High-Rate Maximum Runlength Constrained Coding Schemes Using Nibble Replacement

Kees A. Schouhamer Immink Turing Machines Inc.
Willemskade 15b-d, 3016 DK Rotterdam
The Netherlands
Nanyang Technological University, Singapore
Email: immink@turing-machines.com

Summary - *We will present simple and systematic constructions of high-rate binary maximum runlength constrained codes. The new construction has the virtue that large look-up tables for encoding and decoding are not required. We will compare the error propagation performance of codes based on the new construction with that of prior art codes.*

I. Introduction

We will discuss efficient high-rate block codes with applications in hard disk drives (HDD) and optical recording systems that impose a constraint on the maximum run of transitions in the encoded sequence, i.e. MTR-constrained codes. MTR-constrained codes have at most k consecutive 'one's, and are therefore related to conventional k-constrained codes that have a maximum run of k consecutive 'zero's.

Properties and constructions of MTR *runlength-limited* codes have been collected in [1]. These codes are usually used in conjunction with 1/(1$\oplus$ D2) and 1/(1$\oplus$ D) precoding, respectively. MTR(j, k, t) codes impose a j-constraint to limit the number of alternating binary symbols at the channel input to $j + 1$, a k-constraint to limit the number of like symbols at the channel input to $k + 1$, and a twins t-constraint to avoid quasi-catastrophic error propagation for partial-response channels with spectral nulls at dc and the Nyquist frequency.

Low-rate k-constrained codes are usually constructed in practice using table look-up. A first difficulty arising from the adoption of a very high rate modulation code is the considerable complexity usually required to realize practical hardware implementations of the encoding and decoding processes. A second difficulty is *error propagation*, a phenomenon where a single error in the received word may result in massive amounts of decoded errors. Various methods for systematically designing high-rate k-constrained codes have been published in the art, see [1] for a survey. Kautz [2] was probably the first who presented a simple algorithmic method, called *enumerative encoding*, for translating user words into k-constrained codewords and *vice versa*. Wijngaarden & Immink presented various codes of rate $1-1/n$, where subsequences that violate the maximum runlength are iteratively removed to obtain a k-constrained sequence [3].

An straightforward way of constructing high-rate codes is by *interleaving* coded and uncoded symbols, where the coded symbols are obtained from a low-rate k-constrained base code. An advantage of the interleaving scheme is that error propagation is confined to the 'coded' part of the codewords. For example, constrained words generated by a rate 8/9, $k = 3$, code can be interleaved with uncoded (user) symbols. A rate 24/25, $k = 11$, based on interleaving of the above rate 8/9, code, has been presented by Fisher & Fitspatrick [4] and Sonntag [5]. McEwen *et al.* [6] [7] and McClellan [8] presented interleaved codes of rates including 32/33, 48/49, 56/57, 72/73, 50/51. For example, in the rate 50/51 code, the base code has rate 10/11, and the 11-bit word produced by the base code encoder is divided into four nibbles of lengths 3, 3, 3, and 2. The four nibbles are interleaved with four 10-bit user data. Then, the new codeword consists of the four 10-bit symbols interleaved with the 11-bit base code nibbles, for a total of 51 bits.

Denissen & Tolhuizen presented an alternative coding method, where the codeword consists of L q-bit nibbles, where w, $0 < w < 2^q$, specific q-bit characters are removed from the repertoire of 2^q possible nibbles using Galois Field arithmetic GF(2^q) [9]; the code was proposed to be able to use the inadmissible nibble(s) as synchronization markers. The code requires one additional (redundant) q-bit nibble, and the number of user nibbles that can be accommodated in this code is at most $\lfloor (2^q - w)/w \rfloor$, where w denotes the number of inadmissable nibbles. This method can be used to generate an RLL sequence, since in case, we bar the 'all-zero' nibble, each q-bit nibble generated has at least one '1', so that the maximum (zero) run length of the cascaded sequence of q-bit nibbles equals $k = 2(q - 1)$.

Silvus & Anderson [10] presented an alternative to the above coding method, where the codeword is partitioned into L q-bit nibbles, and where w vexatious nibbles are inadmissible, where the binary input word is converted into an integer in a base $M = 2^q - w$. The base-M integers, in turn, are translated, using a look-up table, into the allowed q-bit nibbles. A disadvantage of this method is a) rapidly mounting code complexity with growing codeword length, and b) massive error propagation during decoding [11].

In this article, we will study a simple alternative to the above

This project was supported by grant *Theory and Practice of Coding and Cryptography*, Award Number: NRF-CRP2-2007-03

coding methods using a 'nibble replacement' algorithm for constructing k-constrained codes. The usage of large look-up tables is avoided and average error propagation is less than that in prior art enumerative schemes. It is assumed that a given n-bit codeword can be partitioned into L q-bit nibbles, L, $q > 0$, that is $Lq = n$. The nibble replacement guarantees that w predefined inadmissible nibbles will be excluded from the repertoire of all, 2^q, possible nibbles. The work on the nibble replacement technique presented here was stimulated by the coding techniques presented by Wijngaarden & Immink [12] and McEwen *et al.* [6] [7]. We have tried to cast their *ad hoc* procedures into a more systematic framework. The structure of this paper is as follows.

We start, in Section II, with a description of a new code construction, a "nibble replacement" technique, where one redundant bit is added to a block of $n-1$ user (source) bits. Disallowed nibbles will be removed. In Section III, we will work out some constructions so that the performance of the new code can be compared with that of prior art codes. In Section IV, we will present results of error propagation simulations showing the performance difference between the new codes and prior art codes. Section V concludes the paper.

II. Nibble replacement technique

It is assumed that $(n-1)$ binary user symbols $(a_1, \ldots, a_{n-1})$ are translated into a codeword of length n bits. The integer n is a multiple of the integer q, that is, $n = Lq$, where q is the nibble size and L the number of q-bit nibbles in a codeword. The aim of the encoding procedure is to guarantee that each q-bit nibble generated is a member of a predefined set of nibbles. The $(n-1)$ user bits are partitioned into $(L-1)$ blocks of q bits and one block of $(q-1)$ bits. Let $(a_1, \ldots, a_{n-1})$ denote the binary user data word, which is partitioned, as said, into L nibbles, $\boldsymbol{u}_i$, $1 \leq i \leq L$. The first nibble, $\boldsymbol{u}_1 = (a, a_1, \ldots, a_{q-1})$, called *pivot nibble*, contains $q-1$ user bits plus a bit called *pivot bit* a. The value of the pivot bit, a, will be determined by the encoder as we will explain in the next subsection. The remaining $(L-1)$ q-bit nibbles are defined by $\boldsymbol{u}_i = (a_{(i-1)q}, \ldots, a_{iq-1})$, $2 \leq i \leq L$.

A. Encoding algorithm

The encoder will remove all nibbles whose decimal representation is less than w, and will replace them with nibbles whose decimal representation is larger than or equal to w. The encoder proceeds as follows. Set the pivot bit, a, equal to '1'. The encoder scans the sequence of L nibbles. If all nibbles $\boldsymbol{u}_i$ in the address range $2 \leq i \leq L$ are admissible, that is $\mathrm{dec}(\boldsymbol{u}_i) \geq w$, where the integer function $\mathrm{dec}(z)$ equals the decimal representation of the binary integer z, then we transmit the codeword $\boldsymbol{u}_1 \boldsymbol{u}_2 \ldots \boldsymbol{u}_L$. In other words, no changes are effected to the input word except pre-pending the extra '1'. However, in case there is at least one inadmissible nibble $\boldsymbol{u}_v$ in the codeword at position v, that is, $\mathrm{dec}(\boldsymbol{u}_v) < w$ some changes are effected to the codeword. The encoder will replace the first found inadmissible nibble at address v by the pivot nibble $\boldsymbol{u}_1$. Further, the pivot bit is set to '0' (to signal the receiver that the source word has been modified), and the remaining $q-1$ bits of the pivot nibble will be replaced by data indicative of the address v and the value, $\boldsymbol{u}_v$, of the disallowed nibble. After the two changes in the pivot nibble and the first found disallowed nibble have been effected, the encoder then iterates till the end of the codeword, treating a replaced nibble as a pivot nibble in the first replacement in a similar manner as done in the first pass, and a found disallowed nibble is replaced by the current pivot nibble and so on, so that all disallowed nibbles will be replaced.

Clearly, the information on both the position and the value of a first nibble replaced must be uniquely restored by the decoder. Said information, that is the position and the value of a first nibble replaced, will be packed into at most $q-1$ bits of the pivot nibble. The number of bit combinations available in the pivot nibble equals $2^{q-1} - w$, and the number of 'position and value' combinations needed equals $w(L-1)$. Thus a nibble replacement code is possible if

$$2^{q-1} - w \geq w(L-1).$$

So that we infer that the maximum number of nibbles, L, that can be accommodated equals

$$L \leq \left\lfloor \frac{2^{q-1}}{w} \right\rfloor. \tag{1}$$

We note that the redundancy of the nibble replacement technique is about a factor of $2\ln(2) \approx 1.39$ higher than that of the base-conversion coding scheme [11]. The next algorithm provides a formal description of the nibble replacement routine.

Replacement algorithm/Encoding routine
Input: The integers L, q, w, and the binary $(n-1)$-bit data word $(a_1, \ldots, a_{n-1})$.
Define the L q-bit nibbles $\boldsymbol{u}_1$ = $(a, a_1, \ldots, a_{q-1})$ and $\boldsymbol{u}_i$ = $(a_{(i-1)q}, \ldots, a_{iq-1})$, $2 \leq i \leq L$.
Output: List of encoded nibbles $\boldsymbol{u}_i$, where $\mathrm{dec}(\boldsymbol{u}_i) \geq w$, $1 \leq i \leq L$.
Initialize the pivot bit $a = 1$ and the integer $v = 1$.
For $i = 2, \ldots, L$ **if** $w_p = \mathrm{dec}(\boldsymbol{u}_p) < w$ **then begin** $\boldsymbol{u}_p = \boldsymbol{u}_v$, $\boldsymbol{u}_v = \mathrm{bin}((p-1)w + w_p)$, $v = p$ **end**. By unpacking the L q-bit nibbles $\boldsymbol{u}_i$, we obtain the n-bit codeword $(a_1, \ldots, a_n)$.

At the conclusion of the above recursive replacement routine, all inadmissible nibbles will be removed, and replaced by admissible nibbles. Note that the pivot bit equals '1' in case the user data is sent unmodified or it equals '0' in case one or more nibble replacements have been made. As a result, the receiver can easily detect that a nibble replacement has been effected. The modified nibbles contain either position pointer information or the data of the pivot nibble. The modified nibble with the highest index contains the data of the pivot nibble, i.e. equals $\boldsymbol{u}_1$. Decoding of the received codeword can be accomplished in a straightforward way by recursively undoing the replacements. Two worked coding examples will be helpful.

Example *1:* Let $w = 1$, $q = 3$ and $L = 4$; let the user data be '11 000 110 111' (the spaces between the groups of bits are written

for clerical convenience). We append a '1', and obtain '111 000 110 111'. The first (and only) occurrence of an 'all-zero' nibble is at position $p = 2$. We replace that nibble with the content of the first nibble, '111', that is $\boldsymbol{u}_2$ = '111', and the first nibble, $\boldsymbol{u}_1$, is set to $\text{bin}(p-1)$ = '001'. Then we obtain, as there are no other 'all-zero' nibbles to be replaced, the codeword '001 111 110 111'.

Example *2:* Let $w = 2$, $q = 4$, and $L = 4$; let the user data be '000 0001 0000 1111'. We append a '1', and obtain '1000 0001 0000 1111'. Set $v = 1$. We find the first inadmissible nibble at position $p = 2$. Then we set $v = p = 2$, $\boldsymbol{u}_2 = \boldsymbol{u}_1$ = '1000', $w_p = \text{dec}(\boldsymbol{u}_2) = 1$, and $\boldsymbol{u}_1$ = $\text{bin}((p-1)w + w_p)$ = '0011', where the integer function $\text{bin}(z)$ equals the q-bit binary representation of the integer z. So that we obtain after the first replacement the intermediate result '0011 1000 0000 1111'. The second inadmissible nibble is found at position $p = 3$. We have $\boldsymbol{u}_3 = \boldsymbol{u}_2$ = '1000', $w_p = \text{dec}(\boldsymbol{u}_3) = 0$, $\boldsymbol{u}_2 = \text{bin}((p-1)w + w_p)$ = '0100'. So that we obtain as final result '0011 0100 1000 1111'.

Remark The above algorithm is written to remove the w lowest ranking inadmissible nibbles. The coding algorithm is, however, more general, and can be employed to remove any set of inadmissible nibbles. We can do so by a straightforward translation of the set of $2^q - w$ allowed nibbles in the above algorithm into a second predefined set of $2^q - w$ nibbles allowed in a different scheme. Alternatively, we may rewrite the above algorithm by implementing an adjustment of the presetting of the pivot bit and the other pointer data.

The following decoding routine will retrieve the original data word $(a_1, \ldots, a_{n-1})$.

Decoding routine
Input: The integers L, q, w, and the L encoded q-bit nibbles $\boldsymbol{u}_i$.
Output: L decoded q-bit nibbles $\boldsymbol{v}_i$.
Initialize the integer $p = 1$ and copy the q-bit nibbles $\boldsymbol{v}_i$ = $\boldsymbol{u}_i$, all i.
repeat $s = \text{dec}(\boldsymbol{u}_p)$; $po = p$; $p = 1 + s$ div w; $\boldsymbol{v}_p$ = bin(dec $(\boldsymbol{u}_{po}) - (p-1) * w)$ **until** $(s \geq 2^{n-1})$; $\boldsymbol{v}_1 = \text{bin}(s)$.

The above routine recursively replaces the nibbles that were replaced by the encoder. The routine halts when the most significant bit (msb) of a replaced nibble $\boldsymbol{u}_p$ equals '1'. By invoking a straightforward reshuffling routine, the original data word $(a_1, \ldots, a_{n-1})$ can be retrieved from the L decoded q-bit nibbles $\boldsymbol{v}_i$. Note that the sequence of integers p, which refer to the positions where nibbles are replaced, during the various replacement (decoding) steps is increasing in magnitude as they refer to higher indices. This is a useful property that can be optionally exploited to signal the decoding circuitry and/or halt the above decoding routine in case bit errors have been introduced during transmission or storage of the L nibbles $\boldsymbol{u}_i$. Then the halting condition becomes **until** $((s \geq 2^{n-1}) \text{or} (p \leq po))$. Alternatively, a decoder may detect these malformations and report an un-decodable codeword or/and may halt the recursion after L steps.

TABLE I
ENCODING TABLE FOR A RATE 8/9 CODE, $q = 3$ AND $L = 3$.

$\boldsymbol{f}$	$p\boldsymbol{u}_1$	$\boldsymbol{u}_2$	$\boldsymbol{u}_3$
00	$1a_1a_2$	$a_3a_4a_5$	$a_6a_7a_8$
01	001	$a_3a_4a_5$	$1a_1a_2$
10	010	$1a_1a_2$	$a_6a_7a_8$
11	011	$1a_1a_2$	111

B. Simplified Alternative construction

The simplified encoder procedure to be presented here is slightly less complex than the general procedure discussed in the previous section, though at a cost of a slightly reduced code rate. We choose $w = 1$ and $L = q$, so that $n = q^2$, $q > 1$, and the code rate is $1 - 1/q^2$. In similar vein as in the general procedure, let $(a_1, \ldots, a_{n-1})$ denote the binary user data, which are partitioned into q nibbles, $\boldsymbol{u}_i$, $1 \leq i \leq q$.

Encoding routine
Input: The integer q, and the binary $(n-1)$-bit data word $(a_1, \ldots, a_{n-1})$. Define the $L = q$ q-bit nibbles $\boldsymbol{u}_1$ = $(1, a_1, \ldots, a_{q-1})$ and $\boldsymbol{u}_i$ = $(a_{(i-1)q}, \ldots, a_{iq-1})$, $2 \leq i \leq L$.
Output: List of q encoded nibbles $\boldsymbol{u}_i$, where $\text{dec}(\boldsymbol{u}_i) > 0$, $1 \leq i \leq q$.
Define the q-bit nibble $\boldsymbol{f} = (0, f_2, \ldots, f_q)$, where $f_i = 1$ if $\text{dec}(\boldsymbol{u}_i) = 0$ or $f_i = 0$ if $\text{dec}(\boldsymbol{u}_i) \neq 0$, $2 \leq i \leq q$, and let $s = \sum_2^q f_i$.
If $s = 0$ then transmit the codeword $\boldsymbol{u}_1\boldsymbol{u}_2 \ldots \boldsymbol{u}_q$.
If $s = 1$ and $f_l = 1$ then transmit the codeword $\boldsymbol{f}\boldsymbol{u}_2 \ldots \boldsymbol{u}_{l-1}$ $\boldsymbol{u}_1$ $\boldsymbol{u}_{l+1}$ $\ldots \boldsymbol{u}_q$, where $\boldsymbol{u}_1$ is substituted for the 'all-zero' nibble $\boldsymbol{u}_l$.
If $s > 1$ then the encoder substitutes $\boldsymbol{u}_1$ for the 'all-zero' nibble having the least index. The remaining 'all-zero' nibbles are set to 'all-one' (or any other allowed value). Substituting the 'all-zero' words with the 'all-one' word is an arbitrary choice, and dependent on the specific application; many variants are possible.

Example *3:* Let $q = 4$, and let the user data be '000 0001 0000 0000'. Append a '1', so that we obtain the intermediate word '1000 0001 0000 0000'. The nibbles at positions 3 and 4 are inadmissible, so that we obtain $\boldsymbol{f}$ =(0,1,1). Let $\boldsymbol{u}_1 = 0\boldsymbol{f}$ = '0011', $\boldsymbol{u}_3 = \boldsymbol{u}_1$ = '1000', and $\boldsymbol{u}_3$ = '1111'. Transmit: '0011 0001 1000 1111'.

Example *4:* We have worked out the encoding table, see Table I, for a rate 8/9, code with $q = 3$ and $L = 3$. For example, let the user data be '11 000 110'. We find $\boldsymbol{f} = (1, 0)$, and transmit '010 111 110'. The code shown above is essentially the same as the rate 16/17 code with eight bits interleaved uncoded data presented by Wijngaarden *et al.* [12].

III. RESULTS AND COMPARISONS WITH PRIOR ART CODES

In this subsection, we will discuss some practicalities, and we will compare the performance of newly constructed

TABLE II
SELECTED VALUES OF q AND L, AND MINIMUM VALUE OF k FOR BASE-8 AND 10 USER SYSTEMS.

q	L	$qL-1$	k
7	55	384	12
9	57	512	14
17	241	4096	18
7	43	300	12
9	89	800	15
11	91	1000	15
13	77	1000	15
19	279	5300	19

codes with that of prior art codes. The majority of maximum runlength block codes presented in the (patent) literature is characterized by a rate $R = 1-1/n$, where n is the codeword length. Virtually all known channel coding schemes for data storage or telecommunications are based on 8-bit (byte) ECC symbols, as they are historically the *de facto* standard. It has been anticipated that the use of 10-bit ECC symbols and thus new methods are required to achieve improved density and error propagation performance in the context of 10-bit ECC symbols. It would be useful if the new coding system could support a base of 8-bit and/or 10-bit systems. In this context, for a nibble replacement technique, the choice $q = 8$ is not attractive as then the codeword can support $8L-1$ user bits, which is neither a multiple of 8 nor 10. Also a combination of eight codewords that each support $8L-1$ bits does not lead to an attractive system. We therefore take a closer look at systems where $q \neq 8$.

Let $q = 9$. Then, a very attractive parameter choice is $L = 57$, since $qL-1 = 512$, which leads to a byte-oriented system of 64 bytes input data. After a simple trial and error, we simply find for $k = 13$, $w = 5$, so that $L \leq 51$. We also find $k = 14$, $w = 37$, and $L \leq 85$. So that we conclude that $k = 14$ is the smallest k for which we can construct an $(n-1) = 512$-bit nibble replacement system. After a few computer runs, we compiled Table II, which shows selected values of q and L, and the minimum value of k possible with this parameter choice for user systems with a base-8 and 10.

IV. ERROR PROPAGATION

In this section, we will report on simulations of experiments with error propagation. It is assumed that a binary source word, $\boldsymbol{b}$ is translated into a binary codeword $\boldsymbol{x}$ using a specified coding algorithm. Assume that during transmission of $\boldsymbol{x}$ a single error is made, i.e. we receive $\boldsymbol{x}'$, $d_H(\boldsymbol{x}, \boldsymbol{x}') = 1$, where $d_H(\boldsymbol{x}, \boldsymbol{y})$ denotes Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. Decoding of $\boldsymbol{x}'$ will result in the word $\boldsymbol{b}$'. In particular we will investigate the *error propagation*, i.e. $d_H(\boldsymbol{b}, \boldsymbol{b}')$.

The computation of a major part of the error propagation of the nibble replacement technique is amenable to analysis. The probability, p_h that h of the $L-1$ nibbles are replaced during encoding equals (binomial distribution)

$$p_h = \binom{L-1}{h} (1-p_1)^{L-h-1} p_1^h, \tag{2}$$

where $p_1 = w/2^q$. As $L \approx 2^{q-1}/w$, we have $Lp_1 \approx 1/2$, so that we may approximate the above binomial distribution with a Poisson distribution

$$p_h \approx \frac{1}{\sqrt{e}} (Lp_1)^h \frac{1}{h!}.$$

We see that the source word is in $p_0 = 1/\sqrt{e} \approx 61$ percent of the time transmitted without any modification. Assume h nibbles are replaced during encoding, then a single decoding error will result, unless the pivot bit or the modified nibbles are received in error. Thus, in case h nibbles are replaced, the probability of a single error equals

$$p_h(qL-1-hq)/(qL).$$

Averaging, using (2), yields the probability, p_{single}, that there is a single decoding error, i.e.,

$$p_{single} = \frac{1}{qL} \sum_{h=0}^{L-1} p_h(qL-1-hq) = \frac{qL-1-q/2}{qL}. \tag{3}$$

We conclude from the above, that in the nibble replacement technique the probability of single errors in the decoded codeword grows to unity with growing codeword length, and that, as a result, error propagation is diminishing with increasing codeword length, and we conclude that massive error propagation is essentially absent in the nibble replacement technique.

For k-constrained codes it is possible to reduce the error propagation even further more by observing that the encoder is 'overactive' in replacing nibbles. The encoder as described above always replaces the w disallowed nibbles, while, clearly it only needs to replace a nibble if that nibble in conjunction with the previous and/or next nibble violates the k constraint. That is, if the sum of the 'zero'-runlengths of the tail and nose of the previous and upcoming nibble is greater than k. Assume $w = 1$, then the probability, p_1', that the juxtaposition of two consecutive nibbles violates the $k = 2(q-1)$ constraint equals

$$p_1' = \frac{q+1}{2^{2q}}.$$

Then the probability, p_h', that h of the $L-1$ nibbles are replaced during encoding equals

$$p_h' = \binom{L-1}{h} (1-p_1')^{L-1-h} {p_1'}^h,$$

The probability of retrieving single errors equals

$$p_{single}' = \frac{qL-1-q(q+1)2^{-q-1}}{qL}. \tag{4}$$

We conclude from the above that, on the average, single errors are in the majority, and that for $q > 8$ multiple errors are essentially only made if the pivot bit is received in error. Figure 1 shows simulation results of the error propagation in case the pivot bit is inverted. We observe that the propagation histograms are bell shaped showing a peak at approximately q errors.

The prior art coding technique based on enumerative coding shows a completely different error propagation behavior. Figure 2 displays the error propagation of a k-constrained code (k=6) based on Kautz-type enumeration [2], which allows code constructions with very high efficiency [13]. The diagram

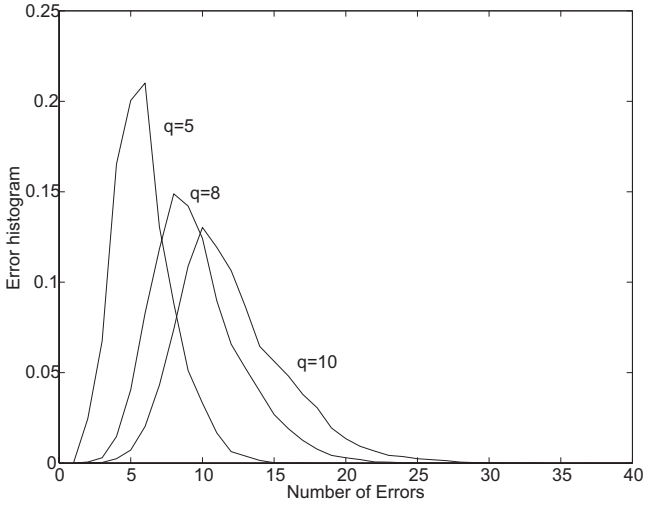

Fig. 1. Histogram of error propagation of the nibble replacement technique for $w = 1$, $L = 2^{q-1}$, $q = 5$, $q = 8$, and $q = 10$ versus the number of bit errors in the decoded word in case the pivot bit is inverted.

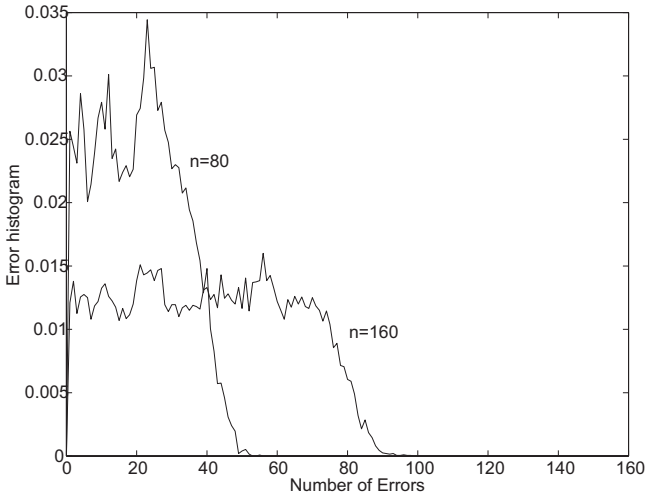

Fig. 2. Histograms of error propagation of Kautz-type enumerative coding for $k = 6$, $n = 80$ and $n = 160$. The average number of errors is around 20 and 40, respectively.

shows a typical behavior of enumerative-type systems: up to around $n/2$ the histogram is constant, and rapidly decays for larger values. The average error propagation increases with growing codeword length, and equals around $n/4$.

V. Conclusions

We have presented a new scheme, called nibble replacement technique, for generating k-constrained sequences by avoiding, w, predefined disallowed q-bit nibbles. The nibble replacement technique removes all disallowed nibbles and replaces them by allowed nibbles. The position and original value of the replaced nibbles can be uniquely decoded by the receiver, and the replacements made can be unmade. We have presented results of error propagation based on computer simulations.

References

[1] K.A.S. Immink, *Codes for Mass Data Storage Systems,* Second Edition, ISBN 90-74249-27-2, Shannon Foundation Publishers, Eindhoven, Netherlands, 2004.

[2] W.H. Kautz, 'Fibonacci Codes for Synchronization Control', *IEEE Trans. Inform. Theory,* vol. IT-11, pp. 284-292, 1965.

[3] A.J. de Lind van Wijngaarden and K.A.S. Immink, 'Construction of Constrained Codes using Sequence Replacement Techniques', *IEEE Journal on Selected Areas of Communications,* 2010.

[4] K.D. Fisher and J. Fitzpatrick, 'Rate 24/25 Modulation Code for PRML Recording Channels', US Patent 5,757,294, May 1998.

[5] J.L. Sonntag, 'Apparatus and Method for Increasing Density of Run-length Limited Codes without Increasing Error Propagation', US Patent 5,604,497, Feb. 1997.

[6] P. McEwen, B. Zafar, and K. Fitzpatrick, 'High Rate Runlength Limited Codes for 8-bit ECC Symbols', US Patent 6,201,485, March 2001.

[7] P. McEwen, K. Fitzpatrick, and B. Zafar, 'High Rate Runlength Limited Codes for 10-bit ECC Symbols', US Patent 6,259,384, July 2001.

[8] M.A. McClellan, 'Runlength-limited Code and Method', US Patent 6,285,302, Sept. 2001.

[9] A.J.M. Denissen and L.M.G.M. Tolhuizen, 'Conversion Arrangement for a Magnetic Recording/Reproducing Apparatus of the D-type', US Patent 5,644,582, July 1997.

[10] G.L. Silvus and K.D. Anderson, 'Error Correction Coding Utilizing Numerical Base Conversion for Modulation Coding', US Patent 6,959,412, Oct. 2005

[11] K.A.S. Immink, 'High-Rate Maximum Runlength Constrained Coding Schemes Using Base Conversion', 2010 International Symposium on Information Theory and its Applications (ISITA), Taichung, Taiwan, Oct. 17-20, 2010.

[12] K.A.S. Immink and A.J. de Lind van Wijngaarden, 'Simple high-rate constrained codes', *Electronics Letters,* vol. 32, no. 20, pp. 1877, Sept 1996

[13] K.A.S. Immink, 'A Practical Method for Approaching the Channel Capacity of Constrained Channels', *IEEE Trans. Inform. Theory,* vol. IT-43, no. 5, pp. 1389-1399, Sept. 1997.