

# High-Rate Maximum Runlength Constrained Coding Schemes Using Base Conversion

Kees A. Schouhamer Immink

Turing Machines Inc.

Willemskade 15b-d, 3016 DK Rotterdam The Netherlands

Email: immink@turing-machines.com

**Summary** - We will study simple and systematic constructions of high-rate binary maximum runlength constrained codes, which are based on base conversion, where specific subsequences are disallowed. We will compare the error propagation performance of base-change codes with that of prior art codes.

**Key words:** magnetic recording, hard disk drives, channel capacity, constrained code, runlength-limited code, RLL sequence, maximum runlength constrained code, MTR.

## 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 such *runlength-limited* codes have been collected in [1]. 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 and 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]. Though the rate of these  $k$ -constrained codes is very close to capacity, there is always a desideratum to explore alternative methods that have other attractive properties.

Denissen and Tolhuizen presented a systematic coding method, where the codeword consists of  $L$   $q$ -bit nibbles, where  $w$ ,  $0 < w < 2^q$ , specific  $q$ -bit nibbles are removed from the

repertoire of  $2^q$  possible nibbles using Galois Field arithmetic  $GF(2^q)$  [4]. The code requires one additional (redundant)  $q$ -bit nibble. The authors showed that 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 inadmissible nibbles. The code was proposed to be able to use the inadmissible nibble(s) as synchronization markers, but, clearly, this method can also be used to generate a constrained sequence, for example, a maximum runlength constrained sequence. In case, for example, we bar the 'all-zero' nibble, each  $q$ -bit nibble generated has at least one 'one', so that the maximum zero run length of the cascaded sequence of  $q$ -bit nibbles equals  $k = 2(q - 1)$ . By judiciously disallowing a plurality of nibbles with large 'zero' runlengths at the tail or nose, it is possible to guarantee a smaller maximum run length. Other (multiple) constraints can easily be accommodated.

In a recently granted US patent, Silvus and Anderson [5] presented a simple alternative to the above coding method, where, as above, the codeword consists of  $L$   $q$ -bit nibbles, where  $w > 0$  vexatious nibbles are inadmissible. The binary input word is converted into an integer in a base  $M = 2^q - w$  using a base translating routine, where the base- $M$  integers, in turn, are translated, using a look-up table of size  $M$ , into the allowed binary  $q$ -bit nibbles. In the preferred embodiment described in their patent, the number of deleted nibbles,  $w = 2^q - p^\alpha$ ,  $p$  is a prime number, and  $\alpha$ , a positive number, is judiciously chosen. This specific choice is advantageous as it makes it possible to use a second base,  $M = p^\alpha$ , that can accommodate a Reed-Solomon ECC based on Galois Fields [6]. Note that the method presented by Denissen and Tolhuizen and Silvus and Anderson have the virtue that *any* unwanted words can be deleted without change of the basic algorithm so that multiple (runlength) constraints can be accommodated.

Silvus and Anderson's base-change coding method is conceptually speaking very simple and attractive, but to the best of the author's knowledge, a quantitative analysis of this method has not been presented, and this paper intends to fill that gap. The structure of the paper is as follows. We start, in Section II, with the presentation of properties of a base-change coding technique [5], where the binary input word is translated into a non-negative integer represented by symbols in an  $M$ -ary,  $M = 2^q - w$ , alphabet. The  $M$ -ary symbols are, in turn, translated into the  $M$  admissible  $q$ -bit nibbles using a (small)

<sup>0</sup>This project was supported by grant *Theory and Practice of Coding and Cryptography*, Award Number: NRF-CRP2-2007-03

look-up table. In Section III, we will show that the hardware required for the  $M$ -base arithmetic can be prohibitively large for large values of the codeword length  $n$ , and thus larger values of the code efficiency. The code may suffer from mass error propagation, and therefore, in Section IV, we will present results of simulations showing the error propagation performance difference between the new codes and prior art codes. Section V concludes the paper.

## II. REMOVING UNWANTED NIBBLES USING NUMERICAL BASE CONVERSION

We assume that the  $n$ -bit codeword consisting of binary symbols can be partitioned into  $L$   $q$ -bit nibbles, so that,  $n = Lq$ . The aim of the encoding procedure is to translate arbitrary user (source) data into an  $n$ -bit codeword consisting of  $L$   $q$ -bit nibbles, where  $w > 0$  specific predetermined nibbles are forbidden. Let  $M = 2^q - w$  be the number of allowed  $q$ -bit nibbles. The encoding of a user word of arbitrary binary data into a multiple,  $L$ , of  $q$ -bit nibbles can simply be implemented by converting the 2-base input word into an  $M$ -base word [5]. Once we have obtained the  $M$ -base word, the  $M$ -ary symbols are translated into the corresponding  $2^q - w$  admissible  $q$ -bit nibbles using a  $q$ -bit look-up table. Routines for efficiently converting from a first base into a second base have been published in the literature. We will now take a look at the rate of such base-conversion codes. The number of admissible codewords is

$$(2^q - w)^L.$$

The maximum rate,  $R_0$ , of a nibble removing code is

$$R_0 = \frac{\log_2(2^q - w)}{q} \approx 1 - \frac{1}{\ln(2)} \frac{w2^{-q}}{q}, \quad w \ll 2^q. \quad (1)$$

For a rate  $(n-1)/n$ , block code the parameters must satisfy the inequality

$$(2^q - w)^L \geq 2^{qL-1}, \quad (2)$$

or

$$L \leq \left\lfloor \frac{1}{q - \log_2(2^q - w)} \right\rfloor \approx \left\lfloor \ln(2) \frac{2^q}{w} \right\rfloor. \quad (3)$$

In the next subsection, we will focus on the generation of maximum runlength limited codes.

### A. Maximum runlength limited codes

Assume, in the above scheme, that the 'all-zero'  $q$ -bit nibble is forbidden, so that each nibble contains at least one 'one'. Then, as a result, the maximum 'zero' run length of a series of cascaded  $q$ -bit nibbles generated by the above scheme is simply

$$k = 2(q-1).$$

Table I shows results of computations for  $w=1$ , where  $\hat{L}$  denotes the largest integer that satisfies inequality (3), and  $\hat{n} = q\hat{L}$ . The (maximum) rate of this simple base-conversion code is

$$R_0 \approx 1 - \frac{1}{(k+2)\ln(2)} 2^{-k/2}, \quad k \gg 1,$$

TABLE I  
MAXIMUM CODEWORD LENGTH  $\hat{n} = q\hat{L}$  (BITS) FOR WHICH A RATE  $1 - 1/\hat{n}$ , MULTI-NIBBLE CODE CAN BE CONSTRUCTED AS A FUNCTION OF NIBBLE SIZE  $q$ . THE ALL-ZERO WORD IS SUPPRESSED.

$q$	$\hat{L}$	$\hat{n} = q\hat{L}$	$k = 2(q-1)$
2	2	4	2
3	5	15	4
4	10	40	6
5	21	105	8
6	44	264	10
7	88	616	12
8	177	1416	14

which, for large  $k$ , is a far cry from the capacity,  $C(k)$ , of  $k$ -constrained sequences [1]

$$C(k) \simeq 1 - \frac{1}{4\ln 2} 2^{-k}, \quad k \gg 1. \quad (4)$$

In the above scheme for generating  $k$ -constrained codes, only one inadmissible nibble,  $w = 1$ , the 'all-zero' nibble, was removed. In a straightforward way, we can generalize the above concept for generating sequences with a maximum runlength smaller than  $2(q-1)$  by judiciously barring more nibbles. For a general setting, we first define an  $n$ -bit  $klr$ -sequence. A  $klr$ -sequence is a  $k$ -constrained sequence that satisfies two additional constraints:

- $l$  constraint - the number of consecutive leading 'zero's of the sequence, that is, the number of 'zeros' preceding the first 'one', is at most  $l$ .
- $r$  constraint - the number of consecutive trailing 'zero's of the sequence, that is, the number of 'zero's' succeeding the last 'one', is at most  $r$ .

The number of  $klr$ -sequences of given length  $q$  can be found by recursion [1]. For  $k = l + r \geq q - 2$ , we simply have

$$w = 2^{q-1}(2^{-l} + 2^{-r}) - 1. \quad (5)$$

In case the smallest words in the lexicographic order are made inadmissible, we have  $r = q - 1$ . Then by removing the

$$w = 2^{q-1-l} \quad (6)$$

lowest ranking nibbles from the repertoire, we can ensure a  $k$ -constraint, where  $k = q - 1 + l$ . Combining (1) and (6) yields a simple approximation of the rate of the above base conversion code

$$1 - \frac{1}{4\ln 2} \frac{2^q}{q} 2^{-k}.$$

We conclude that the redundancy of this base-conversion code is a factor of  $\frac{2^q}{q}$  larger than the least redundancy possible (see (4)).

Disallowing the  $w$  smallest words in the lexicographic order has the advantage that an extra look-up table for translating the set of integers in the  $M$ -base representation into the set of allowed  $q$ -bit nibbles is not required. We can, however, design more efficient codes when we remove nibbles with both a large number of leading and trailing zeros. To that end, we constrain

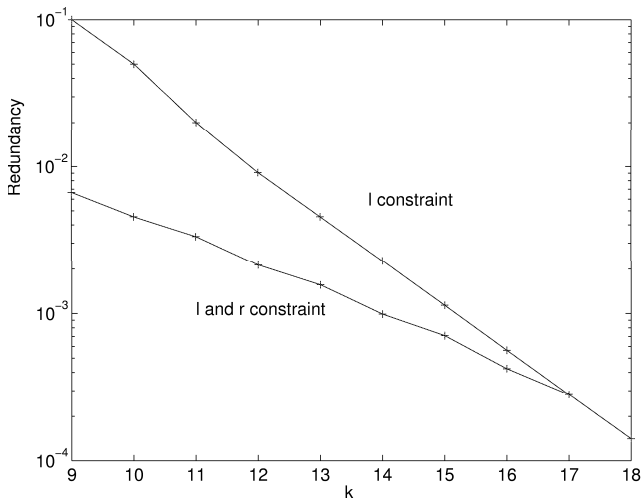


Fig. 1. Redundancy of base-conversion codes with  $l$  and both  $l$  and  $r$  constraints versus  $k$  for  $q = 10$ .

both the leading and trailing runlength, and let  $l = k - r$ . Substitution of  $l = k - r$  in (5) yields

$$w = 2^{q-1}(2^{r-k} + 2^{-r}) - 1 \\ = 2^{q-k/2-1}(2^{r-k/2} + 2^{-r+k/2}) - 1, \quad k \geq q - 2. \quad (7)$$

For the optimum, i.e. with the highest rate, base-conversion code of given  $k$ , we allow those nibbles that satisfy  $l = \lfloor k/2 \rfloor$  and  $r = k - l$ . In case  $k$  even, we choose  $l = r = k/2$ , then we simple have

$$w = 2^{q-k/2} - 1.$$

Then we simply find, for even  $k$ , that the rate of an optimum base-conversion code, equals

$$1 - \frac{1}{\ln(2)} \frac{2^{q-k/2} - 1}{q2^q}, \quad q - 2 \leq k \leq 2(q - 1).$$

For  $k$  odd we can write down a similar expression, which is slightly more involved and therefore omitted. Figure 1 shows the redundancy of the above code constructions versus  $k$  for  $q = 10$  with a) an  $l$  constraint only and b) both an  $l$  and  $r$  constraint. As we may notice, a constraint on both the number of leading and trailing zeros leads to more efficient coding than having a constraint on the number of leading zeros alone. It can easily be verified that for  $k = 2(q-1)$  and  $k = 2(q-1) - 1$  both codes have the same redundancy.

### III. COMPLEXITY CONSIDERATIONS

In this subsection, we will take a closer look at the complexity of an encoder and decoder which are based on base-conversion coding. With the  $(n - 1)$ -bit (binary) user word,  $(a_1, \dots, a_{n-1})$ , as an input, the encoder converts the binary input into an  $M$ -base integer, that is, it generates a vector of  $L$   $M$ -base integers  $c_i \in \{0, \dots, M - 1\}$  that satisfy the relation

$$\sum_{i=1}^{n-1} a_i 2^{i-1} = \sum_{i=1}^L c_i M^{i-1}.$$

Base conversion is well known, and is therefore not considered here, see for example [7]. The integers  $c_i$  are translated into  $q$ -bit nibbles, that are transmitted or stored. The conversion of  $c_i$  into  $q$ -bit nibble is accomplished using a look-up table or by regular Kautz enumeration. At the decoder's site the received  $q$ -bit nibbles are translated into the integers  $\hat{c}_i$ . The decoder performs the summation  $\sum_{i=1}^L \hat{c}_i M^{i-1}$ . The outcome of this straightforward accumulate-summation routine is the restored binary user word. It must be appreciated that all operations are performed in binary, and also the coefficients  $c_i$  and powers of  $M$  are represented by binary words. Thus, we require an  $n$ -bit adder, an  $n$ -bit multiplier plus storage for  $L$  integers of size at most  $n$  bits. In a traditional Kautz-type enumeration the decoder computes the inner product  $\sum_{i=1}^n r_i w_i$ , where  $r_i \in \{0, 1\}$  are the  $n$  binary integers of the received codeword, and  $w_i$  (positive integers) are weighting coefficients. We conclude that the hardware requirements of the base-change encoder, both of encoder and decoder, does not significantly differ from the hardware requirements of the conventional Kautz enumeration.

### IV. ERROR PROPAGATION

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

In order to compare the above results, we have computed the error propagation of two alternative coding schemes. Figure 2 shows results of computer simulations of error propagation in the base-conversion coding scheme presented in Section II. The diagram shows a typical behavior that we have observed frequently in enumerative coding 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  $n/4$ . Figure 3 displays the error propagation of a  $k$ -constrained code ( $k=6$ ) based on Kautz-type enumeration [2]. With Kautz-type enumerative coding it is possible to construct codes with very high efficiency [8]. The diagram 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  $n/4$ . Figure 4 displays the error propagation of a rate  $1 - 1/n$  sequence replacement scheme proposed by Wijngaarden and Immink [3]. The rate of this code is higher than that of the nibble replacement method described here, but less than that of the Kautz-type enumerative coding scheme. In a similar vein as in the nibble replacement method, the encoder of the sequence replacement method adds one pivot bit, which indicates that the input word has been modified or not. The diagram shows error propagation results in case the pivot bit

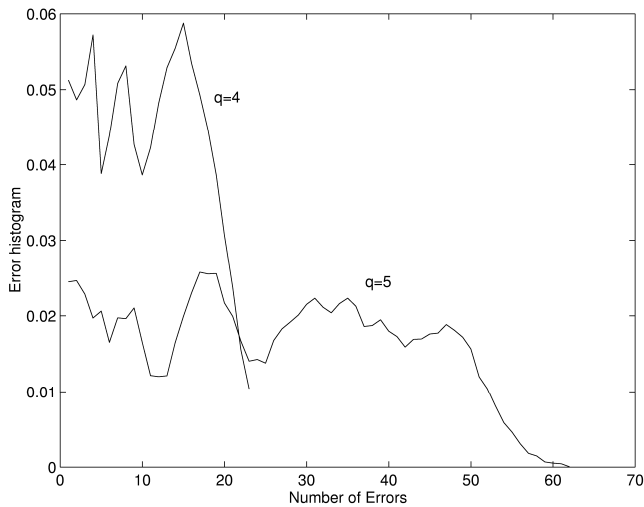


Fig. 2. Histogram of error propagation of base-conversion coding for  $k = 8$ ,  $q = 4$  ( $n = 40$ ) and  $q = 5$  ( $n = 105$ ) and  $w = 1$ .

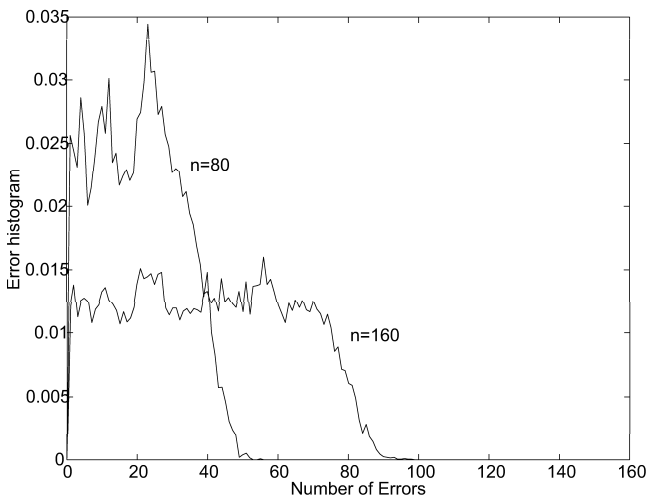


Fig. 3. Histograms of error propagation of Kautz-type enumerative coding for  $k = 6$ ,  $n = 80$  and  $n = 160$ . The average number of errors is 20 and 40, respectively.

was inverted. We observe that the error propagation is massive, and increases with codeword length.

## V. CONCLUSIONS

We have studied a coding scheme presented by Silvas and Anderson, called base-change technique, for generating  $k$ -constrained sequences. In a base-change scheme,  $w > 0$ , predefined disallowed  $q$ -bit nibbles are avoided. The base-change technique removes all disallowed nibbles and replaces them by allowed nibbles. The redundancy of the base-change technique is higher than that of the conventional enumeration technique. The error propagation of both schemes are essentially the same. The base-change method by Silvas and Anderson offers an alternative to other prior art methods, but it does not, as far we have investigated, offer any significant advantages over the prior art methods.

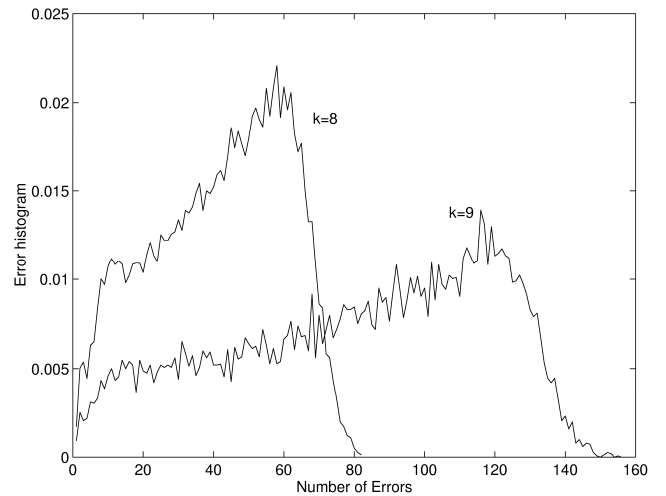


Fig. 4. Histograms of error propagation of Wijngaarden-Immink-type sequence replacement coding for  $k = 8$  ( $n = 137$ ) and  $k = 9$  ( $n = 266$ ) in case the pivot bit is inverted.

## 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] A.J.M. Denissen and I.M.G.M. Tolhuizen, 'Conversion Arrangement for a Magnetic Recording/Reproducing Apparatus of the D-type', US Patent 5,644,582, July 1997.
- [5] G.L. Silvas and K.D. Anderson, 'Error Correction Coding Utilizing Numerical Base Conversion for Modulation Coding', US Patent 6,959,412, Oct. 2005.
- [6] S.B. Wicker and V.K. Bhargava, Eds, *Reed-Solomon Codes and Their Applications*, IEEE Press, 1994.
- [7] D.E. Knuth, 'The Art of Computer Programming', Addison Wesley, 3rd Edition, 1997.
- [8] 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.