying such a lower bound for a given delay and code can be obtained by examining the syndrome equation set. If one hypothesizes that no prior incorrect error estimates have been made and that all prior estimates have been fedback, one can examine the resultant first error probability to obtain a bound.

As an illustration, consider the previously discussed K=5, rate $\frac{1}{2}$ systematic code for which the first six syndrome equations are:

$$
\begin{aligned}
Z_{t-5} &= e^{(2)}_{t-5} + e^{(1)}_{t-5} + e^{(1)}_{t-6} + e^{(1)}_{t-7} + 0 + e^{(1)}_{t-9} \\
Z_{t-4} &= e^{(2)}_{t-4} + e^{(1)}_{t-4} \quad e^{(1)}_{t-5} + e^{(1)}_{t-6} + 0 + e^{(1)}_{t-8} \\
Z_{t-3} &= e^{(2)}_{t-3} + e^{(1)}_{t-3} + e^{(1)}_{t-4} + e^{(1)}_{t-5} + 0 + e^{(1)}_{t-7} \\
Z_{t-2} &= e^{(2)}_{t-2} + e^{(1)}_{t-2} + e^{(1)}_{t-3} + e^{(1)}_{t-4} + 0 + e^{(1)}_{t-6} \\
Z_{t-1} &= e^{(2)}_{t-1} + e^{(1)}_{t-1} + e^{(1)}_{t-2} + e^{(1)}_{t-3} + 0 + e^{(1)}_{t-5} \\
Z_{t} &= e^{(2)}_{t} + e^{(1)}_{t} + e^{(1)}_{t-1} + e^{(1)}_{t-2} + 0 + e^{(1)}_{t-4}
\end{aligned}
\qquad (5.23)
$$

Here with $\Delta = 5$ the decoder is now to estimate $e^{(1)}_{t-5}$.

If one now considers the conditional probability of the decoder making a mistake in estimating $e^{(1)}_{t-5}$ conditioned on all prior estimates being both correct and correctly fedback, the following set of modified syndrome equations apply.

$$
\begin{aligned}
Z'_{t-5} &= e^{(2)}_{t-5} + e^{(1)}_{t-5} \\
Z'_{t-4} &= e^{(2)}_{t-4} + e^{(1)}_{t-4} + e^{(1)}_{t-5} \\
Z'_{t-3} &= e^{(2)}_{t-3} + e^{(1)}_{t-3} + e^{(1)}_{t-4} + e^{(1)}_{t-5} \\
Z'_{t-2} &= e^{(2)}_{t-2} + e^{(1)}_{t-2} + e^{(1)}_{t-3} + e^{(1)}_{t-4} + 0 \\
Z_{t-1} &= e^{(2)}_{t-1} + e^{(1)}_{t-1} + e^{(1)}_{t-2} + e^{(1)}_{t-3} + 0 + e^{(1)}_{t-5} \\
Z_{t} &= e^{(2)}_{t} + e^{(1)}_{t} + e^{(1)}_{t-1} + e^{(1)}_{t-2} + 0 + e^{(1)}_{t-4}
\end{aligned}
\qquad (5.24)
$$

Here the upper right triangle of error bits in (5,23) has been removed and the associated syndrome bits modified to corrected values to reflect the error bit removal through feedback.

Faced with the modified set of syndrome equations for this idealized conditional case, one can compute a ROMMAP or SUBMAP look-up table as previously discussed. Then the conditional probability of incorrectly estimating $e_{t-5}^{(1)}$ can be computed exactly in the ROMMAP case or bounded in the SUBMAP case as explained in Sections 5.1.3 and 5.2.2 respectively. In the ROMMAP case this idealized probability of error will be a lower bound on ROMMAP performance *no matter how many additional syndrome bits might be* used. In the SUBMAP case the idealized probability will be a tight upper bound on the P_e and being a tight upper bound for such an idealized case will be useful as a loose lower bound to actual performance.

This SUBMAP lower bound has been computed for the K=5 (Δ=5) and K=11 (Δ=10) decoders and also plotted on Figure 5.6 and 5.8 respectively. Since the assumption of no previous incorrect error estimates is quite idealized, this first error probability lower bound is relatively loose. Indeed as shown in later simulation results for the K=5 code in Figure 5.9, continued increases in syndrome length with a fixed delay causes performance to tend

toward an improvement limit ≤1 db above the loose lower bound.

5.5 Simulation Results

In this section performance curves from Monte Carlo simulation for a number of SUBMAP decoder implementations are presented. Following some preliminary explanation regarding simulation methods, actual SUBMAP performance results for two systematic codes listed by Bussgang [19] for constraint lengths of K=5 and K=11 are given. Although primary emphasis is placed on SUBMAP decoder performance, some cases were simulated in which the conventional feedback decoding approach (using minimum weight error vectors) as well as the SUBMAP decoder were run with the same error inputs to enable a side-by-side comparison of the two methods. The performance curves show the improvement possible not only by using longer constraint length codes but also that by using more syndrome bits in the SUBMAP decoding of a particular code. Also displayed is the extent of performance improvement achievable with no complexity increase by merely substituting the SUBMAP table for the conventional feedback look-up tables.

5.5.1 Method of Simulation

The simulation described in the following section was done in FORTRAN on a Digital Equipment Corporation KL 10 time-sharing

system (in batch mode) located at the University of Southern California. All codes considered were optimum rate $\frac{1}{2}$ systematic codes given in Bussgang [19]. Two uniform pseudo-random number generators (PRN) were used interchangeably at critical points to check the quality of the error generation process. Appendix B gives additional details on these PRN generators. To insure the statistical validity of simulation results, error bit inputs to the two syndrome former input ports were:

(1) independently generated through separate calls to the PRN generator using seeds selected to guarantee no repetition during 10×10^6 calls to the PRN generator; and

(2) simulated for at least 10^6 error bits per syndrome former input except at higher SNR levels were up to 5×10^6 error bits per input were simulated. All P_e points plotted herein are based on at least 100 decoding errors.

The general operation of the simulation program consisted of reading in and storing the appropriate look-up table previously generated in accordance with Section 5.2. This look-up table would have been created as either a SUBMAP table or as a minimum weight table corresponding to the decoder type being simulated. The PRN generator then supplied two PRNs uniformly distributed between 0 and 1. These were used to create $e_i^{(1)}$ and $e_i^{(2)}$ error bit inputs

based on the BSC crossover probability given by the symbol SNR (E_s/N_0) as specified in Section 5.1.2. The syndrome former then computed the syndrome bit Z_i for input to the syndrome vector while error bits were shifted along each clock period in an $\underline{E}$ vector. These two error bits were actually input to two parallel decoders, one decoding based strictly on the current $\underline{Z}$ with no feedback of error estimates and a second using feedback of error decisions to update the syndrome vector. Thus actual performance for a particular look-up table could be simulated in one pass both with and without feedback. The actual P_e for both decoders was computed by counting the number of times during the simulation that the estimated error $\hat{e}_{i-\Delta}^{(1)}$ did not equal the actual error bit $e_{i-\Delta}^{(1)}$ and dividing by the number of error bits processed during the simulation through one input to the syndrome former.

Since the operation of the syndrome former is independent of the actual data that might have been input to the encoder, the above simulation effectively assumes all zero data and thus only error inputs entering the syndrome former.

The PRN generators used were such that, given a starting "seed" number, the generator would output the identical stream of PRNs. Thus it was possible to run simulations of various SUBMAP and conventional feedback schemes using the identical input error bit streams. This gives a more reasonable comparison of techniques

than non-identical input streams.

5.5.2 Decoder Performance for K=5 Code

The decoding of the K=5 binary systematic rate $\frac{1}{2}$ code (G1 = 10000, G2 = 11101) used in previous performance prediction discussions was simulated for both the SUBMAP and the conventional feedback approaches. As described in Section 5.5.1, the performance of each decoder table was simulated both with and without feedback. The decoding delay was fixed at $\Delta = 5$ (i.e. estimate $e_{i-5}^{(1)}$) with MAX = 5 in the table creation step. The design SNR was always 2 db (E_s/N_0).

Figure 5.9 shows simulation results for the K=5 code SUBMAP decoder with and without feedback for LSYN = 6 and LSYN = 11. Also simulated but not shown was LSYN = 15. Figure 5.9 also gives computed upper bounds and the loose lower bound for Δ=5. It can be seen that the simulated performance falls extremely close to the predicted tight upper bounds and well above the loose lower bound. The futility of increasing LSYN much beyond two constraint lengths is actually indicated by what is not plotted since both predicted performance and simulation results for LSYN = 15 fell so close to LSYN = 11 bound and results so as to make plotting them infeasible.

Also evident is the minimal improvement achievable by using feedback with the SUBMAP decoder. Indeed using feedback in the LSYN = 11 case produced somewhat poorer simulated performance.

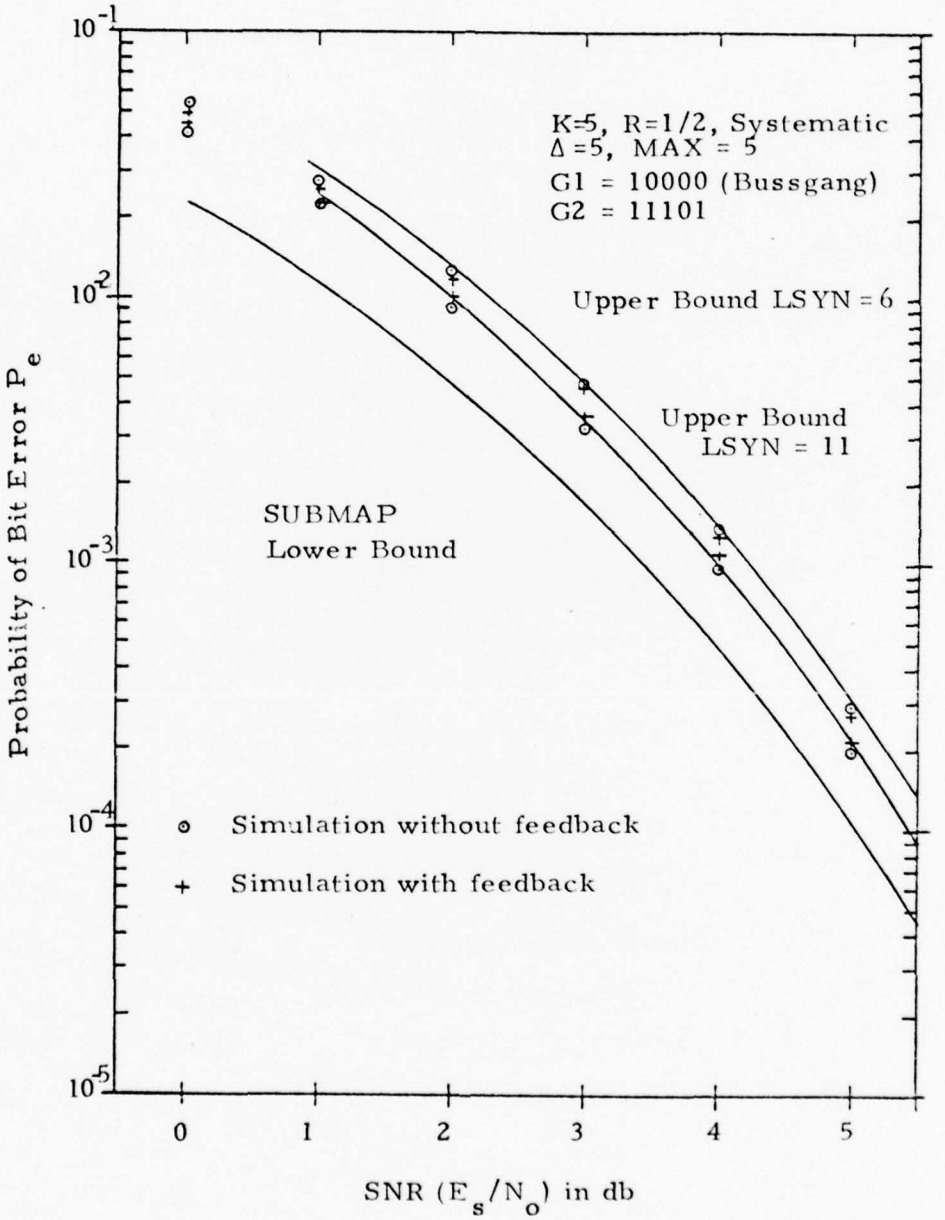


Figure 5.9 SUBMAP Simulation, K = 5

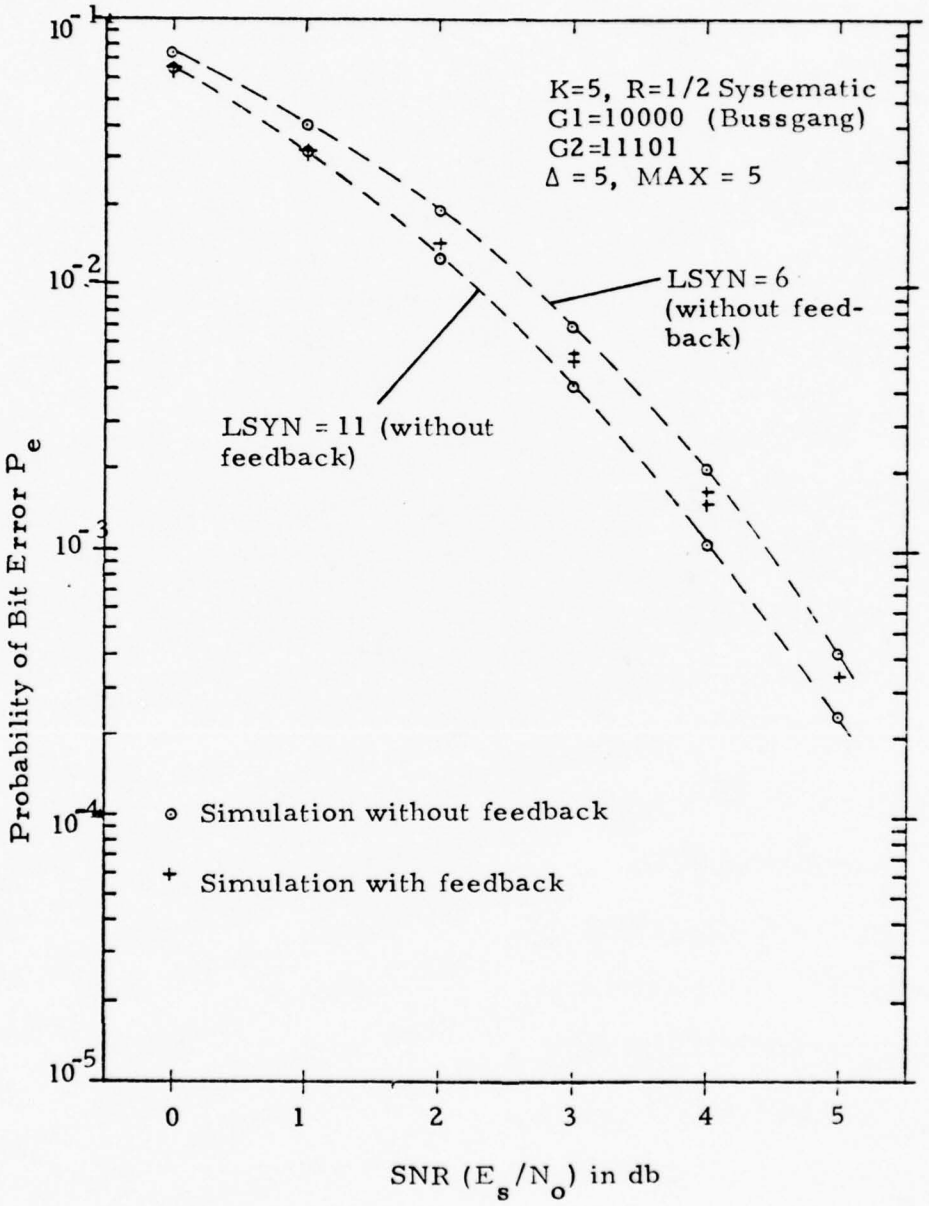


Figure 5.10 Minimum Weight Simulation, K=5

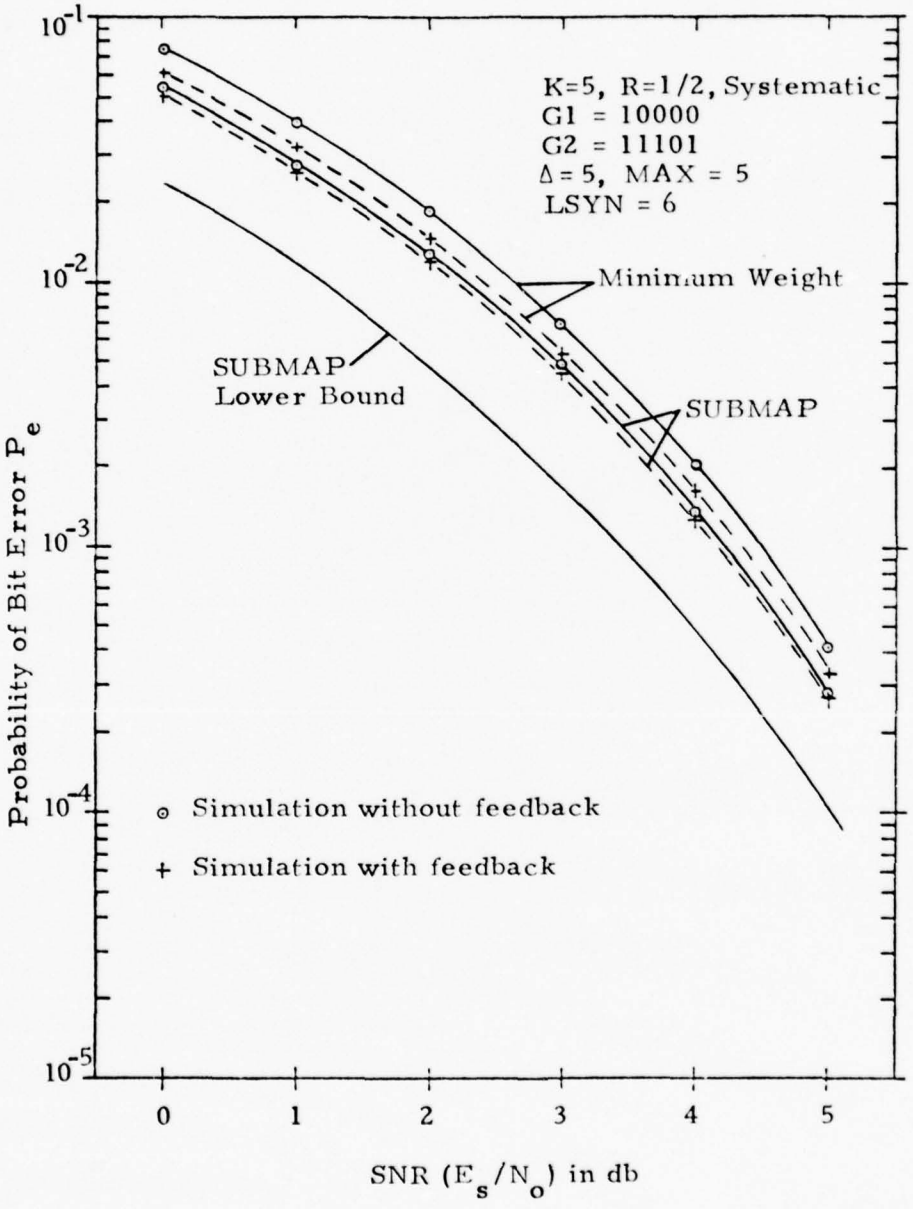


Figure 5.11 Comparison of SUBMAP and Conventional Feedback, K = 5

Figure 5.10 shows simulation results for a minimum weight look-up table decoder with and without feedback for the same K=5 code for LSYN=6 and LSYN=11. The improvement due to feedback is seen to be more substantial for the minimum weight case than for the SUBMAP case at LSYN=6. However at LSYN = 11 feedback appears to be counterproductive since feedback results for the minimum weight approach practically coincide for both LSYN = 6 and LSYN =11.

Figure 5.11 combines selected SUBMAP and conventional feedback results on one graph to show the relative superiority of the SUBMAP approach over the minimum weight approach. A careful comparison indicates that the SUBMAP approach even without feedback slightly outperforms conventional feedback decoding. However the absolute performance improvement for the short (K=5) constraint length code and LSYN combinations considered was roughly 0.2 dB at lower SNR levels.

In summary, although these simulation results confirm the accuracy of the predicted SUBMAP performance bounds, they reveal only a minimal improvement over conventional feedback decoder performance. Therefore the following section deals with SUBMAP performance for a longer constraint length (K=11) code.

5.5.3 Decoder Performance for K=11 Code

The SUBMAP decoding of the K=11 systematic rate $\frac{1}{2}$ code specified by

$$G1 = 10000000000$$

$$G2 = 11011100100 \qquad (5.25)$$

was simulated both with and without feedback for LSYN = 11 and LSYN = 13. A design SNR of 2 db (E_s/N_0) was used with MAX=5 for look-up table creation. The decoding delay was fixed at $\Delta = 10$ (i.e. estimate $e_{i=10}^{(1)}$). Also simulated was decoding using a minimum weight approach as used in conventional feedback.

Figure 5.12 shows SUBMAP simulation results for the K=11 code for LSYN = 11 and LSYN = 13. Evident is a slight performance improvement obtained by using feedback with the SUBMAP table at both LSYN values. Shown as solid lines are the predicted upper bounds and the loose lower bound. Again the accuracy of the upper bounds is confirmed.

Figure 5.13 shows simulation results using a minimum weight look-up table to demonstrate conventional feedback performance for the K = 11 code at LSYN = 11 and 13. Shown as solid lines are the predicted upper bounds and the loose lower bound. Evident is the performance improvement obtained by using feedback.

Figure 5.14 combines SUBMAP and conventional feedback results for LSYN = 11 on one graph to show the relative superiority of the SUBMAP approach. Thus is evident the performance improvement possible with no decoding complexity increase by

merely substituting a SUBMAP look-up table for the minimum weight table in a conventional feedback decoder.

The statement was made in Section 5.3 that it is better to use a longer constraint length code rather than merely increasing LSYN for a shorter constraint length code. This fact is shown to be true in Figure 5.15 where simulation results for K=5 (LSYN = 11) and K = 11 (LSYN =11) have been compared. It is evident that the K=11 code is preferable if hardware complexity equivalent to a size $2^{LSYN} = 2^{11}$ is feasible. The improvement is to be expected since the minimum distance of the K=5 code was 4 while that of the K=11 code was 7. The additional error correction capability exhibits itself in the steeper slope to the K=11 curve of P_e.

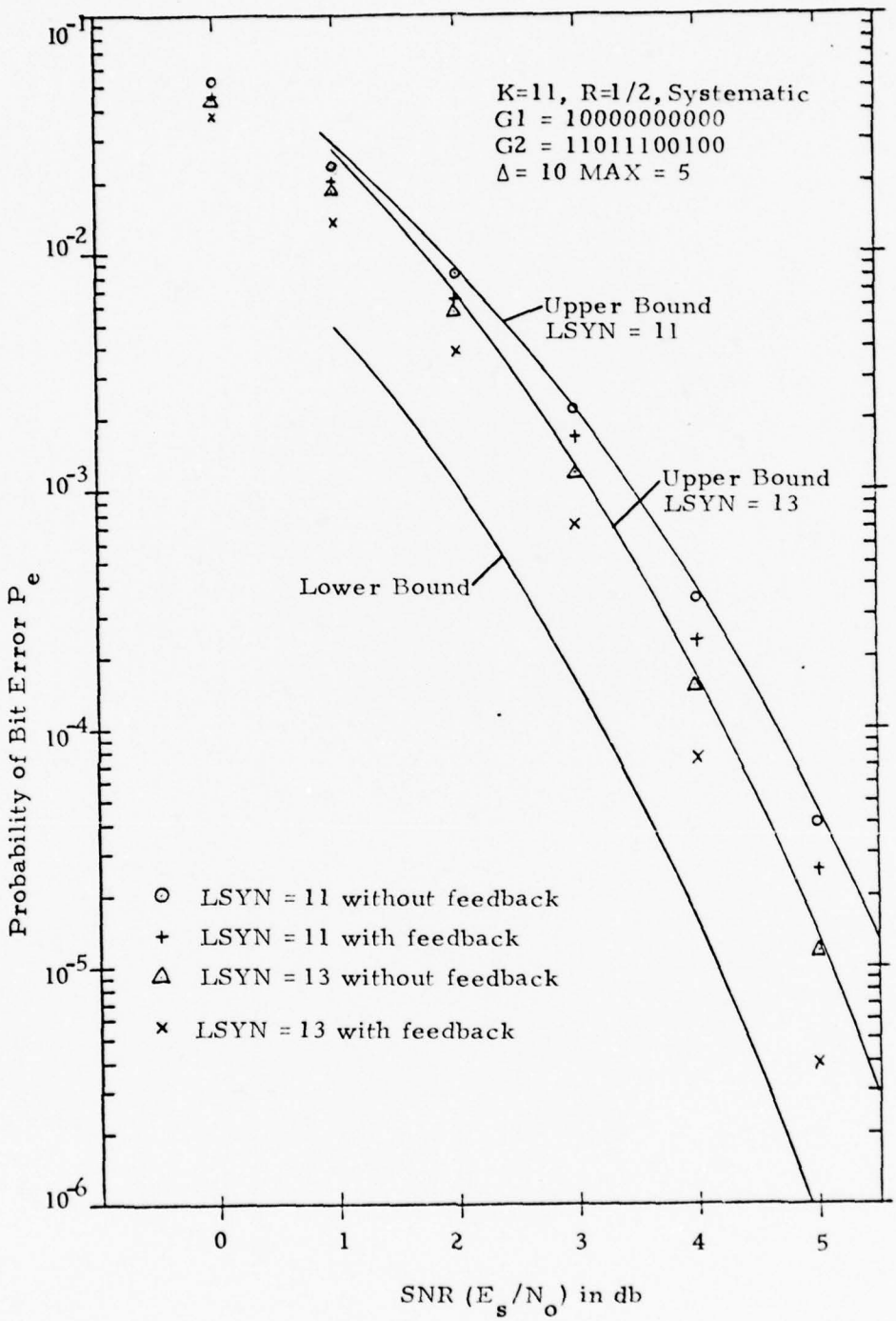


Figure 5.12 SUBMAP Simulation, K=11

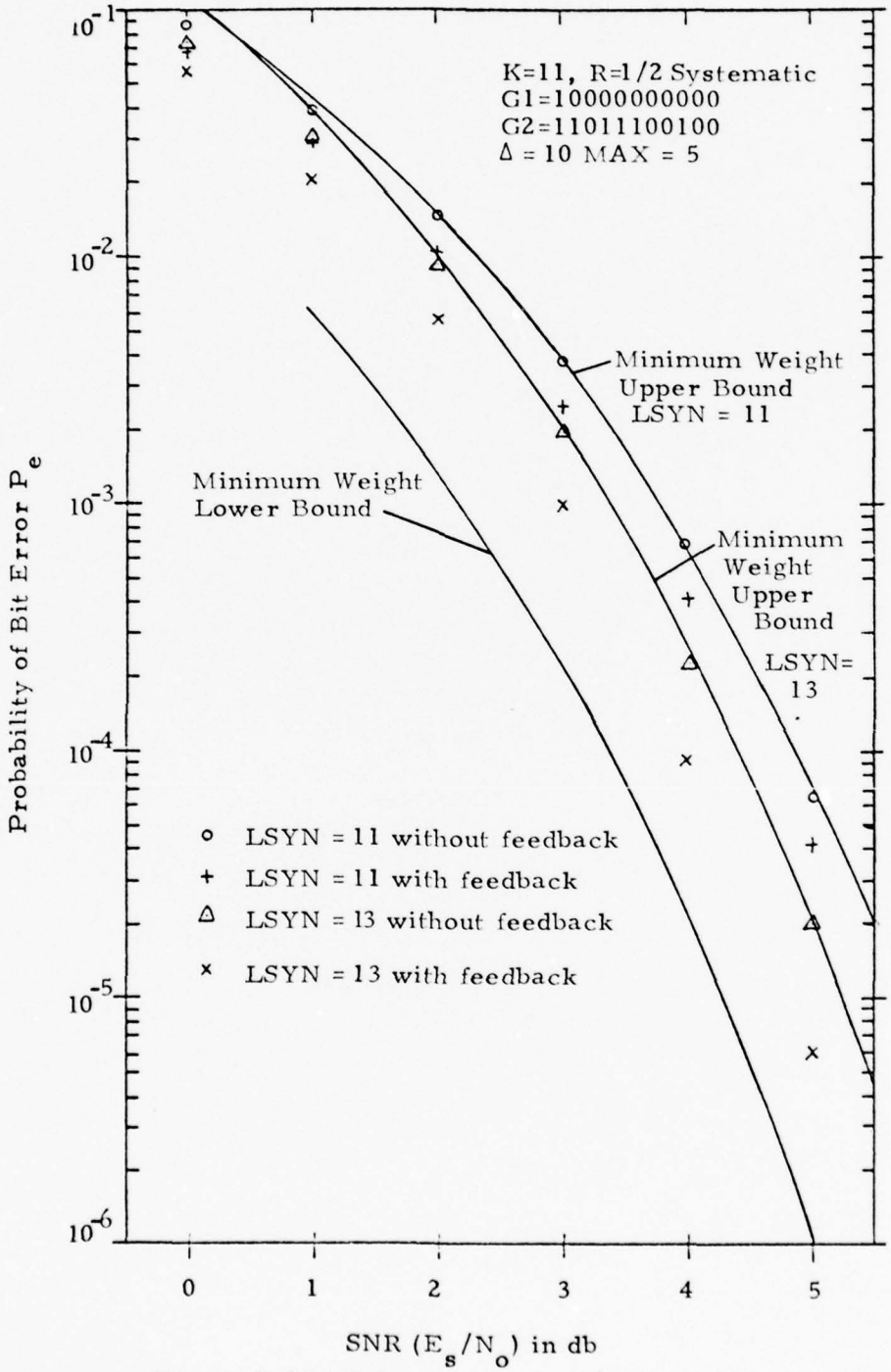


Figure 5.13 Minimum Weight Simulation, K = 11

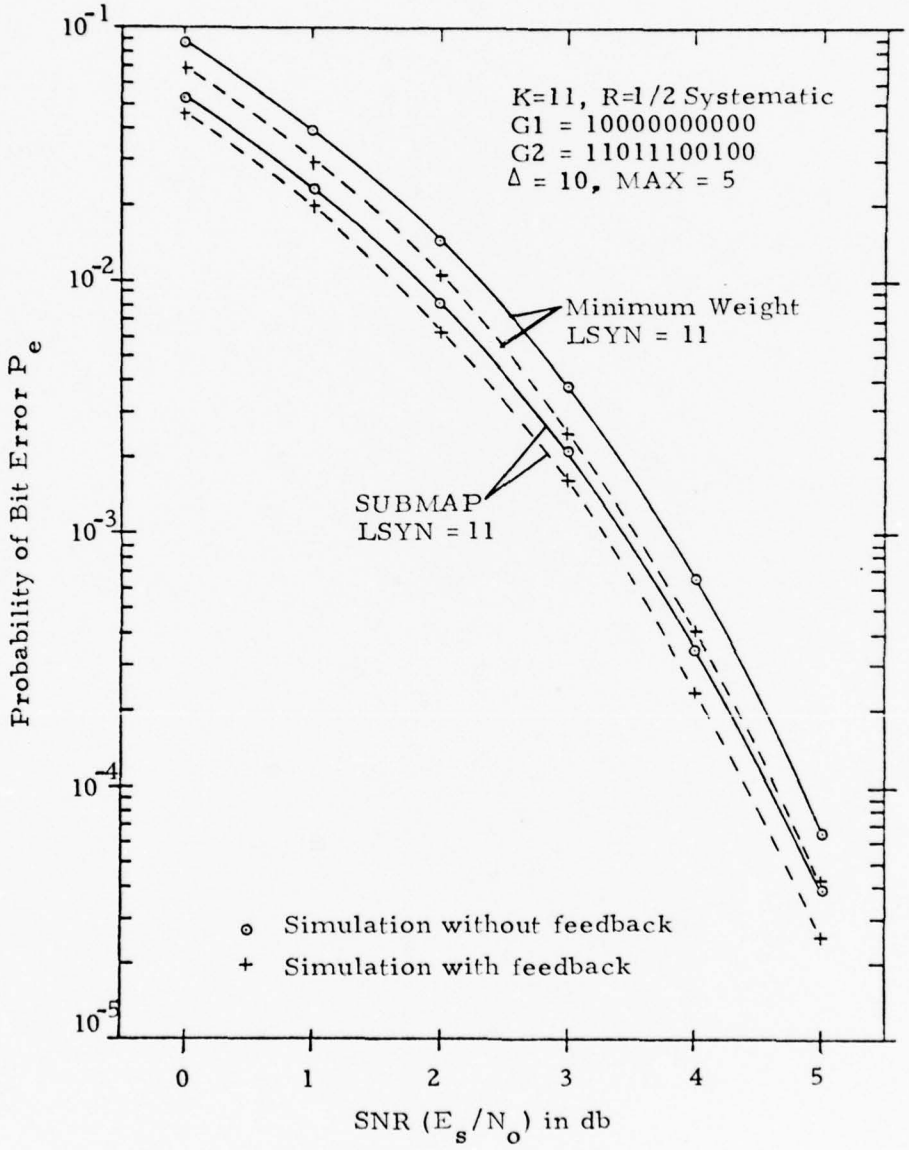


Figure 5.14 Comparison of SUBMAP and Conventional Feedback, K = 11

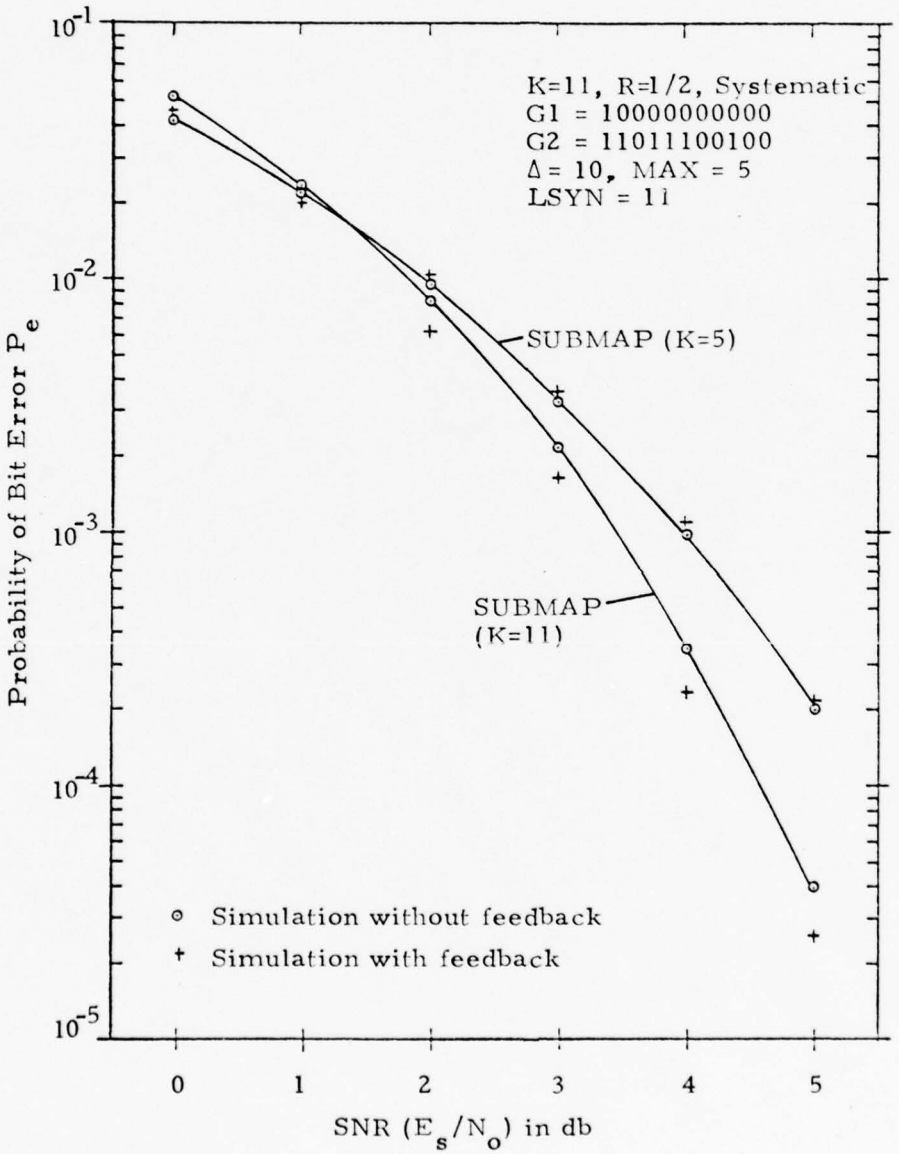


Figure 5.15 Effect of Constraint Length on SUBMAP Performance

Chapter 6

CONCLUSIONS AND RECOMMENDATIONS

This investigation developed in Chapter 3 several optimal MAP error bit estimation algorithms for decoding convolutional codes transmitted over a binary symmetric channel. These represent a set of error decoding approaches complementary to similar techniques previously used for data estimation. One advantage of such error bit estimation using syndromes is that the resulting performance is independent of a priori data probability. Thus an arbitrary actual data source output can be communicated reliably without having to preprocess the data prior to encoding just to insure equilikely input bits. This is in contrast to standard Viterbi maximum likelihood sequence decoding which requires equilikely input probabilities. Also in Chapter 3 corresponding optimal MAP error algorithms have been derived to handle the more realistic case of error decoding under a real time or finite delay restriction.

During the Chapter 4 investigation of implementable ROMMAP and SUBMAP suboptimal decoders using MAP error bit estimation with finite delay, it was seen that conventional feedback decoding is a rough approximation to MAP error bit estimation. The results of simulation discussed in Chapter 5 showed that a SUBMAP decoder based on MAP estimation outperforms conventional feedback decoding

for the same complexity. Noted also was the fact that feeding back error decisions in the SUBMAP case does provide additional improvement.

The complexity parameter for ROMMAP, SUBMAP, and conventional feedback is basically the look-up table size (2^{LSYN}). Thus, based on the simulation results of Chapter 5, for a given complexity one is inclined to favor use of a SUBMAP look-up table instead of a minimum weight look-up table as used in conventional feedback decoders. One other point worth mentioning in this connection is the comparative "weights" of the look-up tables, that is the number of 1's in the table entries. In all cases considered here, the SUBMAP table always had significantly less 1's than did the minimum weight table of the same 2^{LSYN} size. For example Table 6.1 shows comparative weights for the K = 11 systematic code at LSYN = 11 and 13. This investigation assumed and simulated read only memory (ROM) look-up tables in which the syndrome vector $\underline{Z}$ was converted to an address in a 2^{LSYN}-size table where each address location contains either a 1 or 0. However if an alternative device such as one based on an associative memory were used, the SUBMAP table implementation would be much simpler than that of the minimum weight table. This is due to the smaller number of 1-associated syndromes $\underline{Z}$ (i.e. the lighter-weight table) of the SUBMAP approach which would require a significantly smaller associative memory

LSYN	Look Up Table Size	Table Type	Design SNR (E_s/N_0 in db						
			0	1	2	3	4	5	6
11	2048	SUBMAP Table	117	144	149	151	152	152	152
		Min. Wt. Table	618	618	618	618	618	618	618
13	8192	SUBMAP Table	562	615	676	724	724	724	724
		Min. Wt. Table	2300	2300	2300	2300	2300	2300	2300

Table 6.1. Look-up Table Weights for K = 11 Code Example.

device. This appears to be an area worthy of some additional investigation in light of continuing advances in LSI capability.

The simulation results of Chapter 5 also demonstrated the accuracy of the tight upper bounds on performance derived in Section 5.2.2 thereby implying that these bounds represent a valuable design tool for predicting actual decoder performance. Section 5.3 confirmed the expected improvement in SUBMAP decoder performance with increasing syndrome vector length. This section also validated the rule of thumb that, with a complexity limit on syndrome vector length, the SUBMAP designer should select a code having as long a constraint length as possible within this syndrome length for best SUBMAP performance.

The practical hardware limits to increasing look-up table size is obviously a major limitation on continued SUBMAP performance improvement. Thus it is felt that further development of the techniques derived in Chapter 3 for optimal error bit estimation could lead to an implementation scheme that would bridge the gap between the performance of the look-up table approach of SUBMAP(and indeed of conventional feedback) and that of Viterbi maximum likelihood sequence decoding. Bridging this gap with such a MAP error bit estimation approach based on syndromes would enable decoder performance to be improved yet still maintain performance independent of a priori input data probability.

APPENDIX A

VITERBI DECODING WITH NONEQUILIKELY INPUT SEQUENCES

As stated by Viterbi [14],

> ...for completely general channels...<u>if all input data are equally likely</u>, the decoder which minimizes the error probability is one which compares the conditional probabilities, also called likelihood functions, $P(\underline{y}/\underline{x}^{(*)})$ where $\underline{y}$ is the overall received sequence and $\underline{x}^{*}$ is one of the possible transmitted sequences, and decides in favor of the maximum. This is called a maximum likelihood decoder. The likelihood functions are given or computed from the specifications of the channel. Generally it is more convenient to compare the quantities $\log P(\underline{y}/\underline{x}^{(*)})$, called log likelihood functions and the result is unaltered since the logarithm is a monotonic function of its (always positive) argument.

The important point for the reader to bear in mind is that Viterbi's excellent description of his now famous algorithm applies only to the case of <u>equally likely input sequences</u>.

To analyze the case where the input data sequences are <u>not</u> equilikely, a maximum a posteriori decoder should be considered. This is one which compares the conditional probabilities

$$P(\underline{x}^{(m)}/\underline{y}) = \frac{P(\underline{y}/\underline{x}^{(m)})P(\underline{x}^{(m)})}{P(\underline{y})} \qquad (A.1)$$

for candidate sequences $\{\underline{x}^{(m)}\}$ and decides in favor of the maximum.

Now consider the specific case of Section 3.2 where a rate 1/3 decoding problem has been converted to a rate 1/2 encoder and

channel with error bit sequence inputs. This provides a good example of what adaptation is necessary for a Viterbi algorithm for a nonequilikely input case. Thus Figure 3.3.b is the "error encoder" that will now be analyzed. Here define $P(e_t^{(i)} = 1) = p$ and $P(e_t^{(i)} = 0) = (1-p)$ since the binary symmetric channel still applies. In this case Eq. A.1 becomes

$$\underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ P(\underline{c}(\underline{E})/\underline{Z}) = \frac{P(\underline{Z}/\underline{c}(\underline{E})P(\underline{c}(\underline{E}))}{P(\underline{Z})} \right\} \qquad (A.2)$$

where to obtain A.2 the following substitutions have been made:

(1) $\underline{y}$ becomes $\underline{Z}$ = the vector of received syndrome bits;

(2) $\underline{x}^{(m)}$ becomes $\underline{c}(\underline{E})$ = one of the possible "transmitted" sequences; and

(3) $\underline{E} \triangleq (e_0, e_1, \ldots, e_N)$ = input error vector.

Note that each possible input error vector $\underline{E}$ corresponds to only one $\underline{c}(\underline{E})$ just as a data encoder output maps directly back to one data input sequence in the more conventional data encoding. Due to the bit-to-bit independence of all error bits, A.2 can be rewritten as:

$$\underset{\text{all } \underline{c}(\underline{E})}{\text{maximize over}} \left\{ P(\underline{c}(\underline{E})/\underline{Z}) = \prod_{i=0}^{N-1} P(Z_i/c_i(\underline{E}))P(c_i(\underline{E})) \right\}. \qquad (A.3)$$

The elements in Eq. A.3 are:

(1) $Z_i = \begin{bmatrix} Z_i^{(2)} \\ Z_i^{(3)} \end{bmatrix}$ = "received" syndrome bits at time i;

(2) $c_i(\underline{E}) = \begin{bmatrix} c_i^{(2)} \\ c_i^{(3)} \end{bmatrix}$ = "transmitted" bits at time i; and

(3) N = length of the input error sequence.

Taking logarithms yields:

$$\underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ \log P(\underline{c}(\underline{E})/\underline{Z}) = \sum_{i=0}^{N-1} \log(P(Z_i/c_i(\underline{E}))P(c_i(\underline{E}))) \right\} \qquad (A.4)$$

where the sum is over all branches in the error encoder trellis path corresponding to this particular $\underline{E}$ vector.

Now define $H(\underline{a}, \underline{b}) \triangleq$ Hamming distance between vectors $\underline{a}$ and $\underline{b}$. Also note that the $P(c_i(\underline{E}))$ is just the a priori probability of the input error bit $e_i^{(1)}$ for the $\underline{c}(\underline{E})$ in question (either p or 1-p since the "inputs" are characterized by the binary symmetric channel crossover probability and there is a one-to-one mapping from inputs to outputs). Hence,

$$\underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ \log P(\underline{c}(\underline{E})/\underline{Z}) = \sum_{i=0}^{N-1} \log(p^{H(Z_i, c_i(\underline{E}))} (1-p)^{n_0 - H(Z_i, c_i(\underline{E}))} P(e_i^{(1)})) \right\} \qquad (A.5)$$

$$= \underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ \sum_{i=0}^{N-1} [H(Z_i, c_i(\underline{E}))\log p + (n_0 - H(Z_i, c(\underline{E}))\log(1-p) + \log P(e_i^{(1)})] \right\} \qquad (A.5)$$

where n_0 = the number of bits in Z_i and $c_i(\underline{E})$ ($n_0 = 2$ in this specific case).

Rearranging yields:

$$\underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ \sum_{i=0}^{N-1} \left[-H(Z_i, c_i(\underline{E})) \log \frac{1-p}{p} + \overbrace{n_0 \log(1-p)}^{\text{constant}} + \log P(e_i) \right] \right\} . \tag{A.6}$$

It should be noted that the third term of the summation will equal log(p) or log(1-p) depending on the value of $e_i^{(1)}$ specified by the error input vector $\underline{E}$ being considered. Also, if $P(e_i^{(1)} = 1) = P(e_i^{(1)} = 0) = \frac{1}{2}$ (as would be true only if the error inputs were equilikely), Eq. A.6 would reduce to the conventional Viterbi decoding rule:

$$\underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ \sum_{i=0}^{N-1} \left[-H(Z_i, c_i(\underline{E}))K_1 + K_2 \right] \right\} \tag{A.7}$$

with $K_1 = \log\left(\frac{1-p}{p}\right)$ and $K_2 = n_0 \log(1-p) + \log(\frac{1}{2})$.

This would direct the decoder to select the path $\underline{c}(\underline{E})$ minimizing the Hamming distance between $\underline{Z}$ and $\underline{c}(\underline{E})$.

On the contrary, for the nonequilikely input case under consideration here, $P(e_i^{(1)} = 1) \neq P(e_i^{(1)} = 0)$. Thus $\log P(e_i^{(1)})$ must be included in all computations and a modified metric must be used. Generally A.6 requires that for a maximum, the trellis path would be selected which both:

(1) minimizes $H(\underline{Z}, \underline{c}(\underline{E}))$ weighted by $\log\left(\frac{1-p}{p}\right)$; and

(2) maximizes $\log P(\underline{E}) = \sum_{i=0}^{N-1} \log P(e_i^{(1)})$.

The quantity $\log P(e_i^{(1)})$ is either $\log(p)$ or $\log(1-p)$. Since $0 \leq p \leq 1$ and $0 \leq 1-p \leq 1$, this logarithm is nonpositive. The least negative logarithm corresponds to $e_i^{(1)} = 0$. Thus the maximization in A.6 implies that in selecting $\hat{\underline{E}}$ one should "lean" toward an input sequence with minimum weight whose output sequence $\underline{c}(\hat{\underline{E}})$ is closest to the actual received syndrome sequence $\underline{Z}$.

With the definition $H(\underline{E}, 0) \triangleq$ Hamming distance between a candidate $\underline{E}$ and the all-zeros vector, A.6 can be expressed entirely in terms of Hamming distances. Neglecting constant terms, the A.6 rule can be written in an equivalent form as:

$$\underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ \log \left(\frac{p^{H(\underline{Z}, \underline{c}(\underline{E}))} p^{H(\underline{E}, \underline{0})}}{(1-p)^{H(\underline{Z}, \underline{c}(\underline{E}))} (1-p)^{H(\underline{E}, \underline{0})}} \right) \right\} \tag{A.7}$$

$$\Rightarrow \underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ \log \left[\left(\frac{1-p}{p} \right)^{-[H(\underline{Z}, \underline{c}(\underline{E})) + H(\underline{E}, \underline{0})]} \right] \right\}$$

$$\Rightarrow \underset{\text{over all } \underline{c}(\underline{E})}{\text{maximize}} \left\{ -[H(Z, c(E)) + H(E, 0)] \underbrace{\log \left(\frac{1-p}{p} \right)}_{\text{constant}} \right\}$$

$$\Rightarrow \underset{\text{over all } \underline{c}(\underline{E})}{\text{minimize}} [H(\underline{Z}, \underline{c}(\underline{E})) + H(\underline{E}, \underline{0})] \quad \text{Optimum decoding rule for nonequilikely inputs.} \tag{A.8}$$

Thus the decoding rule for error sequences in this case requires the selection of $\hat{\underline{E}}$ as that input error sequence which minimizes the Hamming distance sum given by A.8.

Since the Hamming distance $H(\underline{E}, \underline{0})$ is the same as the Hamming weight of the error vector $\underline{E}$, the decoding rule for nonequilikely input sequences can also be written as:

$$\underset{\text{over all } \underline{c}(\underline{E})}{\text{minimize}} \left\{ [H(\underline{Z}, \underline{c}(\underline{E})) + W(\underline{E})] \right\} \qquad \text{(A.9)}$$

where $W(\underline{E})$ is the weight of the error vector $\underline{E}$.

APPENDIX B

COMPUTER PROGRAMMING CONSIDERATIONS

The ROMMAP and SUBMAP creation programs for the K = 5 and K = 7 codes discussed in Section 5.1 and 5.2 were initially programmed on an IBM 360/44 system in FORTRAN IV. The necessary conversion from SNR (E_s/N_o) to BSC crossover probability p was done as explained in Section 5.1.2 using the double precision mathematical function DERFC(x) [30].

Due to 360 core limitations and operating system reliability, the maximum syndrome length for which SUBMAP look-up tables could be created using the 360 was approximately LSYN = 13 (requiring about 2 hours of CPU time). Consequently the SUBMAP creation and performance bounding programs were converted to the KL-10 Digital Equipment Corporation system using FORTRAN-10 [31]. Since FORTRAN-10 provides a very limited selection of mathematical functions, the SNR-to-p conversion was implemented using a called subroutine MERFC from the IMSL library of subroutines [32]. A comparison of both ERFC functions was made to verify that no significant differences existed over the range of SNR investigated in this work.

The simulation programs described in Section 5.5 were all run on the KL-10 system which could practically support syndrome lengths of LYSN = 17 (equivalent to roughly 4 hours of CPU time).

SNR (E_s/N_o) in db

Figure 5.13 Minimum Weight Simulation, K = 11 152

Since simulation results could be sensitive to the quality of the pseudo-random number (PRN) generator used, two different generators were implemented and compared. These were:

(1) GGUB, a standard subroutine from the IMSL library [32]

and (2) REGOL, a PRN subroutine coded at USC.

It was found that for the SNR range investigated (0 to 5 db), no appreciable difference in simulation results was obtained. Figure B.1 shows comparison results for REGOL and GGUB for a typical rate 1/2 simulation with 10^6 calls for each of the two error inputs. It can be seen that the simulation results for both PRN's coincide except at the higher SNR level of 5 db E_s/N_o. Thus in practice, higher SNR levels (e.g., 4 and 5 db) were simulated using greater than 10^6 calls per input always insuring 100 or more actual errors were created.

For reference purposes a brief description of the REGOL subroutine as implemented by G.G. Finn and J.M. Smith of USC follows.

FUNCTION REGOL

AUTHORS

G.G. FINN, IMAGE PROCESS. INST.
J.M. SMITH, DEPT. ELEC. ENG.
UNIV. SO. CALIF.

PURPOSE

GENERATES A STANDARD UNIFORMLY DISTRIBUTED RANDOM NUMBER (UNIFORM ON (0, 1)).

USAGE

X = REGOL(IR)

DESCRIPTION OF PARAMETERS

IR - INPUT. FOR THE FIRST ENTRY THIS VARIABLE MUST BE INITIALIZED TO ANY ODD INTEGER IN THE EXCLUSIVE RANGE (0, 34359738368). UPON RETURN IT WILL CONTAIN A NEW ODD INTEGER TO BE USED ON THE NEXT ENTRY TO THE SUB-PROGRAM. *THIS VARIABLE MUST NEVER BE* REDEFINED BY THE CALLING PROGRAM; OTHERWISE THE SEQUENCE OF RANDOM NUMBERS WILL BE DISTURBED.

REGOL - OUTPUT. STANDARD UNIFORMLY DISTRIBUTED REAL RANDOM NUMBER.

REMARKS

THIS SUBPROGRAM IS SPECIFIC TO THE DECSYSTEM-10 AND DECSYSTEM-20 SERIES OF COMPUTERS.

METHOD

THIS SUBPROGRAM REALIZES THE MULTIPLICATIVE CONGRUENTIAL METHOD (PERIOD 2**33) USING A MULTIPLIER CHOSEN ACCORDING TO THE RECOMMENDATION OF REFERENCE 1.

REFERENCE

[1] J.H. AHRENS, U. DIETER, AND A. GRUBE, COMPUTING, VOL. 6, PP. 121-138, 1970.

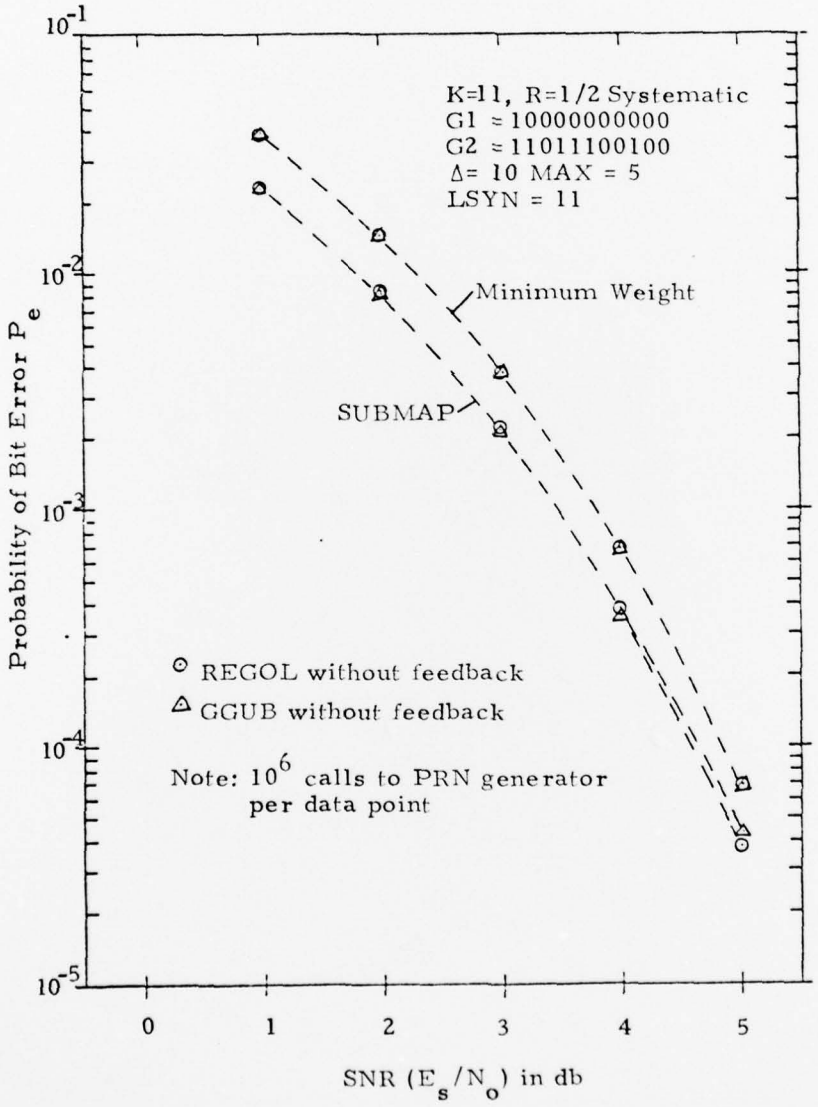


Figure B.1 Comparison of REGOL and GGUB PRN's in Simulation

of simulation discussed in Chapter 5 showed that a SUBMAP decoder based on MAP estimation outperforms conventional feedback decoding

APPENDIX C

OPTIMAL FINITE DELAY DECODING WHEN $\Delta \leq \nu$

In the optimal decoding algorithm when $\Delta \leq \nu$ as noted in Chapter 3 Section 3.3, there is a branch from Step 1 to Step 4 given below.

Step 4: (Final average and decision)

For i = 0, 1 calculate

$$\Lambda_i = \sum_{e_{t-\nu+1}} \cdots \sum_{e_t} q(e_{t-\nu} = i, e_{(t-\nu+1,\, t)})$$

If $\Lambda_0 > \Lambda_1$, $\hat{e}_{t-\nu} = 0$, otherwise $e_{t-\nu} = 1$.

Continue with Step 5.

For illustration, consider $t = \nu+1$. Step 4 yields

$$\Lambda_0 = \sum_{e_2} \cdots \sum_{e_{\nu+1}} q(e_1 = 0, e_{(2,\, \nu+1)}) = \left\{ \begin{array}{l} \text{sum of all paths in error} \\ \text{tree with } e_1 = 0, \text{ i.e., the} \\ \text{lower half tree.} \end{array} \right\}$$

$$\Lambda_1 = \sum_{e_2} \cdots \sum_{e_{\nu+1}} q(e_1 = 1, e_{(2,\, \nu+1)}) = \left\{ \begin{array}{l} \text{sum of all paths in error} \\ \text{tree with } e_1 = 1, \text{ i.e., the} \\ \text{upper half tree.} \end{array} \right\}$$

But there is a number for each $q(e_{(1,\, \nu+1)})$ path in the error tree. Thus Step 4 merely requires the adding together of the $q(e_{(1,\, \nu+1)})$'s at the end of the appropriate half trees to make the decision on $\hat{e}_{t-\nu}$ (in this case on $\hat{e}_1$).

For completeness, Step 5 following Step 4 is given below.

Step 5: (following Step 4 for case of $\Delta \leq \nu$).

Increase t by 1.

Calculate

$$q(e_{(t-\nu, t)}) = P(e_t)P(Z_t/e_{(t-\nu, t)}) \sum_{e_{t-\nu-1}} q(e_{(t-\nu-1, t-1)})$$

Return to Step 1.

Step 5 would cause $t = \nu+1$ to become $t = \nu+2$ and

$$\underbrace{q(e_{(2, \nu+2)})}_{\substack{2^{\nu+1} \text{ values} \\ \text{in this set}}} = P(e_{\nu+2})P(Z_{\nu+2}/e_{(2, \nu+2)}) \sum_{e_1} q(e_{(1, \nu+1)})$$

(average over e_1)

$$= P(e_{\nu+2})P(Z_{\nu+2}/e_{(2, \nu+2)}))\left[\begin{array}{l} q(e_1=0, e_{(2, \nu+1)})P(e_1=0)+ \\ q(e_1=1, e_{(2, \nu+1)})P(e_1=1) \end{array}\right]$$

Here one branch in the upper half tree has been added to the corresponding path in the lower half tree with each path weighted by $P(e_1=0)$ or $P(e_1=1)$. This results in a new tree as shown in Figure C.1.

Step 1: (following Step 5 when $\Delta \leq \nu$)

Set $P(e_{t-\nu}, Z_{(1, t)}, e_{(t-\nu+1, t)}) = q(e_{(t-\nu, t)})$

Go to Step 4 to continue.

Here, with $t = \nu+2$ now, one would have:

$$\underbrace{P(e_2, Z_{(1,\nu+2)}, e_{(3,\nu+2)})}_{2^{\nu+1} \text{ values}} \Leftarrow \underbrace{q(e_{(2,\nu+2)})}_{2^{\nu+1} \text{ values}}$$

and the algorithm would continue by going to Step 4 again.

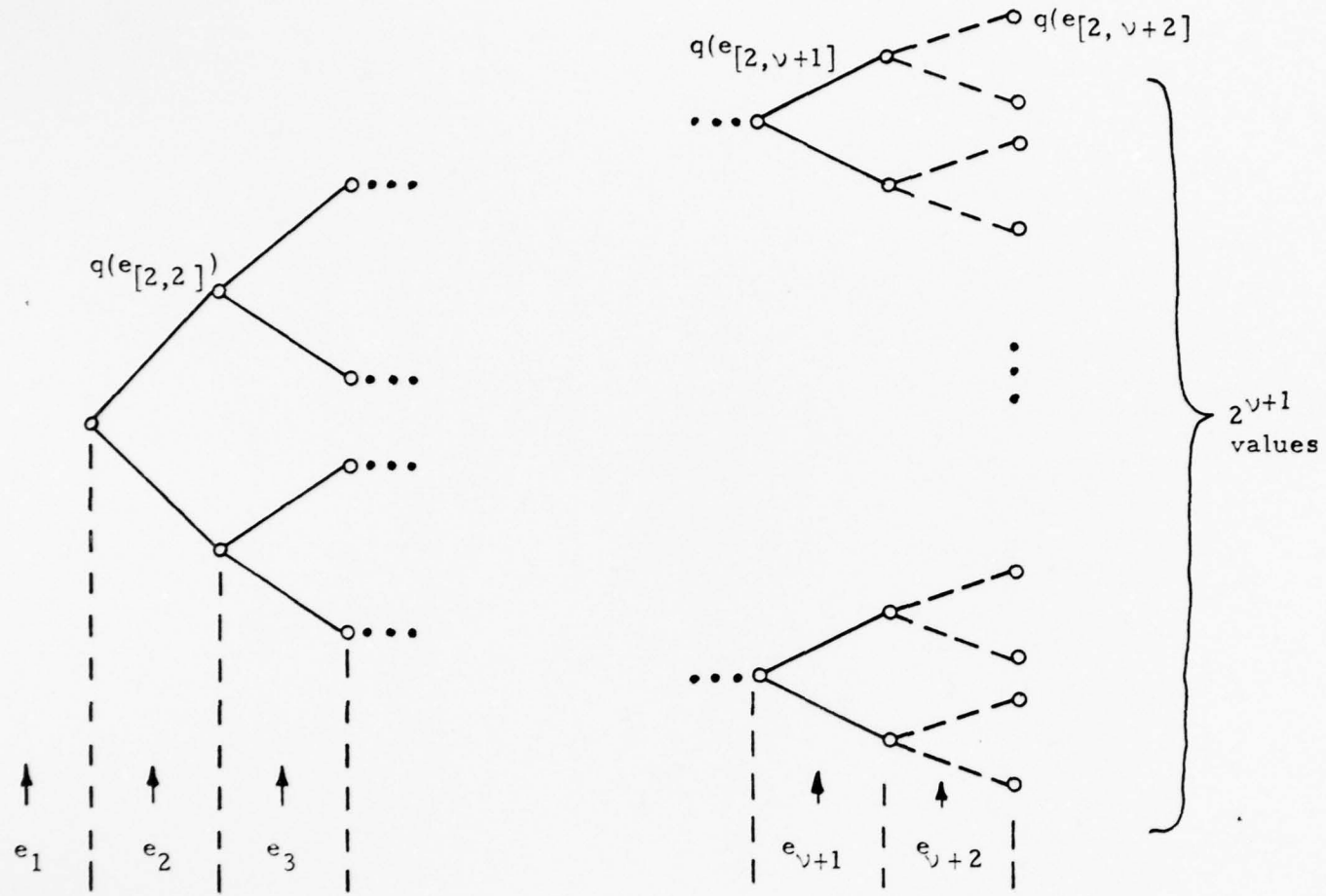


Figure C.1 New Error Tree from Steps 5 in case $\Delta \leq \nu$

REFERENCES

[1] C. E. Shannon, "A Mathematical Theory of Communication," Bell System Tech. Journal, Vol. 27, pp. 379-423, July 1948.

[2] P. Elias, "Coding for Noisy Channels," 1955 IRE National Convention Record, Vol. 3, Part 4, pp. 37-46, March 1955.

[3] B. H. Batson and R. W. Moorehead, "The Space Shuttle Orbiter Telecommunications System," Communications Society, IEEE, Vol. 14, No. 3, pp. 18-23, May 1976.

[4] W. W. Wu, "New Convolutional Codes-Part I," IEEE Trans. Communications, Vol. COM-23, No. 9, pp. 942-956, September 1975.

[5] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. Inform. Theory, Vol. IT-13, No. 2, pp. 260-269, April 1967.

[6] G. D. Forney, Jr., "Review of Random Tree Codes," in Final Report on a Coding System Design for Advanced Solar Missions, Appendix A, Codex Corp., Watertown, Mass., Contract NAS2-3637, December 1967.

[7] J. L. Massey, Threshold Decoding, M.I.T. Press, Cambridge, Mass., 1963.

[8] E. R. Berlekamp, Algebraic Coding Theory, McGraw-Hill Book Co., New York, 1968.

[9] P. L. McAdam, "MAP Bit Decoding of Convolutional Codes," Ph.D. Dissertation, Graduate School, University of Southern California, February 1974.

[10] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," IEEE Trans. Inform. Theory, Vol. IT-20, No. 2, pp. 284-287, March 1974.

[11] L. Lee, "Real Time Minimal-Bit Error Probability Decoding of Convolutional Codes," IEEE Trans. on Communications, Vol. COM-22, No. 2, pp. 146-151, February 1974.

[12] _____, "Author's Reply," IEEE Trans. on Communications, Vol. COM-22, No. 11, pp. 1894-1895, November 1974.

[13] J. P. M. Schalkwijk and A. J. Vinck, "Syndrome Decoding of Binary Rate 1/2 Convolutional Codes," IEEE Trans. on Communications, Vol. COM-24, No. 9, pp. 977-985, September 1976.

[14] A. J. Viterbi, "Convolutional Codes and Their Performance in Communications Systems," IEEE Trans. on Communications, Vol. COM-19, No. 5, pp. 751-772, October 1971.

[15] W. W. Wu, "New Convolutional Codes-Part II," IEEE Trans. Communications, Vol. COM-24, No. 1, pp. 19-33, January 1976.

[16] J. A. Heller, "Feedback Decoding of Convolutional Codes," in Advances in Communications Systems - Theory and Applications, Vol. 4, A. J. Viterbi (ed.), Academic Press, New York, pp. 261-278, 1975.

[17] J. A. Heller and I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communications," IEEE Trans. on Communications Technology, Vol. COM-19, No. 5, pp. 835-848, October 1971.

[18] R. G. Gallager, Information Theory and Reliable Communication, John Wiley and Sons, Inc. New York, 1968.

[19] J. J. Bussgang, "Some Properties of Binary Convolutional Code Generators," IEEE Trans. Inform. Theory, Vol. IT-11, No. 1, pp. 90-100, January 1965.

[20] B. D. Fritchman and J. C. Mixsell, "Comments on 'Real-Time' - Minimal-Bit-Error Probability Decoding of Convolutional Codes," IEEE Trans. on Communications, Vol. COM-22, No. 11, pp. 1892-1894, November 1974.

[21] K. Abend and B. D. Fritchman, "Statistical Detection for Communication Channels with Intersymbol Interference," Proceedings of the IEEE, Vol. 58, No. 5, pp. 779-785, May 1970.

[22] G. D. Forney, Jr., "Structural Analysis of Convolutional Codes via Dual Codes," IEEE Trans. on Inform. Theory, Vol. IT-19, No. 4, pp. 512-518, July 1973.

[23] _____, "Convolutional Codes I: Algebraic Structure," IEEE Trans. Inform. Theory, Vol. IT-16, No. 6, pp. 720-738, November 1970, and Vol. IT-17, No. 3, pp. 360, May 1971.

[24] _____, "Convolutional Codes II: Maximum-Likelihood Decoding," Information and Control, Vol. 25, pp. 222-266, July 1974.

[25] _____, "Convolutional Codes III: Sequential Decoding," Information and Control, Vol. 25, pp. 267-297, July 1974.

[26] W. W. Wu, "New Convolutional Codes-Part III," IEEE Transactions on Communications, Vol. COM-24, No. 9, pp. 946-955, September 1976.

[27] J. P. Robinson, "Error Propagation and Definite Decoding of Convolutional Codes," IEEE Trans. on Inform. Theory, Vol. IT-14, No. 1, pp. 121-128, January 1968.

[28] E. A. Bucher and J. A. Heller, "Error Probability Bounds for Systematic Convolutional Codes," IEEE Trans. on Inform. Theory, Vol. IT-16, No. 2, pp. 219-224, March 1970.

[29] C. S. Liu, and L. W. Nolte, "Performance Comparison Between a Simplified and the Optimal Decoder for Minimizing Symbol Error Rate," IEEE Trans. on Communications, Vol. COM-24, No. 1, pp. 134-138, January 1976.

[30] IBM System/360 and System/370 FORTRAN IV Language, GC28-6515-10, International Business Machines Corp., May 15, 1974.

[31] DEC System 10 FORTRAN Programmer's Reference Manual, Order No. AA-0944E-TB, Digital Equipment Corp., Maynard, MA., January 1977.

[32] IMSL Library 2, Edition 5, International Mathematical and Statistical Libraries Inc., Houston, Texas 77036, November 1975.

[33] P. L. McAdam, L. R. Welch and C. L. Weber, "MAP Bit Decoding of Convolutional Codes," presented at the 1972 International Symp. Inform. Theory, Asilomar, Ca.

[34] D. J. Costello, Jr., "Free Distance Bounds for Convolutional Codes," IEEE Trans. on Inform. Theory, Vol. IT-20, No. 3, pp. 356-365, May 1974.

[35] _____, "A Construction Technique for Random-Error-Correcting Convolutional Codes," IEEE Trans. on Inform. Theory, Vol. IT-15, No. 5, pp. 631-636, September 1969.

[36] K. J. Larsen, "Convolutional Codes with Maximal Free Distance for Rates 1/2, 1/3, and 1/4," IEEE Trans. on Inform. Theory, Vol. IT-19, No. 3, pp. 371-372, May 1973.

[37] E. Paaske, "Short Binary Convolutional Codes with Maximal Free Distance for Rates 2/3 and 3/4," IEEE Trans. on Inform. Theory, Vol. IT-20, No. 5, pp. 683-689, September 1974.

[38] S. Lin and H. Lyne, "Some Results on Binary Convolutional Code Generators," IEEE Trans. on Inform. Theory, Vol. IT-13, No. 1, pp. 134-139, January 1967.

[39] K. S. Gilhousen, Coding Systems Study for High Data Rate Telemetry Links, Linkabit Corp., San Diego, California, January 1971.

[40] H. L. Van Trees, Detection, Estimation, and Modulation Theory, Part I, John Wiley and Sons, New York, 1968.

[41] I. M. Jacobs, "Sequential Decoding for Efficient Communication from Deep Space," IEEE Trans. on Communication Technology, Vol. COM-15, No. 4, pp. 492-501, August 1967.

[42] J. Justesen, "Algebraic Construction of Rate $1/\nu$ Convolutional Codes," IEEE Trans. on Inform. Theory, Vol. IT-21, No. 5, pp. 577-582, September 1975.

[43] G. K. Huth and C. L. Weber, "Notes on Convolutional Codes," unpublished, U.S.C., Dept. of EE.

[44] J. M. Wozencraft and I. M. Jacobs, Principles of Communication Engineering, John Wiley and Sons, New York, 1965.

[45] J. P. Odenwalder, "Optimal Decoding of Convolutional Codes," Ph.D. Dissertation, Department of System Science, University of California at Los Angeles, 1970.