How to design Trellis Codes with Large Distance Properties

RANJAN BOSE
Department of Electrical Engineering
Indian Institute of Technology, Hauz Khas, New Delhi 110016 INDIA
http://paniit.iitd.ac.in/~rbose

Abstract: - We present a novel method for constructing trellis codes, based on recursive nonlinear equations. Using this method, trellis codes having good code rates ($R = k/n$) can be constructed. At the same time, the method allows the code-designer to construct a trellis with a large free distance, d_{free}, at the cost of a larger number of states in the trellis. It has been shown in this paper that for the code rate, R, less than a critical rate, R_c, the free distance can be made as large as desired for large n. This design methodology provides the freedom to play with the code performance and the code rate of the trellis code. Using this method, a rate 3/4 trellis code has been constructed with $d_{free} = 12$.

Key-Words:- Trellis Codes, Free Distance, Error Control Codes

1 Introduction

Convolutional codes and other trellis codes can be conveniently described using a trellis [1, 2]. A trellis is a graph whose nodes are in a rectangular grid, semi-infinite to the right. The number of nodes in each column is fixed, and corresponds to the states in the trellis diagram. The code rate of a trellis code reflects the fraction of the codeword that consists of the information symbols, and is defined as

$$R = \frac{k}{n}, \tag{1}$$

where k is the information length (prior to coding) and n is the codeword length (after encoding). The free distance, d_{free}, of a tree code is the smallest Hamming distance between any two distinct code sequences [2, 3].

Trellis codes are typically constructed using a shift register and a logic circuit, as shown in Fig. 1. The constraint length, ν, is an important descriptor of a trellis code. For a binary trellis code, the number of states in the trellis is given by 2^{ν}. Because of practical consideration, practical trellis codes use small integer values of n and k (frequently, $k = 1$). Therefore, it is difficult to design a practical trellis code with code rate close to unity, as is possible for block codes (such as a Reed-Solomon code [4]).

Several authors have explored the connection between quasi-cyclic codes and convolutional codes [5-8]. The construction of MDS convolutional codes is discussed in [9] where again the connection between quasi-cyclic codes and convolutional codes is exploited. A code construction procedure for geometrically uniform trellis codes has been proposed in [10]. Analog error correcting codes based on chaotic dynamical systems have been studied in [11] where the authors mention that a convolutional encoder can be emulated using a one-dimensional dynamic system. Most of the trellis codes in use today have limited number of states (typically less than 128). Almost all of these codes have been discovered using exhaustive searches. However, the search for good codes becomes computationally prohibitive as we increase the number of states [12].

In this paper we propose a new method for generating trellis codes using a recursive nonlinear equation. We call them *Kola codes*. This trellis construction method allows us to specify the number of states in the trellis as well as the number of branches emanating from each node. It also gives us a handle on the code

performance (in terms of the free distance, d_{free}) and the code rate of the trellis code. For this design methodology, the code rate is independent of the trellis diagram. Our method can also be used as a constructive technique for trellis codes with large number of states. The paper is organized as follows. Section 1 is the introduction. In section 2 we give the trellis code construction methodology. The free distance of the constructed trellis code is analyzed in section 3. The paper concludes in section 4.

2 Code construction methodology

As shown in Fig. 1, trellis codes are typically generated using a memory unit, which basically is a shift register, and a logic circuit [2, 3]. The computation of the codeword frame is done using the logic circuit and the codeword frame is then shifted out. The code rates are typically 1/2, 1/3, 2/3 etc. because of practical constraints [1].

We propose a recursive nonlinear equation to generate the trellis-coded output for an input bit-stream. Consider, for example, the following recursive nonlinear equation

$$x_m = F(x_{m-1}), \tag{2}$$

where

$$F(x_{m-1}) = a.\, x_{m-1}(1 - x_{m-1}), \tag{3}$$

and $0 \le x_m \le 1$. If the starting value in (3) is x_0, the value after m-fold iteration of the recursive nonlinear equation is

$$x_m = F^{(m)}(x_0). \tag{4}$$

The method for constructing the trellis code is as follows. The number of states in the trellis is a design parameter, and is given by 2^s. Thus, large trellises (if required) can be easily constructed by increasing the value of s. Each state takes a value between 0 and 1, and is given by

$$\left(\frac{i-1}{2^s}\right) \le S_i \le \left(\frac{i}{2^s}\right), \tag{5}$$

where i is an integer such that $0 \le i \le 2^s$. Thus, the states can be visualized as non-overlapping segments of equal length on the line between 0 and 1. Any number in the interval [0, 1] will belong to one and only one state, and upon s-fold iteration of the recursive equation given in (2) will make a transition to some other state. The iterations given by (3) can be used to carry out state transitions. The encoder can, therefore, be represented using a trellis diagram with 2^s states.

In order to carry out encoding, the input sequence is first segmented into frames of length k. Let the decimal value of the input frame (of size k bits) be D, where D is an integer such that $0 \le D \le 2^k - 1$. We start with an initial state, S_0. The encoder maps the initial state to the next state obtained after D-fold iteration as given in (4), i.e.,

$$S_D = F^{(D)}(S_0). \tag{6}$$

Every state transition (the branch connecting two nodes in the trellis) has an output vector (n-tuple) associated with the transition. Since there are exactly 2^k branches emanating from a single node, the output vectors can be picked from any appropriate (n, k, d) block code, where d is the minimum distance of the code [13]. The encoding procedure is very simple. Given any state, for a certain input information frame, a state transition occurs as per (6). A state transition corresponds to moving along a branch in the trellis diagram. The output codeword frame is the n-tuple associated with the branch. Thus, the recursive nonlinear equation can be used to generate a trellis diagram having 2^s states, with every node having 2^k branches emanating from it. We will call this class of codes: Kola (n, k, s) codes. If the total number of states (2^s) is greater than the total number of branches emanating from each node (2^k), it is possible to design a trellis that does not have parallel branches.

There can be several choices of the recursive nonlinear equation given in (3). The condition that the recursive equation must satisfy is that it should visit all the 2^s states with equal frequency. Thus, a finite state machine with 2^k states can also be used.

Next, let

$$s = k + r, \tag{7}$$

where r is an integer such that $r << k < s$. The number of ways to choose from the 2^s states in order to assign the 2^k branches is given by

$$N = \binom{2^s}{2^k} = \binom{2^{k+r}}{2^k} = \frac{2^{k+r}!}{(2^k!)(2^{k+r} - 2^k)!} \tag{8}$$

Using the Stirling's approximation, $p! \approx \sqrt{2\pi p}\, p^p e^{-p}$ for large p, and some algebraic manipulations, (8) can be simplified to

$$N > \frac{1}{2\pi}\left(\frac{\left(2^r\right)^{\left(2^r\right)}}{\left(2^r - 1\right)^{\left(2^r - 1\right)}} \right)^{2^k}. \tag{9}$$

As we increase the value of k, the number of choices available for assigning the branches emanating from a node grows approximately as $\sim \alpha^{2^k}$, where $\alpha > 1$. This gives us more flexibility for designing good Kola codes for large values of k. This is important because in order to obtain good codes with code rates close to unity, we will require block codes with large values of n and k [12].

3 Free distance of the trellis codes

We will now estimate the free distance, d_{free}, of the Kola (n, k, s) codes obtained using the proposed constructive technique. d_{free} represents the minimum distance between arbitrarily long (possibly infinite) encoded sequences and determines the error correcting capability of the trellis code [3]. Let the length of an error event, L, be the number of time instances after which a path in the trellis diverges from and then merges back to a reference path. Since each outgoing branch from a node is labeled with one of the codewords of an (n, k, d) block code, in case there are parallel transitions, the free distance of the trellis code is given by

$$d_{free} \geq d, \; L = 1. \tag{10}$$

Next, consider the case where no parallel transitions occur. We make an assumption that the states generated by the recursive nonlinear equation given by (2) are equally probable. In this case, the probability that a path in the trellis diverges from and merges back to the reference path after L time instances is given by (i.e., the probability of an error event of length L)

$$P_L = \left(1 - \frac{1}{2^k}\right)\left(1 - \frac{2^k}{2^s}\right)^{(L-2)}\left(\frac{2^k}{2^s}\right), L \geq 2. \tag{11}$$

The first term on the right hand side represents the probability of a path diverging from the reference path, the second term corresponds to the probability of not merging back to the reference path for (L – 2) time instances, and the last term is the probability of the path merging back to the reference path. The d_{free} of the trellis code corresponding to the length of error event, L, can be lower-bounded as

$$d_{free} \geq Ld, \; L \geq 2. \tag{12}$$

The expected value of d_{free} for the case where no parallel transitions occur is lower bounded by

$$\overline{d_{free}} \geq \sum_{L=2}^{\infty}\left(1 - \frac{1}{2^k}\right)\left(1 - \frac{2^k}{2^s}\right)^{(L-2)}\left(\frac{2^k}{2^s}\right)(Ld). \tag{13}$$

Substituting $x = 2^{-k}$ and $y = 2^{-(s-k)}$, (13) can be written as

$$\overline{d_{free}} \geq ((1-x)yd)\left(\sum_{L=2}^{\infty} L(1-y)^{L-2}\right)$$

$$= ((1-x)yd)\left(\sum_{L=1}^{\infty} L(1-y)^{L-1} + \sum_{L=1}^{\infty}(1-y)^{L-1}\right)$$

$$= ((1-x)yd)\left(y^{-2} + y^{-1}\right). \tag{14}$$

Under the assumption that $x << 1$ (i.e., for a large value of k), (14) can be simplified to

$$\overline{d_{free}} \geq yd\left(y^{-2}+y^{-1}\right) > y^{-1}d = 2^{(s-k)}d\,. \qquad (15)$$

Next, let us define the critical rate, R_c, as follows:

$$R_c = \frac{s}{n}\,. \qquad (16)$$

The critical rate can be varied from 0 to a very large number, since both s and n are design parameters. Using (1), (15) and (16) we can obtain the following lower-bound:

$$\overline{d_{free}} > 2^{(R_c-R)n}d\,. \qquad (17)$$

An important conclusion that can be drawn from (17) is that for $R < R_c$, $\overline{d_{free}} \to \infty$ as $n \to \infty$. Also, the $\overline{d_{free}}$ of the constructed trellis codes increase exponentially with increasing n. Since, the number of errors, t, that can be corrected using a trellis code increases linearly with d_{free} [14], the average residual error probability $\overline{P_e} \to 0$ as $n \to \infty$. From (16) we observe that if $s \geq n$, $R_c \geq 1 > R$. Therefore the condition $s \geq n$ will guarantee trellis codes with large $\overline{d_{free}}$ for sufficiently large n.

Suppose the (n, k, d) code, whose codewords are mapped onto the outgoing branches from the node of a trellis, is a linear block code. The Singleton bound for a linear (n, k, d) block code satisfies [13]

$$d \leq n - k + 1. \qquad (18)$$

Using (1) and (18), we eliminate n from (17) to obtain

$$\overline{d_{free}} > 2^{(R_c-R)n}d \geq 2^{(R_c-R)\left(\frac{d-1}{1-R}\right)}d$$

$$= 2^{\left(\frac{R_c-R}{1-R}\right)(d-1)}d = 2^{\eta_R(d-1)}d\,, \qquad (19)$$

where, the rate efficiency, η_R, is defined as

$$\eta_R = \left(\frac{R_c-R}{1-R}\right). \qquad (20)$$

To illustrate the trellis code construction methodology, lets take the following example. Let the (4, 3, 2) cyclic code be the linear block code whose codewords are to be associated with the branches emanating from the nodes in the trellis. The generator matrix of this code is given by [12]

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \qquad (21)$$

Let us choose s = 5 which gives the total number of states in the trellis equal to 2^5 = 32. Here, n = 4, k = 3, d = 2, R = 3/4 and R_c = 1.25. We have restricted ourselves to small values of n, k and s so that the trellis diagram can be shown comfortably. The trellis constructed using these parameters is shown in Fig. 2. The codewords of the (4, 3, 2) cyclic code have been randomly assigned to the branches of the trellis. The free distance of this trellis code is d_{free} = 12. We compare our Kola (4, 3, 5) code with some of the good convolutional codes obtained from computer searches [12]. It can be seen from Ta. 1 that the Kola (4, 3, 5) code is superior to the best possible rate 1/3, 32-state convolutional code in terms of the code rate.

4 Conclusions

We have proposed a new method for constructing trellis codes with large free distances, d_{free}. The proposed method uses a nonlinear recursive equation to generate the trellis diagram. The number of states in the trellis is a design parameter. The free distance of the trellis code can be increased at the cost of a larger number of states. However, the code rate need not be sacrificed in order to increase the free distance. Thus, the code performance can be improved at the cost of decoding complexity.

The distinct advantages of this method are:

1. The method gives the code-designer the flexibility to choose the code rate, R (by choosing n and k). This allows the designer to control the excess bandwidth requirement.
2. The method gives the code-designer the flexibility to choose the number of states in the trellis. This allows the designer to determine the error correcting capability of the trellis code.
3. We can *construct* good trellis codes with large free distances. Our method is useful in the sense that it provides a constructive technique for trellis codes with large number of states.

It has been demonstrated that as we increase the value of k, the number of choices available for assigning the branches emanating from a node grows approximately as $\sim \alpha^{2^k}$, where $\alpha > 1$.

We have defined a new parameter critical rate as $R_c = \frac{s}{n}$, where 2^s is the number of states in the trellis and n is the codeword length. An interesting result is that the expected value of the free distance of the constructed trellis is lower-bounded as: $\overline{d_{free}} > 2^{(R_c - R)n} d$. Thus, for $R < R_c$, $\overline{d_{free}} \to \infty$ as $n \to \infty$. Also, the $\overline{d_{free}}$ of the constructed trellis codes increase exponentially with increasing n. Since, the number of errors, t, that can be corrected using a trellis code increases linearly with d_{free}, the average residual error probability $\overline{P_e} \to 0$ as $n \to \infty$.

References:

[1] R.E. Blahut, *Theory and Practice of Error Control Codes*, Addison-WesleyReading, Massachusetts, 1983.

[2] J.H. van Lint, *Introduction to Coding Theory*, Springer-Verlag, Heidelberg, 1999.

[3] R.H. Morelos-Zaragoza, *The Art of Error Correcting*, John Wiley and Sons, New York, NY, 2002.

[4] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM J.*, vol. 8, pp. 300-304, Jun. 1960.

[5] J. Justesen, "New convolutional code constructions and a class of of asymptotically good time varying-codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 220-225, Mar. 1973.

[6] Y. Levy and D.J. Costello, Jr., "An algebraic approach to constructing convolutional codes from quasicyclic codes," *DIMACS Ser. Discr. Math. Theor. Comput. Sci*., vol. 14, pp. 189-198,1993.

[7] J.L. Massy, D.J. Costello, Jr. and J. Justesen, "Polynomial weights and code constructions," *IEEE Trans. Inform. Theory*, IT-19, pp. 101-110, Jan. 1973.

[8] R.M. Tanner, "Convolutional codes from quasicyclic codes: A link between the theories of block and convolutional codes," Univ. Calif., Tech. Rep., UCSC-CRL-87-21, Nov. 1987.

[9] R. Smarandache, H. Gluesing-Luerssen and J. Rosenthal, "Construction fo MDS-convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 2045 – 2049, Jul. 2001.

[10] Y. Levy and D.J. Costello, Jr., "A geometric construction procedure for geometrically uniform trellis codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 1498 – 1513, Sep 1996.

[11] B. Chen and G.W. Wornell, "Analog Error Correcting Codes based on Chaotic Dynamical Systems," *IEEE Trans. Commun.*, vol. 46, no. 7, pp. 881-890, Jul. 1998.

[12] R. Bose, *Information Theory, Coding and Cryptography*, Tata McGraw-Hill, New Delhi, 2002.

[13] F.J. MacWilliams and N.J. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam, The Neatherlands: North Holland, 1977.

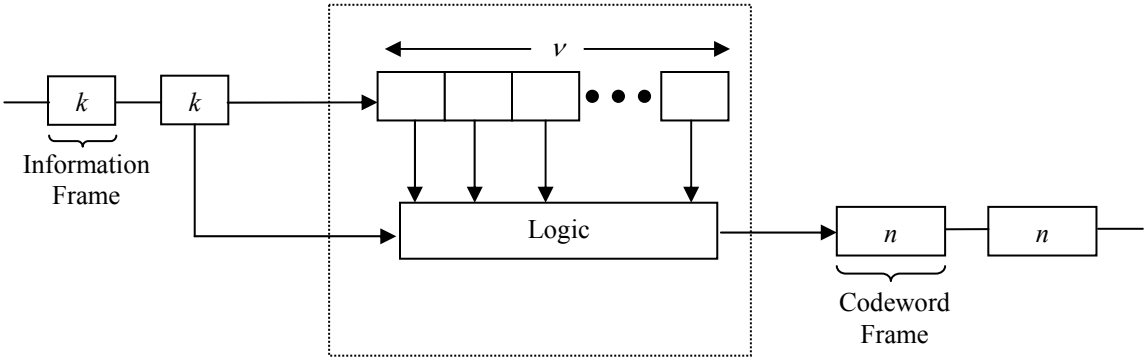

Fig. 1 A shift register encoder that generates a trellis code.

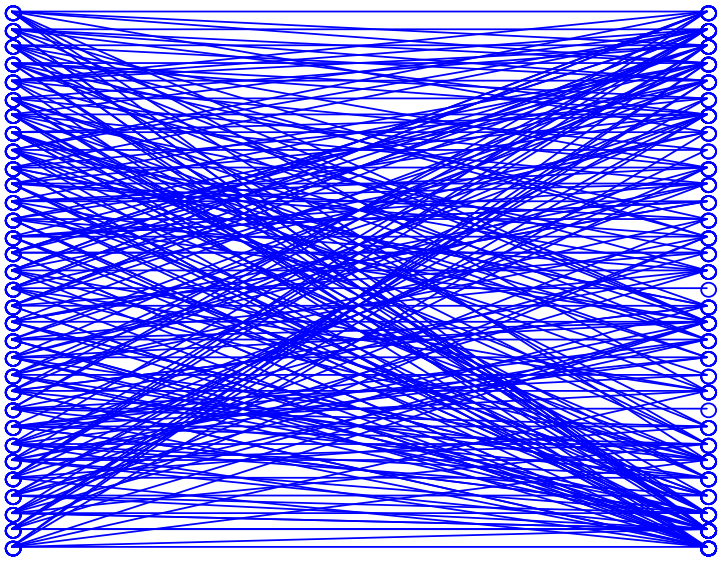

Fig. 2 The trellis diagram for the Kola (4, 3, 5) code, obtained using the parameters $n = 4$, $k = 3$, $d = 2$, and $s = 5$ and the recursive function $x_m = 4.\ x_{m-1}(1 - x_{m-1})$.

Tab. 1 Comparison of Kola (4, 3) code with other convolutional codes.

Code	**No. of states**	***n***	***k***	***R***	d_{free}
Rate 1/2 Non-catastrophic	32	10	5	1/2	7
Rate 1/2 Catastrophic	32	10	5	1/2	8
Rate 1/3	32	15	5	1/3	12
Kola (4, 3, 5) code	32	4	3	3/4	12