Weakly-Constrained Codes for Suppression of Patterning Effects in Digital Communications

Alexander Shafarenko, *Senior Member, IEEE*, Anton Skidin, and Sergei K. Turitsyn, *Member, IEEE*

***Abstract*—We propose weakly-constrained stream and block codes with tunable pattern-dependent statistics and demonstrate that the block code capacity at large block sizes is close to the the prediction obtained from a simple Markov model published earlier. We demonstrate the feasibility of the code by presenting original encoding and decoding algorithms with a complexity log-linear in the block size and with modest table memory requirements. We also show that when such codes are used for mitigation of patterning effects in optical fibre communications, a gain of about 0.5dB is possible under realistic conditions, at the expense of small redundancy (≈10%).**

***Index Terms*—Patterning effects, weakly-constrained codes, coding theory, pre-encoding.**

I. Introduction

THE patterning effect due to inter-symbol interference (ISI) manifests itself in digital communication as dependence of the transmission result for an information bit on the surrounding pattern, i.e. the neighbouring bits. ISI imposes one of the most severe restrictions on the speed of data transmission and can be due to physical mechanisms of widely varying origins. For instance, in fibre optic digital communication, pattern dependence of errors can be caused by the gain saturation of a semiconductor optical amplifier (see, e.g., [1]) or by resonance interactions between pulses in bit-overlapping transmission regimes [2], [3]. Note that the current trend of ever-increasing channel rate makes it necessary to use increasingly shorter carrier pulses thus also increasing the effects of dispersive broadening of the signal and consequently bit-overlapping. Therefore the negative impact of the patterning effects may become a very serious factor in future optical systems.

One well-researched example of ISI is optical fibre communication at high bit rates limited by intra-channel four-wave-mixing (ICFWM) [2], [3], which generates "ghost" pulses. Under such conditions the main contribution to the bit error rate comes from the ghosts pulses that appear in logical-zero time slots surrounded by symmetric patterns of logical-ones [4]. Mitigation of intra-channel nonlinear effects by channel coding has been proposed in [5], [6]. Suppression of the ICFWM effects by employing a pre-encoding was studied in [4], [7]-[9]. Direct modeling of bit-error-rate in high-speed wave-division-multiplexing optical communications, see [10], [11], has shown the utility of skewed statistics pre-encoding of the type discussed in [7]. In the present paper we will extend the analysis begun in [7] with consideration and evaluation of specific coding schemes.

It should be noted that practical optical transmission lines have to rely on forward-error correction techniques for the provision of a low-BER communication channel. When the BER is small by itself, the FEC can cope with the errors due to both the channel noise and patterning well enough not to require any additional measures. However, as the patterning effects grow stronger, at some point the FEC scheme starts to deteriorate rapidly and it is at that point that error prevention becomes essential in maintaining the low-BER regime. The solution that we proposed in [7], and which we continue to study here, is to use a weakly constrained code to adjust the incidence of undesirable patterns down in order to reduce the part of the BER that is due to patterning and, consequently, bring the combined BER back under the FEC break-down threshold. Since the FEC threshold is usually very sharply defined, the most economical pre-encoding scheme ought to be tuneable: any extra redundancy below the threshold is more effective if utilised by the FEC itself. Weakly constrained codes are codes that reduce the incidence of patterns of a certain kind; the amount of reduction is directly linked with the code redundancy and is controllable by a parameter that can be varied almost continuously [7]. This makes them ideally suited to the task of controllable pattern reduction for the purposes of FEC.

We will present specific block and stream codes for weakly-constrained coding and will demonstrate that for some of these codes, which can be realized in practice, the redundancy at a given level of pattern elimination is close to the Markov model obtained in [7]. Finally, for our block codes we propose a log-linear complexity encoding/decoding algorithm and evaluate the code gain under various strengths of the patterning effect.

II. From "Ghost" Pulses to Pattern Reduction

Denote as $b_1 b_2 ... b_n$ the transmitted sequence of bits. If $b_k = b_l = b_m = 1$ and the bit $b_{k+l-m} = 0$, then the "ghost" pulse effect establishes a false 1 in the position $k + l - m$ upon reception (see [4], [12] for more details). The patterns 1101, 1011, and 11011 are especially prone to "ghost" pulses. For example, if $b_p b_{p+1} b_{p+2} b_{p+3} = 1101$, then $k = p$, $l = p + 3$, $m = p + 1$ and the bit $b_{k+l-m} = b_{p+2} = 0$ can be received as 1. Since the capacity of a "ghost"-constrained channel is zero (see [12] for the proof), the complete suppression of the effect is impossible. However, for any finite sequence complete

Paper approved by O. Milenkovic, the Editor for Coding Theory and Applications of the IEEE Communications Society. Manuscript received November 19, 2008; revised January 17, 2010.

A. Shafarenko is with the School of Computer Science, University of Hertfordshire, College Lane, Hatfield, AL10 9AB, UK (e-mail: A.Shafarenko@herts.ac.uk).

A. Skidin is with the Institute of Computational Technologies SB RAS, Novosibirsk, 6 Acad. Lavrentjev Avenue, 630090, Russia (e-mail: ask@skidin.org).

S. K. Turitsyn is with the Aston University, Photonics Research Group, Birmingham B4 7ET, UK (e-mail: s.k.turitsyn@aston.ac.uk).

Digital Object Identifier 10.1109/TCOMM.2010.080310.100026

elimination of "ghost" pulses can be achieved by constrained coding (see [12]).

It is noteworthy that some "resonance" patterns are more hazardous than others. It is proven that the 101 pattern, and, of course, patterns containing it, such as 11011, 1101 etc., make the main contribution to the channel bit error rate. This was observed experimentally and described in [4], [10] and [13]. The papers [10] and [13] describe a channel simulation and report some error statistics obtained in it.

In contrast to the complete elimination of "ghost" pulses, which is extremely expensive in terms of the channel capacity[1], a mere reduction of 101 can be achieved considerably more cheaply. Indeed, by using our proposed codes 50% of the undesirable pattern occurrences can be removed at 6% redundancy. Such a major reduction in strong-patterning situations leads to a similar reduction in the BER, which in turn can return the FEC procedure to the normal, subcritical, regime.

Could the patterns 101 be reduced or eliminated using standard coding techniques? For example, the codes RLL(d, k) with $d \geq 2$ are suitable for such elimination as they ensure that at least two 0s are present between any pair of 1s . Indeed, the codes $RLL(1, \infty)$ and $RLL(2, \infty)$ have been proposed for the worst pattern suppression (see [14]). Unfortunately, RLL-codes have very small capacities ($C(2, \infty) = 0.5515$, $C(1, \infty) = 0.6942$) and are therefore rather expensive for long-haul optical transmission systems. Other codes were suggested as a cheaper alternative, following exactly the same logic as ours (small patterns associated with resonances instead of all possible resonances), for example, a simple code for avoiding the pattern "11011" is described in paper [4], with a redundancy of only 20%; it completely eliminates the 11011 pattern. We have been guided by the same level of acceptable redundancy (20% or less) but our purpose has been rather different: as mentioned in section I, we wish to propose a code with tuneable redundancy that eliminates just enough occurrences of the undesirable pattern to bring the BER down to a level that FEC can process. For long patterns that have a small rate of occurrence, e.g. 11011, which would occur in about 3% of quintuplets in a random packet, tuning could be compatible with complete elimination: indeed, one could control the number of different undesirable patterns to eliminate. However, if one were to completely eliminate short patterns, in our case triplets, those contribute from about 15% to 30% redundancy each, which makes fine tuning difficult.

The code presented below achieves near continuous tuneability using a version of the enumeration method [17]. We must emphasise that the code in question can **not** be regarded as an instance of the general RLL code, nor of Vasic's graph model[4]. Since it reduces, rather than eliminates, the incidence of a pattern, it is a weakly-constrained code rather than a constrained code.

The next section will lay theoretical foundations of our approach.

[1]the best redundancy we have found in [12] was about 30%, but they also quote redundancies up to 85%, see [12]p.70

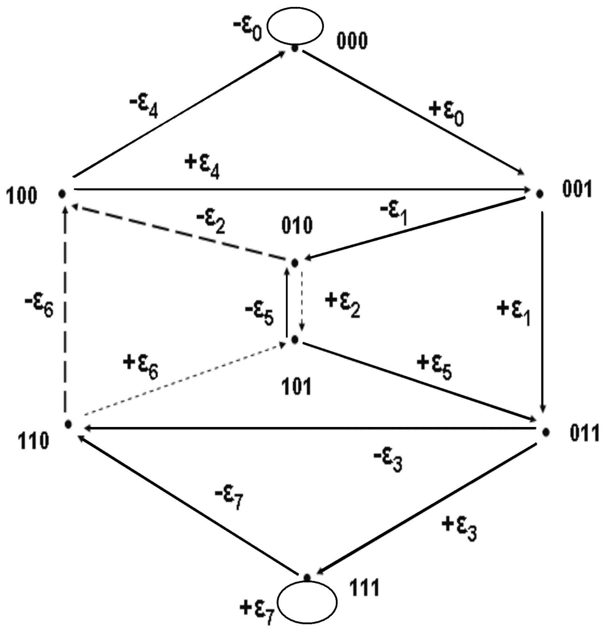

Fig. 1. The transition diagram with eight skew-parameters.

III. Skewed Encoding and its Associated Redundancy

The BER of a received message is given by the formula $BER = \sum_{k=0}^{7} P_k Q_k$, where P_k is the average frequency of the triplet k in the input bit string, and Q_k is the error probability for the central digit in the triplet[2]. The message is assumed to be long enough for neglecting the triplets at its ends that involve bits outside the message. It is an uneven distribution of errors Q_k caused by the patterning effects that makes it possible to reduce the error rate by reducing the frequency of some triplets. All that is required is a mapping of the source message onto code words, which is called skewed pre-encoding [8].

It is convenient to illustrate the skewed pre-encoding principle using the Markov chain in Fig. 1. The vertices of the graph correspond to the state of the process, which consists of the three most recently transmitted digits including the current one. The transmission of the next digit is depicted as the transition from the current state to the next, keeping the most recent two bits and adjoining to them a new one (either 0 or 1), with some probability that depends on the current state. All transitions in the diagram generally occur with a probability different from 1/2, which would be the case with a random bit stream without pre-encoding. The difference between the probabilities for the transitions from the same state k is $\varepsilon_k = T_{k,2k+1} - T_{k,2k}$, where $T_{k,2k} = (1 - \varepsilon_k)/2$ and $T_{k,2k+1} = (1 + \varepsilon_k)/2$, assuming modulo 8 indexing, which is a measure of the statistical skew of the encoder. As mentioned above, triplet 5 (pattern 101) was found to be the main cause of errors in some optical transmission systems. A reduction in the incidence of the triplet can be achieved by annulling all εs, except $\varepsilon_2 = \varepsilon_6 = -\varepsilon < 0$, see the dashed lines in the figure. The per-bit entropy of the transmitted signal $h =$

[2]We enumerate triplets according to their binary code, e.g. triplet 6 is 110.

TABLE I
EQUATIONS FOR P_k

$P_0 = \frac{1}{M_1+M_2K}$
$P_1 = \frac{1+\varepsilon_0}{1-\varepsilon_4}P_0$
$P_2 = \left(\frac{(\varepsilon_5-\varepsilon_1)(1+\varepsilon_0)}{(1+\varepsilon_5)(1-\varepsilon_4)} + \frac{(1-\varepsilon_5)(1-\varepsilon_7)}{(1+\varepsilon_5)(1+\varepsilon_3)}K\right)P_0$
$P_3 = P_6 = \frac{1-\varepsilon_7}{1+\varepsilon_3}KP_0$
$P_4 = P_1 = \frac{1+\varepsilon_0}{1-\varepsilon_4}P_0$
$P_5 = \left(\frac{(1+\varepsilon_2)(1+\varepsilon_0)}{(1-\varepsilon_2)(1-\varepsilon_4)} + \frac{(\varepsilon_6-\varepsilon_2)(1-\varepsilon_7)}{(1-\varepsilon_2)(1+\varepsilon_3)}K\right)P_0$
$P_7 = KP_0$
$M_1 = 1+2\frac{1+\varepsilon_0}{1-\varepsilon_4} + \frac{(1+\varepsilon_2)(1+\varepsilon_0)}{(1-\varepsilon_2)(1-\varepsilon_4)} + \frac{(\varepsilon_5-\varepsilon_1)(1+\varepsilon_0)}{(1+\varepsilon_5)(1-\varepsilon_4)}$
$M_2 = 1+2\frac{1-\varepsilon_7}{1+\varepsilon_3} + \frac{(1-\varepsilon_5)(1-\varepsilon_7)}{(1+\varepsilon_5)(1+\varepsilon_3)} + \frac{(\varepsilon_6-\varepsilon_2)(1-\varepsilon_7)}{(1-\varepsilon_2)(1+\varepsilon_3)}$
$K = \frac{(1+\varepsilon_0)(1+\varepsilon_3)(\varepsilon_2(\varepsilon_5-\varepsilon_1)+2+\varepsilon_1+\varepsilon_5)}{(1-\varepsilon_4)(1-\varepsilon_7)(\varepsilon_5(\varepsilon_2-\varepsilon_6)+2-\varepsilon_2-\varepsilon_6)}$

$\dfrac{3+\mathrm{g}(\varepsilon)+\varepsilon}{4+\varepsilon}$, and thus the code redundancy $R = \dfrac{1-\mathrm{g}(\varepsilon)}{4+\varepsilon}$, where $\mathrm{g}(\varepsilon) = -\dfrac{1+\varepsilon}{2}\log_2\dfrac{1+\varepsilon}{2} - \dfrac{1-\varepsilon}{2}\log_2\dfrac{1-\varepsilon}{2}$.

In [8] the probabilities P_k for arbitrary $\vec{\varepsilon} = (\varepsilon_0, \varepsilon_1, ..., \varepsilon_7)$ were found. They are shown in Table I. Ref [8] states that the minimum-redundancy code for a given 101 skew ties the parameters pairwise: $\varepsilon_0 = \varepsilon_4$, $\varepsilon_1 = \varepsilon_5$, $\varepsilon_2 = \varepsilon_6$, and $\varepsilon_3 = \varepsilon_7$. Consequently we shall assume these ties in the sequel.

IV. ZU LANGUAGE

Let us now consider a slightly different representation of a bit string. Imagine a language based on two sets of symbols: z_k representing a string of zeroes of size k and similarly u_k, a string of ones sized k, $k > 0$. Any bit string s of nonzero length represents the following structure:

$$s = \ldots UZUZUZ \ldots$$

where each Z stands for z_k with some k, generally different for different Z, and each U similarly for a u_k. The string can begin and/or end with either Z or U, but since the end-effects are not important for statistical calculations on a long string, we can limit ourselves to strings of UZ-pairs without loss of generality.

In an uncorrelated random bit string, the frequency of each of the symbols is $P(u_k) = P(z_k) = 2^{-k}$. It is easy to calculate the entropy of the string given those probabilities. Since the order of Zs and Us is fixed, the entropy is the sum of the entropies of the Z- and U-substrings, and furthermore, the sum of the entropies of individual Z and U symbols. Consequently,

$$H(U) = -\sum_{k=1}^{\infty} P(u_k)\log_2 P(u_k) = 2$$

and the average length in bits of a symbol u_k is

$$L(U) = \sum_{k=1}^{\infty} kP(u_k) = 2$$

and the same for symbols z_k. Thus the information density of the uncorrelated string is precisely 1 bit/digit as expected.

We are now prepared to analyse a 101-skewed code using this language, by introducing a skew similarly to the Markov model discussed earlier. Let us reduce the probability of z_1, which is equal to the frequency of the 101 triplet, and increase the probability of every other z_k, $k > 1$ to compensate:

$$P(z_k) = \varepsilon \cdot \theta(k-2)2^{-(k-1)} + (1-\varepsilon)2^{-k} .$$

Here $\theta(m) = 0$ for negative m and 1 otherwise. This distribution is correctly normalised for any ε. A direct calculation of the entropy gives

$$H(Z) = 1 + \varepsilon + \mathrm{g}(\varepsilon) ,$$

where $\mathrm{g}(\varepsilon)$ is exactly as defined earlier. The average length of one Z is

$$L(Z) = 2 + \varepsilon .$$

Remembering that Zs account for half of the string symbols and the other half has $H(U) = L(U) = 2$ the redundancy

$$R = 1 - \frac{H(Z)+H(U)}{L(Z)+L(U)} = \frac{1-\mathrm{g}(\varepsilon)}{4+\varepsilon}$$

exactly as suggested by the Markov model. The frequency of the 101 triplet in a long ZU string is the probability of the symbol z_1, $(1-\varepsilon)/2$, normalised by the length of the ZU pair, $4+\varepsilon$, which gives $(1-\varepsilon)/(8+2\varepsilon)$, again in full agreement with the Markov model.

It is notable that the ZU-language is a handy instrument for constrained code analysis. Other state frequencies can be reformulated in the ZU language very easily. For example, the frequency

$$P_7 = \frac{1}{L(ZU)}\sum_{k=3}^{\infty}(k-3)P(u_k) .$$

The symmetries inherent in the Markov solution, such as $P_1 = P_4$ can now be proven in the general case by observing that the pattern 100 can only be detected in the Uz_kU situation with $k \geq 2$ but then the same Uz_kU combination would deliver the 001 pattern. Similarly, $P_3 = P_6$.

The Markov model is still very useful because, as the aforementioned example of P_7 shows, a finite calculation in the Markov chain corresponds to an infinite summation over the ZU language, which may or may not be possible analytically. Nevertheless, unlike the Markov chain, the ZU approach empowers us to make the next step. Below we use the ZU-language for calculation of the bit-stuffing code properties.

V. WEAKLY-CONSTRAINED CODES FOR PRE-ENCODING

Initially, weakly-constrained codes were presented by K.A.S.Immink in [15] and were intended for magnetic recording systems. In contrast to constrained (strictly-constrained) codes these codes do not strictly observe the channel constraints, rather their codewords violate the channel constraints infrequently. Weak constraints are motivated by the fact that if the channel is not free of random errors due to the noise, codes that always observe the constraint have no advantage: it will be violated by errors anyway. Thanks to the method of weakly-constrained coding, very effective RLL-codes have been constructed (see [16]). Codes for reducing the frequency of triplet 101 are weakly-constrained also, since the degree of triplet elimination depends on the parameter ε ($0 \leq \varepsilon \leq 1$). This parameter is the "degree of weakeness" for our code: at

$\varepsilon = 1$ the code is strongly-constrained (all 101s are removed from the message), and at $\varepsilon = 0$ the code has no constraints whatsoever.

A. Bit stuffing

Let us now consider a practical code. After the example of bit stuffing which is well-known in digital recording technology, let us insert an extra 0 in a Z-symbol to make it longer and thus reduce the probability of the pattern 101. For this to be practical, we need to be able to reverse the effect of stuffing, i.e. we need to know that an extra 0 has been inserted in the received bit sequence so that we may remove it. This can be achieved by making the stuffing decision conditional on the preceding U-symbol. Here is our proposed code C_t:

$$C_t(u_k z_m) = \begin{cases} u_k z_m, & \text{if } k < t \\ u_k z_{m+1} & \text{if } k \geq t \end{cases} .$$

The inverse code is, obviously,

$$C_t^{-1}(u_k z_m) = \begin{cases} u_k z_m, & \text{if } k < t \\ u_k z_{m-1} & \text{if } k \geq t \end{cases} ,$$

and is undefined on any $u_k z_0$ combination with $k \geq t$, as such combinations are prevented by the direct code. Code C_t is controllable via the value of t: since the frequency of u_k falls exponentially with k, by choosing t large enough one can achieve the smallest desirable degree of stuffing. As a limiting case, code C_1 always stuffs an extra zero into a Z, thus completely eliminating the symbol z_1 and its corresponding pattern 101.[3] We conclude that the code C_t gives a good degree of control over the probability of the pattern 101 at large t.

Notice that the code we have described suffers from the same problem as convolution codes in FEC: it has the tendency to spread an error by making nonlocal coding decisions. Indeed the above code mapping is local only in the ZU representation as it maps a ZU pair on a ZU pair. However an error $1 \to 0$ disrupting a string of 1s within a single U will effectively insert an extra ZU pair. This means that a code of the type described above would require some support from an FEC code to neutralise those types of errors; however, when the skew-related effects are much stronger than random errors, the bit-stuffing code could be quite effective in combination with FEC.

Let us now evaluate the effect of bit stuffing on redundancy and the frequency of 101 as we did with the Markov model. It is easy to see that the probability of the UZ pair is as follows:

$$P(u_k z_m) = \begin{cases} 2^{-(k+m)}, & \text{if } k < t \\ 2^{-(k+m-1)} & \text{if } k \geq t, m \geq 2 \\ 0 & \text{if } k \geq t, m = 1 \end{cases} ,$$

since the stuffed symbol retains its frequency but becomes one bit longer. After a straightforward calculation we obtain the average length of the resulting UZ-pair:

$$L(C_t(UZ)) = 4 + 2^{-(t-1)} .$$

[3]The reader might think that at least C_1 is an instance of RLL, the more so that there is no tuning parameter left in the code. Still it is not the case: C_1 allows 1s to come in a series uninterrupted by 0s, while eliminating single 0s, and not multiple 0s, in the output, which is something quite impossible with RLL.

Unsurprisingly, the length is 5 when $t = 1$ as we always stuff one zero into the Z symbol, which corresponds to the 20% redundancy under the Markov chain with $\varepsilon = 1$. As for the entropy $H(C_t(UZ))$, it remains the same $H = 4$ since the code is reversible, hence the image under the code contains exactly the same information[4]. The output frequency $P(Uz_1)$ is

$$P(Uz_1) = \sum_{k=1}^{t-1} P(u_k)P(z_1) =$$
$$\sum_{k=1}^{t-1} 2^{-k-1} = \frac{1}{2}\left(1 - 2^{-(t-1)}\right)$$

which corresponds to the Markov model with $\varepsilon = 2^{-(t-1)}$. Note, however, that the entropy per bit for the practical code $h(\varepsilon) = 4/(4+\varepsilon)$ is different from that for the Markov chain, and so is redundancy: $R = \varepsilon/(4+\varepsilon)$. The redundancy curve is plotted in Fig. 2.

B. Weakly-constrained block codes

One way of dealing with the error propagation of a bit-stuffing code is to use finite blocks. The idea of a weakly-constrained block code is as follows.

Let W_m be a list of all length-m bit strings. Denote as $W_m(j)$ the jth entry of the list and as $L_{m,k}$ the number of length m bit strings with no more than k 101s. The initial members of W_m with indices $0 \leq j < L_{m,0}$ have no triplets 101, those with indices $L_{m,0} \leq j < L_{m,1}$ have exactly one 101 etc. Finally codewords with indices $L_{m,k_{max}-1} \leq j < L_{m,k_{max}} = 2^m$ contain k_{max} triplets 101. The k_{max} value can be found for each specific m by construction (for example $k_{max} = 2$ if $m = 5$, because 5-bit strings with three or more 101s do not exist).

Let us represent the source bitstream S as a sequence of small blocks s_i with the length n ($n \leq m$): $S = (s_1, s_2, ..., s_p)$. Denote as $D = (d_0, d_1, ..., d_p)$ the encoded message, where d_i is a length-m data block. Now the block code $B(m, n)$ converts s_i blocks into d_i blocks thus: $d_i = W_m(s_i)$, $i = 1..p$. This way only the codewords with indices less than 2^n are used in the coding process. For decoding we need a table $W_m^{-1}(j)$, $0 \leq j < 2^m$: $W_m^{-1}(q) = j$ if $W_m(j) = q$. Then the decoding process can be defined as $s_i = W_m^{-1}(d_i)$, $i = 1..p$. This code has the redundancy $R = (m - n)/n$.

Code example: Let us consider a code $B(9, 8)$ that maps 8-bit strings onto 9-bit strings. It can be verified that $k_{max} = 4$. The values of $L_{9,k}$ are given in Table II.

It is clear from Table II that B(9,8) codewords contain no more than one triplet 101, since, for this code, $0 \leq j < 2^n = 256$, $W_9(j)$ for $0 \leq j < 200$ does not contain 101, and $W_9(j)$ for $200 \leq j < 399$ contains exactly one 101.

The per-bit frequency of 101 in the output is approximately $\frac{1}{mL_{m,k}} \sum_{i=0}^{k} iG_{m,i}$. $G_{m,i}$ is the number of m-bit strings that contain exactly i triplets 101. This is an approximation that neglects the triplets formed at the junctures of neighbouring blocks. These depend on the frequency of certain bits at the

[4]this can also be established by direct calculation

TABLE II
$L_{9,k}$ VALUES FOR $B(9,8)$ WEAKLY-CONSTRAINED BLOCK CODE

k value	$L_{9,k}$
0	200
1	399
2	490
3	511
4	512

beginning and end of a block. Indeed, the combination 101 will arise around the starting bit of a block that starts with 01, provided that it follows a block ending with 1. Similarly, a 101 combination around the last bit of a block arises when the block ends with 10, and the next one starts with 1. The exact per-bit frequency of the 101 combinations is, consequently,

$$\frac{1}{mL_{m,k}}\sum_{i=0}^{k} iG_{m,i} + \frac{\phi_{e1}\phi_{b01}+\phi_{e10}\phi_{b1}}{m} .$$

Here ϕ_{b01},ϕ_{b1},ϕ_{e1}, and ϕ_{e10} are frequencies of the blocks that begin with 01, begin with 1, end with 1, and end with 10, respectively. Note that due to the assumed randomness of the source bit stream, the neighbouring patterns are uncorrelated, which justifies the use of products of frequencies in the above formula.

We have constructed block codes in table form for m=8, 16 and 28 by exhaustive search and computed all ϕ and $L_{m,k}$ for some small k. For comparability, Fig. 2 shows R vs effective ε, i.e. such at which P_5 of the Markov chain matches the value for the block code. Codes with $m = 8$ and $m = 16$ are also practical due to their small table sizes while the 28-bit code is less so as it requires about 1Gb of table space, but even a code as large as this does not approach the redundancy level of the Markov chain. For larger block codes the table method is impractical, and an effective encoding algorithm is required.

VI. Analysis of the Weakly-constrained Block Code

In the previous section we introduced and discussed the block codes constructed numerically in table form. This gave us a chance to compare the statistics of constrained block coding with the Markov model, if only numerically and approximately. In this section we will give an analytical view of the block-code statistics which shows further deviations from the idealised Markov model and explains and quantifies them.

A. Frequency of 101 triplets

Consider a block of m bits. Out of the 2^m bit strings there will be some $G_{m,0}$ strings that contain no triplet 101, $G_{m,1}$, strings that contain exactly one such triplet, etc. Let us determine the value of $G_{m,k}$ analytically.

Proposition 6.1: The following recurrence relation holds:

$$G_{m,k} = 2G_{m-1,k} - G_{m-2,k} + G_{m-3,k} + G_{m-2,k-1} - G_{m-3,k-1} , \qquad (1)$$

with the initial data
$G_{3,0} = 7, G_{3,1} = 1, G_{3,k} = 0$ for any $k > 1$;
$G_{4,0} = 12, G_{4,1} = 4, G_{4,k} = 0$ for any $k > 1$;
$G_{5,0} = 21, G_{5,1} = 10, G_{5,2} = 1, G_{5,k} = 0$ for any $k > 2$.

Proof: Denote as $S_{m,k}$ a set of bit strings of length m having exactly k 101 triplets, $|S_{m,k}| = G_{m,k}$, and denote as $S^a_{m,k} \subset S_{m,k}$ the part made up of strings with the opening segment a. Similarly introduce $G^a_{m,k} = |S^a_{m,k}|$.

Consider an arbitrary $(m-1)$-bit string and denote it as B. Attach the next bit b from the data stream to the beginning of the string and denote the resulting string as $b \triangleright B$. Assume that the data stream is statistically unbiased and uncorrelated: $P_{b=0} = P_{b=1} = 1/2$. Now determine the conditions under which $(b \triangleright B) \in S_{m,k}$:

If $b = 1$, then $(b \triangleright B) \in S_{m,k}$ in two disjoint cases:

1) $B \in S_{m-1,k}$ and $B \notin S^{01}_{m-1,k}$ (there are $G_{m-1,k} - G^{01}_{m-1,k}$ such strings);
2) $B \in S^{01}_{m-1,k-1}$ (there are $G^{01}_{m-1,k-1}$ such strings).

If $b_m = 0$, then obviously $(b \triangleright B) \in S_{m,k}$ if $B \in S_{m-1,k}$ (there are $G^0_{m,k} = G_{m-1,k}$ such strings). Since these cases are mutually exclusive, we have the following equation:

$$G_{m,k} = G^0_{m,k} + G^1_{m,k} = G_{m-1,k} + G^{01}_{m-1,k-1} + G_{m-1,k} - G^{01}_{m-1,k} = 2G_{m-1,k} + G^{01}_{m-1,k-1} - G^{01}_{m-1,k} .$$

Since prefixing 0 to a string does not increase the number of 101s, $G^{01}_{m,k} = G^1_{m-1,k}$. Consequently, $G^1_{m,k} = G_{m-1,k} + G^{01}_{m-1,k-1} - G^{01}_{m-1,k}$.

We arrive at the following simultaneous equations:

$$\begin{aligned} G_{m,k} &= 2G_{m-1,k} + G^{01}_{m-1,k-1} - G^{01}_{m-1,k} \\ G^{01}_{m,k} &= G^1_{m-1,k} \\ G^1_{m,k} &= G_{m-1,k} + G^{01}_{m-1,k-1} - G^{01}_{m-1,k} . \end{aligned}$$

After straightforward substitutions, we arrive at the equation from the Proposition. The initial data can be verified by direct calculation. ■

The above recurrence relation is useful because it has the ability to predict the redundancy of the weakly-constrained block code for large values of m, for which direct numerical evaluation may be difficult (though arguably still possible using the Monte-Carlo method), while the cost of evaluating the recurrences is small.

Let us first restrict ourselves to the code with minimum redundancy: $k = 0$, and denote $G_{m,0}$ as G_m. Proposition 6.1 will now come to: $G_m - 2G_{m-1} + G_{m-2} - G_{m-3} = 0$ with the initial data $G_3 = 7, G_4 = 12, G_5 = 21$. For $y_m = G_{m+3}$ $(m = 0, 1, ...)$. we have $y_{m+3} - 2y_{m+2} + y_{m+1} - y_m = 0$. Using standard methods, the solution of this equation can be found: $G_{m+3} = y_m = c_1\xi^m + \rho^m(c_2 \cos m\varphi + c_3 \sin m\varphi)$, where $c_1 \approx 6.8486$, $c_2 \approx 0.1514$, $c_3 \approx -0.0496$, $\varphi \approx 1.408$, $\rho \approx 0.7549$.

Note that since $\rho < 1$, $y_m = O(\xi^m)$ in the limit of large m. Since the number of code words of length m is in fact $G_m = y^{m-3}$ and the number of all strings of length m is 2^m, the asymptotic code capacity $\frac{1}{m}\log_2 G_m \to \log_2 \xi \simeq 0.8114$ bits per digit. This is higher than the Markov chain result 0.8. Remember that our block codes are optimised for minimum redundancy at a given skew by sorting the list of code words in the ascending order in the number of 101 triplets per word.

Evidently, this procedure effectively changes not only the two pairs of transitions (in the language of the Markov chain) but perhaps all of them to some extent.

B. Redundancy of the block code for arbitrary ε

Denote as $P(x)$ the probability that the bit sequence x occurs in the *encoded* message. For example, P(1010) is the probability that four consecutive bits taken at random will turn out to be the sequence 1010. Also denote as $S(m,k)$ a set of length-m sequences, where each sequence has *exactly* k triplets 101.

First of all, the codeword space of the block code for any ε will include all codewords containing zero triplets 101. Therefore $P_{000} \neq 0$, because the zero codeword is included in the codeword space anyway. Similarly $P_{100} \neq 0$ and $P_{110} \neq 0$.

Lemma 6.2: For any $k \geq 0$, $m \geq 5$, the equation $P(00000) = P(00100)$ holds.

Proof: Since neither the replacement $00000 \to 00100$, nor its inverse modifies the frequency of 101s in the sequence, these replacements exchange a codeword $B \in S(m,k)$ with another codeword $B' \in S(m,k)$. Similarly it can be proven that $P(00001) = P(00011)$, $P(10001) = P(10011)$, $P(00010) = P(00110)$, $P(11101) = P(11011)$, $P(001100) = P(001000)$, $P(000010) = P(001110)$. ■

Since the block code for each ε will contain at least one codeword with patterns 0000, 0001 1000, 1001, 1110, 1111, 0110, 0111, the inequalities $|\varepsilon_0| \neq 1$, $|\varepsilon_4| \neq 1$, $|\varepsilon_3| \neq 1$ and $|\varepsilon_7| \neq 1$ are correct. Therefore according to Table I, $P_1 = P_4 = P_0(1+\varepsilon_0)/(1-\varepsilon_4)$, $P_3 = P_6 = P_7(1-\varepsilon_7)/(1+\varepsilon_3)$ for any skew-parameter vector $\vec{\varepsilon} = (\varepsilon_0, \varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5, \varepsilon_6, \varepsilon_7)$. Let us find this vector using the above codeword properties. Bearing the above lemma in mind and remembering the transition probabilities of the Markov chain, we derive the skew parameter vector:

$$\vec{\varepsilon} = \varepsilon_0 \left(1, -1, \frac{\varepsilon_0^2 - 2\varepsilon_0 + 5}{(1+\varepsilon_0)^2}, -1, \right. \\ \left. 1, -1, \frac{\varepsilon_0^2 - 2\varepsilon_0 + 5}{(1+\varepsilon_0)^2}, -1 \right), \qquad (2)$$

where every skew parameter is expressed in terms of a single skew ε_0 (see Table III).

However, to be able to juxtapose the multi-skew block code and a single-skew Markov chain, we must first define the basis for comparison. Such a basis could be an effective skew value ε_{eff} such that the density of 101 in the block code at a given $\vec{\varepsilon}$ corresponds to the density of 101 in the single-skew Markov chain at the skew value ε_{eff}. In other words, ε_{eff} indicates how much the single-skew Markov chain would have to be skewed to achieve the same density of the undesirable states. The latter is simply $P_5 = (1-\varepsilon_{eff})/(8+2\varepsilon_{eff})$ whereas from the Markov chain density and the $\vec{\varepsilon}$ expression above we get

$$P_5 = \frac{\varepsilon_0^4 + 6\varepsilon_0^2 + 8\varepsilon_0 + 1}{8(2\varepsilon_0^2 + \varepsilon_0 + 1)} = (1-\varepsilon_{eff})/(8+2\varepsilon_{eff})\,.$$

This gives us ε_0 in terms of ε_{eff}.

TABLE III
CALCULATION OF THE ε_i VALUES

Property	Corollary
$P(00001) = P_0 P_{0\to0} P_{0\to1}$ $P(00011) = P_0 P_{0\to1} P_{1\to3}$	$\varepsilon_1 = -\varepsilon_0$
$P(10011) = P_4 P_{4\to1} P_{1\to3}$ $P(10001) = P_4 P_{4\to0} P_{0\to1}$	$\varepsilon_4 = \varepsilon_0$
$P(00010) = P_0 P_{0\to1} P_{1\to2}$ $P(00110) = P_1 P_{1\to3} P_{3\to6} =$ $P_0 \frac{1+\varepsilon_0}{1-\varepsilon_4} P_{1\to3} P_{3\to6}$	$\varepsilon_3 = -\varepsilon_0$
$P(00000) = P_0 P_{0\to0} P_{0\to0}$ $P(00100) = P_1 P_{1\to2} P_{2\to4} =$ $P_0 \frac{1+\varepsilon_0}{1-\varepsilon_4} P_{1\to2} P_{2\to4}$	$\varepsilon_2 = 1 - \frac{(1-\varepsilon_0)^3}{(1+\varepsilon_0)^2} =$ $\frac{\varepsilon_0(\varepsilon_0^2 - 2\varepsilon_0 + 5)}{(1+\varepsilon_0)^2}$
$P(001100) = P_1 P_{1\to3} P_{3\to6} P_{6\to4}$ $P(001000) = P_1 P_{1\to2} P_{2\to4} P_{4\to0}$	$\varepsilon_6 = \varepsilon_2 =$ $\frac{\varepsilon_0(\varepsilon_0^2 - 2\varepsilon_0 + 5)}{(1+\varepsilon_0)^2}$
$P(11101) = P_7 P_{7\to6} P_{6\to5}$ $P(11011) = P_6 P_{6\to5} P_{5\to3} =$ $P_7 \frac{1-\varepsilon_7}{1+\varepsilon_3} P_{6\to5} P_{5\to3}$ If $\varepsilon_6 = -1$, the ε_5 value has no any meaning and can be assigned as $\varepsilon_5 = \varepsilon_1$.	$\varepsilon_5 = \varepsilon_1 = -\varepsilon_0$ if $\varepsilon_6 \neq -1$
$P(000010) = P_0 P_{0\to0} P_{0\to1} P_{1\to2}$ $P(001110) = P_1 P_{1\to3} P_{3\to7} P_{7\to6} =$ $P_0 \frac{1+\varepsilon_0}{1-\varepsilon_4} P_{1\to3} P_{3\to7} P_{7\to6}$	$\varepsilon_7 = \varepsilon_3 = -\varepsilon_0$

C. Calculation of the redundancy

The above-mentioned quartic equation is of the form:

$$\varepsilon_0{}^4 + 2\varepsilon_0{}^2(3 - 8P_5) + 8\varepsilon_0(1 - P_5) - 8P_5 + 1 = 0$$

and can be proven to have a single root in the interval $-1 \leq \varepsilon_0 \leq 1$ given that $0 \leq P_5 \leq 1/8$. Obtaining the root numerically presents no technical problem, so ε_0 can be assumed to be functionally dependent on ε_{eff} in a known way.

Consequently, let us express the entropy in terms of ε_0. Using the general formula from [8] and interpreting $\vec{\varepsilon}$ as a function of ε_0 we get

$$h(\varepsilon_0) = -\sum_{k=0}^{7} P_k(\vec{\varepsilon}(\varepsilon_0)) f(\vec{\varepsilon}(\varepsilon_0)),$$

where $f(x) = \frac{1+x}{2}\log_2\frac{1+x}{2} + \frac{1-x}{2}\log_2\frac{1-x}{2}$. By simplifying the equations from Table I for four εs as mentioned above and using 2 we establish that

$$h(\varepsilon_0) = f(\varepsilon_0)(P_0(\varepsilon_0) + P_1(\varepsilon_0) + P_3(\varepsilon_0) + \\ P_4(\varepsilon_0) + P_5(\varepsilon_0) + P_7(\varepsilon_0)) + \\ f(\varepsilon_2(\varepsilon_0))(P_2(\varepsilon_0) + P_6(\varepsilon_0)).$$

By varying ε_{eff} (and via it, ε_0) over its range, we obtain the curve Bk∞ in Fig. 2. Observe that block size 128 is roughly as close to the result obtained from the single-skew Markov chain as the theoretical limit of all weakly-constrained codes with a single skew, i.e. the curve "Markov chain". It seems likely that for any practical purpose block size 128 will be sufficient. Another consideration would be the required level of interleaving and/or FEC to mitigate significant error propagation that a block code longer than 128 would be prone to.

D. Code gain

We assume that the vector of error probabilities is $\vec{Q} = q(1,1,1,1,1,M,1,1)$. Since the code constructed above is a

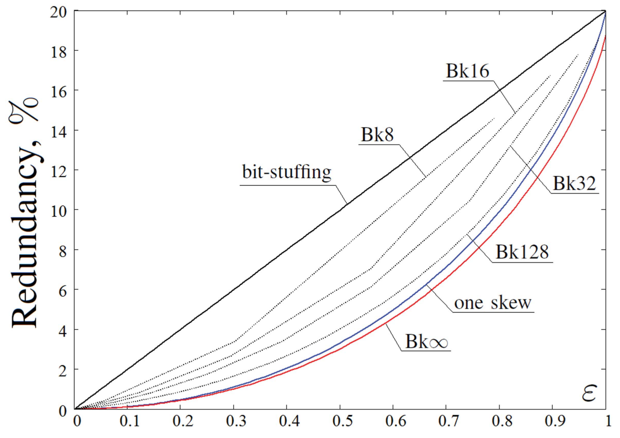

Fig. 2. $R(\varepsilon)$ graph for various codes. Here BkX marks the block code with block length of X bits; the curve "one skew" shows the redundancy of Markov chain with one skew parameter ($\varepsilon_2 = -\varepsilon$); Bk$\infty$ marks the limiting block code with infinite block length.

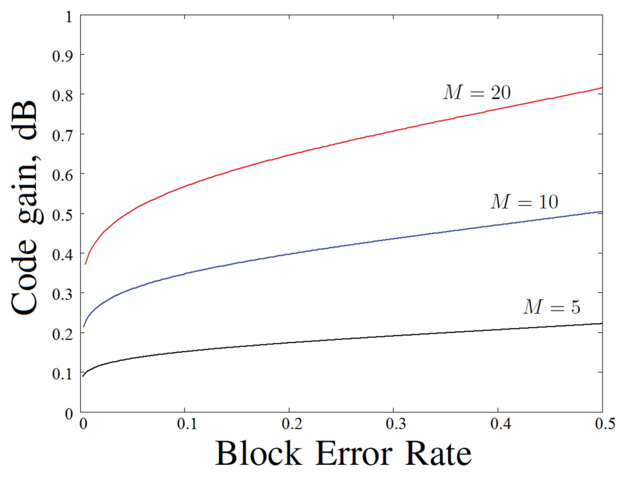

Fig. 3. Block code gain versus BER ($R = 10.27\%$, block length — 128 bit).

block code and since it is capable of propagating errors up to the block borders, it makes sense to express the code gain using the block error rate ($BLER$) rather than the BER. For the sake of clarity denote as BER_0 and $BLER_0$ the BER and the $BLER$ for the unencoded data block respectively; also denote as $BER(\varepsilon)$ and $BLER(\varepsilon)$ the BER and the $BLER$ for the encoded data block. Obviously, $BER_0 = q + \frac{1}{8} \cdot q(M-1)$ and $BER(\varepsilon) = q + P_5(\varepsilon)q(M-1)$. Then $BLER_0 = 1-(1-BER_0)^p$, where p is the unecoded block length.

Since our codes have the redundancy $R(p,\varepsilon)$, which depends on the block length and the value of ε, $BLER(\varepsilon) = 1-(1-BER(\varepsilon))^{p(1+R(p,\varepsilon))}$. Using this equation the value of $BER'(\varepsilon) = 1-(1-BLER(\varepsilon))^{1/p}$ can be found. The bit error rate $BER'(\varepsilon)$ corresponds to the unencoded data block with the block error rate $BLER(\varepsilon)$. After elementary transformations we get $BER'(\varepsilon) = 1-(1-BER(\varepsilon))^{1+R(p,\varepsilon)}$. It can be proven that $BER'(\varepsilon) \geq BER(\varepsilon)$ for any value $R(p,\varepsilon)$.

Remember that in our case the distortion of data during transmission is data-dependent, which is what is usually termed signal-dependent noise (SDN). Since methods for SDN analysis are not readily available, below we quantify SDN in terms of the effective AWGN. To quantify the BER improvement due to our block code we introduce an effective code gain defined as $\Gamma = 20\log_{10}(A(M,q)/A_c(M,q,\varepsilon))$, where $A(M,q)$ is the RMS of the effective AWGN over the source signal, i.e. the magnitude that corresponds to the observed BER, and where $A_c(M,q,\varepsilon)$ is the RMS of the effective AGWN after the block coding. Then $A^2(M,q) = \sigma^2$, where σ^2 is the standard variation of the AWG noise. Similarly, $A_c^2(M,q,\varepsilon) = \sigma'^2$, where σ'^2 is the variation of the total noise for the encoded signal. The σ values can be quantified by solving the equations: $BER_0 = \frac{1}{2} \cdot \Phi\left(\frac{1}{\sigma\sqrt{2}}\right)$, $BER'(\varepsilon) = \frac{1}{2} \cdot \Phi\left(\frac{1}{\sigma'\sqrt{2}}\right)$. Here $\Phi(x)$ is a complementary error function. Finally we get $\Gamma = 10\log_{10}\frac{\sigma^2}{(\sigma')^2}$.

The value of Γ corresponds to the increase of SNR that delivers the same error probability reduction in the case of AGWN without patterning as the block code delivers under the patterning defined by the vector $\vec{Q}$.

Fig. 3 shows the code gain as a function of the BER for the block code Bk128 (see Fig. 2) taken at 10% redundancy. It is remarkable that a gain of about 0.5dB is possible for a channel with $BLER = 0.3$ ($BER = 2.7 \cdot 10^{-3}$) from the simple code with $R = 10\%$. It should be borne in mind that the gain displayed in the figure does not take into account any processes associated with decoding (and long block codes are almost as prone to error propagation as convolution codes). Nevertheless, if adequately protected by FEC, the effects of error propagation with a moderate block size can be alleviated. Schemes for a constrained code and FEC concatenation are considered in [17]. Similar schemes can be used in our case, too. It may also be possible to devise a soft decoding scheme based on the algorithm presented in the next section, which is less prone to error propagation effects.

E. Encoder/decoder algorithms for large block sizes

In contrast to the Markov model the weakly-constrained block code described above includes, by construction, the maximum number of code words satisfying the pattern constraint for a given ε. Consequently it has the least redundancy among all codes with that ε. It is therefore quite important to find algorithms that realise the encoder and decoder mappings efficiently and to get an idea of their complexity.

For a small m, the best coding strategy is table look-up: build a table of code words indexed by the source message block and use it for encoding. This way encoding is achieved by a single table reference per block. Decoding could be done similarly, by preparing the inverse table in advance. As the value of m increases (in practice already at $m \geq 32$), storage requirements become prohibitive, necessitating a computational rather than purely table-based solution. As we have seen earlier, the redundancy curve saturates near $m = 128$, where table look-up is many orders of magnitude beyond any reasonable storage capacity, so the need for an efficient algorithm will not be eliminated by improvements in storage capacity in the foreseeable future. Besides low redundancy, an advantage of a large m is the fact that the list of codewords, which is of the length $L_{m,k}$, could be shortened to the length 2^n where $n = \lfloor log_2 L_{m,k} \rfloor$ increasing redundancy by a fraction of one bit for every n bits of the message, a negligible increase for $m = 128$ (corresponding to $n > 0.8 \cdot 128 \sim 100$). The benefit of the shortening is that the shortened code replaces a string of n bits by a string of m bits and then applies the same function to the next portion of n bits. Below we shall only deal with a single block.

Specification Given the index of an m-bit code word in the (imaginary) table for block code (m, k), compute the code word. Assume that $m \geq 5$, otherwise use actual table look-up. Similarly for decoding: given a valid m-bit code word for the block code (m, k), i.e. any m-bit string with no more than k 101-triplets, find the index of this code word in the imaginary code table. The code table is assumed to have been sorted in ascending order within each fixed-k segment.

Solution idea: Generally, encoding/decoding procedures are based on enumerative techniques. In [17] enumerative algorithms for RLL-sequences were proposed. Below we develop similar algorithms for a constrained code other than RLL, namely the weakly-constrained block code for the suppression of triplets 101.

Denote as N the index of the code word that contains no more than k 101-combinations. Then $0 \leq N < L_{m,k}$. The algorithm emulates a reference to the code look-up table, in which elements with indices $0 \leq N < G_{m,0}$ have no 101 triplets, those with indices $G_{m,0} \leq N < G_{m,0} + G_{m,1}$ have exactly one 101 triplet, etc. and generally those with indices $\sum_{j=0}^{k-1} G_{m,j} \leq N < \sum_{j=0}^{k} G_{m,j} = L_{m,k}$ have exactly k undesirable triplets. Consequently this virtual table is laid out in the increasing order of codeword values and nondecreasing order of triplets-per-word as required for the block code.

For encoding we need first to find the value of d: the number of 101-triplets in the result code word, $d = \max\{p \mid N \geq \sum_{j=0}^{p-1} G_{m,j}\}$. Next compute $M = N - L_{m,d-1}$ the codeword index in the section of the table with d 101-triplets, counting from the bottom of the segment; it is obvious that $0 \leq M < G_{m,d}$.

Denote the i-th bit of the code word as b_i ($0 \leq i < m-1$ and the numbering is from the least significant bit). The code word is computed bit-by-bit starting with the most significant bit. The computation closely follows the recurrence relation (Eq. 1).

1. Find b_{m-1}: Introduce a running M_i, $0 \leq i \leq m-1$ and set $M_{m-1} = M$. If $0 \leq M < G_{m-1,d}$ we determine $b_{m-1} = 0$ and then we set $M_{m-2} = M_{m-1}$. Otherwise, if $G_{m-1,d} \leq M < G_{m,d}$, we determine $b_{m-1} = 1$ and set $M_{m-2} = M_{m-1} - G_{m-1,d}$.

2. Assume that $p \geq 1$ most significant bits have been determined : $b_{m-1}, b_{m-2}, \ldots, b_{m-p}$. These bits will include $d_p \leq d$ 101-triplets. We determine that $b_{m-p-1} = 0$, and set $M_{m-p-1} = M_{m-p}$, if $0 \leq M_{m-p} < \mathbf{A}$, and otherwise we conclude that $b_{m-p-1} = 1$, and set $M_{m-p-1} = M_{m-p} - \mathbf{A}$. Here $\mathbf{A}$ is a threshold value, which depends on the two previously determined bits (we can legitimately assume $b_m = 0$ for the very first application of step 2).

The threshold values are summarised in Table IV. After elementary transformations that use previously established identities, we can eliminate the upper indices. Threshold values expressed in terms of $G_{m,k}$ are given in Table V.

TABLE IV
THRESHOLD VALUES

b_{m-p+1}	b_{m-p}	$\mathbf{A}$
0	0	$G_{m-p-1,d-d_p}$
0	1	$G^0_{m-p-1,d-d_p} + G^1_{m-p-1,d-d_p-1}$
1	0	$G_{m-p-1,d-d_p}$
1	1	$G^0_{m-p-1,d-d_p} + G^1_{m-p-1,d-d_p-1}$

TABLE V
THRESHOLD VALUES FOR ENCODING/DECODING ALGORITHM

b_{m-p+1}	b_{m-p}	$\mathbf{A}$
0	0	$G_{m-p-1,d-d_p} = \mathbf{A_0}(m-p, d-d_p)$
0	1	$G_{m-p-1,d-d_p-1} + G_{m-p-2,d-d_p} - G_{m-p-2,d-d_p-1} = \mathbf{A_1}(m-p, d-d_p)$
1	0	$G_{m-p-1,d-d_p} = \mathbf{A_0}(m-p, d-d_p)$
1	1	$G_{m-p-1,d-d_p-1} + G_{m-p-2,d-d_p} - G_{m-p-2,d-d_p-1} = \mathbf{A_1}(m-p, d-d_p)$

For any possible $m-p$ and $d-d_p$ combination, the values of $\mathbf{A}_{\{0,1\}}$ can be obtained from a table prepared once beforehand, bearing in mind the obvious boundary values $G_{m,0} = 1$ for $m < 0$, $G_{m,0} = 2^m$ for $0 \leq m < 3$, $G_{m,k} = 0$ for $k < 0$, and $G_{m,k} = 0$ whenever $k > 0$ and $m < 3$.

F. Encoding/decoding algorithms

The encoding/decoding algorithms are presented in the Table VI. The critical path for processing one bit through the encoding algoritm contains one left shift (t_s), four assignments (t_a), two bitwise comparisons (t_{bc}), one comparison (t_c) and one subtraction (t_s). Denoting the register transfer cost as T, we get the total time per loop cycle: $9T + t_s + 4t_a + 2t_{bc} + t_c + t_s$. In reality the cost of comparisons and subtractions is generally dependent on m, due to the carry propagation delay in subtractions and the result propagation towards the lower digits in comparisons. However, various solutions are available which accelerate the propagation, making the overall complexity estimate logarithmic in m. The assignment cost, too, can be reduced: by speculatively executing both alternatives of the *if* while unrolling the loop to increase its pipelined concurrency. A microelectronic implementation of those measures can easily be obtained, but any technical details would go beyond the scope of the present paper. We only wish to note that the cost is generally $O(m \log m)$ and that the constant in it can be made quite small by employing modern design principles in microelectronics.

Similarly, the critical path for the decoding algorithm includes one left shift (t_s), three assignments (t_a), two bitwise comparisons (t_c), one addition (t_{add}) and one subtraction (t_s). The total time per loops cycle is thus $7T + t_s + 3t_a + 2t_c + t_{add}$, which is similar to that for the encoding algorithm and the same considerations regarding implementation apply.

G. Memory requirement

Let us estimate the read-only storage size, which is required for m-bit block-coding, for any $0 \leq k \leq k_{max}$. For this one needs to tabulate $\mathbf{A}_0(i, j)$ and $\mathbf{A}_1(i, j)$, where $\mathbf{A}_0(i, j) =$

TABLE VI
ENCODING/DECODING ALGORITHMS

Encoding algorithm:	
Input: a code word index N.	**Output:** The length-m code word (denote it as B) that corresponds to index N.
bit[$0 : m-1$] $B = 0$;	code word
bit[0:2] $s = 0$;	its most recently computed three bits
int c;	loop counter
int d;	the number of 101-triplets
$d = \max\{p \mid N \geq \sum_{j=0}^{p-1} G_{m,j}\}$	in the codeword
bit[$0 : m-1$] $M = N - L_{m,d-1}$;	sequence number in subset of sequences which has exactly d 101-triplets ($0 \leq M < G_{m,d}$)
for $c \leftarrow (m-1)$ **downto** 0	
ShiftLeft s;	
if $M \geq \mathbf{A}_{\mathbf{s[1]}}[c+1, d]$ **then**	
$M \leftarrow M - \mathbf{A}_{\mathbf{s[1]}}[c+1, d]$;	
$B[c] \leftarrow 1$;	
$s[0] \leftarrow 1$;	
if $s = 101$ **then** $d \leftarrow d - 1$.	
if $M = 0$ **and** $d = 0$ **then**	
break.	
Decoding algorithm:	
Input: A length-m code word (B).	**Output:** The code word index N that corresponds to B.
int $d =$ triplets(B)	the number of 101-triplets in the codeword B
int $N = \sum_{j=0}^{d-1} \mathbf{A_0}[m+1, j]$;	index N
bit[0:2] $s = 0$;	its most recently computed three bits
int c;	loop counter
for $c \leftarrow (m-1)$ **downto** 0	
ShiftLeft s;	
$s[0] \leftarrow B[c]$;	
if $s[0] = 1$ **then**	
$N \leftarrow N + \mathbf{A}_{\mathbf{s[1]}}[c+1, d]$;	
if $s = 101$ **then** $d \leftarrow d - 1$.	

$G_{i-1,j}$, $\mathbf{A}_1(i,j) = G_{i-1,j-1} + G_{i-2,j} - G_{i-2,j-1}$, $0 \leq i \leq m$ and $0 \leq j \leq k_{max}$. Since $\mathbf{A}_0(i,j) = G_{i-1,j} < 2^m$, $G_{i-1,j-1} < 2^{m-1}$ and $G_{i-2,j} < 2^{m-2}$, $\mathbf{A}_1(i,j) < 2^{m-1} + 2^{m-2} < 2^m$. We get two $(m+1) \times (k_{max}+1)$ tables, one for $\mathbf{A}_0$ and one for $\mathbf{A}_1$. Each table requires $(m+1) \cdot (k_{max}+1) \cdot m$ bits.

The total storage read-only capacity is $S \leq 2m(m+1)(k_{max}+1)$ bits. In particular, block coding requires no more than 262 Kbytes of memory, if the block size is 128 bits, for any k.

To summarise the results of this section, the time complexity of the encoder/decoder is log-linear in m, $T = O(m \log m)$, and the read-only storage requirement is quadratic in m and linear in k, $S = O(m^2 k)$.

VII. WEAKLY-CONSTRAINED BLOCK CODE FOR OTHER TRIPLETS

Since nonlinear effects in optical fibre communications can lead to intricate error statistics, not necessarily resulting from the "ghost" pulse phenomena alone, suppression of other triplets may be desirable in data transmission systems.

In this section we will denote as $G^a_{m,k}$ the number of length-m sequences which contain no more than k occurrences of the triplet a. For each triplet we have established recurrence relations similar to Eqs. 1. They are listed in Table VII and can be employed in an encoding/decoding algorithm. For calculation of $G^a_{m,k}$ the following initial data must be used: $G^a_{m,0} = 2^m$, $G^a_{m,k} = 0$ for $k > 0$ and $0 \leq m < 3$, $G^a_{m,k} = 0$ for $k < 0$.

TABLE VII
$G^a_{m,k}$ RECURRENCE RELATIONS FOR ANY TRIPLETS

Triplets, a	Recurrence relation $G^a_{m,k}$ ($m \geq 3$, $k \geq 0$)
000 111	$G^a_{m,k} = G^a_{m-1,k} + G^a_{m-2,k} + G^a_{m-3,k} + G^a_{m-1,k-1} - G^a_{m-2,k-1} - G^a_{m-3,k-1}$
101 010	$G^a_{m,k} = 2G^a_{m-1,k} - G^a_{m-2,k} + G^a_{m-3,k} + G^a_{m-2,k-1} - G^a_{m-3,k-1}$
100 001 110 011	$G^a_{m,k} = 2G^a_{m-1,k} - G^a_{m-3,k} + G^a_{m-3,k-1}$

TABLE VIII
SKEW-PARAMETER TABLE FOR SOME TRIPLETS

Triplet	Skew-parameter vector
000	$\overrightarrow{\varepsilon}(t) = t\left(-\frac{t^2+4t+7}{t^2+4t-1}, 1, \frac{t+3}{1-t}, 1, -\frac{t^2+4t+7}{t^2+4t-1}, 1, \frac{t+3}{1-t}, 1\right)$ $t = \varepsilon_3$
101	$\overrightarrow{\varepsilon}(t) = t\left(1, -1, \frac{t^2-2t+5}{(1+t)^2}, -1, 1, -1, \frac{t^2-2t+5}{(1+t_0)^2}, -1\right)$ $t = \varepsilon_0$
100	$\overrightarrow{\varepsilon}(t) = t\left(1, -1, \frac{t-3}{t+1}, -1, 1, -1, \frac{t-3}{t+1}, -1\right)$ $t = \varepsilon_0$

Table VII shows that some triplets have the same value of $G^a_{m,k}$. Let us dwell on this a little. Denote as $\overline{a}$ the inversion of the triplet a (e.g. $\overline{100} = 011$) and denote as $\overleftarrow{a}$ the triplet a placed in the reverse order (e.g. $\overleftarrow{100} = 001$).

Proposition 7.1: $G^a_{m,k} = G^{\overline{a}}_{m,k}$ $G^a_{m,k} = G^{\overleftarrow{a}}_{m,k}$.

Proof: Denote as $S^a(m,k)$ the set of length-m sequences where each sequence has exactly k occurrences of the triplet a. If sequence $Q \in S^a(m,k)$, then obviously $\overline{Q} \in S^{\overline{a}}(m,k)$, and vice-versa. Therefore $G^{\overline{a}}_{m,k} = |S^{\overline{a}}(m,k)| = |S^a(m,k)| = G^a_{m,k}$.

Similarly, when sequence $Q \in S^a(m,k)$ is replaced by $\overleftarrow{Q}$, each occurrence of the triplet a is replaced by $\overleftarrow{a}$. Hence the sequence $\overleftarrow{Q}$ will contain exactly k occurrences of the triplet $\overleftarrow{a}$, and so $\overleftarrow{Q} \in S^{\overleftarrow{a}}(m,k)$. Therefore $G^{\overleftarrow{a}}_{m,k} = |S^{\overleftarrow{a}}(m,k)| = |S^a(m,k)| = G^a_{m,k}$. ■

As a corollary, $G^{000}_{m,k} = G^{111}_{m,k}$, $G^{010}_{m,k} = G^{101}_{m,k}$, $G^{001}_{m,k} = G^{100}_{m,k}$ and $G^{100}_{m,k} = G^{011}_{m,k} = G^{110}_{m,k}$.

Using the block code properties we can find the skew-parameter vector and plot the minimum-redundancy graph for each triplet as proposed above for 101. We have found the skew-parameter vector for triplets 000 and 100 in addition to 101. The results are summarized in Table VIII. Using these vectors the minimum-redundancy curves for 000, 101 and 100 triplets can be built. Redunancy curves versus ε for limiting block codes are shown in Fig. 4. Here ε is the "degree of weakeness" for our weakly-constrained code and $P_i = \frac{1-\varepsilon}{8}$, where P_i is the rate of eliminating triplet i: when $\varepsilon = 1$, the code eliminates all triplets i; and when $\varepsilon = 0$ the code does not eliminate any triplets.

VIII. CONCLUSIONS

A detailed analysis of block and stream versions of weakly-constrained codes generalizing the first theoretical results in

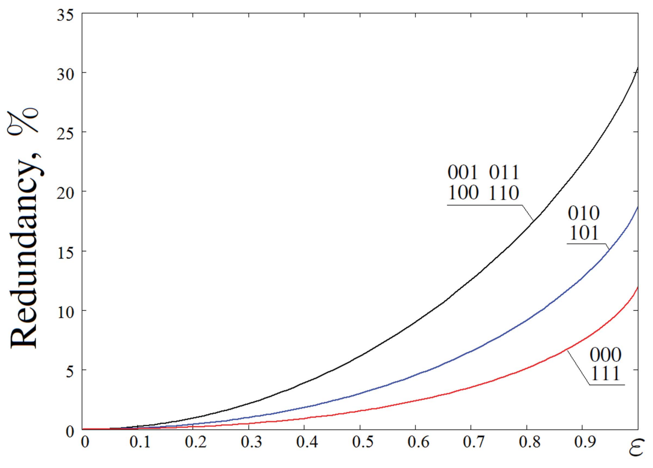

Fig. 4. Limiting block code redundancy for various triplets.

[8] has been presented. A redundancy measure is obtained against the effective skewing factor ε for specific stream and block codes, and it turns out to be somewhat better than the theoretical model published previously. The effective code gain for practical block codes has been obtained and plotted against a range of bit error rates at various magnitudes of the patterning effect. It is shown that using these simple codes in a hard-decision situation provides an effective gain of about 0.5dB. For large blocks, which correspond to the lowest redundancy for a given "degree of weakness", efficient encoder/decoder algorithms are proposed, whose time-complexity grows no faster than log-linearly in the block size, and whose storage requirements are very small. Due to their simplicity they should permit a simple microelectronic (and potentially optical) implementation. The analysis presented here can be used in code design for channels with a variety of pattern-dependent statistics. Note also that the proposed approach can potentially be used in areas such as magnetic recording, where pre-encoding is routinely used for the avoidance of undesirable bit patterns.

Future work will include investigating the decoder's ability to participate in soft-decision schemes, such as the product of an LDPC code and a weakly-constrained block code of the kind discussed in the present paper.

References

[1] F. Matera, A. Mecozzi, M. Settembre, I. Gabitov, H. Haunstein, and S. K. Turitsyn, "Theoretical evaluation of the noise growth and the system performance for a link constitued by a chain of N optical amplifiers with in-line filters," *OFC'98 Tech. Dig.*, WM23, p. 202, 1998.

[2] R. J. Essiambre, B. Mikkelsen, and G. Raybon, "Intra-channel cross-phase modulation and four-wave mixing in high-speed TDM systems," *Electron. Lett.*, vol. 35, pp. 1576-1578, 1999.

[3] P. V. Mamyshev and N. A. Mamysheva, "Pulse-overlapped dispersion-managed data transmission and intrachannel four-wave mixing," *Opt. Lett.*, vol. 24, pp. 1454-1456, 1999.

[4] B. Vasic, V. S. Rao, I. B. Djordjevic, R. K. Kostuk, and I. Gabitov, "Ghost-pulse reduction in 40-Gb/s systems using line coding," *IEEE Photon. Technol. Lett.*, vol. 16, pp. 1784-1786, July 2004.

[5] A. H. Gnauck, A. Mecozzi, M. Shtaif, and J. Wiesenfeld, "Modulation scheme for tedons," U.S. patent application, #20020126359, 2001.

[6] E. G. Shapiro, M. P. Fedoruk, S. K. Turitsyn, and A. Shafarenko, "Reduction of nonlinear intrachannel effects by channel asymmetry in transmission lines with strong bit overlapping," *IEEE Photon. Technol. Lett.*, vol. 15, pp. 1473-1475, Oct. 2003.

[7] A. Shafarenko, K. S. Turitsyn, and S. K. Turitsyn, "Skewed coding for suppression of pattern-dependent errors," in *Proc. 31st European Conf. Optical Commun. (ECOC 2005)*, vol. 2, pp. 193-194, Glasgow, United Kingdom, Sep. 2005.

[8] A. Shafarenko, K. S. Turitsyn, and S. K. Turitsyn, "Information-theory analysis of skewed coding for suppression of pattern-dependent errors in digital communications," *IEEE Trans. Commun.*, vol. 55, no. 2, pp. 237-241, 2007.

[9] B. Djordjevic and B. Vasic, "Nonlinear BCJR equalizer for suppression of intrachannel nonlinearities in 40 Gb/s optical communications systems," *Opt. Express*, vol. 14, pp. 4625-4635, May 29, 2006.

[10] S. K. Turitsyn, M. P. Fedoruk, O. V. Shtyrina, A. V. Yakasov, A. Shafarenko, S. R. Desbruslais, K. Reynolds, and R. Webb, "Patterning effects in a WDM RZ-DBPSK SMF/DCF optical transmission at 40Gbit/s channel rate," *Optics Commun.*, vol. 277, no. 2, pp. 264-268, 2007.

[11] B. Slater, S. Boscolo, A. Shafarenko, and S. K. Turitsyn, "Mitigation of patterning effects at 40 Gbits/s by skewed channel pre-encoding," *J. Optical Netw.*, vol. 6, no. 8, pp. 984, 2007.

[12] N. Kashyap, P. H. Siegel, and A. Vardy, "Coding for the optical channel: the ghost-pulse constraint," *IEEE Trans. Inf. Theory*, vol. 52, no. 1, pp. 64-77, 2006.

[13] E. G. Shapiro, M. P. Fedoruk, and S. K. Turitsyn, "Direct modelling of error statistics at 40 Gbit/s rate in SMF/DCF link with strong bit overlapping," *Electron. Lett.*, vol. 40, no. 22, pp. 1436-1437, 2004.

[14] N. L. Swenson and J. M. Cioffi, "Sliding-block line codes to increase dispersion-limited distance of optical fiber channels," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 3, pp. 486-498, 1995.

[15] K. A. S. Immink, "Weakly constrained codes," *Electron. Lett.*, vol. 33, no. 23, pp. 1943-1944, Nov. 1997.

[16] M. Jin, K. A. S. Immink, and B. Farhang-Boroujeny, "Design techniques for weakly constrained codes," *IEEE Trans. Commun.*, vol. 51, no. 5, pp. 709-714, May 2003.

[17] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inf. Theory*, vol. 43, no. 5, pp. 1389-1399, 1997.

Alex Shafarenko received his graduate Diploma in Physics from the Novosibirsk State University Department of Physics in 1983, and a Ph.D. from the Siberian Branch of the Russian Academy of Sciences in 1990. He joined faculty at the University of Surrey, England as Senior Lecturer within the Department of Electronic and Electrical Engineering and was subsequently made a Reader. He was appointed to his current position Professor of Software Engineering in 2000 by the University of Hertfordshire as he joined the School of Computer Science. Dr Shafarenko has led several international research projects in the area of advanced signal processing and parallel computing.

Anton Skidin received his graduate Diploma in Electronic Engineering from Kurgan State University in 2005. He is currently a Research Fellow of the Institute of Computational Technologies Russian Academy of Sciences Siberian Branch.

Sergei K. Turitsyn graduated from the Department of Physics of the Novosibirsk University, Russia in 1982 and received the Ph.D. degree in Theoretical and Mathematical Physics from the Institute of Nuclear Physics, Novosibirsk, Russia in 1986. From 1992 to 1998, he was with the Institute for Theoretical Physics I, Heinrich-Heine University, Duesseldorf, Germany, first as a Humboldt Fellow and later as Leader of the collaborative projects with Deutsche Telekom. He joined the Photonics Research Group in the School of Engineering and Applied Science, Aston University, United Kingdom in 1998 and is now one of the Photonics Group leaders. Professor Sergei Turitsyn was a recipient of the Royal Society Wolfson Research Merit Award in 2005.