Upper bounds on the number of errors corrected by a convolutional code

Justesen, Jørn

Published in:
I E E E Transactions on Information Theory

Link to article, DOI:
10.1109/TIT.2003.822600

Publication date:
2004

Document Version
Publisher's PDF, also known as Version of record

Upper Bounds on the Number of Errors Corrected by a Convolutional Code

Jørn Justesen, *Member, IEEE*

***Abstract*—We derive upper bounds on the weights of error patterns that can be corrected by a convolutional code with given parameters, or equivalently we give bounds on the code rate for a given set of error patterns. The bounds parallel the Hamming bound for block codes by relating the number of error patterns to the number of distinct syndromes.**

***Index Terms*—Convolutional codes, Hamming bound.**

I. Introduction

The aim of this correspondence is to study the error patterns that can be corrected by a binary convolutional code with a given rate and memory. In particular, we derive bounds similar to the Hamming bound for block codes. As in the block code bound, the argument is based on upper bounding the number of correctable error patterns by the number of distinct syndromes. In some cases it is more convenient to fix the set of error patterns and the memory and to derive an upper bound on the rate.

The only previous Hamming-type upper bound that we are aware of is the result in [1].

For block codes, there is a direct relationship between distances and correctable error patterns. Thus, the Hamming bound is often stated as an upper bound on the minimum distance of a block code. Since we cannot apply the same geometric argument to convolutional codes, we prefer the following statement of the block code bound.

Hamming bound for (N, K) block codes: If a linear code can correct all error patterns of weight $\leq T$, then

$$\sum_{j \leq T} \binom{N}{j} \leq 2^{N-K} \tag{1.1}$$

where the left side of the inequality enumerates the error patterns and the right side is the number of distinct syndromes.

It is well known that in most cases it is not possible to correct all error patterns of the weight indicated by the Hamming bound. However, the bound gives a useful estimate of the performance of a code when the decoding is maximum likelihood and the rate is close to the capacity of the binary symmetric channel.

Syndromes are essential to our arguments. In the discussion of trellis decoding, we use the version that is based on syndrome former states. The structure of encoders and syndrome formers is not the topic of this analysis, and we make some simplifying assumptions at the expense of generality.

In Section II, we derive a bound on correctable error patterns by relating them to nonzero syndromes of a specific length. In Section III, this approach is extended to longer error sequences of increasing weight. In Section IV, we consider segmentation of error sequences using concepts from trellis decoding. We then derive an upper bound based on a variable-length description in Section V.

Manuscript received November 30, 2000; revised September 30, 2003.

The author is with COM, Technical University of Denmark, DK-2800 Lyngby, Denmark (e-mail: jju@com.dtu.dk).

Communicated by R. Koetter, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2003.822600

II. Finite Error Patterns

In this section, we derive an upper bound on the number of errors T that can be corrected for any location of the errors. This result is also an upper bound on the free distance since

$$T = \left\lfloor \frac{d_{\text{free}} - 1}{2} \right\rfloor. \tag{2.1}$$

It was proved in [2] that an (n, k) convolutional code with memory M (measured in bits) has a dual $(n, n-k)$ code with the same memory. M has the interpretation that the number of states in minimal encoders and syndrome formers is 2^M. Whenever encoders or syndrome formers are referred to, we always assume that they are minimal and basic. However, we assume the additional simplifying property.

Definition 2.1: A pair of encoders for a convolutional code and its dual are said to be *regular* if their memories measured in blocks m and m' satisfy

$$M = mk = m'(n-k). \tag{2.2}$$

A code is called regular if it has a regular encoder and syndrome former.

The purpose of this definition is only to focus the discussion on the typical situation. Clearly, there are values of M such that (2.2) cannot be satisfied, and some codes have slightly different structures, but the results could be modified to include such cases at the expense of a more complex notation.

Definition 2.2: If a sequence is zero before block i, has nonzero blocks i and $i+j-1$, and is zero from block $i+j$ on, we say that then *length* of the sequence is j.

If the error pattern has length s blocks, the number of nonzero syndrome blocks is at most $s + m'$.

Lemma 2.1: For a regular (n, k) code with memory M, the error patterns of length s are checked by at most $(n-k)(s+m')$ syndrome bits.

Our first bound will be based on syndrome sequences that are zero with the exception of a finite segment, and we get an upper bound on T by comparing the number of error patterns to the number of distinct syndromes.

Lemma 2.2: A regular code of rate R and memory M can correct at most T errors, where for any $s > 0$

$$\sum_{j \leq T} \binom{ns}{j} \leq 2^{n(1-R)s+M}. \tag{2.3}$$

Proof: We count the number of error patterns of length s blocks. From Lemma 2.1 we get an upper bound on the number of relevant syndrome bits, and this gives an upper bound on the number of distinct syndromes.

This is exactly the translation of the Hamming bound using the "inverse concatenation construction" [3]. We find an upper bound on T by applying (1.1) to terminated codes with parameters $N = sn$, $K = ks - M$, and taking the minimum for $s > 0$. We can get a different interpretation of the bound from the right-hand side of the equation by noting that 2^M is the number of states of the syndrome former.

Lemma 2.3: Two error patterns of length sn can be distinguished only if the syndromes are different or the states of the syndrome former are different at the end of the error patterns.

The block code bound (1.1) is tight for short codes and codes of high rate. However, for finite lengths, the bound of Lemma 2.2 is not tight. The reason is that more error patterns have to be corrected by the same syndromes. If the first jn bits of an error pattern coincide with the first blocks of a codeword, the first jk syndrome bits equal zero. A similar argument applies to the end of the error pattern. Thus, more low-weight error patterns than those counted by the left-hand side of (2.3) have to be distinguished by the syndromes on the right-hand side.

In order to state the modified upper bound, we need a lower bound on the number of truncated codewords of low weight. Only the first term of the correction is considered here. If the best (n, k) convolutional code has no truncated codeword of weight less than t, and a_t is a lower bound on the number of weight t truncated words on both sides of the sn bits under consideration, we get the following.

Theorem 1: A regular code of rate R and memory M can correct at most T errors, where

for any $s > 0$

$$\sum_{j \le T} \binom{sn}{j} + \sum_{j \le T-t} a_t \binom{sn}{j} \le 2^{sn(1-R)+M}. \tag{2.4}$$

From capacity considerations it is clear that the asymptotic bound is not changed.

Proof: The additional error patterns have nonzero syndrome bits only in the window under consideration. A different distribution of truncated word weights gives a larger number of low-weight error patterns.

Example 1 : Consider a $(3, 2)$ code with $M = 6$. If we consider $s = 4$, there are 10 syndrome bits, and Lemma 2.2 is satisfied with $T = 3$. In order to apply Theorem 1, we notice that the first block can have three vectors of weight 2. Furthermore, there is at least one segment of length 2 blocks which has a zero second block. The same argument applies to the weight of the last blocks in a codeword. Thus, we evaluate Theorem 1 with $a_2 = 8$. The 1024 syndrome values must identify not only 299 error patterns of weight 0–3 among the 12 bits, but also eight double errors in adjacent blocks and 96 triple errors where two errors are outside and the last error inside the window, a total of 403.

The example shows that the syndromes are used more efficiently than indicated by (2.3). There are no simple cases where the upper bound on T is changed by the correction, but the bounds on longer error patterns can change, as we demonstrate when we return to the example in Section III.

III. More Errors Can Be Corrected in Longer Segments

The bound (2.4) gives a minimal value of T for a finite range of s, but eventually it increases with increasing s. Similarly, a typical convolutional code contains only a small number of short codewords of low weight. Thus, more errors can be corrected, provided that at most T of the errors occur in any s' consecutive blocks. A lower bound of this form may be expressed in terms of the extended (or active) distances [4], [5] of the code. However, since these distances are obtained as minimum weights of codewords that are constrained not to pass the zero state, they are not minimum distances of linear block codes. Thus, we cannot obtain upper bounds on specific codes by translating block code bounds. However, it may be noted that asymptotically the number of correctable errors is upper-bounded by the same expression as the Gilbert-type lower bound on extended distances. Thus, TR/M is upper-bounded by the Costello bound on the free distance [4], and for long error patterns the relative number of correctable errors is upper-bounded by $H^{-1}(1 - R)$.

To get an upper bound we shall again enumerate the relevant error patterns. Either the total number of errors is at most T, or there is at least one error in each of the blocks at the ends. Thus, we have the following.

Theorem 2 : If a regular (n, k) convolutional code corrects any combination of T errors, and any combination of $T + 1$ errors such that at most T are in s' consecutive blocks, then

$$\sum_{i,j=1}^{T} \binom{n}{i} \binom{n}{j} \binom{(s'-1)n}{T-i-j+1} + \sum_{j \le T} \binom{(s'+1)n}{j} \le 2^{(s'+1)n(1-R)+M}. \tag{3.1}$$

Proof: All error patterns of at most weight T are counted. In addition, error patterns of length $s' + 1$ and weight $T + 1$ are included if they have at least one error in the first and last block.

Example 2 : We consider a $(2, 1)$ $M = 2$ code. The terminated block codes have parameters $(2s, s - 2)$. Applying (2.4) with $M = 2$ and $a_2 = 2$ we get

$$\sum_{j=0}^{2} \binom{2s}{j} + 2 = 3 + s + 2s^2 < 2^{s+2}$$

indicating that two errors can be corrected. The upper bound is $T \le 2$, since $T = 3$ would violate (2.3). Applying (3.1) with $T = 2$ and $s' = 3$, we get 37 error patterns of weight at most 2 from the last term and 20 error patterns of weight 3 from the first term. However, we can get a slightly tighter bound by the same reasoning as in Theorem 1: the same syndrome should correct two patterns of weight 2 and eight patterns that have a double error outside the four blocks and a single error at least two blocks away. Thus, the total number of error patterns becomes 67 exceeding the number of syndromes. Repeating this calculation for $s' = 4$ blocks we get $56 + 2$ error patterns of weight at most 2, and $28 + 8$ error patterns of weight 3. Thus, the 128 syndromes should identify 84 error patterns within the five blocks and 10 additional patterns. Thus, the upper bound allows three errors in five blocks, while the code with generators (111) and (101) actually has a word of length 5 and weight 6, and we cannot be sure to correct three errors until the length reaches six blocks.

Example 1 (Continued): For the $(3, 2)$ code with $M = 6$ we found $T = 3$. Here we consider correcting four errors provided that short segments contain at most three errors. Taking $s' = 3$ we may apply Theorem 2 to get 258 error patterns of weight 4 in addition to the 403 error patterns of lower weight. From this calculation, the 10 syndrome bits are sufficient. However, if we take into account the errors patterns that have double errors outside the 12-bit window, there are 448 additional error patterns of weight 4, and 10 syndrome bits are not enough. Theorem 2 is satisfied for $s' = 4$.

As indicated by Lemma 2.3, the number of correctable error patterns increases by a factor 2^{n-k} for each additional block. Consequently, the weight of the error patterns can also increase with the length, but since this approach does not give any simple expressions, we omit the details.

IV. Trellis Decoding

We use the following version of trellis decoding of the convolutional code. As the first step, we assume that the $n - k$ syndrome bits are computed for each received block. We take the syndromes as the input to the trellis decoder, and let the trellis consist of states and transitions in the syndrome former. In this way, we can study the error pattern directly in a way that is independent of the data. It may be noted that for our purpose it is important that all decisions are data independent, even then we choose among several equally likely possibilities, whereas many Viterbi decoders choose a particular data bit in such situations.

For a block code, a list of the error patterns indexed by the syndromes gives a complete description of the error correction process. A partial list of syndromes and error patterns for a convolutional code could be constructed, but the complete list is in general not finite. It has been noted by several authors [6], [7] that, at least in principle, the performance of a convolutional code on a discrete memoryless channel where the log-likelihood ratios are approximated by rational branch weights can be modeled as a finite-state process. We refer to such a model as a big trellis model (BTM).

Definition 4.1 : The *BTM* is a finite state model of the decoding process for a given convolutional code. The states of the BTM are identified by a vector of reduced metric values and an indication of the syndrome former state that is chosen in the back search. The output is the corrected error pattern.

Since we only discuss the binary symmetric channel, the metric values indicate the Hamming weight of the error patterns, and we assume that in each step the state metrics are reduced to make the smallest one equal zero. Clearly, the number of states is finite, although it may be exponential in the number of encoder states. Note that even though the selected state refers to the back-search phase of the decoding, this model proceeds in the forward direction only. Each state transition in the BTM reflects a step in the Viterbi algorithm, but it has only one branch connecting the two preferred states. On the other hand, there may be several possible transitions for a given state and a given input syndrome, each labeled with the minimum-weight error on the corresponding branch. Whenever there is an ambiguous decision in the Viterbi algorithm, it is important to make a specific choice of error pattern.

Even for codes of moderate size, the number of states in a BTM makes it infeasible for actual computations. However, the model provides some insight into the general properties of correctable error patterns. We use the term *ground state* to indicate the state of the BTM for a sufficiently long error-free input.

Definition 4.2 : A correctable error pattern is the labeling of a path that leaves the ground state of the BTM on the first branch and ends in the ground state. A basic error pattern ends the first time it returns to the ground state.

Lemma 4.1 : Any concatenation of basic error patterns results in a correctable error pattern.

Proof: The decoding process is described by the BTM with the segments joined in the ground state.

This variable-length description of the error patterns may be seen as a parallel to the variable-length description of codewords [8]. The connection to the syndrome sequences is made by the following lemma.

Lemma 4.2 : There is a one-to-one correspondence between correctable error patterns and syndrome segments, and a syndrome segment has the same length as the error pattern measured in number of branches.

Proof: For a given initial state of the syndrome former, a particular error sequence gives a unique syndrome sequence. Since the syndrome former is returned to the zero state at the end of each segment, that is always the initial state. Even though from a given state the syndrome does not uniquely determine the error, the error sequence is unique once the syndrome former is returned to a particular state.

Lemma 4.3 : The basic error patterns satisfy the prefix condition.

Proof: Since the error sequence is given, the correct syndrome former state is known, and the error sequence is recognized as a basic error pattern when the BTM reaches the ground state. Thus, it is not a prefix of another basic error pattern.

We note that in Viterbi decoding, a block is decoded when all surviving branches share the same first branch. When we reach the end of a correctable error pattern, the metric vector indicates that the decoder interprets the current segment as essentially an error-free codeword. However, as discussed in Section II, a short error pattern may be identical to the start of a codeword, and depending on the minimum distance and constraint length of the code, there is a finite delay in detecting the beginning of a nonzero error pattern.

We can now state an important upper bound.

Theorem 3: Assume that each branch of the convolutional code has $n-k$ redundant symbols. Let the number of basic error patterns of length L, measured in branches, be $A(L)$. Then the number of redundant symbols on each branch must satisfy

$$\sum_{L} A(L)2^{-L(n-k)} \leq 1. \tag{4.1}$$

Proof: It follows from Lemma 4.2 that the error patterns correspond to syndromes of the same length (measured in blocks), and the inequality then follows from the Kraft–MacMillan theorem.

Theorems 1 and 2 provide information about short error patterns. However, for these sequences to become basic error patterns in the sense of Theorem 3, they have to be followed by a suitable number of zero blocks. In Section V, we shall discuss the set of lengths.

Theorem 3 implies a tree description of the basic error patterns. If this description can be converted to a finite-state description of the correctable error patterns, we shall prove in Section V that the result is equivalent to the upper bound of [1]. However, the present version of the upper bound has the advantage that an estimate can be obtained by restricting the sum (4.1) to a finite number of terms and a desired integer value can be used for $n-k$.

V. Finite-State Descriptions of Error Patterns

In this section, we introduce a simplified description of the error patterns. The intention is to describe a set which has some of the simplicity of a set of bounded weight in the block code case. However, at the same time the set should be a good approximation to the set of correctable error patterns for a convolutional code.

Let the upper limit on the number of correctable errors be approximated by a function of the following type.

Definition 5.1 : A set of bounded weight is a set of sequences of blocks e_i with weight $t_i = W(e_i)$ which satisfy

$$t_{kj} = \sum_{i=k}^{k+j-1} t_i \leq T_j \tag{5.1}$$

for

$$T_j = \max\{T, \lfloor \alpha j + \beta \rfloor\} \tag{5.2}$$

where T is an integer and α and β are positive rational numbers.

Given a weight sequence t_k, we may check that (5.1) is satisfied in the following way. Let j' be the largest value of j such that $\alpha j < T$. Verify that any window of length j' contains at most T errors. Then, starting from the beginning of the sequence, calculate the sum t_{0j} as long as it satisfies

$$\alpha j < t_{0j} \leq \alpha j + \beta. \tag{5.3}$$

If the last inequality is not satisfied, the sequence clearly does not satisfy (5.1). When we reach a block $j = k_1$ where the first inequality

is not satisfied, the summation is reset, and we start calculating a new sum t_{k1j} from the next block.

It was proved in [1] that a set of bounded weight can be generated (or recognized) by a finite-state machine. The essential state variable in the description is

$$\sigma = -v\alpha + \sum_{j=u}^{u+v-1} t_j \tag{5.4}$$

which is updated with the term $t_j - \alpha$ in each transition. In addition, the state may have to include a finite number of past inputs. Thus, σ tracks the deviation of the total weight from the linear function αj.

We can now derive a final version of the upper bound by counting the number of error patterns of each length that satisfy (5.1) and (5.2). For that purpose, we need the adjacency matrix of the finite state source, i.e., a matrix P where the elements p_{ij} indicate the number of transitions between two states. From the adjacency matrix of the source we can find the total number of correctable error patterns of a certain length, or we may split the zero state into a starting state and a final state and determine the number of basic error patterns of each length.

If the number of redundant symbols in each block is $n-k$, it was proved in [1] that the largest eigenvalue of P, λ satisfies

$$\log \lambda \leq n - k.$$

This result is also a consequence of Theorem 3, which may be seen in the following way: If $A(x)$ is the length generating function for the set of basic error patterns, we may express the total number of error patterns terminating in the ground state by

$$u(P - xE)^{-1}u' = \frac{A(x)}{1 - A(x)}$$

where $u = (1, 0, 0, \ldots, 0)$. Thus, the smallest pole of this function is $x = 2^{-n+k}$, which implies $A = 1$ and $\lambda = 1/x$.

We now have the following.

Theorem 4 : If a regular convolutional code of rate R with memory M corrects a set of bounded weight with parameters $(T, \alpha, \beta), T$ must satisfy Theorem 1, and the largest eigenvalue of P, the adjacency matrix for the set of correctable error patterns with block length n, satisfies

$$\lambda \leq 2^{n(1-R)}.$$

Example 2 (Continued): The extended row distances of the $(111, 101)$ code are $d_j^r = \lfloor 4 + j/2 \rfloor$ for $j > 2$ branches. We shall derive an upper bound on the rate of a code where the weights of the correctable error patterns are given by $T_j = \lfloor 7/4 + j/4 \rfloor$. These values agree with $T = 2$ and $T_5 = 3$ which were found earlier. The error patterns may be counted in a systematic way by introducing a finite-state source as outlined above. Thus, there are eight states with $\sigma = 0, 1/4, \ldots, 7/4$. The first state corresponds to the ground state of the BTM. The adjacency matrix of the finite-state source is

$$P = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The number of error patterns starting in the ground state and ending in arbitrary states after j blocks may be calculated as

$$[1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]\, P^j\, [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]'.$$

Without including error patterns extending outside the j blocks we get 84 for $s = 5$ in agreement with the first part of the example. We can include a double error in the previous block by changing the left vector to (10000001). Alternatively, we may split the zero state of the source and obtain the generating function for the number of basic error patterns

$$A(x) = \frac{2x^4 - 3x^8}{1 - 8x^4} = 2x^4 + 13x^8 + 104x^{12} + \cdots.$$

Here the two patterns of length 4 have weight 1 and the 13 patterns of length 8 have weight 2. Using this generating function and adding the length 1 zero branch we may apply Theorem 3 and verify that the rate is upper-bounded by $R < 0.556$. In [1], the upper bound was expressed in terms of the maximal eigenvalue of P, $\lambda = 1.853$, where $\log \lambda = 0.890$ is a lower bound on the redundancy per block. The rate–$1/2$, memory-2 code generated by $(111, 110)$ has row distances that increase by 2 in every three branches, but the free distance is only 4. It may be verified that the upper bound for codes correcting $5/3 + j/3$ errors in j branches is $R < 0.467$. Thus, a code of rate $1/2$ cannot correct all patterns of two errors and one error in every three blocks.

Example 1 (Continued): In Sections II–IV we found $T = 3$ and $T_5 = 4$. These values are consistent with $T_j = \lfloor 11/4 + j/4 \rfloor$ and $T_j = \lfloor 3 + j/5 \rfloor$. However, only the last set satisfies Theorem 4 for $R = 2/3$. The upper bound on the rate for this set is $R < 0.676$. Thus, for $R = 2/3$, $M = 6$, and $T = 3$, the upper bound on α is only slightly greater than $1/5$.

VI. Conclusion

The block code Hamming bound gives a direct performance estimate and is related to syndrome decoding in an obvious way. We have provided a similar relation between syndromes and correctable error patterns for convolutional codes. The upper bounds serve to estimate the number of errors corrected by a convolutional code when maximum-likelihood (ML) decoding is used.

References

[1] J. Justesen, "Bounded distance decoding of unit memory codes," *IEEE Trans. Inform. Theory*, vol. IT-39, pp. 1616–1627, Sept. 1993.

[2] G. D. Forney Jr., "Structural analysis of convolutional codes via dual codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 512–5518, Sept. 1973.

[3] ——, "Convolutional codes II: Maximum likelihood decoding," *Inform. Contr.*, vol. 25, pp. 222–266, July 1974.

[4] R. Johannesson and K. Zigangirov, *Fundamentals of Convolutional Coding*. Piscataway, NJ: IEEE Press, 1999.

[5] C. Thommesen and J. Justesen, "Bounds on distances and error exponents of unit memory codes," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 637–649, Sept. 1983.

[6] J. P. M. Schalkwijk and A. J. Vinck, "Syndrome decoding of binary rate 1/2 convolutional codes," *IEEE Trans. Commun.*, vol. COM-24, pp. 977–985, Sept. 1976.

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

[8] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751–772, Oct. 1971.