RZ 3723 (# 99733) 01/12/2009 (updated 03/24/2009)
Electrical Engineering 15 pages

Research Report

Efficient Encoding of a Class of Maximum-Transition-Run and PRML Codes

(Updated version of March 24, 2009)

Thomas Mittelholzer

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Efficient Encoding of a Class of Maximum-Transition-Run and PRML Codes

Thomas Mittelholzer

IBM Research, Zurich Research Laboratory, Rüschlikon, Switzerland

March 12, 2009

Abstract

A large class of maximum-transition-run (MTR) block codes is presented that is based on a novel low-complexity enumerative encoding scheme. This new class of MTR codes is obtained by a general design method to construct capacity-efficient MTR codes with predetermined j and k constraints and reduced error propagation at the decoder. Typically, these codes are designed to improve the distance properties of a generalized partial-response detector trellis. Another way to use MTR codes consists in applying even/odd interleaving to construct long high-rate PRML(G,I,M) modulation codes that satisfy tight global $G = 2k$ and interleaved $I = k$ constraints. Furthermore, these (G,I) constrained codes satisfy the $M = 2j$ constraint, i.e., they have limited runs of alternating $2T$ magnets at the channel input.

Keywords

Modulation codes, enumerative encoding, MTR(j, k)-constraints, G, I and M constraints, reverse concatenation.

I. Introduction

In magnetic recording and optical storage, modulation constraints on the recorded binary sequences are employed to facilitate timing recovery and ensure efficient operation of the detector [1]. A frequently used modulation constraint is the k constraint, which limits the maximum number of consecutive zeros to k. In this report, we will focus on the maximum transition run (MTR) constraint, which has been introduced by Moon and Brickner to provide coding gain for extended partial response channels, which are based on partial-response maximum likelihood (PRML) detection [2]. Typically, MTR codes satisfy two constraints, the k constraint and the j constraint. Recall that the j constraint is dual to the k constraint, i.e., it limits the maximum number of consecutive ones in a binary sequence to j.

The construction of MTR codes with a j constraint and no limitations on the k constraint can be approached through Fibonacci codes, which were introduced by Kautz [3]. By applying bit inversion to all codewords in a Fibonacci code, the j constraint is transformed into a k constraint. Thus, for a pure j constraint or a pure k constraint, the Fibonacci codes and the bit-flipped Fibonacci codes form a class of highly efficient enumerative modulation codes. However, in the case of mixed MTR constraints, i.e., with finite values of j and k, the class of Fibonacci codes does not achieve the desired constraints. For these mixed MTR constraints, a novel enumerative encoding/decoding scheme is presented that is a generalization of the Fibonacci codes and generates MTR modulation codes with predetermined j and k constraints.

In storage systems, user data is typically first encoded by an error-correcting code (ECC) before it is passed through the modulation encoder, and therefore error propagation at the modulation decoder is an important practical issue. For this reason, most practical modulation block codes are relatively short. Some MTR codes presented are especially designed to have reduced error propagation. In fact, error propagation to the left cannot be controlled because of the carry propagation in the enumerative decoder; however, the maximum error propagation to the right (in bits) is a design parameter, which can be suitably selected in the construction of the enumerative MTR codes proposed here.

In a reverse concatenation (RC) scheme, the order of the ECC encoder and the modulation encoder is reversed, i.e., user data is first modulation-encoded before it is passed through the ECC encoder. Hence, during read back, the ECC decoder operates before the modulation decoder and, thus, there is no error propagation except for the extremely rare event that the ECC decoder makes an error. For this reason, one can choose to use long modulation codes in an RC scheme, which operate close to the capacity of the constraint selected. RC architectures have been considered in various papers [4], [5], [6], [7] and, recently, have been implemented in hard-disk drive products. For the RC framework, a design method for long, high-rate MTR codes is presented and illustrated by numerous examples of capacity-efficient codes.

The report is organized as follows. In Section II, the new MTR scheme is introduced with its enumerative encoder/decoder algorithms. In Section III, a design method for practical MTR codes with efficient encoders/decoders is presented. Section IV illustrates the construction of long, high-rate PRML(G, I, M) modulation codes for PRML recording schemes from long, high-rate MTR codes. In Section V, the implementation aspects of MTR encoders/decoders are discussed and compared to those of generalized Fibonacci codes.

II. Enumerative Encoding of MTR Codes

Suppose that finite j and k constraints are specified, where, without loss of essential generality, one can assume $j \leq k$ (otherwise one applies bit inversion to all codewords, which interchanges the j and the k constraints). For each N, a MTR block code of length N will be defined based on a set of weights $\{(v_n, w_n)\}$, $n = 1, 2, \ldots, N$, which will be called *MTR-weights*. The sequences $\{v_n\}$ and $\{w_n\}$ will be called *k-weights* and *j-weights*, respectively. They are determined by initial conditions and linear recursions, which reflect the k and the j constraints. The initial MTR-weights are defined by

$$
\begin{aligned}
v_n &= 0 && n = 1, 2, \ldots, k+1 && (1)\\
w_n &= 2^{n-1} && n = 1, 2, \ldots, j+1, && (2)
\end{aligned}
$$

and the two interlinked recursions are given by

$$
\begin{aligned}
v_{n+1} &= v_{n-k} + w_{n-k} && n \geq k+1 && (3)\\
w_{n+1} &= w_n + w_{n-1} + \ldots + w_{n-j} - v_{n+1} + v_{n-j} && n \geq j+1. && (4)
\end{aligned}
$$

Note that the j-weights form a monotonically increasing sequence and the k-weights are monotonically non-decreasing.

We start with the description of an encoder of a code of length N. First, we define the input space U, which consists of all integers u in the half-open interval $[L_N, U_N)$ with lower and upper boundaries L_N and U_N given by

$$
\begin{aligned}
L_N &= v_{N-k} + w_{N-k} && (5)\\
U_N &= w_N + w_{N-1} + \ldots + w_{N-j} + v_{N-j}. && (6)
\end{aligned}
$$

Thus, $U = \{u : L_N \leq u < U_N\}$. There is an obvious way to map binary inputs $u_1, u_2, \ldots, u_N$ into integers u in the input space U, namely by adding the offset L_N to the sum $\sum_{i=1}^{N} u_i 2^{N-i}$, i.e., by the assignment $u = \sum_{i=1}^{N} u_i 2^{N-i} + L_N$. Moreover, one needs to ensure that u does not exceed the upper bound U_N by suitably restricting the binary inputs. In the remaining part of this Section, we will not be concerned with this mapping, and we will assume that the input is an integer u belonging to the input space U.

Given an input $u \in U$, the encoder computes the binary output sequence $x_1, x_2, \ldots, x_N$ based on the following algorithm:

$$\begin{array}{l} \text{For } \ell = 1 \text{ to } N \text{ do:} \\ \quad \text{if } u \geq v_{N-\ell+1} + w_{N-\ell+1} \\ \qquad x_\ell = 1,\ u = u - w_{N-\ell+1} \\ \quad \text{else} \\ \qquad x_\ell = 0. \end{array} \tag{7}$$

The encoding algorithm leads to a representation of the input as a weighted sum with binary coefficients

$$u = \sum_{\ell=1}^{N} x_\ell w_{N-\ell+1}. \tag{8}$$

The decoder is based on this weighted-sum representation. Given a codeword $x_1, x_2, \ldots, x_N$, the decoder first initializes the output to $u = 0$ and then performs the following N steps:

$$\text{For } n = 1 \text{ to } N \text{ do:} \qquad \text{if } x_{N-n+1} = 1 \text{ then } u = u + w_n\ . \tag{9}$$

The code of length N, which consists of all output sequences $x_1, \ldots, x_N$ generated by inputs $u \in U$, will be called a *strict-sense enumerative maximum transition run code* with parameters j and k. The encoding and decoding algorithms (7) and (9) will be referred to as eMTR-encoder and eMTR-decoder.

While maintaining the same encoding and decoding algorithms (7) and (9), one can weaken the defining equations (1) – (6) for the MTR-weights and obtain a larger class of codes. Specifically, one requires the j-weights to be positive integers, while generalizing the initial conditions and the two recursions for the MTR-weights to

$$\begin{array}{rcll} v_n & \geq & 0 & n = 1, 2, \ldots, k+1 \qquad (10) \\ w_n & \leq & 2^{n-1} & n = 1, 2, \ldots, j+1 \qquad (11) \\ v_{n+1} & \geq & v_{n-k} + w_{n-k} & n = k+1, \ldots, N-1 \qquad (12) \\ w_{n+1} & \leq & w_n + w_{n-1} + \ldots + w_{n-j} - v_{n+1} + v_{n-j} & n = j+1, \ldots, N-1. \qquad (13) \end{array}$$

Moreover, the lower and upper bounds of the input space can be chosen as

$$\begin{array}{rcl} L_N & \geq & v_{N-k} + w_{N-k} \qquad (14) \\ U_N & \leq & w_N + w_{N-1} + \ldots + w_{N-j} + v_{N-j}. \qquad (15) \end{array}$$

The resulting codes will be called *enumerative maximum transition run (eMTR)* codes and a code with parameters (N, j, k) will be called an eMTR(j, k) code of length N. Note that for a given set of parameters (N, j, k), there is exactly one strict-sense eMRT(j, k) code of length N, but there are many possible length-N eMTR(j, k) codes. The more general recursions and lower and upper bounds (10) – (15) allow one to construct practical codes with tighter constraints at the codeword boundaries and to design complexity-efficient encoders/decoders. This comes at the price of a slight reduction in the number of codewords compared with strict-sense eMRT codes, which have the largest number of codewords for a given set of parameters (N, j, k).

Next, we will derive the basic properties of eMTR codes. In particular, we will show that the codes satisfy the predetermined j and k constraints and that the encoding is indeed based on codeword enumeration.

Proposition 1: eMTR(j, k) codes satisfy the j constraint.

Proof:
Case 1 (no j constraint violation at the left boundary):

As the input u lies in the input space U, one has $u < U_N$, which by (15) implies

$$u < w_N + w_{N-1} + \ldots + w_{N-j} + v_{N-j}. \tag{16}$$

Suppose that $j+1$ or more consecutive components $x_1, x_2, \ldots, x_{j+1}, \ldots$ were one, which means that in the first $j+1$ steps in the encoding algorithm (7), one has

$$\begin{aligned} u &\geq v_N + w_N \\ u - w_N &\geq v_{N-1} + w_{N-1} \\ u - w_N - w_{N-1} &\geq v_{N-2} + w_{N-2} \\ &\vdots \\ u - w_N - w_{N-1} \ldots - w_{N-j+1} &\geq v_{N-j} + w_{N-j}. \end{aligned}$$

The last inequality can be rewritten as

$$u \geq v_{N-j} + w_N + w_{N-1} + \ldots + w_{N-j+1} + w_{N-j},$$

and this is a contradiction to (16). Hence, there cannot be $j+1$ consecutive ones and, thus, the j constraint must be satisfied at the left boundary.
Case 2 (no j constraint violation inside a codeword):

Suppose $x_n = 0$ and $j+1$ or more components $x_{n+1}, x_{n+2}, \ldots, x_{j+1}, \ldots$ were one. Based on (7), the condition $x_n = 0$ translates into the inequality

$$u^{(n)} < v_{N-n+1} + w_{N-n+1} \tag{17}$$

for the input $u = u^{(n)}$ at step n in the algorithm, which also implies $u^{(n+1)} = u^{(n)}$ for the input at step $n+1$. The run of $j+1$ consecutive ones implies that at steps $n+1, n+2, \ldots, n+j+1$ in the algorithm, one has the following inequalities:

$$\begin{aligned} u^{(n+1)} &\geq v_{N-n} + w_{N-n} \\ u^{(n+1)} - w_{N-n} &\geq v_{N-n-1} + w_{N-n-1} \\ u^{(n+1)} - w_{N-n} - w_{N-n-1} &\geq v_{N-n-2} + w_{N-n-2} \\ &\vdots \\ u^{(n+1)} - w_{N-n} - w_{N-n-1} \ldots - w_{N-n-j+1} &\geq v_{N-n-j} + w_{N-n-j}. \end{aligned}$$

Making use of (13), the last inequality can be rewritten as

$$\begin{aligned} u^{(n+1)} &\geq v_{N-n-j} + w_{N-n} + w_{N-n-1} + \ldots + w_{N-n-j+1} + w_{N-n-j} \qquad (18) \\ &\geq v_{N-n+1} + w_{N-n+1}. \qquad (19) \end{aligned}$$

This last inequality (19) is a contradiction to (17) because $u^{(n+1)} = u^{(n)}$. Hence, the j constraint cannot be violated. ■

Proposition 2: eMTR(j, k) codes satisfy the k constraint.

Proof:
Case 1 (no k constraint violation at the left boundary):
If $k+1$ or more consecutive components $x_1, x_2, \ldots, x_{k+1}, \ldots$ were zero, then there is no weight-substraction in the first $k+1$ steps in the encoding algorithm (7) and, thus,

$$u < v_{N-k} + w_{N-k}, \tag{20}$$

which contradicts the fact that $u \in U$, in particular, $v_{N-k} + w_{N-k} \leq u$. Hence, there cannot be $k+1$ consecutive zeros and, thus, the k constraint must be satisfied at the left boundary.
Case 2 (no k constraint violation inside a codeword):
Suppose $x_n = 1$ and $k+1$ or more components $x_{n+1}, x_{n+2}, \ldots, x_{k+1}, \ldots$ were zero. Based on (7), the condition $x_n = 1$ translates into the inequality

$$u^{(n)} \geq v_{N-n+1} + w_{N-n+1} \tag{21}$$

for the input $u = u^{(n)}$ at step n in the algorithm. The input at step $n+1$ is obtained by subtracting the j-weight w_{N-n+1}, i.e., $u^{(n+1)} = u^{(n)} - w_{N-n+1}$. The consecutive run of $k+1$ zeros implies that at steps $n+1, n+2, \ldots, n+k+1$ in the algorithm, one has the following inequalities:

$$\begin{aligned} u^{(n+1)} &< v_{N-n} + w_{N-n} \\ u^{(n+2)} = u^{(n+1)} &< v_{N-n-1} + w_{N-n-1} \\ u^{(n+3)} = u^{(n+1)} &< v_{N-n-2} + w_{N-n-2} \\ &\vdots \\ u^{(n+k+1)} = u^{(n+1)} &< v_{N-n-k} + w_{N-n-k}. \end{aligned}$$

Making use of (12), the last inequality can be rewritten as $u^{(n+1)} < v_{N-n+1}$, which implies

$$u^{(n)} < v_{N-n+1} + w_{N-n+1} \tag{22}$$

and, thus, is a contradiction to (21). Hence, the k constraint cannot be violated. ■

The eMTR encoder/decoder can be described in terms of enumerative encoding techniques [8]. However, in contrast to [8], the enumeration of codewords does not start at 0 but at the offset L_N. Enumerative encoding is a generic technique for the construction of an encoder/decoder for some block code. To this end, one introduces the lexicographical order on the set of all eMTR(j, k) codewords, which are a subset of all binary length-N sequences. The ordering is chosen such that the leftmost bit x_1 in a sequence $x_1, x_2, \ldots, x_N$ is the most significant bit.

Proposition 3: (a) Enumerative encoding: The eMTR encoder specifies an order-preserving one-to-one map from input space U onto the eMTR(j, k) code of length N.
(b) The length-N eMRT(j, k) code with lower and upper bounds L_N and U_N given by (14) and (15) contains $U_N - L_N$ codewords.

Proof: Part (a): Let u and u' be two input values in U and suppose $u > u'$. Let $\mathbf{x} = [x_1, \ldots, x_N]$ and $\mathbf{x}' = [x'_1, \ldots, x'_N]$ be the corresponding codewords generated by the eMTR encoder. We will show that $\mathbf{x} > \mathbf{x}'$ with respect to the lexicographic ordering by running two eMTR-encoders in parallel, one for the input u and the other for the input u'. Note that as $u \neq u'$, the two codewords $\mathbf{x}$ and $\mathbf{x}'$ must be distinct by (8).

Case 1: Suppose $x_1 \neq x'_1$. As $u > u'$, it follows that in the first step of the encoding algorithm one must have $x_1 = 1$ and $x'_1 = 0$, which implies $\mathbf{x} > \mathbf{x}'$.

Case 2: Suppose the first $\ell - 1$ components of both codewords are the same and $x_\ell \neq x'_\ell$. At the ℓ-th step, the two encoders have as inputs $u^{(\ell)} = u - \sum_{i=1,\ldots,\ell-1} x_i w_{N-i+1}$ and $u'^{(\ell)} = u' - \sum_{i=1,\ldots,\ell-1} x_i w_{N-i+1}$. As $u > u'$, clearly, $u^{(\ell)} > u'^{(\ell)}$ and, at the ℓ-th encoding step, the outputs are $x_\ell = 1$ and $x'_\ell = 0$, which implies $\mathbf{x} > \mathbf{x}'$.

Part (b): Follows from (a) and the fact that the input space U has $U_N - L_N$ elements. ■

III. Practical eMTR codes

Practical codes should maintain tight constraints across codeword boundaries and have simple encoder/decoder implementations. For this reason, practical eMTR codes are based on the generalized recursions (10) to (15).

The parameter $j = j(n)$ in (13), which determines the j constraint, can be chosen to be dependent on the location n. For instance, if $j(n) = 4$, then the encoder (7) generates a codeword that satisfies $x_\ell x_{\ell+1} x_{\ell+2} x_{\ell+3} x_{\ell+4} = 0$, where $\ell = N - n + 1$. Thus, an eMTR code has a j-profile that may depend on the location n in a constrained codeword and thus satisfies non-uniform modulation constraints, e.g., tighter constraints at the codeword boundaries or at some other specified locations [7]. Similarly, the eMTR code has a k-profile that may depend on the location n in a codeword.

Moreover, to achieve efficient encoding and decoding (see [9]), one chooses the MTR-weights v_n and w_n to have a limited span S, i.e., in binary notation each MTR-weight has no more than S non-zero most significant bits, with all lower bits being zero. More generally, the k-weights v_n have a span $S^{(k)}$ and the j-weights w_n have a (possibly different) span $S^{(j)}$. The finite span property ensures that while encoding/decoding codeword bit x_ℓ, the computations at the ℓ-th step can be done by one sliding-window $S^{(j)}$-bit adder (or subtractor) with carry and one sliding-window comparison that is $\max\{S^{(j)}, S^{(k)}\} + 1$ bits wide [7]. In the following, we will always assume $j \leq k$, which is not an essential restriction because the role of j and k are interchanged by inverting all bits in a codeword. Moreover, one typically selects $S^{(j)} \leq S^{(k)}$.

Tighter constraints at the boundaries can be enforced by selecting appropriate initial conditions and suitable lower and upper bounds for the input space. To obtain the tighter constraints at the codeword ending, $j_e < j$ and $k_e < k$, the initial conditions of the recursions are modified as follows:

$$v_n = \begin{cases} 0 & n = 1, 2, \ldots, k_e + 1 \\ 1 & n = k_e + 2, \ldots, k + 1. \end{cases} \tag{23}$$

$$w_n = \begin{cases} 2^{n-1} & n = 1, 2, \ldots, j_e + 1 \\ \sum_{\ell=1}^{n-1} w_\ell - v_n & n = j_e + 2, \ldots, j + 1. \end{cases} \tag{24}$$

The MTR-weights are given by the two interlinked recursions

$$v_{n+1} = 2^{n-S^{(k)}} \lceil 2^{S^{(k)}-n} (v_{n-k} + w_{n-k}) \rceil \qquad n = k+1, \ldots, N-1 \tag{25}$$

$$w_{n+1} = 2^{n-S^{(j)}} \lfloor 2^{S^{(j)}-n} (\sum_{\ell=n-j}^{n} w_\ell - v_{n+1} + v_{n-j}) \rfloor \qquad n = j+1, \ldots, N-1, \tag{26}$$

where $w_n = 0$ for $n \leq 0$ and where $\lceil t \rceil$ and $\lfloor t \rfloor$ denote the smallest integer, which is at least as large as t, and the largest integer not exceeding t, respectively. By construction, the j-weights have span $S^{(j)}$ and the k-weights have span $S^{(k)}$.

To impose tighter constraints $j_b < j$ and $k_b < k$ at the beginning of codewords, one defines the upper and lower bounds on the input space as

$$\begin{aligned} L_N &= v_{N-k_b} + w_{N-k_b} && (27) \\ U_N &= w_N + w_{N-1} + \ldots + w_{N-j_b} + v_{N-j_b}. && (28) \end{aligned}$$

The resulting code of length N contains $U_N - L_N$ codewords. As mentioned in the preceding section, the eMTR encoder is combined with an offset mapper to achieve the encoding of K input bits $u_1, u_2, \ldots, u_K$ into binary codewords $x_1, x_2, \ldots, x_N$ of length N, where $K = \lfloor \log_2(U_N - L_N) \rfloor$. Note that the offset mapper is essentially a binary $S^{(j)}$-bit adder with carry, which adds the $S^{(j)}$ most significant bits of the lower bound L_N to the first $S^{(j)}$ bits of the binary input sequence.

The interlinked recursions (25) – (26) do not make the best use of the allowed j-span for codes with small j and large N. Slightly better weights can be obtained by decomposing each j-weight w_n into a suitable power of two and a "mantissa part" of span $S^{(j)}$. Instead of the simple 2^{n-1} part above, one determines the best exponent for each j-weight. Namely, for $n > j$ one defines the exponent $e(n+1)$ of w_{n+1} as $\lfloor \log_2(v_{n-j} - v_{n-k} - w_{n-k} + \sum_{\ell=n-j}^{n} w_\ell) \rfloor$. The interlinked recursions (25) – (26) are then replaced by

$$\begin{aligned} \tilde{w}_{n+1} &= v_{n-j} - v_{n-k} - w_{n-k} + \sum_{\ell=n-j}^{n} w_\ell && n = j+1\ldots, N-1 \quad (29) \\ e(n+1) &= \lfloor \log_2(\tilde{w}_{n+1}) \rfloor && n = j+1, \ldots, N-1 \quad (30) \\ w_{n+1} &= 2^{e(n+1)+1-S^{(j)}} \lfloor 2^{S^{(j)}-e(n+1)-1}(\tilde{w}_{n+1}) \rfloor && n = j+1, \ldots, N-1 \quad (31) \\ v_{n+1} &= 2^{e(n+1)+1-S^{(k)}} \lceil 2^{S^{(k)}-e(n+1)-1}(v_{n-k} + w_{n-k}) \rceil && n = k+1, \ldots, N-1. \quad (32) \end{aligned}$$

Examples of good eMTR codes are presented below. We recall that all these eMTR codes are designed to be used with a $1/(1 \oplus D)$ precoder. To construct long eMTR codes, it is sufficient to specify the spans $S^{(j)}$ and $S^{(k)}$, the length N, the constraints j, k, and the boundary constraints j_b, j_e and k_b, k_e. Given these parameters, the MTR-bases are fully determined either by the simpler set of equations (23) – (28) or by the tighter ones, where (25) – (26) are replaced by (29) – (32).

A. An eMTR($j = 2, k = 7$) Code of Rate-16/19

In Table I, an eMTR code of length $N = 19$ with MTR-weights v_n and w_n is specified. The table also displays the j and k profiles. The tight constraints at the codeword boundaries ensure that the $j = 2$ and $k = 7$ constraints are maintained across codeword boundaries. The lower and upper bound on the input space U are $L_{19} = 8065$ and $U_{19} = 75482$. Therefore, the code contains 67417 codewords and gives raise to a rate-16/19 block code. The capacity of the $j = 2$, $k = 7$ constraint is 0.873230 (rounded up to 6 decimal digits), which results in a rate efficiency of 96.4357%. Note that there exists no length-19 ($j = 2, k = 6$) code of rate 16/19 [2].

The k-weights v_n have no limitation on their span, and thus the encoder needs to perform comparison operations that are up to $N = 19$ bits wide. However, the j-weights w_n have a

TABLE I
WEIGHTS AND CONSTRAINTS PROFILE OF AN eMTR($j = 2, k = 7$) CODE OF RATE-16/19.

n	1	2	3	4	5	6	7	8	9	10
w_n	1	2	3	6	11	19	35	64	118	216
v_n	0	0	0	0	0	1	1	1	1	2
j_n	1	1	2	2	2	2	2	2	2	2
k_n	1	2	3	4	4	5	6	7	7	7
n	11	12	13	14	15	16	17	18	19	
w_n	396	720	1312	2400	4352	8000	14592	26624	48640	
v_n	3	6	11	20	36	65	119	218	399	
j_n	2	2	2	2	2	2	2	2	1	
k_n	7	7	7	7	7	7	7	7	3	

span of $S^{(j)} = 7$. This implies that a bit error in the detector will not propagate through more than 7 bits to the right at the modulation decoder output. Such reduced error propagation is a desirable property that helps to reduce the byte-error rate at the output of the modulation decoder.

The code satisfies an additional constraint – known as the twins constraint (or t constraint) [10], that excludes quasi-catastrophic error propagation on a detector trellis for a partial-response polynomial of the form $(1 - D^2)(1 - P(D))$, where $1 - P(D)$ has no roots on the unit circle. In particular, the $j = 1$ constraint at the codeword boundaries enforce a twins constraint $t = 19$. By eliminating the two sequences $\mathbf{y}_1 = 1001100110011001100$ and $\mathbf{y}_2 = 0011001100110011001$ from the code, one can enforce a $t = 17$ constraint. The two sequences correspond to the input values $u_1 = 62216$ and $u_2 = 24824$. Hence, an encoder for this $t = 17$ constraint code is obtained from the eMTR encoder (7) by a simple modification, viz., the omission of u_1 and u_2 from the input space. To encode that eMTR$(2, 7)$ code with $t = 17$ of rate-16/19, one can choose the input space to consist of all integers in the range $L_{19} \le u < L_{19} + 2^{16} + 1$, except for the values u_1 and u_2. By deleting further codewords, the twins constraint can be slightly tightened further; however, at a price of a more complex encoder/decoder.

B. An eMTR($j = 3, 4, k = 18$) Code of Rate-16/17

An eMTR code of length $N = 17$ with MTR-weights v_n and w_n is specified in Table II. The table also displays the j and k profile. The code satisfies a $j = 3$ constraint except for the boundary where across codeword boundaries a $j = 4$ constraint is maintained. Furthermore, the code ensures a $k = 18$ constraint. The lower and the upper bound on the input space U are $L_{17} = 100$ and $U_{17} = 65654$, hence, the code contains 65554 codewords. Proceeding as in [10], one can delete 17 codewords that either start in the first 15 bits with periodic patterns of period 4 of the form 00110011..., 01100110..., 11001100..., and 10011001... or end in the last 16 bits with one of these periodic patterns of period 4. The resulting code with 65537 codewords satisfies a twins constraint with $t = 14$. This generalized eMTR($j = 3, 4, k = 18, t = 14$) code has a simple enumerative encoder/decoder.

Following [10], one can construct a look-ahead rate-16/17 sliding block code (see Fig. 9 in [10]) based on the eMTR($j = 3, 4, k = 18, t = 14$) code by applying the substitution Table X in [10]. The resulting code satisfies the constraints $j = 3$, $k = 12$, $t = 9$ and has a rather simple

TABLE II

WEIGHTS AND CONSTRAINTS PROFILE OF AN EMTR($j = 3, 4, k = 18$) CODE OF RATE-16/17.

n	1	2	3	4	5	6	7	8	9
w_n	1	2	4	7	14	27	52	100	193
v_n	0	0	0	0	0	0	0	0	0
j_n	1	2	2	3	3	3	3	3	3
k_n	1	2	3	4	5	6	7	8	9
n	10	11	12	13	14	15	16	17	
w_n	372	716	1380	2660	5127	9883	19050	36720	
v_n	0	1	1	1	1	1	1	1	
j_n	3	3	3	3	3	3	3	2	
k_n	9	10	11	12	13	14	15	9	

encoder/decoder.

C. Two eMTR($j = 3, k = 14$) Codes of Length 68 and Rate-16/17

The k-weights of the first eMTR code of length $N = 68$ have a maximum span of $S^{(k)} = 15$ and are given by

$$v_n = \begin{cases} 0 & \text{for } n = 1, 2, \ldots, 8 \\ 1 & \text{for } n = 9, \ldots, 15 \\ 2^{n-16} & \text{for } n = 16, \ldots, 68. \end{cases}$$

The j-weights $\{w_n\}$ of the first code are specified in Table III. From the representation as integers of the form $w_n 2^{13-n}$, it is apparent that the j-weights have a maximum span of $S^{(j)} = 12$. By design, the code satisfies a $j = 3$ constraint and a $k = 14$ constraint within codewords and across codeword boundaries. As $S^{(j)} = 12$, error propagation is limited to at most 12 bits to the right at the modulation decoder output. The input space U is determined by the lower and upper bounds $L_N = 2 \times 2^{N-12}$ and $U_N = 258 \times 2^{N-12}$, where $N = 68$; thus, the code contains 2^{N-4} codewords. The capacity of the ($j = 3, k = 14$) constraint is 0.9467; thus, this rate-16/17 code has an efficiency of 99.41%.

A second eMTR($j = 3, k = 14$) code of length 68 and rate-16/17 is obtained by choosing a smaller j-span and using recursions (29) – (32) together with (23) – (24) and (27) – (28). In particular, the following parameters uniquely determine the second eMTR($j = 3, k = 14$) code: $S^{(j)} = 10$, $S^{(k)} = 16$, $k_b = 7$, $k_e = 7$,$j_b = 1$, $j_e = 2$ and $N = 68$. This code has an error propagation of at most $S^{(j)} = 10$ bits to the right, which is less than that of the first code.

D. Three eMTR($j = 4, k$) Codes of Length 66 and Rate-32/33

The construction of the three $j = 4$ eMTR codes is based on (23) – (28). The codes are uniquely determined by the following nine parameters, which are specified in Table IV: the two spans $S^{(j)}$ and $S^{(k)}$, the length N, the constraints j, k, and the boundary constraints j_b, j_e and k_b, k_e. All codes have rate-64/66. The first two codes satisfy a $k = 10$ constraint, and the third code satisfies a $k = 11$ constraint. Thus, the first two codes have an efficiency of 99.48% whereas the third code has an efficiency of 99.46% since the capacities of the ($j = 4, k = 10$) constraint and the ($j = 4, k = 11$) constraint are 0.9748 and 0.9750, respectively (see [11]). The last code has the shortest j-span $S^{(j)}$ and, therefore, it is the most efficient one in terms of complexity and has the least error propagation.

TABLE III

WEIGHTS OF AN EMTR($j = 3, k = 14$) CODE OF LENGTH 68 AND RATE-16/17.

n	1	2	3	4	5	6	7	8	9	10	11	12
$w_n 2^{13-n}$	4096	4096	4096	3584	3584	3456	3328	3200	3072	2960	2852	2748
n	13	14	15	16	17	18	19	20	21	22	23	24
$w_n 2^{13-n}$	2649	2553	2460	2371	2285	2202	2122	2045	1970	1899	1830	1763
n	25	26	27	28	29	30	31	32	33	34	35	36
$w_n 2^{13-n}$	1699	1637	1577	1520	1464	1411	1359	1310	1262	1216	1172	1129
n	37	38	39	40	41	42	43	44	45	46	47	48
$w_n 2^{13-n}$	1088	1048	1010	973	937	903	870	838	807	778	749	722
n	49	50	51	52	53	54	55	56	57	58	59	60
$w_n 2^{13-n}$	695	670	645	621	598	576	555	534	515	496	478	460
n	61	62	63	64	65	66	67	68				
$w_n 2^{13-n}$	443	427	411	396	381	367	353	340				

TABLE IV

PARAMETERS OF THREE EMTR($j = 4, k$) CODES OF LENGTH 66 AND RATE-32/33.

$S^{(j)}$	$S^{(k)}$	k	k_b	k_e	j	j_b	j_e	N	rate	capacity
10	13	10	5	5	4	2	2	66	0.9697	0.9748
11	11	10	5	5	4	2	2	66	0.9697	0.9748
9	13	11	5	6	4	2	2	66	0.9697	0.9750

E. Long eMTR Codes with $j = 7$

The construction of long $j = 7$ eMTR codes is based on (23) – (28). The codes are uniquely determined by the following parameters: the spans $S^{(j)}$, $S^{(k)}$, the length N, the constraints j, k, and the boundary constraints j_b, j_e and k_b, k_e. These parameters are specified in Table V for various codes with $k = 7$ to 10. For all these codes, the maximum length N is given such that the codes have rate-$(N-1)/N$. For comparison purposes, the capacity of the constraint is also given with an accuracy of four decimal digits (see [11]). The first and the second code satisfy the same constraints and both achieve essentially the same rate. The second code has a smaller j-span $S^{(j)}$ at the cost of a larger k-span $S^{(k)}$. The smaller j-span results in less error propagation at the decoder and also at a smaller encoding/decoding complexity despite the

TABLE V

PARAMETERS OF LONG EMTR CODES WITH $j = 7$ CONSTRAINT AND RATE-$(N-1)/N$.

$S^{(j)}$	$S^{(k)}$	k	k_b	k_e	j	j_b	j_e	N	rate	capacity
13	13	7	3	4	7	3	4	129	0.9922	0.9942
12	17	7	3	4	7	3	4	130	0.9923	0.9942
11	11	8	4	4	7	3	4	168	0.9940	0.9957
11	11	9	4	5	7	3	4	202	0.9950	0.9964
10	14	9	4	5	7	3	4	200	0.9950	0.9964
9	11	10	5	5	7	3	4	210	0.9952	0.9968

larger k-span value. Similar comments apply to the fourth, fifth and sixth codes.

F. Long eMTR Codes with $j = 6$

The construction of long $j = 6$ eMTR codes is based on (23) – (28). The codes are uniquely determined by the following parameters: the spans $S^{(j)}$, $S^{(k)}$, the length N, the constraints j, k, and the boundary constraints j_b, j_e and k_b, k_e. These parameters are specified in Table VI for various codes with $k = 7, 8$ and 9. The first two codes have rate-$(N-3)/N$, whereas the last four codes have rate-$(N-1)/N$. The dimension of the codes is denoted by K, i.e., each code has at least 2^K codewords. Again, for comparison purposes, the capacity of the constraint is given (with an accuracy of four decimal digits). The last code with $S^{(j)} = 9$ is interesting from a complexity and error-propagation point of view.

TABLE VI
PARAMETERS OF LONG eMTR CODES WITH $j = 6$ CONSTRAINT AND RATE-K/N.

$S^{(j)}$	$S^{(k)}$	k	k_b	k_e	j	j_b	j_e	N	K	rate	capacity
13	13	7	3	4	6	3	3	301	298	0.9900	0.9912
12	15	7	3	4	6	3	3	300	297	0.9900	0.9912
11	11	8	4	4	6	3	3	100	99	0.9900	0.9927
10	14	8	4	4	6	3	3	100	99	0.9900	0.9927
11	11	9	4	5	6	3	3	113	112	0.9912	0.9934
9	10	9	4	5	6	3	3	102	101	0.9902	0.9934

G. Long eMTR Codes with $j = 5$

The construction of the first six $j = 5$ eMTR codes in Table VII is based on (23) – (28). These codes are uniquely determined by the following parameters: the spans $S^{(j)}$, $S^{(k)}$, the length N, the constraints j, k, and the boundary constraints j_b, j_e and k_b, k_e. The maximum length N is given such that the first three codes have rate-$(N-3)/N$, the fourth code has rate-$(N-2)/N$, and the fifth and sixth codes have rate-$(N-1)/N$. The dimension of the codes is denoted by K, i.e., each code has at least 2^K codewords. Again, for comparison purposes, the capacity of the constraint is given (with an accuracy of four decimal digits). Note that the fourth code satisfies a $k = 10$ constraint within the codeword and a $j = 11$ constraint across codeword boundaries. Similarly, the sixth code satisfies $k = 18$ within codewords and $k = 19$ at codeword boundaries.

TABLE VII
PARAMETERS OF LONG eMTR CODES WITH $j = 5$ CONSTRAINT AND RATE-K/N.

$S^{(j)}$	$S^{(k)}$	k	k_b	k_e	j	j_b	j_e	N	K	rate	capacity
12	12	9	4	5	5	2	3	207	204	0.9855	0.9873
11	11	10	5	5	5	2	3	204	201	0.9853	0.9877
10	12	10	5	5	5	2	3	201	198	0.9851	0.9877
10	15	10	5	6	5	2	3	136	134	0.9853	0.9879
10	15	14	7	7	5	2	3	64	63	0.9844	0.9880
10	14	18	9	10	5	2	3	65	64	0.9846	0.9881
9	13	12	6	6	5	2	3	136	134	0.9853	0.9880

The last code in Table VII has a j-span of $S^{(j)} = 9$, which is the smallest among all codes in this table. To achieve this small j-span, it was necessary to base the construction of this code on the tighter recursions (29) – (32).

IV. High-Rate PRML(G, I, M) Codes

In partial-response maximum-likelihood (PRML) based recording systems, one uses PRML(G,I) modulation codes that satisfy (i) a global run-length constraint for synchronization purposes, and (ii) a run-length constraint in the even and the odd interleave allowing the use of a short path memory in the Viterbi detector without a substantial performance degradation [12]. We recall that PRML(G,I) codes are used with a $1/(1 \oplus D^2)$ precoder and that the (G,I) constraints are defined prior to the precoder. Namely, the maximum number of consecutive zeros in the coded binary sequences is limited to G, and the maximum number of consecutive zeros in both the even and odd interleaves of the coded sequences is limited to I.

By using a length-N eMTR(j, k) code in both the even and odd interleaves, one obtains a PRML(G, I) code of length $2N$, with $G = 2k$ and $I = k$ [7]. The j constraint in each interleave translates into the M constraint. This is a constraint that limits the runs of alternating $2T$ magnets $\ldots + + - - + + - - \ldots$ in the channel input sequences (i.e., after precoding) to $\lfloor M/2 \rfloor + 1 = j + 1$, where T denotes the symbol duration [13], [14]. In the context of anti-whistle codes, an equivalent constraint is referred to as k_4^a constraint [14]. We will denote PRML codes constructed from eMTR(j, k) codes, as PRML(G, I, M) codes to emphasize the three constraints $G = 2k$, $I = k$ and $M = 2j$. The M constraint is also known as VFO (variable-frequency oscillator) constraint. It is a desirable constraint in tape-recording systems [15], which employ a phase-locked loop (PLL) that acquires phase lock based on a long alternating $2T$ VFO pattern. The VFO constraint ensures that there is no modulation-encoded data sequence, which can be mistaken for a long VFO pattern.

In addition to the (G, I, M) constraints, the resulting PRML(G, I, M) codes satisfy further constraints, which are known as k_2 and k_4^b constraints [14]. The k_2 constraint limits the length of alternating channel input sequences $\ldots + - + - \ldots$, and the k_4^b constraint limits the length of channel input sequences of period 4 of the form $\ldots + + + - + + + - \ldots$ or their antipodal version. Thus, PRML(G, I, M) codes derived from eMTR(j, k) codes satisfy k, k_2, k_4^a and k_4^b constraints, i.e., these PRML(G, I, M) codes limit the length of all periodic channel input sequences of period 1, 2 and 4. The constraints are related by $k = G+1$, $k_2 = 2I+1$, $k_4^a = M+1$ and $k_4^b = 2I + 2$.

The two-way interleaving construction based on eMTR(j, k) codes results in PRML(G, I, M) codes of length $2N$. Typically, these PRML codes are chosen to have even dimension, say $2(N-1)$, and the $2(N-1)$ input bits are split into an even and an odd bit stream of $N-1$ bits each and then encoded by the eMTR encoders in both interleaves. By using a prefix encoder, one can also allow an odd number of input bits for the PRML(G, I, M) code [16]. In this case, prior to the eMTR encoding, the input bit stream is partitioned into two bit streams by the prefix encoder and each of the two bit streams is encoded separately by the two eMTR encoders in the even and the odd interleave. The corresponding eMTR codes must have the appropriate number of codewords. For example, if the $2N-1$-bit input stream is partitioned into two equal parts, each eMTR code must have at least $2^{(2N-1)/2}$ codewords. Examples of such constructions will be given below.

A. Long High-Rate PRML($G = 14, I = 7, M$) Codes

To construct PRML($G = 14, I = 7, M$) codes from the eMTR codes with $j = 7$, as given in Table V, one applies bit inversion to the eMTR codes, which transforms an eMTR($j = 7, k = \kappa$) into an eMTR($j = \kappa, k = 7$) code with dual constraints. Because bit inversion is applied, the M constraint of the PRML($G = 14, I = 7$) codes is $M = 2k$, where k refers to the k constraint of the underlying eMTR code in Table V. In this way, the six codes in Table V give raise to six PRML($G = 14, I = 7, M$) codes of rate $(2N - 2)/2N$ with M constraints $14, 14, 16, 18, 18,$ and 20, respectively.

By using a prefix encoder [16], one can construct a rate-207/208 PRML($G = 14, I = 7, M = 22$), which in the even and the odd interleave uses the bit-inverted eMTR code specified in Table VIII. The construction of this eMTR code is based on (23) – (28). Note that this eMTR code has a fractional dimension $K = 103.5$, which means that it contains at least $2^{103.5}$ codewords.

TABLE VIII

PARAMETERS OF AN eMTR($j = 7, k = 11$) CODE OF LENGTH 104 WITH AT LEAST $2^{103.5}$ CODEWORDS.

$S^{(j)}$	$S^{(k)}$	k	k_b	k_e	j	j_b	j_e	N	K	rate	capacity
9	12	11	5	6	7	3	4	104	103.5	0.9952	0.9970

B. Long High-Rate PRML($G = 12, I = 6, M$) Codes

In a similar way as in the preceding subsection, one can construct PRML($G = 12, I = 6$) codes from the eMTR codes with $j = 6$ in Table VI. The resulting six codes are PRML($G = 12, I = 6, M$) codes with $M = 14, 14, 16, 16, 18$ and 18 and rates 596/602, 594/600, 198/200, 198/200, 224/226 and 202/204, respectively.

The following five PRML($G = 12, I = 6, M$) codes are derived from the interleaving construction in conjunction with a prefix encoder [16]. The even and odd interleaves of these PRML codes are eMTR codes, which are determined by the parameters in Table IX and Equations (23) – (28). The resulting PRML codes have length $2N$ and dimension $2K$.

TABLE IX

PARAMETERS OF LONG eMTR CODES WITH $j = 6$ CONSTRAINT AND AT LEAST 2^K CODEWORDS.

$S^{(j)}$	$S^{(k)}$	k	k_b	k_e	j	j_b	j_e	N	K	rate	capacity
11	15	10	5	5	6	3	3	50	49.5	0.9900	0.9938
12	12	10	5	5	6	3	3	50	49.5	0.9900	0.9938
12	12	10	5	5	6	3	3	204	202.5	0.9926	0.9938
11	16	11	6	7	6	3	3	56	55.5	0.9911	0.9942
9	11	12	6	6	6	3	3	50	49.5	0.9900	0.9941

C. Long High-Rate PRML($G = 10, I = 5, M$) Codes

Similarly, one can also construct PRML($G = 10, I = 5, M$) codes from the eMTR codes with $j = 5$ in Table VII. The resulting seven PRML($G = 10, I = 5, M$) codes have rates 408/414, 402/408, 396/402, 268/272, 126/128, 128/130 and 268/272, and M-constraints $18, 20, 20, 22, 28,$ 38 and 24 respectively.

Two further PRML($G = 10, I = 5, M$) codes of length $2N = 200$ and dimension $2N-3 = 197$ are obtained from the interleaving construction in conjunction with a prefix encoder [16]. The underlying eMTR($j = 5, k$) codes of length 100 are uniquely determined by the eight leftmost parameters in Table X. The codes contain more than $2^{98.5}$ codewords. The first code satisfies a uniform $k = 11$ constraint within codewords and a $k = 12$ constraint at the codeword boundary, whereas the second code satisfies an overall $k = 11$ constraint.

TABLE X

Parameters of two long eMTR codes with $j = 5$ constraint and at least 2^K codewords.

$S^{(j)}$	$S^{(k)}$	k	k_b	k_e	j	j_b	j_e	N	K	rate	capacity
11	11	11	6	6	5	2	3	100	98.5	0.9850	0.9880
10	15	11	5	6	5	2	3	100	98.5	0.9850	0.9879

V. Encoding of Fibonacci and eMTR Codes

The class of generalized Fibonacci codes can be considered to be a subclass of the class of eMTR codes: it consists of those eMTR codes that have a $k = \infty$ constraint, i.e., with no k constraint. More specifically, if one selects $k = N$ in the generalized recursions (10), (11), (12) and (13), then the k-weights $\{v_n\}$ are all zero and the resulting length-N eMTR($j, k = N$) code is a generalized Fibonacci code. In particular, the code is fully determined by the j-weights $\{w_n\}$. In this case, the eMTR encoder (7) and the eMTR decoder (9) are Fibonacci encoders and decoders [3].

For a nontrivial k constraint, the eMTR codes are characterized by two set of weights, the k-weights $\{v_n\}$ and j-weights $\{w_n\}$, and the eMTR encoder/decoder algorithms are slightly different from the Fibonacci encoders/decoders. The Fibonacci encoders/decoders only have to store the j-weights, whereas the eMTR encoders/decoders need to store the j-weights and the combined weights $\{v_n + w_n\}$, $n = 1, 2, \ldots, N$. Thus, the memory requirement for eMTR codes is about twice that for generalized Fibonacci codes. When working with the j-weights and the combined weights, the operational complexity of the eMTR encoders/decoders is similar to that of Fibonacci encoders/decoders. Both encoders and decoders can be implemented by sliding-window algorithms of window length $\max\{S^{(j)}, S^{(k)}\}$ [7]. In particular, for the typical case where $S^{(j)} \leq S^{(k)}$, the encoding (7) of a codeword bit x_ℓ at the ℓ-th step requires an $S^{(k)}$-bit-wide comparison and one subtraction of two $S^{(j)}$-bit-wide numbers. At the decoder (9), each processed bit needs one $S^{(j)}$-bit-wide addition with carry.

There is an important structural difference between Fibonacci and eMTR encoders, namely, the eMTR encoders need a preliminary offset mapper to account for the fact that the input space is in the range $L_N \leq u < U_N$, where the lower bound is strictly positive for a nontrivial k constraint, i.e., for $k < N$. The complexity of this offset mapper amounts to one $S^{(j)}$-bit-wide addition with carry, which is a very minor increase in the operational complexity of the overall eMTR encoder. Similarly, at the decoder side, a corresponding offset mapper is needed, which subtracts the lower bound L_N from the bit stream generated by the decoding algorithm (9).

VI. Conclusions

A novel class of MTR block codes has been presented that is characterized by two sets of weights, the j-weights and the k-weights. These weights are determined by two recursion formulae, which reflect the predetermined j and k constraints. These eMTR codes have efficient enumerative encoding/decoding algorithms with reduced error propagation at the decoder. These encoders/decoders are very similar to the ones of generalized Fibonacci codes, except for an additional simple offset mapper, which accounts for the fact that the enumeration does not start at zero but at some positive offset. Numerous examples of long high-rate eMTR codes illustrate the efficiency of these codes.

Two-way interleaving of bit-inverted eMTR(j, k) codes gives raise to PRML$(G = 2j, I = j, M = 2k)$ codes. Such long high-rate PRML(G, I, M) codes are suitable for reverse concatenation architectures with partial symbol interleaving in tape/optical recording and recording systems for hard-disk drives.

Although the code examples presented are essentially limited to codes with uniform modulation constraints, one can easily construct codes with nonuniform constraints as illustrated in [7] for the class of generalized Fibonacci codes. For codes with uniform and nonuniform constraints, the same general code design method applies. Another way to generalize the code construction and to further reduce error propagation is to consider nonuniform j-span parameters as in the construction of variable span Fibonacci codes [17].

References

[1] K. A. S. Immink, P. H. Siegel, and J. K. Wolf,“Codes for digital recorders,” *IEEE Trans. Inform. Theory*, vol. 44, pp. 2260–2299, Oct. 1998.

[2] J. Moon and B. Brickner,“Maximum transition run codes for data storage systems,” *IEEE Trans. Magn.*, vol. 32, pp. 3992–3994, Sept. 1996.

[3] W. H. Kautz, “Fibonacci codes for synchronization control,” *IEEE Trans. Inform. Theory*, vol. 11, pp. 284-292, Apr. 1965.

[4] W. G. Bliss, “Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation,” *IBM Tech. Discl. Bull.*, vol. 23, pp. 4633–4634, Mar. 1981.

[5] J. L. Fan and A. R. Calderbank, “A modified concatenated coding scheme, with applications to magnetic data storage,” *IEEE Trans. Inform. Theory*, vol. 44, pp. 1565–1574, July 1998.

[6] A. J. van Wijngaarden and K. A. S. Immink, “Maximum runlength-limited codes with error control capabilities,” *IEEE J. Select. Areas Commun.*, vol. 19, pp. 602–611, Apr. 2001.

[7] M. Blaum, R. D. Cideciyan, E. Eleftheriou, K. Lakovic, T. Mittelholzer, and B. Wilson, “High-rate modulation codes for reverse concatenation,” *IEEE Trans. on Magnetics*, vol. 43, issue 2, part 2, pp. 740–743, Feb. 2007

[8] T. M. Cover, “Enumerative source encoding,” *IEEE Trans. Inform. Theory*, vol. 19, pp. 73–77, Jan. 1973.

[9] K. A. S. Immink, “A practical method for approaching the channel capacity of constrained channels,” *IEEE Trans. Inform. Theory*, vol. 43, pp. 1389–1399, Sept. 1997.

[10] R. D. Cideciyan, E. Eleftheriou, B. H. Marcus, and D. S. Modha, “Maximum transition run codes for generalized partial response channels,” *IEEE J. Select. Areas Commun.*, vol. 19, pp. 619–634, Apr. 2001.

[11] B. J. Brickner, *Maximum transition run coding and pragmatic signal space detection for digital magnetic recording*, Ph.D. Thesis, Univ. of Minnessota, 1998.

[12] P. H. Siegel and J. K. Wolf, “Modulation and coding for information storage,” *IEEE Comm. Mag.*, vol. 29, pp. 68–86, Dec. 1991.

[13] R. D. Cideciyan, F. Dolivo, R. Hermann, W. Hirt, and W. Schott, ”A PRML system for digital magnetic recording,” *IEEE J. Select. Areas Commun.*, vol. 10, pp. 38–56, Jan. 1992.

[14] J. W. M. Bergmans, *Digital Baseband Transmission and Recording*, Kluver Academic Publishers, Dordrecht, 1996.

[15] G. A. Jaquette, “LTO: A better format for mid-range tape,” *IBM J. Res. Dev.*, vol. 47, no. 4, pp. 429–444, 2003.

[16] M. Blaum, R. Cideciyan, E. Eleftheriou, K. Lakovic, T. Mittelholzer, and B. Wilson,“Enumerative encoding with non-uniform modulation constraints,” *Proc. IEEE Intl Symp. on Information Theory*, Nice, France, June 2007, pp. 1831–1835.

[17] M. Blaum and K. Lakovic, “Variable span Fibonacci codes with limited byte error propagation,” *Proc. IEEE Intl Symp. on Information Theory*, Nice, France, June 2007, pp. 2541–2544.