Error-Correcting Codes

Math240-5C:
Ibrahim Topal
Antriana Savoulli
Hubert Grzelczak
Artin Lee
Amara Pandya

March,2024

Contents

1 Introduction **2**
1.1 What are codes and errors? 2
1.2 Definitions used in the text 3

2 Repetition Code **4**
2.1 How it works? 4
2.2 Application of Repetition Code 4
2.3 Is it a good error-correction and error-detection method? 7

3 Hamming Code **7**
3.1 Intuitive introduction 7
3.2 Generator matrix 10
3.3 Parity-Check matrix 10
3.4 Application of Hamming Code 11

4 More Codes **12**
4.1 Modern Codes 12
4.2 Application 13

5 Challenges **14**

6 Further Reading **15**

Abstract

Error correcting codes play a pivotal role in ensuring the integrity of transmitted data across various communication channels. This paper explores two fundamental error correcting codes: the repetition code and the Hamming code, highlighting their applications and significance in mitigating transmission errors. The repetition code, a basic but effective error correction technique, involves the duplication of data to detect and correct errors. Its simplicity makes it suitable for applications where reliability is paramount, such as space image transmission, where even minor errors can compromise critical information. By replicating data multiple times, the repetition code enhances resilience against noise and interference, thereby facilitating accurate image reconstruction in space missions.

In contrast, the Hamming code represents a more sophisticated error correction method that provides both error detection and correction capabilities. Named after its inventor Richard Hamming, this code employs parity bits to identify and correct single-bit errors within data words. While initially developed for computer memory systems, Hamming codes find diverse applications in various digital communication systems. By employing a systematic approach to error correction, Hamming codes offer robustness against transmission errors, ensuring reliable data exchange.

Through the examination of these error correcting codes and their applications, this paper underscores the importance of error detection and correction mechanisms in modern communication systems. By implementing appropriate coding schemes tailored to specific requirements, such as the repetition code for space image transmission and the Hamming code for general-purpose error correction, communication systems can effectively mitigate the impact of transmission errors and uphold data integrity.

1 Introduction

1.1 What are codes and errors?

In the expansive realm of technology, codes serve as the fundamental language facilitating communication between humans and machines. Essentially, codes are structured sets of rules or symbols designed to convey information, finding utility across diverse domains from linguistics and mathematics to computing.

The origins of coding can be traced back to the 19th century, notably with Ada Lovelace's collaboration with Charles Babbage on the Analytical Engine. Their visionary efforts laid the groundwork for modern programming, marking the inception of a symbiotic relationship between humans and machines.

The evolution of coding witnessed significant milestones, particularly with the emergence of electronic computers in the mid-20th century, epitomized by the completion of the ENIAC (Electronic Numerical Integrator And Computer) in 1945 [5], which is often regarded as one of the earliest general-purpose electronic digital computers. These machines, capable of executing various tasks,

spurred the development of high-level programming languages like Fortran, fostering the widespread adoption of coding across diverse applications.

Central to our exploration is the realm of error correcting codes, a vital aspect of coding theory. As modern communication increasingly relies on information transmission, ensuring reliable and accurate data exchange becomes imperative. Error correcting codes emerged as a response to this necessity, enabling the detection and correction of errors that may occur during data transmission.

In the domain of machines, precision and reliability are paramount. Machines fundamentally understand and process information as sequences of 1s and 0s. However, when engaged in communication, they encounter challenges, particularly with data transmission. External factors such as electromagnetic interference and signal attenuation can disrupt the integrity of transmitted data, resulting in errors.

These errors manifest as deviations from the intended message, particularly in scenarios where data travels extensive distances or encounters interference. To address this vulnerability, error detection mechanisms become crucial safeguards, identifying anomalies within transmitted data and enabling error localization and correction.

Despite advancements, no error detection or correction method is infallible. However, the hallmark of a good error correction code lies in its ability to eliminate the majority of errors, making communication highly reliable. Evaluating effectiveness involves considering factors such as error detection and correction ability, adaptability to diverse environments, and mitigation of burst errors.

While striving for error-free communication remains a goal, ongoing advancements aim to enhance error detection and correction capabilities, inching closer to a future where machine communication is virtually error-free. Through this journey, we explore various error correcting code methods and applications.

1.2 Definitions used in the text

Definition 1.1: Hamming Distance [2]
The Hamming distance refers to the number of different bits between codes of the same length. This is most often used as a measure of the number of errors that happen during data transfers such that a message that has a Hamming distance of 2 will have 2 bits changes due to errors. A rigorous definition is as follows:
Let v, w be sequences of length n. The Hamming distance $d(v, w)$ from v to w is the number of coordinates where v and w differ:
$d(v, w)=|\{i : 1 \leq i \leq n, v_i \neq w_i\}|$

Definition 1.2: e-error correction [1]
Let e be a positive integer. A code C of length n is said to be e-error correcting if the following holds: For any message w of length n there is at most one $c \in C$ such that $d(v, w) \leq e$

Definition 1.3 :Hamming Bound/Perfect code [1]
A perfect code describes a code that for every possible message received will be able to detect that an error has occurred and will assign that incorrect message to a single correct message. This is a very useful property as it means that the under the given parameters a given perfect code will always correct a message with no ambiguity about the accuracy of the correction. A code is considered a perfect code when it follows the Hamming Bound which is defied as such:
Let C be a code of length n over an alphabet of q symbols (for a binary code this value will be 2). Where C has a minimum Hamming distance of d. The Hamming bound is defined as:

$$|C| \leq \frac{q^n}{\sum_{i=0}^{e} \binom{n}{i}(q-1)^i}$$

2 Repetition Code

2.1 How it works?

Definition 2.1: Repetition code
A repetition code is a simple form of error-correcting code that works by repeating each bit of the original message a prearranged number of times (typically three times) and then the receiver uses a simple voting scheme to decode the message, determining the correct value of each bit.
Note: For a code of length and minimum Hamming distance n a repetition code can correct up to t errors such that: $t = \frac{n-1}{2}$. If n is odd then it is a perfect code.[1]

Example 2.1: Simple binary
If we want to transmit the bit sequence 101 using a repetition code, then the encoding maps each bit to either all zeros or ones so we get 111000111, where each bit is repeated three times.

Let's say that two errors corrupt the transmitted bits and the received sequence is 101100111.

By using a majority vote (selecting the value that appears most frequently), the receiver can correct the error and determine that the original sequence was 101.

2.2 Application of Repetition Code

Error Correcting Codes play a crucial role in the reliable transmission and storage of data, including images, across noisy channels or media. When an image is transmitted over a network (like the internet) or stored on a physical medium (like a hard disk), it is susceptible to various types of errors. These errors can be caused by interference, signal degradation, or hardware malfunctions, leading to corrupted data that can significantly affect the quality and integrity of the received image. ECCs are designed to address these challenges

by ensuring that even if some of the data is corrupted, the original image can still be accurately reconstructed.

If we zoom in enough to any image, we see a set of pixels (coloured blocks), and the combination of all these pixels is what displays the final image. Every pixel in an image, represented by a single colour, is identified with a number from 0 to 255; 0 being white, and 255 being black.

To identify any errors in the image, these numbers must be converted into binary (decimal to binary conversion). To convert a decimal number to binary, follow these steps:

1. Divide the number by 2.
2. Write down the remainder.
3. Divide the quotient by 2.
4. Repeat steps 2 and 3 until the quotient is 0.
5. The binary number is the remainders read in reverse order.

Example: Convert 13 to Binary

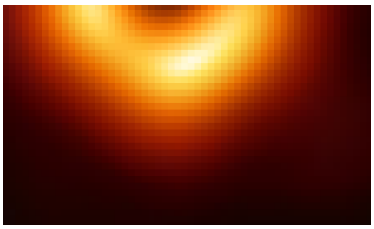
Quotient	**Divisor**	**Remainder**
13	2	1
6	2	0
3	2	1
1	2	1
0		

The binary representation of the decimal number 13 is read from the bottom to the top of the *Remainder* column: 1101.

Repetition codes can be used in the following way:

We start with an image that we want to share, for example, this picture of the black hole.

If we zoom in enough, we can see each of the pixels that formulate this image.



Repetition codes function by transmitting the image data several times. For each pixel, the binary codes received from these multiple transmissions are compared against each other. This comparison involves analysing the binary representation of each pixel's color value across the different transmissions. The most frequently occurring binary code for each pixel is then selected as the correct value, effectively correcting any errors introduced during transmission. For this example, we will focus on a random 12 block pixel (for simplistic reasons, repetition codes check every single pixel's code).

206	178	78	50
255	200	108	64
198	185	120	70

11001000

206	178	78	50
255	200	108	64
198	185	120	70

11001000

206	178	78	50
255	201	108	64
198	185	120	70

11001001

11001000
11001000
11001001
11001000

(

In the first and second image, the 6th pixel has been represented as 200. However, in the third image it has been incorrectly transmitted and is now 201. The difference between the original pixel value of 200 (binary representation 11001000) and the incorrectly transmitted value of 201 (binary representation 11001001) is a single bit. This is a common type of error that can occur during data transmission due signal degradation, or other interference. Repetition

codes stack the codes on top of one another, and compare; taking the most common number in column. We see that in the column to the most right has the bit 0 as it is the most common, and therefore the code is corrected, nullifying the incorrectly transmitted 1.

2.3 Is it a good error-correction and error-detection method?

Repetition codes are simple and easy to implement, but they are not very efficient because they require transmitting each bit multiple times. Additionally, they can only correct errors if the number of errors is within a certain threshold, typically half the number of repetitions. For example, in a repetition code where each bit is repeated three times, it can correct one error per group of three bits. If there are too many errors, the receiver may not be able to determine the correct value of the bits.

Therefore, whether repetition codes are a good error-correction and error-detection method depends on the specific context, and that is why they are often used in combination with other coding techniques.

3 Hamming Code

3.1 Intuitive introduction

Invented by Richard Wesley Hamming in 1950[2] ; Hamming codes were invented to correct errors introduced by punched card readers which were left alone overnight. Being one of the oldest error correcting methods, it's still in use as it is one of the first examples of a Perfect code.

Before attempting to understand the comprehensive method of creating and analysing Hamming codes it is useful to first have an understanding behind the intuition that was used to create them .

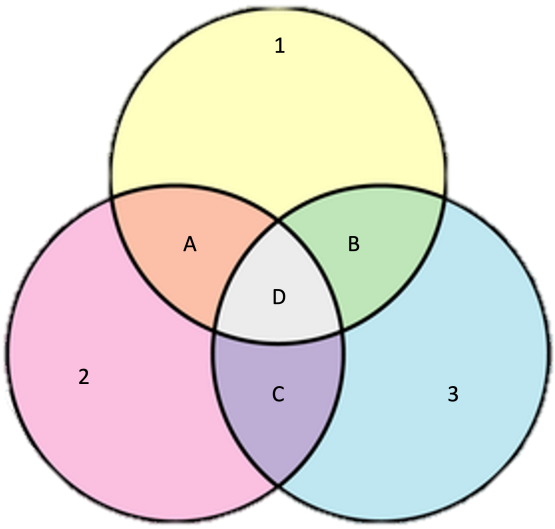


Figure 3.1

As seen in (Figure 3.1) we begin with a code of length 4 consisting of ($ABCD$). This code then has an additional 3 error correcting bits (123) added called parities. This then creates a message of length 7 resulting in a code ($ABCD123$). As code is written in base 2, let us use an example of a possible code. Let's take an original message of (1010). Now we determine the parities based on the original message with each parity corresponding to 3 of the 4 bits of the message. The parities count if the number of 1's in its corresponding region is odd or even. Thus taking the first parity as corresponding to the bits A B D in our message we see (100) thus giving us a parity 1. We repeat this for the remaining parities resulting in (1010) becoming (1010101). This intuitive understanding can then be turned into a comprehensive method.

Definition 3.1: Parity
A parity is a single bit of data that has been added to a code to describe whether the number of 1's in its corresponding bits is odd or even. Convention being a 1 indicating an odd number of 1's and a 0 indicating an even number.

We can define the specific relations of each parity as a matrix A.
For (Figure 3.1) we can define parity 1 depending on ABD as:
$1 : A \bigoplus B \bigoplus D$
Thus we can define all the parities as:
$1 : A \bigoplus B \bigoplus D$
$2 : A \bigoplus C \bigoplus D$
$3 : B \bigoplus C \bigoplus D$
This can be shown in the matrix A such that :

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Here we have each row representing a parity and each column representing a bit of the original code. Where a 1 is present at an intersection is a point where a parity depends on that bit of the original code.

Different types of Hamming code

A Hamming code can be made with varying length depending on the length of the original message and the number of parities added. It is convention that a Hamming(n,k) corresponds to a code with original message of length k and a final message of length n.

(Figure 3.1) is an example of a Hamming(7,4) code corresponding to an original message of length 4 having 3 parities added to create an output message of length 7. This type of Hamming code is able to correct 1 bit errors if it is known that the Hamming distance of the message is 1 or less. If the Hamming distance is assumed 2 or less then it is only able to identify 1 or 2 bit errors without correcting them. This happens as although a Hamming(7,4) code is able to detect 1 and 2 bit errors but it is unable to differentiate between them. Therefore if we attempt to error-correct a 2 bit error we often cause further problems as we change a bit that was not an error.

Another common type is the Hamming(8,4) code which consists of a message of length 4 and 4 added parities resulting in a message of length 8. As compared to the Hamming(7,4) for this type we add an additional parity that counts the other 7 bits. This allows for the code to be able to detect 1 and 2 bit errors when the Hamming distance is 2 but it also allows the correction of 1 bit errors when the Hamming distance is 2. Compered to the Hamming(7,4) the Hamming(8,4) sacrifices message length in exchange to be able to work in situation where more data loss is expected.

It should be noted that not all Hamming codes of the same length are equivalent. The notation Hamming(n,k) only describes the message length and number of parities whilst making no distinction of the areas the individual parities encompass. Taking (Figure 3.1) as an example, the image shows a Hamming(7,4) where the final message can look like (ABCD123) whilst at the same time it could also look like (ABCD321) with both fulfilling the Hamming(7,4) definition but both being generated in slightly different form. It is convention for the first bits of the final message to be at the start followed by the parities as such although the code (A123BCD) does follow the definition of a Hamming(7,4) it will not be seen often. This is why it is important when using Hamming codes that both the sending and receiving components are both given the same defined structure so that the message is properly received.

3.2 Generator matrix

[3] Transforming codes into Hamming codes can be done through a generator matrix G . Taking a Hamming(n,k) code we define a generator matrix G of dimensions [k,n] where each row correspond to a bit of the initial message and the columns correspond to a bit of the final message. The formula for a standard generator matrix is defined as:

$$G = [I_k| - A^T]$$

Here we have a matrix comprised of two components. Its left side is simply the identity with its size determined by the length of the original message. This ensures that the first digits of the final message consist of the initial message. The A transpose ensures that the following parities are generated with their corresponding dependencies.

To apply a generator matrix we conduct matrix multiplication such that with a initial code $[ABCD]$ and the expected parities [123]: $\begin{bmatrix} A & B & C & D \end{bmatrix} G = [ABCD123]$

Example 3.1: Generator matrix
Let us take an initial code [1001] and attempt to encode it into a Hamming(8,4) with the parity relations defined as:

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Such that

$$\begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 2 & 1 & 2 \end{bmatrix}$$

This when brought to base 2 will give us a final code of [10010010]

3.3 Parity-Check matrix

[3] The parity-check matrix H is a matrix defined as a counterpart to the generator matrix such that $HG^T = 0$. It is the matrix used to check if any errors are present in the transmitted code defined as:

$$H = [A|I_{n-k}]$$

This matrix shows us whether all parities of a result have the appropriate number of 1's to give the result that is shown. This is done through multiplication

such that assuming a final code $[ABCD123]$ we have :
$H[ABCD123]^T = [000]$
Where a 0 shows a corresponding parity is correct and a 1 shows an error with the corresponding parity.

Example 3.2: Parity-Check matrix
Following from the example of the Generator matrix we will assume a final result of [10010010]. Using the definition of H we can calculate :

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore we can check the result:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

This converted to modulo 2 gives us [0000] showing that this result was without errors.
Now let us introduce 2 errors into the code such that we will be using [11110010]:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 4 \\ 4 \end{bmatrix}$$

Which converted to modulo 2 gives us [1100] showing that there has been an error.

3.4 Application of Hamming Code

Hamming codes are some of the oldest error-correction methods that are still used today. They are often used where consistency is more important then efficiency. They are also preferred where large blocks of data are transmitted at a time as their efficiency increases with size. This can be seen as in a Hamming(7,4) a data transmission efficiency is only about 57% usable data. Whilst a Hamming(72,64) has an efficiency of nearly 89% .

Before more modern data transmission techniques Hamming codes where commonly used to compensate for signal quality. With more modern hardware this use has been made obsolete by methods that although not as thorough as Hamming codes are able to more quickly process the small amount of errors that are often caused during transmission.
As a result Hamming codes are currently mostly used to eliminate errors in

hardware such as RAM. This is done through a piece of pieces of hardware known as "Error correction code memory" also known as "ECC memory". This type of data storage is often used in laptops or servers as it is a cheap and efficient solution to data corruption often caused by the high density electronics and high temperatures often caused by these setups. They are most commonly used in Scientific or Financial servers as data corruption is especially dangerous in these fields. Most modern memory chips come with some inbuilt error correction capability or are designed to use an independent ECC module. Currently the most commonly used error-correcting codes are the Hamming code (with the Hamming(72,64) being the most popular) and Hsiao codes.

4 More Codes

4.1 Modern Codes

In contrast to earlier error correcting codes like Hamming codes, which could only check a limited number of bits at once, modern error correcting codes have made significant strides in handling larger blocks of data. These advanced codes can cover more bits in a single operation, providing greater resilience against transmission errors. The Reed-Solomon Code [4], for instance, is renowned for its ability to correct errors in blocks of data spanning hundreds or even thousands of bits. By employing sophisticated encoding and decoding algorithms, Reed-Solomon codes offer robust error correction capabilities, making them indispensable in modern communication systems where data integrity is paramount.

The Reed-Solomon code is a powerful error correction technique widely employed in modern communication systems. It operates on blocks of data and adds redundancy to enable the detection and correction of errors during transmission. Unlike some other error correction codes that focus on correcting single-bit errors, Reed-Solomon codes excel at handling burst errors, where consecutive bits or symbols are corrupted.

The basic principle behind Reed-Solomon coding involves treating the data as coefficients of a polynomial. The sender encodes the original data by evaluating the polynomial at specific points and transmitting both the original data and the additional redundant symbols calculated from the polynomial. At the receiver end, the received symbols are also treated as coefficients of a polynomial. By performing polynomial division, the receiver can calculate the remainder, which corresponds to the errors introduced during transmission. Using this remainder, the receiver can then correct the errors and reconstruct the original data. One of the key advantages of Reed-Solomon codes is their ability to correct errors even when a significant portion of the transmitted data is corrupted. This makes them particularly suitable for applications where reliability is crucial, such as digital storage devices, wireless communication systems, and satellite communication.

4.2 Application

One prominent application of Reed-Solomon codes is in optical storage media, such as compact discs (CDs) and digital versatile discs (DVDs). These storage devices utilise Reed-Solomon coding to ensure the integrity of the stored data and to mitigate errors caused by scratches, dust, or other imperfections on the disc surface. In optical storage systems, data is stored as pits and lands on the surface of the disc, representing binary information. However, the physical nature of optical media makes them susceptible to various forms of damage that can introduce errors during data retrieval. Reed-Solomon codes provide a robust mechanism to detect and correct these errors, thereby enhancing the reliability of data retrieval from optical discs.

Another application of Reed-Solomon coding is in digital television broadcasting. In digital television transmission, data is encoded using various modulation techniques and transmitted over the airwaves. However, factors such as atmospheric interference, multipath propagation, and signal attenuation can degrade the transmitted signal and introduce errors. Reed-Solomon codes are employed to add redundancy to the transmitted data, allowing receivers to detect and correct errors, ensuring high-quality video and audio playback. In summary, Reed-Solomon codes represent a crucial tool for error correction in modern communication systems. Their ability to handle burst errors and their widespread adoption in various applications, including optical storage media and digital television broadcasting, highlight their importance in ensuring reliable data transmission and storage.

Reed-Solomon code is another example of error-correcting code. Introduced by Irving Stoy Reed and Gustave Solomon in 1960 [4], it has a wide range of applications in digital communications and storage.

Reed-Solomon code in the compact disc was the first use of strong error-correcting code in a mass-produced consumer product. Using a system called Cross-Interleaved Reed-Solomon Coding (CIRC) in CDs, error bursts up to 4000 bits (or 2.5mm on the discs surface) could be completely corrected. This code was further implemented in DVDs, allowing them to play sound and motion unaffected by scratches on the disc.

Reed-Solomon code proves useful in two-dimensional bar codes. It allows 2D bar codes to be read by scanners, even if pieces of the code is damaged by, for example, being wrinkled on a package or getting wet.

A notable application of Reed-Solomon code was encoding digital pictures sent back by the Voyager program - a program launched in 1977 where two spacecrafts were sent off in exploration of Jupiter, Saturn, Uranus and Neptune. The Voyagers communication system achieved a data rate of 21,600 bits per second from 2 billion miles away and required less energy than a common wristwatch battery. Their mission to collect data for transmission back to Earth was made possible by Reed-Solomon code and the Voyager spacecrafts are still active to this day, being the farthest man-made object in space.

The Hubble Telescope, launched in 1990, also used Reed-Solomon code, as

well as Mars Pathfinder, Mars Exploration Rover, Galileo, and Cassini missions.

5 Challenges

While error correction methods such as repetition codes and Hamming codes have significantly improved the reliability of data transmission, it is essential to acknowledge their limitations. These methods, though effective, are not infallible and can still produce errors. However, the pursuit of error-free communication is not confined to mathematical techniques alone.

Innovations beyond mathematical approaches also play a crucial role in enhancing data integrity. For instance, upgrading transmission mediums from copper wires to carbon fiber can reduce susceptibility to interference and signal degradation, thereby minimizing transmission errors. Similarly, advancements in signal processing algorithms, error detection protocols, and hardware design contribute to bolstering the resilience of communication systems.

Combining diverse methods, both mathematical and non-mathematical, offers a multifaceted approach to error correction, increasing the likelihood of error-free communication and inching closer to perfection. By leveraging a spectrum of techniques tailored to address specific challenges, we can mitigate the impact of errors and optimize data transmission reliability.

In the ever-evolving landscape of communication technology, the quest for perfection remains ongoing. While complete elimination of errors may be an idealistic goal, continuous innovation and integration of diverse methodologies pave the way for increasingly reliable and robust communication systems. As we navigate towards this vision of error-free communication, collaboration across disciplines and relentless pursuit of improvement will drive us closer to achieving this elusive goal.

6 Further Reading

The study of error-correcting codes is a deep and rich field, bridging theoretical mathematics with practical applications in computer science, telecommunications, and digital data storage. For those interested in expanding their understanding beyond the basics of Hamming codes, repetition codes, and other fundamental error-correcting strategies, the following resources are highly recommended:

- **Toff K. Moon, "Error Correction Coding: Mathematical Methods and Algorithms, John Wiley & Sons, 2003"**
- **W. Cary Huffman and Vera Pless, "Fundamentals of Error-Correcting Codes, Cambridge University Press, 2003, ISBN 978-0-521-78280-7"**
- **David J. C. MacKay, "Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003, ISBN 978-0-521-64298-9"**
- **Elwyn Berlekamp, "The Performance of Block Codes"**
- **Stephen B. Wicker and Vijay K. Bhargava, "Reed-Solomon Codes and Their Applications. IEEE Press, 1994"**

References

[1] Peter J. (Peter Jephson) Cameron. *Introduction to algebra*. 2nd ed. Oxford mathematics. Oxford: Oxford University Press, 2008.

[2] R. W. Hamming. "Error detecting and error correcting codes". In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[3] W. Cary Huffman. *Fundamentals of Error-Correcting Codes*. Cambridge: Cambridge University Press, 2003.

[4] I. S. Reed and G. Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304.

[5] Michael Williams. "Reckoners: The Prehistory of the Digital Computer, from Relays to the Stored Program Concept, 1935-1945". In: *Technology and Culture* 26.2 (1985), pp. 339–341.