On the Probability of Error of Convolutional Codes

Steven S. Pietrobon, *Member, IEEE*

***Abstract* — New upper and lower bounds and approximations on the sequence, event, first event, and bit error probabilities of convolutional codes are presented. Each of these probabilities are precisely defined and the relationship between them described. Some of the new bounds and approximations are found to be very close to computer simulations at very high error ratios. Simple modifications to the traditional union upper bound are also described for both hard and soft decision channels that allow better performance estimates to be made.**

***Index Terms* — Convolutional Codes, Bit Error Probability, Event Error Probability, Upper Bounds, Lower Bounds.**

I. Introduction

A FUNDAMENTAL QUESTION of any error control scheme is its probability of error. The probability of error determines the performance of the coding scheme. For convolutional codes, there are several ways of measuring the probability of error. The easiest to understand is the *bit error probability* P_b. This is the probability that a decoded information bit will be in error. Other criteria are also used.

With convolutional codes there does not appear to be a way of easily finding the code which minimizes the required E_b/N_0 (signal to noise ratio per information bit) for a specified error probability. Usually, code searches are performed whereby the performance of all possible codes are found and the best code is then selected. The criteria that have been commonly used are:

a) maximize the minimum free Hamming or squared Euclidean distance [1,2,3],

---

A paper submitted to the *IEEE Transactions on Information Theory* 31 January 1994. Revised 4 January 1996 and 2 April 1996. This work is based upon work performed under the sponsorship of the Australian Research Council. The material in this paper was partially presented at the International Symposium on Information Theory and its Applications, Sydney, Australia, November 1994.

The author is with the Satellite Communications Research Centre, University of South Australia, The Levels SA 5095, Australia. email: steven@spri.levels.unisa.edu.au

b) find the set of codes that satisfy a) and of those codes find the code that minimizes the number of nearest neighbors [4,5], or

c) use the transfer function upper bound to minimize the required E_b/N_0 for a specified bit error ratio [6,7].

Of the above criteria, c) is the most accurate. However, this is provided that the error ratio is very low, for example, around 10^{-3} (for constraint lengths up to seven) or 10^{-6} (for constraint lengths greater than seven). Since convolutional codes are commonly used as inner codes in concatenated systems [1] (so as to take advantage of the soft error correcting capability of a Viterbi decoder) the codes should be optimized for an error ratio of around 10^{-2} to 10^{-3} (the outer code takes this error ratio and produces an output error ratio of 10^{-5} to 10^{-10}).

If an output error ratio of 10^{-10} is desired from a concatenated scheme, only the low complexity codes found for an error ratio of 10^{-3} in [6] can be said with high confidence to be optimal. Where a concatenated output error ratio of 10^{-5} is desired (as in deep space communications) it is not known if the codes that have been found are optimal.

Thus it would be extremely advantageous if the required E_b/N_0 could be accurately found for high error ratios (around 10^{-2}) in a simple manner. A code search could then be performed to find the best codes for use in a concatenated system.

We have the problem though of finding this probability of error. At first glance the problem would appear to be simple as the encoders for many convolutional codes are simple in themselves (consisting of a few delay and exclusive OR gates). However, when it comes to determining the error probability (especially in an exact form) the reverse appears to be true, it becomes extremely complicated! One reason for this could be that the code sequences are of infinite length. For the hard decision channel a considerable breakthrough was achieved in [8]. A Markovian technique was used to determine the probability of error exactly.

In this paper we take a different approach to determining the probability of error. We start with a simple code and examine its *sequence error probability* P_s, *event error probability* P_e, *first event error probability* P_f, and finally its *bit error probability* P_b. Clear mathematical definitions of these probabilities are given. These definitions are used to derive new upper and lower bounds as well as the traditional upper and lower bounds. These bounds can be used in code searches to find better codes or prove that existing codes are the best. We shall also examine what is required to determine the exact error probability so that a better understanding of convolutional codes can be obtained.

II. Probability of Sequence Error

Let us consider a binary rate k/n linear convolutional code with ν binary delay elements in the encoder. Since the code is linear, we can assume the all zero sequence is transmitted over an additive white Gaussian noise channel and we wish to determine if we make an error at the decoder (using a maximum likelihood decoding algorithm). We say an event error occurs when the decoded path leaves the all–zero path and then returns. This can occur many times in a decoded sequence. A sequence error occurs when the decoded sequence is not the all zero path. Each possible sequence error has one or more event errors.

Let the j'th error event starting at time i be called $e_{j,i}$ ($1 \le j \le \infty$, $0 \le i \le \infty$).

Example 1: For the two state rate 1/2 convolutional code with generator polynomials given by $g_0 = 2_8$ and $g_1 = 3_8$ we have $e_{1,0} = 1+X+X^3$, $e_{2,0} = 1+X+X^2+X^5$, $e_{1,1} = X^2+X^3+X^5$, etc. Figure 1 shows the trellis for these error events.

We say that the trellis in Figure 1 has a length N of two (corresponding to two information bits). We define the sequence error probability $P_{s,N}$ for a length N trellis as

$$P_{s,N} = P\left[\bigcup_{i=0}^{N-1} \bigcup_{j=1}^{f(i)} e_{j,i}\right], \tag{1}$$

where f(i) is the number of event errors starting at time i. The meaning of this equation is best explained through an example.

Example 2: Let us consider the code given in Example 1 and the trellis in Figure 1. We see that

$$P_{s,2} = P\left(e_{1,0} \cup e_{1,1} \cup e_{2,0}\right). \tag{2}$$

This is the probability that $e_{1,0}$ or $e_{1,1}$ or $e_{2,0}$ is decoded due to errors on the channel. For a binary symmetric channel (BMC) with channel error probability p, it is not too difficult to determine this probability. We can express (2) as

$$P_{s,2} = P\left(e_{1,0}\right) + P\left(e_{1,1} \cap \overline{e_{1,0}}\right) + P\left(e_{2,0} \cap \overline{e_{1,0}} \cap \overline{e_{1,1}}\right). \tag{3}$$

The first term in (3) is very easy to determine. Since $e_{1,0}$ has a weight of three, an error will occur if two or three channel errors occur in the first, second, and fourth bit positions. Thus

$$P\left(e_{1,0}\right) = \binom{3}{2}p^2(1-p) + \binom{3}{3}p^3 = 3p^2 - 2p^3. \tag{4}$$

Note that $P_{s,1} = P\left(e_{1,0}\right)$. The second term in (3) is a little more difficult to determine. The correlation between $e_{1,1}$ and $e_{1,0}$ must be taken into account. This is the probability of channel errors

causing $e_{1,1}$ to be decoded and $e_{1,0}$ to not be decoded. Examining the trellis in Figure 1, we see that the fourth bit position is a 1 for both $e_{1,1}$ and $e_{1,0}$. Let us assume that a 0 is received with probability 1–p in this position. $e_{1,0}$ will not be decoded if there are zero or one error in the first and second positions (this occurs with probability $(1-p)^2 + 2p(1-p) = 1-p^2$). $e_{1,1}$ will be decoded if the third and sixth bit positions are in error (this occurs with probability p^2). The probability of all these events occurring is $(1-p)(1-p^2)p^2$. Now assume that a 1 is received in the fourth position with probability p. $e_{1,0}$ will not be decoded if there are no errors in the first and second positions and $e_{1,1}$ will be decoded if there are one or two errors in the third and sixth bit positions. The probability of these events occurring is $p(1-p)^2(2p(1-p)+p^2)$. Summing these two probabilities we obtain

$$P\left(e_{1,1} \cap \overline{e_{1,0}}\right) = 3p^2(1-p)^2. \tag{5}$$

The third term in (3) is actually easier to determine than the second term. This is due to the high correlation of $e_{1,0}$ and $e_{1,1}$ with $e_{2,0}$. For $e_{2,0}$ to be decoded there must be two or more errors in bit positions 1, 2, 3, and 6. We can easily see that three or four errors in these positions results in either $e_{1,0}$ or $e_{1,1}$ also being decoded (contrary to what we want). Thus only two bit error patterns need be considered. There are only four possible patterns, either one error in positions 1 or 2 and one error in positions 3 or 6. However, these errors only occur half the time since the number of errors is exactly half the Hamming weight of $e_{2,0}$. Position 4 must be a 0 (with probability 1–p) to ensure that either $e_{1,0}$ or $e_{1,1}$ are not selected. We don't need to consider the fifth bit position since an error in this position does not affect which path is selected. Therefore we have

$$P\left(e_{2,0} \cap \overline{e_{1,0}} \cap \overline{e_{1,1}}\right) = 2p^2(1-p)^3. \tag{6}$$

Summing (4) to (6) we obtain the exact expression for (3) as

$$P_{s,2} = 8p^2 - 14p^3 + 9p^4 - 2p^5. \tag{7}$$

If one considers a length N trellis, $P_{s,N}$ is equivalent to the block error probability, i.e., the probability that the decoded sequence is not the all zero sequence. We can immediately see from this that as N goes to infinity $P_s = P_{s,\infty} = 1$ for any non–zero p, i.e., as N increases it becomes more and more likely that we are going to have a sequence error. Thus, it would appear that P_s is not very useful. However, as will be shown in this and the following section P_s can provide very useful information on the determination of the event error probability P_e (despite the fact that it is equal to one).

To simplify the determination of P_s we note that

$$P_{s,N} = 1 - P\left[\bigcap_{i=0}^{N-1} \bigcap_{j=1}^{f(i)} \overline{e_{j,i}}\right]. \tag{8}$$

Using a symbolic programming language (such as Mathematica) the probability term in (8) can be systematically determined.

Example 3: For the two state code used in Examples 1 and 2 we have

$P_{s,3} = (0, 0, 13, -30, 25, -4, -5, 2)$,

$P_{s,4} = (0, 0, 18, -45, 18, 81, -150, 117, -45, 7)$,

$P_{s,5} = (0, 0, 23, -60, -18, 336, -714, 792, -525, 208, -45, 4)$,

$P_{s,6} = (0, 0, 28, -75, -\frac{317}{4}, \frac{3021}{4}, -1713, 2031, -\frac{2541}{2}, \frac{371}{2}, 327, -262, \frac{339}{4}, -\frac{43}{4})$,

where $(a_0, a_1, a_2, a_3, ...)$ are the coefficients of the polynomial $a_0 + a_1p + a_2p^2 + a_3p^3 + ...$ From these equations we can see that the coefficient for p^2 is equal to 5N–2 and for N > 2 the coefficient of p^3 is equal to –15(N–1). If we use the method in (3) for determining $P_{s,N}$ then we see that there are N terms involving $e_{1,i}$, N–1 terms involving $e_{2,i}$, N–2 terms involving $e_{3,i}$, etc. Thus normalising $P_{e,N}$ by N can be thought of determining the effect of just one $e_{j,i}$, for $1 \le j \le \infty$ (as N goes to infinity). We can see that each of the terms involving $e_{1,i}$ will always have three error patterns with two channel errors. Similarly, the terms involving $e_{2,i}$ will always have on average two error patterns with two channel errors. Thus the coefficient equation for p^2 is true for all N. If we assume the coefficient equation is true for p^3 for N > 6 then

$$\lim_{N \to \infty} \frac{P_{s,N}}{N} = 5p^2 - 15p^3 + \cdots \tag{9}$$

Note that in reality (9) must be equal to zero since $P_{s,\infty} \le 1$. This would indicate that (9) is an infinite degree polynomial with an infinite number of roots ranging continuously from zero to one. Fourier series of various waveforms can also have this property.

III. Probability of Event and First Event Error

The event error probability P_e is a much more useful criteria than than the sequence error probability P_s. We will discuss the first event error probability P_f later.

A. Event Error Probability

P_e is defined as the average probability that an error event diverges from the all zero path at time i and remerges some time later (assuming the all zero sequence is sent). Note that we only count

those error event paths that leave and enter the all zero path. For an infinite length sequence, P_e is the same for all i. We can estimate the event error probability of a length N decoded sequence ε with N(ε) event errors as:

$$\hat{P}_{e,N} = \frac{N(\epsilon)}{N}. \tag{10}$$

We have that $P_e = P_{e,\infty}$ and $\hat{P}_e = \hat{P}_{e,\infty}$. The exact event error probability for a length N trellis can therefore be expressed as

$$P_{e,N} = \frac{1}{N} \sum_{\epsilon \in E} N(\epsilon) P(\epsilon), \tag{11}$$

where E is the set of possible decoded error sequences and P(ε) is the probability of an error sequence ε being decoded. Determining (11) exactly appears to be much more difficult than finding $P_{s,N}$. The extra complication comes from having to know which error events the decoder selects for a particular channel error sequence. For example, a long single event error may overlap two shorter event errors. For a particular channel error sequence, the more likely error events needs to be selected, something which did not matter in determining $P_{s,N}$ (since multiple event errors were all counted as one sequence error). Since each ε is counted only once in determining $P_{s,N}$, we have the following theorem:

Theorem 1: The event error probability $P_{e,N}$ satisfies the following lower bound

$$P_{e,N} \geq \frac{P_{s,N}}{N}. \tag{12}$$

Proof: The proof follows from (11), the fact that $N(\epsilon) \geq 1$, and the fact that we can express $P_{s,N}$ as

$$P_{s,N} = \sum_{\epsilon \in E} P(\epsilon). \tag{13}$$

For $P_e = P_{e,\infty}$, (12) is not a very good lower bound since the right hand side is equal to zero!

Now that we've shown a way of determining $P_{e,N}$ exactly (albeit in a very complicated fashion) we can now examine some other bounds on P_e. We can see from (10) that for a rate 1/n convolutional code with memory m an upper bound for $P_{e,N}$ is

$$P_{e,N} \leq \frac{1}{N} \left\lfloor \frac{N + \nu}{L_{min}} \right\rfloor, \tag{14}$$

where L_{min} is the shortest length error event path (note that $L_{min} = \nu + 1$ for rate 1/n codes). The right hand side of (14) is the maximum number of event errors we can have in a length N trellis. Taking N to infinity we obtain a simple upper bound for P_e.

$$P_e \leq \frac{1}{L_{min}}. \tag{15}$$

To determine P_e for p = 1 (which occurs when the received sequence is inverted on a noiseless channel) we define the *density* ϱ of an event error e_j (the density is the same for all i) as

$$\rho(e_j) = \frac{w(e_j)}{L(e_j)}, \tag{16}$$

where $w(e_j)$ is the Hamming weight and $L(e_j)$ is the length of e_j. We say that e_j is *densest* if $\varrho(e_j)$ is maximum out of all the possible error events.

Theorem 2: The event error probability P_e for p = 1 is given by

$$P_e = \frac{|\Delta|}{\sum_{j \in \Delta} L(e_j)}, \tag{17}$$

where Δ is the set of error events with maximum density.

Proof: For p = 1, the received sequence consists of all ones. The decoder will find the code sequence that is closest to the received sequence. The error events with the largest number of ones per event length will then be selected, i.e., the densest error events. If there are two or more error events with the same maximum density, we must average the lengths of these events.

Example 4: For the two state code used in previous examples the densest error event is e_1 with a density of 1.5. Thus $P_e = 0.5$ for p = 1 (which is equal with the upper bound in (15)).

If the all ones sequence is a code word the densest error event must be the all ones sequence (with a maximum density of n for a rate 1/n code). The all ones sequence must have a length of infinity, implying that (17) is equal to zero. This situation occurs for all rate 1/n codes that are rotationally invariant to a 180° phase rotation of a signal set (where by definition the code must have the all ones code sequence).

B. First Event Error Probability

The first event error probability P_f is defined as the probability of leaving the all–zero path at time i = 0 (assuming the trellis starts at time i = 0). We can estimate P_f by averaging the number of times a Viterbi decoder chooses a non–zero path decision for state zero. In effect, a non–zero path decision at time i is equivalent to leaving the all–zero path at time i = 0 (by reversing the sequences

in time). In comparison, to estimate P_e, we need to wait for the decoder to produce a decoded sequence. Since the decision at time 0 is effectively independent of later decisions, we can express the first event error probability as

$$P_{f,N} = P\left[\bigcup_{j=1}^{F(N)} e_{j,0}\right], \tag{18}$$

where F(N) is the number of error events starting at time i = 0.

Simple upper and lower bounds can be found from (18). Using the union bound, an upper bound for $P_{f,N}$ is

$$P_{f,N} \leq \sum_{j=1}^{F(N)} P(e_{j,0}), \tag{19}$$

and a lower bound for for $P_{f,N}$ is

$$P_{f,N} \geq P(e_d), \tag{20}$$

where $d = d_{min}$ is the minimum distance of the length N code and e_d is one of the error events of weight d.

We can see that in any decoded sequence, the error events that are selected will “block” other error events that leave the all–zero path. Thus,

$$P_{e,N} \leq P_{f,N}, \tag{21}$$

and we form the traditional union bound for $P_{e,N}$ [9] by substituting (19) into (21)

$$P_{e,N} \leq \sum_{j=1}^{F(N)} P(e_{j,0}), \tag{22}$$

It must be said that (22) is a very poor bound for $P_{e,N}$. When p = 1, the bound is equal to N, which is much greater than 1. This bound does not take into account the correlation of the sequences for high p (or even low p for some channels).

Example 5: For the two state code given in the previous examples we have

$$P_{f,2} = (0, 0, 5.5, -7, 2.5),$$

$$P_{f,3} = (0, 0, 5.5, -2, -14, 20, -10.5, 2),$$

$$P_{f,4} = (0, 0, \frac{11}{2}, 4, -\frac{157}{4}, \frac{249}{4}, -\frac{183}{4}, \frac{67}{4}, -\frac{5}{2}),$$

We can see that the coefficient of p^2 is 5.5 which is 10% greater than the expected value for P_e. Thus, (21) is not asymptotically identical for low p for this code. For low to medium p, the bound is not very close to the simulation of P_e.

C. Lower Bounds and Approximations for P_e

The determination of a simple lower bound for P_e better than (12) appears to be very difficult for a number of reasons. One may think that a lower bound on P_e based on the probability of the most likely error event would be sufficient (i.e., $P(e_j)$ where e_j has the smallest weight). We can immediately see from (15) that this cannot be so. For $p = 1$, $P(e_j) = 1$ which can be greater than the upper bound in (15) (since $L(e_j)$ is greater than one for a trellis without parallel transitions). The following is a derivation of an approximation for P_e.

Consider a length N trellis and error event e_j with length $L = L(e_j)$ and probability $P = P(e_j)$. Let the code sequences consist of all possible combinations of $e_{j,iL}$. The maximum number of e_j that can fit in the trellis is

$$M = \left\lfloor \frac{N + \nu}{L} \right\rfloor . \tag{23}$$

These error events are independent of each other, so the probability of each sequence with i event errors is $P^i(1 - P)^{M-i}$. Thus, counting the probability of these code sequences, one would think that a lower bound on $P_{e,N}$ is

$$\begin{aligned} P'_{e,N} &= \frac{1}{N} \sum_{i=1}^{M} i \binom{M}{i} P^i (1 - P)^{M-i} \\ &= \frac{M}{N} \sum_{i=1}^{M} \binom{M-1}{i-1} P^i (1 - P)^{M-i} \\ &= \frac{MP}{N} \sum_{j=0}^{M-1} \binom{M-1}{j} P^j (1 - P)^{M-1-j} \\ &= \frac{P(e_j)}{N} \left\lfloor \frac{N + \nu}{L(e_j)} \right\rfloor . \end{aligned} \tag{24}$$

Taking the limit as N goes to infinity we obtain

$$P'_e = \frac{1}{L(e_j)} P(e_j). \tag{25}$$

For 180° rotationally invariant codes we can see that (25) violates (17) for all finite length e_j (where P'_e is finite for $p = 1$, where as shown previously P_e is equal to zero). This problem arises because two or more of the error events may be decoded into a longer single error event for some error sequences. Thus P_e could be smaller than (25). This is illustrated for 180° rotationally invariant codes where the all ones code sequence will always be decoded for $p = 1$.

Despite the problems of (25) being an actual lower bound for P_e, we can easily calculate it to see how it compares with other bounds and computer simulations. In this case we should choose the most likely error event (i.e., the path with the smallest weight d_{min}) with the shortest length.

Example 6: For the two state code used in previous examples the most likely and only error event is e_1. This event also has the shortest length. Thus

$$P_e' = 1.5p^2 - p^3. \tag{26}$$

For small p, P_e' is not very tight. For p = 1 though, $P_e' = P_e = 0.5$.

One can obtain further improvements to P_e' by "filling in" the gaps of the sequences used in the determination of P_e'. For example, for the two state code the code sequences are combinations of $e_{1,0}$, $e_{1,2}$, $e_{1,4}$, etc. The fill in events are $e_{1,1}$, $e_{1,3}$, $e_{1,5}$, etc. These fill in events occur provided that $e_{1,2i}$ don't occur, or in general if the events that intersect with $e_{1,2i-1}$ don't occur (in this case $e_{1,2i-2}$ and $e_{1,2i}$). Thus, we can we write an "approximation" of P_e for a length $L_j = L(e_j)$ error event as

$$P_e \approx \frac{1}{L_j}\left[P(e_{j,0}) + P(\overline{e_{j,0}} \cap e_{j,1} \cap \overline{e_{j,L_j}}) + \cdots + P\left(\bigcap_{i=0}^{L_j-2} \overline{e_{j,i}} \cap e_{j,L_j-1} \cap \overline{e_{j,L_j}}\right)\right]. \tag{27}$$

Example 7: For the two state code the approximation of P_e using (27) is (with j = 1)

$$\begin{aligned} P_e &\approx \frac{1}{2}\left(P(e_{1,0}) + P(\overline{e_{1,0}} \cap e_{1,1} \cap \overline{e_{1,2}})\right) \\ &= (0, 0, 3, -6, 5, -2.5, 1). \end{aligned} \tag{28}$$

Notice that the coefficient of p^2 in (28) is the same as that produced for each of the $e_{1,i}$ terms in $P_{s,N}$. We can continue the "filling in" process with other error events with similar equations to (27) to improve the approximation of P_e.

Example 8: For the two state code we can include three additional terms for e_2 (since e_2 has length three). These terms are

$$\begin{aligned} &P(\overline{e_{1,0}} \cap \overline{e_{1,1}} \cap \overline{e_{1,2}} \cap \overline{e_{1,3}} \cap e_{2,1}) + P(\overline{e_{1,0}} \cap \overline{e_{1,1}} \cap \overline{e_{1,2}} \cap \overline{e_{1,3}} \cap \overline{e_{2,0}} \cap e_{2,1} \cap \overline{e_{2,3}}) \\ &\quad + P(\overline{e_{1,1}} \cap \overline{e_{1,2}} \cap \overline{e_{1,3}} \cap \overline{e_{1,4}} \cap \overline{e_{2,0}} \cap \overline{e_{2,1}} \cap e_{2,2} \cap \overline{e_{2,3}} \cap \overline{e_{2,4}}) \\ &= (0, 0, 6, -36, 92, -\frac{517}{4}, \frac{423}{4}, -46, \frac{113}{32}, \frac{211}{32}, -\frac{47}{16}, \frac{3}{16}, \frac{5}{32}, -\frac{1}{32}). \end{aligned} \tag{29}$$

The approximation to P_e is then obtained by dividing (29) by three and adding it to (28). We see that the coefficient for p^2 is 5 (the same as in P_s) and the coefficient for p^3 is –18. Thus for low p, the approximation should be very close to the actual P_e and will be asymptotically identical. Figure

2 plots the approximation of P_e obtained from (28), and that obtained from (28) and (29). The plot for including the effects of e_1 and e_2 (j = 2) is almost the same as the simulation for $p < 0.15$.

D. Upper Bounds

We can obtain an even tighter upper bound to P_e. We can expand $P_{f,N}$ as

$$P_{f,N} = \sum_{j=1}^{F(N)} P\left(\bigcap_{k=1}^{j-1} \overline{e_{k,0}} \cap e_{j,0} \right). \tag{30}$$

For the $e_{j,0}$ term in (30) only the error events starting at time i = 0 are excluded. We can expand this to include all the event errors that $e_{j,0}$ covers (the start and finish of these events lie between the start and finish of $e_{j,0}$). That is, we are excluding additional events that would prevent $e_{j,0}$ from being decoded. Let $P_{e,N}^u$ be this new upper bound on $P_{e,N}$.

Example 9: For the two state code we have

$$P_{e,1}^u = P(e_{1,0}) = 3p^2 - 2p^3,$$

$$P_{e,2}^u = P_{e,1}^u + P(\overline{e_{1,0}} \cap \overline{e_{1,1}} \cap e_{2,0}) = P_{e,1}^u + 2(1-p)^3p^2,$$

$$P_{e,3}^u = P_{e,2}^u + P(\overline{e_{1,0}} \cap \overline{e_{1,1}} \cap \overline{e_{1,2}} \cap \overline{e_{2,0}} \cap \overline{e_{2,1}} \cap e_{3,0}) = P_{e,2}^u + (1-p)^4p^3,$$

$$P_{e,4}^u = P_{e,3}^u + 2(1-p)^6p^3.$$

For N > 1, the coefficient of p^2 is five, exactly what is expected. For N > 3, the coefficient of p^3 is –5, and for N > 5, the coefficient of p^4 is –8. Notice that the additional term for each increasing N is only a function of the nearest error path. This is due to the high correlation of the longer sequences with the shorter sequences. That is, most of the error sequences have already been counted for the shorter error events and there are only very few additional error events that need to be counted. The best approximation for P_e is N = 2, but even then the bound is not too close. The approximate lower bound for j = 2 in Example 8 is much closer to P_e.

The example above indicates that an approximate upper bound may be found that is as simple to determine as (22), that is more accurate, and that will not diverge. For a hard decision channel we let this approximation be

$$P_e^a = P(e_1) + \sum_{w=d_{min}}^{\infty} \frac{N_w}{2^{(w+1) \bmod 2}} \binom{w}{\lceil w/2 \rceil} p^{\lceil w/2 \rceil}(1-p)^{\lfloor w/2 \rfloor}, \tag{31}$$

where N_w is the number of paths of weight w excluding e_1. Equation (31) is actually simpler to determine than (22) since only one binomial term needs to be determined for each set of paths of weight w.

Example 10: For the two state code this approximation gives the coefficient of p^2 as 6, 20% greater than the actual value (this is the same value as would be found from (22)). The approximation is also greater than one for large values of p. Compared to a computer simulation, this approximation is not very tight, but it is better than (22).

The approximation can be improved (with a small increase in the complexity of its computation) by making (31) more like the terms in Example 9. e.g.,

$$P_{e,N}^{a'} = P(e_1) + \sum_{j=2}^{F(N)} \frac{1}{2^{(w_j+1) \bmod 2}} \left(\begin{matrix} w_j \\ \lceil w_j/2 \rceil \end{matrix} + 1 \right) p^{\lceil w_j/2 \rceil} (1-p)^{l_j - \lceil w_j/2 \rceil}, \tag{32}$$

where w_j and l_j are the weight and number of channel bits of e_j, respectively. We have decreased the number of possible combinations in how we make errors and introduced more (1–p) factors.

Example 11: Figure 3 plots $P_{e,N}^{a'}$ for N = 2 to 4. For N > 1, the coefficient of p^2 is 5, making the curves very accurate for low p (although this may be just a coincidence for this code). Notice that the curves do not exceed 1. This is due to the (1–p) factors reducing the probability for high values of p.

Up to this point we have only investigated hard decision channels, since these channels are easier to analyze. Since (31) and (32) are fairly simple to compute, we can develop a similar equation for a soft decision channel with infinite quantization. For either BPSK or QPSK modulation we have for a single event

$$P(e_j) = p_q(w_j) = Q\left[\sqrt{\frac{2w_j k}{n}\frac{E_b}{N_0}}\right], \tag{33}$$

where w_j is the Hamming weight of e_j and E_b/N_0 is the energy per bit to noise density ratio. We let an approximation of P_e for soft decision errors be

$$P_e^q = p_q(w_1) + \sum_{w=d_{min}}^{\infty} N_w \left(p_q(w) - p_q(w+1) \right). \tag{34}$$

IV. Probability of Bit Error

An estimate of the probability of bit error P_b for a length N trellis is

$$\hat{P}_{b,N} = \frac{N_b(\epsilon)}{N}, \tag{35}$$

where N_b is the number of bit errors for an error sequence ε. The exact bit error probability for a length N trellis can therefore be expressed as

$$P_{b,N} = \frac{1}{N} \sum_{\epsilon \in E} N_b(\epsilon) P(\epsilon). \tag{36}$$

Since $N_b(\epsilon) \geq N(\epsilon)$ then

$$P_{b,N} \geq P_{e,N}. \tag{37}$$

Determining the exact $P_{b,N}$ for a length N trellis is even more difficult than finding $P_{e,N}$. One needs to determine only those channel error sequences that decode only into each error sequence.

A. Lower Bounds

The approximate lower bounds that were developed in the previous section can be used with some modification as true lower bounds for $P_b = P_{b,\infty}$.

Theorem 3: The bit error probability P_b of a convolutional code is lower bounded by

$$P_b \geq \frac{N_b(e_j) P(e_j)}{L(e_j)}. \tag{38}$$

where e_j is the error event with the smallest bit density $N_b(e_j)/L(e_j)$.

Proof: The derivation of (38) is similar to the derivation of (25), except that we count the number of bits along the error event path. We choose the event with the lowest bit density as this guarantees that the bound is satisfied. If any other events were to be selected they would have a higher value than that given by the lower bound. (For event error probability the lowest event density is defined as $1/L(e_j)$. Since $L(e_j)$ can be equal to infinity the lower bound will equal 0.)

We can keep adding terms to (38) (from lower to higher bit density) in a similar way to the lower bound approximation of P_e to obtain an even tighter lower bound on P_b.

Example 12: We use the two state code on a hard decision channel as in previous examples. We form the lower bound with one information bit times the term in (28) and two information bits times one third the term in (29). We obtain

$$P_{b,2}^{l} = (0, 0, 7, -30, \frac{199}{3}, -\frac{266}{3}, \frac{143}{2}, -\frac{92}{3}, \frac{113}{48}, \frac{211}{48}, -\frac{47}{24}, \frac{1}{8}, \frac{5}{48}, -\frac{1}{48}). \tag{39}$$

Here the coefficient of p^2 is 7 (the same as the exact solution found in [8]). For j = 2, the lower bound is fairly close to simulations for low p. For higher p though, the bound diverges from the simulation.

B. Upper Bounds

The traditional upper bound is formed from the union bound as given by (22). That is,

$$P_{b,N} \le \sum_{j=1}^{F(N)} N_b(e_j)P(e_j). \tag{40}$$

For some error event e_j we can see that there might be some error sequences that decode into a sequence with more bit errors. However, since all possible error patterns for all the events are counted, the upper bound in (40) is satisfied.

For high channel error ratios though, it is well known that (40) is not very tight. To make the bound tighter, we can use the tighter upper bounds and approximations to P_e that were found in the previous section. These equations may not be true upper bounds on P_e because we are assuming that all the channel error sequences for each error event are decoded only to that error event (which may not be the case). Thus we say these equations are approximations.

In forming these equations the first term e_1 should correspond to the path with one information bit (or the impulse response of the encoder). This ensures, for p = 1, that the approximation to P_b is equal to 1 for this value (otherwise the approximation will be greater than one).

Example 13: We will use the upper bound approximation that was found for P_e in the previous section. Using the two state code as in Example 9 we have

$$P_{b,1}^a = P(e_{1,0}) = 3p^2 - 2p^3,$$

$$P_{b,2}^a = P_{b,1}^a + 2P(\overline{e_{1,0}} \cap \overline{e_{1,1}} \cap e_{2,0}) = P_{b,1}^a + 4(1-p)^3p^2,$$

$$P_{b,3}^a = P_{b,2}^a + 3P(\overline{e_{1,0}} \cap \overline{e_{1,1}} \cap \overline{e_{1,2}} \cap \overline{e_{2,0}} \cap \overline{e_{2,1}} \cap e_{3,0}) = P_{b,2}^a + 3(1-p)^4p^3,$$

$$P_{b,4}^a = P_{b,3}^a + 8(1-p)^6p^3.$$

The coefficient of p^2 is the same as determined by the lower bound, i.e., 7. We plot $P_{b,N}^a$ for N = 2 to 4 in Figure 4. We see that $P_{b,2}^a$ is closer to the simulation than the lower bound. This is opposite to P_e where the lower bound approximation was closer than the upper bound.

The upper bound approximations for P_e given in (31), (32), and (34) can also be used for P_b, with the appropriate coefficients to count the number of bits per error event. A slight modification

of (34) was successfully used in [10] to find the best low rate convolutional codes for a concatenated coding scheme.

Example 14: Again, we will use the tightest approximation as found in Example 11. Figure 5 plots the approximation for N = 2 to 4. We see that the approximation is very tight for $p < 0.07$ and N = 2, becoming slightly looser for higher p.

V. Conclusions

We have investigated the sequence, first event, event, and bit error probability of convolutional codes. Using the two state code on a binary symmetric channel we have demonstrated many aspects, some of them surprising, of the probability of error of convolutional codes.

Using P_s as a lower bound to P_e, the first coefficient (and perhaps the second) for the two state code was exactly found. It was shown that a non–trivial lower bound for P_e could not be found using the probability of a single event error (although it was shown that a lower bound could be found for P_f). Tighter upper bounds that were better than the traditional union upper bound were investigated. Using the two state code as an example, it was shown that we can easily tighten the traditional upper bounds without a large increase in complexity (for both hard and soft decision channels). A new lower bound for P_b has also been given.

It was found for the two state code that the lower bound approximation is closer to P_e than the tightest upper bound. However, for P_b the situation reverses, with the upper bound approximation being closer than the tightest lower bound. Since P_b is usually the desired parameter, the upper bound approximation is a good indication of the code's performance (if it can be determined). Similarly to P_e, we can tighten the traditional upper bound to P_b very simply. These new approximations could be used in code searches to find better codes than before, especially at higher error ratios.

Acknowledgment

The author would like to thank Wade Farrell of the Satellite Communications Research Centre for providing the simulation results used in this paper.

REFERENCES

[1] J. P. Odenwalder, "Optimal decoding of convolutional codes," Ph.D. thesis, University of California, LA, 1970.

[2] K. J. Larsen, "Short convolutional codes with maximum free distance for rates 1/2, 1/3, and 1/4," *IEEE Trans. Inform. Theory*, vol. IT–19, pp. 371–372, May 1973.

[3] E. Paaske, "Short binary convolutional codes with maximal free distance for rates 2/3 and 3/4," *IEEE Trans. Inform. Theory*, vol. IT–20, pp. 683–689, Sep. 1974.

[4] R. Johannesson and E. Paaske, "Further results on binary convolutional codes with an optimum distance profile," *IEEE Trans. Inform. Theory*, vol. IT–24, pp. 264–268, Mar. 1978.

[5] S. S. Pietrobon, R. H. Deng, A. Lafanechère, G. Ungerboeck, and D. J. Costello, Jr., "Trellis–coded multidimensional phase modulation," *IEEE Trans. Inform. Theory*, vol. 36, pp. 63–89, Jan. 1990.

[6] P. J. Lee, "New short constraint length, rate 1/N convolutional codes which minimize the required SNR for given desired bit error rates," *IEEE Trans. Commun.*, vol. COM–33, pp. 171–177, Feb. 1985.

[7] P. J. Lee, "Further results on rate 1/N convolutional code constructions with minimum required SNR criteria," *IEEE Trans. Commun.*, vol. COM–34, pp. 395–399, Apr. 1986.

[8] M. R. Best, M. V. Burnashev, Y. Lévy, A. Rabinovich, P. C. Fishburn, A. R. Calderbank, and D. J. Costello, Jr., "On a technique to calculate the exact performance of a convolutional code," *IEEE Trans. Inform. Theory*, vol. 41, pp. 441–447, Mar. 1995.

[9] A. J. Viterbi and J. K. Omura, "Principles of digital communication and coding," McGraw–Hill, New York, 1979.

[10] S. S. Pietrobon, "Low rate convolutional codes optimised for use in concatenated codecs," *Int. Symp. on Inform. Theory and its Applications*, Sydney, NSW, pp. 19–24, Nov. 1994.

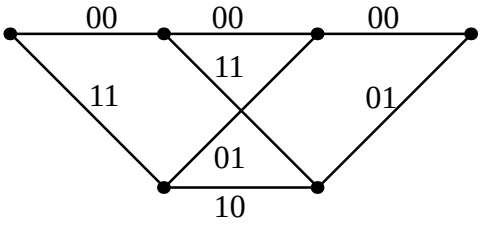


Figure 1: Trellis for two state rate 1/2 convolutional code and N = 2.

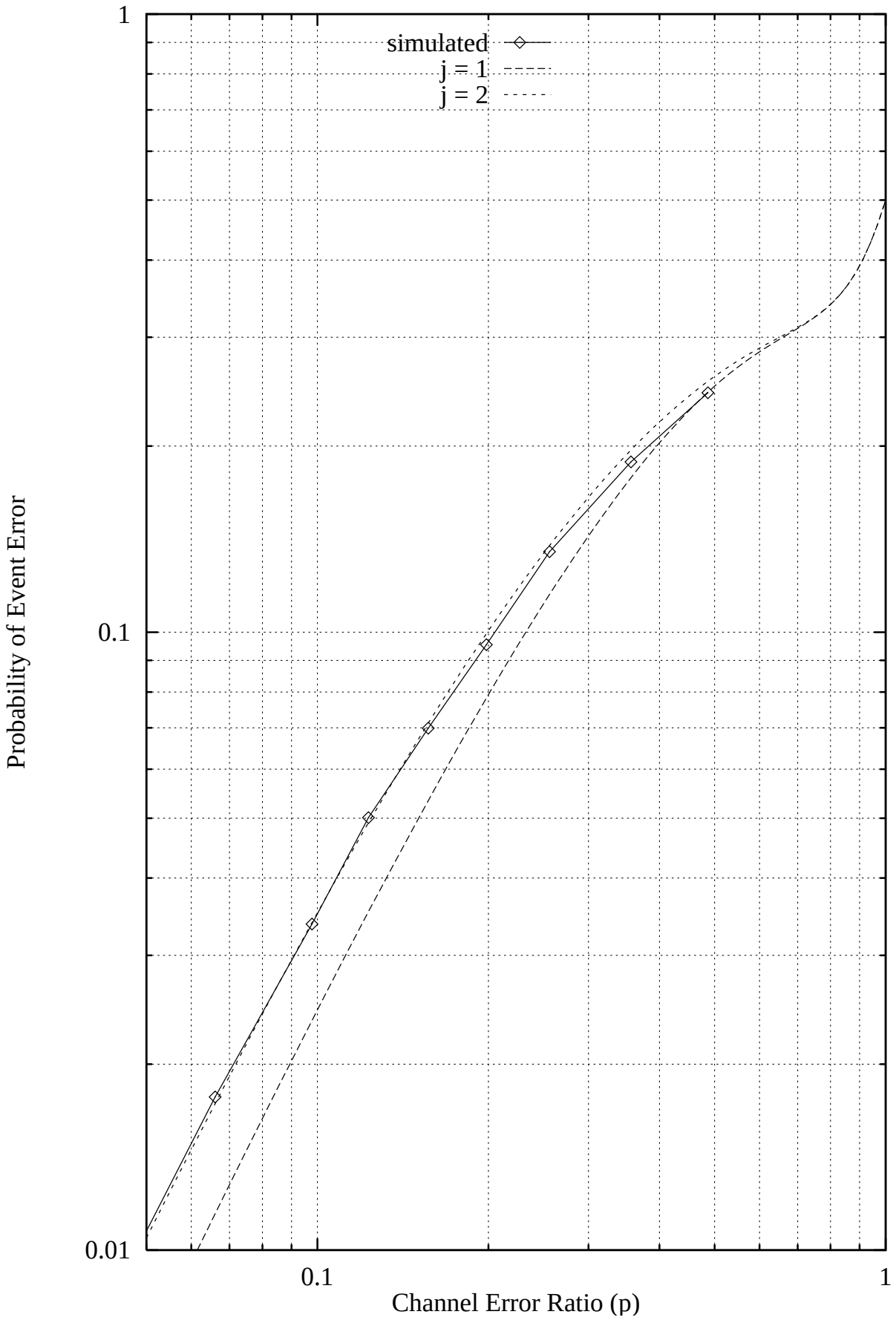


Figure 2: Probability of Event Error (lower bound approximation) versus p for two state rate 1/2 convolutional code on BSC.

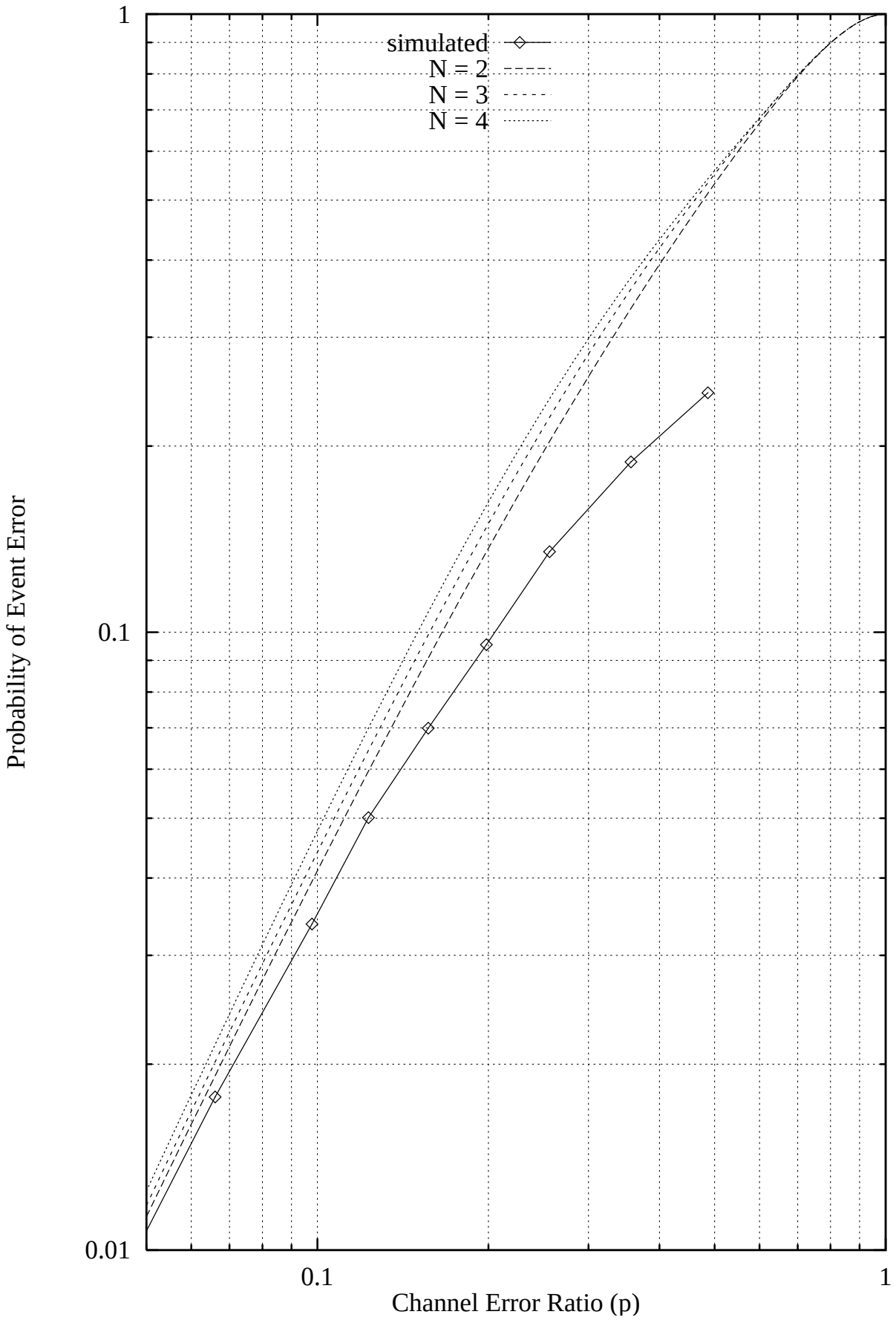


Figure 3: Probability of Event Error (union bound approximation) versus p for two state rate 1/2 convolutional code on BSC.

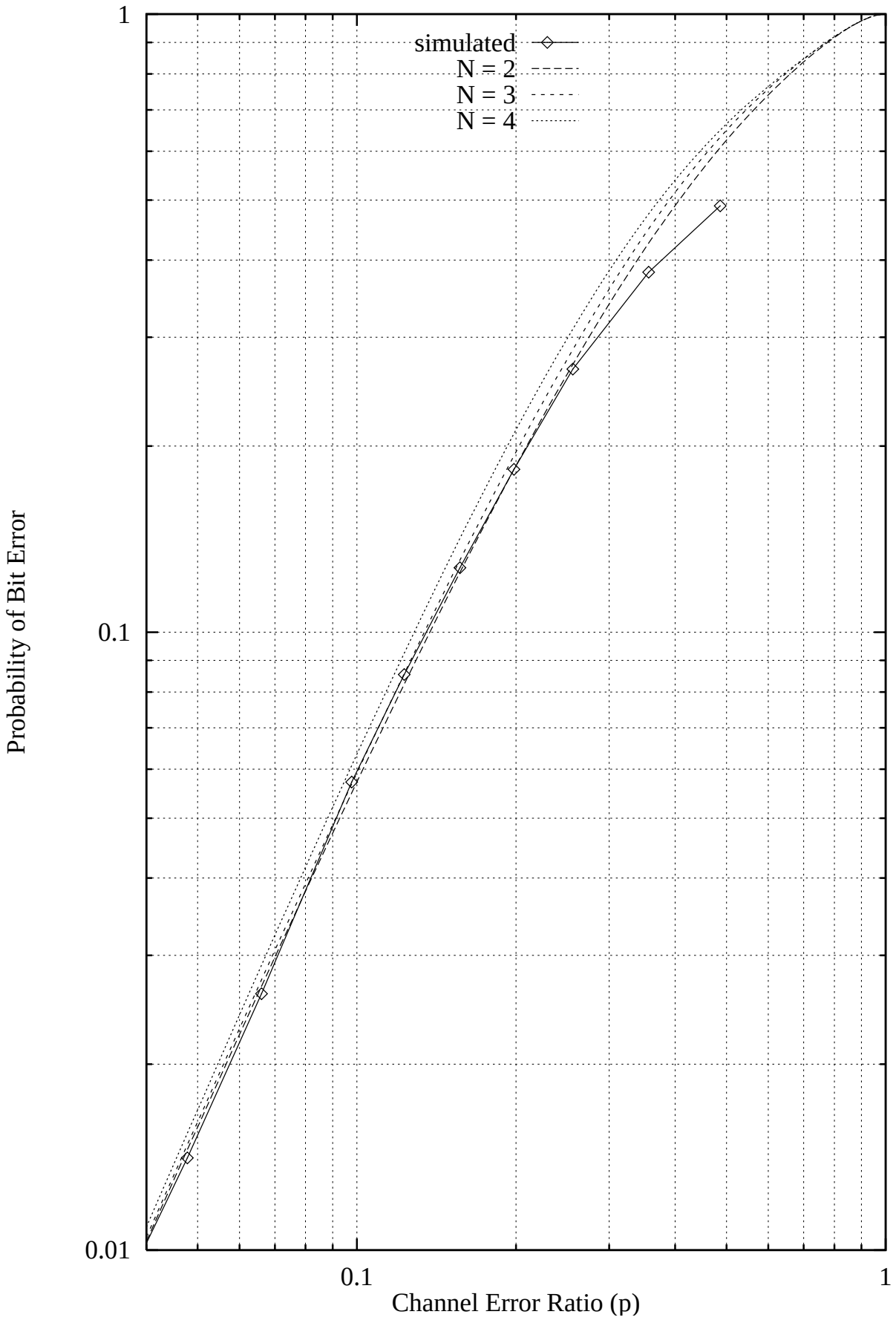


Figure 4: Probability of Bit Error (upper bound approximation) versus p for two state rate 1/2 convolutional code on BSC.

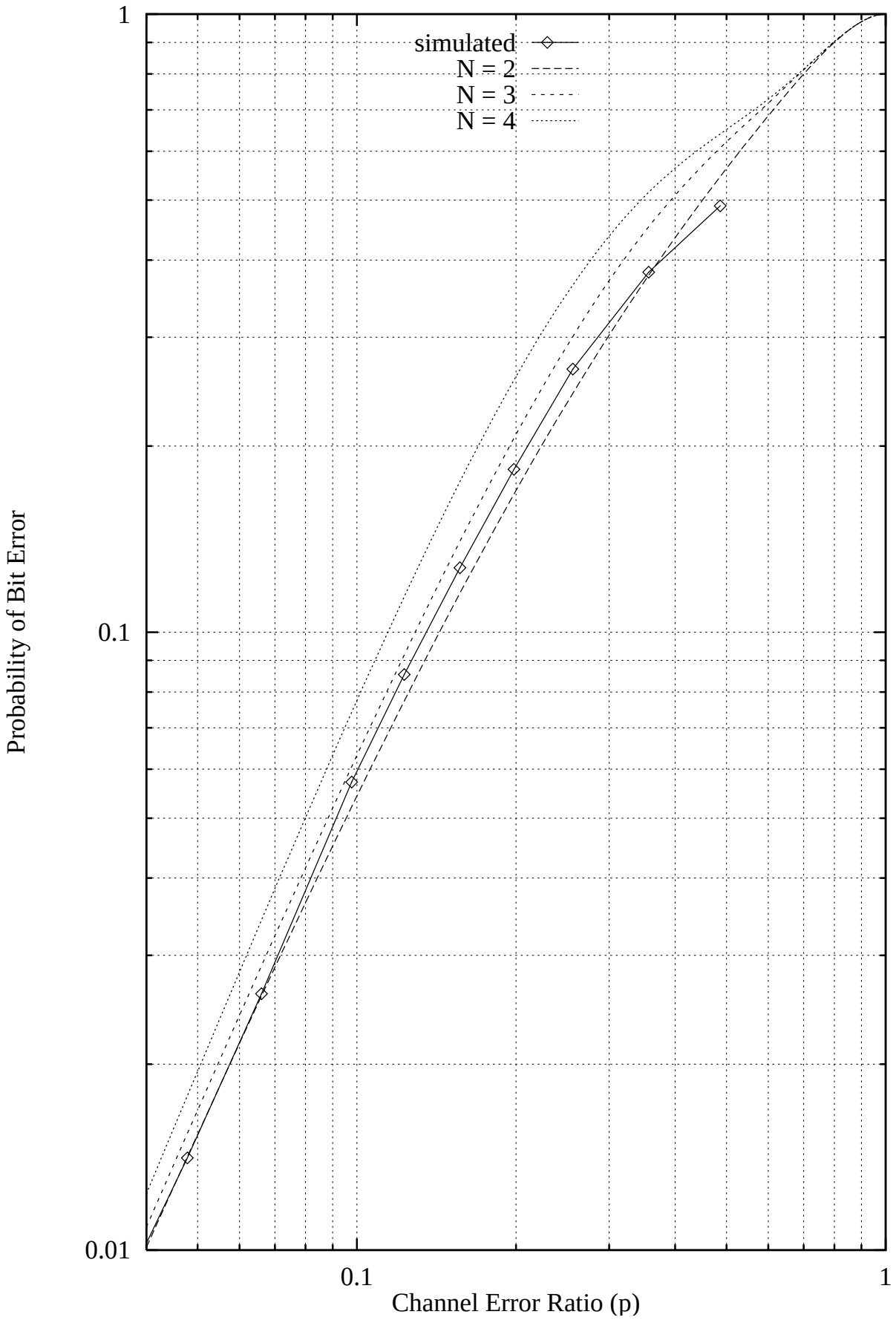


Figure 5: Probability of Bit Error (union bound approximation) versus p for two state rate 1/2 convolutional code on BSC.

Figure Captions:

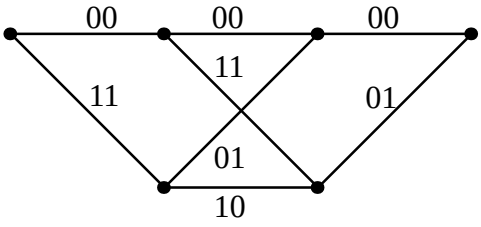
Figure 1: Trellis for two state rate 1/2 convolutional code and N = 2.

Figure 2: Probability of Event Error (lower bound approximation) versus p for two state rate 1/2 convolutional code on BSC.

Figure 3: Probability of Event Error (union bound approximation) versus p for two state rate 1/2 convolutional code on BSC.

Figure 4: Probability of Bit Error (upper bound approximation) versus p for two state rate 1/2 convolutional code on BSC.

Figure 5: Probability of Bit Error (union bound approximation) versus p for two state rate 1/2 convolutional code on BSC.


00
00
00
11
11
01
01
10