

Maximum Transition Run Codes for Generalized Partial Response Channels

Roy D. Cideciyan, *Senior Member, IEEE*, Evangelos Eleftheriou, *Senior Member, IEEE*,
Brian H. Marcus, *Fellow, IEEE*, and Dharmendra S. Modha, *Member, IEEE*

Abstract—A new twins constraint for maximum transition run (MTR) codes is introduced to eliminate quasi-catastrophic error propagation in sequence detectors for generalized partial response channels with spectral nulls both at dc and at the Nyquist frequency. Two variants of the twins constraint that depend on whether the generalized partial response detector trellis is unconstrained or j -constrained are studied. Deterministic finite-state transition diagrams that present the twins constraint are specified, and the capacity of the new class of MTR constraints is computed. The connection between (G, I) constraints and $MTR(j)$ constraints is clarified. Code design methodologies that are based on look-ahead coding in combination with violation detection/substitution as well as on state splitting are used to obtain several specific constructions of high-rate MTR codes.

Index Terms—Constrained coding, magnetic recording, maximum transition run codes, modulation coding, quasicatastrophic error propagation.

I. INTRODUCTION

THE THEORETICAL foundation of coding for input-constrained channels, also known as modulation coding, was established in Shannon's classic study of discrete noiseless channels [1]. Among the various methods for constructing efficient modulation codes that have been developed in the past 50 years, the state-splitting algorithm of Adler *et al.* [2] provides a systematic and mathematically rigorous approach to designing finite-state encoders and sliding-block decoders for finite-type constrained systems. In practice, however, all approaches to code construction including state splitting and others such as look-ahead coding require that the right choices be made during the code construction procedure in order to obtain good codes. For an extensive discussion of the large body of work on modulation coding, the reader is referred to [3].

Peak detection systems employing runlength-limited (RLL) (d, k) constrained codes are predominant in digital magnetic storage at low densities. RLL (d, k) codes reduce the effects of pulse interference and prevent the loss of clock synchronization. Rate-2/3 RLL(1,7) and rate-1/2 RLL(2,7) codes have been widely used in the digital recording industry [3]. At mod-

erate storage densities, the introduction of partial response maximum-likelihood sequence detection (PRML) (see [4] and references therein) requires a different class of constrained codes. This class of codes, collectively known as PRML (G, I) codes, facilitates timing recovery and gain control, and limits the path memory length of the sequence detector, and therefore the decoding delay, without significant degradation of detector performance. The most widely used code rates in the industry have been 8/9 [5] and 16/17 [6]. In general, high code rates and small error bursts at the modulation decoder output are desirable in order to minimize performance degradation due to rate loss and error propagation at the modulation decoder.

More recently, maximum transition run (MTR) (j, k) codes have been introduced by Moon and Brickner to provide coding gain for extended partial response channels [7]. The maximum possible code rate for the original MTR $(j = 2, k)$ constraints was less than 8/9, leading to an unacceptable loss of performance due to the low code rate. Time-varying MTR $j = 2, 3$ constraints permit the design of higher rate MTR codes at the expense of incorporating the j -constraint into the 8- or 16-state detector, i.e., by implementing a time-varying trellis in the sequence detector. The rates of the codes that have been designed to satisfy time-varying MTR constraints are 8/9 [8] and higher [9], [10]. These various classes of MTR codes increase the minimum Euclidean distance in the sequence detector by eliminating the dominant error events and are therefore known as distance-enhancing codes. In general, the performance-enhancing features of the class of high-rate MTR codes render them more attractive than conventional (G, I) codes. Ultimately, the coding gain of an MTR code is determined by the tradeoff among various factors such as rate loss, error propagation at the MTR decoder, and performance-improving properties of the MTR code.

In this paper, we focus on MTR codes that also have constraints to aid timing recovery and gain control, as well as to limit quasi-catastrophic error propagation in the sequence detector. In particular, we develop a framework for analyzing quasi-catastrophic error propagation in the context of MTR codes. For this purpose, we introduce an additional constraint that applies to all classes of MTR codes. We refer to this new class of constraints as the twins constraint. The definition of the twins constraint presupposes knowledge of the detector trellis. Two cases of sequence detection are considered: first, detection on an unconstrained trellis similar to the detection configuration in the case of (G, I) codes, and second, detection on a j -constrained trellis that may be time-varying. In both cases, the exhaustive presentation of the twins constraint

Manuscript received April 17, 2000; revised July 19, 2000.

R. D. Cideciyan and E. Eleftheriou are with IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland (e-mail: cid@zurich.ibm.com; ele@zurich.ibm.com).

B. H. Marcus and D. S. Modha are with IBM Research, Almaden Research Center, San Jose, CA 95120 USA (e-mail: marcus@almaden.ibm.com; modha@almaden.ibm.com).

Publisher Item Identifier S 0733-8716(01)01748-6.

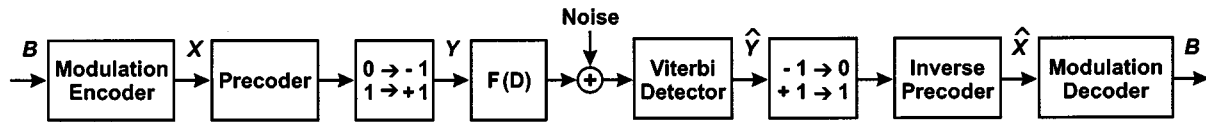


Fig. 1. Recording system model.

via deterministic finite-state transition diagrams allows the computation of maximum possible code rates for the new class of MTR constraints that includes the twins constraint.

Conventional (G, I) codes and MTR codes have been discussed separately in the literature, but no connection has been made between them. We show that (G, I) codes are intimately related to MTR codes. Specifically, we demonstrate that precoded (G, I) constraints are a subclass of precoded MTR constraints. This relationship allows an alternative code construction methodology for (G, I) codes that is based on employing a different precoder, namely, the precoder used by MTR codes.

Finally, we present constructions of several high-rate efficient MTR codes having a variety of special properties. Two code design methodologies are employed. The first one, look-ahead coding in combination with violation detection/substitution, is a method that gives a great deal of freedom to the code designer and can lead to good codes if the code designer makes judicious choices. The second one involves the state-splitting algorithm and is based on the mathematical discipline of symbolic dynamics. For an introduction into symbolic dynamics, the reader is referred to the textbook [11].

The paper is organized as follows. In Section II, we describe the recording system model, emphasizing modulation coding, generalized partial response shaping, and detection. In Section III, we characterize the sequences that must be eliminated in order to limit quasi-catastrophic error propagation when an MTR code is used on a generalized partial response channel. In Section IV, we make some general remarks about computing capacity for constraints that simultaneously satisfy MTR constraints, facilitate timing recovery, and limit quasi-catastrophic error propagation. We also present tables of the capacities of such constraints. In Section V, we establish a connection between the standard PRML (G, I) constraints and MTR constraints; in particular, we show how any (G, I) constraint can be regarded as a special kind of MTR constraint. In Section VI, we provide several constructions of specific MTR codes using block coding and look-ahead coding techniques. In Section VII, we give constructions of specific MTR codes using the state-splitting technique. Finally, Section VIII contains a summary of the main results and conclusions.

II. CONSTRAINTS FOR RECORDING CHANNELS

A. Recording System Model

The communications model of the recording system considered in this paper is shown in Fig. 1. This model can also be regarded as the recording chain seen by an outer Reed–Solomon (RS) encoder/decoder. The input sequence to the modulation encoder is a binary sequence denoted $\{b_i\}$, where $b_i \in \{0, 1\}$. Although the inputs to the recording system

are usually RS-encoded, we follow convention and assume that there are no constraints on these inputs; so the allowable input sequences are drawn from the set B of all possible binary sequences. The modulation encoder generates binary sequences $\{x_i\} \in X$ that satisfy a desired constraint, such as PRML (G, I) constraints [4]–[6], [12] or MTR constraints [7]–[9], [13]–[16]. The modulation encoder is followed by a precoder of the form $1/(1 \oplus D)$ (in the MTR case) or $1/(1 \oplus D^2)$ (in the (G, I) case). The output of the precoder is then mapped into two-level channel input sequences $\{y_i\} \in Y$, where $y_i \in \{-1, +1\}$ and Y denotes the set of all possible channel input sequences. In the following the set of all bipolar sequences over the alphabet, $\{+1, -1\}$ is denoted \bar{B} .

The generalized partial response channel characterized by the polynomial $F(D) = 1 + \sum_{i=1}^L f_i D^i$, where L is the channel memory and $f_i, i = 1, 2, \dots, L$, are real coefficients, models a chain of signal-processing functions in disk drives, including write precompensation, write driver, magnetic read/write process, preamplifier, automatic gain control, low-pass filtering, sampling, equalization, and noise whitening. In particular, we will consider the important class of channel polynomials of the form $F(D) = (1 - D)^n(1 + D)^m(1 - P(D))$, where $n, m \geq 0$, $P(D)$ is an approximation of a predictor filter and $1 - P(D)$ is assumed to have no roots on the unit circle [17]. In this case, the output of the partial response channel is $y(D)F(D)$, and the noise at the input to the detector can be approximated by an additive white Gaussian noise (AWGN) source, provided that the equalizer and the predictor filter are sufficiently long [17]. This class of generalized partial response channel polynomials is significant in practice and, when combined with sequence detection, gives rise to noise-predictive maximum-likelihood (NPML) systems [18], [19].

The Viterbi detector in Fig. 1 performs maximum-likelihood sequence detection on the generalized partial response trellis and provides an estimate of the channel input sequence $\{\hat{y}_i\} \in \hat{Y}$, where \hat{Y} is the set of possible output sequences of the Viterbi detector; \hat{Y} contains Y , and typically, Y is a proper subset of \hat{Y} . The detector is followed by the inverse precoder of the form $(1 \oplus D)$ (in the MTR case) or $(1 \oplus D^2)$ (in the (G, I) case). Finally, the modulation decoder delivers an estimate of the recorded binary sequence $\{\hat{b}_i\} \in B$.

Two types of detector trellises will be considered: unconstrained trellis, which is the standard 2^L -state generalized partial response trellis (here, $\hat{Y} = \bar{B}$) and constrained trellis, where some edges or states are deleted from the standard trellis and the trellis may be time varying (here, $\hat{Y} \subsetneq \bar{B}$ incorporates certain constraints of the modulation code). As the precoder is invertible, the set of all possible sequences \hat{X} at the output of the inverse precoder will satisfy $\hat{X} = B$ for unconstrained trellises and $\hat{X} \subsetneq B$ for constrained trellises. In particular, \hat{X} may differ from \bar{X} .

The modulation code and the detector trellis are often designed jointly in order to improve performance. For example, the MTR constraint, denoted $M(j)$ or $MTR(j)$, limits the number of consecutive ones to j . Equivalently, at the partial response channel input, the maximum runlength of alternating $+1$ and -1 symbols is $j + 1$; we denote this associated constraint by $M_c(j)$. If the code constraint is incorporated into the detector, then the Viterbi detection process can yield coding gain (at least for some partial response targets [7]). As an added benefit, there can be a reduction in complexity of the Viterbi detector: it can be shown that for $L = 4$ and $L = 5$, the $(j = 2)$ -constrained generalized partial response trellis has 14 detector states instead of 16 [7], [13] and 26 detector states instead of 32, respectively. In both cases, the set of sequences at the output of the Viterbi detector will be the bipolar sequences $\hat{Y} = M_c(j = 2)$, and the set of sequences at the output of the inverse precoder will be $\hat{X} = M(j = 2)$.

B. Undesired Sequences

The issue of undesired sequences at the input to a partial response recording channel has received a great deal of attention in the past, especially in connection with PRML detection. For example, a string of consecutive ones at the input to the precoder requires the write head to switch so fast that it may not be able to saturate the medium; thus, an MTR constraint is useful for eliminating undesired sequences (in addition to the benefits of enhanced coding gain and decreased complexity, mentioned in the previous section). As another example, the presence of a long string of zeros at the partial response channel output can degrade the tracking performance of the timing and gain control loops [4]. In addition, a long string of zeros in the subsequence of even bit positions (or the subsequence of odd bit positions) can require a long path memory for the sequence detector in order to avoid significant performance degradation. Lucid treatments of these issues in connection with partial response channels of the form $F(D) = (1 - D)^m(1 + D)^n$, $n, m \geq 0$ can be found in [3], [15], and [20]. It can readily be seen that the same results hold for generalized partial response channels of the form $F(D) = (1 - D)^m(1 + D)^n(1 - P(D))$, $n, m \geq 0$ (again, $1 - P(D)$ has no roots on the unit circle).

In order to facilitate timing and gain control algorithms, the only channel input sequences that need to be eliminated are those that have spectral energy at the frequencies where the channel has spectral nulls. In particular, for channels of the form $(1 - D)^m(1 - P(D))$, $m \geq 1$, channel input sequences $(+1)$ and (-1) with a spectral null at dc should be eliminated, where (s) denotes the sequence that is obtained by periodically repeating the string s , e.g., $(ab) = ababab \dots$. The well-known k -constraint limits the maximum length of zeros at the input of the $1/(1 \oplus D)$ precoder to k or equivalently limits the length of channel input patterns of type $(+1)$, (-1) to $k + 1$ (note that an $MTR(j)$ constraint becomes a k -constraint under binary complementation). We use the notation $M(j, k)$ to denote the constraint that is simultaneously j and k constrained (and likewise $M_c(j, k)$ to denote the associated constraint at the partial response channel input). Typically, j needs to be small (on the order of 1, 2, 3, or 4) in order to have any significant effect, but k can be much greater (on the order of 10 to 15 or even larger).

Similarly, it can readily be seen that for an arbitrary channel polynomial $F(D)$ that has spectral nulls at dc and the Nyquist frequency, the channel input sequences $(+1 - 1)$ and $(-1 + 1)$ with a spectral line at the Nyquist frequency should also be eliminated. For example, for channel polynomials of the form $(1 - D^2)(1 - P(D))$, the channel input sequences $(+1)$, (-1) with a spectral line at dc together with the sequences $(+1 - 1)$, $(-1 + 1)$ with a spectral line at the Nyquist frequency should be eliminated. The well-known G -constraint (which is simply a k -constraint in this context) limits the maximum length of zeros at the input of the $1/(1 \oplus D^2)$ precoder to G or equivalently limits the maximum length of channel input patterns of all four types $(+1)$, (-1) , $(+1 - 1)$, and $(-1 + 1)$ to $G + 2$.

Another desirable code property is the elimination of quasi-catastrophic error propagation that is inherent in maximum-likelihood sequence detection for partial response channels with spectral nulls [21]. This property allows a reduction of the path memory size of the sequence detector without degrading its bit error rate performance. Quasi-catastrophic error propagation is avoided by eliminating channel-input error sequences $\{e_i\} = \{y_i - \hat{y}_i\}$ that have spectral energy only at those frequencies where the channel has spectral nulls. For channels of the form $(1 - D)^m(1 - P(D))$, $m \geq 1$, the k -constraint at the input of a $1/(1 \oplus D)$ precoder is sufficient to eliminate quasi-catastrophic error propagation, because it limits the maximum length of channel-input error patterns of type $(+2)$, (-2) to $k + 1$. For channels of the form $(1 - D^2)(1 - P(D))$, it can readily be seen that it is necessary and sufficient to limit the maximum length of channel-input error patterns of the type $(+2)$, (-2) , $(+2 - 2)$, $(-2 + 2)$, $(+2 0)$, $(0 + 2)$, $(-2 0)$, $(0 - 2)$. In general, these channel-input patterns exhaustively characterize the undesired quasi-catastrophic sequences for arbitrary channel polynomials of the form $F(D) = (1 - D)^m(1 + D)^n(1 - P(D))$, where $n, m \geq 1$ (see also [15]). The well-known I -constraint at the input of a $1/(1 \oplus D^2)$ precoder limits the maximum length of channel-input error patterns of type $(+2)$, (-2) , $(+2 - 2)$, $(-2 + 2)$ to $2I + 2$ and of type $(+2 0)$, $(0 + 2)$, $(-2 0)$, $(0 - 2)$ to $2I + 3$. Note that an additional G -constraint, $G \leq 2I$, further reduces the maximum length of error patterns of type $(+2)$, (-2) , $(+2 - 2)$, $(-2 + 2)$.

III. QUASI-CATASTROPHIC ERROR PROPAGATION FOR MTR CONSTRAINTS

In general, $MTR(j, k)$ codes do not avoid quasi-catastrophic error propagation in sequence detectors for partial response channels with spectral nulls both at dc and the Nyquist frequency. The k -constraint avoids channel-input error sequences that have spectral energy only at dc, whereas the j -constraint avoids channel-input error sequences that have spectral energy only at the Nyquist frequency. Therefore, an additional constraint is needed to limit the maximum length of channel-input error sequences of type $(+2 0)$, $(0 + 2)$, $(-2 0)$, $(0 - 2)$ that have spectral energy both at dc and at the Nyquist frequency. In the remainder of the paper, we will concentrate on channel polynomials of the form

$F(D) = (1 - D)^m(1 + D)^n(1 - P(D))$, $m, n \geq 1$, which contain spectral nulls at these two frequencies.

A parameter that is closely related to the truncation depth of the trellis is the maximum number of branches associated with two distinct trellis paths that have the same sequence of output labels and, therefore, accumulate zero distance. We define the maximum run of accumulated zero-distance as

$$r \triangleq \max\{n: \varepsilon_1 = \varepsilon_2 = \dots = \varepsilon_{n-1} = \varepsilon_n = 0\} \quad (1)$$

where $\{\varepsilon_i\}$ is the channel-output error sequence with D -transform $\varepsilon(D) = (y(D) - \hat{y}(D))F(D)$ [16]. We emphasize that $\{y_i\}$ should be a valid channel input sequence, whereas $\{\hat{y}_i\}$ can be a sequence that corresponds to any trellis path. Clearly, the sequence detector suffers from quasi-catastrophic error propagation if $r = \infty$. In practical systems, r is usually a small number. For example, it can readily be verified that for (G, I) -constrained NPML systems, $r = 2I + 3 - L$. A value of r on the order of 20 to 30 can be effective for limiting quasi-catastrophic error propagation with modest additional complexity.

The j and k constraints of an MTR code ensure that the length of the channel-input error patterns $\{e_i\}$ of type $(+2)$, (-2) , $(+2 - 2)$, $(-2 + 2)$ is at most $\max(j + 1, k + 1)$; thus, a finite maximum length of channel-input error patterns of type $(+2 0)$, $(0 + 2)$, $(-2 0)$, $(0 - 2)$ is necessary and sufficient to guarantee a finite r and to render the code nonquasi-catastrophic.

Definition: The output sequence of an encoder $\{x_i\}$ satisfies a twins constraint, also referred to as a t -constraint, if it does not allow $t + 1$ consecutive pairs of zeros or ones ("twins") that are the binary complement of an allowable string $\hat{x}_i, \hat{x}_{i+1}, \dots, \hat{x}_{i+2t+1}$ at the output of the inverse precoder $(1 \oplus D)$.

For example, if the string 001100111100 is allowable at the inverse precoder output, i.e., its precoded version is an allowable string on the trellis, and $t = 5$, then the complement 110011000011 is forbidden at the encoder output. This additional t -constraint on the MTR encoded sequences ensures a finite maximum run of accumulated zero-distance, which implies no quasi-catastrophic error propagation. The reader may verify that a twins constraint, together with a j and k constraint, leads to a maximum run of accumulated zero-distance $r = \max(j + 1, k + 1, 2t + 3) - L$. The twins constraint can be characterized by a finite set of forbidden strings and is therefore a system of finite type [11] (as are other widely used constraints in magnetic recording, such as RLL(d, k), PRML(G, I), and MTR(j, k) constraints); such systems tend to be more amenable to good code construction.

Note: The definition of a twins constraint requires a specification of a detector trellis; in particular, the twins constraint, by itself, does not in general specify a constrained system of sequences.

A. Twins Constraint for Unconstrained Trellis

If sequence detection is performed on an unconstrained generalized partial response trellis, the set of valid sequences at

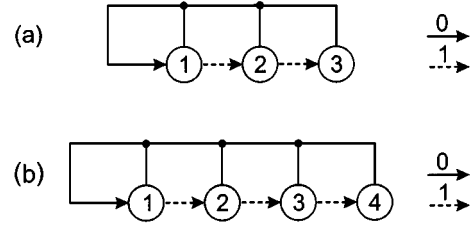


Fig. 2. FSTDs presenting constrained systems: (a) MTR($j = 2$) and (b) MTR($j = 3$).

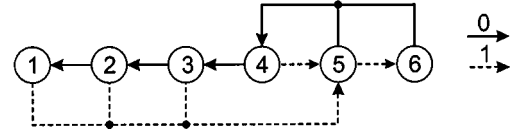


Fig. 3. FSTD presenting constrained system MTR($j = 2, k = 4$).

the inverse precoder output is the set of all binary sequences, i.e., $\hat{X} = B$. In this case, the t -constraint must forbid all possible $t + 1$ consecutive pairs of zeros or ones at the MTR encoder output; we use the notation $M(j, k, t)$ to denote this twins constraint that is also simultaneously j and k constrained. For example, the string 1100110011 would be allowed if $t = 5$, whereas it would be forbidden, if $t = 4$. Therefore, the t -constraint for unconstrained detector trellises implies an upper bound of $2t + 1$ on the j and k constraints, i.e., $j \leq 2t + 1$ and $k \leq 2t + 1$. Consistent with our earlier notation, we use $M_c(j, k, t)$ to denote the corresponding constraint at the partial response channel input.

A finite-state transition diagram (FSTD) is a useful way to represent a constraint. Such a diagram consists of states (or vertices) and labeled, directed transitions between states such that the allowable constrained sequences are precisely the sequences obtained by traversing paths of the diagram. The FSTD is called deterministic if at each state, all outgoing transitions have distinct labels. For instance, the MTR($j = 2$) and MTR($j = 3$) constraints are presented by deterministic FSTDs in Fig. 2, and the MTR($j = 2, k = 4$) constraint is presented by a deterministic FSTD in Fig. 3. For the $M(j, k, t)$ constraint, a deterministic FSTD is described as follows.

We label the states of this FSTD with pairs $(\alpha - \beta, \gamma)$, where α is the runlength of ones ending at the current state, β is the runlength of zeros ending at the current state, and γ is the runlength of binary sequences of type $\dots a_5 a_5 a_4 a_4 a_3 a_3 a_2 a_2 a_1 a_1$ or $\dots a_5 a_5 a_4 a_4 a_3 a_3 a_2 a_2 a_1$ ending at the current state [22]. For example, $\dots 10001111$ leads to the state $(4, 6)$, whereas $\dots 100011110$ leads to the state $(-1, 7)$, indicating that there is a transition from the state $(4, 6)$ to the state $(-1, 7)$ labeled 0. The set of vertices V associated with this FSTD lies on an integer lattice and can be described by the union of two sets

$$\begin{aligned} V = \{ & (\alpha, \gamma): 1 \leq \alpha \leq j, \quad 1 \leq \gamma \leq 2t + 1 \\ & \alpha \leq \gamma, \quad \alpha = \gamma \bmod 2 \} \\ & \cup \{ (-\beta, \gamma): 1 \leq \beta \leq k, \quad 1 \leq \gamma \leq 2t + 1 \\ & \beta \leq \gamma, \quad \beta = \gamma \bmod 2 \}. \end{aligned} \quad (2)$$

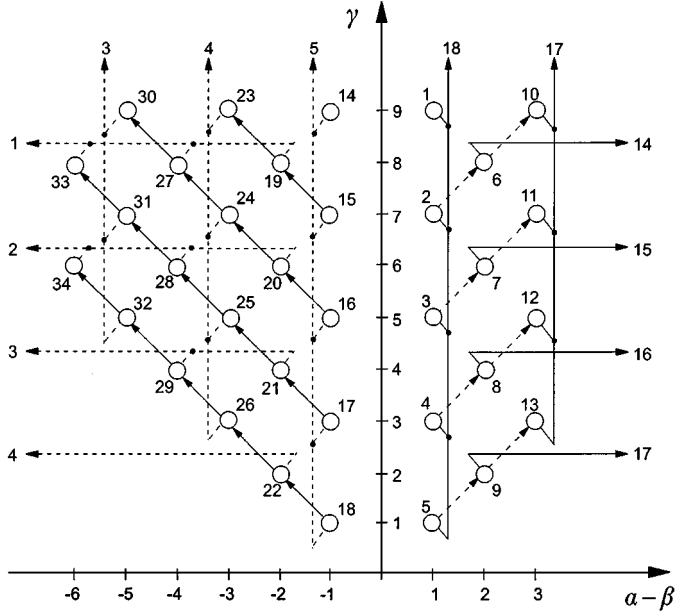


Fig. 4. FSTD presenting constrained system $M(3, 6, 4)$ (an arrow represents a set of transitions, and the number next to an arrow indicates the terminal state associated with that set of transitions).

For $j \geq 2$, the total number of states N is then given by

$$N = (j+k)t + 2 - \frac{1}{2} \left\lfloor \frac{j-3}{2} \right\rfloor \left(\left\lfloor \frac{j-3}{2} \right\rfloor + 1 \right) - \frac{1}{2} \left\lfloor \frac{j-3}{2} \right\rfloor \left(\left\lfloor \frac{j-3}{2} \right\rfloor + 1 \right) - \frac{1}{2} \left\lfloor \frac{k-3}{2} \right\rfloor \left(\left\lfloor \frac{k-3}{2} \right\rfloor + 1 \right) - \frac{1}{2} \left\lfloor \frac{k-3}{2} \right\rfloor \left(\left\lfloor \frac{k-3}{2} \right\rfloor + 1 \right), \quad (3)$$

and the state transitions are given by the rules

$$\begin{aligned} \text{"1"}: (\alpha, \gamma) &\rightarrow (\alpha + 1, \gamma + 1), & \text{if } \alpha \geq 1, & (\alpha + 1, \gamma + 1) \in V \\ \text{"0"}: (\alpha, \gamma) &\rightarrow (-1, \gamma + 1), & \text{if } \alpha \geq 1, \gamma = 0 \bmod 2 & \\ \text{"0"}: (\alpha, \gamma) &\rightarrow (-1, \alpha), & \text{if } \alpha \geq 1, \gamma = 1 \bmod 2 & \\ \text{"0"}: (-\beta, \gamma) &\rightarrow (-\beta - 1, \gamma + 1), & \text{if } \beta \geq 1, & (-\beta - 1, \gamma + 1) \in V \\ \text{"1"}: (-\beta, \gamma) &\rightarrow (1, \gamma + 1), & \text{if } \beta \geq 1, \gamma = 0 \bmod 2 & \\ \text{"1"}: (-\beta, \gamma) &\rightarrow (1, \beta), & \text{if } \beta \geq 1, \gamma = 1 \bmod 2. & \end{aligned} \quad (4)$$

Fig. 4 illustrates the 34-state transition diagram for the constraint $M(j=3, k=6, t=4)$.

B. Twins Constraint for j -Constrained Trellis

For channels with memory $L \geq j + 1$, the j -constraint can readily be incorporated into the detector to reduce the number of states or branches in the trellis, and to increase the maximum possible code rate by allowing new potential code sequences that were forbidden in $M(j, k, t)$. Fig. 5 illustrates in a Venn diagram, where the set members are sequences, the relationship between the various constrained systems as seen at the partial response channel input and the sequence detector

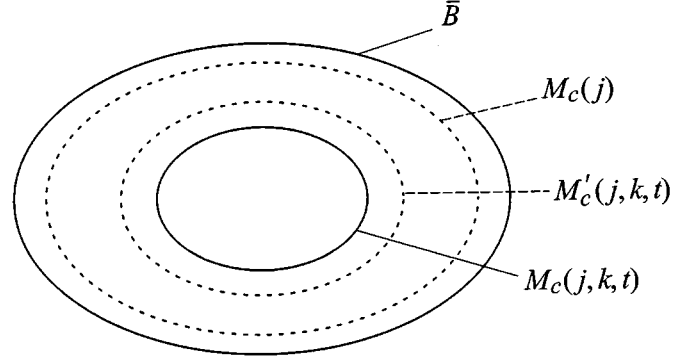


Fig. 5. Relationship between MTR-constrained systems and detector output sequences.

output. Assuming an unconstrained trellis, we have the partial response channel input sequences $Y = M_c(j, k, t)$, represented by the innermost solid circle, and the detector output sequences $\hat{Y} = \bar{B}$ represented by the outermost solid circle. This corresponds to the case studied in the previous section. By incorporating the j -constraint into the trellis of the sequence detector, we effectively constrain the possible detector output sequences to $\hat{Y} = M_c(j)$ represented by the outermost dashed circle. Thus, new potential encoder output sequences represented by the area between the innermost solid and innermost dashed circles are allowed. We denote the constraint defined by the innermost dashed circle by $M'_c(j, k, t)$, with the corresponding constraint prior to precoding denoted by $M'(j, k, t)$. Note that the set difference $M'(j, k, t) \setminus M(j, k, t)$ consists of all sequences that satisfy the $MTR(j, k)$ constraint and contain a string of at least $t+1$ pairs of zeros or ones but whose complement violates the $MTR(j)$ constraint. Note that for $j=2$ or $j=3$, any j -constrained sequence consisting of consecutive pairs of zeros or ones and whose binary complement is also j -constrained must consist of an alternating string of 00 and 11, and so $M'(j=2, k, t)$ simply requires that the maximum length of a string of the form 00110011... or 11001100... be $2t$ (in addition to the $M(j=2, k)$ constraint).

For the $M'(j, k, t)$ constraint, a deterministic FSTD is described as follows. The set V' of vertices for this FSTD lies again on an integer lattice and can be described as the union of three sets (we consider only the case $k \geq j$)

$$\begin{aligned} V' = \{ & (\alpha, \gamma): 1 \leq \alpha \leq j, 1 \leq \gamma \leq 2t+1, \alpha \leq \gamma \\ & \alpha = \gamma \bmod 2 \} \\ & \cup \{ (-\beta, \gamma): 1 \leq \beta \leq j, 1 \leq \gamma \leq 2t+1, \beta \leq \gamma \\ & \beta = \gamma \bmod 2 \} \\ & \cup \left\{ (-\beta, \gamma): j+1 \leq \beta \leq k, \gamma = 2 \left\lfloor \frac{j}{2} \right\rfloor \right\}. \end{aligned} \quad (5)$$

For $k \geq j \geq 2$, the total number of states N' can be computed using the formula

$$N' = 2jt + k - j + 2 - \left\lfloor \frac{j-3}{2} \right\rfloor \left(\left\lfloor \frac{j-3}{2} \right\rfloor + 1 \right) - \left\lfloor \frac{j-3}{2} \right\rfloor \left(\left\lfloor \frac{j-3}{2} \right\rfloor + 1 \right) \quad (6)$$

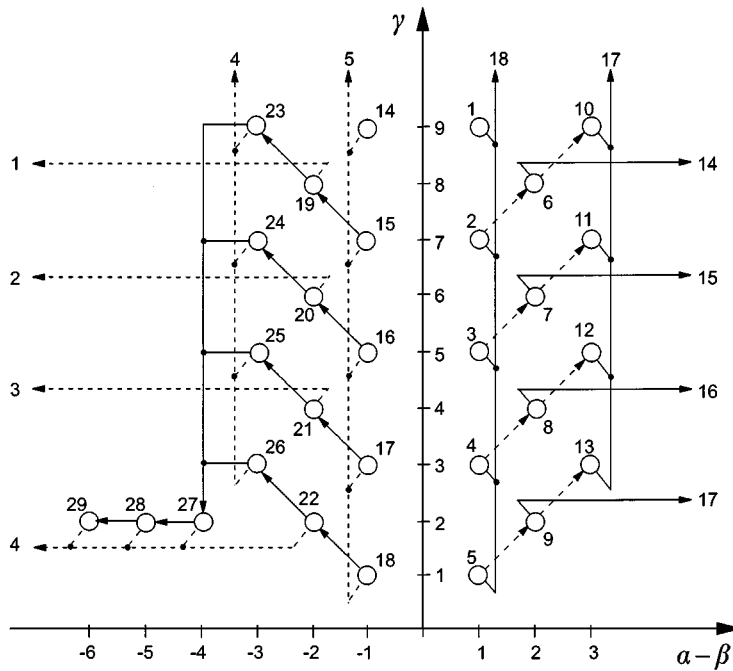


Fig. 6. FSTD presenting constrained system $M'(3, 6, 4)$ (an arrow represents a set of transitions, and the number next to an arrow indicates the terminal state associated with that set of transitions).

and the state transitions are given by the rules

$$\begin{aligned}
 \text{"1"}: (\alpha, \gamma) &\rightarrow (\alpha + 1, \gamma + 1), \\
 &\text{if } \alpha \geq 1, (\alpha + 1, \gamma + 1) \in V' \\
 \text{"0"}: (\alpha, \gamma) &\rightarrow (-1, \gamma + 1), \\
 &\text{if } \alpha \geq 1, \gamma = 0 \bmod 2 \\
 \text{"0"}: (\alpha, \gamma) &\rightarrow (-1, \alpha), \\
 &\text{if } \alpha \geq 1, \gamma = 1 \bmod 2 \\
 \text{"0"}: (-\beta, \gamma) &\rightarrow (-\beta - 1, \gamma + 1), \\
 &\text{if } \beta \geq 1, (-\beta - 1, \gamma + 1) \in V' \\
 \text{"0"}: (-\beta, \gamma) &\rightarrow \left(-\beta - 1, 2 \left\lfloor \frac{j}{2} \right\rfloor\right), \\
 &\text{if } \beta \geq 1, \left(-\beta - 1, 2 \left\lfloor \frac{j}{2} \right\rfloor\right) \in V' \\
 \text{"1"}: (-\beta, \gamma) &\rightarrow (1, \gamma + 1), \\
 &\text{if } \beta \geq 1, \gamma = 0 \bmod 2 \\
 \text{"1"}: (-\beta, \gamma) &\rightarrow (1, \beta), \\
 &\text{if } \beta \geq 1, \gamma = 1 \bmod 2.
 \end{aligned} \tag{7}$$

Fig. 6 illustrates the 29-state transition diagram for the constraint $M'(j = 3, k = 6, t = 4)$. The set of states that are arranged in a vertical line corresponding to a fixed β , $\beta \geq j + 1$ in Fig. 4 collapse into a single state $(-\beta, 2\lfloor j/2 \rfloor)$, $j + 1 \leq \beta \leq k$ in Fig. 6. An alternative approach to constructing equivalent FSTDs for $j = 2$ and $j = 3$, which track the run of all four phases of patterns of type (0011) arriving at a state, has been taken in [16]. A special case of the constrained system described in (5)–(7) has been used in [8] and [14] to eliminate period-4 quasi-catastrophic sequences of type (0011) at the input of the

$1/(1 \oplus D)$ precoder for E²PR4 and modified E²PR4 systems. Note, though, that for $j \geq 4$, these sequences are not necessarily periodic as claimed in [8].

IV. CAPACITY OF MTR CONSTRAINTS

The capacity of a constrained system S represents the maximum achievable code rate of an encoder generating sequences from S . It is given by $\text{Cap}(S) = \log_2 \lambda_{\max}(A)$, where $\lambda_{\max}(A)$ is the largest real eigenvalue of the adjacency matrix A associated with a deterministic FSTD that presents S [1], [11], [23]. For instance

$$\begin{aligned}
 \text{Cap}(M(j = 2)) &\approx 0.8791 \\
 \text{Cap}(M(j = 3)) &\approx 0.9468 \\
 \text{Cap}(M(j = 4)) &\approx 0.9752.
 \end{aligned}$$

For an MTR constraint $M(j, k, t)$, the adjacency matrix, and therefore the capacity, can be computed from (2) and (4). Tables I–III list the capacity of various constraints $M(j, k, t)$ for $j = 2, 3$, and 4 by truncating the numbers after the sixth digit following the decimal point. We remark that for reasons of symmetry, the capacity of the constrained system $M(j = a, k = b, t = c)$ is equal to the capacity of the constrained system $M(j = b, k = a, t = c)$. Tables IV–VI list the capacity of MTR constraints $M'(j, k, t)$ for $j = 2, 3$ and 4 by truncating the numbers after the sixth digit following the decimal point.

It is not hard to show that

$$\lim_{k \rightarrow \infty, t \rightarrow \infty} \text{Cap}(M(j, k, t)) = \text{Cap}(M(j)).$$

Thus, also

$$\lim_{k \rightarrow \infty, t \rightarrow \infty} \text{Cap}(M'(j, k, t)) = \text{Cap}(M(j)).$$

TABLE I
CAPACITY OF MTR CONSTRAINTS $M(j = 2, k, t)$ FOR UNCONSTRAINED DETECTOR TRELLIS

t	k								
	2	3	4	5	6	7	8	9	10
1	0.551463	0.637088	–	–	–	–	–	–	–
2	0.650899	0.747205	0.772712	0.786634	–	–	–	–	–
3	0.679286	0.781989	0.814167	0.831134	0.836644	0.839656	–	–	–
4	0.688789	0.790071	0.828494	0.847904	0.855144	0.859157	0.860525	0.861272	–
5	0.692203	0.793352	0.833940	0.854013	0.862591	0.867323	0.869204	0.870263	0.870630
6	0.693470	0.794181	0.836133	0.856337	0.865673	0.870693	0.872960	0.874244	0.874765
7	0.693948	0.794533	0.837013	0.857283	0.866995	0.872139	0.874618	0.875997	0.876634
8	0.694130	0.794624	0.837377	0.857653	0.867569	0.872753	0.875360	0.876786	0.877488
9	0.694199	0.794662	0.837527	0.857803	0.867820	0.873022	0.875694	0.877141	0.877882
10	0.694225	0.794672	0.837589	0.857863	0.867929	0.873139	0.875846	0.877302	0.878063
11	0.694235	0.794677	0.837614	0.857887	0.867978	0.873190	0.875915	0.877374	0.878147
12	0.694239	0.794678	0.837625	0.857897	0.867999	0.873212	0.875947	0.877408	0.878186

TABLE II
CAPACITY OF MTR CONSTRAINTS $M(j = 3, k, t)$ FOR UNCONSTRAINED DETECTOR TRELLIS

t	k								
	2	3	4	5	6	7	8	9	10
1	0.637088	0.694241	–	–	–	–	–	–	–
2	0.747205	0.834520	0.850147	0.858590	–	–	–	–	–
3	0.781989	0.867061	0.893311	0.907863	0.910964	0.912597	–	–	–
4	0.790071	0.875696	0.907137	0.922734	0.928302	0.931518	0.932231	0.932603	–
5	0.793352	0.878139	0.911889	0.928056	0.934762	0.938345	0.939691	0.940478	0.940654
6	0.794181	0.878850	0.913612	0.929879	0.937201	0.941002	0.942654	0.943557	0.943901
7	0.794533	0.879059	0.914243	0.930538	0.938162	0.942036	0.943850	0.944819	0.945248
8	0.794624	0.879120	0.914476	0.930778	0.938539	0.942439	0.944341	0.945336	0.945810
9	0.794662	0.879138	0.914563	0.930865	0.938689	0.942599	0.944543	0.945549	0.946046
10	0.794672	0.879144	0.914595	0.930896	0.938749	0.942662	0.944626	0.945636	0.946146
11	0.794677	0.879145	0.914607	0.930908	0.938772	0.942687	0.944661	0.945673	0.946188
12	0.794678	0.879146	0.914612	0.930912	0.938782	0.942697	0.944675	0.945687	0.946206

V. CONNECTION BETWEEN MTR AND (G, I) CONSTRAINTS

Let $P(G, I)$ denote the set of (G, I) constrained sequences, and let $P_c(G, I)$ denote the set of all allowable sequences after mapping the binary outputs of the $1/(1 \oplus D^2)$ precoder applied to $P(G, I)$ into bipolar symbols. Thus, both $P_c(G, I)$ and $M_c(j, k, t)$ pertain to bipolar sequences at the input to the generalized partial response channel. The following proposition [22] states that the $P_c(G, I)$ constraints are a subclass of the $M_c(j, k, t)$ constraints.

Proposition: $P_c(G, I) = M_c(G + 1, G + 1, I)$.

Proof: The $1/(1 \oplus D^2)$ precoder following the (G, I) encoder can be represented as the serial concatenation of two $1/(1 \oplus D)$ precoders, as shown in Fig. 7. The G -constraint translates into the j -constraint, $j = G + 1$, as well as the k -constraint, $k = G + 1$, at the output of the first $1/(1 \oplus D)$ precoder. It can readily be shown that the I -constraint at the output of the (G, I) encoder translates into the t -constraint,

$t = I$, at the output of the first $1/(1 \oplus D)$ precoder. Therefore, the (G, I) -constrained system $P_c(G, I)$ is identical to the MTR-constrained system $M_c(G + 1, G + 1, I)$. Q.E.D.

As the capacity of a constrained system is not affected by an invertible rate-1 code (such as a precoder), we conclude the following.

Corollary: $\text{Cap}(P(G, I)) = \text{Cap}(M(G + 1, G + 1, I))$.

This result is consistent with computed capacities (compare the capacity of the (G, I) constraint (see, e.g., [24]) with the capacity of the corresponding MTR constraint in Tables I–III).

The connection between (G, I) and MTR constraints suggests a new approach for constructing a (G, I) code that employs the $1/(1 \oplus D)$ precoder instead of the commonly used $1/(1 \oplus D^2)$ precoder. For example, it is well known that a rate-8/9 ($G = 3, I = 6$) code can be implemented as a block code because a list of 272 freely concatenatable codewords exists [5]. An alternative code construction methodology would be to use a $M(j = 4, k = 4, t = 6)$ code associated with a

TABLE III
CAPACITY OF MTR CONSTRAINTS $M(j = 4, k, t)$ FOR UNCONSTRAINED DETECTOR TRELLIS

t	k								
	2	3	4	5	6	7	8	9	10
2	0.772712	0.850147	0.864320	0.871955	–	–	–	–	–
3	0.814167	0.893311	0.915723	0.928185	0.930853	0.932243	–	–	–
4	0.828494	0.907137	0.934253	0.948104	0.952674	0.955281	0.955858	0.956154	–
5	0.833940	0.911889	0.941533	0.955975	0.961585	0.964625	0.965677	0.966281	0.966416
6	0.836133	0.913612	0.944539	0.959137	0.965376	0.968639	0.969965	0.970695	0.970951
7	0.837013	0.914243	0.945812	0.960445	0.967037	0.970393	0.971883	0.972680	0.973009
8	0.837377	0.914476	0.946359	0.960999	0.967776	0.971168	0.972754	0.973583	0.973956
9	0.837527	0.914563	0.946595	0.961233	0.968108	0.971514	0.973154	0.973996	0.974396
10	0.837589	0.914595	0.946698	0.961333	0.968258	0.971670	0.973338	0.974187	0.974601
11	0.837614	0.914607	0.946742	0.961375	0.968325	0.971739	0.973423	0.974274	0.974697
12	0.837625	0.914612	0.946762	0.961393	0.968356	0.971771	0.973462	0.974315	0.974741

TABLE IV
CAPACITY OF MTR CONSTRAINTS $M'(j = 2, k, t)$ FOR j -CONSTRAINED DETECTOR TRELLIS

t	k								
	2	3	4	5	6	7	8	9	10
1	0.551463	0.637088	0.681917	0.705786	0.718904	0.726305	0.730562	0.733044	0.734504
2	0.650899	0.747205	0.788831	0.809001	0.819355	0.824853	0.827843	0.829472	0.830380
3	0.679286	0.781989	0.826469	0.847561	0.858127	0.863601	0.866496	0.868048	0.868886
4	0.688789	0.790071	0.833407	0.853896	0.864136	0.869427	0.872218	0.873708	0.874510
5	0.692203	0.793352	0.836591	0.856988	0.867163	0.872410	0.875172	0.876645	0.877436
6	0.693470	0.794181	0.837227	0.857540	0.867672	0.872898	0.875648	0.877114	0.877901
7	0.693948	0.794533	0.837531	0.857819	0.867939	0.873157	0.875903	0.877366	0.878152
8	0.694130	0.794624	0.837593	0.857870	0.867985	0.873200	0.875944	0.877407	0.878193
9	0.694199	0.794662	0.837622	0.857896	0.868008	0.873223	0.875967	0.877429	0.878214
10	0.694225	0.794672	0.837629	0.857901	0.868013	0.873227	0.875970	0.877433	0.878218
11	0.694235	0.794677	0.837631	0.857903	0.868015	0.873229	0.875972	0.877435	0.878220
12	0.694239	0.794678	0.837632	0.857903	0.868015	0.873229	0.875973	0.877435	0.878220

$1/(1 \oplus D)$ precoder. Clearly, Fig. 7 shows that the $(G = 3, I = 6)$ block code in [5] combined with the $1/(1 \oplus D)$ precoder results in a $M(j = 4, k = 4, t = 6)$ two-state code; the corresponding block decoder combined with the inverse precoder $(1 \oplus D)$ leads to a sliding-block decoder with a ten-bit window.

Now, a rate-8/9 $M(j = 4, k = 4, t = 6)$ block code does not exist because the maximum number of freely concatenatable nine-bit codewords for this constraint turns out to be only 232 (see the procedure, due to Freiman and Wyner, for computing the maximum size of such a set of codewords in [23]). However, we can construct a rate-8/9 $M(j = 4, k = 4, t = 6)$ six-state code that is block-decodable. Although such a code is more complicated than is the well-known rate-8/9 $(G = 3, I = 6)$ block code, it may have a slight advantage described as follows.

In partial response channels, an isolated error in the detector output is among the most common error events. For the standard $(G = 3, I = 6)$ code, such an event generates an error of the form 101 after $(1 \oplus D^2)$ inverse-precoding, and such an event can span two (G, I) codewords (and, therefore, two user bytes)

with a probability of 2/9. On the other hand, for the alternative $M(j = 4, k = 4, t = 6)$ code, the isolated error generates an error of the form 11 after $(1 \oplus D)$ inverse-precoding, and such an event can span two MTR codewords (and, therefore, two user bytes) with a probability of only 1/9. Of course, how much this would affect the ultimate user byte error rate would depend on the details of the data-to-codeword assignment.

VI. CODE DESIGN METHODOLOGIES I: BLOCK CODES, LOOK-AHEAD CODING, AND SUBSTITUTION RULES

The techniques of look-ahead coding and violation detection combined with substitution have been successfully applied in the past to design efficient constrained codes such as RLL(d, k) codes [25], [33], MTR(j, k) codes [26], and PRML(G, I) codes [27]. The MTR code design methodologies presented in this section are based on the use of the state transition diagrams for MTR constraints and the application of the look-ahead coding technique and violation detection combined with substitution.

TABLE V
CAPACITY OF MTR CONSTRAINTS $M'(j = 3, k, t)$ FOR j -CONSTRAINED DETECTOR TRELLIS

t	k								
	2	3	4	5	6	7	8	9	10
1	0.637088	0.694241	0.726640	0.744486	0.754400	0.759984	0.763169	0.765002	0.766065
2	0.747205	0.834520	0.870763	0.887666	0.896007	0.900257	0.902463	0.903621	0.904233
3	0.781989	0.867061	0.903461	0.920320	0.928513	0.932612	0.934698	0.935771	0.936327
4	0.790071	0.875696	0.911530	0.928002	0.935967	0.939933	0.941943	0.942972	0.943503
5	0.793352	0.878139	0.913770	0.930146	0.938058	0.941994	0.943986	0.945005	0.945529
6	0.794181	0.878850	0.914374	0.930697	0.938582	0.942504	0.944488	0.945503	0.946025
7	0.794533	0.879059	0.914548	0.930857	0.938734	0.942652	0.944634	0.945648	0.946169
8	0.794624	0.879120	0.914596	0.930898	0.938773	0.942689	0.944671	0.945684	0.946205
9	0.794662	0.879138	0.914609	0.930910	0.938784	0.942700	0.944681	0.945694	0.946215
10	0.794672	0.879144	0.914613	0.930913	0.938787	0.942703	0.944684	0.945697	0.946218
11	0.794677	0.879145	0.914614	0.930914	0.938788	0.942704	0.944685	0.945698	0.946219
12	0.794678	0.879146	0.914614	0.930914	0.938788	0.942704	0.944685	0.945698	0.946219

TABLE VI
CAPACITY OF MTR CONSTRAINTS $M'(j = 4, k, t)$ FOR j -CONSTRAINED DETECTOR TRELLIS

t	k								
	2	3	4	5	6	7	8	9	10
2	0.772712	0.850147	0.864320	0.871955	0.876045	0.878240	0.879422	0.880060	0.880405
3	0.814167	0.893311	0.915723	0.928185	0.934338	0.937441	0.939025	0.939841	0.940264
4	0.828494	0.907137	0.934253	0.948104	0.954740	0.958016	0.959659	0.960493	0.960917
5	0.833940	0.911889	0.941533	0.955975	0.962892	0.966294	0.967994	0.968850	0.969285
6	0.836133	0.913612	0.944539	0.959137	0.966112	0.969534	0.971240	0.972098	0.972532
7	0.837013	0.914243	0.945812	0.960445	0.967426	0.970847	0.972551	0.973407	0.973839
8	0.837377	0.914476	0.946359	0.960999	0.967981	0.971401	0.973103	0.973958	0.974390
9	0.837527	0.914563	0.946595	0.961233	0.968212	0.971631	0.973332	0.974186	0.974618
10	0.837589	0.914595	0.946698	0.961333	0.968310	0.971727	0.973427	0.974281	0.974713
11	0.837614	0.914607	0.946742	0.961375	0.968351	0.971767	0.973467	0.974321	0.974753
12	0.837625	0.914612	0.946762	0.961393	0.968368	0.971784	0.973484	0.974338	0.974769

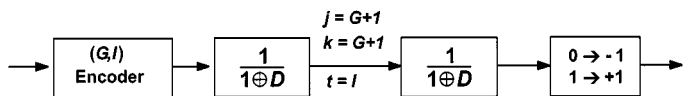


Fig. 7. Connection between MTR and (G, I) constraints.

The encoders and decoders designed using these methodologies are usually both of sliding-block type. As high-rate codes for partial response channels reduce the performance penalty associated with the rate loss, we emphasize the design of high-rate MTR codes that can efficiently be implemented using Boolean logic.

The first code design example improves the parameters of the rate-6/7 $M(j = 2, k = 8)$ code detailed in [26]. Specifically, k is reduced from eight to seven while ensuring that the code is not quasi-catastrophic. Two modified versions of the rate-6/7 code are then used in conjunction with a rate-4/5 code to construct a rate-16/19 partitioned-block code that is amenable to efficient Boolean implementation. A further increase in code rate is achieved by relaxing the $j = 2$ constraint. A rate-16/17 block code that satisfies a $j = 3, 4$ time-varying MTR constraint [14]

TABLE VII
SUBSTITUTION TABLE FOR THE RATE-6/7 $M'(2, 7, 9)$ LOOK-AHEAD CODE

Prohibited Pattern							Substitute Pattern						
x_4	x_5	x_6	$x_7,$	x_1	x_2	x_3	x_4	x_5	x_6	$x_7,$	x_1	x_2	x_3
x_4	0	1	1,	1	0	0	x_4	0	1	0,	1	1	0
0	0	1	1,	1	0	1	0	1	0	0,	1	1	0
1	0	1	1,	1	0	1	0	1	1	0,	1	1	0
x_4	0	0	0,	0	0	0	x_4	0	0	0,	1	1	0

is employed to obtain a rate-16/17 $MTR(j = 3)$ look-ahead code.

A. Rate-6/7 $MTR(j = 2)$ Look-Ahead Code

A set of 57 potential codewords is generated by starting in state two in Fig. 2(a) and making seven transitions such that states one or two are the terminal states. This set of 57 codewords can be freely concatenated to obtain sequences that satisfy the $M(j = 2)$ constraint. This set of 57 codewords is aug-

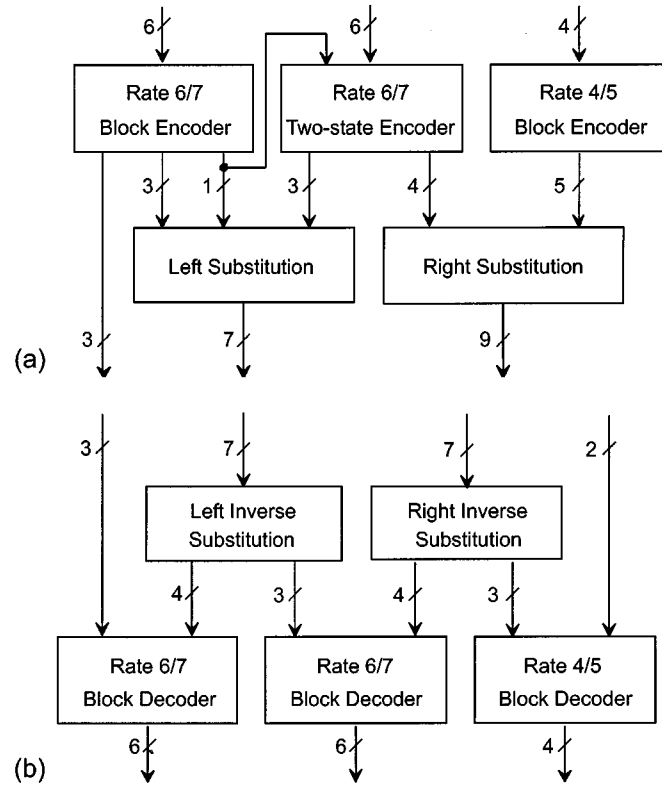


Fig. 8. Implementation of the rate-16/19 MTR($j = 2$) code: (a) block encoder and (b) block decoder.

mented by 11 seven-bit codewords that are generated by starting in state two and ending in state three. The first two bits of the 11 codewords are either 00 or 01 or 10, and the last three bits of the 11 codewords are always 011. In hexadecimal format, the 11 codewords are 03,0B,13,23,43,1B,2B,4B,33,53,5B. In this way, a total of 68 potential codewords is obtained. The codewords 00,01,40,33 are discarded to arrive at a rate-6/7 block code.

Concatenating freely any of the 64 codewords from the above list would give rise to sequences that occasionally violate the $M(j = 2)$ constraint at codeword boundaries. Furthermore, the maximum zero run length is $k = 10$. The substitutions given in Table VII avoid the violation of the $M(j = 2)$ constraint at codeword boundaries and reduce k to 7. Note that the commas in Table VII indicate codeword boundaries. The decoder resolves these substitutions by looking two bits forward and three bits backward; i.e., the window size for the sliding-block decoder is 12. The worst case for the constraint that limits the path memory happens when the string $\dots 10011, 0011001, 1001100, 1\dots$ occurs; i.e., we have $t = 9$. The sliding-block encoder and the sliding-block decoder can be implemented with very few logic gates and the complexity of the Boolean implementation is similar to the one detailed in [26].

B. Rate-16/19 MTR($j = 2$) Block Code

Fig. 8 shows the partitioned-block structure of the encoder for the rate-16/19 MTR($j = 2$) block code. The rate-16/19 block code is composed of three subcodes that are described in the following. The first rate-6/7 subcode is a block code that is defined by discarding the set of four codewords 00,01,4C,33 from the initial list of 68 potential codewords that were initially available

for the rate-6/7 block code discussed in the previous subsection. The second rate-6/7 subcode is a two-state code where the last bit of the 7-bit codeword of the first rate-6/7 subcode determines the current state s . There are 62 state-independent codewords obtained from the same initial list of 68 codewords by discarding the set of six codewords 00,4C,33,19,56,06. One of the remaining two state-dependent codewords is 56 if $s = 0$ or 33 if $s = 1$. The other state-dependent codeword is 4C if $s = 0$ or 06 if $s = 1$. Finally, the third rate-4/5 subcode is a block code that is obtained in the following way. A total of 17 potential codewords can be generated by starting in state two in Fig. 2(a), making five transitions and ending in state one or two. In hexadecimal format, these codewords are 00,01,02,04,05,06,08,09,0A,0C,0D,10,11,12,14,15,16. Out of these 17 codewords, 00 is discarded to obtain the rate-4/5 block code, a list of 16 codewords.

All codewords of the first rate-6/7 subcode cannot start with 11, and all codewords of the third rate-4/5 subcode cannot end with 11. Therefore, violation of the $M(j = 2)$ constraint at the boundaries of 19-bit codewords is not possible. However, violations could occur at the boundary between two seven-bit codewords or between a seven-bit codeword and a five-bit codeword that follows it. The substitutions in Tables VIII and IX ensure that violations of the $M(j = 2)$ constraint are not present after the substitutions. Furthermore, they reduce k and t to shorten the path memory and aid timing recovery and gain control. The substitution operations associated with Tables VIII and IX can be regarded as a rate-7/7 and a rate-10/10 block code, respectively.

This code satisfies the $M'(j = 2, k = 9, t = 6)$ constraint. The efficiency of a code is defined as the ratio of the code rate

TABLE VIII
LEFT SUBSTITUTION TABLE FOR THE RATE-16/19 $M'(2, 9, 6)$ BLOCK CODE

Prohibited Pattern							Substitute Pattern						
x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_4	0	1	1	1	0	0	x_4	0	1	0	1	1	0
0	0	1	1	1	0	1	0	1	0	0	1	1	0
1	0	1	1	1	0	1	0	1	1	0	1	1	0
x_4	0	0	0	0	0	0	x_4	0	0	0	1	1	0

and the capacity [1]. From Table IV, the capacity associated with these constraints is approximately 0.8771, resulting in an efficiency of 96.01%. The main feature of this code is the partitioned-block structure, which allows a relatively simple Boolean implementation for the encoder and the decoder.

Fig. 8 depicts the decoder for the rate-16/19 $MTR(j = 2)$ block code. The decoder performs a one-to-one mapping from the set of all allowable 19-bit words $x_1x_2x_3 \cdots x_{19}$ into the set of 16-bit data words. After detecting violations, i.e., $x_8x_9 = 11$ and/or $x_{15}x_{16} = 11$, the decoder performs inverse substitution using Tables VIII and IX. Following the operation of inverse substitution, three decoders perform block decoding.

A rate-16/19 $MTR(j = 2, k = 7, t = \infty)$ block code does exist [7]. However, this code is not a partitioned-block code and is therefore not amenable to efficient Boolean implementation.

C. Rate-16/17 $MTR(j = 3)$ Look-Ahead Code

A rate-16/17 block code exists [14] that satisfies the $j = 3$ constraint uniformly except at the border of two 17-bit codewords where the constraint is relaxed to $j = 4$. We refer to this special time-varying j -constraint (with period 17) as the mod 17 $j = 3, 4$ constraint. A total of $65753 > 2^{16}$ potential codewords can be generated by starting in state two in Fig. 2(b), making 17 transitions and terminating in states one, two, or three. Among these codewords, 199 codewords begin or end with ten zeros. After discarding these codewords and 17 more codewords that start with the first 15 bits of one of the strings (1001) or (0110) or (0011) or (1100) or end with the first 16 bits of one of the strings (1001) or (0110) or (0011) or (1100), a set of 65537 codewords is obtained that can be freely concatenated without violating the constraints $j = 3, 4, k = 18$, and $t = 14$.

This rate-16/17 $j = 3, 4$ block code can be used to construct a $j = 3$ look-ahead code. Fig. 9 shows a possible implementation of the rate-16/17 $j = 3$ look-ahead code. The rate-16/17 $j = 3, 4$ block encoder generates a 17-bit codeword that depends solely on the 16 bits at its input. Using one codeword look-ahead, the 17-bit output window of the block encoder is shifted by 13 time periods to the left. Violations in these shifted encoder output window are detected and replaced by the strings given in Table X. In the absence of any violation, no replacement is made. The substitution operation is a one-to-one mapping and can be regarded as a rate-17/17 block code. However, the domain for this mapping is only a subset of the set of all 17-bit strings. The decoder first performs inverse substitution by detecting violations, i.e., raising a flag whenever 17-bit strings at the input of the decoder end with

01110 or 01110xxx, where xxx stands for the unknown last three bits. The inverse substitution operation can be again regarded as a rate-17/17 block code. Using one codeword look-ahead, the 17-bit output window after inverse substitution is shifted by four time periods to the left. The decoder then maps the shifted 17-bit words into the original 16-bit words by using the inverse of the one-to-one mapping that has been used by the rate-16/17 block code. The code that is obtained this way satisfies the constraint $M'(j = 3, k = 12, t = 9)$.

The capacity of the constraint $M'(3, 12, 9)$ is approximately 0.9466. The efficiency of the code is therefore 99.42%. A similar $M'(j = 3, k = 11, t = 16)$ look-ahead code has been independently constructed in [14].

D. Time-Varying $MTR(j = 3, 4)$ Codes

$MTR(j = 2)$ codes [7] and time-varying $MTR(j = 2, 3)$ codes [8], [9], i.e., MTR codes that do not allow three consecutive transitions to appear in any two encoded sequences at positions offset by one symbol, have the interesting property of eliminating many error events, including all error events $\{e_i\}$ of the type $\{\pm 2, \mp 2, \pm 2\}$, $\{\pm 2, \mp 2, \pm 2, \mp 2\}$, $\{\pm 2, \mp 2, \pm 2, \mp 2, \pm 2\}$, and so on. These error events correspond to mistaking the polarity of an alternating write current for three or more channel symbol intervals. However, the distance gain by eliminating these error events is offset by a significant rate loss penalty. For example, the capacity of the mod 2 $MTR(j = 2, 3)$ constraint (with period 2) is approximately 0.9162 [8].

It has been observed that for the Lorentzian recording model and NPML detection, the most dominant error events are $\{\pm 2, \mp 2, \pm 2\}$, $\{\pm 2, \mp 2, \pm 2, \mp 2 \pm 2\}$, and $\{\pm 2, \mp 2, \pm 2, \mp 2, \pm 2, \mp 2, \pm 2\}$ (see, e.g., [10]). An alternative coding strategy is therefore to allow the error event $\{\pm 2, \mp 2, \pm 2\}$ to occur and to eliminate all other error events that correspond to mistaking the polarity of an alternating write current for four or more channel symbol intervals. In this way, higher rate codes can be constructed, thereby reducing the signal-to-noise ratio (SNR) penalty due to rate loss. We next introduce time-varying $MTR(j = 3, 4)$ codes that do not allow four consecutive transitions to appear in any two encoded sequences at positions offset by one symbol. As one particular example, Fig. 10 shows the section of the code trellis corresponding to two symbol intervals of a mod 2 $MTR(j = 3, 4)$ (with period 2) constraint and the corresponding adjacency matrix. The capacity associated with this constraint is approximately 0.9613, allowing the construction of rate-24/25 time-varying $MTR(j = 3, 4)$ codes. Note that the capacity of $MTR(j = 4)$ is approximately 0.9752. Another example in this new class of codes is the mod 17 $MTR(j = 3, 4)$ constraint (with period 17) that gives rise to the rate-16/17 block code that allows four consecutive transitions to occur only at codeword boundaries, as was discussed in the previous section.

E. Rate-8/10 $MTR(j = 2)$ Block Code that Avoids Colliding Dibits

By a dibit (or tribit), we mean two (or three) consecutive magnetic transitions. Prior to $1/(1 \oplus D)$ -precoding, this is equivalent to two (or three) consecutive ones: 11 (or 111). So, an

TABLE IX
RIGHT SUBSTITUTION TABLE FOR THE RATE-16/19 $M'(2, 9, 6)$ BLOCK CODE

Prohibited Pattern									Substitute Pattern								
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
x_{11}	0	1	1	1	0	0	x_{18}	x_{19}	x_{11}	0	1	0	1	1	0	x_{18}	x_{19}
x_{11}	0	1	1	1	0	1	x_{18}	x_{19}	x_{11}	0	0	0	1	1	0	x_{18}	x_{19}
0	0	0	0	0	0	0	x_{18}	x_{19}	0	1	1	0	1	1	0	x_{18}	x_{19}
0	0	1	1	0	0	1	1	0	0	1	0	0	1	1	0	1	0

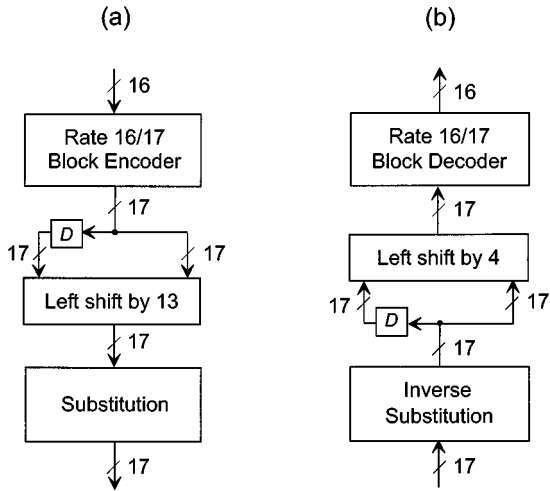


Fig. 9. Implementation of the rate-16/17 MTR($j = 3$) code: (a) sliding-block encoder and (b) sliding-block decoder.

MTR($j = 2$) code is precisely the same as a code that forbids tritbits. Thus, in a typical coded MTR($j = 2$) sequence, we may have numerous dibits but no tritbits.

The closest appearance of two dibits in such a code would occur when there is only one intervening 0: 11011. We call such a pattern a “colliding dibit” for the following reason. In very high-density magnetic recording, write precompensation schemes are used to mitigate the problem of nonlinear transition shift, a deleterious effect of intersymbol interference. One particular scheme [28] would adjust the writing of the string 11011 such that the two dibits collide (more precisely, such that the second transition of the first dibit collides with the first transition of the second dibit) and, therefore, cannot be resolved upon reading; the situation is even worse if there are tritbits. Thus, for such a scheme, it is desirable to encode the data to prohibit colliding dibits, in addition to satisfying the MTR($j = 2$) constraint, equivalently to prohibit the two strings: 111 and 11011. We denote this constraint by $M(j = 2, c = 2)$, where in general c denotes the minimum number of intervening zeros between two appearances of 11 [similarly, we have the constraints $M(j, k, c)$, $M(j, k, t, c)$, and $M'(j, k, t, c)$]. A deterministic FSTD for $M(j = 2, c = 2)$ is shown in Fig. 11. The capacity of the constraint is approximately 0.8579.

A very simple rate 8:10 block code for $M'(j = 2, k = 10, t = 6, c = 2)$ can be constructed as follows. There are a total of 299 potential codewords of length 10 that can be generated by starting at state two in Fig. 11, and ending at states one,

two, or four. The sequences obtained by concatenating these words satisfy the $M(j = 2, c = 2)$ constraint. In this list of codewords, there are 19 words that begin or end with six zeros. Discarding these words, the resulting code achieves the $k = 10$ constraint. Among the remaining 280 codewords, 23 begin with 1001100 or 0110011 or 0011001. Discarding these words, we are left with 257 codewords that (when freely concatenated) satisfy the $M'(j = 2, k = 10, t = 6, c = 2)$ constraint.

Higher rate codes for this constraint do exist; for instance, in principle, a rate 6:7 code can be constructed (note that $6/7 \approx 0.8571$), although it may be complicated; in Section VII-B, we will outline the construction of a rate-16/19 (≈ 0.8421) finite-state code.

VII. CODE DESIGN METHODOLOGIES II: FINITE-STATE CODES VIA STATE SPLITTING

The state-splitting algorithm is a method of constructing finite-state modulation encoders that have high efficiency, limited decoder error propagation, and satisfy strong constraints. We refer the reader to [23, Section IV] for complete details of the algorithm, but for expository purposes, we give a brief summary here.

Given a constraint S , the algorithm begins with a deterministic FSTD G that presents S (for the remainder of this paper, G will denote an FSTD and not a G -constraint). Given a desired code rate, $p/q \leq \text{Cap}(S)$, we first construct an FSTD G^q presenting S^q , the version of the constraint where sequences are divided into nonoverlapping q -tuples; in particular, the FSTD G^q is labeled with binary q -tuples. Then, states of G^q are iteratively split in (perhaps several) rounds; within each round, a subset U of states is split by partitioning outgoing edges from each element of U , thereby creating descendants of each split state, and incoming edges are replicated to each descendant state. Ultimately, we arrive at an FSTD H , which presents S and contains a sub-FSTD H' such that each state of H' has at least 2^p outgoing edges. By deleting excess edges, we obtain an FSTD E that presents a subsystem of S^q and has exactly 2^p outgoing edges at each state. We obtain an encoder by “tagging” the outgoing edges of E at each state with distinct binary p -tuples; the tagging can be regarded as a data-to-codeword assignment.

Let A denote the adjacency matrix of the FSTD G . Then, A^q is the adjacency matrix of G^q . The state-splitting process is guided by a so-called $(A^q, 2^p)$ approximate eigenvector, namely, a nonnegative integer vector v , not identically 0,

TABLE X
SUBSTITUTION TABLE FOR THE RATE-16/17 $M'(3, 12, 9)$ LOOK-AHEAD CODE

Prohibited Pattern								Substitute Pattern							
x_{14}	x_{15}	x_{16}	x_{17}	x_1	x_2	x_3	x_4	x_{14}	x_{15}	x_{16}	x_{17}	x_1	x_2	x_3	x_4
x_{14}	0	1	1,	1	1	0	x_4	x_{14}	0	x_4	0,	1	1	1	0
0	0	0	0,	0	0	0	0	0	1	1	1,	0	0	0	0
1	0	0	1,	1	0	0	1	0	1	1	1,	0	0	0	1
0	0	1	1,	0	0	1	1	0	1	1	1,	0	0	1	1
1	1	0	0,	1	1	0	0	0	1	1	1,	0	1	0	0
0	1	1	0,	0	1	1	0	0	1	1	1,	0	1	1	0

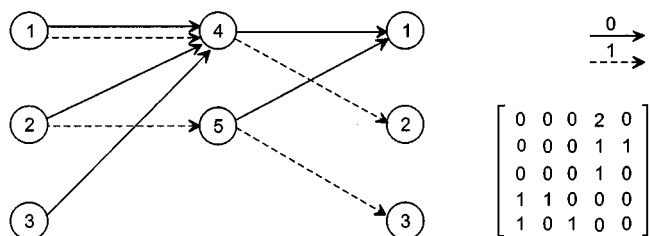


Fig. 10. Trellis section for time-varying mod 2 MTR($j = 3, 4$) constraint.

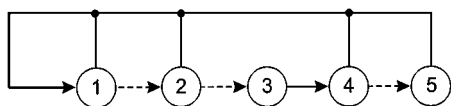


Fig. 11. FSTD presenting constrained system MTR($j = 2, c = 2$).

indexed by the states of the FSTD and satisfying the vector inequality

$$A^q v \geq 2^p v$$

(an algorithm due to Franaszek [29] (see also [23]) can be applied to find an approximate eigenvector with minimal maximum entry). The entries of an approximate eigenvector are often called “weights.” The idea is that beginning with the FSTD G^q , ultimately, each state s , with $v_s > 0$, is split into v_s descendant states, each with weight 1. The final FSTD H will then have an approximate eigenvector whose entries are zeros and ones. The subgraph H' is defined by the states of H corresponding to states whose entries are 1; and then the encoder E is defined by deleting excess edges and “tagging” the remaining edges with binary p -tuples. This would appear to yield an encoder with $\sum v_s$ states. However, typically, many states can be merged together resulting in a far simpler encoder; state merging is described as follows (the merging can take place before, during, or after splitting).

For a state s in an FSTD, the follower set $F(s)$ is the set of sequences that can be generated from state s . Whenever

$$F(s_2) \subseteq F(s_1) \tag{8}$$

states s_1 and s_2 can be merged to form a new state that has the same follower set as s_2 without violating the constraint. Moreover, if

$$F(s_2) \subseteq F(s_1) \quad \text{and} \quad v_{s_2} = v_{s_1} \tag{9}$$

then the new state inherits the common weight v_{s_2} (i.e., the new FSTD formed by merging the two states has an approximate eigenvector obtained by copying all weights of unmerged states and using the common weight for the weight of the merged state). Also, if

$$F(s_2) \subseteq F(s_1) \quad \text{and} \quad v_{s_2} \leq v_{s_1} \tag{10}$$

then typically (though not always) state s_2 can be split into v_{s_2} states, each of which merges into one of the v_{s_1} states into which s_1 is split. Similar results hold for a chain of follower set inclusions involving more than two states.

The resulting code will be sliding-block-decodable with a window consisting of a certain number of blocks: m blocks coming from the memory of the constraint (namely, the number of q -bit blocks needed to determine a terminal state in the original FSTD), one block coming from the current block to be decoded and a blocks of anticipation coming from the number of rounds of state splitting; hence, the total window size is $m+a+1$ blocks. But often we can arrange the data-to-codeword assignment in a sufficiently consistent way that the memory of the decoder is reduced to zero. Thus, it typically happens that if only one round of splitting is required to construct an encoder, the window of the decoder will consist of only $2(= 0 + 1 + 1)$ blocks.

A. Rate-7/8 MTR($j = 2$) Finite-State Code

Recall that the capacity of the MTR($j = 2$) constraint is approximately 0.8791, which is just slightly above $7/8 = 0.875$. Thus, it is possible to design a rate $7 : 8$ MTR($j = 2$) code. In fact, such a code can be designed by state splitting. Here, we outline the construction of such a code that also satisfies the $M'(j = 2, k = 14, t = 10)$ constraint, has a sliding window decoder consisting of two eight-bit blocks, and has “reasonable” encoder and decoder complexity.

We begin the construction with the very simple deterministic FSTD G [shown in Fig. 2(a)] for the $M(j = 2)$ constraint, and then delete some excess words in order to impose the k - and t -constraint. The adjacency matrix for G is

$$A = A_G = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

For a rate 7 : 8 code, we need the adjacency matrix of the eighth power of G

$$A^8 = A_{G^8} = \begin{bmatrix} 81 & 44 & 24 \\ 68 & 37 & 20 \\ 44 & 24 & 13 \end{bmatrix}.$$

The Franaszek algorithm yields $v = (6, 5, 3)$ as an $(A^8, 2^7)$ -approximate eigenvector; i.e.,

$$A^8 v \geq 128v \geq 2^7 v. \quad (11)$$

It turns out that we need only one round of splitting to construct an encoder according to the approximate eigenvector v : the first state is split into six descendants, the second state into five descendants, and the third state into three descendants. Moreover, because of the containment relationship among the follower sets

$$F(3) \subset F(2) \subset F(1)$$

[this should be clear from Fig. 2(a)], we can use (10) to merge three descendants of state three into three of the five descendants of state two, and the five descendants of state two into five of the six descendant states of state one. This yields an encoder with only six states. An application of the general result in [30, Corollary 1] reveals that this is the smallest number of encoder states possible for a rate 7 : 8 encoder into this constraint.

The memory of the constraint is only two bits (and, thus, at most one eight-bit block) long. As only one round of splitting is required, the decoder has a sliding window of at most $3(= 1 + 1 + 1)$ blocks. But with care in the data-to-codeword assignment, the decoder memory can be reduced to zero, and so the decoder actually has a two-block sliding window. Moreover, the foregoing can still be carried out even with the deletion of the edge labeled 00000000 from state one to state one in G^8 (this reduces the $(1, 1)$ -entry of A^8 from 81 to 80; we can verify that the inequality (11) still holds if the $(1, 1)$ -matrix-entry is reduced to 80). It follows that the k -constraint is at most 14. Moreover, by judicious selection of the set of edges to delete from the final split graph, we can limit the maximum number of alternating pairs of zeros and ones to $t = 10$. In this way, we arrive at a rate 7 : 8 $M'(2, 14, 10)$ encoder, which is sliding-block decodable with a sliding window equal to two blocks. Moreover, the complexity is reasonable. Using the data-to-codeword assignment heuristics in [31] and the Berkeley SIS logic synthesis program, we constructed such an encoder with approximately 400 multiple-input gates and a corresponding total of 1900 cells in CMOS 6SF. For this assignment, we divided the six states into two groups of three and defined the logic separately for each group. Further savings in logic may be obtained by sharing the logic across the groups. In particular, these estimates of the number of gates/cells should be regarded as rough upper bounds.

It may well be that even tighter k - and t -constraints can be enforced for $MTR(j = 2)$ at rate 7 : 8 with reasonable complexity and error propagation. The capacity of $MTR(j = 2, k = 8)$ is still above 7/8, but such a code would probably be complicated. On the other hand, a reasonable state-splitting construction for a rate 7 : 8 $MTR(j = 2, k = 12, t < \infty)$ code is not out of the question. For such a code, we could begin with the 14-state

FSTD G for $MTR(j = 2, k = 12)$, apply state-splitting and merging operations, and delete edges to enforce a t -constraint. The Franaszek algorithm produces an $(A_{G^8}, 2^7)$ approximate eigenvector of length 14 with maximum entry 6. Although a state-splitting construction based on such a vector may look complicated, it can be greatly simplified by the merging condition (9). We illustrate a construction of this type in the next section.

B. Rate-16/19 $MTR(j = 2)$ Finite-State Code that Avoids Colliding Dibits

Recall from Section VI-E the notion of an $MTR(j = 2)$ constraint that also avoids colliding dibits, denoted $M(j = 2, c = 2)$, whose capacity is approximately 0.8579. In that section, we discussed the construction of a rate 8 : 10 $M(j = 2, c = 2)$ block code, in fact a rate 8 : 10 $M'(j = 2, k = 10, t = 6, c = 2)$ block code. Here, we outline the construction of a higher rate $(16/19 \approx 0.8421) M(j = 2, c = 2)$ finite-state code that satisfies $M'(j = 2, k = 13, t = 11, c = 2)$. In order to control complexity as well as decoder error propagation, we make use of a time-varying version of the state-splitting algorithm, introduced in [32]. This technique yields a time-varying encoder in two alternating phases, one at rate 8 : 9 and the other at rate 8 : 10, for an overall rate of 16 : 19.

As in the preceding example, we could begin with the very simple FSTD, shown in Fig. 11, for $MTR(j = 2, c = 2)$, and then delete some excess words (after splitting) in order to enforce the k - and t -constraints. After an initial examination of capacities, we decided to apply a somewhat more aggressive k -constraint and pursue a $M(j = 2, k = 13, c = 2)$ code, and so we begin with a deterministic FSTD G for that constraint; this FSTD turns out to have 17 states.

From G , we form the FSTD $G^{(9,10)}$: this is a “two-phase” FSTD whose state set consists of two disjoint subsets, S_0 and S_1 , with all outgoing transitions from states in S_0 labeled with nine-bit blocks and ending at a state in S_1 , and all outgoing transitions from states in S_1 labeled with ten-bit blocks and ending at a state in S_0 ; we refer to the two phases as phase 0 and phase 1. All sequences obtained by traversing this FSTD (concatenating strings of alternating nine-bit and ten-bit blocks) satisfy the $M(j = 2, k = 13, c = 2)$ constraint. Letting A_G denote the adjacency matrix of G , we see that the adjacency matrix of $G^{(9,10)}$ is

$$A_{G^{(9,10)}} = \begin{bmatrix} 0 & A_G^9 \\ A_G^{10} & 0 \end{bmatrix}.$$

An $(A_{G^{(9,10)}}, 2^8)$ approximate eigenvector is of the form, $v = (u, w)$, where both u and w are vectors indexed by the states of G and

$$A_G^{10} u \geq 2^8 w \quad \text{and} \quad A_G^9 w \geq 2^8 u.$$

Here, Franaszek’s approximate eigenvector algorithm yields vectors u and w of length 17 with maximal entry 3 for u and 4 for w . Now, using the inclusion relationships among follower sets and the merging condition (9), states of $G^{(9,10)}$ can be merged into a new two-phase FSTD G' with corresponding approximate eigenvector $v' = (u', w')$, where u' has length 5

and maximal entry 3 and u' has length 7 and maximal entry 4. For instance, for phase 0, it turns out that all states with weight (i.e., u -entry) equal to 3 form a chain of follower set inclusions and, hence, can be merged to a state that still has weight 3, whereas the states with weight equal to 2 do not form a chain of follower set inclusions; they break up into two groups that do and, hence, yield two merged states, both with weight 2. The situation is analogous for states with weight 1.

It then turns out that we can split G' in one round to obtain a new FSTD G'' with an approximate eigenvector having only 0,1 entries; upon merging states of G'' , we obtain a new two-phase FSTD G''' with only three states in phase 0 and only four states in phase 1 (this merging is accomplished using not only the inclusion relationships among the follower sets, but also intersections of follower sets). After tagging all edges with eight-bit input labels, we obtain a time-varying encoder that alternates between the two phases, one at rate 8:9 and the other at rate 8:10, as desired. This encoder satisfies the $M(j = 2, k = 13, c = 2)$ constraint and has two phases (three encoder states in phase 0: the 8:9 phase, and four encoder states in phase 1: the 8:10 phase); as in the preceding example, excess edges can be deleted to enforce a t -constraint, in fact the encoder satisfies the $M'(j = 2, k = 13, t = 15, c = 2)$ constraint, and the edges can be endowed with input tags such that the window of the sliding-block decoder consists of only two blocks (in a time-varying way, alternately a nine-bit code block followed by a ten-bit code block and a ten-bit code block followed by a nine-bit code block). Again, using the heuristics in [31] and the Berkeley SIS logic synthesis program, we obtain an encoder with approximately 250 (or 350) multiple-input gates in phase 0 (or 1), corresponding to approximately 1250 (or 1750) cells in CMOS6SF. Again, these estimates should be regarded as rough upper bounds.

Finally, we remark that there are tradeoffs between the k - and t -constraints for $MTR(j = 2, c = 2)$ codes. For instance, we can obtain a rate 16:19 code similar to the one discussed in this section with t reduced from 15 to 11 at the expense of increasing k from 13 to 16.

VIII. CONCLUSION

MTR codes that avoid quasi-catastrophic error propagation in sequence detectors for generalized partial response channels with spectral nulls at dc and the Nyquist frequency have been presented. The codes satisfy an additional new constraint, which we call the twins constraint, to eliminate quasi-catastrophic error propagation. The twins constraint for unconstrained and j -constrained sequence detectors has been studied in detail. MTR constraints in conjunction with the twins constraint have been characterized using deterministic FSTDs. The capacity of this extended class of MTR constraints was computed.

The introduction of the twins constraint coupled with the observation that the $1/(1 \oplus D^2)$ precoder can be represented by the cascade of two $1/(1 \oplus D)$ precoders revealed a connection between (G, I) and MTR constraints. Specifically, it has been shown that $1/(1 \oplus D^2)$ -precoded (G, I) constraints are a subclass of $1/(1 \oplus D)$ -precoded $M(j, k, t)$ constraints. The link between $M(j, k, t)$ and (G, I) constraints provides a new

methodology for constructing $1/(1 \oplus D)$ -precoded codes that constrain the input of the partial response channel in the same manner as $1/(1 \oplus D^2)$ -precoded (G, I) codes.

Finally, the methods of state splitting and look-ahead coding in combination with violation detection/substitution have been applied to the construction of MTR codes for digital data storage. Design examples for high-rate MTR codes that minimize rate loss have been emphasized. The use of performance-enhancing MTR codes in high-density magnetic recording is a promising approach for providing high data reliability with low implementation complexity.

ACKNOWLEDGMENT

The authors are grateful to several IBM colleagues for help with this work. In particular, they thank J. Coker and R. Galbraith for collaborations on practical aspects of code designs, and R. Olson for suggesting the use of MTR codes in conjunction with write precompensation schemes. They additionally thank M. Chen, A. Dholakia, T. Mittelholzer, and R. New for helpful discussions. They also thank the anonymous referees for their careful reading, valuable comments, and suggestions to improve the presentation. Finally, R. D. Cideciyan and E. Eleftheriou would like to acknowledge the editorial suggestions provided by L.-M. Pavka.

REFERENCES

- [1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.
- [2] R. L. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding-block codes: An application of symbolic dynamics to information theory," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 5–22, Jan. 1983.
- [3] 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.
- [4] 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.
- [5] J. Eggenberger and A. M. Patel, "Method and apparatus for implementing optimum PRML codes," U.S. Patent 4 707 681, Nov. 17, 1987.
- [6] A. M. Patel, "Rate 16/17 (0,6/6) code," *IBM Tech. Discl. Bull.*, vol. 31, pp. 21–23, Jan. 1989.
- [7] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Trans. Magn.*, vol. 32, pp. 3992–3994, Sept. 1996.
- [8] W. G. Bliss, "An 8/9 rate time-varying trellis code for high density magnetic recording," *IEEE Trans. Magn.*, vol. 33, pp. 2746–2748, Sept. 1997.
- [9] K. K. Fitzpatrick and C. S. Modlin, "Time-varying MTR codes for high density magnetic recording," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 1997, pp. 1250–1253.
- [10] B. E. Moision and P. H. Siegel, "Distance enhancing constraints for noise predictive maximum likelihood detectors," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 1998, pp. 2730–2735.
- [11] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge, U.K.: Cambridge Univ. Press, 1995.
- [12] B. H. Marcus, P. H. Siegel, and J. K. Wolf, "Finite-state modulation codes for data storage," *IEEE J. Select. Areas Commun.*, vol. 10, pp. 3–37, Jan. 1992.
- [13] W. G. Bliss, S. She, and L. Sundell, "The performance of generalized maximum transition run trellis codes," *IEEE Trans. Magn.*, vol. 34, pp. 85–90, Jan. 1998.
- [14] T. Nishiya, K. Tsukano, T. Hirai, T. Nara, and S. Mita, "Turbo-EEPRML: An EEPR4 channel with an error-correcting post-processor designed for 16/17 rate quasi-MTR code," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 1998, pp. 2706–2711.

- [15] R. Karabed, P. H. Siegel, and E. Soljanin, "Constrained coding for binary channels with high intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1777–1797, Sept. 1999.
- [16] R. D. Cideciyan and E. Eleftheriou, "Nonquasicatastrophic maximum transition run codes," in *Proc. IEEE Int. Symp. Inform. Theory*, Sorrento, Italy, June 2000, p. 256.
- [17] P. R. Chevillat, E. Eleftheriou, and D. Maiwald, "Noise predictive partial-response equalizers and applications," in *Proc. IEEE Int. Conf. Commun.*, June 1992, pp. 942–947.
- [18] E. Eleftheriou and W. Hirt, "Noise-predictive maximum-likelihood (NPML) detection for the magnetic recording channel," in *Proc. IEEE Int. Conf. Commun.*, June 1996, pp. 556–560.
- [19] J. D. Coker, E. Eleftheriou, R. L. Galbraith, and W. Hirt, "Noise-predictive maximum likelihood (NPML) detection," *IEEE Trans. Magn.*, vol. 34, pp. 110–117, Jan. 1998.
- [20] S. A. Altekar, M. Berggren, B. E. Moision, P. H. Siegel, and J. K. Wolf, "Error-event characterization on partial-response channels," *IEEE Trans. Inform. Theory*, vol. 45, pp. 241–247, Jan. 1999.
- [21] G. D. Forney and A. R. Calderbank, "Coset codes for partial-response channels; or, coset codes with spectral nulls," *IEEE Trans. Inform. Theory*, vol. 35, pp. 925–943, Sept. 1989.
- [22] R. D. Cideciyan and E. Eleftheriou, "(0,G/I) codes are a subclass of MTR codes," *Electron. Lett.*, vol. 36, pp. 642–643, Mar. 2000.
- [23] B. H. Marcus, R. M. Roth, and P. H. Siegel, "Constrained systems and coding for recording channels," in *Handbook of Coding Theory*, R. Brualdi, C. Huffman, and V. Pless, Eds. Amsterdam, The Netherlands: Elsevier, 1998.
- [24] M. C. Stefanovic and B. V. Vasic, "Channel capacity of (0,G/I) codes," *Electron. Lett.*, vol. 29, pp. 243–245, Jan. 1993.
- [25] G. V. Jacoby and R. Kost, "Binary two-thirds rate code with full word look-ahead," *IEEE Trans. Magn.*, vol. 20, pp. 709–714, Sept. 1984.
- [26] B. Brickner and J. Moon, "Design of a rate 6/7 maximum transition run code," *IEEE Trans. Magn.*, vol. 33, pp. 2749–2751, Sept. 1997.
- [27] A. J. van Wijngaarden and K. A. S. Immink, "Combinatorial construction of high rate runlength-limited codes," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 1996, pp. 343–347.
- [28] R. Olson, private communication.
- [29] P. A. Franaszek, "On future-dependent block coding for input-restricted channels," *IBM J. Res. Develop.*, vol. 23, pp. 75–81, 1979.
- [30] B. H. Marcus and R. M. Roth, "Bounds on the number of states in encoder graphs for input-restricted channels," *IEEE Trans. Inform. Theory*, vol. 37, pp. 742–758, May 1991.
- [31] D. S. Modha and B. H. Marcus, "Art of constructing low-complexity encoders/decoders for constrained block codes," *IEEE J. Select. Areas. Commun.*, vol. 19, pp. 589–601, Apr. 2001.
- [32] J. Ashley and B. Marcus, "Time-varying encoders for constrained systems: An approach to limiting error propagation," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1038–1043, May 2000, to be published.
- [33] M. Cohn, G. V. Jacoby, and C. A. Bates III, U.S. Patent 4337458, June 1982.



Roy D. Cideciyan (S'82–M'86–SM'98) received the Dipl.-Ing. degree from Aachen University of Technology, Germany, in 1981 and the M.S.E.E. and Ph.D. degrees from the University of Southern California, Los Angeles, in 1982 and 1985, respectively.

He joined IBM Research, Zurich Research Laboratory, Rüschlikon, Switzerland, in 1986, where he has been working in the areas of signal processing and coding for magnetic recording, coding for high-speed local-area networking, and wireless communications. His research interests include

performance evaluation, signal processing, coding, and synchronization for magnetic recording and transmission systems.



Evangelos Eleftheriou (S'81–M'86–SM'00) received the electrical engineering degree from the University of Patras, Greece, in 1979, and the M.Eng. and Ph.D. degrees in electrical engineering from Carleton University, Ottawa, Ontario, Canada, in 1981 and 1985, respectively.

He joined the IBM Zurich Research Laboratory, Rüschlikon, Switzerland, in 1986, where he has been working in the areas of high-speed voiceband data modems, wireless communications, and coding and signal processing for the magnetic recording channel.

Since 1998, he has managed the magnetic recording and wired transmission activities at the IBM Zurich Research Laboratory. His primary research interests lie in the areas of communications and information theory, particularly signal processing and coding for recording and transmission systems. He holds over 30 patents (granted and pending applications) in the areas of coding and detection for transmission and digital recording systems, and he was named a Master Inventor at IBM Research in 1999.

He was Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS from 1994 to 1999 in the area of equalization and coding. He is Co-Guest Editor of the *IEEE Journal on Selected Areas of Communications* the Special Issue "The Turbo Principle: From Theory to Practice."



Brian H. Marcus (M'84–F'00) received the B.A. degree from Pomona College, Pomona, CA, in 1971 and the Ph.D. degree in mathematics from the University of California (UC) at Berkeley in 1975.

He held an IBM T.J. Watson Postdoctoral Fellowship in mathematical sciences in 1976–1977. From 1975 to 1985, he was Assistant Professor and then Associate Professor of Mathematics (with tenure) at the University of North Carolina at Chapel Hill. Since 1984, he has been a Research Staff Member at the IBM Almaden Research Center, San Jose, CA. He

has held visiting teaching positions at several universities, including Stanford University, Stanford, CA, and UC Berkeley. He is the author of more than 50 research papers in mathematics and engineering journals, as well as a textbook, *An Introduction to Symbolic Dynamics and Coding* (Cambridge, U.K.: Cambridge Univ. Press, 1995). He also holds five patents.

Dr. Marcus is a member of Phi Beta Kappa. His current research interests include constrained coding, error correction coding, channel detection, and symbolic dynamics. He was a corecipient of the Leonard G. Abraham Prize Paper Award of the IEEE Communications Society in 1993 and gave an invited plenary lecture at the 1995 International Symposium on Information Theory.



Dharmendra S. Modha (S'92–M'95) received the B.Tech. degree in computer science and engineering in 1990 from the Indian Institute of Technology, Bombay, and the M.S. and Ph.D. degrees in electrical and computer engineering in 1993 and 1995, respectively, from the University of California at San Diego.

From June 1990 to August 1991, he founded, managed, and profitably sold a software development company called Solutions in Bombay, India. Following a one-year postdoctoral research fellow-

ship at the Department of Electrical and Computer Engineering, University of California at San Diego, he has been a Research Staff Member in the Computer Science Theory Group at the IBM Almaden Research Center, San Jose, CA, since 1997. He holds one patent. His current research interests include data/text/hypertext mining and low-density parity check codes.

Dr. Modha is a member of the IMS.