

SOME NEW ERROR-CORRECTING (d,k) TRELLIS CODES FOR CLOCK EXTRACTION ON BANDWIDTH LIMITED CHANNELS

A.S.J. HELBERG, H.C. FERREIRA and P.G.W. VAN ROOYEN

Cybernetics Laboratory, Faculty of Engineering,
Rand Afrikaans University, P O Box 524, JOHANNESBURG, 2000

II. NOTATION

ABSTRACT: In this paper five new error-correcting (d,k) trellis codes are presented. These codes are suitable for use on bandwidth limited channels where certain input constraints are imposed on the codestream. These codes also facilitate the clock extraction on such channels. The construction techniques for these codes are presented. The codes were evaluated by determining their bit error rates on the binary symmetric channel and measuring their power spectral densities.

I. INTRODUCTION

Modulation codes impose runlength constraints on coded sequences either to comply with the input restrictions of some communication channels or to aid in receiver synchronization and detection processes. Runlength constrained codes, i.e. (d,k) codes, find application on digital magnetic and optical recorders. The parameters d and k are respectively the minimum and maximum number of code "zeros" between successive code "ones" in the non-return-to-zero-inverse (NRZI) representation. These (d,k) codes usually do not have error correcting capabilities. The current widely used scheme to include error correcting capabilities in the code stream, is the "concatenated" scheme, where the information is precoded by an outer error correcting scheme and the error correcting code sequence is encoded by an inner constrained code.

Recently, "combined codes", i.e. (d,k) constrained recording codes capable of correcting errors, have been published [1-6]. Indications are that higher code rates and hence information rates can be obtained by the use of combined codes rather than concatenated codes [6]. However, few simple combined trellis codes, have been published so far. In this paper we present five new such codes. Some of these simple codes complement the known codes while others represent improvements with regards to either free distance d_{∞} , or actual bit error rate performance, or rate $R = m/n$, or the parameter k , or complexity.

The codes we constructed are trellis codes, i.e. the codes have some form of memory inherent in their structure. There are several known methods to describe codes with memory. We briefly discuss the different representations of codes with memory (also known as finite state machine (FSM) codes) as used in the construction of the new codes.

ii.i Transition diagrams

The transition diagram can be considered as a directed graph and is a structure composed of *vertices*, drawn as small circles and of edges or *orientated branches*, drawn as lines between pairs of vertices, with arrow signs pointing from one vertex to the other. A transition diagram describing an \mathcal{K} -state machine contains \mathcal{K} vertices, each vertex representing a different state; the state represented by a vertex is identified by the label attached to this vertex. The branch connecting two vertices is labelled "input alphabet symbol / output alphabet symbol." By construction, a branch pointing from vertex σ_i to vertex σ_j places in evidence the input symbols which cause the machine to pass from state σ_i into state σ_j and the output symbols which accompany the passage. Every unique input symbol causes every state to pass into exactly one other state; consequently the branches originating from any given vertex are labeled with the total number of r input-output pairs, where r is the size of the input alphabet.

ii.ii The transition matrix

The transition matrix is the mathematical counterpart of the transition diagram; it enables one to carry out mechanically a number of operations which, in the transition diagram, can be carried out visually.

For each input symbol we define:

the output matrix Γ of dimension $\mathcal{K} \times \mathcal{R}$;
the state transition matrix T , which is an $\mathcal{K} \times \mathcal{K}$

binary matrix with (i, j) th entry

$$T(i, j) \equiv \begin{cases} 1, & \text{if state } i \text{ is connected to state } j, \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where \mathcal{R} is the number of input symbols and \mathcal{S} is the number of states

ii.iii The edge array

We now present a new representation of a transition diagram by way of an *edge array* A . A similar representation has been used in multilevel line codes by Benedetto *et. al* [7], which they called the symbol transition matrix. The edge array combines the matrices Γ and T into one array A which is defined as follows:

The entries, a_{ij} , of the edge matrix is given by:

$$a_{ij} \triangleq \begin{cases} \text{edge label,} & \text{if there is an edge} \\ & \text{from state } i \text{ to state } j \\ \text{"-" (empty space),} & \text{if there is no edge} \\ & \text{from state } i \text{ to state } j \end{cases} \quad (2)$$

The edge array is suitable for the representation of either FSM or FSTD as the entries are the labels of the edges. For the purpose of this study, the edge array for a FSTD is denoted by A , and for a FSM by F

III. CONSTRUCTION TECHNIQUES FOR THE NEW ERROR-CORRECTING CONSTRAINED CODES

iii.i Sequence state method.

In this section we discuss a coding technique that is an extension of the sequence state technique developed by Franaszek *et al.* [8]. The extension of Franaszek's method involves adding criteria that guarantees a certain free distance to the criteria of the sequence state method. Using the sequence state method, we do not have to concern ourselves with the concatenability of the individual code symbols as this is taken care of by the FSTD approach of the constrained code construction technique. The new method can be summarized as follows:

From the Markov model (FSTD) describing the code constraints we develop a higher order graph that satisfies the sequence state criteria of having one or more subsets of states that have enough transitions to certify a specified rate. From these subsets we choose a subset that guarantees a certain free distance according to the following criterion:

Criterion 1

The free distance obtainable from a certain subset is at least as large as the sum of the minimum Hamming distance between any two code symbols on the diverging edges of any state and the minimum Hamming distance between any two code symbols on the remerging edges of any state. If the constrained symbols on any edge is not used on any other edge of the FSTD, then the code will not suffer from a distance building deficiency. Using the edge array notation, the free distance is at least as large as the sum of the minimum Hamming distance between symbols in any row and the minimum Hamming distance between the symbols in any column. Thus,

$$d_{\infty} \geq \min_i [\min_j [d_H(i) + \min_j [d_H(j)]]] \quad (3)$$

where d_H denotes the Hamming distance between code symbols and i, j respectively the rows and columns of the edge matrix A . Thus $d_H(i)$ denotes the Hamming distance between code symbols in row i .

A subset that satisfies all the above conditions has enough transitions to certify a specified rate and build a certain free distance (found by using criterion 1). These codes are non-catastrophic. An example illustrates the use of this method more clearly.

Example : We develop a rate $R = 1/4$, $(d, k) = (1, 2)$, $d_{\infty} = 3$ code as follows: From the Markov model describing the (d, k) constraints (see figure 1) we find the higher order edge graph G^4 . Using the edge array notation, we note that the graph has a higher order edge matrix A as shown in equation (4).

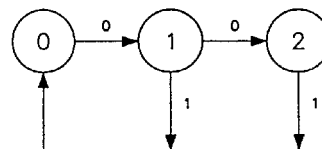


Figure 1 Markov model for the $(d, k) = (1, 2)$ constraints

$$A = \begin{bmatrix} 0101 & 0010 & 0100 \\ 0101 & & 1010 & 0100 \\ 1001 & & & \\ 1001 & 1010 & - & \end{bmatrix} \quad (4)$$

Consideration of this matrix shows that there are at least 2 entries per row, thus there are enough edges leaving each state to construct a rate $R = 1/4$ code. Thus this graph satisfies the state sequence conditions of Franaszek. When constructing a normal runlength limited (RLL) code we would now have eliminated certain transitions to find a subset that has the smallest and thus the least complex FSTD. However, to achieve error-correcting capabilities, we eliminate those transitions that do not contribute any distance building properties. As mentioned previously, the rows indicate outgoing edges and the associated code symbols while the columns indicate incoming edges and their associated code symbols. We now find a submatrix of A that builds the largest free distance and that does not suffer from the distance building deficiency. Careful consideration of A shows that a such a subset, A_r , exists that guarantees a free distance $d_\infty = 3$.

$$A_r = \begin{bmatrix} 0101 & 0010 \\ 1001 & 1010 \end{bmatrix} \quad (5)$$

The minimum Hamming distance between code symbols per row is 2 (row 2) and the minimum Hamming distance between code symbols per column is 1 (column 2). Thus the minimum free distance achievable is three. In this case, due to the simplicity of the final FSTD, the actual free distance $d_\infty = 3$. This scheme is non-catastrophic as all code symbols differ from each other and occur only once in the array. We now map the set $\{0,1\}$ onto the edges to form the FSM that describes a rate $R = 1/4$, $d_\infty = 3$, $(d,k) = (1,2)$ code as shown in (6).

$$F = \begin{bmatrix} 0/0101 & 1/0010 \\ 0/1001 & 1/1010 \end{bmatrix} \quad (6)$$

Applying the above procedure, a rate $R = 1/5$, $(d,k) = (1,2)$, $d_\infty = 6$ code was also constructed. We present the array F which describes the FSM of this code.

$$F = \begin{bmatrix} 0/00101 & 1/01010 \\ 0/01001 & 1/10010 \end{bmatrix} \quad (7)$$

The minimum Hamming distance between code symbols per row is 4 and the minimum Hamming distance between code symbols per column is 2. Thus the minimum free distance is six. This code also is non-catastrophic. Criterion 1 is sufficient to guarantee a certain free distance. It is however possible to achieve better distance building properties by changing the FSM structure to extend the length of remerging paths. To evaluate the free distance

achieved this way is very difficult as the codes are not linear. We have to investigate every possible path and check that the code does not have a distance building deficiency. The codes developed in this section are very simple, low rate codes. We now investigate a method to achieve higher rates.

iii.ii Shift register graph structure.

The two specific codes developed in section ii.ii have two states each and the merging of paths is possible after two steps. In this section we attempt to increase the length of remerging paths by finding a FSTD that has such a property and mapping constrained symbols onto the edges of this diagram. This is done by considering the contents of a shift register as the state number and determining the edges by considering the input to the shift register that causes the shift register to change state. This approach has the advantage of determining a graph structure that gives the longest possible path lengths before remerging. A two bit shift register graph is shown in figure 2. This two bit shift register graph has the property that after two paths have split, at least three edges in the graph are transversed before the paths remerge in any other state. These graphs also have a fully symmetrical structure, i.e. looking from any state, the same structure is observed.

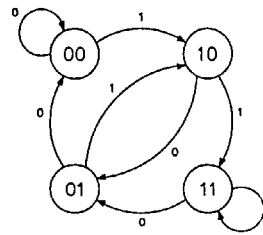


Figure 2 Two bit shift register graph.

To construct a certain code we have to map constrained symbols onto the edges of the graph. We start by using a permutation of the two bit shift register graph that allows some freedom in the combination of constrained symbols and the FSTD.

We construct a rate $R = 1/3$, $(d,k) = (1,3)$, $d_\infty = 4$ code using a permutation of a four state, two bit shift register graph. The set of constrained symbols of length 3 satisfying the $(d,k) = (1,3)$ constraints is $\{100, 101, 010, 001\}$. When mapping these code symbols onto the edges of the graph, great care has to be taken to ensure that the constraints are preserved. We used a four state shift register graph that has the following connection matrix T:

$$T = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \quad (8)$$

Careful mapping of the constrained symbols onto this graph gives an array A that satisfies the $(d,k) = (1,3)$ constraints. The array F that describes the FSM of this code is found by mapping the set of one bit binary symbols onto the edges of A . Note that this code is systematic; the last constrained code bit is the same as the information bit.

$$F = \begin{bmatrix} 0/100 & 1/101 & - & - \\ - & - & 1/001 & 0/010 \\ - & - & 0/010 & 1/001 \\ 0/010 & 1/001 & - & - \end{bmatrix} \quad (9)$$

Applying criterion 1 to this array, we find that the minimum Hamming distance per row is one (row 1) while the minimum Hamming distance per column is also one (column 2). A thorough inspection of all the possible diverging and remerging paths showed that the free distance is $d_{\infty} = 4$, which is larger than what is expected if only criterion 1 is applied. This effect is due to the structure of the shift register graph. This new code also does not suffer from the distance building deficiency although there are code symbols repeated in the graph. This is also an attribute of the graph, there is however not a known way to determine under which circumstances the code will always build distance.

Using a similar mapping technique an *ad hoc* rate $R = 1/3$, $(d,k) = (1,5)$, $d_{\infty} = 3$ was constructed. This mapping was done on a particular three state graph of which the transition matrix is shown in equation (10). A three state graph cannot be a shift register graph and does not have the property of at least three transitions before merging as does the four state shift register graph, but it improves on a two state graph in the sense that repetition of code symbols can be accommodated without losing distance building properties. We present the edge array F for this code.

$$T = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad (10)$$

$$F = \begin{bmatrix} - & 1/000 & 0/101 \\ 1/010 & - & 0/100 \\ - & 1/010 & 0/001 \end{bmatrix} \quad (11)$$

iii.iii Violation of a Hamming distance preserving mapping.

In [4] the Hamming distance preserving mapping is presented as a method to construct constrained error-correcting codes. The Hamming distance preserving mapping construction method involves the one-to-one mapping of the unconstrained symbols of a convolutional base code, onto constrained code symbols, satisfying (d,k) constraints, according to certain rules as explained in [4]. The most important rule is that the Hamming distance between any two constrained symbols must be greater or equal to the Hamming distance between the two unconstrained symbols mapped onto them. It is possible to achieve higher rate codes if a violation of the Hamming distance preserving criterium can be allowed. In this section we develop a code that utilizes the properties of a specific convolutional code to allow a violation of the Hamming distance preserving mapping to construct a constrained error-correcting code. A summary of the method is given along with a code developed from the application of the method.

iii.iv. Summary of Hamming distance preserving mapping violation method.

Let a set of constrained symbols large enough to allow a one-to-one mapping violating the Hamming distance array be given. To determine when a violation of the Hamming distance preserving mapping is allowable, the following steps can be used.

Step 1 : Choose a suitable convolutional code that maximizes the rate of the combined code and allows a one-to-one mapping of unconstrained symbols onto constrained symbols.

Step 2 : Determine those constrained symbols that violate the Hamming distance preserving mapping criterium as well as their antecedents.

Step 3 : Determine the convolutional code sequences that contain $d_{\infty} - 1$ non-zero symbols.

Step 4 : Find those unconstrained symbols that are influenced by the violations by comparing the antecedents of the constrained symbols that violate the Hamming distance preserving mapping criterium to the sequences determined in step 3. If there are none such unconstrained symbols, then the violated Hamming distance preserving mapping can be used without losing any free distance. If there are unconstrained symbols that occur in the sequences determined in step 3 and are the antecedents of constrained symbols that violate the Hamming distance preserving mapping criterium, continue to step 5

Step 5 : Investigate the other non-zero symbols of the sequence/s that contain the symbols found in step 4. Determine the distance built by these unconstrained symbols and thus find the distance that needs to be built by the unconstrained symbols that are the antecedents of constrained symbols that violate the Hamming distance mapping criterium. With this information a modified Hamming distance array, is found that would not cause the loss of free distance in the mapping of the unconstrained symbols of the base code onto the constrained symbols.

Step 6 : Compare the original constrained symbol Hamming distance array with the new array. Using the Hamming distance preserving mapping criteria, determine a valid mapping, if possible.

The code we developed using this approach is a rate $R = 2/5$, $(d,k) = (1,6)$, $d_{\infty} = 3$ code described by the following edge array:

$$F = \begin{bmatrix} 00/00000 & 10/10100 & 01/01010 & 11/10010 \\ 00/10000 & 10/00100 & 01/01000 & 11/00010 \\ 00/01000 & 10/00010 & 01/10000 & 11/00100 \\ 00/01010 & 10/10010 & 01/00000 & 11/10100 \end{bmatrix} \quad (12)$$

It is interesting to note that this code also satisfies the minimum free distance criterium as described in (3) of having $d_{\infty} = 3$.

IV. RESULTS.

We evaluate the codes we have constructed according to two performance criteria. The first is the performance of the code on a simulated binary symmetric channel. We have measured the bit error rate performance of the different new codes on this channel. We show the bit error rate for the rate $R = 1/3$, $(d,k) = (1,3)$, $d_{\infty} = 4$ in figure 3 and list a constant κ (which relates the logarithm of the bit error probability to the logarithm of the channel error probability) for the other new codes in table I. When we consider the bit error rate curves of the new codes we notice that their gradients vary according to the error-correcting capabilities of the code. For a code that has a free distance $d_{\infty} = 3$, we find that the bit error rate curve has a gradient greater or equal to $t+1 = 2$ on the logarithmic scale. Similarly, a $d_{\infty} = 6$ code has a gradient of at least $t+1 = 3$. This is a way to verify the error-correcting capability of a code. All the new codes satisfy this condition. Another property of the codes we determined is the power spectrum. This was measured on a bench test developed by Van Rooyen [9]. This test determines the spectral response of (d,k) limited codes over

bandwidth limited channels. The results of these tests are shown in figures 4 - 8.

TABLE I
BIT ERROR RATE CONSTANTS FOR THE NEW
COMBINED ERROR-CORRECTING
CONSTRAINED CODES

Parameters			
d	k	d_{∞}	κ
1	2	3	1,65
1	2	6	2,5
1	3	4	1,5
1	5	3	1,85
1	6	3	1,3

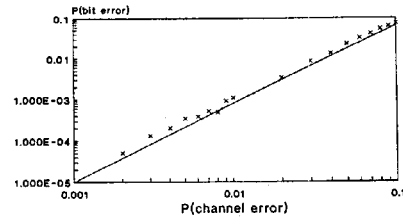


Figure 3 Bit error rate of the $R = 1/3$, $(d,k) = (1,3)$, $d_{\infty} = 4$ code.

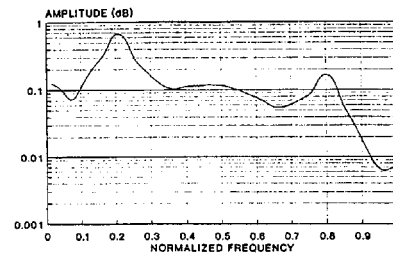


Figure 4 Power spectrum of the $R = 1/5$, $(d,k) = (1,2)$, $d_{\infty} = 6$ code.

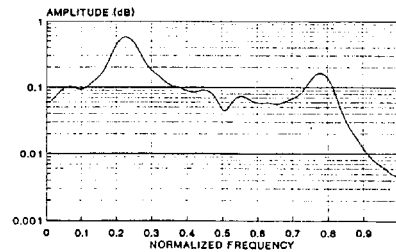


Figure 5 Power spectrum of the $R = 1/4$, $(d,k) = (1,2)$, $d_{\infty} = 3$ code.

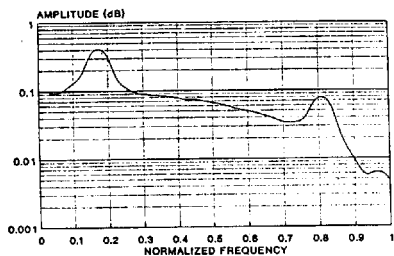


Figure 6 Power spectrum of the $R = 1/3, (d,k) = (1,3), d_{\infty} = 4$ code.

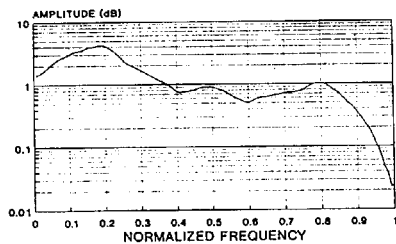


Figure 7 Power spectrum of the $R = 1/3, (d,k) = (1,5), d_{\infty} = 3$ code.

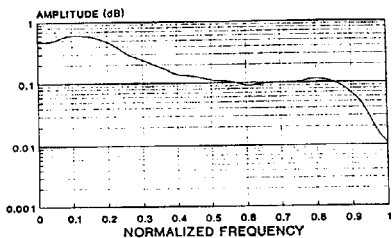


Figure 8 Power spectrum of the $R = 2/5, (d,k) = (1,6), d_{\infty} = 3$ code.

V. SUMMARY

We have presented five new error-correcting codes of relevant constraints and importance. These codes were constructed using a combination of systematic and empirical techniques which are extensions of known methods to build either constrained codes or error-correcting codes. These new codes were compared to known codes of the same class and order.

In some cases the new codes supplement the known codes and in other cases the codes improve on known codes with regards to either rate, error-correcting capabilities or complexity.

REFERENCES

- [1] HC Ferreira, J. F. Hope and A. L. Nel, "Binary rate four eighths, runlength constrained, error correcting magnetic recording modulation code," IEEE Trans. Magn., Vol 22, pp. 1197-1199, September 1986.
- [2] P Lee and J. K. Wolf, "Combined error correction/modulation coding," IEEE Trans. Magn., Vol 23, pp. 3681-3683, September 1987.
- [3] Y Lin and J. K. Wolf, "Combined ECC/RLL codes," IEEE Trans. Magn., Vol 24, pp. 2527-2529, November 1988.
- [4] HC Ferreira, D. A. Wright and A. L. Nel, "Hamming distance preserving mappings and trellis codes with constrained binary symbols," IEEE Trans. on Information Theory, Vol 35, pp 1098-1103, September 1989.
- [5] CA French and Y. Lin, "Performance comparison of combined ECC/RLL codes," presented at IEEE ICC '90 Conf., Atlanta, 1990.
- [6] M Blaum, "Combining ECC with modulation: Performance comparisons," Proc. IEEE Globecom Conf., San Diego, pp. 901.3.1-901.3.4, December 1990.
- [7] S. Benedetto, V. Castellani and G. de Vincentiis, "On a class of non-linear multilevel codes for synchronous data transmission," Proceedings of the IEEE International Conference on Communications, Seattle, Vol. 1, pp. 12-17, 1973.
- [8] P.A. Franaszek, "Sequence-state encoding for digital transmission," Bell Syst. Techn. J., Vol. 47, pp. 143 - 157, Jan 1968.
- [9] P.G.W. van Rooyen, "Modulation codes for mobile communications," Thesis for Masters degree in Electrical and Electronic Engineering, Rand Afrikaans University, 1991.