Exploring Coding Theory With Linear Algebra

Team No-37 : Priya Mandot(2022102053), Jayesh Sutar(2022101001), Abhinav S(2022102037)

Abstract

Coding theory is a branch of mathematics that deals with the design and analysis of error-correcting codes, which are essential for reliable data transmission and storage. This project focuses on exploring the interplay between coding theory and linear algebra, demonstrating how linear algebraic techniques and concepts are utilized to study and construct efficient error-correcting codes. The project begins by providing an introduction to coding theory,Linear codes and Parity Check Matrices.It then delves into the Error Correction techniques and use of linear codes in security.Furthermore, the project touches upon Examples of coding Theory and Applications.

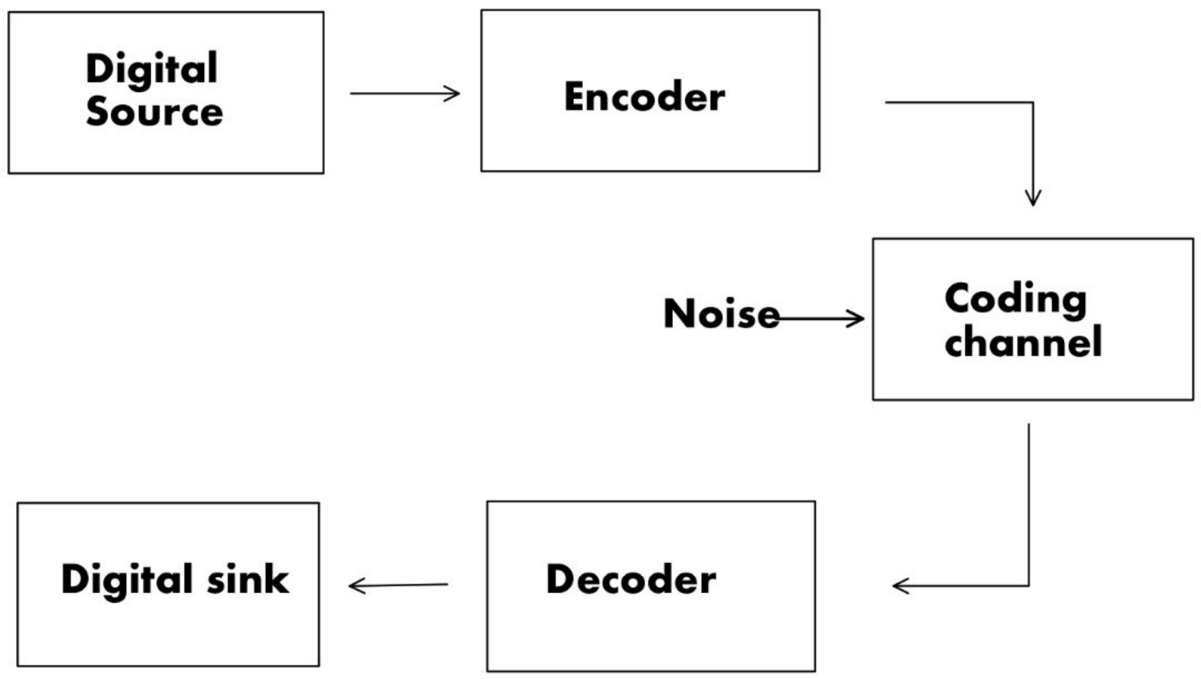
1 State of the Art Literature

1.1 Hamming Codes:Error Reducing Techniques-by Rohitkumar R Upadhay

This paper generally focuses on binary linear codes. In this case, the messages are encoded in blocks of bits, called codewords, and any modulo-2 linear combination of codewords actually is also a codeword, or so they generally thought. This paper discusses some potential Decoding Methods to beats Standard Decoding Algorithms literally is followed by several algorithms that attempt to maximize the reduction in errors along with ananalysis of the performance and scalability of each algorithm, which literally is fairly significant. firstly, it discusses Standard Decoding Methods on hamming Codes and Error-reduction limits on Standard Decoding techniques.Then it discussed two methods of Decoding: 1.Minimum Sum Decoding 2.Minimum of Maximum Decoding

1.2 A Mathematical Theory of Communication- by Shannon

In 1948 Shannon, published the paper “A Mathematical Theory of Communication” which stated that by proper encoding of information, transmission and noise errors can be reduced without sacrificing the transmission or storage rate.2 As a result of this groundbreaking research, rigorous effort has been put into design and efficient encoding and decoding methods for error control.



1.3 Low Density Parity Check Code(LDPC)-by Soumya Borwankar and Dhruv Shah

This paper basically expresses the core fundamentals and brief overview of the research of R. G. GALLAGER [1] on Low-Density Parity-Check (LDPC) codes and various parameters related to LDPC codes like, encoding and decoding of LDPC codes, code rate, parity check matrix, tanner graph.

2 Introduction to Coding Theory

2.1 Definition and Significance of Coding Theory

Recently, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. This demand is being accelerated by the unfolding of large scale high speed data networks which exchange, process and store the digital information in various fields. Ensuring reliable reproduction of data by controlling the errors is an important area of focus for the designers.

A typical data transmission or storage system involves a source encoder that transforms thesource information into a sequence of binary digits(bits) called the information sequence. This conversion is done such that the number of bits per unit time required to represent the source output is minimal and can be reconstructed without ambiguity. This sequence is then converted into a discrete encoded sequence which is modulated to make it suitable for transmission through the coding channel. It is during this transmission that the data is corrupted by noise. This corrupted data is then received by a demodulator which produces a discrete sequence to be processed by the decoder. The decoding is based on the rules of channel encoding and the noise characteristics of the channel. It results in an estimate of the original source output and delivers it to the destination. This paper mainly focuses on the design, analysis and implementation of codes that can detect and correct errors that occur during data transmission and storage.

Coding theory is the branch of mathematics and computer science that deals with the study of error correcting codes. The primary goal of coding theory is to ensure reliable and secure communication in the presence of noise, interference, or other transmission errors. By adding minimal redundancy to the original data, coding theory aims to create encoded versions (codewords) that can withstand errors and be efficiently decoded to recover the original information. Codes are mathematical constructs that map a set of input symbols (such as bits or characters) to a set of output symbols (codewords). The design of these codes involves carefully choosing the mapping rules to provide error detection and correction capabilities. Mathematical techniques such as algebraic structures, combinatorics, probability theory and linear algebra are used. Linear codes, Hamming codes Reed-Solomon codes, turbo codes and LDPC codes are some examples of error-correcting codes studied in coding theory.

Coding theory finds numerous practical applications in data storage and retrieval, modern digital wireless communication, network communication, cryptography, security, DNA sequencing and molecular computing. The field continues to evolve, and new applications are being explored, such as machine learning, quantum computing, and data compression. The principles and techniques of coding theory provide the foundation for reliable and secure communication in numerous domains, contributing to advancements in technology and information processing.

2.2 Role of error-correcting codes in reliable communication

Noise and interference in the communication channels can introduce distortions to the transmitted data leading to inaccurate or incomplete information at the receiver. Interference involves signal attenuation, electromagnetic obtrusion, congestion, cross-talk, multi-path propagation or channel fading in wireless communication. This could lead to faulty decision making, miscommunication and erroneous analysis.In the digital world, information is passed as binary bits and during transmission individual bits can be flipped or transposed. Even a single bit error can have cascading effects leading to significant data inaccuracies.4 Thus, this shows the importance of error correcting codes to mitigate the impact of noise and overall improve the accuracy and reliability of the received data.

Error-correcting codes provide mechanisms to detect and correct errors due to corruption of the information during channel transmission, thus increasing the accuracy of the communicated information. These provide information on the location of the error giving the system the ability to correct them. Redundancy is introduced in a structured and systematic manner to compensate for a certain number of errors and ensure accurate retrieval.

These redundant/parity bits are based on mathematical operations performed on original source output. The resulting codeword after encoding consists of the original data along with properly placed parity bits. On receival the decoding algorithm analyzes the parity bits and the received data to identify position of errors, which can then be flipped to their original state.

The effectiveness of these codes is determined by

- Code rate: The ratio of number of information bits to the total number of bits in the codeword. A higher rate results in efficient use of the communication channel but typically provide less error-correcting capabilities.
- Minimum distance: Represents the minimum number of bit changes required to convert one valid codeword into another. A higher minimum distance provides better error detection and correction. This distance is equal to the number of differing bits between two codewords.
- Error-Correcting capability: Code's ability to correct a certain number of errors. To correct d single bit errors, a minimum distance of 2d+1 is required.
- Complexity: Highly complex algorithms may require significant computational resources making them less desirable in certain applications.

2.3 Overview of linear algebra's fundamental role in coding theory

1. **Error Correction**:Linear algebra provides tools to design error-correcting codes. Linear codes,which are a class of codes widely used in practice, utilize linear algebraic concepts to encode and decode data. Linear codes are defined using linear combinations and linear transformations, allowing for efficient error detection and correction.
2. **Code construction**: Linear algebra techniques are employed to construct codes with desirable properties. For example, cyclic codes are a class of linear codes with efficient encoding and decoding algorithms, and their structure is based properties of finite fields and polynomial arithmetic.
3. **Code analysis**: Linear algebra allows for the analysis of code properties and performance. The weight distribution of a code, which determines its error-correcting capabilities, can be analyzed using linear algebraic techniques such as matrix rank and null space computations.
4. **Decoding algorithms**: Linear algebra provides the foundation for many decoding algorithms. For instance, the widely used syndrome decoding algorithm uses linear algebraic operations to compute error syndromes and correct errors in received codewords.

3 Linear Codes

3.1 Definition and properties of linear codes

3.1.1 Definition

Linear codes are a vector subspace of a Galois field GF(q)(or finite fields).By definition of vector subspace, a subset C of GF(q) is a linear code if and only if:

$$u + v \in C \qquad \forall u, v \in C \tag{1}$$

$$a \cdot u \in C \qquad \forall a \in GF(q) \tag{2}$$

The dimension k of this vector subspace is the number of codewords, n is the length of the code, and the linear code is denoted as a q-ary[n, q^k]-code. If a minimum distance is specified this is a [n, q^k, d]-code. Square brackets denote the linear aspect of the code.

3.1.2 Properties

- The weight of the smallest, non-trivial (non zero) codeword is the minimum distance of the code.
- Generator Matrices: These codes can be defined by listing only the asis vectors of the code's subspace, instead of listing every single codeword. This generator matrix can be obtained from the matrix of the entire code by the use of row reduction. The entire code can be regenerated by expressing all the linear combinations of the rows of the generator matrix.For example

  The [4,8] code $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ has 2^k=8 ,implies k=3 and reduces the generator matrix

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

  The code can then be "re-generated" by creating all linear combinations of rows of the generator matrix:

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \text{x[1 0 0 1] + y[0 1 0 1] + z[0 0 1 1]}$$

As an example letting x=1,y=1 and z=0 generates the second codeword in the original codeward list as follows:

$$[1\ 0\ 0\ 1] + [0\ 1\ 0\ 1] = [1{+}0\ 0{+}1\ 0{+}0\ 1{+}1] = [1\ 1\ 0\ 0]$$

Repeating this process for all combinations of 0 and 1 for all x,y,z will generate the entire linear code. Any generator matrix of the form [I_k—A], with I_k being the identity matrix of dimension k, is said to be in standard-form. Codewords generated from the standard form have their first k digits match the message vector. An example is given in the next section.

3.2 Encoding and decoding processes using linear algebraic operations

Codewords themselves are encoded messages. Part of the codeword is the message digits, and the additional portion of the codeword is redundancy to allow for error detection and correction.

Let v_1,v_2 ...v_k be the message vectors to be encoded. Multiplying these with the generator matrix G on the right produces the codeword. The rows of the generator matrix G are represented as r_1,r_2,...r_k Symbolically:

$$v \cdot G = \sum v_i \cdot g_i \tag{3}$$

Let G for a [7,4,4]-code:

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

and the message vector to be encoded: v = 111,and

$$c = v \cdot G \tag{4}$$

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

Encoding the same message using a standard form generator matrix results in,

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

Because this codeword is generated from a matrix in standard form, the first kth digits of the codeword will match the message vector. The remainder of the codeword is redundancy bits.

$$1111000 = \begin{bmatrix}1 & 1 & 1 & \mid & 1 & 0 & 0 & 0\end{bmatrix}$$

3.2.1 Parity Check Matrix

For any linear code, the set of all vectors orthogonal to every codeword is known as the dual-code and is denoted by $C^{\perp}$

$$\text{For Example if C} = \begin{bmatrix}0 & 0 & 0\\1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix} \text{ then } C^{\perp} = \begin{bmatrix}0 & 0 & 0\\1 & 1 & 1\end{bmatrix}$$

Then generator Matrix for $C^{\perp}$ is Parity Check matrix of C itself The parity-check matrix H can be obtained from G, the generator matrix of C,

$$\text{G} = \left[\begin{array}{c|cccc} & a_{1,1} & a_{2,2} & \cdots & a_{1,n-k}\\ I_k & \vdots & \vdots & \vdots & \vdots\\ & a_{k,1} & a_{k,2} & \cdots & a_{k,n-k}\end{array}\right]$$

then

$$\text{H} = \left[\begin{array}{cccc|c} a_{1,1} & a_{2,2} & \cdots & a_{1,n-k} & \\ \vdots & \vdots & \vdots & \vdots & I_k\\ a_{k,1} & a_{k,2} & \cdots & a_{k,n-k} & \end{array}\right]$$

Using same generating matrix from above,

$$\text{G} = \begin{bmatrix}1 & 1 & 0 & 1 & 0 & 0 & 1\\0 & 0 & 0 & 1 & 1 & 1 & 1\\0 & 1 & 1 & 1 & 1 & 0 & 0\end{bmatrix} \text{,then A} = \begin{bmatrix}1 & 0 & 1 & 1\\1 & 1 & 0 & 1\\1 & 1 & 1 & 0\end{bmatrix}$$

thus

$$\text{H} = [-A^T|I_k] = \begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0\\0 & 1 & 1 & 0 & 1 & 0 & 0\\1 & 0 & 1 & 0 & 0 & 1 & 0\\1 & 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}$$

In the binary case the negative signs have no effect since -1 =1. Using the parity-check matrix it is possible to describe the entire code C as follows:

for a vector x of length n,$\{x \in C | x \cdot H^T = 0\}$

3.3 Examples of Linear Correction Linear Codes

- Brute Force Repetition:
  A very obvious coding scheme is to repeat information. For example, the scheme of three, where [1] would be passed as [1,1,1] and [0] as [0,0,0]. This has a code rate =1/3 and while decoding the majority vote of each block of three received bits is chosen. This has the capability of detecting only one error. By using longer repetition codes, a low probability of error can be obtained, but the rate approaches zero, thus making it a very inefficient error correcting scheme. Also, overhead is large, here there is a 200% increase in data size.
  just repeating the bits is fairly inefficient. We could do better if we could have a compact way to figure out which bit got flipped (if any).

- Hamming Distance:
  A key issue in designing any error correcting code is making sure that any two valid codewords are sufficiently dissimilar so that corruption of a single bit (or possibly a small number of bits) does not turn one valid code word into another. To measure the distance between two codewords, we just count the number of bits that differ between them. This count is called the Hamming distance.

  - For Error Correction:
    To design a code that can detect d single bit errors, the minimum Hamming distance for the set of codewords must be d + 1 (or more). That way, no set of d errors in a single bit could turn one valid codeword into some other valid codeword
  - For Error Detection:
    To design a code that can correct d single bit errors, a minimum distance of 2d + 1 is required. That puts the valid codewords so far apart that even after bit errors in d of the bits, it is still less than half the distance to another valid codeword, so the receiver will be able to determine what the correct starting codeword was.

3.3.1 Hamming Codes

When retransmission or feedback from sender is not a feasible option, forward error correction techniques are used to allow the receiver to detect and correct errors. Hamming codes provide Forward Error Correction(FEC) using a ”block parity” mechanism that can be inexpensively implemented. In general, their use allows the correction of single bit errors and detection of two bit errors per unit data, called a code word.

The fundamental principal embraced by Hamming codes is parity. Hamming codes, as mentioned before, are capable of correcting one error or detecting two errors but not capable of doing both simultaneously. You may choose to use Hamming codes as an error detection mechanism to catch both single and double bit errors or to correct single bit error. This is accomplished by using more than one parity bit, each computed on different combination of bits in the data.

The amount of parity data added to Hamming code is given by the formula 2p $\geq$ d + p + 1, where p is the number of parity bits and d is the number of data bits. We only consider cases for p $\geq$ 3.

The case of p=3 is used in the following discussion to develop a (7,4) code using even parity that fixes a single bit error, but larger code words are typically used in applications. A code where the equality case of Equation 1 holds is called a perfect code of which a (7,4) code is an example.

A Hamming code word is generated by multiplying the data bits by a generator matrix G using modulo-2 arithmetic. This multiplication's result is called the code word vector $(c_1, c_2, c_3, ...c_n)$, consisting of the original data bits and the calculated parity bits.

The generator matrix G used in constructing Hamming codes consists of I (the identity matrix) and a parity generation matrix A: G = [I_4 — A].

An example of Hamming code generator matrix:

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \text{ parity check matrix } H = [A^T|I_3] : H = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Consider the set of vectors of length 7 in the null space of H (the vectors which when multiplied by H give 000). From the theory of linear spaces, since H has rank 3, we expect the null space of H to have dimension 4. These 2^4 codewords are

0000000	0001111	0010110	0011001
0100101	0101010	0110011	0111100
1000011	1001100	1010101	1011010
1100110	1101001	1110000	1111111

Since the set of codewords is the null space of a matrix, it is linear in the sense that the sum of any two codewords is also a codeword. The set of codewords therefore forms a linear subspace of dimension 4 in the vector space of dimension 7.

Looking at the codewords, we notice that other than the all-0 codeword, the minimum number of 1's in any codeword is 3. This is called the minimum weight of the code. We can see that the minimum weight of a code has to be at least 3 since all the columns of H are different, so no two columns can add to 000. The fact that the minimum distance is exactly 3 can be seen from the fact that the sum of any two columns must be one of the columns of the matrix. Hence if a codeword c is corrupted in only one place, it will differ from any other codeword in at least two places and therefore be closer to c than to any other codeword.

The multiplication of a 4-bit vector (d_1, d_2, d_3, d_4) by G results in a 7-bit code word vector of the form $(d_1, d_2, d_3, d_4, p_1, p_2, p_3)$. It is clear that the A partition of G is responsible for the generation of the actual parity bits. Each column in A represents one parity calculation computed on a subset of d. The Hamming rule requires that p=3 for a (7,4) code, therefore A must contain three columns to produce three parity bits.

If the columns of A are selected so each column is unique, it follows that (p_1, p_2, p_3) represents parity calculations of three distinct subset of d. As shown in the figure below, validating the received code word r, involves multiplying it by a parity check to form s, the syndrome or parity check vector.

Column of A	Parity bits	Parity Calculation
1	p_1	$d_1 \oplus d_3 \oplus d_3$
2	p_2	$d_1 \oplus d_2 \oplus d_4$
3	p_3	$d_1 \oplus d_2 \oplus d_3$

Example:
Original message m: 1001
At the transmitter, Codeword C = $m \cdot G$ = 1001001
At the receiver, Received r = 1011001
By Syndrome decoding s = $r \cdot H^T$
if there, are no errors s = 0
if a bit error has occurred,s != 0
here, $r \cdot H^T$ = 101, matches the third coloumn of H => error in $3^r d$ bit

4 Syndrome Based Decoding Algorithms

4.1 Introduction

In the realm of digital communication and data storage, errors are an inevitable occurrence.Whether it's due to noisy channels, hardware malfunctions, or even natural phenomena, such errors can corrupt the integrity of transmitted or stored information. To combat this challenge, researchers and engineers have devised various error correction techniques to detect and correct these errors efficiently.

In these Error Correction Methods.One of the error correction method is the syndrome-based decoding algorithm. The syndrome-based decoding algorithm operates on the principles of linear codes and employs advanced mathematical concepts to analyze and correct errors.

4.1.1 What is Syndrome

Let's consider a linear code C, which is a subspace of the vector space F^n over a finite field F. This code is defined by a generator matrix G, which spans the code and provides a systematic representation of the encoded data.

Given a received vector$r \in F^n$, the syndrome S of r with respect to the code C is computed as the product of the received vector r and the transpose of the parity-check matrix H of the code:

S $= r \cdot H^T$

Here, H is a parity-check matrix that defines the code C. It is constructed in such a way that the product of a codeword (a vector in C) with the transpose of H is always zero.By examining the syndrome, error correction algorithms can determine the error patterns and locations within the received vector. This information allows for the precise identification and correction of errors, ultimately recovering the original error-free data.

4.2 Syndrome Decoding using Syndrome Table

Syndrome-based decoding is an important algorithm in coding theory used to correct errors in encoded messages. It relies on the concept of syndromes, which are derived from the received message and the generator matrix of the code. Here is the step by step explanation of this algorithm:

1.Encoding: The sender encodes the original message using an error-correcting code. This code typically involves adding redundant bits to the original message to create a codeword.

2.Transmission: The codeword is transmitted over a noisy channel, which may introduce errors in the received message.

3.Syndrome calculation: The receiver receives the corrupted message and calculates the syndrome. The syndrome is obtained by multiplying the received message by the transpose of the parity-check matrix of the code. The parity-check matrix contains the information about the code's structure and is used to detect errors.

4.Syndrome analysis: The receiver compares the calculated syndrome with a precomputed table or database of syndromes associated with specific error patterns. Each syndrome corresponds to a particular error pattern, which helps identify the location and type of errors.

5.Error correction: Based on the identified error pattern, the receiver applies appropriate error correction techniques to correct the errors. This typically involves flipping the bits at the error locations to restore the original message.

6.Decoding: The corrected message is then extracted by removing the redundant bits added during the encoding process, resulting in the original message

Syndrome-based decoding exploits the algebraic properties of error-correcting codes and the syndrome calculations to identify and correct errors. It is commonly used with linear block codes, such as Hamming codes, Reed-Solomon codes, and Bose-Chaudhuri-Hocquenghem (BCH) codes.

The algorithm can be further enhanced using advanced techniques like iterative decoding, which iteratively refines the error correction process by iteratively calculating syndromes and updating error estimates. Other algorithms, such as the maximum likelihood decoding and belief propagation algorithms, are used for more complex codes like turbo codes and LDPC codes.

Example: Let consider the example of Hamming Code(7,4).This encodes 4-bit message in 7-bit codeward. The original message consists of 4 bits, let's say "1010". The encoder takes this message and applies a linear transformation using a generator matrix to obtain a 7-bit codeword.

$$\text{Let,G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Which will give 1010110 as keyword.Now The codeword "1010110" is transmitted over a noisy channel, potentially introducing errors. Let's say the received codeword is "1010010".

The receiver calculates the syndrome by multiplying the received codeword by the transpose of the parity-check matrix of the code. For the Hamming(7,4) code, the parity-check matrix is:

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

To obtain syndrome of given codeward we use Error Syndrome Equation which is given by

$$Syndrome = c \cdot H^T \tag{5}$$

where c is received codeword and H is parity-check matrix. Now if there is error occurs duration cummunation then

$$c = v + e \tag{6}$$

where v is original codeword and e is error vector by multiplying both sides with H^T,

$$c \cdot H^T = v \cdot H^T + e \cdot H^T \tag{7}$$

Since v is codeword $v \cdot H^T = 0$,So

$$syndrome = c \cdot H^T = e \cdot H^T \tag{8}$$

For all single error patterns e_i we can obtain Corresponding Syndrome

Syndrome analysis: The receiver compares the calculated syndrome "110" with a precomputed table of syndromes associated with error patterns. For the Hamming(7,4) code, the error patterns and their corresponding syndromes are:

Table 1: This is caption for the table

Syndrome	Error Pattern
000	No error
111	Error in bit 1
101	Error in bit 2
011	Error in bit 3
110	Error in bit 4
100	Error in bit 5
010	Error in bit 6
001	Error in bit 7

As the syndrome "110" matches the entry for "Error in bit 5," the receiver can conclude that there is an error in the 5^th bit of the received codeword.

Error correction: The receiver corrects the error by flipping the 5^th bit of the received codeword. The corrected codeword becomes "1010110".

Decoding: The receiver removes the redundant bits from the corrected codeword, which results in the original 4-bit message. The decoded message is "1010", which matches the original message.

4.3 Syndrome decoding using Error locator polynomial

4.3.1 Error Locator Polynomial

The error locator polynomial is a mathematical representation used in certain error correction algorithms, such as the Reed-Solomon codes,BCH codes. It is denoted by $\wedge$(x) and is defined as a polynomial over a finite field F.

Mathematically, the error locator polynomial $\wedge$(x) is represented as:

$\wedge(\mathrm{x}) = 1 + \wedge_1 x + \wedge_2 x^2 + ... + \wedge_{m-k} x^{m-k}$

In this equation, $\wedge_i$(for i = 1 to m-k) represents the coefficients of the error locator polynomial. These coefficients are elements from the finite field F, and x is the indeterminate or variable.

4.3.2 General Steps of Algorithm

1. **Syndrome Calculation**: The received codeword is multiplied by the transpose of the parity-check matrix to obtain the syndrome. The syndrome is a set of values that indicate the presence and locations of errors in the received codeword.

2. **Syndrome Calculation**: The received codeword is multiplied by the transpose of the parity-check matrix to obtain the syndrome. The syndrome is a set of values that

indicate the presence and locations of errors in the received codeword.

3. **Syndrome-to-Polynomial Conversion**: The syndromes are interpreted as coefficients of the error locator polynomial. The error locator polynomial is constructed by considering the syndromes as the coefficients in descending order of powers of x.

4. **Syndrome Polynomial Division**: The error locator polynomial is divided by the generator polynomial associated with the code. This division process is performed using mathematical algorithms, such as polynomial long division or Euclidean algorithm, depending on the specific code.

5. **Error Location**: The roots of the error locator polynomial are determined. The roots represent the error locations in the received codeword. The number and positions of the roots indicate the number and positions of errors.

6. **Error Correction**: Based on the error locations identified, the errors are corrected by modifying the corresponding bits in the received codeword.

4.3.3 Example

Consider a BCH code in GF(2^4) with d=7 and g(x) = x^{10} + x^8 x^5 +x^4+ x^2 + x + 1.

Let the message to be transmitted be [1 1 0 1 1], or in polynomial notation, M(x) = x^4 + x^3 + x + 1.by dividing $x^1$0M by g(x) and taking the remainder, resulting in x^9 + x^4 + x^2 or [1 0 0 0 0 1 0 1 0 0]).These are appended to the message, so the transmitted codeword is [1 1 0 1 1 1 0 0 0 0 1 0 1 0 0]

Now, imagine that there are two bit-errors in the transmission, so the received codeword is [1 0 0 1 1 1 0 0 0 1 1 0 1 0 0].In polynomial notation: R(x) = C(x) + x^{13} + x^5 + x^{14} + x^{11} + x^{10} + x^9 + x^5 + x^4 + x^2
IN order to correct the errors, first calculate the syndromes. Taking α = 0010, we have s_1 = R(α^3) = 1011, s_2 = 1001, s_3 = 1011, s_4 = 1101, s_5 = 0001, and s_6 = 1001. Next, apply the Peterson procedure by row-reducing the following augmented matrix.

$$[S_{3x1} — C_{3x3}] = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \\ s_2 & s_3 & s_4 & s_5 \\ s_3 & s_4 & s_5 & s_6 \end{bmatrix}$$

$$= \begin{bmatrix} 1011 & 1001 & 1011 & 1101 \\ 1001 & 1011 & 1101 & 0001 \\ 1011 & 1101 & 0001 & 1001 \end{bmatrix} = \begin{bmatrix} 0001 & 0000 & 1000 & 0111 \\ 0000 & 0001 & 1011 & 0001 \\ 0000 & 0000 & 0000 & 0000 \end{bmatrix}$$

Due to the zero row, S_{3x3} is singular, which is no surprise since only two errors were introduced into the codeword. However, the upper-left corner of the matrix is identical to [S_{2x2} | C_{2x1}], which gives rise to the solution λ_2 = 10000, λ_1 = 1011. The resulting error locator polynomial is $\wedge$(x) = 1000x_2 + 1011x + 0001, which has zeros at 0100 = α^-13 and 0111 = α^-5. The exponents of α correspond to the error locations.As -13 means we have error at 14^th bit from the end and -5 means we have error at 5^th bit from the end.

5 Ensuring Reliability and Security

5.1 Introduction

In our increasingly interconnected and data-driven world, the reliable transmission and secure storage of information have become paramount. Whether it's transmitting sensitive data across communication channels or safeguarding valuable data in storage systems, ensuring both reliability and security is a critical challenge.

Linear algebra forms the bedrock of many cryptographic techniques and error correction codes.such As Linear block are used in McEliece cryptosystem.

5.2 Code Performance Analysis

Code performance analysis evaluating various parameters and properties of error-correcting codes to assess their effectiveness in detecting and correcting errors.code performance analysis provides insights into the error-correcting capabilities, efficiency, and limitations of different codes. This analysis helps in selecting appropriate codes for specific applications and optimizing their performance. Here are some key aspects considered in such analysis:

Minimum Distance: The minimum distance of a code, denoted as d, measures the smallest number of bit or symbol positions in which any two distinct codewords differ. Higher minimum distance generally leads to better error-correcting capability.

Hamming Bound: The Hamming bound is a theoretical limit on the performance of a code, given by the formula d $\leq$ n - k + 1, where n is the codeword length and k is the number of message bits. It provides an upper bound on the achievable minimum distance for a given code.

Error Detection: The ability of a code to detect errors is determined by the minimum distance. A code with minimum distance d can detect up to (d - 1) errors. By analyzing the code's minimum distance, you can evaluate its error detection capabilities.

Error Correction: Error correction is achieved by exploiting the algebraic properties of the code. Linear codes, for example, can correct up to floor((d - 1)/2) errors. By studying the minimum distance and algebraic properties, you can assess the code's error correction capabilities.

Syndrome Decoding: Syndrome decoding is a technique used to identify error patterns and correct them. It involves calculating the syndrome of the received codeword using parity-check matrices and then determining the error locations and correcting them based on the syndrome information.

Performance Bounds: Various bounds, such as the Singleton bound, Gilbert-Varshamov bound, and Plotkin bound, provide theoretical limits on the achievable performance of codes. These bounds help in assessing the effectiveness of codes in terms of their error-correcting capabilities.

Encoding and Decoding Complexity: Analyzing the encoding and decoding complexity of a code is essential to evaluate its practical implementation. Codes with low encoding and decoding complexity are desirable in applications where real-time or efficient processing is required.

5.3 Linear codes in Cryptography

Linear codes, particularly in the context of the McEliece cryptosystem, play a crucial role in cryptography. The McEliece cryptosystem is an asymmetric encryption scheme based on the hardness of decoding random linear codes.

Here's how linear codes are utilized in the McEliece cryptosystem:

Key Generation: The key generation process involves creating a linear code that will be used for encryption and decryption. The code is typically a binary linear code defined by its generator matrix or parity-check matrix. The choice of the code parameters, such as code length, dimension, and minimum distance, impacts the security and efficiency of the cryptosystem.

Encryption: To encrypt a message in the McEliece cryptosystem, the original message is first encoded using the chosen linear code. This encoding process involves multiplying the message vector by the generator matrix of the code, resulting in a codeword. Random error bits are then added to the codeword to introduce noise.

Error Correction: In the McEliece cryptosystem, the inherent error-correcting capability of the linear code is crucial for the security of the scheme. The recipient of the encrypted message performs error correction by decoding the received codeword using the parity-check matrix of the code. The error correction process corrects the added errors and retrieves the original codeword.

Decryption: Once the error correction is performed, the recipient obtains the decoded codeword. To obtain the original message, the codeword is multiplied by the inverse of the generator matrix, resulting in the original message vector.

The security of the McEliece cryptosystem relies on the hardness of decoding random linear codes, which is known to be computationally challenging. By using a random linear code, the encryption process introduces a layer of complexity that makes it difficult for adversaries to retrieve the original message without knowledge of the private key.

Linear codes, with their well-established properties and efficient decoding algorithms, provide the necessary foundations for the McEliece cryptosystem.

6 Low Density Parity Check Codes

6.1 Introduction

It is a class of linear block codes. The name comes from the fact that the parity matrix contains fewer 1's than 0's. It provides performance very close to the capacity of several channels, and decoding has linear time complexity. It was first used by Gallagher in his PhD thesis in 1960. It was mostly ignored due to Reed-Solomon codes discussed briefly later, as well as the computational effect in implementing decoders for it.

6.2 Representation

LDPC codes can be represented via matrices as well as graphs.

1. Matrix representation: The following matrix represents a Parity matrix $\mathbf{H}_{n\times m}$ for a (8,4) code:

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}_{4\times 8}$$

Here, for each row, we can define two values ω_r and ω_c which denote the number of ones in each row and column, respectively. $\mathbf{H}$ is of low density if and only if $\omega_r \ll n$ for each row and $\omega_c \ll m$ for each column. For this, the parity check matrix should be quite large. So, this example cannot be considered as a low-density parity check matrix.This is given just for ease in explanation and for graphical representation.

2. Graphical representation: Tanner introduced a way to represent it using what was named Tanner graphs, which are bipartite graphs. These also help to describe the decoding algorithm. In bipartite graphs,nodes of the graph are separated into two different types, and there is a connection only between nodes of two different types,nothing between them. The two types of nodes were named by Tanner as check nodes (c-nodes) and variable nodes (v-nodes).

The graph has m check nodes,equal to the number of parity bits, and n variable nodes,equall to the number of bits in the codeword. Let the check nodes be represented by f_i and variable nodes by c_i. The graph is defined such that there is an edge from f_i to c_j if $H_{ij} = 1$.

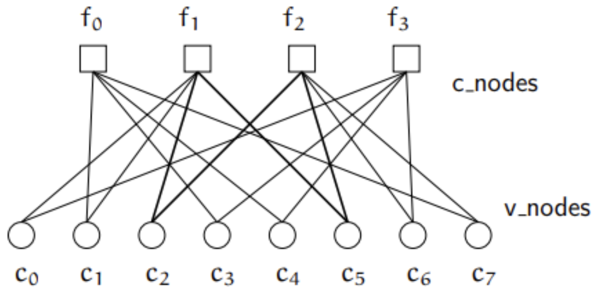


Figure 1: Tanner Graph

6.3 Constructing LDPC codes

Constructing LDPC codes is very easy.Any Sparse matrix will do the trick.But in case of random matrix,the complexity of encoding is also high as sometimes some check nodes might be connected to more variable nodes than others which nullifies one of advantage of LDPC codes which is parallel computation.

6.4 Complexity, Performance and Iterative Decoding

The LDPC codes work close to capacity as describes in Shannon's theorem only for very large block lengths which means a very large parity check matrix as well.We can represent **H** as

$$\mathbf{H} = \left[\mathbf{P}^T \,\middle|\, \mathbf{I}\right]$$

From this we can find generator matrix G using Gaussian elimination and can be represented as

$$\mathbf{G} = [\mathbf{I} \,|\, \mathbf{P}]$$

This submatrix **P** is generally not sparse.So,multiplication with codeword will take more time.So,encoding complexity is higher.If block length gets very high,due to this,the performance will not be good as complexity increases in O(n^2).

In order to solve this problem,a divide and conquer iterative algorithm is employed.Instead of doing such complex calculation,a series of local calculations is done and message is sent between check and variable nodes.This approach breaks down the problem into manageable sub-problems.During iterative decoding,each node performs local calculation based on received codeword and parity check equations.This involves updating probabilities of bit values of the message based on the received information.These local calculations reduce the complexity of algorithm.

The sparse matrix provided an advantage here as each check node only has a small subset of variable nodes reducing computational complexity.In iterative decoding,check nodes and variable nodes exchange information between them each time updating the likelihood of bit values of the original codeword.Upon,doing this iteratively,we arrive at the original codeword. It has performance very close to that of maximum likelihood decoder which is practically impossible.So,it has a lot of practical applications.

6.5 Hard Decision decoding of LDPC codes

Now,a type of iterative decoding used in LDPC codes called Hard Decision decoding is explained using an example. Let us consider an example of a codeword suppose

$$\mathbf{c} = \begin{bmatrix}1 & 0 & 0 & 1 & 0 & 1 & 0 & 1\end{bmatrix}$$

Here,the sparse matrix considered is the example given during explanation of representation. Due to noise in the channel,suppose 1 bit flipped,let's say bit c_1. Suppose due to that,the receiver receives

$$\mathbf{y} = \begin{bmatrix}1 & 1 & 0 & 1 & 0 & 1 & 0 & 1\end{bmatrix}$$

The steps in correcting error and decoding original codeword include

1.First,all the v-nodes c_i send message to their c-nodes f_j which will be the bit that they think is their correct one.The only information they have to go by is their corresponding received bit y_i.For example,v-node c_0 sends bit 1 to c-nodes f_1 and f_3.Similarly,v-node c_1 sends bit y_1 which is bit 1 to c-nodes f_1 and f_0.

2.In response to this,all check nodes sends a message to each of its variable nodes.Each check node is connected to 4 variable nodes.So,each check nodes receives information from 3 of its variable nodes and finds what the fourth bit should be such that the parity equations are satisfied and sends it to that node.Here,if no change in bits of variable nodes is happening,it means the error has been corrected and original codeword has been retrieved.

c-node	**Received Messages**	**Sent Messages**
f_0	$c_1 \to 1$, $c_3 \to 1$, $c_4 \to 0$, $c_7 \to 1$	$0 \to c_1$, $0 \to c_3$, $1 \to c_4$, $0 \to c_7$
f_1	$c_0 \to 1$, $c_1 \to 1$, $c_2 \to 0$, $c_5 \to 1$	$0 \to c_0$, $0 \to c_1$, $1 \to c_2$, $0 \to c_5$
f_2	$c_2 \to 0$, $c_5 \to 1$, $c_6 \to 0$, $c_7 \to 1$	$0 \to c_2$, $1 \to c_5$, $0 \to c_6$, $1 \to c_7$
f_3	$c_0 \to 1$, $c_3 \to 1$, $c_4 \to 0$, $c_6 \to 0$	$1 \to c_0$, $1 \to c_3$, $0 \to c_4$, $0 \to c_6$

The table denotes the message received by each c-node from its v-nodes and corresponding message sent by it to its v-nodes.

3.Now,each variable node has 3 options,the value that it currently holds and the two values that it received from its variable nodes.Here,a good option would be take the majority one.This will continue again and again in a loop.To stop the loop we will have some threshold on the amount of loops after which the variable nodes makes a hard decision and sends it to their check thereby completing the decoding process.

v-node	y_i	**Received Messages**	**Decision**
c_0	1	$f_1 \to 0,\ f_3 \to 1$	1
c_1	1	$f_0 \to 0,\ f_1 \to 0$	0
c_2	0	$f_1 \to 1,\ f_2 \to 0$	0
c_3	1	$f_0 \to 0,\ f_3 \to 1$	1
c_4	0	$f_0 \to 1,\ f_3 \to 0$	0
c_5	1	$f_1 \to 0,\ f_2 \to 1$	1
c_6	0	$f_2 \to 0,\ f_3 \to 0$	0
c_7	1	$f_0 \to 1,\ f_2 \to 1$	1

The table shows the messages received by each v-node from its c-node in the first loop and the corresponding decision made by it based on majority. In the example taken above,we can see that the codeword is corrected after the first loop itself.

6.6 Encoding

Encoding is fairly easy in LDPC codes.In it we choose certain variable nodes to place our bits on and calculate the values of the others by using the parity check equations corresponding to the parity check matrix.However,one should note that this will again become a quadratic complex algorithm.But in this case also,we can exploit the properties of sparse matrix to make computation easier.

6.7 Applications of LDPC codes

1.LDPC codes are used in optical communication as it enables high speed transmission over high distances.LDPC codes can be efficiently implemented using parallel processing techniques. The encoding and decoding operations of LDPC codes can be done parallely, allowing us to process multiple code symbols at a time. This parallelism enables faster encoding and decoding contributing to the high-speed capabilities of LDPC codes.

2.It is used in HDD(Hard Disk drives),SSD(Solid State Drives) and magnetic tapes to correct errors.

3.It is used in wired communication systems such as Ethernet,power-line communication.etc. Other applications include Next Gen Optical Disks,Digital Video Broadcasting.etc.

6.8 Summary

LDPC codes enable high speed data transfer.In the last decade or so LDPC codes have been significantly researched upon .Scientists have tried to learn about iterative decoding algorithms which is in fact used in another type of code called turbo codes which is used nowadays in satelite communication,optical communication,digital broadcasting.etc.The efficiency of LDPC codes is better than turbo codes but the main disadvantage is inorder to achieve that the block length needs to be very high and also the encoders are more complex.

7 Reed Solomon Codes

7.1 Introduction

Reed Solomon codes are error-correcting codes used in data storage and digital communication. They were named after Irving S. Reed and Gustave Solomon, who independently discovered them in the late 1960s. Reed Solomon codes are considered as one of the most beautiful error-correcting codes and are highly effective in correcting errors that occur in noisy communication channels and data storage.

7.2 Definition

We define the Reed Solomon code $RS_{F_q,S}[q,n,k]$ for integers $q \geq n > k \geq 1$, over a field F_q and a set $S = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq F_q$ as:

$$RS_{F_q,S}[q,n,k] = \{f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)\} \in \mathbf{F}^n$$

where $f \in \mathbf{F}[X]$ is a polynomial of degree $\leq k-1$.

In RS codes, to encode a message **m** having k symbols, we construct a polynomial with the symbols as coefficients and evaluate it at n different points of set S to obtain the codeword associated with **m**.

7.3 Properties of Reed Solomon Codes

1. $RS_{F_q,S}[q,n,k]$ is a linear code since adding two polynomials of a degree results in another polynomial possibly of the same degree.
2. The block length of the codeword in the RS codes defined above is n since we evaluate the polynomial at n different points of the set S.

3. The minimum distance of the code is $d = n - k + 1$, which implies that the mapping is injective.

4. The dimension of the code is k. It represents the number of symbols of the message that can be encoded in one codeword.

7.4 Distance of Reed Solomon Codes

Let us assume $\mathbf{m} = (m_0; m_1; \ldots; m_{k-1}) \neq 0$. Then $f[X]$ representing the codeword will have at most degree $k-1$ roots. A polynomial of degree d defined over F_q has at most d roots in F_q. So the polynomial $f[X]$ has at most $k-1$ roots. Therefore, $\{f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)\}$ has at most $k-1$ zeroes in it. Thus, a maximum of $k-1$ elements can be common for two different messages. Therefore, the minimum distance is $n-k+1$.

Note: Two different polynomials of same degree with positive elements can't evaluate to the same value other than zero for some α_i, as that implies that the coefficients of both polynomials are the same, which in turn implies both polynomials are the same, leading to a contradiction.

7.5 Choice of S

Taking $S = F_q$: We evaluate the polynomial at all elements of F_q, thereby providing the best tradeoff between block length n and field size q. This also maximizes the number of possible codewords and distance.
Taking $S = F \setminus \{0\}$: S contains all non-zero elements of F_q. This is used in applications where zero has some property that makes it unfit as an evaluation point.

7.6 Encoding Message

Consider the message as a $1 \times k$ matrix $\mathbf{m} = \begin{bmatrix} m_0 & m_1 & \cdots & m_{k-1} \end{bmatrix}_{1\times k}$.
Let $S = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$.
Then the generator matrix G can be represented by a $k \times n$ matrix:

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{bmatrix}_{k\times n}$$

Then the codeword $\mathbf{c}$ corresponding to $\mathbf{m}$ can be represented as a $1 \times n$ matrix, which is given by:

$$\mathbf{c} = \mathbf{m}_{1\times k} \cdot G_{k\times n} = \begin{bmatrix} m_1 & m_2 & \cdots & m_k \end{bmatrix}_{1\times k} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{bmatrix}_{k\times n}$$

$$= \begin{bmatrix} f(\alpha_1) & f(\alpha_2) & \dots & f(\alpha_n) \end{bmatrix}_{1\times n}$$

where f is a function $f : F \to F$ given by

$$f(X) = m_0 + m_1 X + \ldots + m_{k-1} X^{k-1} \in \mathbf{F}[X]$$

Characteristics of the evaluation points:

1. The evaluation points should be spread across F so that we can detect errors easily.
2. Each evaluation point should be different.No two evaluation point should be same
3. Degree: The evaluation points should be different powers of the primitive element of F, which is necessary for error correction. A primitive element is an element in the field from which we can generate all the other elements in the field.

7.7 Decoding Message

Suppose the received codeword is $\mathbf{y} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{n-1} \end{bmatrix}_{1\times n}$. We consider a polynomial with these n elements as coefficients, given by:

$$y(X) = y_0 + y_1 X + y_2 X^2 + \ldots + y_{n-1} X^{n-1} = \sum_{i=1}^{n} y_i X^{i-1}$$

Then we calculate this polynomial at $(n-k)$ evaluation points $\beta = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_{n-k} \end{bmatrix}_{1\times(n-k)}$ from the n evaluation points we used during encoding. We call these values the syndrome, represented by:

$$\mathsf{S} = \begin{bmatrix} S_1 & S_2 & \dots & S_{n-k} \end{bmatrix}_{1\times(n-k)}$$

where $S_i = y(\beta_i)$. This method is called syndrome calculation and is used for error detection and correction.

To calculate the syndrome matrix using matrices, we define the parity matrix $\mathbf{H}$ using β as follows:

$$\mathbf{H} = \begin{bmatrix} \beta_1^0 & \beta_1^1 & \cdots & \beta_1^{n-1} \\ \beta_2^0 & \beta_2^1 & \cdots & \beta_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{n-k}^0 & \beta_{n-k}^1 & \cdots & \beta_{n-k}^{n-1} \end{bmatrix}_{(n-k)\times n}$$

The matrix $\mathbf{H}$ is designed such that the null space of $\mathbf{H}^T$ is the set of all codewords in the source code. After syndrome calculation, we check if the syndrome matrix S is zero. If it is, then the received codeword is valid and we can directly extract the original information from it. If not, an error has occurred, and we need to correct it using error correcting algorithms.

To correct errors, we calculate the error locator polynomial $\sigma(X)$ using the Berlekamp-Massey algorithm or Euclidean Algorithm, which is quite complex. It is a polynomial such that $\sigma(\alpha_i) = 0$ if $f(\alpha_i) \neq y_i$, and it is zero otherwise. The roots of the error locator polynomial give the positions in the codeword where errors exist, and the number of roots gives us the number of errors.
Suppose there were l errors at positions given by $e = \{e_1, e_2, \ldots, e_l\}$.

Next, we calculate the error evaluator polynomial $\omega(X)$ given by

$$\omega(X) = \sigma'(X) \cdot S(X) - \sigma(X) \cdot S'(X)$$

where $\sigma'(X)$ refers to the derivative of $\sigma(X)$ and $S'(X)$ is the derivative of $\mathsf{S}(X)$.

Then we calculate the error correction factors for each element in the received codeword. The steps for that include:

1. Create an array f of size n.
2. Sort the array e in increasing order.
3. Initialize a count c to 0.
4. Iterate from $i = 0$ to $i = n - 1$:
(a)If i is not equal to e_c, put $f[i] = 0$.
(b) If $i = e_c$, calculate the error evaluator polynomial at e_c and let it be equal to v. Then, initialize $f[i] = v$ and increment the count c by 1.

To correct the error and calculate the original codeword sent, we calculate:

$$\mathsf{c} = \mathbf{y}_{1\times n} - \mathbf{f}_{1\times n} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{n-1} \end{bmatrix}_{1\times n} - \begin{bmatrix} f_0 & f_1 & \cdots & f_{n-1} \end{bmatrix}_{1\times n}$$

From this codeword, we can retrieve the original message by:

$$\mathbf{m}_{1\times k} = \mathbf{c}_{1\times n} \cdot \mathbf{G}^{-1}_{n\times k}$$

Now, we have corrected the error and retrieved the original message sent.

7.8 Applications and Summary

Reed Solomon Codes is one of the early codes introduced.But still it is used in many applications.It is used in storage devices like CDs,DVDs and hard drives.

One Problem with Reed Solomon codes is its large alphabet size.It achieves singleton bound. That is for a particular rate,it works optimally.Codes which achieve singleton bound are called Maximum distance separable codes.But in order to achieve this distance, a large alphabet size is required so that the conditions of evaluation points are satisfied.
Another issue is how to use these codes with digital communication systems.That can be easily achieved by assigning each element in F_q with a binary string.So,this reduces the encoding problem to that in field F_2.So,the resulting code reduces to a linear binary code.
Suppose we have $RS_{F_q,S}[q,n,k]$ with q as a power of 2.Here distance of the code is d=n-k+1.This will yield a binary code

$$[n\log_2(n), k\log_2(n), d']_2$$

Here d'$\geq d.LetN =$nlog$_2(n)$ and k=n-d+1.So we get a binary code

$$[N, n-(d-1)log_2(n), d]_2$$

But this binary code may not be optimal. BCH codes yield optimal results of

$$[N, n-(d-1)/2\log_e(n+1), d]_2$$

BCH codes has same distance as RS codes and offers better rate as well.
But still Reed Solomon codes are still popular.This is particularly in data storage devices.Here error occurs in bursts.In RS codes,these burst of errors that affect bits occur on an even smaller number of elements in the field.So,even if lot of bits get affected,the number of field elements affected is very less.

7.9 Conclusion

Throughout this paper,we discussed various concepts in coding theory, different coding techniques particularly linear codes like hamming code,LDPC codes,Reed Solomon codes.Though some of these are more efficient than others,each has its own significance.

Error correction techniques, such as error-correcting codes, rely on linear algebra to detect and correct errors that occur during data transmission or storage. Linear codes, which are defined by linear equations and matrices, enable the detection and correction of errors through the use of parity checks and syndromes.

For example,while discussing Reed Solomon codes,we said there are BCH codes which has same distance but offers same rate.But RS codes are still popular.Each code has its own advantages which has significance in a specific calculation.

Coding theory is really interesting.Analysing performance,complexity and other characteristics under different conditions are still going on.Possibly,even though some might disagree,we might create a code which is universal and can be used in all applications regardless of the conditions

8 References

1. Lin, Shu, and Daniel J. Costello. Error Control Coding: Fundamentals and Applications.Pearson-Prentice Hall, 2004.
2. Shannon, Claude E. “A Mathematical Theory of Communication.” The Bell System Technical Journal, 2009, https://doi.org/10.1109/9780470544242.ch1.
3. Justesen, Jørn, and Tom Høholdt. A Course in Error-Correcting Codes. European Mathematical Society, 2017.
4. Pray, Leslie A. “DNA Replication and Causes of Mutation.” Nature News, www.nature.com/scitable/top replication-and-causes-of-mutation-409/. Accessed 7 June 2023.
5. S. Kwon and D. -J. Shin, ”Blind Classification of Error-Correcting Codes for Enhancing Spectral Efficiency of Wireless Networks,” in IEEE Transactions on Broadcasting, vol. 67, no. 3, pp. 651-663, Sept. 2021, doi: 10.1109/TBC.2021.3061984.
6. Cover, T. M., and Joy A. Thomas. Elements of Information Theory. 2nd ed., Wiley-India, 2010.
7. Poole, David. Linear Algebra: A Modern Introduction. Langara College, 2021.
8. University of Michigan Lectures
9. Mano, M. Morris, and Michael D. Ciletti. Digital Design: With an Introduction to Verilog HDL. Pearson, 2013.
10.Hamming Codes:Error Reducing Techniques.,Rohit Kumar Upadhyay 11.Low Density Parity Check Codes (LDPC Codes),Soumya borwankar, Dhruv Shah