References

1. Kuts Yu., Shcherbak L. Statistical phasometry. Ternopil, 2009. 383 p.
2. Lapiga I. Circular statistics application for accuracy improvement of ultrasonic thickness measurement // "Proceedings of IX Conference Polit-2009". Kyiv: NAU, 2009. P. 60.
3. King F. W. Table of selected Hilbert transforms // Hilbert Transforms. Cambridge: Cambridge University Press, 2009. P. 453–533. doi: 10.1017/cbo9780511735271.011
4. Bendat J. S., Piersol A. G. Random Data: Analysis and Measurement Procedures. 4th ed. Hoboken: John Wiley & Sons, Inc., 2010. 640 p. doi: 10.1002/9781118032428
5. High Frequency Ultrasonic Non Destructive Evaluation of Additively Manufactured Components / Karthik N., Gu H., Pal D., Starr T., Stucker B. // 24th Annual Int. Solid Freeform Fabr. Austin, TX, 2013. P. 311–325.
6. Including frequency-dependent attenuation for the deconvolution of ultrasonic signals / Carcreff E., Bourguignon S., Idier J., Simon L., Duclos A. // Proceedings of Meetings on Acoustics. 2013. Vol. 19, Issue 1. P. 055029. doi: 10.1121/1.4800850
7. Tucker B. J., Diaz A. A., Eckenrode B. A. Advanced ultrasonic measurement methodology for non-invasive interrogation and identification of fluids in sealed containers // Nonintrusive Inspection, Structures Monitoring, and Smart Systems for Homeland Security. 2006. doi: 10.1117/12.660439
8. Hilbert-Huang transform and its Applications / N. E. Huang, S. S. P. Shen (Eds.). CRC press, 2005. 324 p. doi: 10.1142/9789812703347
9. The Hilbert-Huang transform in Engineering / N. E. Huang, N. O. Attoh-Okine (Eds.). CRC Press, 2005. 328 p. doi: 10.1201/9781420027532
10. Lu Y., Oruklu E., Saniie J. Application of Hilbert-Huang transform for ultrasonic nondestructive evaluation // 2008 IEEE Ultrasonics Symposium. 2008. doi: 10.1109/ultsym.2008.0365
11. Derhunov O., Kuts Yu. Sposib adaptyvnoi mediannoi filtratsiyi impulsnykh syhnaliv: Pat. No. 103513 UA. MPK G06F 7/02 / zaiavnyk ta patentovlasnyk Nats. aviats. un-t. No. u201504262; declareted: 30.04.2015; published: 25.12.2015, Bul. No. 24.

Пропонуються циклічні коди, що ітеративно декодуються, і які можна розглядати як альтернативу турбо-кодам та LDPC-кодам. Ці коди основані на каскадному поєднанні двох різних циклічних кодів Хеммінга. Для (n, k)-коду виправляються всі помилки кратності до (n–k). Кодова швидкість ИДЦК наближається до одиниці з ростом довжини коду. Використовуються тільки жорсткі рішення, завдяки чому досягається висока швидкодія та проста апаратно-програмна реалізація кодера і декодера

Ключові слова: ітеративне декодування, циклічні коди, коди Хеммінга, лінійна послідовнісна схема, перемежування

Предлагаются итеративно декодируемые циклические коды (ИДЦК), которые можно рассматривать как альтернативу турбо-кодам и LDPC-кодам. Эти коды основаны на каскадном соединении двух различных циклических кодов Хэмминга. Для (n, k)-кода исправляются все ошибки кратности до (n–k). Кодовая скорость ИДЦК приближается к единице с ростом длины кода. Используются только жесткие решения, благодаря чему достигается высокое быстродействие и простая аппаратно-программная реализация кодера и декодера

Ключевые слова: итеративное декодирование, циклические коды, коды Хэмминга, линейная последовательностная схема, перемежение

UDC 681.32
DOI: 10.15587/1729-4061.2018.123207

ITERATIVE HARD-DECISION DECODING OF COMBINED CYCLIC CODES

V. Semerenko
PhD, Associate Professor
Department of computer technique
Vinnytsia National Technical University
Khmelnytske highway, 95,
Vinnytsia, Ukraine, 21021
E-mail: VPSemerenko@ukr.net

1. Introduction

The vast majority of error correction codes, which are widely applied at present, were developed in the 50's and 60's of the last century, immediately after the publication of the seminal paper by C. Shannon on coding theory [1]. These codes did not require any sophisticated algorithms for transforms, which is why they were relatively easily implemented. The only exception was the low-density parity-check (LDPC) codes: the level of development of computer technology in those years did not allow their implementation [2].

Over the following decades, the theory of error correction coding evolved and developed without much innovation. The next significant achievement was the invention of turbo codes [3]. The key innovation to these codes was the idea of iterative decoding, which, in a slightly different

form, had been employed in LDPC codes. The improved capabilities of digital technology have made it possible to fully implement the principles of LDPC encoding.

Since the mid 90-ies of the past century and up until our time, turbo codes and LDPC codes have been widely using in various data transmission systems; in fact they have proven to be the leading codes.

It was only natural that these codes have attracted attention of specialists around the world. Detailed studies have shown that LDPC codes and turbo codes, in addition to their indisputable advantages, have numerous imperfections, which did not allow them to push aside the previously developed codes.

This means that the theory of error correction coding did not end on LDPC codes and turbo codes. Still relevant is the task on developing new methods for encoding and detection of errors, as well as their effective practical implementation in different areas of science and technology. New scientific results will also help known codes to eliminate their drawbacks.

2. Literature review and problem statement

In the theory of error correction coding, almost any new idea will be definitely is compared to the current leaders of publications: LDPC codes and turbo codes. Therefore, we shall start analysis of the scientific literature with these codes.

Almost all publications on LDPC codes and turbo codes necessarily contain a reminder of the main achievement of the specified codes: maximal proximity to the theoretical Shannon limit (border) (for example, [4]). The current record for LDPC codes is 0.0045 dB, and 0.2 dB for turbo codes [5]. This is unconditionally an important characteristic of codes indicating their maximum utilization of communication channel capacity. But is such an achievement a sufficient reason to consider that these codes are the best? For example, the Reed-Solomon codes are far from the Shannon limit, which is not an obstacle for their wide application in different spheres.

The fundamental laws of nature and technology have repeatedly demonstrated that any advantages are balanced by certain shortcomings. If LDPC codes and turbo codes have managed to approach such close to the Shannon limit, at what cost was this achieved? It is known that error correction codes have many other characteristics that are important from a practical point of view. These include code redundancy, detecting and error-correcting capabilities, complexity of software and hardware implementation, code length, delay in obtaining the decision, the possibility of parallelization of computations, etc.

That is why, as indicated in [6], "the Shannon limit is an interesting border from a theoretical point of view, but it is not a practical objective."

In order to correctly estimate the error correction codes, we shall return to the origins of their theory, contained in the well-known theorems by C. Shannon [1]. The distinctive feature of these theorems is that they are based on the idea of random encoding. According to Shannon, the best code is the code that sends a message over an infinitely long time, forming at each point in time random bits of code words. An infinitely long message transmission time is equivalent to its very large length. It is this principle that underlies the creation of LDPC codes and turbo codes.

Then the first distinctive feature of the considered codes becomes comprehensible – the use of codes with huge lengths (up to 10^7 and longer) with the weight distribution, close to the distribution of random variables. In practice, such great lengths are often not needed. For example, in the ATM packet transmission protocols is used the check of the headlines that are 5 bytes long, and the check 48 bytes of useful information [7].

In addition, there are problems related to processing huge arrays of encoded data, complex hardware implementation, low error correction capacity of (n, k)-codes at an average speed of code of k/n=0.5, a long delay in decoding. Each code has its own specific drawbacks too.

That is why most researchers continue to examine and improve classic procedures for decoding turbo codes and LDPC codes. In [8], authors proposed a modification of the bidirectional iterative Viterbi algorithm with a probabilistic solution (BI-SOVA) for decoding the turbo codes. Paper [9] estimated the effectiveness of several techniques for creating a sparse matrix of H of LDPC code for the probabilistic Sum-Product Algorithm (SPA) when transmitting by Gaussian channel.

However, it is the irregular structure of the H matrix that is responsible for most of the drawbacks in classic LDPC codes. The main efforts of researchers are aimed now at developing regular (algebraic) LDPC codes [10]. Most often, quasi-cyclic LDPC codes act as algebraic codes [11]. The main distinctive property of these codes is that the following code word is obtained by the m-bit cyclic shift of another code word, where m is an integer. This type of code is known as a circulant code defined by a circulant polynomial. Thanks to the property of a cyclic shift, it is possible to use for encoding and decoding very simple shift registers and to accelerate the process of encoding. As a result, many algebraically-constructed LDPC codes were adopted as industry standards, specifically (64800, 48600)-LDPC code is applied in the digital TV standard DVB-S2 [12].

In contrast to the turbo-codes, it is possible to efficiently utilize the parallel processing for LDPC codes, which significantly enhances their performance [13].

Close in its ideology to LDPC encoding and turbo encoding is the multithreshold decoding (MTD) based on the majority-logic correction procedures [14]. The advantages of MTD include high-speed performance at minimum hardware cost. However, the application of MTD is limited to the codes with a low correction capability, which is why it is possible to apply it as a basic algorithm, for example, in concatenated codes.

The theory of iterative encoding based on various mathematical apparatus (finite fields, finite geometry, combinatorial methods) continues to develop intensively. [15]. Results obtained for binary Galois fields can be generalized for the nonbinary Galois fields [16].

Despite the noted positive properties of regular LDPC codes, there are also drawbacks to these codes. The most significant problem of algebraic (quasi-cyclic) codes that have high code distance is their poor iterative convergence [10]. That is why such codes are recommended only for small lengths of codes (n<350).

Continuing a research in this in direction it is possible to do a conclusion about the appropriateness of using classic cyclic codes in iterative decoding. Indeed, such codes have been investigated in recent years. But the paradox is that traditional iterative approaches to finding errors are applied

to the cyclic codes. For example, soft decisions for Hamming codes [17], improvement of irregular LDPC decoder using CRC codes [18], two-step algorithms with majority logic for cyclic codes with the finite geometry [19].

In the end, most of the earlier identified drawbacks of these codes are still there: high functional complexity (at soft decoding), low code rate, low correcting capability, the "error floor" effect.

It is obvious that the very ideology of LDPC codes and turbo codes imposes restrictions on those code characteristics that are important for practical application. Therefore, it is necessary to apply other algorithms of code transforms for cyclic codes while remaining on the platform of iterative (multistage) decoding.

3. The aim and objectives of the study

The aim of present work is the development of deterministic iterative decoding of block (cyclic) codes with high performance efficiency and minimal hardware costs based on the mathematical apparatus of linear finite-state machines (LFSM).

To accomplish the set aim, the following tasks must be solved:

– to explore the essence of iterative decoding of LDPC codes and turbo codes and show the possibility of alternative iteratively decodable code;

– to show the possibility of iterative decoding of cyclic codes based on hard decisions and the theory of LFSM;

– to develop a generalized iterative decoding algorithm of cyclic codes and estimate its time and resource complexity;

– to propose means that would be effective for practical implementation of the iterative decoding of cyclic codes.

4. Substantiation of trends in the development of error correction codes

The Shannon theorems warrant the existence, under certain conditions, of such a encoding procedure, which enables data transfer at an arbitrarily small error probability. At the same time, the theorems do not specify the methods to build specific codes to ensure such a perfect data transfer. The theorems only help calculate a mathematical code efficiency criterion by the following principle: the closer a code to the Shannon border (limit), the better the code.

First error correction codes were far from the assigned ideal. An opinion that had gradually formed implied that should such a perfect code exist in theory, it could not be implemented in practice.

The first significant breakthrough from this situation was the idea of concatenation, that is, combined application of several, typically two, codes. Concatenated codes [20] have made it possible to obtain very long codes, with a relatively high correcting capability. However, in this case, the compromise meant a code rate reduction and a larger complexity of the decoding process.

The second, very important, innovation in error-correcting coding theory was the iterative (multistage) decoding. This encoding technique simplified computations, however, at the expense of a substantial delay of decoding procedure.

The idea of iterative decoding is complex for practical implementation. That is why the first iteratively decodable codes (LDPC codes) had been postponed for three decades before they attracted attention of engineers. The iterative approach as it is was initially applied only to convolutional codes (Viterbi algorithm), or at separate stages of decoding of block codes (for example, Berlekamp-Massey algorithm in the procedure of algebraic decoding of cyclic codes).

A logical result of the further development of error correction codes was the union of iterative decoding and concatenation in the new class of codes, known as "turbo-codes". Quite unexpectedly, the turbo-codes had proven to be very close to the desired Shannon limit.

The LDPC codes had been almost immediately recalled; the progress of microelectronics has allowed their practical implementation. These codes became several orders of magnitude closer to the Shannon limit.

However, every step towards this border contributed to increasing the time of encoding-decoding and to complicating the structure of encoder and decoder.

That is why, since the mid-1990s, intensive research were began for resolving this problems. The first problem is related to the complexity of computations: LDPC codes involve complex encoding, turbo codes – complex decoding.

Note that there are many error correction codes with theoretically substantiated methods that accurately detect and correct the errors. However, the number of errors which corrected by the precise methods of correction (that is, using the hard decision) is very limited. According to [21], the correction of more than six errors in a single code word of the Reed-Solomon code at a speed of 40 Mbps is almost unrealize in practice.

That was the reason for the emergence of probabilistic models and soft computing. Soft computing makes it possible to improve the accuracy of decoding, but under condition of the availability of statistical characteristics of the communication channel. Probabilistic data processing is associated with the use of real variables which require complex processor calculations. This is the main reason for the complexity of computations in LDPC codes and turbo codes.

Soft computing uses an additional (non-mathematical) "prompt" on the state of the transmitted code word. Such a "prompt" can be information from a demodulator on the reliability of decoding of separate symbols of the code word. The implementation of such a technique for decoding is based on the most important principle of classic theory of error correction coding, the concept of Maximum Likelihood.

A strategy of maximum likelihood decoder is to produce a list of possible codewords and selecting the most "likelihood able", that is, at a minimum code distance from the transmitted code word (Chase Algorithm [5]). However, this choice is performed only by using soft computing.

The Chase algorithm, as well as similar methods, make use in one form or another of the assumption that random errors with less multiplicity are more likely, "likelihood able", than the errors with greater multiplicity. However, this argument is correct for very simple models of channels only.

During early development of error correcting coding theory when both hardware and temporal resources were limited, the concept of Maximum Likelihood was optimal. When resources became available, and it is necessary to reduce the number of incorrect decoder decisions, then it is possible to "allow" it to check more variants before making a final decision. This is where additional iterations may come in handy.

Two types of errors most often occur in code words. "Random" errors are evenly distributed along at full length of a word and are statistically independent from each other. "Burst errors" are concentrated in one region and are statistically dependent. Only random errors were initially consideration, those can be placed at full length of a code word, while burst errors considerated only as a set of random errors. Such a model of errors was typical for the satellite communication in 50s and 60s of the 20th century.

These positions underlie the creation of the error correcting coding theory, beginning from the Shannon theorems and the concept of minimum code distance.

When different error correction codes emerged, there was a need to compare their capabilities. The most popular criterion for estimating codes was the magnitude of distance to the Shannon limit (E_b/N_0=–1.6 dB) on the BER (bit error rate) curves that show dependences of the probability P_b of the emergence of an erroneous bit in a code word on the ratio of bit energy E_b to the Noise Spectral Density ratio N_0: E_b/N_0 (Fig. 1). This characteristic is rather good for comparing error-correcting capabilities of codes, but only for the class of random errors: the more of such errors corrected, the closer the code's curve to the Shannon limit.

Very typical is the situation with the Fire cyclic codes at BER curve. Fig. 1 shows curves only for three Fire codes, though one could show them for all other codes for a given class, but the overall pattern would remain unchanged: all curves merge almost in one line. Each Fire *(n, k)*-code at a growth of parameters n and k corrects an increasing number of erroneous bits of a code word, but it does not affect the decreasing distance to the Shannon limit. The explanation to this phenomenon is simple: in the Fire codes only the length of corrected burst errors increases while the number of corrected random errors is limited only by single errors [22].

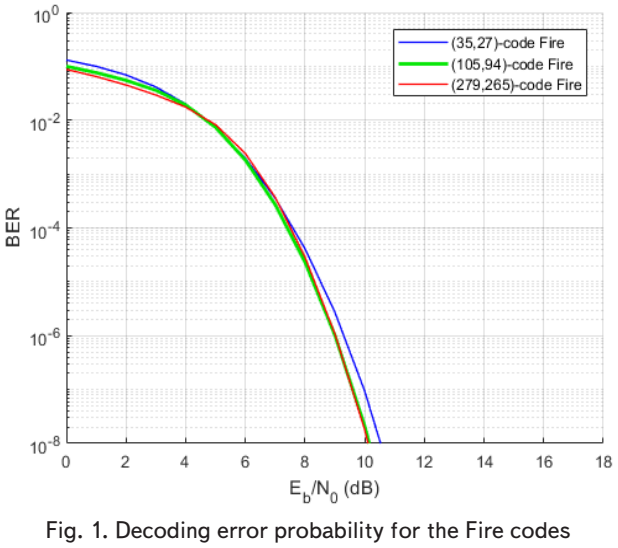

Fig. 1. Decoding error probability for the Fire codes

This is the essence of the strategy for correcting of errors by Shannon – to eliminate burst errors and reduce various distortions only to the statistically independent (random) errors. It is for this purpose interleavers often employ in the error correcting coding. LDPC codes are also based on the statistically independent (random) errors.

However, the burst errors should not be avoided, at least for two reasons.

First, burst errors are not the compact grouping of several erroneous bits of a code word only. The paradox is that such type of errors are much easier to detect and correct than separate erroneous bits far locate from each other. The exception is only very long burst errors (with a length greater than $n–k$).

Second, burst errors are the most accurate mathematical model of distortions, characteristic of wireless and mobile communications [6].

Thus, the reasons for most of the problems related to LDPC codes and turbo codes are that these codes are largely oriented towards the features of the development of the communication of the middle of the mid-20th century.

Therefore, it is necessary to develop the new approaches to error correction decoding that would improve performance and error-correcting capabilities at minimal hardware costs. Solving these tasks is possible on the platform of the iterative decoding of concatenated codes, which has proven its indisputable advantages.

5. Theoretical fundamentals of cyclic code decoding based on the automaton models

A key provision of the Shannon communication theory is the use of random codes, the probabilistic mode of encoding and decoding. This approach requires considerable time and hardware costs.

It is possible to eliminate these problems based on integer arithmetic, that is, by employing hard computing in the Galois fields. We shall confine ourselves to the binary Galois fields, yet the results obtained are easily generalized to the nonbinary Galois fields as well.

Computation in such fields immediately eliminates the problem on performance and hardware costs: shift registers in cyclic codes have long been proven to be effective. Also resolved is the requirement for super large lengths of codes: work results are therefore produced faster.

We shall apply as a mathematical model of cyclic codes the automaton model [23], which is based on the theory of linear finite-state machine (LFSM). According to [24], LFSM is a linear automaton with l inputs, m outputs and r memory cells, which is determined by the state (transition) function

$$S(t+1)=A\times S(t)+B\times U(t),\ \ GF(2) \qquad (1)$$

and by the output function

$$Y(t)=C\times S(t)+D\times U(t),\ \ GF(2),$$

where

$$A=\left|a_{ij}\right|_{r\times r},\ \ B=\left|b_{ij}\right|_{r\times l},\ \ C=\left|c_{ij}\right|_{m\times r}$$

and

$$D=\left|d_{ij}\right|_{m\times l}$$

are the characteristic matrices of LFSM;

$$S(t)=\left|s_i\right|_r$$

is the word of state;

$$U(t)=\left|u_i\right|_l$$

is the input word;

$$Y(t)=\left|y_i\right|_m$$

is the output word.

We shall subsequently apply the following matrices as characteristic matrices of LFSM in (1):

$$A=\begin{vmatrix} 0 & 0 & 0 & \dots & g_0 \\ 1 & 0 & 0 & \dots & g_1 \\ 0 & 1 & 0 & \dots & g_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 & g_{r-1} \end{vmatrix},\quad B=\begin{vmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 0 \end{vmatrix},$$

$$C=\left|0 \quad \dots \quad 0 \quad 1\right|,\quad D=\left|0\right|. \tag{2}$$

The entries of the last column of matrix A in (2) represent coefficients of the generator polynomial of a cyclic code:

$$g(x)=g_0+g_1x+g_2x^2+\dots+g_{r-1}x^{r-1}+g_rx^r,\ GF(2). \tag{3}$$

Selection of characteristic matrices of the r-dimensional LFSM is performed based on requirement for r-controllability of LFSM. As proven in [24], LFSM will become r-controllable if the rank of $r\times r$-matrix

$$L_r=\left[A^{r-1}\times B,\quad A^{r-2}\times B,\quad \dots\ ,\quad A\times B,\quad B\right] \tag{4}$$

will be equal to r. For matrices (2), the L_r matrix contains "1" only in the secondary diagonal:

$$L_r=\begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{5}$$

We shall distinguish between the analytical and the graph models of the automaton model.

The automaton-analytical model is based on the characteristic matrices of LFSM. Based on it, it is possible to give a new definition for a cyclic code.

Definition 1. A set of all binary sequences M of length n, which transform LFSM from any initial state $S_{beg}(t)$ back to state $S_{beg}(t)$ creates a cyclic (n, k)-code Ω over Galois field $GF(2)$. Each such sequence M is a code word Z of the cyclic (n, k)-code.

One can choose, as an automaton-graphical model of a cyclic code, the state transition diagram G_{FA} of LFSM.

Definition 2. Sequence M of n unidirectional edges in the G_{FA} graph, in which the i-th edge corresponds to bit z_i of code word Z over field $GF(2)$, is termed the code path η of graph G_{FA} ($z_i\in Z,\ i=1\div n$).

Definition 3. A set of all code paths η of length n, which originate and end in the initial vertex v_0 of the G_{FA} graph, generates a cyclic (n, k)-code Ω over Galois field GF(2).

We shall consider the essence of operations of systematic encoding and syndrome decoding of cyclic codes based on the automaton models.

From the perspective of systematic encoding, n-bit code word Z contains in k high position an information word I, and in r ($r=n-k$) low position – a check word Ψ:

$$Z=I\Psi=\iota_1\iota_2\dots\iota_k\psi_1\psi_2\dots\psi_r=z_1z_2\dots z_n. \tag{6}$$

When data are transmitted over a communications channel, various disturbances may lead to the distortion of τ positions of code word Z, and then a code word Z_{err} may be obtained with a multiplicity error τ.

A decoder must sequentially perform two tasks:

– to establish the absence or presence of errors in the accepted code word (decoding task);

– in case there are errors, to identify distorted positions of the code word and perform the appropriate correction (error correction task).

Both tasks will be implemented based on the analysis of syndromes [25]. A syndrome in the automaton-analytical model refers to the state that will be reached by LFSM from the initial state (we shall accept it being equal to zero S(0)) under the influence of the code word of a cyclic code. Computing the syndrome is performed during n cycles according to formula (1), in which the code word acts as input word $U(t)$.

If state $S(n)$ turns out to be zero ($S(n)=S(0)$), this would indicate the absence of errors in the accepted code word within the code's correcting capability. If the decoder input receives a distorted code word Z_{err} then the n-th state of LFSM will be the syndrome of occurred error ($S(n)=S_{err}(n)$).

We shall introduce the concept of check window as a continuous cyclic sequence of r positions of the code word (6) (Fig. 2). We shall denote the check window as $X^{(i)}$ if its rightmost position is the i-th position of code word Z:

$$X^{(i)}=z_{i-r+1}\dots z_{i-1}z_i,\quad z_j\in Z,$$

$$j=i\div(i-r+1)\bmod n,\quad i=1\div n.$$

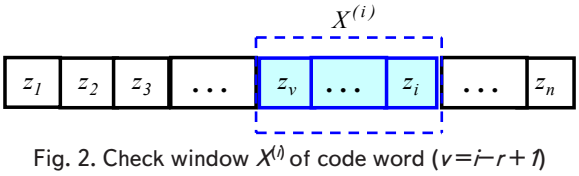

Fig. 2. Check window $X^{(i)}$ of code word ($v=i-r+1$)

Similarly, we shall denote the check window as $X^{(i)}_{err}$, if its rightmost position is the i-th position of code word Z_{err} and there may occur errors in some positions:

$$X^{(i)}_{err}=z_{i-r+1},\dots,z_{i-1}z_i,\quad z_j\in Z_{err},$$

$$j=i\div(i-r+1)\bmod n,\quad i=1\div n.$$

The task on error correction in code word Z_{err} will be considered in two variants: error correction within one Z_{err} r-bit check window, and error correction within the entire code word.

We shall prove the following theorems.

Theorem 1. If codeword Z_{err} contains errors within check window $X^{(n)}_{err}$, then in order to correct such errors and obtain check window $X^{(n)}$, it is necessary to perform a bitwise operation

$$X^{(n)}=X^{(n)}_{err}+(-S_{err}(n)),\quad GF(2), \tag{7}$$

where the sign (–) means inverting the word $S_{err}(n)$, that is, mutual permutation of its positions in line with rule:

$$s_i = s_{r-i},\ \ s_i \in S_{err}(n),\ \ i = 1 \div r.$$

Proof. As shown in [23], under systematic encoding, LFSM from the initial zero state under the influence of word I passes at the k-th step into intermediate state

$$S(k) = A^k \times S(0) + L_k \times I,\ \ GF(2),$$

then, under the influence of word Ψ, passes into final state

$$S(n) = A^r \times S(k) + L_r \times \Psi,\ \ GF(2). \qquad (8)$$

If there are no errors, the state $S(n)$ is equal to zero ($S(n)$=$S(0)$), and equality (8) can be written in the following form

$$A^r \times S(k) = L_r \times \Psi,\ \ GF(2). \qquad (9)$$

Assume there are errors in some positions of the check word: we shall denote this word as Ψ_{err}. Then, at the n-th step, one obtains a non-zero error syndrome, that is, $S(n)\neq S(0)$, and equality (8) takes another form

$$S(n) = S_{err}(n) = A^r \times S(k) + L_r \times \Psi_{err},\ \ GF(2). \qquad (10)$$

After substituting in (10) the value of $A^r \times S(k)$ from (9), we obtain equality:

$$S_{err}(n) = L_r \times \Psi + L_r \times \Psi_{err} = L_r(\Psi + \Psi_{err}),\ \ GF(2).$$

One can record with respect to matrix L_r:

$$-S_{err}(n) = \Psi + \Psi_{err},\ \ GF(2). \qquad (11)$$

Because the r-bit check words Ψ and Ψ_{err} occupy positions from z_{n-r+1} to z_n in code words Z and Z_{err}, respectively, hence they are located in check windows $X^{(n)}$ and $X^{(n)}_{err}$. Hence, the desired equality (7) follows from equality (11).

Consider a general case when a check window of length r can be located in any place of a code word.

Theorem 2. If code word Z_{err} contains errors within check window $X^{(i)}_{err}$, then, in order to correct such errors and obtain check window $X^{(i)}$, it is necessary to perform a bitwise operation

$$X^{(i)} = X^{(i)}_{err} + (-S_{err}(n+i)),\ \ GF(2). \qquad (12)$$

Proof.

The proof of Theorem 2 is continuation of the proof to Theorem 1.

Multiply all terms of equation (11) by the i-th degree (i=1÷n) of matrix A:

$$A^i \times (-S_{err}(n)) = A^i \times \Psi + A^i \times \Psi_{err},\ \ GF(2). \qquad (13)$$

Multiplier $A^i \times (-S_{err}(n))$ denotes condition $-S_{err}(n+i)$, which LFSM will transit from state $-S_{err}(n)$ after i zeros will be entered to its inputs. Multiplier $A^i \times \Psi$ denotes a cyclic subword, which occupies positions from z_v to z_i in code word Z_{err}, that is, it is in check window $X^{(i)}$ (v=(i–r+1) *modn*). Multiplier $A^i \times \Psi_{err}$ denotes a check window $X^{(i)}_{err}$ in the same positions, but some of the positions may contain an error.

Then equation (13) can be written in the following form

$$-S_{err}(n+i) = X^{(i)} + X^{(i)}_{err},\ \ GF(2)$$

hence, the desired equality (12).

6. Permutation decoding of cyclic codes

The main idea of the decoding method, considered in the previous chapter, is to find in the check window a configuration of error that matches the syndrome of this error. Such an algorithm is also termed an "error-trapping" algorithm [26]. A given algorithm is easy to implement in practice, however, it is only applicable for codes of small length and low multiplicity of errors. Such codes are called in [23] the easily decodable codes, and the syndromes of an error that match the configuration of the error – the regular states. In a general case, a cyclic code contains both regular and irregular states of an error; the latter in much larger quantity.

A simple and effective technique for finding a match between syndrome $S_{err}(n)$ and the erroneous bits of code word Z_{err} is the power permutation decoding.

It is known [27] that for any integer ν cyclic codes are invariant relative to the permutations of symbols of form

$$i \to (i + \upsilon) \bmod n,\ \ GF(q^m), \qquad (14)$$

$$i \to (2^{\upsilon} i) \bmod n,\ \ GF(q^m). \qquad (15)$$

In other words, if a generator polynomial $g(x)$ of the cyclic code divides code polynomial $f(x)$, then it will also divide the polynomial $f(x^q)$ whose symbols are rearranged in accordance with rule $i \to q^i$. Therefore, polynomial $f(x^q)$ will also be a code polynomial, and, if no errors occur, as a result of dividing $f(x^q)$ by $g(x)$, we shall obtain a zero syndrome. But in the case when there are errors, the result of dividing $f(x)$ by $g(x)$ will yield a single configuration of erroneous positions of the code word, and the result of dividing $f(x^q)$ by $g(x)$ is a completely different configuration. In [28], such a technique for permutations is referred to as the "decimation".

We shall apply a cyclic power permutation of the form $i \to (2^v i) \bmod n$ (or $i \to (2^{-v} i) \bmod n$), which is equivalent to multiplying the corresponding exponent by 2^v or 2^{-v}. In the simplest variant, power permutation implies that first one records the odd positions of code word Z, and then the even-numbered (it is possible to start with the even positions). As a result, we obtain the new code word Z_v (Fig. 3).

The essence of the method for correcting multiple errors employing a method of power permutation is to build, at each step, the new variant of permutation of the form (14) or (15), to compute the new syndrome, and to detect errors in check windows. Upon error correction in the code word with permutation, it is necessary to perform the inverse power permutation of the word to produce the original code word.

Because a given computation process is repeated many times, in fact we observe here iterative decoding: each iteration has its own variant of permutations.

It is not difficult to realize the meaning of cyclic power permutation. First, the central bits of a code word are shifted towards the edges, and the extreme positions – towards the center. Next, the motion direction changes. Except for the

first position (or two extreme positions at even n) only, all other positions are mixed. As a result, the distorted positions of word Z_{err} are caught during one of the iterations in the check window and are detected.

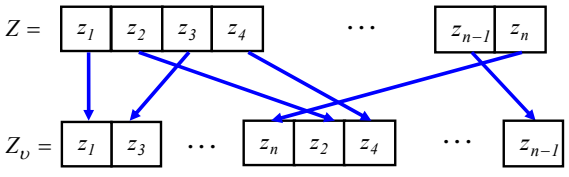

Fig. 3. Power permutation of code word Z (at odd-number n)

Such an operation can also be called the interleaving. In contrast to traditional interleaving when positions of one code word are redistributed by several code words, here the interleaving occurs within a single code word. This allows saving the time for the preparation of transmitted data, since the data transmission party is not involved in the interleaving.

The task of an interleaver in a traditional iterative code (for example, turbo-code) is to replace burst errors with random errors. In the proposed interleaver, the opposite task is performed – random errors are grouped into packets (burst error) in order to get into a decoding window at one of the interleaving steps.

Another important distinguishing feature implies that there is no difference between random and burst errors, since at each iteration of interleaving their role changes with all of them processed by one algorithm.

7. Generalized algorithm for iterative decoding of cyclic codes

Source data for the generalized algorithm for iterative decoding are the accepted n-bit code word Z_f, consisting of k-bit information word I_f and r-bit check word Ψ_f, and reference checksum Σ_{ref}. If the check word is intended to correct errors, the checksum is used to establish the presence or absence of errors.

Two different LFSM are used on the sender and the receiver sides. The first (master) LFSM is applied for encoding and decoding (with error correction) a code word. The second (slave) LFSM acts to confirm the presence or absence of an error. Accordingly, the master and slave cyclic codes are used with two different generator polynomials (3).

The essence of the generalized algorithm is an iterative forming, using master LFSM, of all possible variants of code words, and in searching for an error-free word among them by employing slave LFSM.

Stage 1. The actual checksum is computed as the state $S(k)$ to which slave LFSM passes from the initial zero state $S(0)$ when a word I_f enters its input. According to (1), the following actions must be performed for i=1,... ,k:

$$S(i+1)=A_2\times S(i)+B_2\times z(i),\ \ GF(2),\ \ z(i)\in I_f,$$

where A_2 and B_2 are the characteristic matrices of slave LFSM.

Then $\Sigma_f = S(k)$.

A match between the actual checksum Σ_f and the reference checksum Σ_{ref} can be regarded as a sufficient basis to confirm the absence of errors in word I_f and termination of the algorithm. Otherwise, the data are prepared to begin an iterative error search: the iteration number is set as w=1, permutation power as v=0, and code word Z_f is accepted as the current code word $Z_{cur}^{(w)}$:

$$Z_{cur}^{(w)}=Z_f.$$

Stage 2. The actual error syndrome $S_{err}(n)$ is computed as the state $S(n)$, which master LFSM enters from the initial zero state $S(0)$ when the current code word $Z_{cur}^{(w)}$. enters its input. According to (1), the following actions must be performed for i=1,..., n:

$$S(i+1)=A_1\times S(i)+B_1\times z(i),\ \ GF(2),\ \ z(i)\in Z_{cur}^{(w)},$$

where A_1 and B_1 are the characteristic matrices of master LFSM.

Then $S_{err}(n)=S(n)$. If there are errors in $Z_{cur}^{(w)}$, state $S_{err}(n)$ would be nonzero.

Stage 3. The error words $E_1^{(w)},\ldots,E_\mu^{(w)}$ are formed for μ possible configuration of errors at multiplicity from 1 to r. The following computations are performed for this purpose

$$S''(0)=S_{err}(n),$$

$$S''(n+i)=A_1\times S''(n+i-1),\ \ GF(2),\ \ i=1,\ldots,n-1. \qquad (16)$$

Word $E_j^{(w)}$ is equal to the sum of the n-bit zero word O and r-bit state $S''(j)$ of LFSM, which should be located, relative to word O, in a cyclic interval from the i-th position to the $((i-r+1)\bmod n)$-th position:

$$E_j^{(w)}=O+S''(j),\ \ GF(2),\ \ j=1,\ldots,\mu.$$

One more condition: word $S''(j)$ must contain "1" in the same position as the matrix B, then word $E_j^{(w)}$ will contain "1" in the i-th (i=1÷n) position (which is easy to check). Such conditions are fulfilled by $\mu=(n+1)/2$ error words if we choose a primitive polynomial as the generator polynomial of the master code.

Stage 4. μ variants of the current word correction $Z_{cur}^{(w)}$ are computed:

$$Z_j^{(w)}=Z_{cur}^{(w)}+E_j^{(w)},\ \ GF(2),\ \ j=1...\mu. \qquad (17)$$

Stage 5. If we have previously completed power permutation $2^v>1$ of code word $Z_{cur}^{(w)}$, then a reverse power permutation 2^{-v} is performed for all corrected code words from (17).

Stage 6. The correctness of correction of code words from (17) is checked by computing a checksum $\Sigma_j^{(w)}$ for each word $Z_j^{(w)}$.

Stage 7. If for one value of j the checksum $\Sigma_j^{(w)}$ coincided with reference value Σ_{ref}, then word $Z_j^{(w)}$ is the correct value (within the correcting capability of a cyclic code) of the accepted code word Σ_f. This completes the algorithm.

Otherwise, if one has not exhausted the limit of power permutation quantity for a given code, the iteration number is increased (w=w+1), as well as the power of permutation (v=v+1), and a new current code word $Z_{cur}^{(w)}$ is created:

$$Z_{cur}^{(w)}=Z_{cur,v-1}^{(w-1)},$$

where $Z_{cur,v-1}^{(w-1)}$ is the power permutation 2^v of word $Z_{cur}^{(w-1)}$ at the (w-1)-th iteration. Next, one returns to stage 2.

The end.

8. Example of iterative decoding

We shall employ as an example of iterative decoding the master cyclic (15,11)-code by Hamming with generator polynomial $g_2(x)=1+x+x^4$, and a slave cyclic (31,26)-code by Hamming with generator polynomial $g_1(x)=1+x^3+x^5$. These polynomials are matched with master LFSM with matrices

$$A_1=\begin{bmatrix}0&0&0&1\\1&0&0&1\\0&1&0&0\\0&0&1&0\end{bmatrix},$$

$$B_1=\begin{bmatrix}1\\0\\0\\0\end{bmatrix},$$

and slave LFSM with matrices

$$A_2=\begin{bmatrix}0&0&0&0&1\\1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&1\\0&0&0&1&0\end{bmatrix},$$

$$B_2=\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}.$$

Assume that the encoder on the sender side formed a code word Z. Due to interference in the communication channel, the receiver received codeword Z_f with errors:

$$Z_f=z_1z_2z...z_{15}=100000101001110. \quad (18)$$

The receiver also received the reference checksum which computed for 11 correct information bits (bits $z_1z_2z_3...z_{11}$) of code word Z using slave LFSM:

$$\Sigma_{ref}=\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}. \quad (19)$$

Starting with word (18) and further the starting erroneous positions and their displacements in the execution of the generalized algorithm are shown in red.The decoder, of course, is "not aware" of this fact.

According to the generalized algorithm, first the actual checksum Σ_f is computed. This check parameter is equal to state $S(11)$ of slave LFSM when information bits $z_1z_2z_3$ ... z_{11} from (18) enter to its inputs.

$$S(0)=\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix};$$

$$S(1)=A_2\times S(0)+B_2\times z_1=A_2\times\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}+B_2\times[1]=\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}\dots.$$

As a result, we obtain the actual checksum

$$\Sigma_f=S(11)=\begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix},$$

which does not match the referen ce checksum (19). Therefore a transition to the procedure of error correction in the code word (18) is carried out.

At the second stage of the error correction procedure, an error syndrome $S_{err}(15)$ is computed as the value of state $S(15)$ of master LFSM when a 15-bit code word (18) enters to its inputs. The result is the following error syndrome:

$$S_{err}(15)=S(15)=\begin{bmatrix}0\\1\\1\\1\end{bmatrix}.$$

At stage 3 of the algorithm, the following computations are performed (16):

$$S''(15)=S_{err}^{(\tau)}(15)=\begin{bmatrix}0\\1\\1\\1\end{bmatrix};$$

$$S''(16)=A_1\times S''(15)=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}....$$

All computed words of states $S''(i)$ and the error words that correspond to them are given in Table 1 (blue color denotes check windows). To decoding it will suffice to use only those error words that contain "1" in the i-th bit. Remaining error words (denoted as E_0) are not involved in further calculations because they are the shifted copy of other error words.

At stage 4, it is necessary to compute 8 variants of the corrected current word $Z_{cur}^{(1)}$ according to (17). Results of computations are given in Table 2. Table 2 shows that the erroneous position did not appear in any of eight check windows. Therefore, the decoder could not correct them, moreover, it introduced more errors after correction (shown in green).

Table 1

Calculated error words at iteration 1

State words	Numerical value	Error words (i = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15)
$S''(15)$	0 1 1 1	E_0 = 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
$S''(16)$	1 1 1 1	$E_1^{(1)}$ = 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1
$S''(17)$	1 0 1 1	$E_2^{(1)}$ = 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1
$S''(18)$	1 0 0 1	$E_3^{(1)}$ = 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
$S''(19)$	1 0 0 0	$E_4^{(1)}$ = 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
$S''(20)$	0 1 0 0	E_0 = 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
$S''(21)$	0 0 1 0	E_0 = 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
$S''(22)$	0 0 0 1	E_0 = 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
$S''(23)$	1 1 0 0	$E_5^{(1)}$ = 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
$S''(24)$	0 1 1 0	E_0 = 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
$S''(25)$	0 0 1 1	E_0 = 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
$S''(26)$	1 1 0 1	$E_6^{(1)}$ = 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0
$S''(27)$	1 0 1 0	$E_7^{(1)}$ = 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
$S''(28)$	0 1 0 1	E_0 = 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
$S''(29)$	1 1 1 0	$E_8^{(1)}$ = 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0

Next, it is necessary to check correctness of the performed corrections by computing the checksum similar to the procedure for computing an error syndrome. In this case, the checksum $\Sigma_j^{(1)}$ is equal to the state in which slave LFSM transitions from the initial zero state $S(0)$ when words $Z_j^{(1)}$ from Table 2 enter to its inputs.

Table 2

Variants of the corrected code words at iteration 1

Error words	Corrected code words
$E_1^{(1)}$	$Z_1^{(1)}$ = 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1
$E_2^{(1)}$	$Z_2^{(1)}$ = 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1
$E_3^{(1)}$	$Z_3^{(1)}$ = 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1
$E_4^{(1)}$	$Z_4^{(1)}$ = 1 0 0 1 0 0 1 0 1 0 0 1 1 1 0
$E_5^{(1)}$	$Z_5^{(1)}$ = 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0
$E_6^{(1)}$	$Z_6^{(1)}$ = 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0
$E_7^{(1)}$	$Z_7^{(1)}$ = 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0
$E_8^{(1)}$	$Z_8^{(1)}$ = 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0

Computations reveal that none of the calculated checksums matches the assigned reference value (19). This confirms incorrect corrections of the code word (18). The decoder takes note of this fact and proceeds to the second iteration of decoding.

For the second iteration, one performs power permutation 2^1 of the obtained code word Z_f. As a result, the code word will be obtained and will be declared to be the current code word $Z_{cur}^{(2)}$ for the second iteration:

$$Z_{cur}^{(2)}=1001101000000011. \quad (20)$$

Next, all steps of the algorithm, starting at stage 2, are repeated, but for the code word (20). One can see from (20) that all erroneous positions are placed in a 4-bit check window. It only remains to shift this window and correct the code words, as shown in Table 3.

Next, one performs reverse power permutation 2^{-1} for all corrected code words, as shown in Table 4. In conclusion, correctness of the correction of code words is checked by computing checksum $\Sigma_j^{(w)}$ for each word $Z_j^{(w)}$ from Table 4. The check is successfully finished for code word $Z_{5,f}^{(2)}$, which is the correct code word Z.

Table 3

Variants of corrected code words at iteration 2

Error words	Corrected code words
$E_1^{(2)}$	$Z_1^{(2)}$ = 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1
$E_2^{(2)}$	$Z_2^{(2)}$ = 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1
$E_3^{(2)}$	$Z_3^{(2)}$ = 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1
$E_4^{(2)}$	$Z_4^{(2)}$ = 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1
$E_5^{(2)}$	$Z_5^{(2)}$ = 1 0 0 1 0 1 1 1 0 0 0 0 0 1 1
$E_6^{(2)}$	$Z_6^{(2)}$ = 1 0 0 1 1 1 1 0 1 0 0 0 0 1 1
$E_7^{(2)}$	$Z_7^{(2)}$ = 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1
$E_8^{(2)}$	$Z_8^{(2)}$ = 1 0 0 1 1 0 1 0 0 0 0 0 1 0 1

Table 4

Variants of restored code words at iteration 2

Corrected code words	Restored and corrected code words
$Z_1^{(2)}$	$Z_{1,cor}^{(2)}$ = 0 0 1 0 0 0 1 0 1 0 0 0 1 1 0
$Z_2^{(2)}$	$Z_{2,cor}^{(2)}$ = 0 0 0 0 1 0 1 0 1 0 0 1 1 1 0
$Z_3^{(2)}$	$Z_{3,cor}^{(2)}$ = 1 0 0 0 1 0 0 0 0 0 0 1 1 1 0
$Z_4^{(2)}$	$Z_{4,cor}^{(2)}$ = 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0
$Z_5^{(2)}$	$Z_{5,cor}^{(2)}$ = 1 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1
$Z_6^{(2)}$	$Z_{6,cor}^{(2)}$ = 1 1 0 0 0 0 0 1 0 1 0 1 1 1 1 0
$Z_7^{(2)}$	$Z_{7,cor}^{(2)}$ = 1 0 0 1 0 0 1 0 1 0 0 1 1 1 0

Note that by using the considered algorithm we were able to detect and correct the error of multiplicity τ=3 applying two Hamming codes. To this task, it took two of the four possible iterations for the Hamming (15,11)-code.

9. Discussion of the algorithm for iterative decoding

We shall consider main distinctive features of the proposed algorithm for iterative decoding.

1. First, we shall analyze the obtained coding gain G_a as a result of iterative decoding of cyclic codes. For this purpose, it is possible to employ either known BER curves from the mathematical software MATLAB, or approximate analytical estimate [21]:

$$G_a = 10 \lg[R(t+1)],$$

where $R=k/n$ is code rate of (n, k)-code; t is the number of corrected errors.

In both cases, the magnitude of coding gain G_a will increase in proportion to an increase in the number of iterations and the number of corrected errors, which is equivalent to approaching the Shannon limit (it is inherent for iterative decoding of any codes). And although the proposed iterative code under actual conditions will not be very close to the Shannon limit, it has a code rate, which, with an increase in the length of code, approaches one. This makes it possible to increase by almost two-fold the volume of transmitted useful data per time unit, and requires an insignificant improvement in bandwidth. We remind that turbo codes and LDPC codes have code rate R≈0.5 which requires either a two-fold bandwidth expansion or double transmission rate [6].

2. The length of the proposed code can be arbitrary: both small and large. This significantly extends the scope of its application.

3. In contrast to the decoding algorithm of turbo codes and LDPC codes which use the soft decisions, the proposed algorithm is based only on hard decisions. The advantages of a given approach is the high speed and simple hardware-software implementation of encoder and decoder.

4. Instead of "likelihood able" assumptions about the error positions in code word within the minimum code distance, the generalized algorithm checks all possible variants of the errors with a multiplicity $(n-k)$ of the cyclic (n, k)-code and finds exact solutions.

5. Power permutation performs interleaving inside single code word only, and this operation is carried out on the side of the receiver only.

6. For the objectively compare the proposed and the known iterative decoding algorithms, we shall estimate complexity of these algorithms regarding time and hardware implementation.

First, we shall consider the hardware implementation of encoder for the proposed algorithm (Fig. 4). An encoder of the generalized algorithm consists of two elementary encoders (LFSM 1 and LFSM 2), which are implemented as linear feedback shift registers (LFSR) based on generator polynomials, respectively, $g_1(x)$ and $g_2(x)$. First, we send by the channel a k-bit information word I, then a r_2-bit checksum Σ and, finally, a r_1-bit check word Ψ (Fig. 4). The total number of encoding cycles is n:

$n=k+2r_1$, if $r_1 \geq r_2$, or $n=k+r_2+r_1$, if $r_1<r_2$.

Thus, the time complexity of encoding for the proposed method is linear, that is $f_c^{\tau}(n)=O(n)$. Since the bit capacity of LFSM 1 and LFSM 2 with an increasing of code length grows very slowly, the complexity of hardware implementation of the encoder for the proposed iterative algorithm is constant: $f_c^h(n)=O(1)$. The complexity of hardware implementation of the decoder is more complex – quadratic, that is $f_d^h(n)=O(n^2)$.

It is widely accepted that known iterative codes have similar mathematical functions of complexity: from linear to quadratic. This does not mean, however, that all three analyzed iterative codes have circuit implementation of the same complexity. For example, to represent a single code bit in the proposed method, we use one bit of shift register, while LDPC codes employ a processor element. Accordingly, known codes require increased physical time of decoding, the need for large capacity of main memory grows, there are problems in circuit implementation. For example, a decoder of LDPC code is hardly possible to execute based on modern FPGA as they lack hardware units for computing of standard mathematical functions [29].

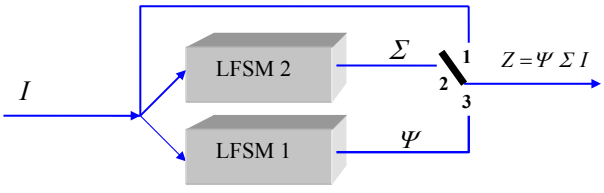

Fig. 4. Structure of the encoder for iterative decoding of cyclic codes

7. The process of decoding using the proposed method can be easily parallelized, in this case parallelization can be applied at multiple levels. First, all iterations can be performed simultaneously. Second, stages 3–6 within each iteration can be carried out in a pipelined manner when at each step of the pipeline one processes one code word. Parallel processing is not only possible, but also very desirable, as it eliminates the need to store a large volume of intermediate data in memory.

8. The proposed method of decoding employs a new principle for the combine of codes. In the known systems, one code (internal) is fully embedded into another code (external), in our case, both block codes are mutually independent with each performing its own task: first, a master code detects errors, then, with the "help" of a slave code, corrects them.

We shall consider substantiation of code selection for such combining.

Number of syndromes of corrected errors should be provided by the number r_1 of check bits in (n_1, k)-code $(r_1=n_1-k_1)$:

$$2^{r_1} \geq \sum_i^{\tau} C_{n_1}^i. \qquad (21)$$

Among traditional cyclic codes, inequality (21) holds as the equality for a Hamming code, single errors ($\tau=1$) and for all r_1.

At iterative decoding, all errors inside the check window $X_{err}^{(i)}$, must be corrected, that is $\tau=r_1$. The total number of error syndromes within a check window is equal to

$$N_w \geq \sum_i^{\tau} C_{r_1}^i.$$

A large number of error words inside the check window are the shifted copy of each other; to decode, it will suffice to use only one of them, for example, containing a "1" in a high position. In case LFSM 1 uses a primitive polynomial as polynomial (3), then such LFSM will generate a binary sequence of maximum period: $T=2^{r1}-1$. In this sequence, "1" is evenly spaced and found exactly 2^{r1-1} times [24].

At $r_1>32$, when number N_w of error syndromes is large enough, then it is possible to limit the multiplicity of corrected errors.

From the standpoint of polynomial representation of cyclic codes, the primitive generator polynomial has the Hamming code. When using a given code, at any code length, decoding will require only half of the error words (Table 1). Therefore, it is best to choose a cyclic Hamming code as the (n_1, k)-code.

Next, we shall consider substantiation for choosing (n_2, k)-code.

The purpose of this code is to check correctness of error correction in code word Z_{err} by the first code. The probability p_m that the (n_2, k)-code-assistant makes the wrong choice is equal to $p_m=2^{-(n2-k)}$. To reduce p_m, one should choose a cyclic code with primitive generator polynomial $g_2(x)$ as the (n_2, k)-code, that is, also a cyclic Hamming code. Good enough validation properties are demonstrated also by the Abrahamson code with generator polynomial $(1+x)g_2(x)$.

Thus, both codes can be cyclic Hamming codes, but necessarily with different generator polynomials ($g_1(x)\neq g_2(x)$). Powers of polynomials $g_1(x)$ and $g_2(x)$ should be sufficiently large: 16, 24, 32, 64. And these parameters are characteristic of CRC codes.

From the positions of CRC check, the CRC (n_2, k)-code will act as a cyclic redundancy check, while the CRC (n_1, k)-code – as a cyclic redundancy code, possibly shortened. The differences between these interpretations of CRC are described in detail in [30].

Such cyclic codes can also be called the iteratively decoded cyclic codes (IDCC).

10. Conclusions

1. We investigated the essence of iterative decoding of LDPC codes and turbo codes and demonstrated that the main criterion for the optimality of error correction codes has been over many years the degree of proximity of the decoding error probability to the theoretical limit (border) by Shannon. At an iterative (multistage) method of decoding, it is really possible to come maximally close to this limit, however, much will have to be sacrificed: increased length of codes, complexity of encoders and decoders, longer decoding time, emergence of other problems.

2. A theoretical substantiation is given for the iterative decoding of cyclic codes using the automaton representation of these codes. Rather than traditional soft decoding of iterative codes, we propose employing hard, that is exact, decisions, which would make it possible to accelerate the process of encoding and decoding at simultaneous minimization of resource costs.

3. We have developed a generalized iterative decoding algorithm for cyclic codes based on power permutation of bits in the code word, as well as the software model and circuits for codecs using linear feedback shift registers.

4. A new type of combined codes is proposed – iteratively decoded cyclic codes (IDCC). IDCC-codes could be recommended for use in systems where CRC control is applied: it will suffice to add a CRC check word to the employed CRC checksum. As a result, it is possible, by using a software means, not only to detect, but also correct errors of large multiplicity.

References

1. Shennon K. Raboty po teorii informatsii i kibernetike. Moscow: Izd-vo inostr. lit., 1963. 829 p.
2. Gallager R. G. Low Density Parity-Check Codes. Cambridge: M.I.T. Press, 1963. 90 p.
3. Berrou C., Glavieux A., Thitimajshima P. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1 // Proceedings of ICC '93 – IEEE International Conference on Communications. 1993. doi: 10.1109/icc.1993.397441
4. Vargauzin V. Vblizi granitsy Shennona // Telemul'timedia. 2005. Issue 6. P. 3–10.
5. Morelos-Saragosa R. Iskusstvo pomekhoustoychivogo kodirovaniya. Metody, algoritmy, primenenie. Moscow: Tekhnosfera, 2006. 320 p.
6. Sklyar B. Tsifrovaya svyaz'. Teoreticheskie osnovy i prakticheskoe primenenie. 2-e izd., ispr. Moscow: Izd. dom «Vil'yams», 2004. 1104 p.
7. Stollings V. Komp'yuternye sistemy peredachi dannyh. 6-e izd. Moscow: Izdatel'skiy dom «Vil'yams», 2002. 928 p.
8. Topalov V. V. The modification the bi-directional Soft Output Viterbi Algorithm for decoding of turbo product codes // Technology audit and production reserves. 2014. Vol. 6, Issue 3 (20). P. 62–65. doi: 10.15587/2312-8372.2014.34592
9. Novikov R. S., Astrahantsev A. A. Vybor parametrov LDPC kodov dlya kanalov s ABGSH // Systemy obrobky informatsiyi. 2014. Issue 1 (117). P. 195–199.
10. Error-Correction Coding and Decoding. Bounds, Codes, Decoders, Analysis and Applications / Tomlinson M., Tjhai C. J., Ambroze M. A., Ahmed M., Jibril M. // Springer International Publisher. 2017. 522 p. doi: 10.1007/978-3-319-51103-0
11. Mankar M. V., Asutkar G. M., Dakhole P. K. Quasi Cyclic Low Density Parity Check Decoder Using Min-sum Algorithm for IEEE 802.11n // IOSR Journal of VLSI and Signal Processing. 2016. Vol. 06, Issue 04. P. 01–07. doi: 10.9790/4200-0604020107
12. ETSI Standard EN 302 307-2 V1.1.1: Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2). European Telecommunications Standards Institute, Valbonne, 2005.
13. Revathy M., Saravanan R. A Low-Complexity Euclidean Orthogonal LDPC Architecture for Low Power Applications // The Scientific World Journal. 2015. Vol. 2015. P. 1–8. doi: 10.1155/2015/327357
14. Zolotarev V. V., Zubarev Yu. B., Ovechkin G. V. Mnogoporogovye dekodery i optimizatsionnaya teoriya kodirovaniya. Moscow: Goryachaya liniya – Telekom, 2012. 239 p.
15. LDPC Code Designs, Constructions, and Unification / Li J., Lin S., Abdel-Ghaffar K., Ryan W. E., Costello D. J. J. Cambridge University Press, 2016. doi: 10.1017/9781316780152
16. Kim S., Sobelman G. E. Scaling, Offset, and Balancing Techniques in FFT-Based BP Nonbinary LDPC Decoders // IEEE Transactions on Circuits and Systems II: Express Briefs. 2013. Vol. 60, Issue 5. P. 277–281. doi: 10.1109/tcsii.2013.2251959
17. Fedorenko S., Kolesnik V. Multi-step decoding of the iteration of Hamming codes // Proc. of the Seventh Joint Swedish-Russian International Workshop on Information Theory. Saint Petersburg, 1995. P. 80–83.
18. Cross-Layer Iterative Decoding of Irregular LDPC Codes using Cyclic Redundancy Check Codes / Yang Z., Li S., Feng H., Honold T., Yu G. // 2009 IEEE Wireless Communications and Networking Conference. 2009. doi: 10.1109/wcnc.2009.4917653

19. Zhang L., Huang Q., Lin S. Iterative decoding of a class of cyclic codes // 2010 Information Theory and Applications Workshop (ITA). 2010. doi: 10.1109/ita.2010.5454113
20. Forni D. Kaskadnye kody. Moscow: Mir, 1970. 207 p.
21. Klark ml. Dzh., Keyn Dzh. Kodirovanie s ispravleniem oshibok v sistemah tsifrovoy svyazi. Moscow: Radio i svyaz', 1987. 392 p.
22. Semerenko V. P. Estimation of the correcting capability of cyclic codes based on their automation models // Eastern-European Journal of Enterprise Technologies. 2015. Vol. 2, Issue 9 (74). P. 16–24. doi: 10.15587/1729-4061.2015.39947
23. Semerenko V. P. Teoriya tsyklichnykh kodiv na osnovi avtomatnykh modelei: monohrafiya. Vinnytsia: VNTU, 2015. 444 p.
24. Gill A. Lineynye posledovatel'nostnye mashiny. Moscow: Nauka, 1974. 288 p.
25. Semerenko V. P. Parallel Decoding of Bose-Chaudhuri-Hocquenghem Codes // Engineering Simulation. 1998. Vol. 16, Issue 1. P. 87–100.
26. Teoriya kodirovaniya / Kasami T., Tokura H., Ivadari E., Inagaki Ya. Moscow: Mir, 1978. 576 p.
27. Prange E. Cyclic error-correcting codes in two symbols. Air Force Cambridge Research Center, 1957. 26 p.
28. Kognovitskiy O. S. Dvoystvenniy bazis i ego primenenie v telekommunikatsiyah. Sankt-Peterburg: Link, 2009. 411 p.
29. Hlynov A. A. Issledovanie printsipov realizatsii LDPC kodeka na PLIS // Materialy mezhdunar. nauch.-tekhn. konf. "INTERMATIC – 2012". Moscow, 2012. P. 150–156.
30. Semerenko V. P. Theory and practice of CRC codes: new results based on automaton models // Eastern-European Journal of Enterprise Technologies. 2015. Vol. 4, Issue 9 (76). P. 38–48. doi: 10.15587/1729-4061.2015.47860

Запропоновано і детально описано оригінальну конструкцію динамічного гірокомпасу на оптичних гіроскопах та метод його використання. Розроблено алгоритм застосування такого гірокомпасування у випадку, коли вібропідставка лазерного гіроскопу відсутня. Доведено високу точність роботи такого гірокомпасу. В умовах його використання можна знизити рівень шуму лазерного гіроскопа. Завдяки обертанню значно компенсується вплив повільно мінливого дрейфу та магнітна складова дрейфу

Ключові слова: гірокомпасування, лазерний гіроскоп, акселерометр, дрейф, кут курсу, навігаційна система, керування рухом

Предложена и детально описана оригинальная конструкция динамического гирокомпаса на оптических гироскопах и метод его использования. Разработан алгоритм применения такого гирокомпасирования в случае, когда виброподставка лазерного гироскопа отсутствует. Доказана высокая точность работы такого гирокомпаса. При условиях его использования можно снизить уровень шума лазерного гироскопа. Благодаря вращению значительно компенсируется влияние переменного дрейфа и магнитная составляющая дрейфа

Ключевые слова: гирокомпасирование, лазерный гироскоп, акселерометр, дрейф, угол курса, навигационная система, управление движением

UDC 531.383
DOI: 10.15587/1729-4061.2018.119735

DEVELOPMENT OF METHOD AND ALGORITHM OF DYNAMIC GYROCOMPASSING FOR HIGH-SPEED SYSTEMS OF NAVIGATION AND CONTROL OF MOVEMENT

V. Uspenskyi
Doctor of Technical Sciences, Professor*
E-mail: uspensky61@gmail.com
I. Bagmut
PhD, Associate Professor*
E-mail: ivan.bagmut@gmail.com
M. Nekrasova
Associate Professor*
E-mail: slava2007@gmail.com
*Department of Computer Modeling of Processes and Systems
National Technical University «Kharkiv Polytechnic Institute»
Kyrpychova str., 2, Kharkiv, Ukraine, 61002

1. Introduction

One of the urgent tasks of creating and improving motion control systems for modern aerospace objects is improving the accuracy of their information subsystems in general and navigation equipment, in particular. The main direction of solving this problem is the use of redundant information coming from inertial sensors and a receiver of satellite navigation signals. This explains why the issues of rational combination of information in such systems and the