ON THE PERFORMANCE OF CONVOLUTIONAL CODES

Thesis by

Ivan M. Onyszchuk

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1990

(Submitted May 23, 1990)
(Corrected May 29, 1992)

Acknowledgement

I am indebted to my advisor, Professor McEliece, and my supervisor at the Jet Propulsion Laboratory (JPL), Dr. Laif Swanson, for their support, guidance, and ideas which enabled me to complete the work described in this thesis and to obtain a Ph.D. degree. Principles that they taught me will be applied for many years.

I appreciate the stimulating conversations, suggestions, and motivation provided by many researchers at JPL, including Kar-Ming Cheung, Sam Dolinar, Fabrizio Pollara, and Joe Statman. In particular, Sam Dolinar's careful and instructive refereeing improved two chapters originally written as progress reports at JPL. Fabrizio Pollara's interest and discussions about coding were inspiring.

I thank Dr. Posner and Professor Vaidyanathan for their excellent courses and for serving on my thesis committee. The comments and help of Kumar Sivarajan led to my understanding of several problems. I feel privileged to know those graduate students with whom I interacted in the EE department. The continuing encouragement and support from my family was instrumental in my completion of university degree programs.

My sincere thanks go to Professor Goodman for starting me on convolutional codes and for providing Sun computer time which proved invaluable during the computation of the NASA code complete path enumerator.

I thank the Air Force Office of Scientific Research and Caltech's Jet Propulsion Laboratory for financial support. I appreciate the generous fellowship provided during my final year of study by the Storage Technology Division at the IBM Almaden Research Center.

Abstract

This thesis contains error bounds, algorithms, and techniques for evaluating the performance of convolutional codes on the Additive White Gaussian Noise (AWGN) channel. Convolutional encoders are analyzed using simple binary operations in order to determine the longest possible "zero-run" output and if "catastrophic error propagation" may occur. Methods and algorithms are presented for computing the weight enumerator and other generating functions, associated with convolutional codes, which are used to upper-bound maximum-likelihood (i.e., Viterbi) decoder error rates on memoryless channels. In particular, the complete path enumerator $T(D, L, I)$ is obtained for the memory 6, rate 1/2, NASA standard code. A new, direct technique yields the corresponding bit-error generating function. These procedures may be used to count paths between nodes in a finite directed graph or to calculate transfer functions in circuits and networks modelled by signal flow graphs. A modified Viterbi decoding algorithm is used to obtain numbers for error bound computations.

New bounds and approximations for maximum-likelihood convolutional decoder first-event, bit, and symbol error rates are derived, the latter one for concatenated coding system analysis. Berlekamp's tangential union bound for maximum-likelihood, block decoder word error probability on the AWGN channel is adapted for convolutional codes. Approximations to bit and symbol error rates are obtained that remain within 0.2 dB of simulation results at low signal-to-noise ratios, where many convolutional codes operate but the standard bounds are useless. An upper bound on the loss caused by truncating survivors in a Viterbi decoder leads to estimates of minimum practical truncation lengths. Lastly, the power loss due to quantizing received (demodulated) symbols from the AWGN channel is studied. Effective schemes are described for uniform channel symbol quantization, branch metric calculations, and path metric renormalization in Viterbi decoders.

Table of Contents

Chapter 1

Introduction

Convolutional codes are an effective forward error correction technique in many digital communication systems including those for deep-space telemetry/image data transmission and satellite links [Clar81]. The predominant cause of data errors in these systems is Additive White Gaussian Noise (AWGN). New standards for digitial cellular radio may include a convolutional code for protection against noise and severe signal fading. Also, direct-sequence, collision-detect multiple access (CDMA) systems, like those developed and proposed recently by Qualcomm Inc., utilize powerful convolutional coding and soft-decision Viterbi decoding to achieve low power consumption. Thus, it seems likely that such coding will be used in future wireless "personal" communication systems for transmitting speech and computer data within buildings.

Unfortunately, the performance of convolutional codes is notoriously difficult to determine analytically. Indeed, little progress has been made since Viterbi's pioneering work [Vit71]. For quiet channels, such as the AWGN channel with high bit signal-to-noise ratio E_b/N_0, decoder error rates are closely upper-bounded by the well-known transfer function method. For a given code rate and high E_b/N_0, decoder error rates depend mainly upon the "free distance" of the convolutional code. The best codes are usually found by extensive computer searches for those having large free distance and also good distance properties. However, coding is often used on channels subject to severe noise levels and/or bursts. Indeed, Shannon theory suggests pushing a channel towards its physical limit and recovering from the numerous transmission errors incurred by coding data. Unfortunately, upper bounds differ markedly from simulation results at moderate E_b/N_0, so coded system analysis becomes more difficult. In fact, the bounds are not useful at the operating points of the deep-space telemetry and proposed digital cellular mobile communi-

cation systems. Although computer simulations may be used to estimate decoder error rates on a very noisy AWGN channel, the close approximations derived in Chapter 4 provide an analytical alternative.

In Chapter 2, a procedure is developed for analyzing convolutional encoders. Simple binary operations are used to determine the maximum number of consecutive all-zero blocks that an encoder can output (the longest "zero-run") and if "catastrophic error propagation" may occur. Forney's work is a reference for the structure and properties of convolutional encoders [Forn70, Forn73]. However, his solutions to the two above problems are involved, indirect, and require considerable algebraic theory to implement or understand. In comparison, the new methods are direct and straightforward applications of elementary linear algebra to binary matrices.

This thesis contains techniques for evaluating the performance of convolutional codes on the AWGN channel at moderate to low signal-to-noise ratios E_b/N_0. The complete path enumerator of a convolutional code is useful for code characterization and upper-bounding decoder error rates. However, this generating function is known only for simple codes and not the powerful or standard ones found in applications. The methods and algorithms developed in Chapter 3 facilitate the computation of the weight enumerator and also of other generating functions, associated with convolutional codes, which are used to upper bound maximum-likelihood (i.e., Viterbi) decoder error rates for memoryless channels. In particular, the complete path enumerator $T(D, L, I)$ is obtained for the memory 6, rate 1/2 NASA standard code. A new technique is described for finding bit-error generating functions without needing complicated functions $T(D, L, I)$. The methods described may be used to count paths between nodes in a finite directed graph such as a signal flow graph. Mason's gain rule for evaluating transfer functions appears to be far more complicated and less efficient than applying the linear algebraic techniques in Chapter 3.

A "concatenated" coding system, consisting of an inner convolutional code followed by an outer block code (e.g., Reed-Solomon) having symbols from $GF(2^b)$,

yields a high coding gain [Clar81]. A Viterbi decoder, which effectively utilizes the analog nature of demodulated channel symbols, outputs data with occasional error bursts that is input to a Reed-Solomon decoder. The powerful burst-error correcting ability of the outer Reed-Solomon code usually results in very few final errors. This type of concatenated coding has proved to be effective for the deep-space and satellite communication channels [Clar81]. The appropriate performance measure of the inner convolutional code is the probability that a b-bit symbol is decoded incorrectly. An upper bound for this probability is derived in Chapter 4 along with a close approximation to the simulated decoder symbol error rate. Berlekamp's tangential union bound on block codeword error probability for the AWGN channel is adapted to convolutional codes, yielding good approximations to Viterbi decoder bit and symbol error rates. Unlike the standard upper bounds which are not useful at low signal-to-noise ratios (where many convolutional codes operate), the new approximations remain within 0.25 dB of simulation results for several codes at $E_b/N_0 = 1.0$ dB.

Two practical problems in Viterbi decoder design are studied at the end of the thesis. An upper bound is derived for the loss caused by the (necessary) truncation of survivors in a Viterbi decoder. The bound may be used for estimating the minimum truncation length required in practical Viterbi decoders. Chapter 5 contains a study of the E_b/N_0 loss due to the required quantizing of demodulated symbols from the AWGN channel before decoding. Optimal schemes are described for this quantization (performed uniformly), branch metric calculations, and renormalization in Viterbi decoders. Most of these results also apply to soft-decision decoding of block codes and to other types of convolutional decoders.

Proofs that the approximations in Chapter 4 are actually upper bounds for error rates, as well as error bounds for convolutional codes used on the Rayleigh fading channel, would both be useful extensions of this work. A challenging task would be adapting material in the last three chapters to trellis codes.

1.1 Convolutional Codes and Viterbi Decoding

A brief overview of convolutional codes and Viterbi decoding is now presented in order to establish notation and definitions that are used in the four following independent chapters. Several textbooks contain detailed treatments of these two subjects [McEl77, Clar81, Vit79]. The description of Viterbi decoding is essential for the error bound derivations in Chapter 4 and for most material in Chapter 5.

A binary, rate $1/n$ convolutional encoder is a linear circuit which produces n bits computed from m state bits stored in a shift register and one input data bit, so the data throughput rate is $1/n$. These encoders, and versions with output bits periodically deleted, are overwhelmingly preferred in practical applications to those having more than one parallel input and/or nonbinary data.

An encoder is usually realized with a length-m shift register, n modulo-2 adders each one producing a binary digit for output, and an n-to-1 multiplexor which serializes these bits. The encoder is represented by n generator polynomials

$$g_{1j}(x) \;=\; g_{1j}^{(0)} \;+\; g_{1j}^{(1)}x \;+\; \cdots \;+\; g_{1j}^{(m)}x^m \qquad\qquad j = 1, \ldots, n$$

where $g_{1j}^{(t)} = 1$ if there is a connection from the t^{th} shift register cell to the j^{th} adder, and x represents a delay of one time unit. These polynomials may be written as entries in the **generator matrix** for a convolutional encoder,

$$\mathbf{G} \;=\; [g_{11}(x) \;\; g_{12}(x) \;\; \cdots \;\; g_{1n}(x)]$$

for rate $1/n$ encoders and size $k \times n$ in general (see Chapter 2). An encoder, the corresponding generator matrix $\mathbf{G}$, and the set of all sequences output by the encoder (called the **code**), will be referred to interchangeably thoughout this thesis. While $\mathbf{G}$ could contain rational functions that represent feedback, Forney proved that there is a feedback-free encoder, without larger memory, that generates the same code as $\mathbf{G}$ [Forn70]. An (n, k, m) convolutional encoder has k inputs, n outputs, and a total of m memory cells in k or fewer shift registers. Alternatively, an encoder will be described by its memory m and data throughput rate k/n.

The current state of a rate $1/n$ encoder is a binary vector $\mathbf{s} = s_1 s_2 \ldots s_m$ where s_i is the contents of the i^{th} memory cell from the data input to the encoder shift register. (Throughout this thesis, binary digits or vectors are written one after the other to imply concatentation into a single binary vector.) The current data input bit u_1 will be the state bit s_1 during the next time unit. An input data stream, called an **information** sequence, represents data for encoding and subsequent modulation and transmission across a channel. A memory 2, rate 1/2, binary convolutional encoder is shown in Figure 1.1. The generator polynomials are $g_{11}(x) = 1 + x + x^2$ and $g_{12}(x) = 1 + x^2$. Output bits y_1 and y_2 are alternately selected as the encoder output by the 2-to-1 multiplexor switch on the right.

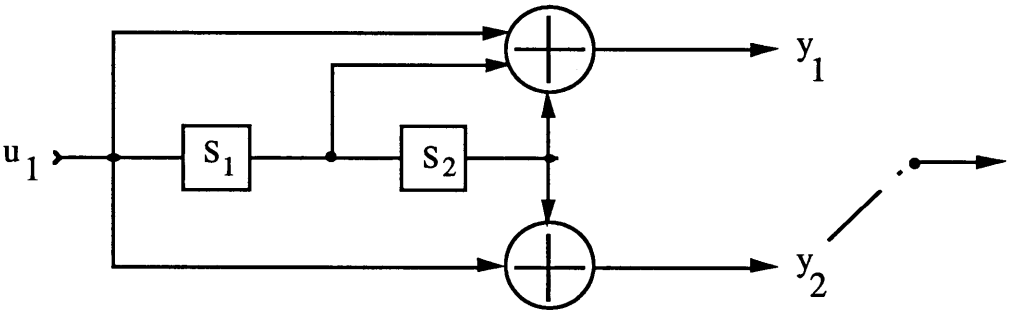


Figure 1.1 A Rate 1/2, Memory 2, Binary Convolutional Encoder.

For each data bit u_1 input, the encoder outputs n bits

$$y_j = [g_{1j}^{(0)}\ g_{1j}^{(1)}\ \cdots\ g_{1j}^{(m)}] \bullet [u_1\ s_1 s_2 \cdots s_m] \qquad j = 1, \ldots, n$$

where $\bullet$ means inner product modulo 2. The set of all possible output sequences generated by **G** is called a **convolutional code**, because information sequences are *convolved* with generator polynomial coefficients. The code generated by the encoder in Figure 1.1 starting from state 00 is described by the **trellis** diagram in Figure 1.2. This example, taken from [McEl77], is used for its simplicity. The four possible encoder states are listed on the left and indicated on the diagram by large points. Each input bit causes the encoder to change from one state to another, a

transition indicated by a trellis **branch** labelled with the input bit, a /, and the two output bits produced. A vertical column of states will be referred to as a **trellis level**. A **trellis path** is a sequence of consecutive trellis branches and has **weight** equal to the total number of 1's in the branch labels. A **fundamental trellis path** goes from the all-zero state to the all-zero state via nonzero states (e.g., the thick path in Figure 1.2). The least weight of any such path is the **free distance** of the convolutional code, denoted d_{free} (5 in Figure 1.2).

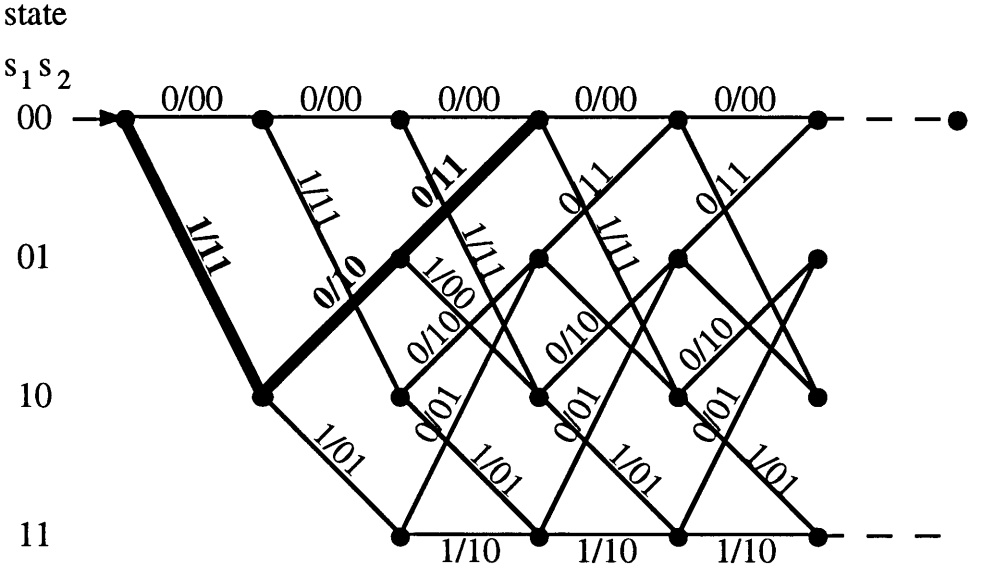


Figure 1.2 Trellis Diagram for the Encoder in Figure 1.1.

The NASA standard code, which has $d_{\text{free}} = 10$, is a popular convolutional code. It worked successfully for images transmitted by the Voyager spacecraft and continues to be chosen in coding applications. For this reason, it will be used in examples throughout this thesis. The encoder is like the one in Figure 1.1, but has 4 additional storage cells between the cells s_1 and s_2, as well as extra connections to the two modulo-2 adders that produce output bits. Since the NASA encoder has memory 6, the corresponding trellis diagram and Viterbi decoder both have 64 states.

A rate $(n-1)/n$ code may be obtained by periodically deleting bits output by a rate 1/2 encoder, to obtain a **punctured** encoder [Cain79]. The corresponding **punctured code** usually has a lower free distance (the price paid for increased rate), than the original rate 1/2 encoder. For example, deleting the third bit in every group of 4 bits output by the encoder in Figure 1.1, results in a rate 2/3 encoder and a code with $d_{\text{free}} = 3$. These high rate codes are used in applications because they may be decoded with a rate $1/n$ Viterbi decoder by inserting nulls at the receiver for deleted bits.

A Viterbi decoder **tracks** an encoder by finding, for each possible encoder state in successive trellis levels, the closest trellis path (called the **survivor path**) to the received channel symbols, which ends in the state. It accomplishes this task by first computing, for each branch into a given trellis level, a **branch metric** which is the Euclidean distance (for the AWGN channel), from the branch label with 0 and 1 mapped to $+1$ and -1, to the corresponding received channel symbols. The metric of a path is the distance from the path to the received sequence and equals the sum of metrics of the branches forming the path. Given the received sequence, the "closest" path is the most probable one transmitted. As explained in Chapter 5, received (demodulated) channel symbols are first quantized by representing them with integers before a decoder uses them.

A Viterbi decoder finds and stores, for each state, a **state metric** equal to the sum of the metrics of all branches forming the survivor path, and a **survivor** which is the information sequence that generates the survivor path. (Note that the most recent m survivor bits for a state are simply the state bits $s_1\ s_2\ \ldots\ s_m$.) Then each of the 2^m current survivors is extended by one branch in the two possible ways (for a rate 1/n encoder) and the metrics for the resulting paths are computed by adding branch metrics to state metrics. The survivor path into each state of a trellis level is obtained as the path with lowest total metric that ends in the state. In effect, a Viterbi decoder contains one trellis level of 2^m states, each one representing an encoder state.

In theory, a Viterbi decoder outputs the information bits that generate the common part of all survivor paths (they all merge far enough back in the trellis). In practice, after every group of trellis levels, the decoder outputs information bits that correspond to branches T or more levels back in the trellis. In order to output information bits, a *fixed state* Viterbi decoder always uses the survivor for one particular state, while a *best state* Viterbi decoder uses the survivor for the state with the best metric at the current trellis level. However, finding this best state is usually infeasible in high-speed decoders or in those with more than a few states. For this reason, fixed state decoders are prevalent now, even though for the same performance they require about twice the **truncation length** T used in best state decoders.

Decoder error probabilities are independent of the path transmitted across a binary-input, output-symmetric, discrete memoryless channel [Vit79, p. 79]. Thus, for simplicity of derivations in this thesis, the all-zero path will be assumed as transmitted. Also, P_d will be the probability that the decoder chooses the survivor into state 0 at any trellis level as a fundamental path at distance d (in terms of binary branch labels) from the all-zero path. For example (see Figure 1.2), with probability P_5, the decoder will select the thick path instead of all zeros as the survivor path into state 00 after 6 channel symbols are received. When this event occurs, the thick path represents an **error event**, defined as a deviation from the transmitted path. Since a convolutional code is linear, the set of all possible error events is the same as the set of all fundamental paths.

References

[Cain79] J. B. Cain, G. C. Clark, Jr., and J. M. Geist, "Punctured Convolutional Codes of Rate (n-1)/n and Simplified Maximum Likelihood Decoding," *IEEE Trans. Inform. Theory*, IT-25, pp. 97–100, January 1979.

[Clar81] G. C. Clark, Jr., and J. B. Cain, *Error-Correction Coding for Digital Communications*. New York: Plenum, 1981.

[Forn70] G.D. Forney, Jr., "Convolutional Codes I: Algebraic Structure," *IEEE Trans. Inform. Theory*, IT-16, pp. 720–738, November 1970.

[Forn73] G.D. Forney, Jr., "Structural Analysis of Convolutional Codes via Dual Codes," *IEEE Trans. Inform. Theory*, IT-19, pp. 512–518, July 1973.

[McEl77] R. J. McEliece, *The Theory of Information and Coding*. Reading, MA: Addison-Wesley, 1977.

[Vit71] A. J. Viterbi, "Convolutional Codes and their Performance in Communication Systems," *IEEE Trans. Commun.*, COM-19, pp. 751–772, October 1971.

[Vit79] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*. New York: McGraw-Hill, 1979.

Chapter 2

Zero Runs, Catastrophic Encoders, and Dual Codes

A binary (n, k, m) convolutional encoder is a finite-state, time-invariant, invertible, linear sequential circuit which produces n bits every clock cycle. These output bits are determined by the current state, represented by m bits, and k bits of data input. The encoder has a total memory of $m \geq 1$ storage cells for input bits, and a data throughput rate $R=k/n$. The results in this chapter may be generalized to include nonbinary encoders, which occur rarely in theory and virtually never in practice, at the expense of significantly complicating but not enhancing the material.

The zero-run problem [Oden70, Forn73] is to find the maximum number Z of consecutive blocks of n zeros which may be output by an encoder starting in any nonzero state. A straightforward algorithm is derived to obtain Z, whose value determines if an encoder may cause "catastrophic error propagation." This new method involves only column reduction of a binary matrix, as compared to the standard test for "catastrophic" encoders which requires reducing a matrix of polynomials. Forney finds Z by constructing a dual minimal encoder (described below) using algebraic techniques which may require several iterations. The new test is direct and simple as opposed to iterative and complicated, and is useful when searching for encoders, with fixed parameters n, k, and m, that perform best on a particular channel. Such searches are performed because there are few methods for designing "good" encoders.

Several key values, produced when the direct algorithm is used to calculate Z for an encoder, specify the memory sizes of a *dual* encoder, whose output sequences are orthogonal to every one from the original encoder. A noncatastrophic dual encoder, with a minimum number of memory cells, is obtained simply as a solution to a system of binary linear equations.

2.1 Convolutional Encoders

In this thesis, an encoder will be represented by a rank-k, $k \times n$ **generator** matrix **G** with entry in row i and column j the binary polynomial

$$g_{ij}(x) = g_{ij}^{(0)} + g_{ij}^{(1)}x + \cdots + g_{ij}^{(m_i)}x^{m_i},$$

which is the transfer function from the i^{th} encoder input u_i to the j^{th} output bit y_j, where x represents a delay of one time unit.

An encoder is a linear sequential circuit, usually realized with k shift registers, a modulo-2 adder for each of the n output bits, and an n-to-1 multiplexor which serializes these bits (see Figure 2.1). An additional 1-to-k demultiplexor might be included to convert a serial input data stream into blocks of k parallel inputs. Data bit u_i enters a shift register which contains $m_i = \max_j \deg g_{ij}(x)$ storage cells. If $g_{ij}^{(t)} = 1$, there is a connection from the output of the t^{th} cell in this shift register to the modulo-2 adder producing output bit y_j. The total encoder **memory** is $m = \sum_{i=1}^{k} m_i$ and the maximum shift register length (maximum memory with respect to any input) is $\widehat{m} = \max_i m_i = \max_{i,j} \deg g_{ij}(x)$.

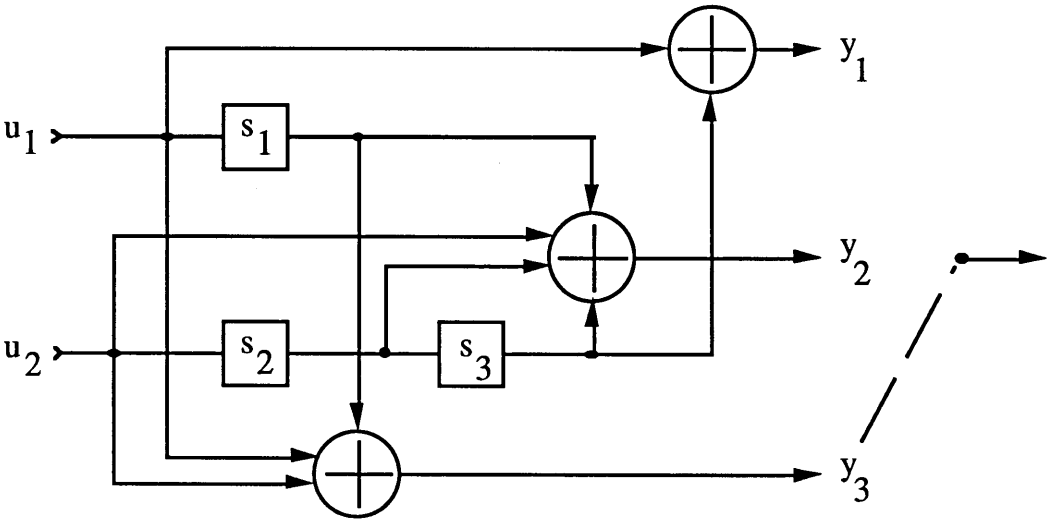


Figure 2.1 A Rate 2/3, Memory 3, Convolutional Encoder.

The current **state** of an encoder is $\mathbf{s} = \mathbf{s}_1\mathbf{s}_2 \cdots \mathbf{s}_k$, the concatenation of binary vectors $\mathbf{s}_i$ equal to the i^{th} shift register contents ($\mathbf{s}_i$ is null if $m_i = 0$). Then

$$y_j \;=\; [\mathbf{g}_{1j}\mathbf{g}_{2j} \cdots \mathbf{g}_{kj}] \bullet [u_1\mathbf{s}_1 u_2\mathbf{s}_2 \cdots u_k\mathbf{s}_k]$$

where $\mathbf{g}_{ij} = g_{ij}^{(0)} g_{ij}^{(1)} \ldots g_{ij}^{(m_i)}$, binary variables or vectors written one after the other means concatenation, and $\bullet$ denotes an inner product modulo 2. The set of all possible output sequences $\mathbf{y}$ generated by $\mathbf{G}$ is called a **convolutional code**, because input sequences $\mathbf{u}$ (which represent information for transmission) are *convolved* with generator polynomial coefficients to produce encoded bits $\mathbf{y}$.

Definition. An encoder $\mathbf{G}$ is **equivalent** to another one $\mathbf{G}'$ if and only if $\mathbf{G}$ and $\mathbf{G}'$ generate the same set of output sequences (i.e., the same code).

For example, multiplying or dividing any row of $\mathbf{G}$ by a polynomial, or performing elementary row operations on $\mathbf{G}$, yields an equivalent encoder $\mathbf{G}'$.

Definition. An encoder $\mathbf{G}$ is called **catastrophic** if there exist two information sequences $\mathbf{u}$ and $\mathbf{v}$ that differ in infinitely many positions but the corresponding output sequences $\mathbf{y}$ and $\mathbf{z}$ differ in finitely many positions.

Catastrophic error propagation occurs when a finite number of channel errors change $\mathbf{y}$ transmitted into $\mathbf{z}$ received, because the decoder might output $\mathbf{v}$ instead of $\mathbf{u}$ (since it cannot determine which was sent), thereby making infinitely many errors.

Let $\Delta_i(\mathbf{G})$ denote the i^{th} $k \times k$ subdeterminant of $\mathbf{G}$ for $i = 1, 2, \ldots, \binom{n}{k}$ and let $\psi(\mathbf{G})$ be the greatest common divisor (GCD) of these subdeterminants.

Theorem 2.1 [Sain68]. An (n, k, m) convolutional encoder $\mathbf{G}$ is catastrophic if and only if, for all integers j, $\psi(\mathbf{G}) \neq x^j$.

Simple techniques for analyzing encoders will be derived in this chapter by expressing $\mathbf{G}$ in terms of binary generator matrices $\mathbf{G}_i$ as

$$\mathbf{G} = \sum_{i=0}^{\widehat{m}} \mathbf{G}_i\, x^i \;.$$

Example 2.1 The rate 2/3 encoder in Figure 2.1 has $m_1 = 1$ and $m_2 = 2$, so the total memory is $m = 3$. The encoder state, $\mathbf{s} = s_1 s_2 s_3$ is one of $2^m = 8$ possibilities.

$$\begin{aligned}
\mathbf{G} &= \begin{bmatrix} 1 & x & 1+x \\ x^2 & 1+x+x^2 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} x^2 \\
&= \quad \mathbf{G}_0 \quad + \quad \mathbf{G}_1 \quad x + \quad \mathbf{G}_2 \quad x^2
\end{aligned}$$

$$\Delta_1(\mathbf{G}) = 1 + x + x^3,$$

$$\Delta_2(\mathbf{G}) = 1 + x^2 + x^3,$$

$$\Delta_3(\mathbf{G}) = 1 + x + x^2 + x^3.$$

Since $\psi(\mathbf{G}) = \mathrm{GCD}(\Delta_1(\mathbf{G}), \Delta_2(\mathbf{G}), \Delta_3(\mathbf{G})) = 1$, $\mathbf{G}$ is noncatastrophic.

Now if $\psi(\mathbf{G}) = x^j$, $j > 0$, then the encoder $\mathbf{G}$ contains j unnecessary memory cells. If in addition, $\mathbf{G}$ is rate $1/n$, then there are j useless delays at the input end of the shift register [Sain68]. Thus for rate $1/n$ encoders, all generator polynomials should be divided by the common factor x^j to remove the j extra delays.

In general, a given rate k/n encoder $\mathbf{G}$, $k > 1$, should first be **reduced**, by creating an equivalent one with possibly less memory, before $\mathbf{G}$ is tested for catastrophic behavior. In the first half of the reduction process, if $\mathrm{rank}(\mathbf{G}_0) < k$, then elementary row operations are performed on $\mathbf{G}$ until $\mathrm{rank}(\mathbf{G}_0) = k$:

• Let $\{\mathbf{g}_f\}_{f \in F}$ be a set of rows in $\mathbf{G}_0$ that sums to zero, where $F \subseteq [1, \ldots, k]$ is an index set. Find an $e \in F$ such that $m_e \geq m_f$ for all $f \in F$. Now replacing $g_{ej}(x)$ in $\mathbf{G}$ by $x^{-1} \sum_{f \in F} g_{fj}(x)$ for $j = 1, \ldots, n$, yields an equivalent encoder that has m_e and thus m reduced by at least 1.

For the second half of reduction, let $\mathbf{T}$ be the $k \times n$ binary matrix whose i^{th} row consists of the coefficients of x^{m_i} in the polynomials forming the i^{th} row of $\mathbf{G}$. Then the above procedure is applied to $\mathbf{T}$ until $\mathrm{rank}(\mathbf{T}) = k$, but with $g_{ej}(x)$ in $\mathbf{G}$ replaced by $\sum_{f \in F} x^{m_e - m_f} g_{fj}(x)$ for $j = 1, \ldots, n$.

A **minimal** encoder $\mathbf{G}$ is noncatastrophic, contains only polynomial entries (instead of rational functions), has $\psi(\mathbf{G}) = 1$, and $m = \max_i \deg \Delta_i(\mathbf{G})$ [Forn70]. Forney gave a procedure for obtaining an equivalent minimal encoder $\mathbf{G}'$ from any given encoder $\mathbf{G}$ and he proved that the memory of $\mathbf{G}'$ is the least possible of any encoder equivalent to $\mathbf{G}$. Thus, restricting $g_{ij}(x)$ at the beginning of this section to be polynomial causes no loss of generality. All equivalent minimal encoders have the same shift register lengths m_i [Forn70]. If an encoder $\mathbf{G}$ is reduced and noncatastrophic, then $\psi(\mathbf{G})=1$.

Example 2.1 (continued) The encoder $\mathbf{G}$ is minimal because $\psi(\mathbf{G})=1$ and m equals the maximum degree of the subdeterminants $\Delta_1(\mathbf{G}), \Delta_2(\mathbf{G})$, and $\Delta_3(\mathbf{G})$. Notice that $\mathbf{G}$ may not be reduced because $\text{rank}(\mathbf{G}_0) = \text{rank}(\mathbf{T}) = k = 2$:

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

For a rate $1/n$ encoder, dividing each $g_{1j}(x)$ by x until it contains a 1 may further reduce the memory. While the resulting encoder is not strictly equivalent to $\mathbf{G}$, only the relative *order* in which output bits are serialized changes but this does not affect the performance of the code on a memoryless channel.

2.2 Zero Runs

An important problem is to find Z, the maximum length of a "zero-run" in trellis branches, output by a convolutional encoder. Since all-zero input produces all-zero output, the encoder must be noncatastrophic and started in a nonzero state for Z to be finite and meaningful. Long zero-runs may cause a receiver to lose synchronization due to a lack of channel symbol transitions. The following example shows how Z may be found, for the minimal encoder in Example 2.1, directly from the binary generator matrices $\mathbf{G}_0, \mathbf{G}_1$, and $\mathbf{G}_2$.

Example 2.2 If $y_1 = y_2 = y_3 = 0$ is the output of the encoder in Figure 2.1, then

$$[u_1 \quad u_2 \quad s_1 \quad s_2 \quad s_3] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = [0 \quad 0 \quad 0]. \tag{2.1}$$

Note that the 5×3 matrix above, which will be called $\mathbf{B}_1$, may be written as

$$\mathbf{B}_1 = \begin{bmatrix} \mathbf{G}_0 \\ \mathbf{G}_1 \\ \mathbf{G}_2 \end{bmatrix}^-$$

where the superscript "$-$" means that all-zero rows in the matrix are removed (in this example, the first row of $\mathbf{G}_2$ is deleted). Each one of the 4 solutions to (2.1) corresponds to a distinct trellis path consisting of one branch labelled with $n = 3$ zeros. Now let u_3 and u_4 be the next input bits following u_1 and u_2 respectively. The encoder output is all zeros when u_1, u_2, u_3, u_4 are input if

$$[u_3 \quad u_4 \quad u_1 \quad u_2 \quad s_1 \quad s_2 \quad s_3] \begin{bmatrix} \mathbf{G}_0 & \mathbf{0} \\ \mathbf{G}_1 & \mathbf{G}_0 \\ \mathbf{G}_2 & \mathbf{G}_1 \\ \mathbf{0} & \mathbf{G}_2 \end{bmatrix}^- = \mathbf{0} \tag{2.2}$$

for which the only nonzero solution vector is [0 1 1 0 1 0 1]. Let $\mathbf{B}_2$ denote the 7×6 matrix in (2.2): each row corresponds to an unknown and there is one column for each equation. The dimension of the space of all-zero trellis paths of two branches equals the dimension of the left nullspace of $\mathbf{B}_2$. This quantity equals 1, the rank of $\mathbf{B}_2$ minus the number of rows. Since

$$\mathbf{B}_3 = \begin{bmatrix} \mathbf{G}_0 & & \\ \mathbf{G}_1 & \mathbf{G}_0 & \\ \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{G}_0 \\ & \mathbf{G}_2 & \mathbf{G}_1 \\ & & \mathbf{G}_2 \end{bmatrix}^-$$

has rank 9, which is also the number of rows, all zeros is the only state and input data which generates an all-zero trellis path of length 3 or more branches. Therefore, $Z = 2$. □

In order to generalize this example, let $d(\tau)$ be the dimension of the space of all-zero trellis paths of length $\tau \geq 1$ branches. Then $d(\tau)$ is the dimension of the left nullspace of

$$\mathbf{B}_\tau = \begin{bmatrix} \mathbf{G_0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{G_1} & \mathbf{G_0} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{G}_{\widehat{m}} & \vdots & & \mathbf{G_0} \\ & \mathbf{G}_{\widehat{m}} & & \vdots \\ \mathbf{0} & \mathbf{0} & & \mathbf{G}_{\widehat{m}} \end{bmatrix}^{-}$$

which has $m + k\tau$ rows, each one corresponding to an unknown, and $n\tau$ columns, one for each equation in

$$[\mathbf{u}^{(\tau)} \quad \cdots \quad \mathbf{u}^{(1)} \quad \mathbf{s}]\ \mathbf{B}_\tau = \mathbf{0} \tag{2.3}$$

where $\mathbf{u}^{(i)}$ is the i^{th} k-bit data block input. This result is summarized by

Theorem 2.2 For a reduced, noncatastrophic encoder,

$$Z = \max\{\tau : d(\tau) \geq 1\} = \max\{\tau : \text{rank}(\mathbf{B}_\tau) < m + k\tau\}.$$

For a catastrophic encoder, $d(\tau) \geq 1$ for all τ (forcing $Z = \infty$) because there is an infinite all-zero trellis path which is generated by an information sequence containing infinitely many 1's and which never returns the encoder to the all-zero state. Theorem 2.3 will show that $Z \leq d(1)$ for a reduced, noncatastrophic encoder. Therefore, only the ranks of $\mathbf{B}_\tau$ for $\tau = 1, 2, \ldots, d(1), d(1)+1$ need be computed in order to obtain Z. These values may be found successively, using elementary column operations: reduced columns of $\mathbf{B}_1$ are substituted into $\mathbf{B}_2$, and after reduction, these are substituted into $\mathbf{B}_3$, or $\mathbf{B}_4$, etc. This process is summarized by

Algorithm 1. Finding Z

- First reduce a given encoder $\mathbf{G}$, so that
  $\text{rank}(\mathbf{G}_0) = \text{rank}(\mathbf{T}) = k$.
- Compute $\text{rank}(\mathbf{B}_1)$ using column operations.
  Then $d(1) = m + k - \text{rank}(\mathbf{B}_1)$.
- **For** $\tau = 2, \ldots, d(1), d(1)+1$
  Compute $\text{rank}(\mathbf{B}_\tau)$ and
  $d(\tau) = m + k\tau - \text{rank}(\mathbf{B}_\tau)$.
  If $d(\tau) = 0$, **output** $Z = \tau - 1$ and **stop.**
- **Output** $Z = \infty$. ($\mathbf{G}$ is catastrophic)

The $d(\tau)$ values are a nonincreasing sequence, starting with $d(0)$ defined equal to m, then $d(1)$ which is usually greater than 0, continuing until $d(m) = 0$ if the encoder is reduced and noncatastrophic. In Example 2.2 where $m = 3$, $d(0) = 3$, $d(1) = 2$, $d(2) = 1$, and $d(3) = 0$.

Claim. $Z \geq \left\lceil \dfrac{m}{n-k} \right\rceil - 1.$

Proof. $\text{rank}(\mathbf{B}_\tau) \leq \min(m + k\tau, n\tau)$.

Now, $d(\tau) \geq 1$ if $\text{rank}(\mathbf{B}_\tau) < m + k\tau$,

which is always true if $n\tau < m + k\tau$, or equivalently,

$$\tau \leq \left\lceil \frac{m}{n-k} \right\rceil - 1 \quad \text{because } \tau \text{ is an integer.} \qquad \square$$

Intuitively, for all zeros to be the **only** input that produces a length-τ, all-zero trellis path, the number of equations in (2.3) must be at least equal to the number of unknowns. Then $n\tau \geq m + k\tau$, which also proves the above claim. If an encoder is catastrophic, then Z is infinite. However, $Z \leq d(1)$ for a noncatastrophic encoder,

as shown in the next theorem. Therefore, $Z = m-1$ for an $(n, n-1, m)$ encoder that is reduced and noncatastrophic, because the upper and lower bounds are equal.

2.3 Catastrophic Encoders

The next theorem specifies the largest possible value of τ for which a reduced, noncatastrophic encoder can output a length τ all-zero trellis path, while going through nonzero states. This number provides the stopping condition for Algorithm 1. First, a simple fact is proven.

Lemma 2.3 Starting from a nonzero state, a reduced encoder cannot enter the all-zero state if its output is always zeros.

Proof. The idea is that a reduced encoder must output a 1 when entering the all-zero state from a nonzero state. In order to return to the all-zero state, the last cell in one or more of the encoder's shift register(s) must contain a 1 and all inputs and other cell contents must be zero. If $k=1$, then $g_{1j}^{(m)} = 1$ for some $j \in [1, \ldots, n]$ because $m = \max_j \deg g_{1j}(x)$. Therefore, at least one output equals 1. For $k > 1$, the matrix $\mathbf{T}$ of highest order coefficients in $\mathbf{G}$ is nonsingular because $\mathbf{G}$ is reduced. Now a 1 in row i and column j of $\mathbf{T}$ corresponds to a connection, from the output of the m_i^{th} cell (the last one) in a shift register, to the modulo-2 adder producing output y_j. Hence, the encoder output is nonzero because $\mathbf{T}$ is nonsingular. □

The following result generalizes and improves upon one by Odenwalder for rate $1/n$ codes [Oden70].

Theorem 2.3 A reduced (n, k, m) encoder is catastrophic if

$$Z > d(1) = m + k - \text{rank}(\mathbf{B}_1).$$

Proof. Let $\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(1+d(1))}$ be a sequence of consecutive encoder states, starting with $\mathbf{s}^{(0)} \neq 0$, for which the encoder output is all zeros during the transition between any of these states. The idea is to show that this sequence (or an initial part

of it) can be extended indefinitely, creating all-zero output from infinitely many 1's input, thereby proving that the encoder is catastrophic. Let $\mathbf{u}^{(i)}$ denote the k-bit input block which causes the transition from $\mathbf{s}^{(i-1)}$ to $\mathbf{s}^{(i)}$ for $i = 1, \ldots, 1+d(1)$. The encoder "contents" at step i will be defined as the binary vector $\mathbf{c}^{(i)} = \mathbf{u}^{(i)}\mathbf{s}^{(i-1)}$. Let $\mathbf{u}^{(0)}$ be the vector of $\widehat{m}$ consecutive k-bit input data blocks which take the encoder from the all-zero state directly into nonzero state $\mathbf{s}^{(0)}$.

Now suppose the encoder outputs only zeros when it contains $\mathbf{c}^{(i)}$ for $i = 1, \ldots, 1 + d(1)$. Then these vectors are linearly dependent because $d(1)$ is the dimension of the left nullspace of $\mathbf{B}_1$. Define j as the least integer such that

$$\mathbf{c}^{(j)} = \sum_{f \in F} \mathbf{c}^{(f)} \quad \text{where} \quad F \subseteq [1, \ldots, j-1].$$

Now $\mathbf{c}^{(j+t)} = \sum_{f \in F} \mathbf{c}^{(f+t)} \quad t = 1, 2, \ldots$ defines an infinite sequence of binary vectors. Notice that $\mathbf{c}^{(i)}$ is *re-defined* (on purpose) for $i = j+1, \ldots, 1 + d(1)$. Recall that $\mathbf{u}^{(i)}$ takes the encoder contents from $\mathbf{c}^{(i-1)}$ to $\mathbf{c}^{(i)}$ for $i = 1, 2, \ldots, j$. Since a convolutional encoder is linear,

$$\mathbf{u}^{(j+t)} = \sum_{f \in F} \mathbf{u}^{(f+t)}$$

changes the encoder contents from $\mathbf{c}^{(j+t-1)}$ to $\mathbf{c}^{(j+t)}$ for $t = 1, 2, \ldots$. Therefore, $\mathbf{c}^{(i)}$ is an infinite sequence of consecutive encoder contents, which by linearity causes all zeros to be output. Since Lemma 2.3 guarantees that the corresponding states $\mathbf{s}^{(i)}$ are nonzero for all i, the input data sequence $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots$ contains infinitely many 1's. Also, $\mathbf{u}^{(i)}$ encodes to all zeros after an initial $\widehat{m} \cdot n$ bits which contain at least one 1. Since the corresponding output sequence differs from all zeros (generated by all-zero input) in only a few positions, where channel errors could change the transmitted 1's to zeros, the encoder is catastrophic. □

Thus, $\text{rank}(\mathbf{B}_{1+d(1)}) = \text{rank}(\mathbf{B}_{1+m+k-\text{rank}(\mathbf{B}_1)})$ determines if an encoder $\mathbf{G}$ is catastrophic. This simple test requires only column additions and perhaps interchanges. For rate $1/n$ encoders, the operations in column reducing $\mathbf{B}_m$ are equivalent to the steps for computing the GCD of the n generator polynomials. For $k > 1$,

the new method does not seem equivalent to running the standard test but realizes it in a very straightforward manner using only simple binary operations. Note that a reduced, noncatastrophic encoder is minimal because no equivalent encoder can exist with less total memory.

Example 2.3 Consider the (8,4,3) minimal encoder with generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & x & 1+x & 1+x & 1+x & 1 \\ 0 & x & 1+x & 1 & 0 & x & 1+x & 1 \\ 0 & 1 & x & 1 & x & 1 & x & 1+x \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}}_{\mathbf{G}_0} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{G}_1} x.$$

A simple calculation yields $d(1) = 3 + 4 - \text{rank}(\mathbf{B}_1) = 1$. $d(2) = 0$ because the 11 $\times$ 16 matrix

$$\mathbf{B}_2 = \begin{bmatrix} \mathbf{G}_0 & \\ \mathbf{G}_1 & \mathbf{G}_0 \\ & \mathbf{G}_1 \end{bmatrix}^-$$

has rank 11. Therefore, $Z = 1$ by Theorem 2.2 so $\mathbf{G}$ is not catastrophic according to Theorem 2.3. The standard test for catastrophic encoders involves computing the invariant factor decomposition of $\mathbf{G}$ [Forn70]. Row and column operations are used to obtain a $k \times n$ diagonal matrix Γ such that $\mathbf{G} = A\Gamma B$. Then $\psi(\mathbf{G})$ is the product of the entries in Γ. Notice that $\psi(\mathbf{G})$ is obtained without computing the $\binom{n}{k}$ $k \times k$ subdeterminants of $\mathbf{G}$.

2.4 Dual Codes

The **dual** to an (n, k) convolutional code C is an $(n, n-k)$ code $C^\perp$ whose sequences are orthogonal to every one in C. Thus if $\mathbf{G}$ is an encoder for C, then an encoder $\mathbf{H}$ for $C^\perp$ must satisfy $\mathbf{G}\mathbf{H}^T = 0$. $\mathbf{H}$ is called a **dual** encoder to $\mathbf{G}$ and has n–k shift registers with lengths $m_i{}^\perp$ for $i = 1, \ldots, n$–k. If $\mathbf{G}$ and $\mathbf{H}$ are both minimal,

then their total memories and subdeterminants are the same [Forn73]. Therefore, if $\mathbf{G}$ is minimal with subdeterminants $\Delta_i(\mathbf{G})$ and $\mathbf{G}$ has $k = n-1$ inputs, then

$$\mathbf{H} = [\Delta_1(x), \Delta_2(x), \ldots, \Delta_n(x)]$$

is a dual minimal encoder to $\mathbf{G}$. Observe that $m^\perp = m$, $\mathbf{H}$ has polynomial entries whose GCD equals 1, and it is easily verified that $\mathbf{G}\mathbf{H}^T = 0$. Dual encoders to those having $k \neq n-1$ will be found in this section.

Theorem [Forn73]. Let $\mathbf{H}$, with memories ${m_i}^\perp$ for $1 \leq i \leq n-k$, be a minimal encoder for the code dual to any rate k/n binary convolutional code. Then the number of distinct all-zero paths of length τ branches in the code trellis is equal to $2^{d(\tau)}$, where

$$d(\tau) = \sum_{i\,:\,{m_i}^\perp > \tau} ({m_i}^\perp - \tau). \tag{2.4}$$

Corollary. The longest all-zero trellis path, nowhere merged with the all-zero state, has length $\widehat{m}^\perp - 1$. Also,

$$\left\lceil \frac{m}{n-k} \right\rceil \leq \widehat{m}^\perp \leq m \qquad (\text{ because } \sum_{i=1}^{n-k} {m_i}^\perp = m^\perp = m).$$

The above results imply that $Z = \max\{\tau : d(\tau) > 0\} = \widehat{m}^\perp - 1 \leq m-1$. Therefore, an (n, k, m) encoder $\mathbf{G}$ is not minimal if $Z \geq m$, in which case either $\mathbf{G}$ is catastrophic or else it may be realized with fewer than m memory cells.

If $\mathbf{G}$ is not catastrophic, then the memory sizes ${m_i}^\perp$ of a dual minimal encoder $\mathbf{H}$ are obtained by inverting (2.4) and using the $d(\tau)$ values obtained from the ranks of $\mathbf{B}_\tau$ for $\tau \geq 1$, and the definition $d(0) = \sum_{i=1}^{n-k} {m_i}^\perp = m^\perp = m$.

Define sr_τ as the number of length-τ shift registers in $\mathbf{H}$, $|\{j : {m_j}^\perp = \tau\}|$.

Lemma 2.4 For $\tau \geq 1$, $\mathrm{sr}_\tau = d(\tau-1) - 2d(\tau) + d(\tau+1)$.

Proof.
$$\begin{aligned} & d(\tau-1) - 2d(\tau) + d(\tau+1) \\ &= \sum_{i:\, m_i{}^{\perp} > \tau-1} (m_i{}^{\perp} - \tau + 1) \;-\; 2 \sum_{i:\, m_i{}^{\perp} > \tau} (m_i{}^{\perp} - \tau) \;+\; \sum_{i:\, m_i{}^{\perp} > \tau+1} (m_i{}^{\perp} - \tau - 1) \\ &= \sum_{i:\, m_i{}^{\perp} > \tau} (m_i{}^{\perp} - \tau + 1) \;+\; \sum_{i:\, m_i{}^{\perp} = \tau} (\tau - \tau + 1) \;-\; 2 \sum_{i:\, m_i{}^{\perp} > \tau} (m_i{}^{\perp} - \tau) \\ &\quad + \sum_{i:\, m_i{}^{\perp} > \tau} (m_i{}^{\perp} - \tau - 1) \;-\; \sum_{i:\, m_i{}^{\perp} = \tau+1} (\tau + 1 - \tau - 1) \\ &= \sum_{i:\, m_i{}^{\perp} = \tau} 1 \\ &= \mathrm{sr}_\tau . \end{aligned}$$

Also, $\mathrm{sr}_0 = n - k - \sum_{\tau=1}^{m} \mathrm{sr}_\tau$. These shift register lengths determine the memory sizes $m_i{}^{\perp}$ (whose sum is $m^{\perp} = m$) of $\mathbf{H}$. The $m_i{}^{\perp}$ values specify the all-zero rows in the binary matrices $\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}_{\widehat{m}^{\perp}}$ because row i in $\mathbf{H}_j$ is zero if $j > m_i{}^{\perp}$. Now since $\mathbf{G}\mathbf{H}^T = 0$,

$$\left(\sum_{i=0}^{\widehat{m}} \mathbf{G}_i \, x^i \right) \left(\sum_{j=0}^{\widehat{m}^{\perp}} \mathbf{H}_j^T \, x^j \right) = 0$$

$$\text{or} \quad \sum_{i=0}^{j} \mathbf{G}_i \, \mathbf{H}_{j-i}^T = 0 \quad \text{for } j = 0, 1, 2, \ldots, \widehat{m} + \widehat{m}^{\perp}. \tag{2.5}$$

A dual encoder $\mathbf{H}$ is a solution, for this system of binary linear equations, which preserves the $m_i{}^{\perp}$ values so that $\mathbf{H}_{\widehat{m}}^{\perp}$ has no additional all-zero rows. One such solution always exists [Forn70].

Claim. The encoder $\mathbf{H}$ obtained by solving (2.5) is minimal.

Proof. $\mathbf{H}$ has the same total memory as $\mathbf{G}$, and also same $m_i{}^{\perp}$ values as any dual minimal encoder to $\mathbf{G}$. By construction, $\mathbf{H}$ is polynomial. If $\mathbf{H}$ was catastrophic, then $\deg \psi(\mathbf{H}) \geq 1$, so there would be a nontrivial linear combination h^* of rows in $\mathbf{H}$ that sums to 0 mod ψ. h^* could be found by row-reducing $\mathbf{H}$ modulo ψ towards triangular form, with all arithmetic modulo ψ, until a zero row is encountered. Let

j be a row of $\mathbf{H}$, such that ${m_j}^{\perp} \geq {m_k}^{\perp}$ for all rows j and k which contribute to h^*. Then replacing row j in $\mathbf{H}$ with h^*/ψ would result in an equivalent encoder with less memory. A similar argument shows that the matrix $\mathbf{T}$ of highest order coefficients in $\mathbf{H}$ must be nonsingular and that $\mathbf{H}$ has full rank $n-k$. Therefore, $\mathbf{H}$ is minimal because it cannot be reduced in any way. □

By repeatedly adding together or interchanging rows of a binary matrix, and then back-substituting values obtained, a minimal $\mathbf{H}$ is easily found. This approach is simple and direct as compared to Algorithm 2 extracted from [Forn70].

Algorithm 2. Forney's Method for Calculating $\mathbf{H}$

1) Row reduce $\mathbf{G}$ to obtain an equivalent encoder $\mathbf{G}'$ which has the k by k identity matrix $\mathbf{I}_k$ as its first k columns, and causal rational functions in the last n–k columns, which form a matrix called $\mathbf{P}$.
2) $\mathbf{H} = [\mathbf{P}^{\mathrm{T}} : \mathbf{I}_k]$ satisfies $\mathbf{G}'\mathbf{H}^{\mathrm{T}} = 0$, so multiply all rows of $\mathbf{H}$ through by denominator polynomials to clear all fractions.
3) Make $\mathbf{H}$ noncatastrophic using the method in the proof above and then reduce it by the procedure described in Section 2.1.

Example 2.4 For the (3,1,2) encoder with transfer function matrix

$$\begin{aligned}\mathbf{G} &= [1+x^2 \quad 1+x+x^2 \quad 1+x+x^2] \\ &= [1 \quad 1 \quad 1] + [0 \quad 1 \quad 1]\,x + [1 \quad 1 \quad 1]\,x^2,\end{aligned}$$

$$d(0) = m^{\perp} = m = 2$$

$$d(1) = 2 + 1 - \mathrm{rank}(\mathbf{B}_1) = 1$$

$$d(2) = 2 + 2 - \mathrm{rank}(\mathbf{B}_2) = 0$$

$$d(\tau) = 0 \quad \text{for} \quad \tau \geq 3$$

$$\text{sr}_1 = d(0) - 2d(1) + d(2) = 0$$

$$\text{sr}_2 = d(1) - 2d(2) + d(3) = 1.$$

Thus $\mathbf{H}$ has one only shift register, of length $m^{\perp} = m = 2$. Setting

$$\mathbf{H}_0 = \begin{bmatrix} h_{11}^{(0)} & h_{12}^{(0)} & h_{13}^{(0)} \\ h_{21}^{(0)} & h_{22}^{(0)} & h_{23}^{(0)} \end{bmatrix} \quad \mathbf{H}_1 = \begin{bmatrix} 0 & 0 & 0 \\ h_{21}^{(1)} & h_{22}^{(1)} & h_{23}^{(1)} \end{bmatrix} \quad \mathbf{H}_2 = \begin{bmatrix} 0 & 0 & 0 \\ h_{21}^{(2)} & h_{22}^{(2)} & h_{23}^{(2)} \end{bmatrix}$$

in (2.5) yields a system of 6 equations with 12 unknowns. (However, $\mathbf{G}_0\mathbf{H}_0^{\text{T}} = 0$ and $\mathbf{G}_0\mathbf{H}_1^{\text{T}} + \mathbf{G}_1\mathbf{H}_0^{\text{T}} = 0$ force $h_{11}^{(0)} = 0$ and $h_{12}^{(0)} = h_{13}^{(0)} = 1$, leaving 4 equations with 9 unknowns.) Each solution for which $\mathbf{H}_2$ is nonzero corresponds to a dual minimal encoder, such as

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 \\ 1+x+x^2 & 1+x^2 & 0 \end{bmatrix}.$$

In summary, a given encoder $\mathbf{G}$ is analyzed, using the simple binary techniques described in this chapter, by running Algorithm 1. $\mathbf{G}$ is first reduced; then the $d(\tau)$ values and Z are computed. If Z is infinite, then the encoder is catastrophic. Otherwise, the $m_i{}^{\perp}$ shift register lengths of a dual minimal encoder $\mathbf{H}$ are computed using Lemma 2.4 and then (2.5) is solved for $\mathbf{H}$.

References

[Forn70] G.D. Forney, Jr., "Convolutional Codes I: Algebraic Structure," *IEEE Trans. Inform. Theory*, IT-16, pp. 720–738, November 1970.

[Forn73] G.D. Forney, Jr., "Structural Analysis of Convolutional Codes via Dual Codes," *IEEE Trans. Inform. Theory*, IT-19, pp. 512–518, July 1973.

[Oden70] J.P. Odenwalder, "Optimal Decoding of Convolutional Codes," Ph.D. dissertation, EE Department, Univ. of Calif. Los Angeles, 1970, pp. 47–48.

[Sain68] M.K. Sain and J.L. Massey, "Inverses of Linear Sequential Circuits," *IEEE Trans. Computers*, C-17, pp. 330–337, April 1968.

Chapter 3

Finding the Complete Path and Weight Enumerators of Convolutional Codes

In order to bound maximum-likelihood decoder error probabilities, Viterbi defined the complete path enumerator $T(D, L, I)$ of a convolutional code [Vit71]. The bit-error generating function, $B(D) = \partial T(D, L, I)/\partial I$ at $L=I=1$, and code weight enumerator $T(D) = T(D, 1, 1)$ are two important quantities for upper-bounding node, bit, or symbol error rates of a maximum-likelihood (e.g., Viterbi) decoder. However, these functions are known only for encoders with few states. Hence, error bounds are evaluated using numerical matrix multiplications [Vit79, p. 163], which require extensive computations for each channel noise level. As an alternative, algorithms were developed to calculate the first few coefficients of these enigmatic generating functions [Cedr89, Roan89]. For example, a modified Viterbi decoding algorithm produced the first 50 nonzero coefficients of $B(D)$ and $T(D)$ for the $m = 14$ Galileo code [Dol88]. However, the number of coefficients required to approximate error bounds depends upon the channel noise level and code rate.

A method for finding $T(D)$ or $T(D, L, I)$ is described in this chapter. These generating functions were obtained, using a determinant algorithm derived in Section 3.3, for the memory 6, rate 1/2, NASA standard code (among many others). An encoder is represented by a state diagram and transition matrix of edge weights. Then, simple matrix operations are used to solve for $T(D)$ or $B(D)$. This method works for transfer functions of circuits or networks modelled by a directed graph.

Poles and residues of $T(D)$ and $B(D)$ are used to determine, or approximate closely, several code properties. For example, the union bounds diverge at the least-magnitude pole of $T(D)$ [Vit71]. Additional poles and residues yield the dominant terms in the partial fraction expansion of $T(D)$, which lead to concise but accurate analytic approximations to the coefficients in the corresponding infinite series. Sim-

ilar techniques are applied to estimate the coefficients in the expansion of $B(D)$. An algorithm is described for calculating the initial coefficients of these generating functions. A new, direct method for computing $B(D)$ without $T(D, L, I)$ also yields generating functions for upper bounding the bit error probabilities of rate $(n-1)/n$ trellis codes constructed by set partitioning signals (e.g., Ungerboeck codes).

3.1 The Complete Path Enumerator

The complete path enumerator $T(D, L, I)$ of a code contains the number of *fundamental* trellis paths having three given parameters: weight, length, and number of input 1s. Recall that $k \cdot n$ transfer functions called generator polynomials $g_{ij}(x)$ may represent an encoder, which has memory $m = \sum_{i=1}^{k} \max_j[\deg\ g_{ij}(x)]$. The state $\mathbf{s}$ of an encoder is the concatenation of all shift registers' contents. The integer s whose binary representation is $\mathbf{s}$, with least significant bit on the left, will also represent an encoder state. The state diagram for an encoder is a directed graph whose edges correspond to branches in the associated code trellis diagram and have labels $a_{i,j} = D^d L I^b$ if there is an edge from state j into state i (otherwise $a_{i,j} = 0$) where b and d are the number of 1's input to and output by the encoder.

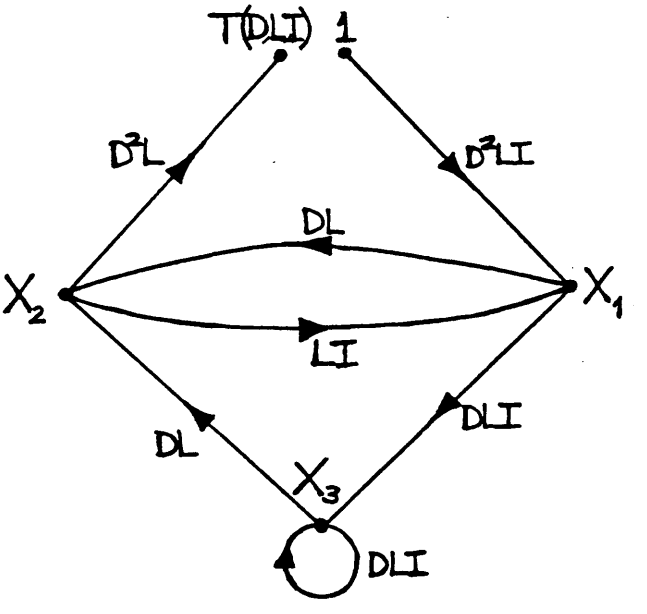


Figure 3.1 A Modified Encoder State Diagram.

In order to emphasize paths which will be counted, the encoder graph is modified by deleting the loop at state zero and representing state zero by one node for incoming branches and another node for outgoing branches (see Figure 3.1). Define $\mathbf{A}$ as the $2^m \times 2^m$ matrix whose entry in row i and column j is $\delta_{i,j} - a_{i,j}$, which is nonzero only if there is a directed edge from state j into state i ($\delta_{i,j} = 1$ if $i = j$ and 0 otherwise). Actually, in this section and the next, the first row and column (called 0) of $\mathbf{A}$ will be ignored because the examples use rate $1/n$ encoders. Let $X_s = X_s(D, L, I)$ be the trivariate generating function of all *simple* paths: those from state 0 into state s via nonzero states. X_s will be called the path **transmission** from state 0 into state s. Note that X_s is indexed by the destination state (s) while the source state (0) is fixed. X_0 counts all simple paths into state 0, called *fundamental* paths, by weight d. Consider the set of linear algebraic equations:

$$\mathbf{A}\ [X_1\ X_2\ \ldots\ X_{2^m-1}]^{\mathrm{T}} = [\mathrm{a}_{1,0}\ \mathrm{a}_{2,0}\ \ldots\ \mathrm{a}_{2^m-1,0}]^{\mathrm{T}}$$

By Cramer's Rule,

$$X_i = \frac{\mathbf{det}(\mathbf{A}_i)}{\mathbf{det}(\mathbf{A})}$$

if $i > 0$, where $\mathbf{A}_i$ is $\mathbf{A}$ with all column i entries $a_{r,i}$ replaced by $a_{r,0}$ for all rows r. The **complete path enumerator** of a convolutional code is

$$T(D, L, I) = X_0 = \sum_{j=1}^{2^k-1} a_{0,z_j} X_{z_j}$$

where $\{z_j\}$ are the 2^k-1 states having edges into state 0 (so $a_{0,z_j} \neq 0$). The code weight enumerator is $T(D) = T(D, 1, 1) = \sum_{d=d_{\mathrm{free}}}^{\infty} a(d) D^d$, where $a(d)$ is the number of fundamental trellis paths of weight d, and d_{free} is the least weight of these paths. The **bit-error** and **path-length** generating functions are defined as

$$B(D) = \left.\frac{\partial T(D, L, I)}{\partial I}\right|_{L=I=1} = \sum_{d=d_{\mathrm{free}}}^{\infty} i(d) D^d$$

$$L(D) = \left.\frac{\partial T(D, L, I)}{\partial L}\right|_{L=I=1} = \sum_{d=d_{\mathrm{free}}}^{\infty} \ell(d) D^d.$$

Notice that $i(d)$ is the total number 1's in all input sequences to the encoder which generate weight d fundamental paths, while $\ell(d)$ denotes the total length in trellis branches of these paths. Also, the denominator of $B(D)$ or $L(D)$ is simply that of $T(D,1,1)^2 = T(D)^2$. These generating functions lead to upper bounds for convolutional decoder error rates.

Example 3.1 The rate 1/2 encoder in Figure 1.1 is in the binary state $\mathbf{s} = s_1 s_2$, or equivalently, the integer state $s = 2s_2 + s_1$. Figure 3.1 leads to the equations $\mathbf{A}\,[X_1\ X_2\ X_3]^{\mathrm{T}} = [a_{1,0}\ 0\ 0]^{\mathrm{T}}$:

$$\begin{bmatrix} 1 & -LI & 0 \\ -DL & 1 & -DL \\ -DLI & 0 & 1-DLI \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} D^2LI \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Then} \quad X_2 = \frac{\mathbf{det}(\mathbf{A}_2)}{\mathbf{det}(\mathbf{A})} = \frac{D^3L^2I}{1-DLI-DL^2I}$$

and $T(D,L,I) = X_0 = D^2LX_2$. Therefore, the code weight enumerator is

$$T(D) = \sum_{d=d_{\text{free}}}^{\infty} a(d)D^d = \frac{D^5}{1-2D} = \sum_{d=5}^{\infty} 2^{d-5}D^d.$$

Hence $d_{\text{free}} = 5$ and $a(d) = 2^{d-5}$ is the number of weight d fundamental paths. Also,

$$B(D) = \left.\frac{\partial T(D,L,I)}{\partial I}\right|_{L=I=1} = \frac{D^5}{(1-2D)^2} = \sum_{d=d_{\text{free}}}^{\infty} i(d)D^d,$$

so $i(d) = (d-4)2^{d-5}$ for $d \geq 5$, and zero otherwise. □

Applying Mason's gain rule, to a modified encoder state diagram regarded as a signal flow graph with unity input, is a complicated method of finding $T(D,L,I)$ and works in practice only for encoders having few states [Lin83]. For encoders with $k > 1$ parallel inputs, an extra row and column are added to $\mathbf{A}$ to account for the transmissions into and out of the split state 0. Thus, $T(D)$ or $T(D,L,I)$ are obtained simply as the ratio of two determinants, which will be calculated simultaneously by adjoining an extra column to $\mathbf{A}$ prior to row-reducing it into triangular form.

3.2 Reducing the Matrix A For Rate 1/n Encoders

For encoders with only $k=1$ input, there are only $3(2^m-1)-2$ nonzero out of $(2^m-1)^2$ entries in $\mathbf{A}$. Since $\mathbf{A}$ is sparse when $m \geq 3$, simple row operations, which take advantage of the locations of these nonzero entries, are performed before computing the determinant. The following new example illustrates the reduction procedure, which works in general on $\mathbf{A}$ for any rate $1/n$ encoder. A similar technique could reduce $\mathbf{A}$ for encoders with $k > 1$, though not as significantly. For the memory 3 encoder with generator polynomials $g_{11}(x) = 1+x+x^3$ and $g_{12}(x) = 1+x+x^2+x^3$,

$$(\mathbf{A}_{2^{m-1}}), \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & (D^2)-1 & 0 & 0 & 0 \\ -D^2 & 1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & -D^2 & 0 & 0 \\ 0 & -D & 0 & (0)1 & 0 & -D & 0 \\ 0 & -D & 0 & 0 & 1 & -D & 0 \\ 0 & 0 & -D & 0 & 0 & 1 & -D \\ 0 & 0 & -D & 0 & 0 & 0 & 1-D \end{bmatrix}$$

with $L=I=1$ to simplify entries. The round brackets (parentheses) above indicate values in $\mathbf{A}_{2^{m-1}}$ which are different from those in $\mathbf{A}$. The determinants of only these two matrices are needed for $T(D)$ and this notation will lead to their simultaneous evaluation.

As in Gaussian elimination, $-a_{r,\lfloor r/2 \rfloor}$ times row $\lfloor r/2 \rfloor$ is added to rows $r = 2$ to 2^m-1 to zero columns 1 to $2^{m-1}-1$ in $\mathbf{A}, (\mathbf{A}_{2^{m-1}})$ below the main diagonal:

$$\begin{bmatrix} 1 & 0 & 0 & (D^2)\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & (D^4)-D^2 & -1 & 0 & 0 \\ 0 & 0 & 1 & (D^2)\ -1 & -D^2 & 0 & 0 \\ 0 & 0 & 0 & (D^5)1-D^3 & -D & -D & 0 \\ 0 & 0 & 0 & (D^5)-D^3 & 1-D & -D & 0 \\ 0 & 0 & 0 & (D^3)-D & -D^3 & 1 & -D \\ 0 & 0 & 0 & (D^3)-D & -D^3 & 0 & 1-D \end{bmatrix}$$

Therefore, $\mathbf{det}(\mathbf{A})$ equals the determinant of the resulting lower right 2^{m-1} by 2^{m-1} submatrix. To further reduce $\mathbf{A}$, row $\lfloor r/2 \rfloor$ times $-a_{r,2^{m-1}+\lfloor r/2 \rfloor}$ is added to each row $r = 2^m-3$ to 2^{m-1} (4 to 5 here), so that columns $2^{m-1}+2^{m-2}$ to 2^m-2 (both 6 here) become zero above the main diagonal. Now $\mathbf{det}(\mathbf{A}) = \mathbf{det}(\tilde{\mathbf{A}})$, where $\tilde{\mathbf{A}}$ is

the new lower right 2^{m-1} by 2^{m-1} submatrix

$$(\tilde{\mathbf{A}}_{2^{m-1}}), \tilde{\mathbf{A}} = \begin{bmatrix} (D^4+D^5)1-D^2-D^3 & -D-D^4 & 0 & -D^2 \\ (D^4+D^5)-D^2-D^3 & 1-D-D^4 & 0 & -D^2 \\ (D^3) \quad -D & -D^3 & 1 & -D \\ (D^3) \quad -D & -D^3 & 0 & 1-D \end{bmatrix}$$

In order to zero column number 2^{m-1} of $\tilde{\mathbf{A}}$,$(\tilde{\mathbf{A}}_{2^{m-1}})$, above the lower right entry $\tilde{a}_{2^{m-1},2^{m-1}} = a_{2^m-1,2^m-1} \neq 0$, $-\tilde{a}_{r,2^{m-1}}/\tilde{a}_{2^{m-1},2^{m-1}}$ times row 2^{m-1} is added to each row $r=1$ to $2^{m-1}-1$. Define $\mathbf{B}, (\mathbf{B}_{2^{m-1}})$ as the resulting upper left 2^{m-2} by 2^{m-2} submatrices :

$$(\mathbf{B}_{2^{m-1}}), \mathbf{B} = \begin{bmatrix} (D^4+D^5-D^6)1-D-D^2-D^3+D^4 & -D+D^2-D^4 \\ (D^4+D^5-D^6) \quad -D^2-D^3+D^4 & 1-2D+D^2-D^4 \end{bmatrix} (1-D)^{-1}$$

This reduction method simultaneously produces two dense 2^{m-2} by 2^{m-2} matrices $(\mathbf{B}_{2^{m-1}}), \mathbf{B}$ with the same determinants as the corresponding original sparse 2^m-1 by 2^m-1 matrices $(\mathbf{A}_{2^{m-1}}), \mathbf{A}$. Then,

$$T(D) = \frac{D^2 \det(\mathbf{B}_{2^{m-1}})}{\det(\mathbf{B})} = \frac{D^6 - 2D^8 + D^9}{1 - 3D + 2D^2 - D^3 + D^4}$$

An efficient algorithm for computing the determinant of $\mathbf{B}$ (described next) will produce $T(D, L, I)$ for the 64-state NASA code, or $T(D)$ for any rate $1/n$ code having memory ≤ 9. For codes with $k>1$, $\mathbf{A}$ could be reduced, but not as significantly as above. Also, an additional row and column for the split state 0 are appended to $\mathbf{A}$ so that one determinant (with an extra column adjoined like $(\mathbf{B}_{2^{m-1}}), \mathbf{B}$ above) yields $T(D)$ or $T(D, L, I)$.

3.3 Computing Determinants

A modified Gaussian elimination algorithm is derived for reducing to triangular form any $N \times N$ matrix $\mathbf{B}$ that has entries from a Euclidean domain such as the set of all polynomials with integer coefficients. Successive matrices $\mathbf{B}^{(N)}$, $\mathbf{B}^{(N-1)}$, $\mathbf{B}^{(N-2)}$, $\ldots, \mathbf{B}^{(1)}$ are computed, each one having the same determinant (up to sign) as $\mathbf{B}$. Using elementary row operations, columns of $\mathbf{B}^{(N)}$ are successively made zero above the main diagonal to create a lower triangular matrix $\mathbf{B}^{(1)}$ with determinant (the

Algorithm 3. Basic Gaussian Elimination
(for a matrix $\mathbf{B}$ which leads to nonzero pivots)

set $\mathbf{B}^{(N)} = \mathbf{B}$: $b_{i,k}^{(N)} = b_{i,k}$ for $1 \le i,k \le N$.
for $j = N$ **to** 2 (step)
  set $\mathbf{B}^{(j-1)} = \mathbf{B}^{(j)}$ and $b_{i,N}^{(j-1)} = 0$ for $i < N$.
  for $i = j-1$ **to** 1 (row index)
    for $k = j-1$ **to** 1 (column index)

$$b_{i,k}^{(j-1)} = \frac{b_{i,k}^{(j)}\, b_{j,j}^{(j)} - b_{i,j}^{(j)}\, b_{j,k}^{(j)}}{b_{j,j}^{(j)}} \tag{3.1}$$

output $\mathbf{B}^{(1)}$: $b_{i,k}^{(1)} = b_{i,k}^{(i)}$ for $k \le i$ and $b_{i,k}^{(1)} = 0$ for $k > i$.

product of diagonal entries) equal to $\mathbf{det}(\mathbf{B})$. Starting with $\mathbf{B}^{(N)} = \mathbf{B}$, step j in the standard Gaussian elimination process yields new entries $b_{i,k}^{(j-1)}$ for $1 \le i,k \le j-1$ in $\mathbf{B}^{(j-1)}$, as shown in Algorithm 3.

The input and output matrices $\mathbf{B}$ and $\mathbf{B}^{(1)}$ are row equivalent. One additional check is required for Algorithm 3 to work for all matrices: if $b_{j,j}^{(j)} = 0$ prior to step j, then a column k with $b_{j,k}^{(j)} \neq 0$ must first be interchanged with column j. Since this operation negates the determinant, a counter t is incremented to record the event. If no such column k exists, row j is zero, so the algorithm should output $\mathbf{det}(\mathbf{B}) = \mathbf{det}(\mathbf{B}^{(j-1)}) = 0$ and then stop, as done in Algorithm 4.

The $\mathbf{A}$ and $\mathbf{B}$ matrices, which are constructed for finding generating functions $T(D)$ or $T(D,L,I)$, have polynomial entries. The reduced matrices $\mathbf{B}^{(N-1)}, \ldots, \mathbf{B}^{(1)}$ will usually contain rational functions. Then (3.1) will be a complicated operation which includes finding the greatest common divisor (GCD) of the numerator and denominator of $b_{i,k}^{(j-1)}$ in order to cancel common factors. When the entries in $\mathbf{B}$ are multivariate polynomials (trivariate when $T(D,L,I)$ is desired), then comput-

ing these GCDs is difficult. Since $\mathbf{det(B)}$ is a polynomial, it should be calculated by working with polynomials only. In order to explain the following simplification of Algorithm 3, entries in $\mathbf{B}$ will be polynomials with integer coefficients. Rational functions may be completely avoided in the reduced matrices by forcing a specific polynomial as the denominator of $b_{i,k}^{(j-1)}$ for all i and k. First write $b_{i,k}^{(j-1)} = n_{i,k}^{(j-1)}/n_{j,j}^{(j)}$. Then from (3.1),

$$b_{i,k}^{(j-1)} = \frac{n_{i,k}^{(j-1)}}{n_{j,j}^{(j)}} = \frac{n_{i,k}^{(j)}\, n_{j,j}^{(j)} - n_{i,j}^{(j)}\, n_{j,k}^{(j)}}{[n_{j+1,j+1}^{(j+1)}]^2\; n_{j,j}^{(j)}/n_{j+1,j+1}^{(j+1)}}$$

$$\text{so} \quad n_{i,k}^{(j-1)} = \frac{n_{i,k}^{(j)}\, n_{j,j}^{(j)} - n_{i,j}^{(j)}\, n_{j,k}^{(j)}}{n_{j+1,j+1}^{(j+1)}} \tag{3.2}$$

As proved later, $n_{i,k}^{(j-1)}$ is always a polynomial, so (3.2) shows that only these numerators need be computed to obtain the reduced matrix $\mathbf{B}^{(1)}$. All denominators in $\mathbf{B}^{(j-1)}$ are forced to be numerators of previous pivots, as illustrated in the forthcoming Example 3.3, and incorporated into Algorithm 4.

Algorithm 4. Finding the Determinant of $\mathbf{B}$

set $n_{i,k}^{(N)} = b_{i,k}$, $t=0$, and $n_{N+1,N+1}^{(N+1)} = 1$.
for $j = N$ **to** 2 (step)
  if $n_{j,j}^{(j)} = 0$
    if there is a column $k < j$ such that $n_{j,k}^{(j)} \neq 0$
      interchange columns j and k, and increment t.
    else output $\mathbf{det(B)} = 0$ and stop.
  for $i = j-1$ **to** 1 (row index)
    for $k = j-1$ **to** 1 (column index)

$$n_{i,k}^{(j-1)} = \frac{n_{i,k}^{(j)}\, n_{j,j}^{(j)} - n_{i,j}^{(j)}\, n_{j,k}^{(j)}}{n_{j+1,j+1}^{(j+1)}} \tag{3.2}$$

output $\mathbf{det(B)} = (-1)^t n_{11}^{(1)}$.

Lemma. The output of Algorithm 4 is $\mathbf{det(B)}$.

Proof. If $\mathbf{B}$ is singular, then an all-zero row will occur at some point and the algorithm will output 0. Otherwise, $n_{i,j}^{(j-1)} = 0$ for all integers $i < j$. Thus $\mathbf{B}^{(1)}$ is zero above its main diagonal, so

$$\begin{aligned}
\mathbf{det(B)} &= (-1)^t \, \mathbf{det}(\mathbf{B}^{(1)}) \\
&= (-1)^t \prod_{j=1}^{N} b_{j,j}^{(j)} \\
&= (-1)^t \prod_{j=1}^{N} \frac{n_{j,j}^{(j)}}{n_{j+1,j+1}^{(j+1)}} \\
&= (-1)^t \, n_{1,1}^{(1)}.
\end{aligned}$$

□

Claim. If $b_{i,k}^{(N)}$ is polynomial for $1 \leq i,k \leq N$, then so is every $n_{i,k}^{(j-1)}$ in (3.2).

Proof. (Induction on j). Since $n_{N+1,N+1}^{(N+1)} = 1$ by definition, $n_{i,k}^{(N-1)}$ is polynomial. Supose that for some integer $h < N-1$, $n_{i,k}^{(j)}$ is polynomial for all $h \leq j \leq N-1$ and $i \leq i,k \leq h-1$. Then by the induction hypothesis, $n_{i,k}^{(h)} n_{h,h}^{(h)} - n_{i,h}^{(h)} n_{h,k}^{(h)}$ is a polynomial. Expanding this expression using (3.2) with $j = h$ proves that $n_{h+1,h+1}^{(h+1)}$ is a (polynomial) factor. Therefore, $n_{i,k}^{(h-1)}$ must be polynomial. □

Since $n_{j+1,j+1}^{(j+1)}$ always divides the numerator of (3.2), it should be excluded from the calculation of $n_{i,k}^{(j)}$ during the previous step. However, the resulting error propagates in a manner that appears uncorrectable. This phenomenon is expected because Algorithm 4 implements a recursive decomposition of the equation

$$\mathbf{det(B)} = \sum_{\sigma \in S_N} \mathrm{sign}(\sigma) \; b_{1,\sigma(1)} \; b_{2,\sigma(2)} \cdots b_{N,\sigma(N)}$$

where S_N is the set of all permutations of the integers 1 to N. It is unfortunate that the $n_{i,k}^{(j)}$ cancellation cannot be avoided. However, this problem also occurs in Sylvester's identity for determinants [Gant59].

Example 3.3 The following matrices illustrate the operation of Algorithm 4 on $\mathbf{B}^{(4)} = \mathbf{B}$ with initial entries $n_{i,j}^{(4)} = b_{i,j}^{(4)}$. When $j = N = 4$, $-b_{34}^{(4)}/b_{44}^{(4)}$, $-b_{24}^{(4)}/b_{44}^{(4)}$, and $-b_{14}^{(4)}/b_{44}^{(4)}$ times row 4 are added to rows 3,2, and 1, respectively, so that all entries in column 4 above $b_{44}^{(4)}$ become zero:

$$\mathbf{B}^{(3)} = \begin{bmatrix} \frac{n_{11}^{(3)}}{b_{44}^{(4)}} & \frac{n_{12}^{(3)}}{b_{44}^{(4)}} & \frac{n_{13}^{(3)}}{b_{44}^{(4)}} & 0 \\ \frac{n_{21}^{(3)}}{b_{44}^{(4)}} & \frac{n_{22}^{(3)}}{b_{44}^{(4)}} & \frac{n_{23}^{(3)}}{b_{44}^{(4)}} & 0 \\ \frac{n_{31}^{(3)}}{b_{44}^{(4)}} & \frac{n_{32}^{(3)}}{b_{44}^{(4)}} & \frac{n_{33}^{(3)}}{b_{44}^{(4)}} & 0 \\ b_{41}^{(4)} & b_{42}^{(4)} & b_{43}^{(4)} & b_{44}^{(4)} \end{bmatrix}$$

Then after the step with $j = 3$, during which $-n_{23}^{(3)}/b_{33}^{(3)}$ times row 3 is added to row 2 and $-n_{13}^{(3)}/b_{33}^{(3)}$ times row 3 is added to row 1,

$$\mathbf{B}^{(2)} = \begin{bmatrix} \frac{n_{11}^{(2)}}{b_{33}^{(3)}} & \frac{n_{12}^{(2)}}{b_{33}^{(3)}} & 0 & 0 \\ \frac{n_{21}^{(2)}}{b_{33}^{(3)}} & \frac{n_{22}^{(2)}}{b_{33}^{(3)}} & 0 & 0 \\ \frac{n_{31}^{(3)}}{b_{44}^{(4)}} & \frac{n_{32}^{(3)}}{b_{44}^{(4)}} & \frac{n_{33}^{(3)}}{b_{44}^{(4)}} & 0 \\ b_{41}^{(4)} & b_{42}^{(4)} & b_{43}^{(4)} & b_{44}^{(4)} \end{bmatrix}$$

$\mathbf{det}(\mathbf{B}) = \mathbf{det}(\mathbf{B}^{(3)}) = \mathbf{det}(\mathbf{B}^{(2)})$ because only elementary row operations on $\mathbf{B}$ have been performed. The final step ($j = 2$) produces a $\mathbf{B}^{(1)}$ matrix identical to $\mathbf{B}^{(2)}$ except that $n_{11}^{(1)}/n_{22}^{(2)} \quad 0 \quad 0 \quad 0$ is the first row. Hence, $\mathbf{det}(\mathbf{B}) = \mathbf{det}(\mathbf{B}^{(1)}) = n_{11}^{(1)}$.

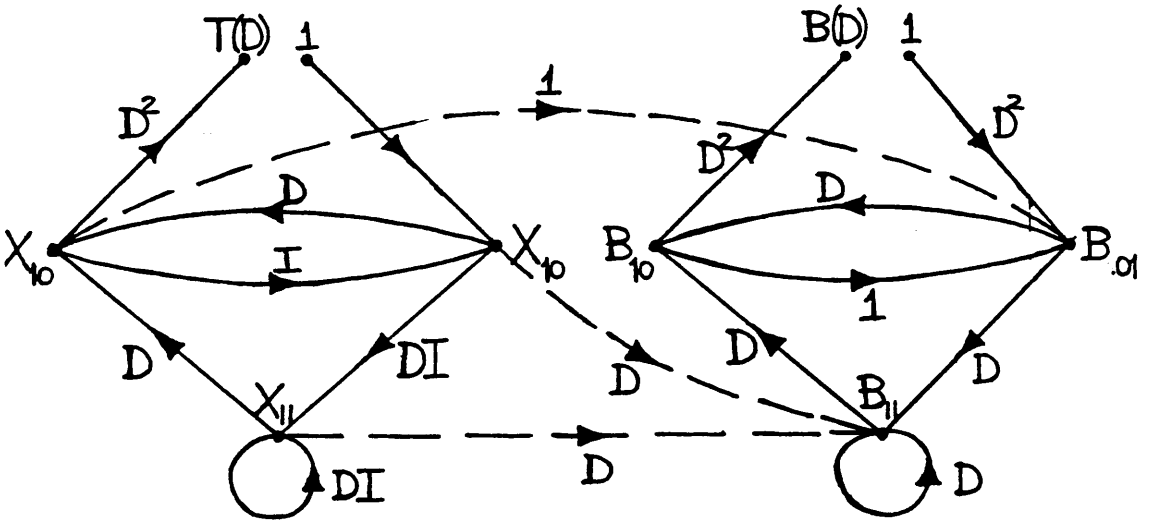
Algorithm 4 was used to compute the determinants of $\mathbf{B}, (\mathbf{B}_{2^{m-1}})$, two 16 by 16 reduced matrices, for the $m = 6$, rate 1/2, NASA standard code. Only a few seconds of computer time were required to calculate $T(D)$. However, the complete path enumerator $T(D, L, I)$ required considerable effort to find, for it contains 1529 numerator and 2799 denominator trivariate terms! A technique in the next section leads to $B(D)$ directly, eliminating the need to obtain the massive generating function $T(D, L, I)$.

3.4 Finding the Bit-Error Generating Function Directly

Computing $T(D, L, I)$ in order to obtain $B(D)$ and $L(D)$ is an expensive and often difficult task. However, a union bound on Viterbi decoder bit error rate (BER) is usually written in terms of the partial derivative of $T(D, L, I)$ with respect to I. On the unquantized, AWGN channel with symbol signal-to-noise ratio E_s/N_0,

$$\text{BER} \leq Q(\sqrt{2d_{\text{free}}E_s/N_0})\, e^{d_{\text{free}}E_s/N_0} \left.\frac{\partial T(D,L,I)}{\partial I}\right|_{L=I=1, D=e^{-E_s/N_0}} \tag{3.3}$$

In this section, a new method is derived for computing $B(D) = \partial T(D, L, I)/\partial I$ at $L = I = 1$, *without finding* $T(D, L, I)$. For any rate k/n encoder, several additional determinants, than those needed to obtain $T(D)$, will be computed to find $B(D)$ or $L(D)$. As explained later, the techniques apply to a more general combinatorial problem of counting paths that have certain properties in a finite directed graph.



$$X_{01} = X_{10} + D^2 \qquad B_{01} = B_{10} + X_{10} + D^2$$

$$X_{10} = DX_{01} + D\,X_{11} \qquad B_{10} = DB_{01} + DB_{11}$$

$$X_{11} = DX_{01} + DX_{11} \qquad B_{11} = DB_{01} + DB_{11} + DX_{01} + DX_{11}$$

Figure 3.2 Connected Encoder State Diagrams.

The key step is to connect an encoder graph with a copy that has different node labels but the same edge labels, as in Figure 3.2. Nodes in the first graph are labelled by path transmissions X_s, while those in the second graph have labels B_s, as defined below. All edges are labelled with D raised to the number of 1's output by the encoder during the transition between the two connected states. Define $B_s(D)$ as the **bit error transmission** from state 0 to state s: the coefficient of D^d in $B_s(D)$ is the total number of 1's in all information sequences which generate weight d trellis paths that go from state 0 to state s via nonzero states. The edges between the two encoder graphs are established by expressing B_s in terms of other path and bit error transmissions as

$$B_s = \sum_{t=0}^{2^m-1} \left[a_{s,t} B_t + i_{s,t} a_{s,t} X_t \right] \tag{3.4}$$

where $a_{s,t}$ is a power of D if there is a transition from state t into state s (and 0 otherwise) while $i_{s,t}$ is the number of 1's input to the encoder for the transition.

Finding $B(D)$ requires first solving the set of equations relating path transmissions X_s, then multiplying these values by labels on the connecting edges, substituting them into the equations (3.3), and finally solving for $B(D)$ by computing one additional determinant. Since all X_s have the common denominator $\mathbf{det}(\mathbf{A})$, this factor is removed when solving for $B(D)$, which has denominator $[\mathbf{det}(\mathbf{A})]^2$. Recall the $2^m \times 2^m$ matrix $\mathbf{A}$ defined in Section 3.2 to represent the edge labels in the modified state diagram of an encoder. Set $L=1$ so that $\mathbf{A} = \mathbf{A}(D, I)$. Using initial, simple row reduction followed by (perhaps several applications of) Cramer's rule and Algorithm 4, the system of algebraic equations

$$\mathbf{A}(D,1) \ [X_{01} \ X_{10} \ \ldots \ X_{2^m-1} \ T(D)]^{\mathrm{T}} \ = \ [a_{0,0} \ \ a_{1,0} \ \ldots \ a_{2^m-1,0}]^{\mathrm{T}}$$

is solved for the path transmissions X_s and code weight enumerator $T(D)$, which all have denominator $\mathbf{det}(\mathbf{A}(D,1))$. This system of equations will be written as $\mathbf{A}(D,1)\ \mathbf{x}^{\mathrm{T}} \ = \ \mathbf{h}(D,I)$. Now define a matrix $\mathbf{E}$ whose entry in the i^{th} row and j^{th} column is

$$e_{i,j} \ = \ \delta_{i,j} \ - \ \left. \frac{\partial \, a_{i,j}(D,I)}{\partial I} \right|_{I=1}$$

where $\delta_{i,j}=1$ if $i=j$ and 0 otherwise. Then solving

$$\mathbf{A}(D,1)\ [B_{01}\ B_{10}\ B_{11}\ B^*]^{\mathrm{T}} \ = \ \mathbf{det}(\mathbf{A}(D,1))\,\mathbf{E}\mathbf{x}^{\mathrm{T}} \ + \ \frac{\partial\,\mathbf{h}(D,I)}{\partial I}\Big|_{I=1}$$

for the numerator of B^* by computing one more determinant, yields $B(D)$ as this polynomial divided by $[\mathbf{det}(\mathbf{A}(D,1))]^2$.

Example 3.4 The connected state diagrams for the encoder in Figure 1.1 are shown in Figure 3.2. Using (3.4), the bit error transmission into state 11 is $B_{11} = DB_{01} + DB_{11} + DX_{01} + DX_{11}$ because all simple paths ending in states 01 and 11 are extended, by one trellis branch with label 01 resulting from a 1 input to the encoder, into state 11. Using Cramer's rule on the left set of equations yields

$$X_{01} \ = \ \frac{D^2 - D^3}{1-2D} \ , \qquad X_{10} \ = \ X_{11} \ = \ \frac{D^3}{1-2D} \ .$$

Computing one more determinant produces $B(D) \ = \ D^5/(1-2D)^2$. □

Example 3.5 By deleting every third bit output by the encoder in Figure 1.1, a rate 2/3, "punctured" encoder is created with a modified state diagram (Figure 3.3) which leads to the system of equations

$$\mathbf{A}(D,I)\ [X_{01}\ X_{10}\ X_{11}\ T(D,I)]^{\mathrm{T}} \ = \ \mathbf{h}(D,I)$$

$$\begin{bmatrix} 1-DI & -D^3I & -DI & 0 \\ -DI & 1-DI & -DI & 0 \\ -D^2I^2 & -I^2 & 1-D^2I^2 & 0 \\ -D^2 & -D^2 & -1 & 1 \end{bmatrix} \begin{bmatrix} X_{01} \\ X_{10} \\ X_{11} \\ T(D,I) \end{bmatrix} = \begin{bmatrix} DI \\ D^3I \\ D^2I^2 \\ 0 \end{bmatrix}$$

Solving $\mathbf{A}(D,1)\mathbf{x}^{\mathrm{T}} = \mathbf{h}(D,1)$ using Cramer's rule and Algorithm 4 yields

$$\begin{aligned} \mathbf{det}(\mathbf{A}(D,1)) \ &= 1-3D+D^3-D^4 \\ X_{01} \ &= (D-D^2+D^4+2D^6-D^8)/\mathbf{det}(\mathbf{A}(D,1)) \\ X_{10} \ &= (D^2+2D^3-D^4-D^5)/\mathbf{det}(\mathbf{A}(D,1)) \\ X_{11} \ &= (2D^2-D^4-D^6+D^8)/\mathbf{det}(\mathbf{A}(D,1)) \end{aligned}$$

$$\mathbf{A}(D,1)\ [B_{01}\ B_{10}\ B_{11}\ B^*] = \mathbf{det}(\mathbf{A}(D,1))\,\mathbf{E}\mathbf{x}^{\mathrm{T}} \ + \ [D\ D^3\ \ 2D^2\ 0]^{\mathrm{T}}$$

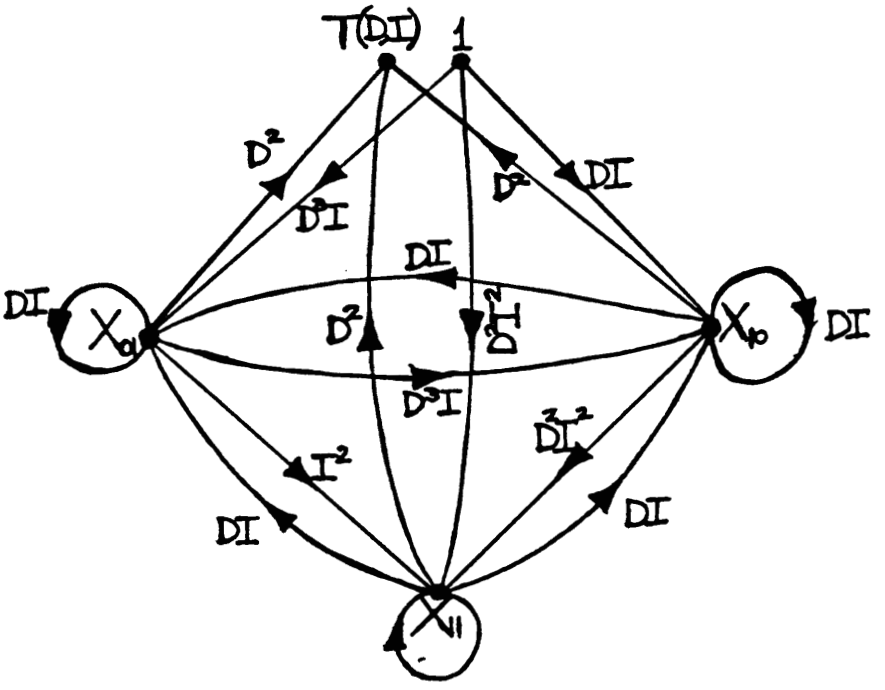


Figure 3.3 Modified State Diagram for a Punctured Encoder.

is then solved for the numerator of B^*, which leads to

$$B(D) = \frac{D^3 + 4D^4 + 3D^5 - 6D^6 - 2D^7 + 4D^8}{(1 - 3D + D^3 - D^4)^2}$$

This method of finding $B(D)$ directly can be applied to the general problem of enumerating paths between any two nodes in a finite directed graph. Edges are labelled with D raised to the "weight" associated with the edge and I raised to some "cost" of the edge. The weight/cost of a path are the sum of its edge weights/costs. Analogous to $T(D)$, a generating function may be obtained which enumerates paths by weights or which contains the cost of all paths having a given weight (like $B(D)$). Thus, the linear algebraic procedures developed in this chapter, for finding generating functions associated with convolutional codes, also work for transfer functions of circuits or networks. Generating functions [Zhav87] for many rate $(n-1)/n$ trellis codes may be obtained using these methods.

3.5 Path Distance and Bit Error Coefficient Approximations

The 76 poles of the NASA code weight enumerator $T(D)$ are plotted on the complex plane along with the unit circle for reference (Figure 3.4). Poles occur in complex conjugate pairs and in positive-negative pairs because $T(D)$ has real coefficients and only even powers of D. The 6 poles with smallest magnitudes (large points) are used to approximate the partial fraction expansion of $T(D)$ as

$$\begin{aligned} T(D) &\approx D^{10}\left[\frac{r_1}{1-\alpha_d D}+\frac{r_1}{1+\alpha_d D}+\frac{r_2}{1-\alpha_b}+\frac{r_2}{1+\alpha_b D}+\frac{r_2^*}{1-\alpha_b^* D}+\frac{r_2^*}{1+\alpha_b^* D}\right] \\ &= D^{10}\sum_{k=0}^{\infty}\left[r_1(\alpha_d)^k+r_1(-\alpha_d)^k+r_2(\alpha_b)^k+r_2(-\alpha_b)^k+r_2^*(\alpha_b^*)^k+r_2^*(-\alpha_b^*)^k\right]D^k \\ &= \sum_{k=0}^{\infty}\left[2r_1(\alpha_d)^{2k}+4\mathrm{Re}\{r_2(\alpha_b)^{2k}\}\right]D^{2k+10} \end{aligned}$$

where * means complex conjugate and $\alpha_d = 2.3876225$ is the reciprocal of the positive, real pole of least magnitude in $T(D)$. Also, $\alpha_b = 1.657193e^{-0.983418\sqrt{-1}}$ is the pole in the upper right quadrant with next smallest magnitude. The residues are

$$r_1 = \frac{-P(\alpha_d^{-1})}{\alpha_d^{-1}Q'(\alpha_d^{-1})} = 3.4103 \quad \text{and} \quad r_2 = \frac{-P(\alpha_b^{-1})}{\alpha_b^{-1}Q'(\alpha_b^{-1})} = 1.075e^{-0.3103\sqrt{-1}}$$

where $D^{10}P(D)/Q(D) = T(D)$ and $Q'(D)$ is the derivative of $Q(D)$. The other poles in Figure 3.3 are ignored because their magnitudes are greater than 0.8 and the corresponding residues have very small magnitudes — less than 0.07.

Define $a(d), i(d)$, and $\ell(d)$ as the coefficients of D^d in $T(D), L(D)$, and $B(D)$ respectively. These generating functions are listed in Section 3.7 at the end of this chapter. Quantities $\ell(d)$ and $i(d)$ are the total length in trellis branches and total number of 1's input to the encoder for all weight d fundamental paths. The above approximation for $T(D)$ leads to

$$\begin{aligned} a(2k+10) &\approx 2r_1(\alpha_d)^{2k}+4\mathrm{Re}\{r_2(\alpha_b)^{2k}\} \\ &= 6.82(2.3876225)^{2k}+4.25(1.657193)^{2k}\cos(0.310-1.967k) \end{aligned}$$

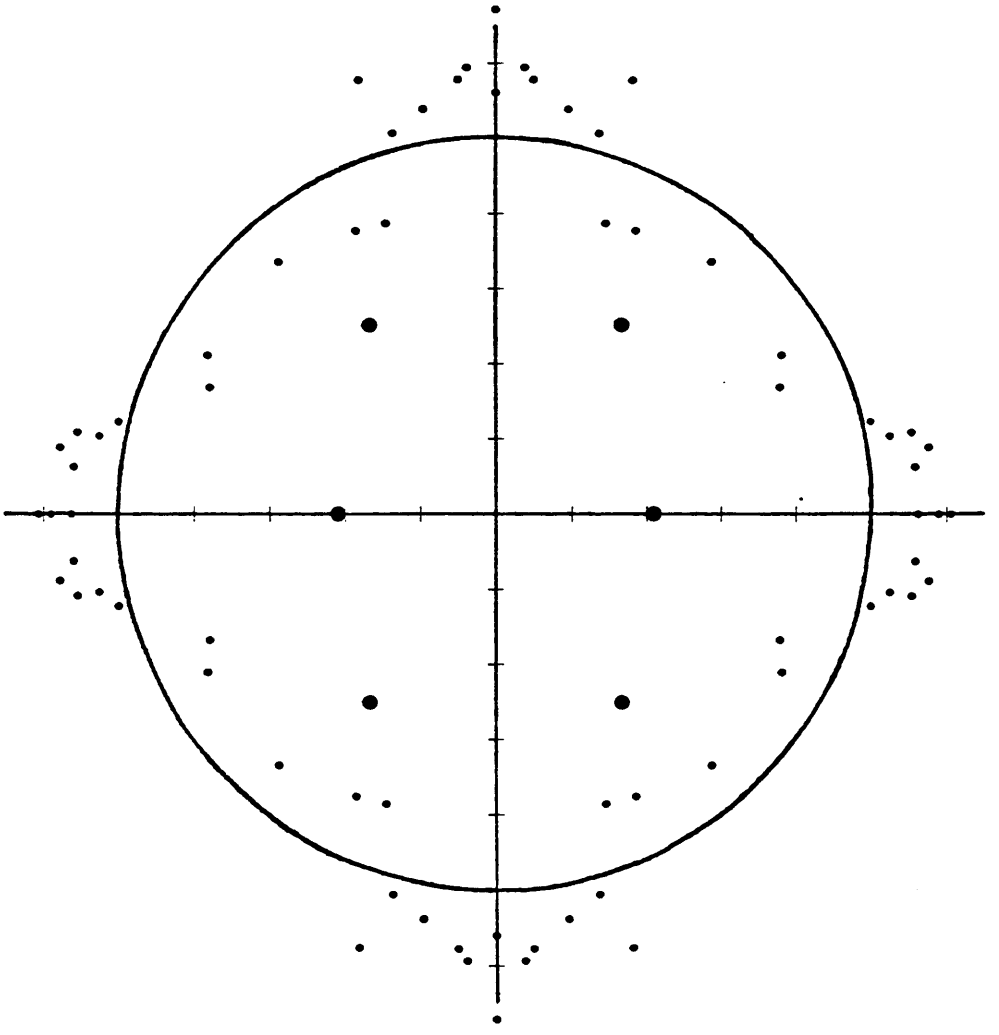


Figure 3.4 Poles of $T(D)$ for the NASA Code.

Similarly, the terms corresponding to the 6 lowest-magnitude poles of $T(D)$ in the partial fraction expansions of $L(D)$ and $B(D)$ lead to

$$\begin{aligned}
\ell(2k+10) &\approx (77.725+22.625k)(2.3876225)^{2k} \\
&\quad + 39.3(1.657193)^{2k}\cos(0.485-1.967k) \\
&\quad + (2k+1)7.2676(1.657193)^{2k}\cos(0.383-1.967k) \\
i(2k+10) &\approx (24.474+12.018k)(2.3876225)^{2k} \\
&\quad + 9.942(1.657193)^{2k}\cos(0.575-1.967k) \\
&\quad + (2k+1)2.8723(1.657193)^{2k}\cos(0.366-1.967k)
\end{aligned}$$

which have a relative error < 0.0001 for $k > 4$. The nearest integer to an approxi-

mation above differs from the true value by 1 or 2 when $k=0,1,2$, or 3.

Since the denominator of $B(D)$ is the square of that for $T(D)$, the union bound on bit error rate diverges when D is equal to z, the smallest magnitude of any pole in $T(D)$. The union bound diverges at $E_b/N_0 = -(n/k)\ln z$, which is 2.4 dB for the NASA code.

The poles of $T(D)$ for a rate $1/3$, $m=5$ code are shown in Figure 3.5 where there is no obvious group of poles which have small magnitudes. Nonetheless, under these adverse circumstances, good analytic approximations to $a(d), \ell(d)$, and $i(d)$ were obtained using only the 8 groups of poles having the lowest magnitudes, which are indicated by large points in Figure 3.5. For many codes, the roots of $T(D)$ were plotted, and typically only a few poles were required to closely approximate $T(D)$. Apparently, poles of small magnitude have residues which are 10 or more times larger than residues of poles which are inside the unit circle but farther removed from the origin.

3.6 An Algorithm for Path Generating Function Coefficients

Finding the complete path weight enumerator of $m>8$ codes currently seems infeasible. In these cases, a trellis searching algorithm easily produces the first twenty nonzero coefficients of the generating functions used for error bounds [Roan89]. An efficient but intricate algorithm for counting the number of paths at various distances employs a limited tree search by taking advantage of the distance structure of a code [Ced89]. However, the algorithm's efficiency has not been determined when $k>1$. Another approach involves using a bidirectional stack algorithm [Roan89], but additional computation is needed to ensure correct values of $a(d)$ or $\ell(d)$.

Viterbi's algorithm, with survivors replaced by vectors of integers that count paths, lengths, or bits, is used on a noiseless channel to compute $a(d), \ell(d)$, and $i(d)$ values. This method has several advantages over the two mentioned above: it is simple and easily programed on a computer (see Algorithm 5); it always produces provably correct results using a minimum number of "steps", each one corresponding

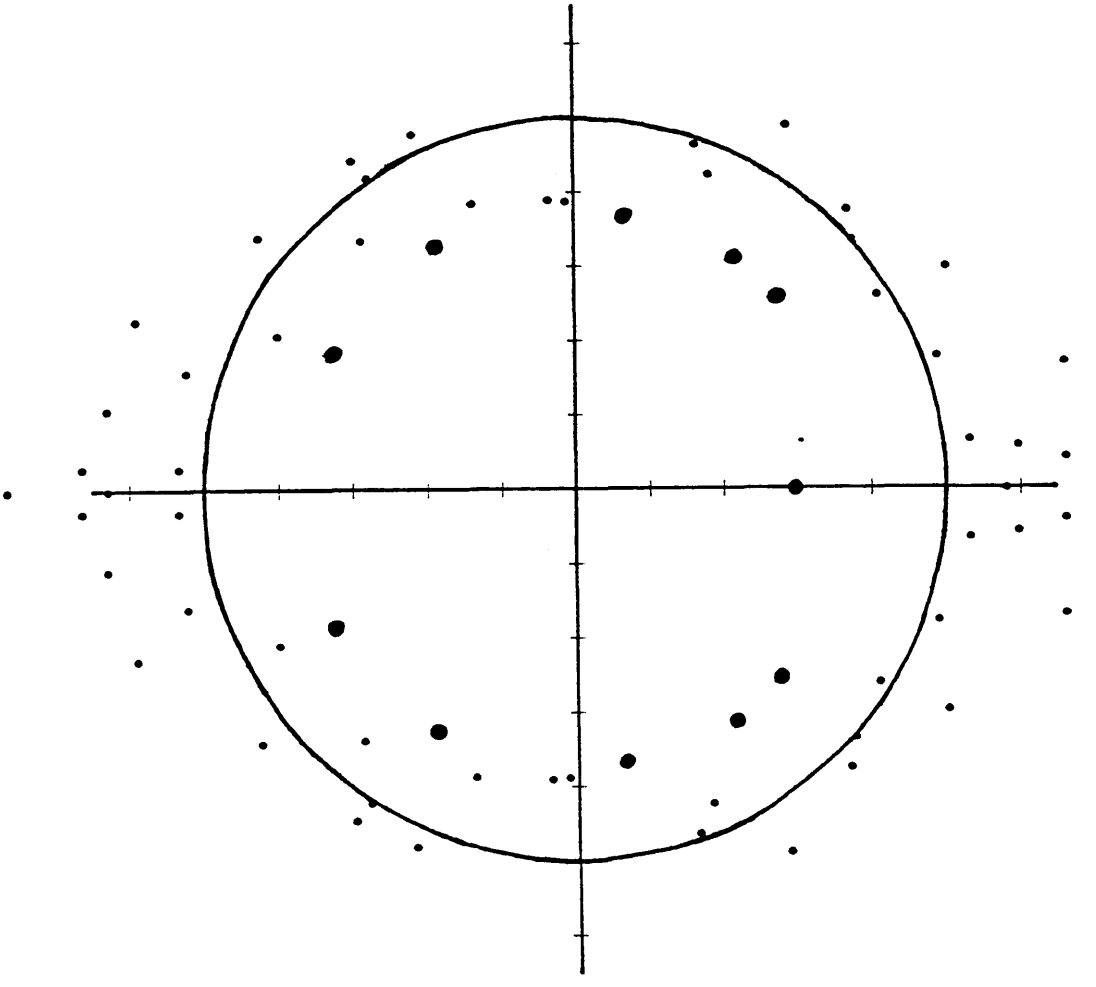


Figure 3.5 Poles of $T(D)$ for a (3,1,5) Code.

to a trellis level; Berlekamp's tangential union bound, which is adapted in the next chapter to obtain close approximations to error probabilities at low E_b/N_0, requires the step-by-step values computed by the algorithm, as does an upper bound (in Section 4.4) which incorporates the effect of finite survivor truncation. Rate $1/n$ codes are treated first to simplify the discussion. Define $\text{out}_0[s]$ and $\text{out}_1[s]$ as the number of ones that the encoder outputs going from state $s0 = \lfloor s/2 \rfloor$ and $s1 = s0 + 2^{m-1}$ into state s. Analogous to a state metric, the entry in row $s > 0$ and column 0 of a matrix W, referred to as $W[s][0]$, will contain the *least weight* of any simple path with length $\leq T$ trellis branches. For $t = 1$ to coeffs (a parameter described later), $W[s][t]$ will be the *number* of simple paths of weight $W[s][0] + t - 1$

and length $\leq T$ branches into state $s > 0$. For state 0, $W[0][t]$ is always kept at 0, except $W[0][1] = 1$. The entries in a second matrix B count either the total number of ones input to the encoder (when the variable len=0) or the total length in trellis branches (when len=1), of all simple paths having length $\leq T$ (again $B[0][t] = 0$ always). These matrices are obtained for successive values of T starting with 1, by extending one branch length at a time (an algorithm "step"), the code trellis starting from state 0 only. Thus the longest path length (T) explored by the algorithm equals the number of "steps" executed. The algorithm terminates after step T^* when W has reached values that will never change, which also forces B to remain constant. At this point, since $W[s][0]$ is the least weight of any simple path into state $s > 0$, $d_{\text{free}} = W[2^{m-1}][0] + \text{out}_1[0]$. Also,

$$W[2^{m-1}][d - d_{\text{free}} + 1] = a(d)$$

$$B[2^{m-1}][d - d_{\text{free}} + 1] = i(d) \qquad \text{if len} = 0$$

for $d = d_{\text{free}}$ to $d_{\text{free}} + \text{coeffs} - 1$. If len $= 1$, then

$$\ell(d) = B[2^{m-1}][d - d_{\text{free}} + 1] + W[2^{m-1}][d - d_{\text{free}} + 1].$$

Algorithm 5 incorporates the above procedure in a C language format. Matrices P and A store *previous* W and B entries corresponding to simple paths of length $\leq T$ which are used to compute new W and B matrices for length $\leq T+1$ simple paths. When change remains 0 after step T^*, W (and thus B) will never change because $W[s][t] = P[s][t]$ for all s and t. $P[s][0]$ is initialized to 999 for $s > 0$, $P[0][1] = 1$, and all other array values are initialized to 0. If any second array index $t +$ offset is < 1 in the $W[s][t]$ and $B[s][t]$ instructions, the array referenced is simply ignored.

The algorithm requires storage for $4 \cdot 2^m(\text{coeffs} + 2)$ integers and the amount of work per step is proportional to this number. The number of steps executed, T^* equals the length, in trellis branches, of the longest fundamental path(s) having weight $d_{\text{free}} + \text{coeffs} - 1$. Algorithm 5 is useful because the number of (and total 1's producing) trellis paths from state 0 having length $\leq T$ branches are available

Algorithm 5. Counting Paths

```
do {
    for (s = 1 to 2^m − 1) {
        s0 = ⌊s/2⌋;  s1 = s0 + 2^(m−1);
        bit = len + (1−len) * (s mod 2) ;
        W[s][0] = min (P[s0][0] + out_0[s], P[s1][0] + out_1[s]);
        offset0 = W[s][0] − P[s0][0] − out_0[s];
        offset1 = W[s][0] − P[s1][0] − out_1[s];
        for (t = 1 to coeffs) {
            W[s][t] = P[s0][t + offset0] + P[s1][t + offset1];
            B[s][t] = A[s0][t + offset0] +
                      A[s1][t + offset1] + bit * W[s][t]; }
    } change = 0;
    for (s = 1 to 2^m − 1)
        for (t = 0 to coeffs)
            if (P[s][t] ≠ W[s][t]) { change = 1;
                P[s][t] = W[s][t];  A[s][t] = B[s][t]; }
} while (change ≠ 0);
```

for $T = m+1, m+2, \ldots$, the number of steps currently completed. These values are required to evaluate two new error bounds derived in the next chapter. The parameter coeffs should be set equal to $\lceil 10n/k \rceil$ because using this many nonzero terms in the union bounds gives results with 2 significant digits of precision when the bounds are tight enough to be useful. Setting coeffs = 0 in Algorithm 5 and ignoring offset0, offset1, A, and B yields d_{free}, which is 35 for the Galileo code [Dol88], obtained with about 32000 bytes of storage and 2 seconds of computer time. For rate k/n codes with $k > 1$, $\text{out}_j[s]$ is defined as the number of 1's output by the encoder as it enters state s with input $j \in [0 \ldots 2^k - 1]$. New $W[s][t]$ and

$B[s][t]$ values are computed using 2^{k_1} entries from each of the P and A matrices, where $k_1 \leq k$ is the number of encoder inputs which are stored in memory cells.

For codes with $2^k << 2^m$ (such as rate $1/n$ with $m > 3$), Algorithm 5 may be improved by using in-place computation of $W[s][t]$ and $B[s][t]$ values which requires looping through groups of 2^k states called butterflies instead of individual states [Onys90]. The storage required decreases by almost one-half by eliminating the "double-buffering" matrices P and A. This improved algorithm produced the first $10n = 40$ nonzero coefficients of $T(D), L(D)$ and $B(D)$ (Table 3.1) for the Galileo code, whose $m = 14$ encoder has 16384 states. Note that the path lengths in Table 3.1 do not include the 14 branches generated by input zeros that return the encoder to the all-zero state. These numbers required only a few minutes of CPU and 7 Mbytes of storage when 4 bytes were used for each integer. The memory used decreases when each integer in the matrices is stored in smallest number of bytes needed (1 for $W[s][0]$ to $W[s][15]$ for the Galileo code).

Table 3.1 Galileo Code Profiles

distance d	fundamental paths $a(d)$	total bit errors $i(d)$	total path lengths $\ell(d) - 14 \cdot a(d)$
35	2	6	7
36	1	2	5
37	4	16	22
38	2	8	12
39	3	11	17
40	5	20	28
41	6	24	46
42	17	76	122
43	24	126	214
44	29	180	285
45	39	255	438
46	66	416	721
47	94	628	1071
48	121	850	1478
49	175	1313	2260
50	277	2086	3643
51	415	3361	5855
52	639	5304	9388
53	934	8010	14161
54	1273	11452	20271
55	1906	17550	31381
56	2878	27332	49172
57	4054	39750	71705
58	5978	60788	109808
59	8864	92738	167861
60	12966	139556	253134
61	18984	210112	383008
62	27949	317798	581467
63	41092	479512	878975
64	60126	720858	1323152
65	87799	1080933	1987235
66	128712	1622990	2992979
67	189880	2451782	4530508
68	278589	3682496	6817868
69	408780	5534126	10261968
70	598271	8283100	15386816
71	875283	12380669	23050515
72	1286052	18596544	34662286
73	1888299	27885609	52045238
74	2768375	41727376	78013493
75	4057688	62421220	116865844
76	5953416	93419654	175122289
77	8732134	139709066	262220198
78	12809968	208928290	392628663
79	18786484	312181796	587384902
80	27548175	466271448	878292728
81	40412499	696477455	1313354906
82	59269748	1039725314	1962719710

3.7 Generating Functions for the (7,1/2) NASA Code

Using the determinant Algorithm 4, the complete path enumerator $T(D, L, I)$ for the memory 6, rate 1/2 NASA code was computed and it contains 1529 numerator and 2799 denominator trivariate terms. The code's weight enumerator $T(D) = T(D, 1, 1)$ is

$$11D^{10} - 6D^{12} - 25D^{14} + D^{16} + 93D^{18} - 15D^{20} - 176D^{22} - 76D^{24} + 243D^{26} + 417D^{28} - 228D^{30} - 1156D^{32} - 49D^{34} + 2795D^{36} + 611D^{38} - 5841D^{40} - 1094D^{42} + 9575D^{44} + 1097D^{46} - 11900D^{48} - 678D^{50} + 11218D^{52} + 235D^{54} - 8068D^{56} - 18D^{58} + 4429D^{60} - 20D^{62} - 1838D^{64} + 8D^{66} + 562D^{68} - D^{70} - 120D^{72} + 16D^{76} - D^{80}$$

$$1 - 4D^2 - 6D^4 - 30D^6 + 40D^8 + 85D^{10} - 81D^{12} - 345D^{14} + 262D^{16} + 844D^{18} - 403D^{20} - 1601D^{22} + 267D^{24} + 2509D^{26} + 389D^{28} - 3064D^{30} - 2751D^{32} + 2807D^{34} + 8344D^{36} - 1960D^{38} - 16133D^{40} + 1184D^{42} + 21746D^{44} - 782D^{46} - 21403D^{48} + 561D^{50} + 15763D^{52} - 331D^{54} - 8766D^{56} + 131D^{58} + 3662D^{60} - 30D^{62} - 1123D^{64} + 3D^{66} + 240D^{68} - 32D^{72} + 2D^{76}$$

$$= 11D^{10} + 38D^{12} + 193D^{14} + 1331D^{16} + 7275D^{18} + 40406D^{20} + \cdots$$

The path-length and bit-error generating functions $L(D)$ and $B(D)$ both have denominators equal to the square of $T(D)$'s denominator above. Their numerators are respectively

$$121D^{10} - 387D^{12} - 706D^{14} + 1460D^{16} + 3970D^{18} - 6157D^{20} - 11643D^{22} + 8725D^{24} + 28677D^{26} + 12195D^{28} + 88D^{30} - 170654D^{32} - 306124D^{34} + 817895D^{36} + 1637616D^{38} - 2879440D^{40} - 6106837D^{42} + 8568521D^{44} + 18636083D^{46} - 22431469D^{48} - 48921504D^{50} + 52678351D^{52} + 113105887D^{54} - 112260733D^{56} - 232580537D^{58} + 217337170D^{60} + 426400859D^{62} - 379787502D^{64} - 696667758D^{66} + 592954735D^{68} + 1013294336D^{70} - 815739185D^{72} - 1311124721D^{74} + 968225450D^{76} + 1509511967D^{78} - 955561827D^{80} - 1548537967D^{82} + 721812022D^{84} + 1419185285D^{86} - 302418615D^{88} - 1166004400D^{90} - 173248817D^{92} + 861869027D^{94} + 545751792D^{96} - 574515412D^{98} - 713158178D^{100} + 345328990D^{102} + 676371119D^{104} - 186475599D^{106} - 515274530D^{108} + 89761092D^{110} + 326707300D^{112} - 38067910D^{114} - 174942675D^{116} + 14010022D^{118} + 79516060D^{120} - 4391424D^{122} - 30654965D^{124} + 1145504D^{126} + 9969013D^{128} - 241287D^{130} - 2706667D^{132} + 39344D^{134} + 603670D^{136} - 4651D^{138} - 107908D^{140} + 354D^{142} + 14883D^{144} - 13D^{146} - 1488D^{148} + 96D^{152} - 3D^{156}$$

and

$$36D^{10} - 77D^{12} - 140D^{14} + 813D^{16} + 269D^{18} - 4414D^{20} + 321D^{22} + 14884D^{24} - 5273D^{26} - 40509D^{28} + 39344D^{30} + 83884D^{32} - 177469D^{34} - 111029D^{36} + 608702D^{38} - 29527D^{40} - 1820723D^{42} + 817086D^{44} + 4951082D^{46} - 3436675D^{48} - 12279246D^{50} + 10300306D^{52} + 27735007D^{54} - 25648025D^{56} - 56773811D^{58} + 55659125D^{60} + 104376199D^{62} - 106695512D^{64} - 170819460D^{66} + 180836818D^{68} + 247565043D^{70} - 270555690D^{72} - 317381295D^{74} + 356994415D^{76} + 360595622D^{78} - 415401723D^{80} - 364292177D^{82} + 426295756D^{84} + 328382391D^{86} - 385686727D^{88} - 264812337D^{90} + 307287819D^{92} + 191225378D^{94} - 215144035D^{96} - 123515898D^{98} + 131946573D^{100} + 71124860D^{102} - 70570661D^{104} - 36310569D^{106} + 32722089D^{108} + 16308558D^{110} - 13052172D^{112} - 6380604D^{114} + 4433332D^{116} + 2147565D^{118} - 1265046D^{120} - 612040D^{122} + 297721D^{124} + 144665D^{126} - 56305D^{128} - 27569D^{130} + 8232D^{132} + 4066D^{134} - 874D^{136} - 435D^{138} + 60D^{140} + 30D^{142} - 2D^{144} - D^{146}$$

References

[Ced89] M. Cedervall and R. Johannesson,"A Fast Algorithm for Computing Distance Spectrum of Convolutional Codes," *IEEE Trans. Inform. Theory*, IT-35, pp. 1146–1159, November 1989.

[Dol88] S. Dolinar,"A New Code for Galileo," *TDA Progress Report 42-93*, Jet Propulsion Laboratory, Pasadena, pp. 83–96, May 15, 1988.

[Gant59] J. K. Gantmacher, *The Theory of Matrices.* New York: Chelsea, 1959.

[Lin83] S. Lin and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: , 1983.

[Onys90] I. M. Onyszchuk,"Finding the Complete Path and Weight Enumerators of Convolutional Codes," *TDA Progress Report 42-100*, Jet Propulsion Laboratory, Pasadena, pp. 203–213, February 15, 1990.

[Roan89] M. Rouanne and D.J. Costello, Jr., "An Algorithm for Computing the Distance Spectrum of Trellis Codes," *IEEE J. Selected Areas of Commun.*, SAC-21, August 1989.

[Vit71] A. J. Viterbi, "Convolutional Codes and their Performance in Communication Systems," *IEEE Trans. Commun.*, COM-19, pp. 751–772, October 1971.

[Vit79] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding.* New York: McGraw-Hill, 1979.

[Zhav87] E. Zehavi and J. K. Wolf, "On the Performance Evaluation of Trellis Codes," *IEEE Trans. Inform. Theory*, IT-33, pp. 196–202, March 1987.

Chapter 4

Symbol and Bit Error Rate Bounds

The appropriate performance measure of a convolutional code, when it is concatenated with an outer block code having symbols from GF(2^b), is the probability that a b-bit symbol is decoded incorrectly, denoted SER_b. By considering all ways in which error events may corrupt a block of b consecutive decoded bits, the union bound on bit error rate (BER) is generalized in the next section to one for SER_b. Then, new lower bounds on the first event and bit error probabilities of a maximum-likelihood decoder are derived.

Berlekamp's tangential union bound for maximum-likelihood block decoder word error rate on the AWGN channel is modified for convolutional codes to obtain estimates of Viterbi decoder error rates. These approximations are quite good at low E_b/N_0, where the union bounds are either very loose or completely useless because they have diverged.

Lastly, the effect of truncating survivors in the Viterbi algorithm is studied. A union bound is derived for bit error rate, taking into account the finite length of survivors in practical decoders. This result is used to estimate the least truncation length required for a given decoded bit error rate.

4.1 A Symbol Error Bound

Recall from Section 1.1 that the all-zero path may be assumed transmitted when error probabilities are derived for maximum-likelihood decoding on a binary-input, output-symmetric, discrete memoryless channel, such as the AWGN channel. Thus, every error event is a fundamental trellis path. For each received n-vector, a (theoretical) Viterbi decoder outputs k bits corresponding to one branch T levels back in the trellis, where T is a fixed, large integer. At each trellis level, with probability P_d, the decoder selects a fundamental path $\mathbf{c}$ of weight d instead of the

all-zero path as the survivor into state 0, eventually causing a decoded bit error for each 1 in the information sequence $\mathbf{u}$ that generates $\mathbf{c}$. Hence, for each block of n received channel symbols, corresponding to k information bits, the expected number of bit errors caused by all paths $\mathbf{c}$ is at most $\sum i(d)P_d$ (the "union" bound [Vit71]). Recall that $i(d)$ is the total number of 1's in all $\mathbf{u}$'s which produce paths $\mathbf{c}$ of weight d. Therefore, a Viterbi decoder's bit error rate is upper bounded as

$$\text{BER} \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} i(d)P_d \,. \tag{4.1}$$

The approach in the following alternative derivation of (4.1) will be used again later. Consider the Viterbi decoding of a fixed block $\mathbf{v}$ of k information bits. Suppose that a particular fundamental path $\mathbf{c}$ of weight d is selected, at some trellis level, as the survivor into state zero instead of the all-zero path. This event occurs with probability P_d and leads to a number of bit errors equal to the number of 1's in $\mathbf{u}$ which overlap $\mathbf{v}$. By considering all possible ways that $\mathbf{c}$ could overlap $\mathbf{v}$, each one being mutually exclusive and occurring with probability P_d, the average number of bit errors in $\mathbf{v}$, caused by $\mathbf{c}$, is at most P_d times the number of 1's in $\mathbf{u}$. Now (4.1) follows by summing this quantity over all paths $\mathbf{c}$ because each fundamental path is a possible error event.

In order to generalize (4.1) to blocks of b consecutive decoded bits, define li for a fundamental path $\mathbf{c}$ as the number of consecutive *information bits*, counted from the first 1 entering the encoder to the last 1 in the corresponding information sequence $\mathbf{u}$. Since m zeros (and possibly more if $k > 1$) are required to return an encoder to the all-zero state, li is at least m less than k times the number (ℓ) of trellis branches in $\mathbf{c}$. Let $li(d)$ be the sum of li over all weight d fundamental paths and recall that $a(d)$ and $\ell(d)$ denote the number and total length in trellis branches of these paths. Consider the Viterbi decoding of a fixed symbol $\mathbf{v}$ consisting of b information bits and assume with probability $1/k$ that $j \in [1, \ldots, k]$ bits of a k-bit trellis branch overlap $\mathbf{v}$ from the left-hand side. Under this "random alignment" assumption, which is always valid when k and b are relatively prime, there are

$\left\lceil \frac{li+b-j}{k} \right\rceil$ mutually exclusive ways, each occurring with probability P_d, that a weight d fundamental path $\mathbf{c}$ could be chosen as the surivivor path into state 0 at some trellis level, such that $\mathbf{v}$ is overlapped by the corresponding information vector $\mathbf{u}$ (which has li bits including the starting and ending 1's). Therefore, the average fraction of decoded symbols which are incorrect because of an error event $\mathbf{c}$ is upper bounded by

$$\sum_{j=1}^{k} \frac{1}{k} \left\lceil \frac{li+b-j}{k} \right\rceil P_d \;=\; \frac{li+b-1}{k} P_d \,,$$

assuming that every possible overlap by $\mathbf{u}$ over $\mathbf{v}$ contains at least one 1. As justified later, this assumption does not significantly contribute to the looseness of the following union bound,

$$\begin{aligned} \mathrm{SER}_b \;&\leq\; \frac{1}{k} \sum_{d=d_{\mathrm{free}}}^{\infty} \sum_{li=1}^{\infty} (li+b-1)\; a(d,li)\; P_d \\ &=\; \frac{1}{k} \sum_{d=d_{\mathrm{free}}}^{\infty} \Big\{ [b-1]a(d) + li(d) \Big\}\; P_d \end{aligned} \tag{4.2}$$

where $a(d,li)$ is the number of fundamental trellis paths having weight d and length li information bits. Notice that for $b=1$, (4.2) is looser than (4.1) because the li information bits of most fundamental paths contain zeros which are not taken into account in (4.2). If b divides k and symbol boundaries are aligned with trellis branches, then there are only $\frac{li+b-k}{k}$ different ways that the li information bits of $\mathbf{c}$ could overlap $\mathbf{v}$, so in this special case, the 1 in (4.2) should be replaced by k.

In general, $li(d) \leq \ell(d)$, but for rate $1/n$, memory m, feedforward convolutional encoders and their punctured variants, $li(d) = \ell(d) - m \cdot a(d)$ because a fundamental trellis path with ℓ branches has length $li = \ell - m$ information bits. For these encoders, (4.2) becomes

$$\mathrm{SER}_b \;\leq\; \frac{1}{k} \sum_{d=d_{\mathrm{free}}}^{\infty} \Big\{ [b-1-m]a(d) + l(d) \Big\}\; P_d \,. \tag{4.3}$$

Since $li(d) \approx \ell(d)$ for encoders with feedback, they perform worse in concatenated systems, according to the above bound at high E_b/N_0 and simulations at $E_b/N_0 = 2$ to 4 dB of the NASA code.

For a rate k/n Viterbi decoder on the unquantized AWGN channel, $P_d = Q(\sqrt{2d(k/n)E_b/N_0})$ where $Q(x) = \int_x^\infty e^{-y^2/2}\frac{dy}{\sqrt{2\pi}}$ [Vit79]. Now from [Abr64],

$$Q(x) \sim \frac{e^{-x^2/2}}{x\sqrt{2\pi}}$$

as x goes to infinity, where $\sim$ means that the ratio goes to 1. Therefore, at high bit signal-to-noise ratio E_b/N_0, the right side of (4.2) is asymptotically equal to

$$k^{-1}\left[(b-1)a(d_{\text{free}}) + li(d_{\text{free}})\right] Q(\sqrt{2d_{\text{free}}RE_b/N_0})$$

which upper-bounds the probability that a b-bit symbol is corrupted by an error event (fundamental path) of weight d_{free}. For future use, define $s_b(d) = (b-1)a(d) + li(d)$. Notice that the SER_b bounds diverge at $E_b/N_0 = 10\log_{10}(-n/k \ln z)$ dB, just as the BER union bound does, where z is the magnitude of a pole having least magnitude in the code's path weight enumerator $T(D)$.

Example 4.1 For the code (CC1) with $g_{11}(x) = 1 + x + x^2$ and $g_{12}(x) = 1 + x^2$,

$$\text{SER}_4 \leq \sum_{d=5}^{\infty} [a(d) + \ell(d)]\, P_d = \sum_{d=5}^{\infty} 2^{d-6}(3d-7)P_d\ .$$

Since any error event corresponds to a fundamental path, all symbols overlapped by the corresponding information sequence $\mathbf{u}$ will be decoded incorrectly because $\mathbf{u}$ cannot contain $m=2$ consecutive zeros in this example. Therefore, this bound is as tight as the bit error union bound, as illustrated in Figure 4.1. □

Note. *All $li+b-1$ possible ways that a length li vector $\mathbf{u}$ could overlap a b-bit decoded symbol will cause an error when the corresponding fundamental path $\mathbf{c}$ is chosen instead of $\mathbf{0}$, unless $\mathbf{u}$ contains at least b consecutive zeros, which requires that $li \geq b+2$ and $b \leq m-1$.*

Since vectors $\mathbf{u}$ of length $li \geq b+2$ usually produce fundamental paths of weight greater than d_{free}, the decoder will choose these paths over $\mathbf{0}$ with very low probability as compared to fundamental paths having weight near d_{free} and whose

u's are short (i.e., 1,11,101,111). Furthermore, each sequence of b consecutive zeros in **u** will result in only one less way that **u** can overlap a symbol **v** and cause it to be decoded incorrectly. Thus for $b > 1$, the SER_b bounds (4.2) or (4.3) will be nearly as tight as the BER union bound. If $b \leq m-1$, define $s_b^-(d_{\mathrm{free}})$ as $s_b(d_{\mathrm{free}})$ minus the number of groups of b consecutive zeros in all information vectors **u** which produce fundamental paths of weight d_{free}. For $b \geq m$, set $s_b^-(d_{\mathrm{free}}) = s_b(d_{\mathrm{free}})$ and observe that $s_1^-(d_{\mathrm{free}}) = i(d_{\mathrm{free}})$.

Example 4.2 For the NASA code, which has $m = 6$,

$$\mathrm{SER}_4 \leq 88P_{10} + 467P_{12} + 2879P_{14} + 24259P_{16} + 158225P_{18} + 1009267P_{20} + \cdots$$

Now $\mathbf{u} = 100001$, which generates a fundamental path **c** having weight $16 > d_{\mathrm{free}} = 10$, can cause a 4-bit symbol error in 8 instead of 9 ways. However, this information sequence, and others containing groups of 4 or 5 consecutive zeros, have no significant effect on this bound which is tight for high E_b/N_0 (see Figure 4.1). Indeed, $s_8^-(d_{\mathrm{free}}) = s_8(d_{\mathrm{free}}) = 88$. The first term of the SER_4 bound, $[-3a(10) + \ell(10)]P_{10} = 88\ Q(\sqrt{10E_b/N_0})$, is indicated in Figure 4.1 by a dotted line. For eight-bit decoded symbols,

$$\mathrm{SER}_8 \leq 132P_{10} + 619P_{12} + 3651P_{14} + 29583P_{16} + 187325P_{18} + 1170891P_{20} + \cdots$$

and since $b = 8 > m-1 = 5$, every symbol overlapped by an error event will be decoded incorrectly. Using only the terms for $d = d_{\mathrm{free}}$ to $d = d_{\mathrm{free}} + \left\lceil \frac{10n}{k} \right\rceil$ in the SER_b bounds yields results with 2 significant digits of precision for E_b/N_0 values at which the bounds are useful (i.e., within 0.2 dB of simulation results). □

Now suppose that a convolutional code is concatenated with an outer block code that has b-bit symbols, rate k'/n', and minimum distance d_{min}. Define $e = \lceil d_{\mathrm{min}}/2 \rceil$ and assume sufficient outer codeword interleaving to make b-bit symbols statistically independent. The concatenated decoder word error (P_{w}) and bit error

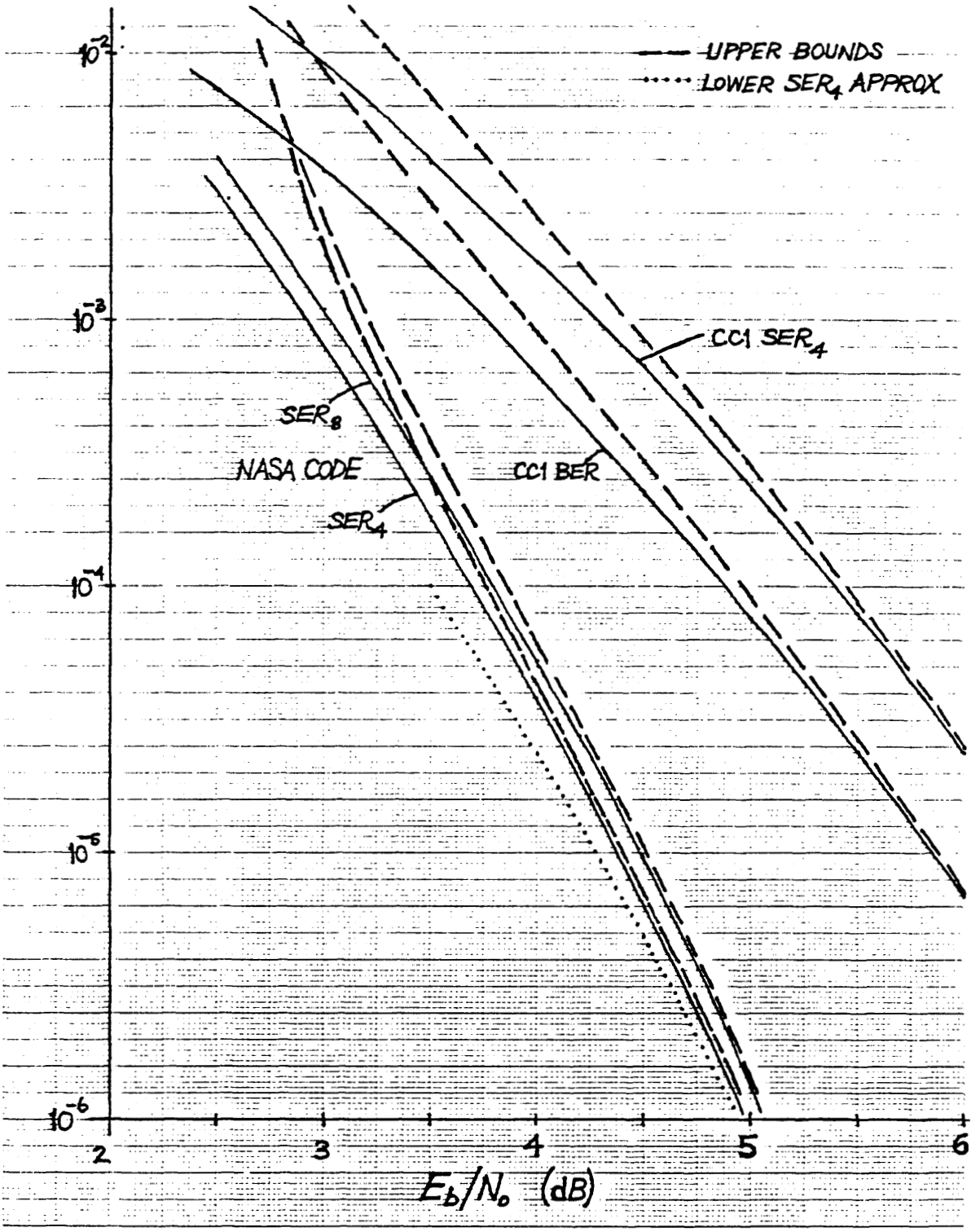


Figure 4.1 Symbol Error Rates for CC1 and the NASA Code.

(P_{bit}) probabilities are

$$P_{\text{w}} = \sum_{j=e}^{n'} \binom{n'}{j} (\text{SER}_b)^j (1-\text{SER}_b)^{n'-j}$$

$$P_{\text{bit}} \approx \frac{1}{2} \sum_{j=e}^{n'} \binom{n'}{j} (\text{SER}_b)^j (1-\text{SER}_b)^{n'-j} \frac{\max(d_{\text{min}}, j)}{n'}$$

when the block decoder corrects only up to $e-1$ errors [Torr84]. Hence,

$$P_{\text{bit}} \sim \frac{d_{\text{min}}}{2n'} \binom{n'}{e} (\text{SER}_b)^e$$

$$= s_b^-(d_{\text{free}})^e \frac{d_{\text{min}}}{2n'} \binom{n'}{e} \left[Q\left(\sqrt{2d_{\text{free}}(k/n)(k'/n')E_b^*/N_0} \right) \right]^e$$

for high $E_b^*/N_0 = (n'/k')E_b/N_0$, the concatenated decoder operating bit signal-to-noise ratio. Since $Q(x) \sim e^{-x^2/2}/(x\sqrt{2\pi})$, the asymptotic coding gain is

$$10 \log_{10} \left(\frac{kk'}{nn'} d_{\text{free}} \lceil d_{\text{min}}/2 \rceil \right) \quad \text{dB}.$$

4.2 A Lower Bound on Bit Error Rate

It is well-known that a lower bound on decoder error rates is the probability of error due to a single path having distance d_{free}. However, this bound is very loose except at very high signal-to-noise ratios. In this section, a lower bound on decoded BER is obtained by subtracting, from the union bound $k^{-1} \sum i(d) P_d$, a second term which accounts for the probability that the received sequence $\mathbf{r}$ is closer to two fundamental paths than to the all-zero path. For simplicity, a corresponding bound is now derived for the "first-event" error probability P_f that a Viterbi decoder makes an error for the first time. With probability P_d, the decoder selects a fundamental path of weight d instead of all-zero (the transmitted and therefore correct path), as the survivor into state 0. Summing probabilities over all $a(d)$ of these paths and all distances d yields the union bound

$$P_f \leq \sum_{d=d_{\text{free}}}^{\infty} a(d) P_d .$$

By the principle of inclusion-exclusion, a lower bound on P_f is obtained by subtracting from $\sum a(d)\, P_d$, the sum, over all pairs of fundamental paths $\mathbf{e}$ and $\mathbf{f}$, of the probability $p_0(\mathbf{e},\mathbf{f})$ that the decoder could select either $\mathbf{e}$ or $\mathbf{f}$ instead of the all-zero path as the survivor into state 0 at any trellis level. Thus $p_0(\mathbf{e},\mathbf{f})$ is the probability that the received channel symbol sequence $\mathbf{r}$ is closer to both $\mathbf{e}$ and $\mathbf{f}$ than to the transmitted all-zero path $\mathbf{0}$.

$$P_f \geq \sum_{d=d_{\text{free}}}^{\infty} a(d)\, P_d - \sum_{\mathbf{e}} \sum_{\mathbf{f}\neq\mathbf{e}} p_0(\mathbf{e},\mathbf{f}) \tag{4.4}$$

An analogous lower bound to (4.4) for BER follows by observing that, if the information vectors which generate $\mathbf{e}$ and $\mathbf{f}$ contain $i_\mathbf{e}$ and $i_\mathbf{f}$ 1's, then the contribution of these paths to (4.1) is $i_\mathbf{e} P_{d(\mathbf{e})} + i_\mathbf{f} P_{d(\mathbf{f})}$ where $d(\mathbf{e})$ and $d(\mathbf{f})$ are the number of 1's in $\mathbf{e}$ and $\mathbf{f}$. If $\mathbf{r}$ is closer to both of these paths than to $\mathbf{0}$, then at least $\min(i_\mathbf{e}, i_\mathbf{f})$ bit errors will occur. It follows from the "weighted" version of the principle of inclusion and exclusion that,

$$\text{BER} \geq \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} i(d) P_d - \frac{1}{k} \sum_{\mathbf{e}} \sum_{\mathbf{f}\neq\mathbf{e}} \max(i_\mathbf{e}, i_\mathbf{f})\, p_0(\mathbf{e},\mathbf{f})\,. \tag{4.5}$$

Now replace 0 with +1 and 1 with −1 on all branch labels, so that $\mathbf{0}$ becomes all +1's. Then for memoryless channels, $p_0(\mathbf{e},\mathbf{f})$ is a function of the noise level and the numbers

$$n_1 = |S_1 = \{i : e_i = +1 \ \text{ AND } \ f_i = -1\}|$$

$$n_2 = |S_2 = \{i : e_i = -1 \ \text{ AND } \ f_i = +1\}|$$

$$n_3 = |S_3 = \{i : e_i = -1 \ \text{ AND } \ f_i = -1\}|$$

where the index i runs from 1 to the maximum length (in bits) of $\mathbf{e}$ or $\mathbf{f}$.

Theorem 4.2 For n_1, n_2, and n_3 all nonzero, on the unquantized AWGN channel,

$$p_0(\mathbf{e},\mathbf{f}) = \frac{1}{\sqrt{2\pi}\sigma_3} \int_{-\infty}^{+\infty} e^{\frac{-(y-\mu_3)^2}{2\sigma_3^2}} Q\left(\frac{y+\mu_1}{\sigma_1}\right) Q\left(\frac{y+\mu_2}{\sigma_2}\right) dy$$

where $\mu_j = n_j\sqrt{E_s} = n_j\sqrt{RE_b}$ and $\sigma_j^2 = n_j\, N_0/2$, E_s and E_b are the transmitted channel symbol (i.e., signal) and information bit energies, $R = k/n$ is the code rate, and N_0 is the one-sided noise spectral density.

Proof. Let the all-zero path **0** be represented by +1's and transmitted as signals with amplitudes $+\sqrt{E_s}$. The channel adds zero-mean, white Gaussian noise with variance $\sigma^2 = N_0/2$. Thus, received (demodulated) channel symbols r_i are independent, conditionally Gaussian random variables with mean $+\sqrt{E_s}$ and variance σ^2. Define the vector $\mathbf{r}$ as $[r_1,\ r_2,\ \ldots,\ r_{n_1+n_2+n_3}]$. Let X_1, X_2 and X_3 be the sum of the variables r_i for those values of i in S_1, S_2, and S_3, respectively. At each trellis level, a Viterbi decoder prefers $\mathbf{e}$ to **0** as the survivor into the all-zero state if

$$||\mathbf{e} - \mathbf{r}|| \leq ||\mathbf{0} - \mathbf{r}||$$

where $||\cdot||$ means Euclidean distance and ties are resolved against the decoder. This condition is equivalent to

$$\sum_{i=1}^{n_1+n_2+n_3} -e_i r_i \ \leq \ \sum_{i=1}^{n_1+n_2+n_3} -r_i \, .$$

Since $\mathbf{e}$ and **0** differ in $n_2 + n_3$ positions, where $e_i = -1$, then

$$\sum_{i \in S_2 \cup S_3} 2r_i \leq 0 \qquad \text{so} \qquad -X_2 \ \geq \ X_3 \, .$$

Similarly, the decoder prefers $\mathbf{f}$ over **0** if $-X_1 \ \geq \ X_3$. Therefore,

$$\begin{aligned} p_0(\mathbf{e},\mathbf{f}) \ &= \ \Pr\{-X_1 \geq X_3 \quad \text{AND} \quad -X_2 \geq X_3\} \\ &= \ \int_{-\infty}^{+\infty} \Pr\{-X_1 \geq y,\ -X_2 \geq y \mid X_3 = y\} \Pr\{X_3 = y\}\, dy \, . \end{aligned}$$

The result follows from the fact that the X_j are independent Gaussian random variables with mean $n_j\sqrt{E_s}$ and variance $n_j\sigma^2$. □

Figure 4.2 shows a lower bound on the BER for the NASA code. Only fundamental paths of weight ≤ 20 were considered as possible error events because these are the most likely ones. Notice that the lower bound diverges at a higher E_b/N_0 than the upper bound because of the double sum over all pairs of these fundamental paths. However, for values of E_b/N_0 greater than the point at which the lower bound is horizontal, this bound helps to determine the accuracy of the union bound.

Another lower bound on BER results from considering the error probability due to only the fundamental paths of weight d_{free}. This probability is then lower bounded by an expression, like that on the right of (4.5), in which the sums are restricted to these paths. For the NASA code, this bound is shown in Figure 4.2 as a dotted line below the simulation data.

For the AWGN channel with very high bit signal-to-noise ratio high E_b/N_0, the union bound

$$\text{BER} \leq \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} i(d) P_d = \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} i(d) Q(\sqrt{2dRE_b/N_0})$$

is asymptotically equal to the first term in the sum,

$$\frac{i(d_{\text{free}})}{k} Q(\sqrt{2d_{\text{free}}RE_b/N_0}).$$

Conjecture. On the AWGN channel, a rate $R = k/n$, maximum-likelihood convolutional decoder's bit error rate is asymptotically equal to

$$\frac{i(d_{\text{free}})}{k} \cdot \frac{e^{-d_{\text{free}}RE_b/N_0}}{\sqrt{4\pi d_{\text{free}}RE_b/N_0}}$$

which is the first term in the BER union bound (4.1). Naturally, this conjecture is true if there is only one fundamental path having weight d_{free} because the error probability due to this path is a lower bound on BER. Essentially, the claim is that at high enough E_b/N_0, the fundamental paths of weight d_{free}, which dominate as error events, occur independently as error events. To prove the conjecture, one must show that $p_0(\mathbf{e}, \mathbf{f})$ for these paths is asymptotically negligible with respect to $P_{d_{\text{free}}}$.

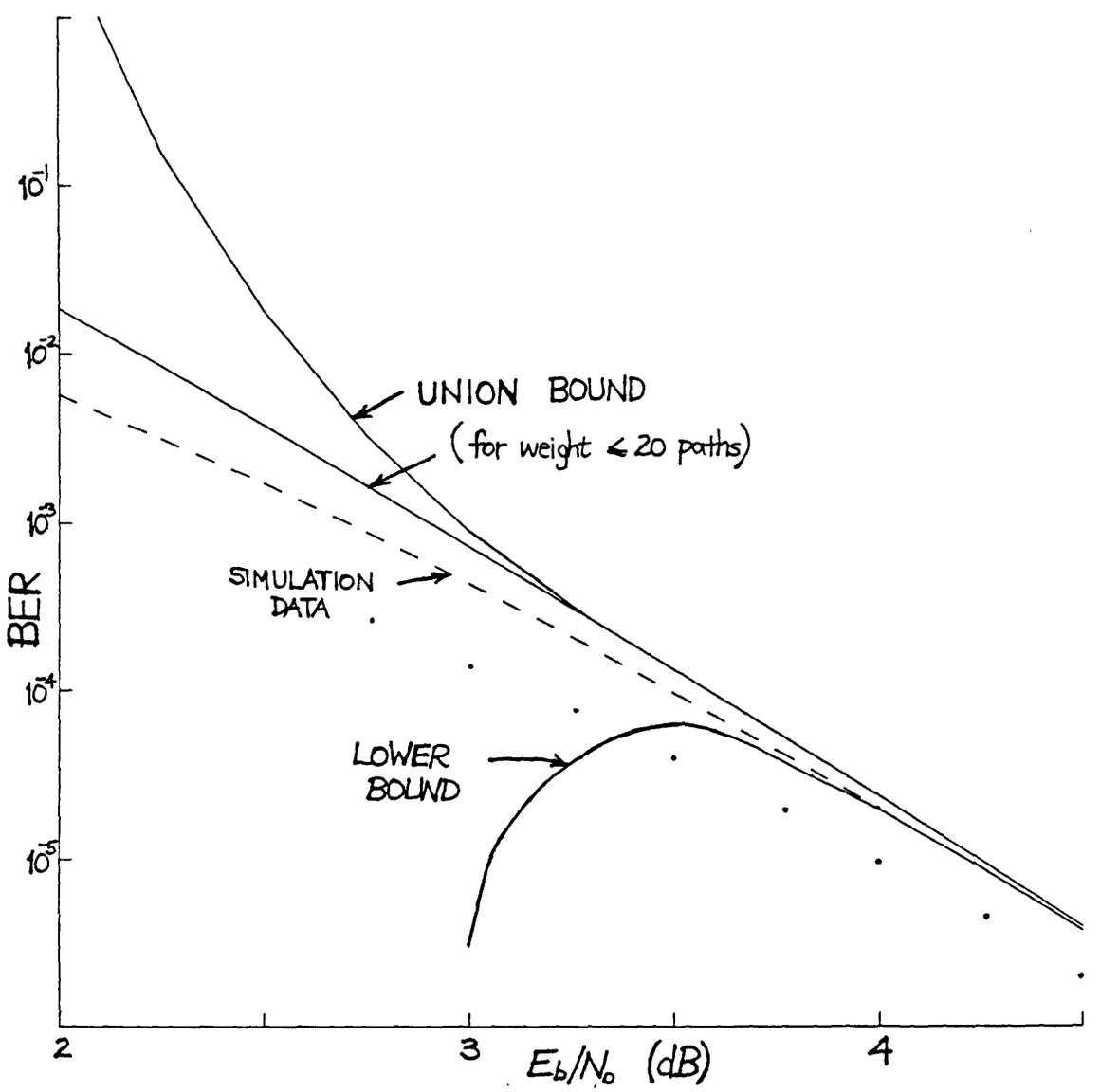


Figure 4.2 Bit Error Rate Bounds for the NASA Code.

4.3 Adapting Berlekamp's Tangential Union Bound

At low bit signal-to-noise ratios E_b/N_0 on the AWGN channel, the performance of block and convolutional codes depends largely upon the geometrical arrangement of error events in Euclidean space. In this section, Berlekamp's tangential union bound for block code error rates on the AWGN channel is used to obtain good approximations for Viterbi decoder error rates. These estimates are useful at low E_b/N_0, where the union bound is not, because they are within 0.2–0.3 dB of several simulated decoder error rates. Berlekamp proved that, for maximum-likelihood decoding of a length n binary block code on the AWGN channel,

$$P_d = \int_{-\infty}^{\infty} Q\left[\left(\sqrt{2nE_s/N_0} - z\right)\sqrt{\frac{d}{n-d}}\right] \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz \tag{4.6}$$

by considering codewords as points on a sphere of radius $\sqrt{2nE_s/N_0}$ in n-dimensional Euclidean space [Berl80]. A noise vector, which consists of n independent, zero-mean, unit variance, Gaussian random variables, is separated into a radial component z and a tangential component. The Q[ ] term in (4.6) is the error probability due to the tangential component of the noise vector, given a fixed value of radial noise z. By summing error probabilities over all nonzero codewords, Berlekamp obtains a union bound on word error probability P_w as

$$P_w \leq \sum_{d=d_{\min}}^{\infty} a(d) \int_{-\infty}^{\infty} Q\left[\left(\sqrt{2nE_s/N_0} - z\right)\sqrt{\frac{d}{n-d}}\right] \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz$$

which he shows is equivalent to the standard union bound $\sum a(d)Q(\sqrt{2dRE_b/N_0})$. However, for any given value of the radial noise component z, the probability of incorrect decoding due to the corresponding tangential noise is at most equal to 1. Therefore, the term

$$\sum_{d=d_{\min}}^{\infty} a(d) Q\left[\left(\sqrt{2nE_s/N_0} - z\right)\sqrt{\frac{d}{n-d}}\right]$$

may be limited to 1 for any value of z. This leads to the **tangential union bound**,

$$P_w \leq \int_{-\infty}^{\infty} \min\left(1, \sum_{d=d_{\min}}^{\infty} a(d) Q\left[\left(\sqrt{2nE_s/N_0} - z\right)\sqrt{\frac{d}{n-d}}\right]\right) \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz \tag{4.7}$$

Unlike the standard union bound, this new bound does not diverge at low E_b/N_0, because the right side is always less than or equal to 1.

Unfortunately, the tangential union bound cannot be directly adapted for Viterbi decoder first-event error probabilities because fundamental paths have different lengths. Therefore, error events do not correspond to signal points in a fixed-dimensional Euclidean space. Nonetheless, close approximations to Viterbi decoder error rates may be obtained by applying the bound to convolutional codes. Previous work in this direction [Mart79] will now be extended. Truncating a convolutional code so that all paths have the same length L yields a block code. However, the large value of L required to retain the important error events (i.e., low weight fundamental paths) causes (4.7) to yield virtually the same results as the standard union bound.

From (4.6), a Viterbi decoder selects a weight d, length ℓ trellis branches, fundamental path as the survivor path into the all-zero state with probability

$$\int_{-\infty}^{\infty} Q\left[\left(\sqrt{2n\ell E_s/N_0} - z\right)\sqrt{\frac{d}{n\ell - d}}\right] \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz\,.$$

Summing this probability over all fundamental paths yields the union bound

$$\begin{aligned} P_f &\leq \sum_{d=d_{\text{free}}}^{\infty} \sum_{\ell=\widehat{m}+1}^{\infty} a(d,\ell) \int_{-\infty}^{\infty} P_{d,n\ell,z} \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz \\ &= \int_{-\infty}^{\infty} \sum_{d=d_{\text{free}}}^{\infty} \sum_{\ell=\widehat{m}+1}^{\infty} a(d,\ell) P_{d,n\ell,z} \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz \end{aligned}$$

where $a(d,\ell)$ is the number of fundamental paths having weight d and length ℓ trellis branches. For a given value of z, the probability of error due to a fundamental path of length ℓ trellis branches and weight d is

$$P_{d,n\ell,z} = Q\left[\left(\sqrt{2n\ell E_s/N_0} - z\right)\sqrt{\frac{d}{n\ell - d}}\right].$$

Berlekamp shows that the union bound above on P_f is equivalent to the standard one (4.1) [Berl80]. Although z may no longer be interpreted as the radial component

of the noise vector, limiting

$$\sum_{d=d_{\text{free}}}^{\infty} \sum_{\ell=\widehat{m}+1}^{\infty} a(d,\ell) P_{d,n\ell,z}$$

to 1 yields the approximation

$$P_f \lesssim \int_{-\infty}^{\infty} \min\left(1, \sum_{d=d_{\text{free}}}^{\infty} \sum_{\ell=\widehat{m}+1}^{\infty} a(d,\ell) P_{d,n\ell,z}\right) \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz\,. \tag{4.8}$$

From the derivation of the bit error union bound (4.1), in which all possible ways that an error event could cause a single bit error were considered, it seems likely that any individual bit is decoded incorrectly with probability at most 1. This claim leads to the **tangential BER approximation**

$$\text{BER} \lesssim \int_{-\infty}^{\infty} \min\left(1, \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} \sum_{\ell=\widehat{m}+1}^{\infty} i(d,\ell) P_{d,n\ell,z}\right) \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz\,. \tag{4.9}$$

Similarly, from the derivation of the symbol error bounds in Section 4.1, an analogous approximation for SER_b is

$$\text{SER}_b \lesssim \int_{-\infty}^{\infty} \min\left(1, \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} \sum_{\ell=\widehat{m}+1}^{\infty} (\ell-m+b-1) a(d,\ell) P_{d,n\ell,z}\right) \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz \tag{4.10}$$

At high E_b/N_0 these two expressions become equal to the union bounds.

The tangential BER approximation (shown in Figure 4.3 for the NASA code) is extremely good at low E_b/N_0 and always better than the standard union bound that is useless for $E_b/N_0 \leq 2.7$ dB. An improved version of Algorithm 5 generated coefficients $a(d,\ell)$ and $i(d,\ell)$ for successive values of d up to 100. Truncating the sum on d at this point made no noticeable difference. A striking example of the tangential BER approximation is illustrated in Figure 4.4 for the memory 14, rate 1/4, Galileo code. The union bound diverges well before the intended operating point of $E_b/N_0 = 0.5$ dB. However, the tangential approximation yields an estimate of the decoded bit error rate that is about twice the simulated value and pessimistic by only 0.3 dB. This discrepancy is slightly smaller than that at the same BER

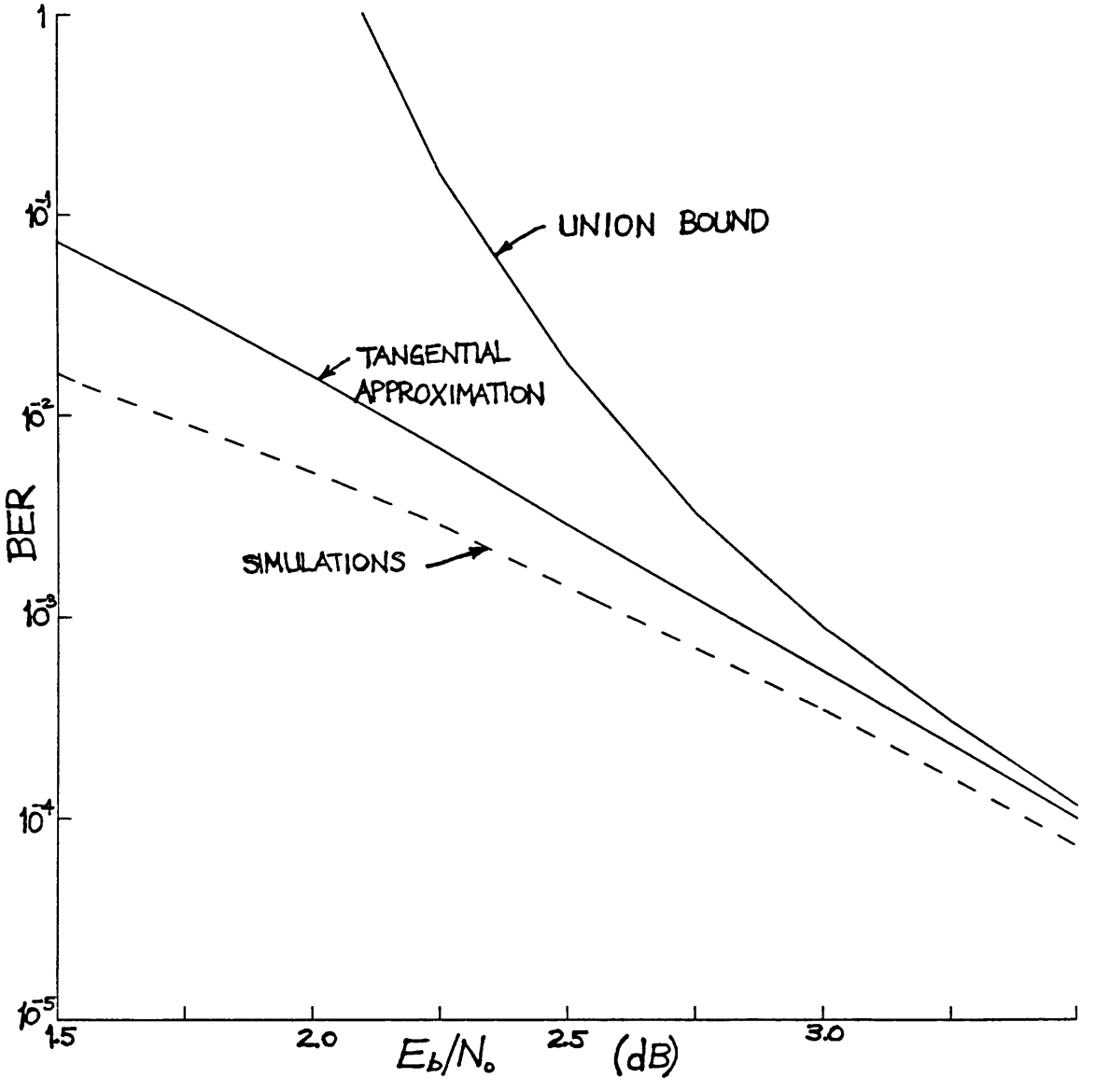


Figure 4.3 Tangential Approximation for the NASA Code.

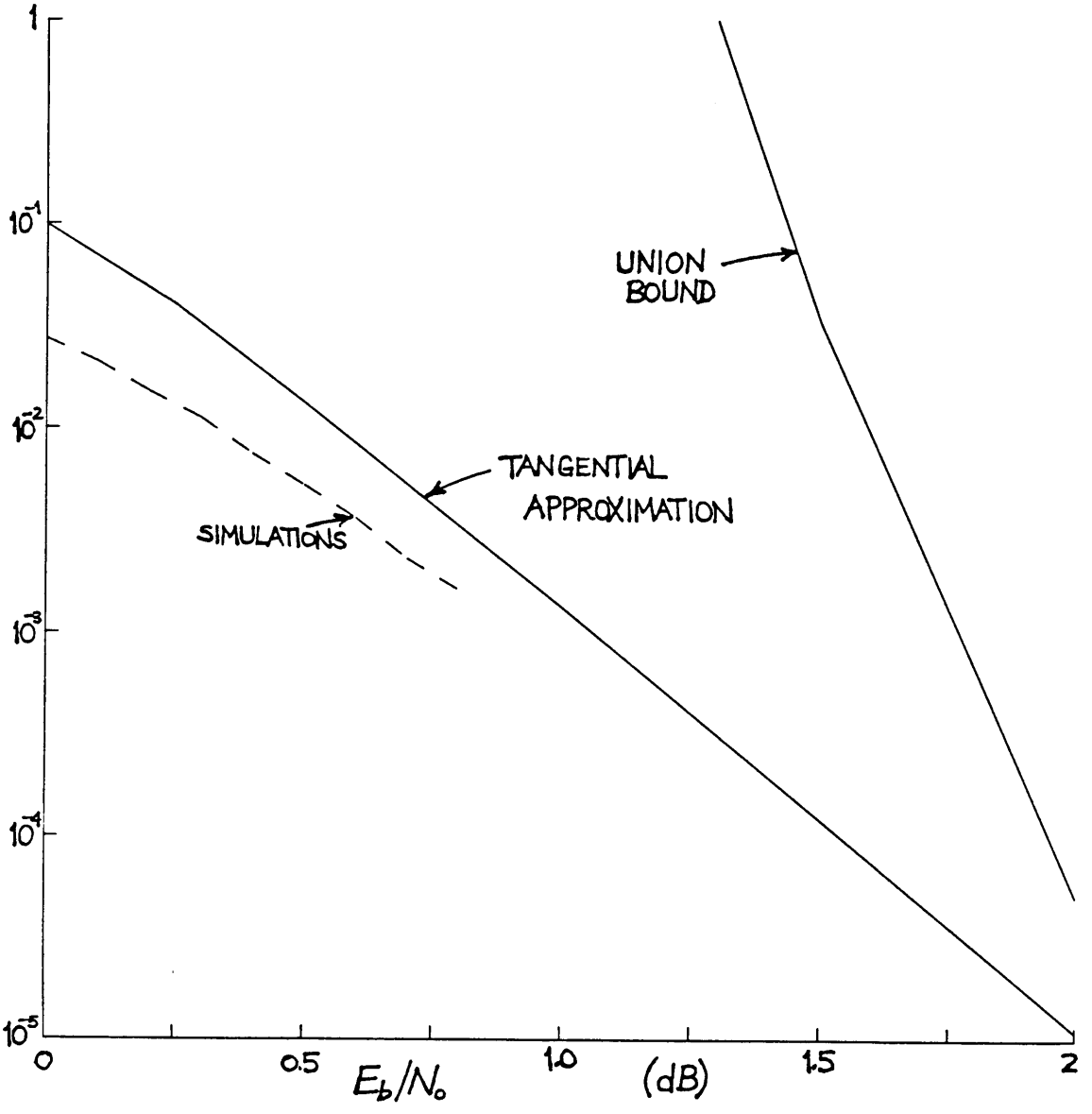


Figure 4.4 Tangential Approximation for the Galileo Code.

for the NASA code because only fundamental paths of weight ≤ 200 were used in evaluating (4.9) although longer error events may be important at extremely low E_b/N_0 near 0 dB.

4.4 Truncation Loss

In order to design efficient and high performance Viterbi decoders, one must know the precise loss from maximum-likelihood decoding (MLD) when survivor paths are truncated to T trellis branches. The problem is to determine the least value of T such that this loss is negligible because hardware for survivor storage usually forms a large (sometimes 50%) portion of the decoder. Recall from Section 1.1 that at each trellis level, a *best* state, truncation length T, Viterbi decoder outputs the information bit(s) corresponding to the T^{th} oldest branch of the survivor path into the state with lowest metric, while a *fixed* state decoder always uses a branch of the survivor into one fixed state. Also, fixed state decoders are widely used in practice because it is usually infeasible to find the state with least metric after each trellis level or groups of trellis levels. However, best state decoders may be analyzed, and the results can be applied to practical decoders. For example, extensive simulations show that a fixed state decoder requires twice the truncation length of a best state decoder [McEl89]. For those (practical) decoders which output bits for several trellis branches at a time, T should be interpreted as the *average* (not the *least*) truncation length because the average BER over the truncation lengths used is not significantly different from the BER of a decoder using the average truncation length to decode one branch at a time. The exact choice of T depends upon the code, channel noise level, acceptable loss from MLD, and the manner in which the Viterbi decoder outputs bits.

Assume that all zeros are transmitted and, at time $t+T$, let $\mathbf{e}$ denote the survivor path into the state s having lowest metric. Let $a_s(d,T)$ be the number of weight d trellis paths, of length T or more branches, which go from state 0 into state s via nonzero states. Consider decoding bit(s) for the trellis branch from level

t to $t+1$. Decoder bit error(s) occur i) if **e** merges with state 0 at some level $> t$ and nonzero information bit(s) correspond to the branch from level t to $t+1$, or else ii) if **e** merges with state 0 at level t but not afterwards, or else iii) with probability 1/2 if **e** merges with state 0 at level $< t$ but not afterwards (see Fig. 1). These situations are shown in Figure 4.5. First, note that MLD errors overbound those in case i). With probability P_d, the decoder selects one of the $a_s(d, T+1)$ weight d trellis paths of type iii). One of the $a_s(d,T) - a_s(d,T+1)$ weight d trellis paths of type ii) causes an average of $k2^{k-1}/(2^k-1)$ bit errors when k bits are decoded for the branch leaving state 0. Applying a union bound yields

$$P_b^{(T)} \leq \sum_{d=1}^{\infty} \left[\frac{i(d)}{k} + \frac{1}{2^k-1} \sum_{s=1}^{2^m-1} \left[2^{k-1} a_s(d,T) - \frac{1}{2} a_s(d,T+1) \right] \right] P_d \qquad (4.11)$$

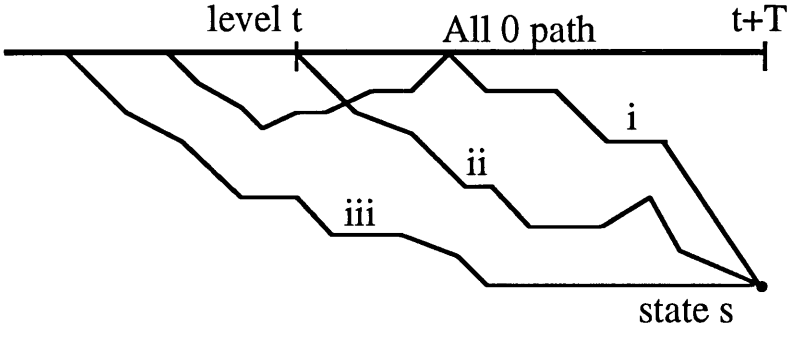


Figure 4.5 Trellis Paths.

This bound generalizes and strengthens a previous one [Hemm77]. Simulations of the $m=6$, rate 1/2 NASA standard code at a BER of 10^{-5} on the unquantized AWGN channel show that a Viterbi decoder with $T = 24, 27, 30$ or $T = \infty$ (i.e., MLD) performs within 0.1 dB of the $P_b^{(T)}$ bound. For other codes, similar accuracy is expected when the MLD bound $k^{-1} \sum i(d) P_d$ is tight.

For each state in a rate $1/n$ Viterbi decoder, the newest survivor bit is the oldest bit (s_m) of the predecessor state for which the state metric plus branch metric are

a minimum. Therefore, only $T-m$ physical storage bits ($kT-m$ in general) per survivor are required for truncation T decoding.

Define T_b^* as the least value of T such that $a_s(d,T) = 0$ for all $d \leq d_{\text{free}}$. A decoder with truncation length T_b^* will have no loss from MLD on "asymptotically quiet" channels such as the AWGN with extremely high bit signal-to-noise ratio E_b/N_0. Several researchers recommend decoding with truncation length T_b^* [Hemm77, And89], but this may cause a large loss, for example in E_b/N_0 on the AWGN channel, when the decoder BER $\geq 10^{-5}$. T_b^* is shown in Table 4.1 for several rate $1/n$ codes with encoder generator polynomials represented in octal form and right-justified so that x^3+x^2+1 is shown as 15.

A maximum-likelihood decoder's performance on a memoryless channel does not change when all encoder generator polynomials are reversed. For if an information vector $\mathbf{u}$, starting and ending with m zeros, encodes to $\mathbf{c}$, then $\mathbf{c}$ reversed is the output of the reversed encoder with input $\mathbf{u}$ reversed. However, if any generator polynomial is not symmetric, the loss for a Viterbi decoder having $T < \infty$ may change because the truncated trellis branch labels and thus $a_s(d,T)$ values are different. Therefore, encoders should be designed with generator polynomials oriented to minimize truncation loss.

Example 4.5 For the (13,17) code, $P_b^{(10)} \leq 4P_6+32P_7+102P_8+240.5P_9+\cdots$ while for the reversed code (15,17), $P_b^{(10)} \leq 2P_6+29P_7+85.5P_8+223.5P_9+\cdots$ (The MLD BER $\leq 2P_6+7P_7+18P_8+49P_9+\cdots$). On the unquantized AWGN channel at $E_b/N_0 = 5.41$ dB, the above bounds are $3.42\times10^{-5}, 2.66\times10^{-5}$, and 10^{-5}, respectively. (The first 20 nonzero terms in (4.11) were sufficient for 3 significant digits of precision). An additional 0.40 or 0.32 dB is required when $T_b = 10$ for the (15,17) or (13,17) code, respectively, to achieve a BER of 10^{-5}. These losses are excessive because only 0.05 dB extra is needed when $T_b = 14$. Furthermore, $P_b^{(10)} \leq 10^{-5}$ for the (5,7) code at $E_b/N_0 = 5.94$ dB, with only half the decoding computation. Hence, truncation length $T_b^* = 10$ for the (15,17) code is insufficient, unless the BER is $< 10^{-8}$. □

The truncation bound (4.11) may be computed using the "path counting" Algorithm 5 described in Section 3.6. The parameter "coeffs" is set to $\lceil 10n/k \rceil$. When the algorithm terminates, $i(d)$ values are recorded. At this point, the algorithm is conceptually at time t in the code trellis of Figure 4.5, so $P[0][0]$ is made 999 to inhibit paths from state 0. Also, A, B and change may be ignored. Then after the T^{th} next step, $a_s(d,T) = W[s][d-W[s][0]+1]$ for $d = W[s][0]$ to $W[s][0]+\text{coeffs}-1$. Thus, Algorithm 4 can easily produce quantities required in (4.11).

A more accurate estimate of truncation loss may be obtained using an approximation developed from the tangential union bound.

In Table 4.1, T_b is the least value of T such that a best state Viterbi decoder will perform within 0.05 dB of MLD on the unquantized AWGN channel for BER $\leq 10^{-5}$. Since each survivor is $L_b = T_b - m$ bits long, the decoder's path memory grows linearly with T. However, the decoder complexity doubles for each increase in m by one while the E_b/N_0 needed for BER $\leq 10^{-5}$ decreases less than 0.5 dB for most codes in Table 4.1. Therefore, a truncation length of at least T_b should be used. A larger T_b may be needed as m increases beyond 6 because the E_b/N_0 gain decreases so the acceptable truncation loss is reduced. For higher/lower error rates, T_b must be greater/smaller to maintain a fixed E_b/N_0 loss. Recall that T_b^* is the least truncation length required for no asymptotic loss from MLD. This theoretical value is (considerably) less than the practical T_b suggested. For best-state decoders that output a block of bits at a time, the average truncation length should be T_b. For fixed state Viterbi decoders, the results of [McEl89] suggest that survivors should be made $2L_b$ bits long.

TABLE 4.1 Truncation Lengths for Several Rate 1/2 or 1/3 Codes.

E_b/N_0	m	octal generators	d_{free}	T_b^*	T_b	bits L_b
5.94	2	5, 7	5	8	10	8
5.46	3	15, 17	6	10	14	11
5.07	4	23, 31	6	13	21	17
4.63	5	75, 57	8	19	27	22
5.68	6	1, 117	6	12	14	8
4.23	6	133, 171	10	27	35	29
3.93	7	345, 237	10	28	46	39
3.63	8	561, 753	12	33	59	51
3.49	9	1167, 1545	12	37	68	59
5.95	2	5, 7 , 7	8	9	11	9
5.19	3	13, 15, 17	10	10	13	10
4.68	4	37, 33, 25	12	13	17	13
5.05	5	1, 75, 67	10	10	13	8
4.25	5	71, 65, 57	13	17	22	17
3.93	6	133, 171, 165	14	21	29	23
3.57	7	251, 233, 357	16	20	37	30
3.37	8	557, 663, 711	18	25	45	37
3.15	9	1765, 1631, 1327	19	26	53	44

References

[Abr64] *Handbook of Mathematical Functions*, U.S. Govt. Printing Office, M. Abramowitz and I. A. Stegun, ed., Washington, D.C., pp. 959–962, 1964.

[And89] J. B. Anderson and K. Balachandran,"Decision Depths of Convolutional Codes," *IEEE Trans. Inform. Theory*, IT-35, pp. 455–459, March 1989.

[Berl80] E. R. Berlekamp, "The Technology of Error-Correcting Codes," *Proc. of the IEEE*, Vol. 68, pp. 564–588, May 1980.

[Hemm77] F. Hemmati and D. J. Costello, Jr., "Truncation Error Probability in Viterbi Decoding," *IEEE Trans. Commun.*, COM-25, pp. 530–532, May 1977.

[Mart79] D. R. Martin, "Error Bounds for Concatenated Reed-Solomon/Convolutional Codes," in *Proc. of MILCOM '79*, Vol. 2, pp. 23.5.1–23.5.6, October 1979.

[McEl89] R. J. McEliece and I. M. Onyszchuk, "Truncation Effects in Viterbi Decoding," in *Proc. of MILCOM '89*, Vol. 2, pp. 27.3.1–27.3.5, October, 1989.

[Torr84] D. J. Torrieri, "The Information-Bit Error Rate for Block Codes," *IEEE Trans. Commun.*, COM-32, pp. 474–476, April 1984.

[Vit79] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, New York: McGraw-Hill, 1979.

Chapter 5

Quantization Loss in Convolutional Decoding

Using $q < 8$ bits to quantize received symbols from the AWGN channel may significantly increase the bit signal-to-noise ratio (E_b/N_0) required to achieve a particular decoded bit error rate (BER). This increase in dB, called quantization loss, is measured with respect to using $q=12$, which causes no measurable loss. The hardware complexity of one section in a convolutional decoder increases linearly with both q and the number of bits (ℓ) used to represent path metrics. The decoder's speed also depends strongly upon q and ℓ. In order to design efficient and high performance decoders, these parameters must be chosen as the smallest values that do not cause a significant E_b/N_0 loss. For example, changing from the best memory 5, rate 1/2 code ($d_{\text{free}} = 8$) to the memory 6, rate 1/2 NASA code yields an additional 0.41 dB coding gain when the BER is 10^{-5}, at the expense of doubling the Viterbi decoding complexity. Since a loss of 0.2 dB would occur in either case if $q=3$, more than 3 quantization bits must be used. Although BER depends upon the code, the quantization loss seems to depend mainly upon the channel.

In the next section, an optimal branch metric and uniform quantization scheme are derived for the AWGN channel. Nonuniform quantizers are not considered because they do not significantly decrease BER. For practical values of q, AWGN channel capacity and cutoff rate are used to compute lower and upper bounds (that lead to accurate estimates) on the optimal uniform quantizer stepsize Δ and the corresponding E_b/N_0 power loss. The range of path metrics and renormalization of state metrics are analyzed, in particular for the memory 14, rate $1/n$ for $n=2$ to 6, new BIG Viterbi Decoder (BVD) at JPL for the Galileo mission [Stat88]. Since the BVD performs twice the computation of a memory 13 decoder having similar design, but requires (at rate 1/4) about 0.1 dB less E_b/N_0 for a bit error rate (BER) of 0.005, even a quantization loss of 0.02 dB may be unacceptable. A probabilistic bound on the maximum difference between any two path metrics is derived to

minimize ℓ. The theoretical results are verified by simulations of three codes: the NASA standard code; the experimental, $m=14$, rate 1/4, Galileo code [Stat88]; and the $m=14$, rate 1/6, "2-dB" code for future deep-space missions [Yuen85]. Since the same quantization losses occurred when symbol error rates were measured, these results apply when a block code is concatenated with an inner convolutional code. Although the examples presented here are for rate $1/n$ Viterbi decoders, this work applies to soft-decision decoding of block codes and to other convolutional decoders.

5.1 Branch Metrics

When an encoded 0/1 is mapped to $+1/-1$ and then transmitted with signal energy $s^2=E_s$, the demodulator output at the receiver is a conditionally Gaussian random variable y with mean $+s/-s$ and the same variance $\sigma^2 = s^2/(2RE_b/N_0)$ as the zero-mean AWGN channel noise. Binary phase-shift keying (BPSK) or quadrature PSK modulation with ideal coherent detection is assumed. For purposes of analysis, y is usually divided by σ because the automatic gain control (AGC) in a wideband receiver makes the channel noise variance essentially unity [Hell71, Gill71]. Since JPL's demodulator always outputs symbols with a mean $s=0.84$ volts, y will not be normalized here and this value of s will be used herein. If y was divided by σ, then so should all quantizer stepsizes in this chapter.

For the AWGN channel, a Viterbi decoder finds the trellis path with minimum Euclidean distance or equivalently, minimum negative inner product, to the received sequence. Thus, the metric for a trellis branch is the inner product of the length n branch label, with 0/1 replaced by $+1/-1$, and the negative of a received vector $[y_1, y_2, \ldots, y_n]$. Note that branches with lower metrics are closer to the received vector and that trellis path metrics are the sum of the constituent branch metrics. If σ does not change significantly, incrementing or multiplying all branch metrics by a constant does not alter the decoder's output. Thus $-y_i$ or $+y_i$, equivalently $(-y_i + |y_i|)/2$ or $(y_i + |y_i|)/2$, is added to the metrics of those branches with a $+1$ or -1 in position i. Therefore, the decoder may add $|y_i|$ to the metrics of branches

having different signs in position i than that of y_i, and zero otherwise. This sign-magnitude method will be used throughout this chapter because it halves the branch and state metric maximum ranges which result from using standard integer metrics [Clar81, Hell71]. The new method is provably optimum for uniform quantization and superior to previous heuristic schemes [Clar81, Gilh70].

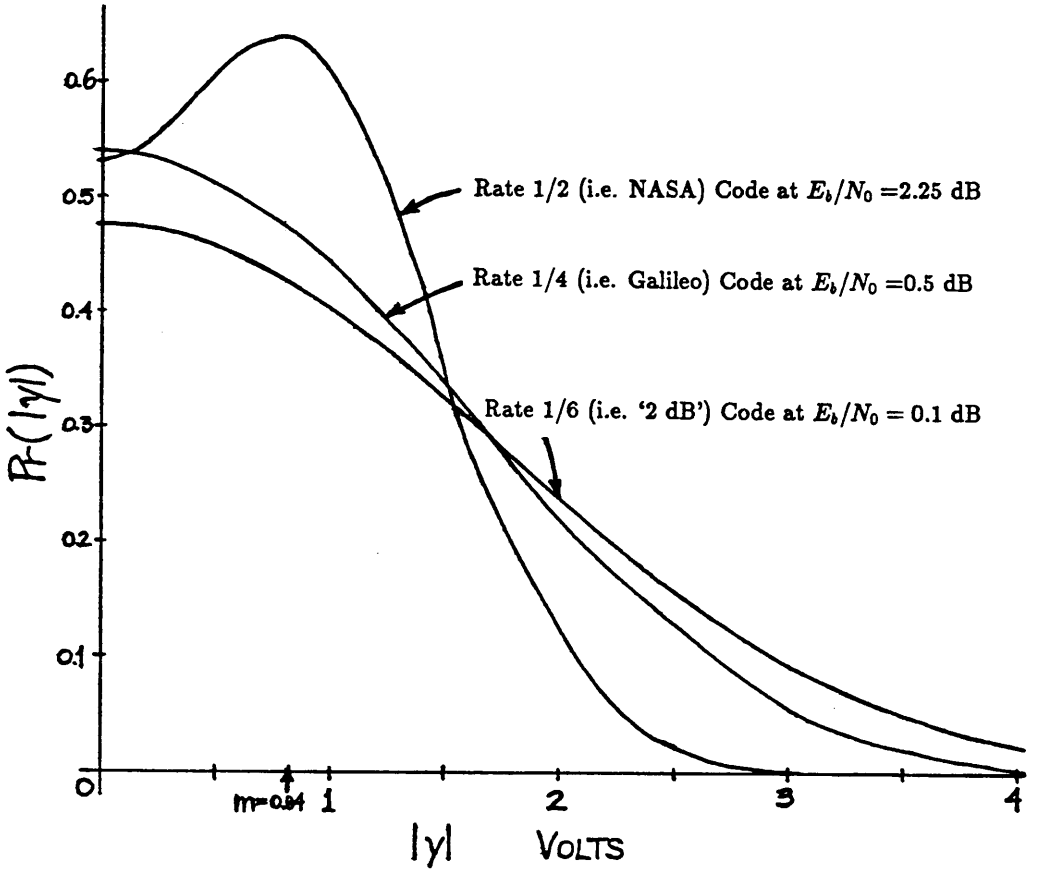


Figure 5.1 Received Signal Magnitude Distribution.

5.2 Channel Symbol Quantization

A decoder first quantizes analog voltages y before computing branch metrics. When zeros and ones are equally likely in the encoder input data,

$$\Pr(|y| = x) = \frac{1}{\sqrt{2\pi}\sigma}\left[e^{-(x-s)^2/2\sigma^2} + e^{-(x+s)^2/2\sigma^2}\right] .$$

The probability distribution of $|y|$ is shown in Figure 5.1 for the operating noise levels of three JPL codes. However, these curves do not aid quantizer design, which should be based upon channel cutoff rate and the fact that integers represent quantized symbols in practical decoders.

Let the random variable J be the quantized value of y and define

$$\Pr(J = j|+1) = \frac{1}{\sqrt{2\pi}\sigma}\int_{(j-0.5)\Delta}^{(j+0.5)\Delta} e^{-(y-s)^2/2\sigma^2}\, dy\ , \quad -2^{q-1}+2 \le j \le 2^{q-1}-2.$$

Then $p_j = \Pr(J = j|+1)$ is the probability that the integer j represents y after quantization. For $j = \pm(2^{q-1}-1), p_j$ is the above integral with limits $(j-0.5)\Delta$ and $+\infty$, or $-\infty$ and $(j+0.5)\Delta$.

Since $|J_1|, \ldots, |J_n|$ are summed to form branch metrics, the absolute error $|J_i - y_i|$ in quantizing y_i is also the contribution to the branch metric error that results from quantization. A decoder using signed integers to represent J_i could conceptually use 0, $\pm\Delta$, $\pm 2\Delta, \ldots, \pm(2^{q-1}-1)\Delta$ for any real number Δ, because multiplying all metrics by Δ has no effect. Therefore, a uniform quantizer should have thresholds spaced Δ volts apart at $\pm\Delta/2$, $\pm 3\Delta/2, \ldots, \pm(2^{q-1})\Delta/2$, because this minimizes the metric error defined above (and also any positive function of $J_i - y_i$, such as the quantizer mean square error). Note that this quantizer will not necessarily minimize decoder BER, the desired objective. A nonuniform scheme might perform better [Vit79, p. 78], at the expense of additional complexity. However, simulations of the NASA code using 3-bit integer branch metrics and nonuniform quantization schemes (including the ones that maximized capacity or cutoff rate) never produced lower BERs than using the best Δ. Simulations of several

codes indicated that nonuniform quantizers with 3,4, or 5 levels would yield slight improvements (but the gain would decrease rapidly with more levels). Thus, only uniform quantization schemes, characterized by q and Δ, are considered herein.

Since J_i is normally one of 7 values from -3 to $+3$ when $q=3$, quantizer levels $+4$ and -4 are appended (Figure 5.2) in this case only, to achieve the BER for 8 levels and standard integer metrics. Only for $q=3$ herein, $2^{q-1}-1$ will be replaced by 4 instead of 3. In rate 1/2 decoders, a branch metric of 8 is decreased to 7 so that $q=3$ bits still represent all possible values. This event occurs with probability $(p_{+4}+p_{-4})^2$, which is only 0.11 for the NASA code at $E_b/N_0 = 2.25$ dB.

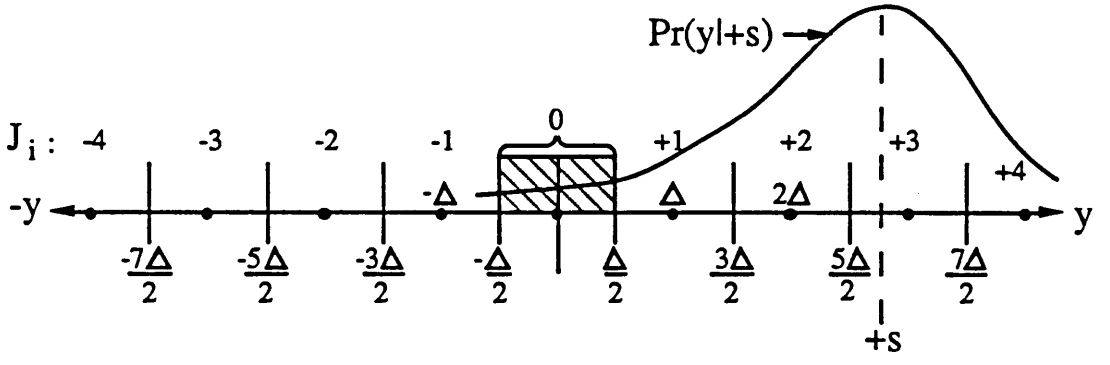


Figure 5.2 An Optimal Quantization Scheme.

5.3 Quantization Loss

The following quantities are used to estimate the E_b/N_0 loss when received channel symbols are quantized with q bits:

$$\gamma = \sum_{j=-2^{q-1}+1}^{2^{q-1}-1} \sqrt{p_j p_{-j}}$$

is almost 0 for high E_b/N_0 and approaches 1 for very noisy channels. The binary-input, q-bit uniformly quantized, AWGN channel capacity and cutoff rate are

$$C_u(q) = 1 - \sum_{j=-2^{q-1}+1}^{2^{q-1}-1} p_j \log_2\left(1+\frac{p_{-j}}{p_j}\right)$$

$$R_0(q) = 1 - \log_2(1+\gamma) \qquad \text{bits per channel use.}$$

Observe how rapidly the maximum possible (uniform) cutoff rate $R_0(q)$ and capacity $C_u(q)$ in Figure 5.3 approach their limits when $\sigma = 0.65$ (the rate 1/2 NASA code at $E_b/N_0 = 2.25$ dB) and when $\sigma = 1.12$ (the rate 1/4 Galileo code at $E_b/N_0 = 0.5$ dB). $R_0(3)$ and $C_u(3)$ are based upon the 9-level quantizer in Figure 5.2 while a $q = 4$ or $q = 5$ quantizer has 15 or 31 levels, respectively. The lines between data points correspond to uniform quantizers having intermediate numbers of levels, such as 24.

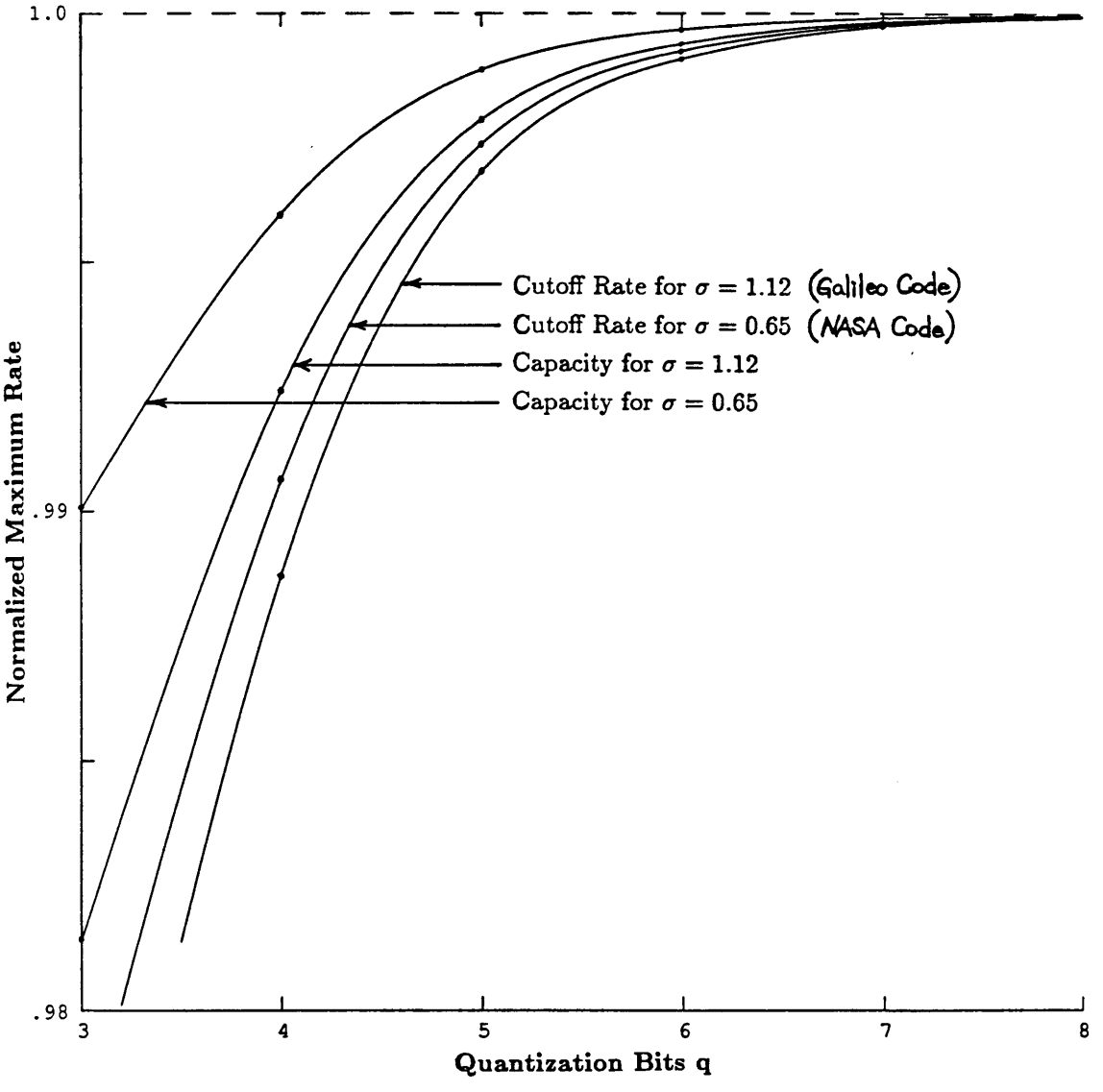


Figure 5.3 Uniformly-Quantized, AWGN Channel Capacities and Cutoff Rates

The curves in Figure 5.3 show that there is negligible cutoff rate loss for $q > 6$, and in fact $R_0(5)/R_0(\infty) \geq 0.996$ suggests a very small quantization loss for $q \geq 5$ (verified by simulations). The above formulae lead to theoretical bounds on quantizer E_b/N_0 loss, shown in Tables 5.1 and 5.2. Capacity loss is the additional E_b/N_0 required for a q-bit quantized channel to have the same capacity as one with $q=16$ (essentially unquantized). It indicates the theoretically least possible E_b/N_0 degradation and should be considered when the code rate is greater than the channel cutoff rate, which is true for the three JPL codes. The measured quantization losses from software simulations agree with the cutoff rate loss, and both sets are virtually independent of the channel noise level. In practice, quantization losses are larger because the noise is usually not perfectly Gaussian. The $q=3$, 9-level loss of 0.14 dB is much less than the well-known (8-level) 0.25 dB value calculated using a union bound [Hell71, Gilh71, Clar81].

TABLE 5.1 Quantization Losses in dB ($\sigma = 0.65$).

q bits	capacity loss	cutoff rate loss	measured
3	0.084	0.135	0.14
4	0.034	0.054	0.05
5	0.010	0.016	—
6	0.003	0.005	—

TABLE 5.2 Quantization Losses in dB ($\sigma = 1.12$ or 1.455).

q bits	capacity loss	cutoff rate loss	measured
3	0.110	0.130	—
4	0.044	0.053	0.05
5	0.012	0.015	0.02
6	0.004	0.005	—

These numbers do not change when an outer block code is concatenated with a convolutional inner code because the simulation losses do not depend upon whether BER, 4-bit SER, 8-bit SER, etc., is measured.

5.4 Quantizer Stepsize

The stepsize Δ that maximizes cutoff rate for a given E_b/N_0 almost minimizes BER and symbol error rate (SER) when E_b/N_0 increases by up to 1 dB. Δ should be chosen to maximize capacity only if the code rate is well above the cutoff rate, otherwise Δ should maximize cutoff rate. For Viterbi decoding at high E_b/N_0, Δ could be chosen to minimize the union bound

$$\text{BER} \leq \sum_{d=d_{\text{free}}}^{\infty} i(d) P_d \ ,$$

where P_d is the probability that the decoder chooses a path at distance d from the one transmitted. Define

$$p(x) = \sum_{j=-2^{q-1}+1}^{2^{q-1}-1} p_j \, x^j$$

and let $[p(x)^d]_i$ denote the coefficient of x^i in $[p(x)]^d$. Then

$$P_d = \frac{[p(x)^d]_0}{2} + \sum_{i=-2^{q-1}+1}^{-1} [p(x)^d]_i \ .$$

According to the union bound, the NASA code E_b/N_0 quantization loss when $q=4$ is at most 0.065 dB for BER $\leq 10^{-5}$ or $E_b/N_0 > 4$ dB. However, simulations indicate that quantization loss is independent of the code, insensitive to the channel noise level, and significantly less for $q = 3$ and $q = 4$ than the values computed using the union bound. Furthermore, the union bound is not accurate at moderate E_b/N_0 and becomes useless at low E_b/N_0 (2 dB) where many convolutional codes operate. Therefore, maximizing channel cutoff rate is the preferred method for determining quantization loss and the best Δ.

In Figure 5.4 for the NASA code, there is a negligible loss for $q \geq 4$ and the BER increases slowly for Δ greater than the optimum, so Δ should be larger instead of smaller than the best value. The labels C and R_0 in Figure 5.4 indicate the stepsizes that maximize $C_u(q)$ and $R_0(q)$, respectively. The Δ that maximizes $R_0(q)$ is a safe choice since it is larger than the value which minimizes BER, and

also because it yields the lowest BER (while maximizing capacity does not) for $q=3$ with 9 quantizer levels (Figure 5.2). The Δ that maximizes $C_u(q)$ is a lower bound on quantizer stepsize, just as the quantization loss according to capacity loss was a lower bound in the last section. The shape and spacing of the BER curves in Figure 5.4 are the same for the corresponding 8-bit SER curves (and also if 4,5,6, or 7-bit SER is measured). Therefore, concatenating the NASA code with the (255,223) Reed-Solomon code does not affect quantization loss, as expected because the loss depends only upon the channel.

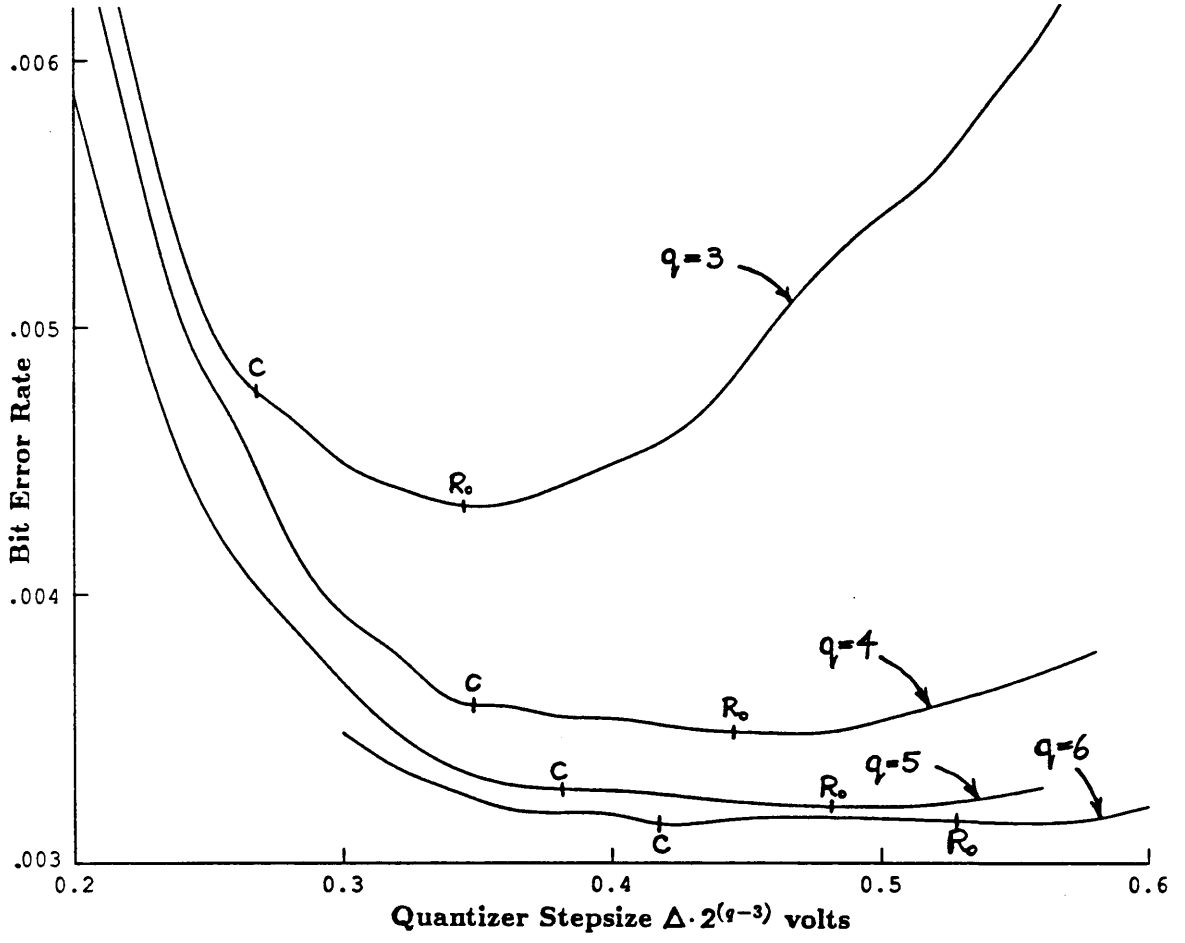


Figure 5.4 Bit Error Rate of the NASA Code at $E_b/N_0 = 2.25$ dB.

Many sets of software simulations were run for the NASA code and the Galileo code. The values of q were 3, 4, 5, or 6 and E_b/N_0 ranged from 0 dB to 3.5 dB.

In all simulations, the Δ's which maximize $C_u(q)$ or $R_0(q)$ were slightly smaller or larger (respectively) than the Δ that minimized BER. For $q=3$ or 4, the Δ's which minimize the quantizer mean-square error or absolute error were too large. Simulations of the two $m=14$ codes revealed a 0.02 dB or 0.05 dB E_b/N_0 quantization loss for $q=5$ or 4 at the BER of 0.005 required for JPL's images and at an 8-bit SER of 0.20 for a concatenated system BER of 10^{-6} (using an outer (255,223) Reed-Solomon code).

5.5 Metric Range and Renormalization in Viterbi Decoders

For each received n-vector and encoder state $\underline{s}$, a Viterbi decoder finds the trellis path, into $\underline{s}$, which has the least metric (sum of the metrics of the consecutive branches forming the path). This trellis path is called the **survivor path** for state $\underline{s}$ and the state metric of $\underline{s}$ is the metric of this path. Recall that a trellis level consists of all the branches corresponding to a received vector of n demodulated channel symbols. When the channel is noisy, the minimum over all state metrics will increase as the decoder processes trellis levels. Since state metrics are stored in ℓ-bit registers, they must all be decreased by a constant before an overflow occurs. This procedure, called **renormalization** could be implemented by subtracting the least state metric from all metrics after each trellis level. However, finding the minimum of the state metrics is impractical in most decoders, such as the fully connected BVD which contains 16384 states. Alternatively, renormalization can be accomplished by zeroing every register's most significant bit (msb), which is equivalent to subtracting $2^{\ell-1}$ from every metric if every register has an msb of 1. Since detecting the latter event is impractical for most decoders (including the BVD), several new methods for renormalization are now described.

After each trellis level, let the random variable M be the difference between the maximum and minimum state metrics. If any state metric is $\geq 2^{\ell-1}+2^{\ell-2}$, (its two most significant bits are 1) and $M \leq 2^{\ell-2}$, then the msb is 1 for all state metrics since they are at least $2^{\ell-1}$, so renormalization should occur. This method requires

that $\ell = 2 + \lceil \log_2 \widehat{M} \rceil$, where $\widehat{M}$ is the maximum state metric range for which the decoder always retains paths closest to the received sequence (i.e., it operates correctly). The following technique uses up 1.4 bits of ℓ. Let W be the maximum of the metrics of the all-zero state, the all-one state, and the state with a 1 input followed by $m-1$ zeros. Since most state metrics differ from one of these three metrics by the contributions from only a few quantized channel symbols, W is very close to the largest state metric (Galileo code simulations verified this). Therefore, renormalization could occur when W exceeds a threshold such as $2^{\ell-1}+2^{\ell-2}+2^{\ell-3}$.

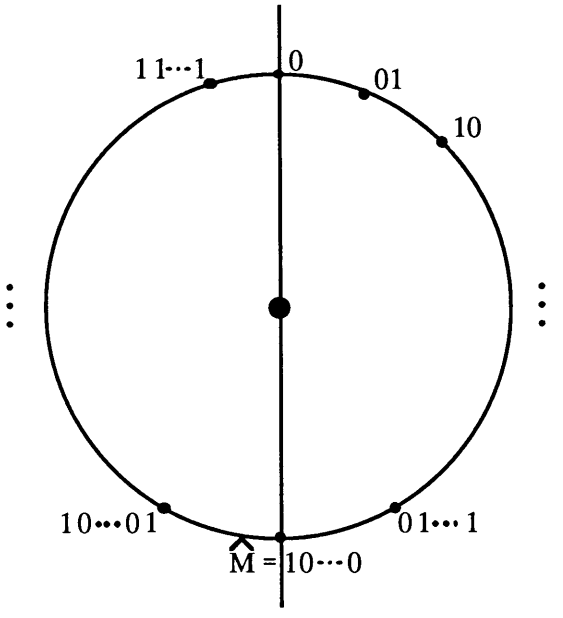


Figure 5.5 Binary Modulo Metrics.

The following scheme for comparing path metrics and renormalizing state metrics is currently the best [Shun90]. M and $\widehat{M}$ must be defined for metrics of paths into a state, not state metrics. Let M be the maximum difference between the metrics for any pair of paths into the same state. If $M \geq \widehat{M}$, the decoder will choose an incorrect survivor path (perhaps causing extra bit errors). By performing path metric arithmetic modulo $2\widehat{M}$, renormalization occurs automatically. Metrics are conceptually arranged on a circle of circumference $2\widehat{M}$ (see Figure 5.5). Since any pair always lies within a half-circle, the decoder finds the least one by comparison without the msb's but reverses the result when the msb's of the two metrics differ

[Shun90]. Using modulo arithmetic requires only one extra bit, so $\ell = 1 + \lceil \log_2 \widehat{M} \rceil$. Furthermore, global signals for renormalization and metric monitoring are eliminated, a crucial advantage in high-speed, parallel decoders. This "best" method will be assumed in the following analysis for rate $1/n$ Viterbi decoders (which also work for rate $(n-1)/n$ punctured codes).

Lemma. $M \leq d_{\text{free}} \cdot (2^{q-1} - 1)$.

Proof. Let b and w be two paths (into some state s) with largest metric difference of all pairs of paths compared by the decoder at the current trellis level. Let b have lower metric than w. Since a convolutional code is linear, adding a weight d_{free} fundamental path to b yields a path c that differs from b in d_{free} positions but coincides with w on a branch into s. Since w is the path with least metric at the previous trellis level into a state which preceeds s, the metric of w is less than or equal to the metric of c, which is at most $d_{\text{free}} \cdot (2^{q-1} - 1)$ plus the metric of b, because the maximum contribution to a branch metric by one received channel symbol y is $2^{q-1} - 1$. □

Corollary. In the absence of noise, $M = M_0 = d_{\text{free}} \lfloor 0.5 + s/\Delta \rfloor$.

M_0 may be a useful design parameter because noisy channel simulations of the Galileo and NASA codes suggest that it is an upper bound on the mean of M and that $2M_0$ is usually larger than M. Typically, d_{free} is substantially less than $n \cdot (m+1)$, the maximum possible. For the Galileo code $d_{\text{free}} = 35$, so $M_0 = 140$ for $q = 5$, $\Delta = 0.20$, and $s = 0.84$. For the rate 1/6 "2-dB" code, $d_{\text{free}} = 56 < n \cdot (m+1) = 90$ so $M_0 = 336$ for $q = 6$ and $\Delta = 0.14$ volts.

Claim. $\Pr(M > \widehat{M}) \leq \sum_{i=\widehat{M}+1}^{d_{\text{free}}(2^{q-1}-1)} [p(x)^{d_{\text{free}}}]_i$.

Proof. Define paths b and w as in the lemma above. An upper bound on $\Pr(M = t)$ is obtained by considering the worst possible case: paths b and w differ in exactly d_{free} positions, and in these positions, the branch labels of w have a different sign

than the quantized channel symbols. For each of these d_{free} independent symbols, the coefficient of x^i in $p(x)$ is the probability that the contribution (maybe negative) to M equals i. Therefore, in this worst case, $\Pr(M = i)$ is the coefficient of x^i in $[p(x)]^{d_{\text{free}}}$. □

This bound is very loose because a worst-case assumption was made on the path metric differences, but it does estimate the probability that path metrics exceed a designed maximum possible range $\widehat{M}$. If $\ell < 1 + \log_2[d_{\text{free}}(2^{q-1} - 1)]$, the decoder will make incorrect decisions between trellis paths when $M > \widehat{M}$, which may occur very rarely.

If modulo arithmetic is not used for path and state metrics, the above results must be adjusted to consider state metrics only instead of path metrics. Then, every occurrence of d_{free} above would be replaced by the maximum weight, over all states s, of the least-weight trellis path from the all-zero state into s. Also, M would remain as originally defined: the difference between the maximum and minimum state metrics after a trellis level.

Myth. If $M > \widehat{M}$, then the decoder fails completely.

Galileo code simulations for $q = 5$ and 4 with short state metric registers having $\ell = 9$ and 8 bits, yielded the same BER because the odd path metric range overflow that occurred did not significantly affect the output, since Viterbi decoders are robust and tolerate occasional state metric disruptions. Further shortening of the state metric registers to 8 or 7 bits resulted in a graceful BER increase, as though q was being decreased. This behavior is expected because the overall trellis path metric resolution is the decoder parameter, affected by input quantization, that influences decisions in a Viterbi decoder.

References

[Clar81] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications*, New York: Plenum Press, 1981.

[Gilh71] K. S. Gilhousen, J. A. Heller, I. M. Jacobs, and A. J. Viterbi, "Coding Systems Study for High Data Rate Telemetry Links," Linkabit Corp., NASA Report CR-114278, Jet Propulsion Lab, January 1971.

[Hell71] J. A. Heller and I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communication," *IEEE Trans. Commun. Tech.*, vol. COM-19, pp. 835–848, October 1971.

[Shun90] C. B. Shung, G. Ungerboeck, P. H. Seigel, and H. K. Thapar, "VLSI Architectures for Metric Renormalization in the Viterbi Algorithm," in *Proc. of ICC '90*, March 1990.

[Stat88] J. Statman, G. Zimmerman, F. Pollara, and O. Collins, "A Long Constraint Length VLSI Viterbi Decoder for the DSN," *TDA Progress Report 42-95*, Jet Propulsion Laboratory, Pasadena, California, pp. 134–142, November 15, 1988.

[Yuen85] J. H. Yuen and Q. D. Vo, "In Search of a 2-dB Coding Gain," *TDA Progress Report 42-83*, Jet Propulsion Laboratory, Pasadena, California, pp. 26–33, November 15, 1985.