Viterbi and Omura, Principles of Digital Communication and Coding

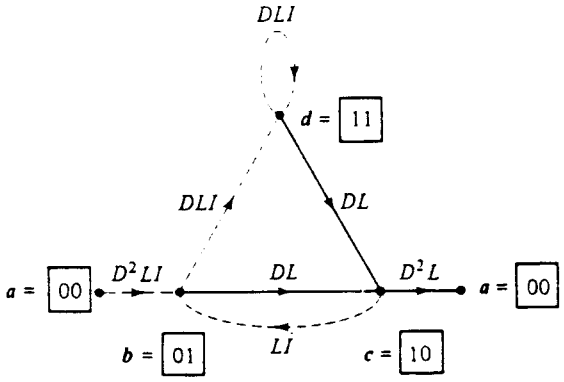


Figure 4.10 State diagram labeled with distance, length, and number of input "1"s.

generating function yields

$$T(D, L, I) = \frac{D^5L^3I}{1 - DL(1 + L)I}$$
$$= D^5L^3I + D^6L^4(1 + L)I^2 + D^7L^5(1 + L)^2I^3$$
$$+ \cdots + D^{5+k}L^{3+k}(1 + L)^kI^{1+k} + \cdots \quad (4.3.3)$$

Thus we have verified that of the two distance 6 paths, one is of length 4 and the other is of length 5, and both differ in two input bits from the all-zeros. Thus, for example, if the all-zeros was the correct path and the noise causes us to choose one of these incorrect paths, two bit errors will be made. Also, of the distance 7 paths, one is of length 5, two are of length 6, and one is of length 7; all four paths correspond to input sequences with three "1"s. If we are interested in the jth node level, clearly we should truncate the series such that no terms of power greater than L^j are included.

We have thus fully determined the properties of all code paths of this simple convolutional code. The same techniques can obviously be applied to any binary-symbol code of arbitrary constraint length and arbitrary rate b/n. However, for $b > 1$, each state equation of the type of (4.3.1) is a relationship among at most $2^b + 1$ node variables. In general, there will be $2^{b(K-1)}$ state variables and as many equations. (For further examples, see Probs. 4.6, 4.17, and 4.18.) In the next two sections we shall demonstrate how the generating function can be used to bound directly the error probability of a Viterbi decoder operating on any convolutional code on a binary-input, memoryless channel.

4.4 PERFORMANCE BOUNDS FOR SPECIFIC CONVOLUTIONAL CODES ON BINARY-INPUT, OUTPUT-SYMMETRIC MEMORYLESS CHANNELS

It should be reasonably evident at this point that the block length nB of a convolutional code is essentially irrelevant, for both the encoder and decoder complexity and operation depend only on the constraint length K, the code rate,

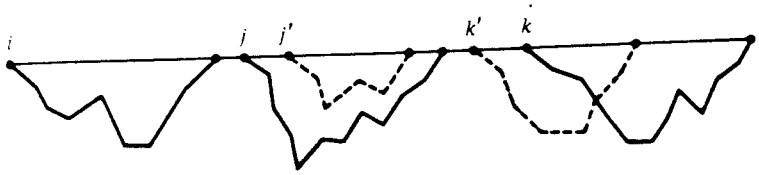


Figure 4.11 Example of error events.

and channel parameters; furthermore, the performance is a function of relative distances among signals, which may be determined from the code state diagram, whose structure and complexity depends strongly on the constraint length but not at all on the block length. Thus it would appear that block error probability is not a reasonable performance measure, particularly when, as is often the case, an entire message is convolutionally encoded as a single block, whereas in block coding the same message would be encoded into many smaller blocks. Ultimately, the most useful measure is *bit error probability* P_b which, as initially defined in Sec. 2.11, is the expected number of bit errors in a given sequence of received bits normalized by the total number of bits in the sequence.

While our ultimate goal is to upper-bound P_b, we consider initially a more readily determined performance measure, the *error probability per node*, which we denote P_e. In Fig. 4.11 we show (as solid lines) two paths through the code trellis. Without loss of essential generality, we take the upper all-zeros path to be correct, and the lower path to be that chosen by the maximum likelihood decoder. For this to occur, the correct path metric increments over the unmerged segments must be lower than those of the incorrect (lower solid line) path shown. We shall refer to these error events as node errors at nodes i, j, and k. On the other hand, the dotted paths which diverge from the correct path at nodes j' and k' may also have higher metric increments than the correct path over the unmerged segments, and yet not be ultimately selected because their accumulated metrics are smaller than those of the lower solid paths. We may conclude from this exposition that a necessary, but not sufficient, condition for a node error to occur at node j is that the metric of an incorrect path diverging from the correct path at this node accumulates higher metric increments than the correct path over the unmerged segment.

We may therefore upper-bound the probability of node error at node j by the probability that any path diverging from the correct path at node j accumulates higher total metric over the unmerged span of the path.

$$P_e(j) \leq \Pr\left[\bigcup_{\mathbf{x}'_j \in \mathcal{X}'(j)} \{\Delta M(\mathbf{x}'_j, \mathbf{x}_j) \geq 0\}\right] \tag{4.4.1}$$

where $\mathbf{x}'_j$ is an incorrect path diverging from the correct path at node j, $\mathcal{X}'(j)$ is the set of all such paths, known as the incorrect subset for node j, $\Delta M(\mathbf{x}'_j, \mathbf{x}_j)$ is the difference between the metric increment of this path and of the correct path $\mathbf{x}_j$ over the unmerged segment.

Employing the union bound, we obtain the more convenient, although looser, form

$$P_e(j) \leq \sum_{\mathbf{x}'_j \in \mathcal{X}'(j)} \Pr\left[\Delta M(\mathbf{x}'_j, \mathbf{x}_j) \geq 0\right] \tag{4.4.2}$$

$P_e(j) = \Pr\left[\bigcup_{\vec{x}'_j \in \mathcal{X}'(j)} \{\vec{x}'_j \text{ survives}\}\right] \leq \sum_{\vec{x}'_j \in \mathcal{X}'(j)} \Pr\{\vec{x}'_j \text{ survives}\}$

but $\Pr\{\vec{x}'_j \text{ survives}\} \leq \Pr[\Delta M(\vec{x}'_j, \vec{x}_j) \geq 0]$ since having a larger metric is a necessary but not sufficient condition to survive, so $P_e(j) \leq \sum_{\mathcal{X}'(j)} \Pr[\Delta M(\vec{x}'_j, \vec{x}_j) \geq 0]$

But each term of this summation is the pairwise error probability for two code vectors over the unmerged segment. For a binary-input channel, this is readily bounded as a function of the distance between code vectors over this segment. For, if the total Hamming distance between code vectors $\mathbf{x}_j$ and $\mathbf{x}'_j$ (over their unmerged segment) is $d(\mathbf{x}'_j, \mathbf{x}_j) = d$, we have from (2.9.19) that, for an output-symmetric channel, the pairwise error probability is bounded by the Bhattacharyya bound

$$P_d \leq \exp\left[d \ln \sum_y \sqrt{p_0(y)p_1(y)}\right] \tag{4.4.3}$$

where $p_i(y)$ is the conditional (channel transition) probability of output y given that the input symbol was $i (i = 0, 1)$. Equivalently, we may express this bound in the more convenient form

$$P_d \leq Z^d \tag{4.4.4}$$

where

$$Z \equiv \sum_y \sqrt{p_0(y)p_1(y)} \tag{3.4.12}$$

Thus given that there are $a(d)$ incorrect paths which are at Hamming distance d from the correct path over the unmerged segment, we obtain from (4.4.1) through (4.4.4)

$$\begin{aligned} P_e(j) &\leq \sum_{d=d_f}^{\infty} \Pr\left\{\begin{array}{l}\text{error caused by any one of } a(d)\\ \text{incorrect paths at distance } d\end{array}\right\} \\ &\leq \sum_{d=d_f}^{\infty} a(d)P_d \\ &\leq \sum_{d=d_f}^{\infty} a(d)Z^d \end{aligned} \tag{4.4.5}$$

where d_f is the minimum distance of any path from the correct path, which we called the free distance in the last section. Clearly (4.4.5) is a union-Bhattacharyya bound similar to those derived for block codes in Chap. 2.

We also found in the last section that the set of all distances from any one path to all other paths could be found from the generating function $T(D)$. For demonstration purposes, let us consider again the code example of Figs. 4.2*a*, 4.4, and 4.5. We found then that

$$\begin{aligned} T(D) = \frac{D^5}{1-2D} &= D^5 + 2D^6 + 4D^7 + \cdots + 2^{k-5}D^k + \cdots \\ &= \sum_{d=5}^{\infty} 2^{d-5}D^d \end{aligned}$$

Thus in this case $d_f = 5$ and $a(d) = 2^{d-d_f}$. The same argument can be applied to any binary code whose generating function we can determine by the techniques of

the last section. Thus we have in general that

$$T(D) = \sum_{d=d_f}^{\infty} a(d)D^d \tag{4.4.6}$$

and it then follows from (4.4.5) and (4.4.6) that

$$P_e(j) \leq T(D)\Big|_{D=Z} \tag{4.4.7}$$

We note also that this node error probability bound for a fixed convolutional code is the same for all nodes when $B = \infty$ and that this is also an upper bound for finite B.

Turning now to the bit error probability, we note that the expected number of bit errors, caused by any incorrect path which diverges from the correct path at node j, can be bounded by weighting each term of the union bound by the number of bit errors which occur on that incorrect path. Taking the all-zeros data path to be the correct path (without loss of generality on output-symmetric channels), this then corresponds to the number of "1"s in the data sequence over the unmerged segment. Thus the bound on the expected number of bit errors caused by an incorrect path diverging at node j is

expected number of detailed bit errors for a single node error

$$E[n_b(j)] \leq \sum_{i=1}^{\infty} \sum_{d=d_f}^{\infty} ia(d, i)P_d \leq \sum_{i=1}^{\infty} \sum_{d=d_f}^{\infty} ia(d, i)Z^d \tag{4.4.8}$$

where $a(d, i)$ is the number of paths diverging from the all-zeros path (at node j) at distance d and with i "1"s in its data sequence over the unmerged segment. But the coefficients $a(d, i)$ are also the coefficients of the augmented generating function $T(D, I)$ derived in the last section. For the running example, we have from (4.3.3) (with $L = 1$ since we are not interested in path lengths)

$$T(D, I) = \frac{D^5 I}{1 - 2DI} = D^5 I + 2D^6 I^2 + 4D^7 I^3 + \cdots + 2^{k-5} D^k I^{k-4} + \cdots$$

$$= \sum_{d=5}^{\infty} 2^{d-5} D^d I^{d-4}$$

and hence

$$a(d, i) = \begin{cases} 2^{d-5} & \text{for } i = d - 4,\ d \geq 5 \\ 0 & \text{otherwise} \end{cases}$$

In this case then,

$$E[n_b(j)] \leq \sum_{d=5}^{\infty} (d - 4)2^{d-5} Z^d = \frac{\partial T(D, I)}{\partial I}\Big|_{I=1,\ D=Z}$$

In general it should be clear that the augmented generating function can be expanded in the form

$$T(D, I) = \sum_{i=1}^{\infty} \sum_{d=d_f}^{\infty} a(d, i) D^d I^i \tag{4.4.9}$$

whose derivative at $I = 1$ is

$$\left.\frac{\partial T(D, I)}{\partial I}\right|_{I=1} = \sum_{i=1}^{\infty} \sum_{d=d_f}^{\infty} i a(d, i) D^d \tag{4.4.10}$$

Consequently, comparing (4.4.8) and (4.4.10), we have

$$E[n_b(j)] \leq \left.\frac{\partial T(D, I)}{\partial I}\right|_{I=1, D=Z} \tag{4.4.11}$$

This is an upper bound on the expected number of bit errors caused by an incorrect path diverging at any node j.

For a rate $1/n$ code, each node (branch) represents one bit of information into the encoder or decoder. Thus the bit error probability defined as the expected number of bit errors per bit decoded is bounded by

$$P_b(j) = E\{n_b(j)\} \leq \left.\frac{\partial T(D, I)}{\partial I}\right|_{I=1, D=Z} \tag{4.4.12}$$

as shown in (4.4.11). For a rate b/n code, one branch corresponds to b information bits. Thus in general

$$P_b(j) = \frac{E\{n_b(j)\}}{b} \leq \frac{1}{b} \left.\frac{\partial T(D, I)}{\partial I}\right|_{I=1, D=Z} \tag{4.4.13}$$

where Z is given by (3.4.12).

$$P_b \leq \lim_{L \to \infty} \frac{1}{bL} \sum_{l=1}^{L} E\{n_b(l)\} \leq \frac{1}{bL} \left.\frac{\partial T(D, I)}{\partial I}\right|_{I=1, D=Z}$$

4.5 SPECIAL CASES AND EXAMPLES

It is somewhat instructive to consider the BSC and the binary-input AWGN channel, special cases of the channels considered in the last section. Clearly the union-Bhattacharyya bounds apply with [see (2.11.6) and (2.11.7) and (3.4.15) and (3.4.17)]

$$(Z)_{\text{BSC}} = \sqrt{4p(1-p)} \tag{4.5.1}$$

and

$$(Z)_{\text{AWGN}} = e^{-\mathcal{E}_s/N_o} \tag{4.5.2}$$

We note also, as was already observed in Sec. 2.11, that if the AWGN channel is converted to the BSC by hard quantization for $\mathcal{E}_s/N_o \ll 1$, then

$$p \approx \frac{1}{2} - \sqrt{\frac{\mathcal{E}_s}{\pi N_o}}$$

in which case

$$-\ln Z \approx -\ln\left[\sqrt{1-4\mathcal{E}_s/\pi N_o}\right] \approx -\ln\left[1-\frac{2}{\pi}\frac{\mathcal{E}_s}{N_o}\right] \approx \frac{(2/\pi)\mathcal{E}_s}{N_o}$$

for a loss of $2/\pi$, or approximately 2 dB, in energy-to-noise ratio.

However, for these two special channels, tighter bounds can be found by obtaining the exact pairwise error probabilities rather than their Bhattacharyya bounds. For the BSC, we recall from (2.10.14) that, for unmerged segments at distance d from the correct path[7]

$$P_d = \begin{cases} \displaystyle\sum_{k=(d+1)/2}^{d} \binom{d}{k} p^k (1-p)^{d-k} & d \text{ odd} \\ \displaystyle\frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2} + \sum_{k=d/2+1}^{d} \binom{d}{k} p^k (1-p)^{d-k} & d \text{ even} \end{cases} \tag{4.5.3}$$

This can be used in the middle expressions of inequalities (4.4.5) and (4.4.8) to obtain tighter results than (4.4.7) and (4.4.12) (see also Prob. 4.10).

Similarly, for the binary-input AWGN channel, we have from (2.3.10) that the pairwise error probability for code vectors at distance d is

$$P_d = Q\left(\sqrt{2d\mathcal{E}_s/N_o}\right) \tag{4.5.4}$$

While we may substitute this in the above expressions in place of $Z^d = e^{-d\mathcal{E}_s/N_o}$, a more elegant and useful expression results from noting that (Prob. 4.8)

$$Q\left(\sqrt{x+y}\right) \le Q\left(\sqrt{x}\right)e^{-y/2} \qquad x \ge 0,\ y \ge 0 \tag{4.5.5}$$

Since $d \ge d_f$ we may bound (4.5.4) by

$$P_d \le Q\left(\sqrt{\frac{2d_f\mathcal{E}_s}{N_o}}\right) e^{-(d-d_f)\mathcal{E}_s/N_o} \tag{4.5.6}$$

which is tighter than the Bhattacharyya bound. Substituting in the middle terms of (4.4.5) and (4.4.8), then using (4.4.6) and (4.4.10), we obtain

$$\begin{aligned} P_e &\le Q\left(\sqrt{\frac{2d_f\mathcal{E}_s}{N_o}}\right) e^{d_f\mathcal{E}_s/N_o} \sum_{d=d_f}^{\infty} a(d)\, e^{-d\mathcal{E}_s/N_o} \\ &= Q\left(\sqrt{\frac{2d_f\mathcal{E}_s}{N_o}}\right) e^{d_f\mathcal{E}_s/N_o} T(D)\Big|_{D=e^{-\mathcal{E}_s/N_o}} \end{aligned} \tag{4.5.7}$$

[7] Ties are assumed to be randomly resolved. Note that unlike the block code case for which (2.10.14) holds, all probabilities here are for pairwise errors.

and

$$P_b = \frac{E[n_b(j)]}{b} \le \frac{1}{b} Q\left(\sqrt{\frac{2d_f \mathcal{E}_s}{N_o}}\right) e^{d_f \mathcal{E}_s / N_o} \sum_{i=1}^{\infty} \sum_{d=d_f}^{\infty} i a(d, i) D^d \Big|_{D = e^{-\mathcal{E}_s / N_o}}$$

$$= \frac{1}{b} Q\left(\sqrt{\frac{2d_f \mathcal{E}_s}{N_o}}\right) e^{d_f \mathcal{E}_s / N_o} \frac{\partial T(D, I)}{\partial I} \Big|_{I=1,\, D = e^{-\mathcal{E}_s / N_o}} \tag{4.5.8}$$

The last bound has been used very effectively to obtain tight upper bounds for the bit error probability on the binary-input AWGN channel for a variety of convolutional codes of constraint lengths less than 10. For, while the computation of $T(D, I)$ for a constraint length K code would appear to involve the analytical solution of $2^{b(K-1)}$ simultaneous algebraic equations (Sec. 4.3), the computation of $T(D, I)$ for fixed values of $D = Z$ and I becomes merely a numerical matrix inversion. Also since $T(D, I)$ is a polynomial in I with nonnegative coefficients and has a nondecreasing first derivative for positive arguments, the derivative at $I = 1$ can be upper-bounded numerically by computing instead the normalized first difference. Thus

$$\frac{\partial T(D, I)}{\partial I} \Big|_{I=1,\, D=Z} < \frac{T(Z, 1+\epsilon) - T(Z, 1)}{\epsilon} \qquad \epsilon \ll 1 \tag{4.5.9}$$

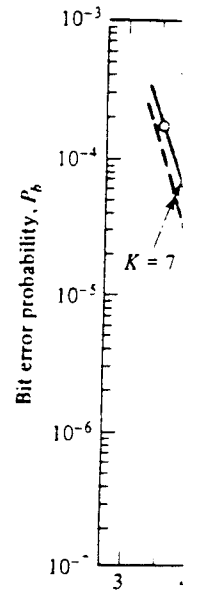
Even the numerical matrix inversion involved in calculating $T(D, I)$ for fixed D and I is greatly simplified by the fact that the diagonal terms of the state equations matrix [see (4.3.1) and Probs. 4.17 and 4.18] dominate all other terms in the same row. As a result, the inverse can be computed as a rapidly convergent series of powers of the given matrix (see Prob. 4.18). The results for optimum rate $\frac{1}{2}$ codes[8] of constraint length 3 through 8 are shown in Fig. 4.12. To assess the tightness of these bounds we show also in the figure the results of simulations of the same codes, but with output quantization to eight levels. For the low error probability region ($\mathcal{E}_b/N_o > 5$ dB), it appears that the upper bounds lie slightly below the simulation. The simulations should, in fact, lie above the exact curve because the quantization loss is on the order of 0.25 dB (see Sec. 2.8). This, in fact, appears to be the approximate separation between simulation and upper bounds, attesting to the accuracy of the bounds.

In all codes considered thus far, the generating function sequence

$$T(D) = \sum_{d=d_f}^{\infty} a(d) D^d \tag{4.5.10}$$

was assumed to converge for any value of D less than unity. That this will not always be true is demonstrated by the example of Fig. 4.13. For this code, the self

[8] The codes were selected on the basis of maximum free distance and minimum number of bit errors caused by incorrect paths at the free distance, i.e., minimum $a(d_f, i)$ (Odenwalder [1970]).



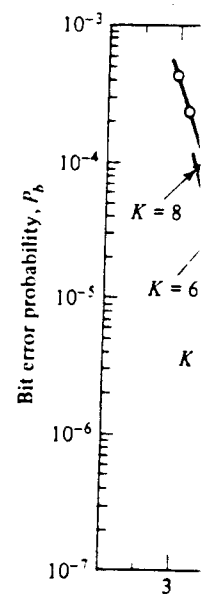
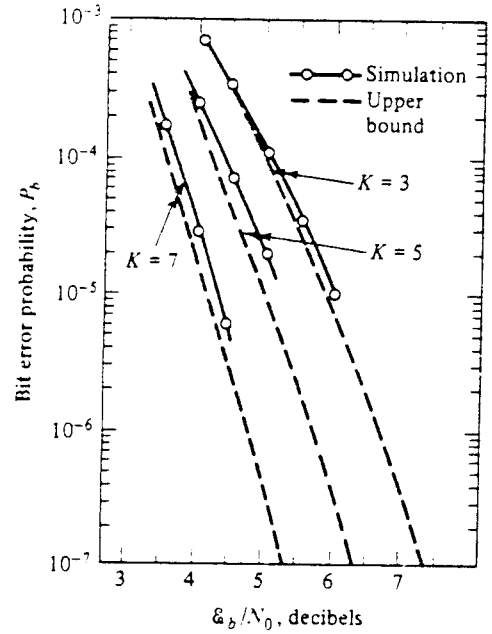
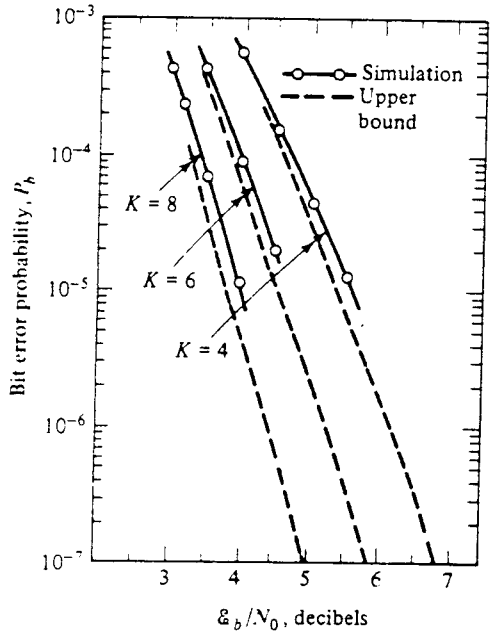


Figure 4.12 P_b as a fi
level quantization an
(*Courtesy of Heller a*



(*a*) $K = 3, 5, 7$



(*b*) $K = 4, 6, 8$

Figure 4.12 P_b as a function of $\mathscr{E}_b/N_o$ for Viterbi decoding of rate $\frac{1}{2}$ codes: simulations with eight-level quantization and 32-bit path memory (solid); upper bounds for unquantized AWGN (dotted). (*Courtesy of Heller and Jacobs [1971].*)

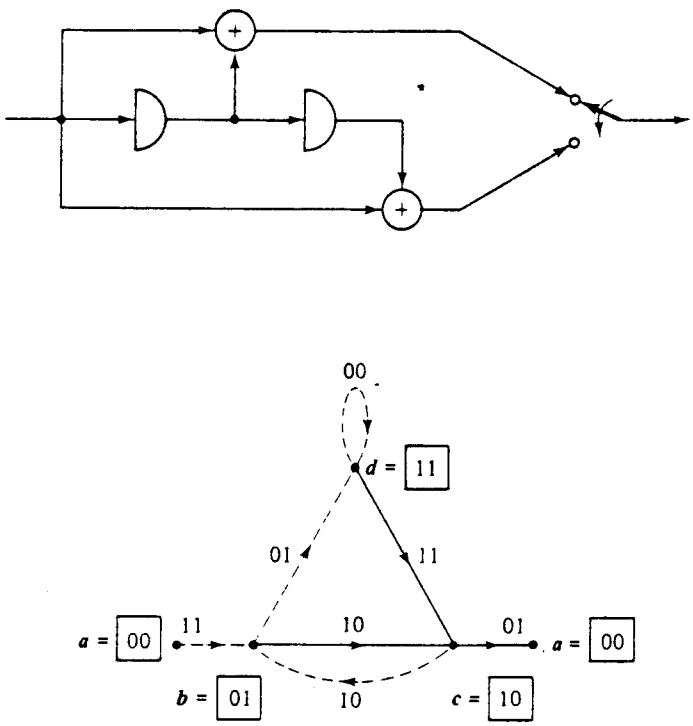


Figure 4.13 Encoder displaying catastrophic error propagation and its state diagram.

loop at state d does not increase distance, so that the path **abddd ... ddca** will be at distance 6 from the correct path no matter how many times it circulates about this self-loop. Thus it is possible on a BSC, for example, for a fixed finite number of channel errors to cause an arbitrarily large number of decoded bit errors. To illustrate in this case, for example, if the correct path is the all-zeros and the BSC produces two errors in the first branch, no errors in the next B branches and two errors in the $(B+1)$st branch, $B-1$ decoded bit errors will occur for an arbitrarily large B. For obvious reasons, such a code, for which a finite number of channel errors (or noise) can cause an infinite number of decoded bit errors, is called *catastrophic.*

It is clear from the above example that a convolutional code is catastrophic if and only if, for some directed closed loop in the state diagram, all branches have zero weight; that is, the closed loop path generating function is D^0. An even more useful method to ensure the avoidance of a catastrophic code is to establish necessary and sufficient conditions in terms of the code-generator sequences $\mathbf{g}_i$. For rate $1/n$ codes, Massey and Sain [1968] have obtained such conditions in terms of the code *generator polynomials* which are defined in terms of the generator

sequences as[9]

$$g_k(z) = 1 + g_{1,k}z + g_{2,k}z^2 + \cdots + g_{(K-1),k}z^{K-1} \qquad k = 1, 2, \ldots, n$$

In terms of these polynomials, the theorem of Massey and Sain (see Prob. 4.11) states that a fixed convolutional code is catastrophic if and only if all generator polynomials have a common polynomial factor (of degree at least one). Also of interest is the question of the relative fraction of catastrophic codes in the ensemble of all convolutional codes of a given rate and constraint length. Forney [1970] and Rosenberg [1971] have shown that, for a rate $1/n$ code, this fraction is $1/(2^n - 1)$, independent of constraint length (see Prob. 4.12). Hence generally, the search for a good code is not seriously encumbered by the catastrophic codes, which are relatively sparse and easy to distinguish.

One subclass of convolutional codes that are not catastrophic is that of the *systematic convolutional codes*. As with systematic block codes, systematic convo-

[9] In this context, z is taken to be an abstract variable, not a real number. The lowest order coefficient can always be taken as one without loss of optimality or essential generality.

Table 4.1 Maximum free distance of noncatastrophic codes

Rate $r = \frac{1}{2}$

K	Systematic† d_f	Nonsystematic d_f
2	3	3
3	4	5
4	4	6
5	5	7
6	6	8
7	6	10
8	7	10

Rate $r = \frac{1}{3}$

K	Systematic† d_f	Nonsystematic d_f
2	5	5
3	6	8
4	8	10
5	9	12
6	10	13
7	12	15
8	12	16

† With feed-forward logic.

lutional codes have the property that the data symbols are transmitted unchanged among the coded symbols. For a systematic rate b/n convolutional code, in each branch the first b symbols are data symbols followed by $n - b$ parity or coded symbols. The coded symbols are generated just as for nonsystematic codes, and consequently depend on the last Kb data symbols where Kb is the constraint length. Since data symbols appear directly on each branch in the state or trellis diagram, for systematic convolutional codes it is impossible to have a self-loop in which distance to the all-zeros path does not increase, and therefore these codes are not catastrophic.

In Sec. 5.7, we show that systematic feed-forward convolutional codes do not perform as well as nonsystematic convolutional codes.[10] There we show that, for asymptotically large K, the performance of a systematic code of constraint length K is approximately the same as that of a nonsystematic code of constraint length $K(1 - r)$ where $r = b/n$. Thus for rate $r = \frac{1}{2}$ and very large K, systematic codes have about the performance of nonsystematic codes of half the constraint length, while requiring exactly the same optimal decoder complexity.

Another indication of the relative weakness of systematic convolutional codes is shown in the free distance, d_f, which is the exponent of D in the leading term of the generating function $T(D)$. Table 4.1 shows the maximum free distance achievable with binary feed-forward systematic codes and nonsystematic codes that are not catastrophic. We show this for various constraint lengths K and rates r. As indicated by the results of Sec. 5.7, for large K the differences are even greater.

4.6 STRUCTURE OF RATE $1/n$ CODES AND ORTHOGONAL CONVOLUTIONAL CODES

While the weight or distance properties of the paths of a convolutional code naturally depend on the encoder generator sequences, both the unmerged path lengths and the number of "1"s in the data sequence for a particular code path are functions only of the constraint length, K, and rate numerator, b. Thus for example, for any rate $1/n$, constraint length 3 code [see (4.3.3)]

$$T_3(L, I) = \frac{L^3 I}{1 - L(1 + L)I} \tag{4.6.1}$$

To obtain a general formula for the generating function $T_K(L, I)$ of any rate $1/n$ code of constraint length K, we may proceed as follows. Consider the state just prior to the terminal state in the state diagram of a constraint length K code (see Fig. 4.10 for $K = 3$). The $(K - 1)$-dimensional vector for this state is $10 \ldots 0$. Suppose this were the terminal state and that when a path reached this state it was considered absorbed (or remerged) without the possibility to go on to either of the

[10] It can be shown (Forney [1970]) that for any nonsystematic convolutional code, there is an equivalent systematic code in which the parity symbols are generated with linear *feedback* logic.