Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 12, n. 1, 2026.

Trabalho apresentado no XLIV CNMAC, Fundação Getulio Vargas - Rio de Janeiro - RJ, 2025.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Encoding and Decoding Process of Data with SageMath

John H. Castillo[1], Juan F. Mafla[2], Catalina M. Rúa-Alvarez[3]
Departamento de Matemáticas y Estadística, Universidad de Nariño, Pasto, Colombia

Abstract. In this work, we address the transmission of digital data in a noisy environment and highlight the importance of coding theory to ensure the integrity of the information. The study focuses on the decoding process using syndromes for the family of binary linear codes in general. The main goal is to demonstrate the simulation of data encoding, transmission, and decoding using SageMath. In this research, the techniques used in [7] are extended to perform simulations involving texts written with ASCII characters and images in black and white; we point out that these ideas can be used to process images in other formats. In this talk, we will present the mathematical background and give several simulations of the encoding and decoding process.

Keywords. Binary Linear Code, SageMath, Text, Image

1 Introduction

Coding theory is a branch of applied mathematics that has evolved to be important in several areas of communications. Its principal objective is to guarantee the integrity of the data transmitted through a communication channel that can be affected by noise. To achieve this, coding and decoding techniques are used that allow errors caused by noise to be detected and, in some cases, corrected. This discipline utilizes various mathematical concepts from different areas of mathematics, including: linear algebra, probability, combinatorics, ring theory and group theory, among others. These elements are essential for the design and analysis of efficient codes that ensure reliable communication, see [2, 4, 5]. Coding theory focuses on communication through noisy channels and the ability to recover information from messages affected by errors. In this context, corrupted messages are those that have undergone alterations during transmission. A fundamental approach in this theory is to understand how to design systems capable of detecting and correcting these errors to ensure reliable and accurate communication.

The process of transmitting information through noisy channels can be visualized as a journey from an origin point (transmitter) to a destination (receiver). During this journey, the information encounters various obstacles, such as electromagnetic interference or atmospheric noise, which may introduce errors into the original message. Coding theory is concerned with developing methods and techniques to mitigate the effects of these obstacles and ensure that the information reaches its final destination in an intelligible manner.

Firstly, we recall some definitions and notations. Let $\mathbb{F}_2 = \{0, 1\}$ be the field with 2 elements, as usual the operations of addition and product are made modulo 2. For $n \in \mathbb{Z}^+$, let $\mathbb{F}_2^n$ denotes the vector space of all n-tuples over $\mathbb{F}_2$. Any non-empty set $\mathcal{C}$ of $\mathbb{F}_2^n$ is called a *binary block code*. Each element $(a_1, a_2, \ldots, a_n)$ (or simply $a_1a_2 \ldots a_n$) in $\mathcal{C}$ is a *codeword*, and if $\mathcal{C}$ contains M elements, then it is said that $\mathcal{C}$ has *length* n and *size* M, or simply that $\mathcal{C}$ is an (n, M)*-code*.

[1]jhcastillo@udenar.edu.co
[2]juanmafla@udenar.edu.co
[3]catalina.rua@udenar.edu.co

A *binary* $[n, k]$*-linear code* $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_2^n$. The *Hamming distance*, $d(\boldsymbol{x}, \boldsymbol{y})$, between two codewords $\boldsymbol{x} = (x_1, \ldots, x_n), \boldsymbol{y} = (y_1, \ldots, y_n) \in \mathcal{C} \subseteq \mathbb{F}_2^n$ is the number of entries where they differ, or equivalently, $d(\boldsymbol{x}, \boldsymbol{y}) = |\{i : x_i \neq y_i,\ 1 \leq i \leq n\}|$. For $\boldsymbol{x} \in \mathbb{F}_2^n$, the *Hamming weight* of $\boldsymbol{x}$ is $\mathrm{wt}(\boldsymbol{x}) = d(\boldsymbol{x}, \boldsymbol{0})$, i.e., $\mathrm{wt}(\boldsymbol{x})$ is the number of non-zero coordinates in $\boldsymbol{x}$. The *minimum distance* $d(\mathcal{C}) = d$ of a linear code $\mathcal{C}$ is defined as the minimum weight among all non-zero codewords, thus we called it a *binary* $[n, k, d]$*-linear code*.

Given a code $\mathcal{C}$ the *minimum distance decoding process* is as described below. Suppose that codewords from a code $\mathcal{C}$ are sent over a communication channel. If a word $\boldsymbol{x}$ is received, it will be decode $\boldsymbol{x}$ to $\boldsymbol{c_x} \in \mathcal{C}$ if $d(\boldsymbol{x}, \boldsymbol{c_x}) = \min_{\boldsymbol{c} \in \mathcal{C}} d(\boldsymbol{x}, \boldsymbol{c})$. If two or more codewords of $\mathcal{C}$ satisfy the last expression, the *complete decoding rule* arbitrarily selects one of them to be the most likely word sent, while the *incomplete decoding rule* requests for a retransmission.

Let t be a positive integer. We recall, that a code $\mathcal{C}$ is a *t-error-detecting code* if, whenever a codeword incurs at least one but at most t errors, the resulting word is not a codeword. A code $\mathcal{C}$ is an *exactly t-error-detecting code* if it is t-error-detecting but not $(t+1)$-error-detecting. Also, a code $\mathcal{C}$ is a *t-error-correcting code* if minimum distance decoding is able to correct t or fewer errors, assuming that the incomplete decoding rule is used. A code $\mathcal{C}$ is an *exactly t-error-correcting code* if it is t-error-correcting but not $(t+1)$-error-correcting. The next result gives the capacity of detection and correction of a code $\mathcal{C}$ in terms of its minimum distance $d(\mathcal{C})$, see theorems 2.5.6 and 2.5.10 in [5].

Theorem 1.1. *Let $\mathcal{C}$ be a code. Then $d(\mathcal{C}) = d$ if and only if $\mathcal{C}$ is an exactly $(d-1)$-error-detecting code and $\mathcal{C}$ is an exactly $\left\lfloor \frac{d-1}{2} \right\rfloor$-error-correcting code.*

A *generator matrix* for an $[n, k]$-linear code $\mathcal{C}$ is any $k \times n$ matrix G whose rows form a basis for $\mathcal{C}$. So the code $\mathcal{C}$ can be seen as

$$\mathcal{C} = \{\boldsymbol{x}G : \boldsymbol{x} \in \mathbb{F}_2^k\}. \tag{1}$$

Thus for each codeword $\boldsymbol{c} \in C$, there exists a $\boldsymbol{x} \in \mathbb{F}_2^k$, such that $\boldsymbol{c} = \boldsymbol{x}G = x_1\boldsymbol{v}_1 + x_2\boldsymbol{v}_2 + \cdots + x_k\boldsymbol{v}_k$, where $B = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ is a base of the vector space $\mathcal{C}$. Note that $\boldsymbol{v}_i$ are the rows of G. Hence, we can perform the encoding process as follows: given a piece of information represented by a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_k)$ it is encoded as $\mathrm{Enc}(\boldsymbol{x}) = \boldsymbol{x}G$.

Also, as an $[n, k]$-binary linear code $\mathcal{C}$ is a subspace of a vector space, it is the kernel of some linear transformation. In particular, there is an $(n-k) \times n$ matrix H, called a *parity-check matrix* for $\mathcal{C}$, such that

$$\mathcal{C} = \{\boldsymbol{v} \in \mathbb{F}_2^n : \boldsymbol{v}H^T = \boldsymbol{0}\}. \tag{2}$$

For the decoding process, the parity-check matrix plays a crucial role. Take $\mathcal{C}$ an $[n, k]$-binary linear code with parity-check matrix H. For $\boldsymbol{u} \in \mathbb{F}_2^n$ the *syndrome* of $\boldsymbol{u} \in \mathbb{F}_2^n$, is defined by $\mathrm{syn}(\boldsymbol{u}) = \boldsymbol{u}H^T$ and the set $\boldsymbol{u} + \mathcal{C} = \{\boldsymbol{u} + \boldsymbol{c} : \boldsymbol{c} \in \mathcal{C}\}$ is a coset of $\mathcal{C}$. A *coset leader* of $\boldsymbol{u} + \mathcal{C}$ is any vector in it with the minimum Hamming weight and the weight of the coset is the Hamming weight of any of its coset leaders. By (2), the code $\mathcal{C}$ consists of all vectors whose syndrome equals $\boldsymbol{0}$. As H has rank $n-k$, every vector in $\mathbb{F}_2^{n-k}$ is a syndrome. Also, by [4, Theorem 1.11.5], two vectors belong to the same coset if and only if they have the same syndrome.

Suppose a message is represented by a codeword and it is sent over a communication channel. Assume, that a vector $\boldsymbol{y}$ is received. Since in nearest neighbor decoding we seek a vector $\boldsymbol{e}$ of smallest weight such that $\boldsymbol{y} - \boldsymbol{e} \in \mathcal{C}$, nearest neighbor decoding is equivalent to finding a coset leader $\boldsymbol{e}$ of $\boldsymbol{y} + \mathcal{C}$. The *Syndrome Decoding Algorithm* can be devised as follows. We begin with a fixed parity-check matrix H of $\mathcal{C}$.

1. For each syndrome $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$, choose a coset leader $\boldsymbol{e_s}$ such that $\mathrm{syn}(\boldsymbol{e_s}) = \boldsymbol{s}$. Create a table of syndromes, pairing each syndrome $\boldsymbol{s}$ with its respective coset leader $\boldsymbol{e_s}$.

DOI: 10.5540/03.2026.012.01.0279

2. After receiving a vector $\boldsymbol{y}$, compute its syndrome $\text{syn}(\boldsymbol{y}) = \boldsymbol{y}H^T = \boldsymbol{s}$.

3. $\boldsymbol{y}$ is then decoded as the codeword $\boldsymbol{y} - \boldsymbol{e_s}$, where $\boldsymbol{e_s}$ is a coset leader of $\boldsymbol{y} + \mathcal{C}$.

Therefore, the decoding process can be seen as a function from $\mathbb{F}_2^n$ in $\mathbb{F}_2^k$ given by $\text{Dec}(\boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{e_s}$. Unfortunately, this function depends on the coset leader $\boldsymbol{e_s}$ taken during the process, actually a coset can have several coset leaders. In fact, from Theorem 1.1, it can be proved that every coset of weight at most $\lfloor (d-1)/2 \rfloor$ has a unique coset leader.

Example 1.1. Consider the $[5, 2, 3]$-binary linear code $\mathcal{C} = \{00000, 11100, 01111, 10011\}$. A parity-check matrix of $\mathcal{C}$ is

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

and its table of syndromes is given in Table 1.

Table 1: Syndromes of $\mathcal{L}$.

Syndrome	Coset leader	Syndrome	Coset leader
000	00000	010	00010
011	10000	001	00001
111	01000	110	00110
100	00100	101	01010

Suppose that the codeword $\boldsymbol{c} = 01111$ was sent and that the received vector was $\boldsymbol{x} = 01011$. Note that $\boldsymbol{x} \notin \mathcal{C}$ and $\text{syn}(\boldsymbol{x}) = 100$. Since, $\boldsymbol{x}$ has the same syndrome of 00100, it is decoded as $\boldsymbol{c} = 01011 - 00100 = 01111$, the real sent word. Note that in this instance just one error was introduced during the transmission. In the other hand, assume that $\boldsymbol{v} = 10110$ is received, again note that this is not a codeword. Then $\text{syn}(\boldsymbol{v}) = \boldsymbol{v}H^T = (101)$. From the above, $\boldsymbol{v}$ has the same syndrome of 01010. But, the coset $\boldsymbol{v} + \mathcal{C}$ has other coset leader 00101. Therefore, $\boldsymbol{v}$ will be decoded either as 11100 or 10011. Of course, just one of them must be a corrected option. This happens, because $\mathcal{C}$ can correct just 1 error.

Consequently, the functions Enc and Dec can be used to simulate the encoding and decoding processes for a given binary linear code. The only requirement is to transform the information into binary vectors, which will be transmitted through a noisy channel. In the next section, we will demonstrate how these processes can be carried out using `SageMath`.

2 Transmission of Data with Coding Theory

We thoroughly explore the process of data transmission through a noisy channel using `SageMath`. With the aid of concrete examples, we provide a practical visualization of how text data and image are transmitted and received across a channel subject to interference and errors. The development of this section is based on the ideas described in [7], which presents coding theory and provides insight into the simulation processes of encoding and decoding of text strings written with up to 64 characters. The main novelty of this research is that we extend the procedures done in [7] for text written with ASCII symbols (up to 256 characters) and images in black and white. In our research, we focuses on utilizing these tools to perform simulations related to coding theory. With the assistance of `SageMath`, it is possible to explore and experiment with different parameters, allowing for a better understanding of the functionality and effectiveness of error-correcting codes in practical scenarios.

DOI: 10.5540/03.2026.012.01.0279

SageMath offers a variety of functions for coding theory, data transmission, and decoding, facilitating the implementation and simulation of various applications within mathematics. Some of these functions include tools for working with binary linear codes. To access these functions, the command `LinearCode()` is used, which allows working with various linear codes, including their generation. Additionally, it enables the determination of the minimum distance of a code, which helps to obtain the error detection and correction capabilities of a given code. In SageMath, the encoding and decoding functions, Enc and Dec, can be implemented using the `encode()` and `decode to message()` commands, respectively. The channel is generated through the `channels.StaticErrorRateChannel` class, which enables the simulation of a channel introducing either static errors or randomly selected errors within a specified range of number errors. Moreover, codewords can be transmitted through the constructed channel using the `transmit()` method.

2.1 Text

To work with text data, it is necessary to define and represent the data as binary vectors. To do this, the ASCII code (an acronym for American Standard Code for Information Interchange) will be used. This code was created in 1963 by the American National Standards Institute (ANSI) after reorganizing and expanding the set of symbols and characters previously used in telegraphy by the Bell telephone company. Initially, it included only uppercase letters and numbers, but in 1967, lowercase letters and some control characters were added, forming what is now known as US-ASCII, which consists of characters ranging from 0 to 127. This set of 128 characters was published as a standard in 1967 and contained everything necessary for writing in English. In 1981, IBM introduced an 8-bit extension of the ASCII code, known as "code page 437". In this version, some obsolete control characters were replaced with graphic characters. Additionally, 128 new characters were added, including symbols, additional graphic signs, and Latin letters necessary for writing in other languages, such as French, Spanish and Portuguese. As a result, the ASCII code was expanded to a total of 256 characters, see [1].

Each element in the list will be represented by an element of $\mathbb{F}_2^8$, corresponding to its index digit, which ranges from 0 to 255. For instance, the letter "b" has an index of 98, and therefore, its binary representation is "01100010". We emphasize that with this approach, we move from encoding text string written with up to 64 characters made in [7] to a total of 256 characters.

Example 2.1. We will simulate the process of coding and decoding for a text with a $[15, 8, 3]$-binary linear code, so by Theorem 1.1 this code can exactly correct 1-error. Thus, if we transmit the message "El español y el portugués son idiomas que provienen del latín", we obtain the results given in Figure 1. It can be observed that the original message remained unchanged when a single error was introduced, which was the expected outcome. However, when two or more errors are introduced, the code is unable to correct them, resulting in alterations to the message, making it impossible to identify the original text. Nevertheless, it is noteworthy that the output message with two errors retains some characters in the same positions as the original text.

```
Message with 0 errors: El español y el portugués son idiomas que provienen del latín
Message with 1 errors: El español y el portugués son idiomas que provienen del latín
Message with 2 errors: ee$epðaÃol y lápor u'õéc¸sojjIá`/as CóD °^l ilOmB Heô La|¸ö
Message with 3 errors: Åd¥0ñáqeft- Åê¬p ùV%á Åð0p|Bc[âiÿ-aS¦}T% º÷îyækqj#Ea~cì(T¯O
Message with 4 errors: H²v~ briÌ ¸ãi |lfiõ£ÕúpáYgbF«h G<Ñ7 QßvaT&ê÷û ë ¸t\Å.ò© äah
Message with 5 errors: U{1ãègYtNÞe -WÕ«D·ÒH5UÅãwgy~¯ øÄÏ{z Ã$P|W²0uCz {mfÿ7` |p q&/!
```

Figure 1: Simulation of coding and decoding process of a text with SageMath. Source: Authors.

DOI: 10.5540/03.2026.012.01.0279

Now, for each number of errors introduced in a codeword, we calculate the number of different characters between the original message and the received one, and we use this value to calculate the error percentage. To carry out this calculation, a seed is set to control the initialization of random values, repeating the process 10 times to obtain the average error rate. In Table 2, we show the output obtained from this simulation. It can be observed that for 1-error in the 10 simulations the message received is always the transmitted one, note that in this case the average error rate is 0%, in the other cases this average is increasing as expected.

Table 2: Average error percentage.

	Number of errors				
	1	2	3	4	5
Average error rate	0%	59.18%	87.87%	98.36%	100%

2.2 Images

Image analysis and processing have evolved over time, and in recent decades, they have become part of everyday practice thanks to technological advancements. Although the mathematical foundations for these tasks have existed for a long time, only with the development of sufficiently advanced software has it been possible to perform the intensive computations required for noise reduction, edge detection, segmentation, and other image analysis, see [3].

A digital image can be understood as a two-dimensional function, denoted as $f(x, y)$, where x and y represent the spatial coordinates of the image, and f indicates the brightness intensity at each coordinate, see [3, p. 65]. The smallest element of an image is the pixel, which stores a value that reflects the intensity of light or color at a specific point in the image. Thus, each pixel is associated to a position (x, y), and its value $f(x, y)$ determines the visual appearance of the image at that point. Moreover, a digital image can be represented as an $M \times N$ matrix of pixels, where M and N are the quantities of vertical and horizontal pixels in the image, respectively.

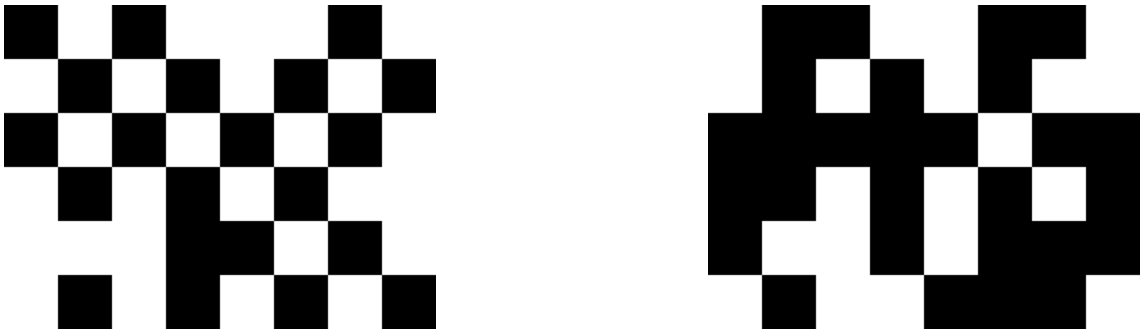
The black and white image format uses only two tones: black and white. In this format, each pixel is represented by a binary value, where 0 denotes black and 1 corresponds to white, see for instance Figure 2. This approach further simplifies the image structure, as it requires only 1 bit per pixel to store the information. Black and white images are ideal for applications that prioritize simplicity and the clarity of shapes or contours.

(a) Image. (b) Binary representation.

Figure 2: Black and white image and its binary representation. Source: [6].

Example 2.2. Following the ideas given previously, we reproduce the encoding and decoding process for Figure 2a with a binary code $\mathcal{C}_1$ with parameters $[12, 6, 3]$. Since the code has minimum

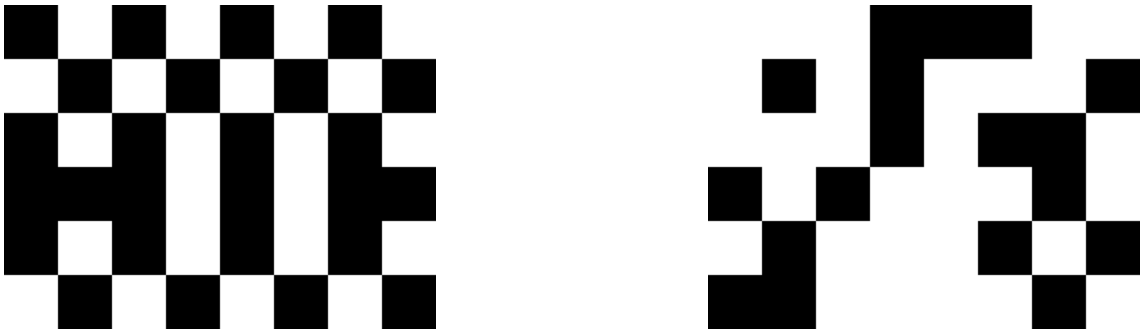
distance equals 3, it is only capable of correcting a single error. When two or more errors are introduced, the images exhibit increasingly noticeable changes in their content, see Figure 3.



(a) Image with 2 errors. (b) Image with 3 errors.

Figure 3: Simulations with a $[12, 6, 3]$-linear code. Source: Authors.

Also note that we can use a code with a greater minimum distance to be able to correct more errors. For instance, if we use a $[15, 4, 8]$-linear code $\mathcal{C}_2$, we get the same image when the number of errors introduced in the transmitted images is less or equal than 3. In this instance, the result image will be different when the number of errors is greater than 3, see Figure 4.



(a) Image with 4 errors. (b) Image with 5 errors.

Figure 4: Simulations with a $[15, 4, 8]$-linear code. Source: Authors.

This procedure can be extended to work with images in gray scale where the intensity of tones varies from 0 (black) to 255 (white). In this case, since each tone of gray is represented by an integer it is converted to its corresponding binary representation with 8 digits. For example, neutral gray, which has a value of 160, is represented in binary form as 10100000. With the binary representation of each pixel, the idea used for transmission of black and white images can be applied. Similarly, this approach can be extended to the transmission of color images in RGB format, where each pixel consists of three intensity values, between 0 and 255, for the red, green, and blue components, respectively.

To avoid a visual comparison of similarity between images, the average error rate calculated in the text transmission, see Table 2, can also be extended to this case.

DOI: 10.5540/03.2026.012.01.0279

3 Discussion

This talk highlights the use of the family of binary linear codes to simulate encoding and decoding processes with the assistance of the computational algebra system `SageMath`. The use of `SageMath` as a simulation tool enabled a practical exploration of the underlying theoretical principles, facilitating the understanding and application of key concepts such as error correction capability, error detection capability, and syndrome decoding, among others. Through the conducted simulations, a clear representation of the encoding, transmission, and decoding processes was achieved, reinforcing both the theoretical value of error-correcting codes and their applicability in real-world contexts. One of the main contributions of this work is the adaptation of the methods used in [7] to encode and decode texts containing any of the 256 ASCII characters, rather than the 64 characters proposed in [7]. We point out that, in the talk we will show that this can be done with grayscale and color images. Nevertheless, the results obtained also indicate the necessity of employing codes with improved error detection and correction capabilities in future research, particularly those with greater minimum distance. Moreover, the adaptation of these techniques to various data types, such as text and images, demonstrates that the principles of coding theory are broadly applicable, provided that appropriate adjustments are made for the specific characteristics of the data formats used. These findings underscore the importance of further exploration and optimization of these methodologies for their implementation in broader and potentially more complex scenarios.

Acknowledgements

This work was partially supported by Vicerrectoría de Investigaciones e Interacción Social at Universidad de Nariño.

References

[1] **ASCII table**. Available in: https://theasciicode.com.ar/. 2024.

[2] J. H. Castillo, J. J. López, and H. M. Ruiz. **Introducción a la teoría de códigos correctores de errores**. Pasto, COL: Editorial Universidad de Nariño, 2025. ISBN: 978-628-7771-19-2.

[3] R. C. Gonzalez and R. E. Woods. **Digital Image Processing**. 4th. ed. Pearson Education, 2018. ISBN: 978-1-292-22304-9.

[4] W. C. Huffman and V. Pless. **Fundamentals of error-correcting codes**. Cambridge, UK: Cambridge University Press, 2010. ISBN: 978-0-521-78280-7.

[5] S. Ling and C. Xing. **Coding Theory: A First Course**. New York: Cambridge University Press, 2004. ISBN: 978-0-521-82191-9.

[6] PNGWing. **Free PNG**. `https://www.pngwing.com/es/free-png-bbhbi`. Last accessed: November 26, 2024. Publication date not specified.

[7] T. D. Timur, D. Adzkiya, and Soleha. "Simulations of linear and Hamming codes using SageMath". In: **Journal of Physics: Conference Series, 974(1)** (2018). DOI: 10.1088/1742-6596/974/1/012064.

