Coding Theory

Agrim Mittal (2022101040), Bipasha Garg (2022111006), Palle Sreeja (2022101081)

June 21, 2023

1 Introduction

In coding theory, linear codes are a type of error-correcting codes defined as subsets of a vector space over a finite field. A linear code, denoted as a $q-ary \quad [n,k]-code$, is a k-dimensional subspace of an n-dimensional vector space $\mathbb{V}$ over a Galois field $GF(q)$. The vectors in the code are n-dimensional and represent the code-words, while k represents the dimension or the number of information symbols in each code-word.

The vectors in a linear code can be represented as binary strings or sequences of symbols from the finite field $\mathbb{F}$. The code-words are obtained by applying a linear transformation or encoding function to the information symbols, often represented by a generator matrix. Linearity is a crucial property of linear codes. It means that if two vectors are in the code, any linear combination of those vectors will also be in the code.

The code is characterized by the following properties:

i. **Closure under vector addition**: The code is closed under the operation of vector addition. This means that if two code-words are in the code, their sum is also a code-word.

$$u + v \in \mathbb{C} \; \forall \, u, v \in \mathbb{C}$$

where $\mathbb{C}$ is a subset of the field $GF(q)$

ii. **Closure under scalar multiplication**: The code is closed under scalar multiplication, where scalar values are elements from the field F. For binary codes, scalar multiplication can only be by 0 or 1.

$$a \cdot u \in \mathbb{C} \; \forall \, u \in \mathbb{C} \;\; \& \;\; a \in \mathbb{F}$$

where $\mathbb{C}$ is a subset of the field $GF(q)$

Linear codes are particularly useful for error detection and correction. By designing the generator matrix or parity check matrix of a linear code, it is possible to detect and correct a certain number of errors or erasures that may occur during message transmission.

Decoding involves using linear algebraic techniques to determine the most likely original code-word based on the received code-word and the properties of the linear code.

The minimum distance of a linear code, denoted as d, is defined as the weight of the smallest, non-trivial code-word in the code. The weight of a code-word is the number of non-zero elements it contains. The minimum distance determines the code's error-detection and error-correction capabilities, as it represents the maximum number of errors that can be reliably detected and corrected.

The relationship between the minimum distance and code-word weight is established by observing that the minimum distance of any two binary code-words is equal to the weight of their sum. Therefore, the minimum distance of the code is equal to or greater than the weight of the smallest, non-trivial code-word.

2 Generator Matrices

Linear codes can be entirely defined by listing only the basis vectors of the code's subspace, as opposed to listing every single code-word. This generator matrix can be obtained from the matrix of the entire code by the use of row reduction. A binary $[n, 2^k]-code$ will reduce to a generator matrix of dimensions $k \times n$.

If $\mathrm{E} : B^m \longrightarrow B^n$ be an encoding function, then the matrix of the form

$$\mathrm{G} = [I_{m\times m} : A_{m\times(n-m)}]_{m\times n}$$

where G is called 'Generator Matrix' and $I_{m\times m}$ represents 'Identity Matrix'.

The entire code can then be re-generated by expressing all linear combinations of the rows of the generator matrix.

Given a linear code $\mathcal{C}$, a generator matrix G of $\mathcal{C}$ is a matrix whose rows generate all the elements of $\mathcal{C}$, i.e., if $\mathrm{G} = (g_1 g_2 \dots g_k)^T$, then every code-word w of $\mathcal{C}$ can be represented as

$$w = c_1 g_1 + c_2 g_2 + \dots + c_k g_k = c\mathrm{G}$$

in a unique way, where $c = (c_1 c_2 \dots c_k)$

NOTE:- 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.

2.1 Implementation

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 $2^k = 8 \longrightarrow k = 3$ and reduces to 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 the rows of the generator matrix:

$$\begin{bmatrix} x & y & z \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = x \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} + y \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} + z \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$$

As an example, letting $x = 1, y = 1$ and $z = 0$ generates the second code-word in the original code-word list as follows:

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

Repeating this process for all possible combinations of values 0 and 1 for x, y and z will eventually result in the generation of the entire linear code.

2.2 Equivalence of Linear Codes

Equivalent codes share the same minimum distance, code-word length and number of code-words, but contain at least some different code-words. By definition, two linear codes are said to be equivalent if it is possible to obtain the generator matrix for one through the following permutations:

i. Swapping the order of rows

ii. Multiplying any row by a non-zero scalar (in the binary case this is generally discarded)

iii. Addition of a scalar from one row to another

iv. Swapping the order of columns

v. Multiplying any column by a non-zero scalar (in the binary case this is generally discarded)

If only the first three operations are used (the row operations), then the resultant matrix is simply another basis of the same subspace and it will generate the exact same code. If the last two operations (the column operations) are used then the resultant matrix will generate an equivalent, but potentially different, code.

2.3 Example

Take the following binary $[7, 4] - code$, in random order, which can be reduced to its generator matrix and further permutated into standard form:

$$\begin{bmatrix}1101001\\1100110\\1011010\\1010101\\0111100\\0110011\\0001111\\0000000\end{bmatrix} \begin{matrix}Replace \rightarrow with\\ r_2 \rightarrow r_1 + r_2\\ r_4 \rightarrow (r_5 + r_6) - (r_1 + r_2)\\ r_6 \rightarrow (r_3 + r_4) - (r_1 + r_2)\\ r_7 \rightarrow r_7 - (r_1 + r_2)\end{matrix} \begin{bmatrix}1101001\\0001111\\1011010\\0000000\\0111100\\0000000\\0000000\\0000000\end{bmatrix}$$

After one more permutation the generator matrix (basis) for the code (vector subspace) remains:

$$\begin{bmatrix}1101001\\0001111\\1011010\\0111100\end{bmatrix} \begin{matrix}Replace \rightarrow with\\ r_3 \rightarrow r_1 + r_2 + r_3\end{matrix} \begin{bmatrix}1101001\\0001111\\0000000\\0111100\end{bmatrix}$$

Using column operations the resultant matrix can be put into standard form. This standard form will produce an equivalent, but potentially different, code from the original:

$$\begin{bmatrix}1101001\\0001111\\0111100\end{bmatrix} \begin{matrix}Swap\\ c_2 \longleftrightarrow c_6\end{matrix} \begin{bmatrix}1001011\\0101101\\0011110\end{bmatrix} is\, standard\, form\, [I_3|A]\, where\, A = \begin{bmatrix}1011\\1101\\1110\end{bmatrix}$$

The resultant generator matrices produce two different, but equivalent, [7, 4, 4]-codes:

$$\begin{bmatrix}1101001\\0001111\\0111100\end{bmatrix} \xrightarrow{generates} \begin{bmatrix}1101001\\1100110\\1011010\\1010101\\0111100\\0110011\\0001111\\0000000\end{bmatrix} while \begin{bmatrix}1001011\\0101101\\0011110\end{bmatrix} \xrightarrow{generates} \begin{bmatrix}1111000\\1100110\\1010101\\0110011\\1001011\\0101101\\0011110\\0000000\end{bmatrix}$$

Code-words themselves are encoded messages. Part of the code-word is the message digits, and the additional portion of the code-word is redundancy to allow for error detection and correction.

Let $v_1v_2v_3 \ldots v_k$ be the message vectors to be encoded. Multiplying these with the generator matrix G on the right produces the code-word. The rows of the generator matrix G are represented as $r_1r_2r_3 \ldots r_k$. Symbolically: $v\mathrm{G} = \sum_{i=1}^{k} v_i r_i$

Encoding message vector $v = 111$ using the example generator matrix from the earlier example:

$$\begin{bmatrix}111\end{bmatrix} \times \begin{bmatrix}1101001\\0001111\\0111100\end{bmatrix} = \begin{bmatrix}1011010\end{bmatrix}$$

If the generator matrix is in standard form the resulting code-word can be further analyzed. Encoding message vector $v = 111$ using the example standard form generator matrix from the earlier example:

$$\begin{bmatrix}111\end{bmatrix} \times \begin{bmatrix}1001011\\0101101\\0011110\end{bmatrix} = \begin{bmatrix}1111000\end{bmatrix}$$

Because this code-word is generated from a matrix in standard form, the first k^{th} digits of the code-word will match the message vector. The remainder of the code-word is redundancy bits. Dissecting the code-word generated in the previous example shows:

$$1111000 = \left[\begin{array}{c|c} 111 & 1000 \\ Message\,Vector & Redundancy\,Bits \end{array}\right]$$

3 Coset Array Decoding

Coset array decoding is a decoding algorithm used in coding theory to correct errors in a received code-word. It is particularly useful for decoding linear block codes. The algorithm works by constructing cosets of the code and finding the coset with the minimum weight, which corresponds to the most likely transmitted code-word.

3.1 Method

Let C_1 be the code consisting of the following binary words of length 5:

$$\begin{bmatrix}00000\\00111\\01001\\01110\\10011\\10100\\11010\\11101\end{bmatrix}$$

- **LISTING ELEMENTS OF COSET**: When creating cosets, it is helpful to leave the information positions unchanged: only change the numbers in parity-check positions.
- So we can create every coset by adding every word that has nonzero elements only in parity-check positions to the code C. The number of unique cosets that are created is equal to the number of words that have nonzero elements only in parity-check positions.

$$\begin{aligned} C_1 &= C_1 + (00000) \\ C_2 &= C_1 + (00001) \\ C_3 &= C_1 + (00010) \\ C_4 &= C_1 + (00011) \end{aligned}$$

- **FINDING COSET LEADER**: There may be more than one word of minimum weight in a coset; choose one of them as the coset leader.For finding the coset leader, for the $n_t h$ coset $C_n = C_1 + a_n$, we find e_n by adding a_n to each element of C_n and then take the coset leader to be the result with the fewest non-zero elements.
- **DECODING**: The procedure used to decode a word x, we first examine the coset $C + x$ to find the coset leader e, and then add e to x to produce the decoded word $e + x$.

11100:

$$C + 11100 = \begin{cases} 11100 \\ 11011 \\ 10101 \\ 10010 \\ 01111 \\ 01000 \\ 00110 \\ 00001 \end{cases} \qquad 00001 = e \ (one\, coset\, leader)$$

Therefore, the decoded word is $e + x = 11101$

3.2 Algorithm

1. Given a received code-word, calculate all possible cosets of the code. Each coset is obtained by adding all possible error patterns to the received code-word.
2. For each coset, calculate the weight of the coset, which represents the number of errors in the received code-word.
3. Select the coset with the minimum weight as the most likely transmitted codeword.
4. If the minimum weight is zero, no errors are detected, and the received codeword is the correct codeword. Otherwise, correct the errors by subtracting the most likely transmitted codeword from the received codeword.
5. The resulting corrected codeword is the decoded message.

4 Parity-Check Matrix

Parity-check matrices are an essential concept in coding theory, specifically in the field of linear codes. They play a crucial role in the detection and correction of errors that occur during the transmission of encoded messages. Parity-check matrices are used to determine whether an encoded message contains any errors by performing mathematical operations on the received code-word.

4.1 Definition

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

For example $C = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ then $C^{\perp} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$. The generator matrix for $C^{\perp}$ is a parity check matrix for C itself. The parity check matrix H can be obtained from G, the generator matrix of C, as follows:

$$G = \left[\begin{array}{c|cccc} I_k & a_{12} & \dots & a_{1n} & a_{1,n+1} \\ & a_{22} & \dots & a_{2n} & a_{2,n+1} \\ & \vdots & \vdots & \ddots & \vdots \\ & a_{n2} & \dots & a_{nn} & a_{n,n+1} \end{array}\right]$$

then

$$\left[\begin{array}{cccc|c} -a_{1,1} & -a_{2,1} & \dots & -a_{k,1} & \\ -a_{1,2} & -a_{2,2} & \dots & -a_{k,2} & I_{n-k} \\ \vdots & \vdots & \ddots & \vdots & \\ -a_{1,n-k} & -a_{2,n-k} & \dots & -a_{k,n-k} & \end{array}\right]$$

In the binary case the negative signs have no effect since $-1 = 1$. Using the standard form matrix from the previous example:

$$G = \begin{bmatrix} 1001011 \\ 0101101 \\ 0011110 \end{bmatrix}, \textit{then } A = \begin{bmatrix} 1011 \\ 1101 \\ 1110 \end{bmatrix}$$

thus

$$H = [-A^T | I_4] = \begin{bmatrix} 1111000 \\ 0110100 \\ 1010010 \\ 1100001 \end{bmatrix}$$

Using the parity-check matrix it is possible to describe the entire code C as follows:

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

If we have a linear code C with a generator matrix G, the dual code $C^\perp$ can be obtained by finding a parity-check matrix H for C. The dual code $C^\perp$ has a generator matrix that is the transpose of the parity-check matrix H.

The rows of a parity check matrix are the coefficients of the parity check equations. That is, they show how linear combinations of certain digits (components) of each code-word equal zero. For example, the parity check matrix

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

compactly represents parity check equations,

$$\begin{aligned} c_3 + c_4 &= 0 \\ c_1 + c_2 &= 0 \end{aligned}$$

that must be satisfied for the vectors (c_1, c_2, c_3, c_4) to be a code-word of C.

The parity-check matrix, denoted as H, is an important matrix used in linear coding theory. It is constructed such that any code-word in the linear code is orthogonal to the rows of the parity-check matrix. In other words, the dot product of any code-word and any row of the parity-check matrix is zero. This property allows the parity-check matrix to detect errors in the received code-word.

4.2 Error Detection

To illustrate this further, let's assume we have a linear code with a generator matrix G and a parity-check matrix H. If we want to encode a message m into a code-word c, we can compute $c = m \cdot G$, where $\cdot$ represents matrix multiplication. This process involves multiplying the message vector by the generator matrix to obtain the code-word.

During transmission, if noise or errors occur, the received code-word r may differ from the transmitted code-word c. To check for errors, we can perform a simple operation using the parity-check matrix:

$$r \cdot H^T = 0$$

If the dot product of the received code-word and the transpose of the parity-check matrix is zero, it indicates that the received code-word is likely error-free. If the dot product is nonzero, it indicates the presence of errors.

4.3 Error Correction

If the dot product is nonzero, we can use the syndrome to identify the specific error pattern. The syndrome is computed as

$$s = r \cdot H^T$$

and it provides information about the error location in the received code-word. By analyzing the syndrome, we can determine which bits in the received code-word are in error and correct them accordingly.

Let's consider a linear code with a parity-check matrix H. Each row of H represents an equation or constraint that the code-words must satisfy. The number of rows in H is usually equal to the number of check equations needed for error detection.

It's important to note that parity-check matrices are typically designed to detect errors, but not correct them. For error correction, other techniques such as syndrome decoding or maximum likelihood decoding algorithms are employed in combination with the parity-check matrix.

4.4 Code Generation by Parity Checks

Let $H = [P|I_r]$ be a canonical parity-check matrix, where I_r is an $r \times r$ identity matrix, and P an arbitrary $r \times (n-r)$.

Let $k = n - r$ and let

$$H = \left[\begin{array}{cccc|cccc} h_1 1 & h_1 2 & \dots & h_1 k & 1 & 0 & \dots & 0 \\ h_2 1 & h_2 2 & \dots & h_2 k & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_r 1 & h_r 2 & \dots & h_r k & 0 & 0 & \dots & 1 \\ & & P & & & I_r & & \end{array}\right]$$

4.5 Encoding Procedure

Given a k-tuple message $x = (x_1, x_2, \dots, x_k)$, we need to compute the corresponding n-tuple code word (frame = message + error code) $y = (y_1, y_2, \dots, y_k, y_{k+1}, \dots, y_n)$, where $k = n - r$, i.e., $n = k + r$.

Set $y_i \leftarrow x_i \; \forall 1 \leq i \leq k$

Compute $y_k + i$ for $1 \leq i \leq r$ as the modulo-2 sum:

$$y_1 h_{11} \oplus y_2 h_{12} \oplus \dots$$

$$\oplus y_k h_{1k} \oplus y_{k+1} h_{1,k+1} = 0, \; since \, h_{1,k+1} = 1$$

$$\implies y_{k+1} = y_1 h_{11} \oplus y_2 h_{12} \oplus \dots \oplus y_k h_{1k}$$

Similarly,

$$y_{k+2} = y_1 h_{21} \oplus y_2 h_{22} \oplus \dots \oplus y_k h_{2k}$$

In general,

$$y_{k+i} = \bigoplus_{j=1}^{k} y_j h_{i,j}$$

For example, given a 4×9 parity-check matrix H as follows:

$$\begin{bmatrix} 110101000 \\ 100110100 \\ 011010010 \\ 001100001 \end{bmatrix}$$

To encode the message tuple (1 1 0 1 0), we can do the following: Here the message tuple is (11010) = $(x_1, x_2, x_3, x_4, x_5)$. H is of the form $[P|I_4]$, where P is an 4×5 matrix and I_4 is the identity matrix.

Let the encoded message tuple be $y = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9)$. Set

$$\begin{aligned} y_1 &= x_1 = 1 \\ y_2 &= x_2 = 1 \\ y_3 &= x_3 = 0 \\ y_4 &= x_4 = 1 \\ y_5 &= x_5 = 0 \end{aligned}$$

The parity-check equations are given by

$$\begin{aligned} y_1 \oplus y_2 \oplus y_4 \oplus y_6 = 0 &\implies y_6 = 1 \\ y_1 \oplus y_4 \oplus y_5 \oplus y_7 = 0 &\implies y_7 = 0 \\ y_2 \oplus y_3 \oplus y_5 \oplus y_8 = 0 &\implies y_8 = 1 \\ y_3 \oplus y_4 \oplus y_9 = 0 &\implies y_9 = 1 \end{aligned}$$

Hence, the encoded message is (1 1 0 1 0 1 0 1 1).

5 Syndrome Decoding

Syndrome decoding is a highly efficient method of decoding a linear code over a noisy channel, i.e. one on which errors are made. In essence, syndrome decoding is minimum distance decoding using a reduced lookup table. This is allowed by the linearity of the code.

The parity-check matrix is especially useful for generating the syndrome of a given coset. The syndrome a vector x length n is defined as $S(x) = xH^T$. Any two vectors in the same coset will have the same syndrome.

Proof:

Let x and y be vectors in a given coset $s.t.$ $x + C = y + C$ (from the definition of coset). It follows that $x - y \in C$. From the definition of Parity Check Matrix $(x - y)H^T = 0$ which means that $xH^T = yH^T$. Thus $S(x) = S(y)$.

5.1 General Method

The key idea is to take advantage of the linearity of the code. Consider the (7 , 4) Hamming code whose generator matrix G is given by $\left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$. From G, we can write out the parity equations as follows:

$$P_1 = D_1 + D_2 + D_4$$
$$P_2 = D_1 + D_3 + D_4$$
$$P_3 = D_2 + D_3 + D_4$$

Because the arithmetic is over $\mathbb{F}_2$, we can rewrite these equations by moving the $P's$ to the same side as the $D's$ (in modulo-2 arithmetic, there is no difference between + and -):

$$D_1 + D_2 + D_4 + P_1 = 0$$
$$D_1 + D_3 + D_4 + P_2 = 0$$
$$D_2 + D_3 + D_4 + P_3 = 0$$

There are $n - k$ such equations. One can express these equations, in matrix notation using a parity check matrix, H, as follows:

$$H \cdot [D_1 D_2 \dots D_k P_1 P_2 \dots P_{n-k}]^T = 0$$

H is the horizontal stacking, or concatenation, of two matrices: A^T, where A is the sub- matrix of the generator matrix of the code ($G = I_{k\times k}|A$) and $I_{(n-k)\times(n-k)}$, the identity matrix. I.e.,

$$H = A^T | I_{(n-k)\times(n-k)}$$

H has the property that for any valid code-word c (which we represent as a $1n$ matrix),

$$H \cdot c^T = 0$$

Hence, for any received word r without errors, $H \cdot r^T = 0$.

Now suppose a received word r has some errors in it. r may be written as c + e, where c is some valid code-word and e is an error vector, represented (like c) as a 1 × n matrix. For such an r,

$$H \cdot r^T = H \cdot (c + e)^T = 0 + H \cdot e^T$$

If r has at most one bit error, then e is made up of all zeroes and at most one "1". In this case, there are n + 1 possible values of $H \cdot e^T$; n of these correspond to exactly one bit error, and one of these is a no-error case (e = 0), for which $H \cdot e^T = 0$. These n + 1 possible vectors are precisely the syndromes.

Syndrome decoding pre-computes the syndrome corresponding to each error. Assume that the code is in systematic form, so each code-word is of the form $D_1 D_2 \dots D_k P_1 P_2 \dots P_{n-k}$. In general, if element i of e is 1 and the other elements are 0, the resulting syndrome $H \cdot e^T$ corresponds to the case when bit i in the code-word is wrong. Under the assumption that there is at most one bit error, we care about storing the syndromes when one of the first k elements of e is 1.

If $H \cdot r^T$ is not all zeroes, and if it does not match any stored syndrome, then the decoder concludes that either some parity bit was wrong, or that there were multiple errors. In this case, it might simply return the first k bits of the received word as the message.

5.2 Correcting Multiple Errors

Suppose we wish to correct all patterns of $\leq t$ errors. In this case, we need to pre-compute more syndromes, corresponding to $0, 1, 2, \ldots t$ bit errors. Each of these should be stored by the decoder. There will be a total of

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots \binom{n}{t}$$

syndromes to pre-compute and store. If one of these syndromes matches, the decoder knows exactly which bit error pattern produced the syndrome, and it flips those bits and returns the first k bits of the code-word as the decoded message.

5.3 Example

Consider the (7,4) Hamming code. The G for this linear block code is $\left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$ Given any $k = 4$-bit message m, the encoder produces an $n = 7 - bit$ code-word, c by multiplying $m \cdot G$. (m is a $1 \times k$ matrix, G is a $k \times n$ matrix and c is a $1 \times n$ matrix)

The parity check matrix, H, for this code is:

$$\left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

Suppose c is sent over the channel and is received by the decoder as r. For concreteness, suppose $c = 1010001$ and $r = 1110001$ (error in the second bit).

The decoder pre-computes syndromes corresponding to all possible single-bit errors. (It actually needs to pre-compute only k of them, each corresponding to an error in one of the first k bit positions of a code-word). In our case, the $k = 4$ syndromes of interest are:

$$\begin{aligned} H \cdot [1000000]^T &= [110]^T \\ H \cdot [0100000]^T &= [101]^T \\ H \cdot [0010000]^T &= [011]^T \\ H \cdot [0001000]^T &= [111]^T \end{aligned}$$

For completeness, the syndromes for a single-bit error in one of the parity bits are, not surprisingly:

$$\begin{aligned} H \cdot [0000100]^T &= [100]^T \\ H \cdot [0000010]^T &= [010]^T \\ H \cdot [0000001]^T &= [001]^T \end{aligned}$$

The decoder implements the following steps to correct single-bit errors:

i. Compute $c^{`} = H \cdot r^T$. In this example, $c^{`} = [101]^T$.

ii. If $c^{`}$ is 0, then return the first $k$ bits of $r$ as the message. In this example $c^{`}$ is is not 0.

iii. If $c^{`}$ is not 0, then compare $c^{`}$ with the n pre-computed syndromes, $H \cdot e_i$, where $e_i = [00 \ldots 1 \ldots 0]$ is a $1 \times n$ matrix with 1 in position i and 0 everywhere else.

iv. f there is a match in the previous step for error vector e_l, then bit position l in the received word is in error. Flip that bit and return the first k elements of r.

In this example, the syndrome for $H \cdot [0100000]^T = [101]^T$, which matches $c^{`} = H \cdot r_T$. Hence the decoder flips the second bit in the received word and returns the first $k = 4$ bits of r. The returned estimate of the message is [1010].

v. If there is no match, return the first k bits of r. If the bit error probability is small, then it is a very good approximation.

6 Perfect Codes

6.1 Definition

A perfect code is a code in which every possible word of a specified length is a valid code-word, and the minimum distance between any two code-words is maximized. This property ensures that no valid message is misinterpreted as an error, and the codes can correct errors up to a certain threshold. Achieving perfect code status is a fundamental goal in coding theory.

Let C be an error-correcting code consisting of N code-words, in which each code-word consists of n letters taken from alphabet A of length q, and every two distinct code-words differ in atleast $d = 2e+1$ places. Then C is said to be perfect if for every possible word w_0 a unique code word w in C in which at most e letters of w differ from the corresponding letters of w_0.

It is straightforward to show that C is perfect if:

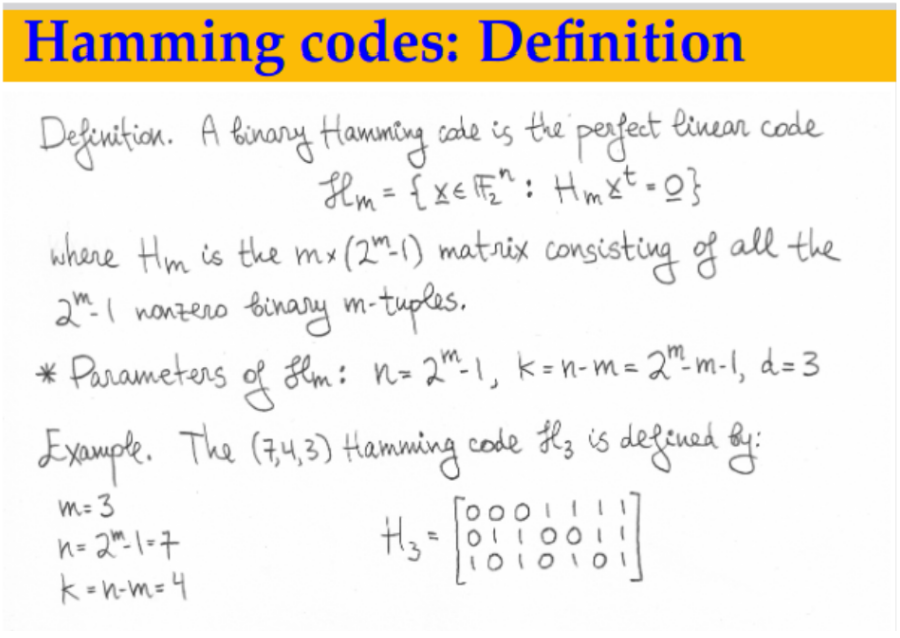
$$\sum_{i=0}^{e} \binom{n}{i} (q-1)^i = \frac{q^n}{N}$$

If C is a binary linear code, then $q = 2$ and $N = 2^k$, where k is the number of generators of C, in which case C is perfect if

$$\sum_{i=0}^{e} \binom{n}{i} = \frac{q^n}{N}$$

6.2 Hamming Codes

Hamming codes are a widely known example of perfect codes. These binary linear codes are designed to correct single-bit errors. For any prime number q, Hamming codes achieve perfect code status. The construction of Hamming codes involves using parity-check and generator matrices based on binary



representations. The minimum distance of a Hamming code determines the number of errors it can correct. For example, the (7, 4) Hamming code has a generator matrix G and a minimum distance of 3, meaning it can correct up to two errors.

A binary Hamming Code H_r of length $n = 2^r - 1$(with $r \geq 2$) is a linear code with parity-check matrix H whose columns consist of all nonzero binary vectors of length r, each used once. H_r is an $(n = 2^r - 1, k = 2^r = 1 - r, d = 3)$ code. Hamming codes are perfect single error-correcting codes.

6.3 Golay Codes

Golay codes, discovered by Marcel Golay in 1949, are another set of perfect codes. They are highly efficient in terms of error correction and have found applications in various fields. The Golay codes come in different variants, including binary and ternary codes. The first Golay code is a binary $[23, 12, 7] - code$. Using the definition of a perfect code described earlier it is easy to see:

$$2^{12}\left\{1 + 23 + \binom{23}{2} + \binom{23}{3}\right\} = 2^{23}$$

The second code is a ternary $[11, 6, 5] - code$. These codes have generator and parity-check matrices that determine their properties. The minimum distance of the Golay codes indicates their error-correction capabilities.

6.4 Reed Solomon Codes

Reed-Solomon codes are another family of perfect codes widely used in various applications, such as data storage systems and digital communication. These codes are defined over finite fields and are capable of correcting multiple errors. Reed-Solomon codes achieve perfect code status by maximizing the minimum distance based on their parameters, such as code length and symbol size. For example, a (15, 11) Reed-Solomon code over a finite field of size 2^4 can correct up to two errors. The mathematical aspects of Reed-Solomon codes involve concepts from algebra and finite field theory.

6.5 Mathematical Aspects of Perfect Codes

The study of perfect codes involves various mathematical concepts and techniques. Coding theory provides the mathematical foundation for analyzing and designing perfect codes. Linear algebra is used to study the properties of generator and parity-check matrices. Finite fields play a crucial role in defining codes over non-binary alphabets. Combinatorics is employed to analyze the code parameters and optimize the code construction. The mathematical aspects of perfect codes allow researchers to understand their performance characteristics and devise efficient error-correction strategies.

6.6 Applications of Perfect Codes

Perfect codes find extensive applications in the transmission of digital data. Due to their ability to achieve maximum error correction, they provide a high level of reliability in critical communication systems. Perfect codes are used in various domains, including satellite communication, data storage systems, and telecommunications. Their efficiency in terms of error correction and reliable data transmission make them indispensable in applications where accuracy and integrity are paramount.

7 Hamming Codes

7.1 Basic Idea

In the process of transmission the information gets corrupted, therefore, we get a signal at the receiver which is different from the original information signal. The general idea for achieving error detection and correction is to add some extra data with the message, which receivers can use to check consistency of the delivered message, and to recover data determined to be corrupted . Hamming code can only correct single bit errors.

7.2 Working

7.2.1 Error Detection

Let's say the receiver has received a signal which has even parity bits (total no.s of 1 is 8). Now allocate the sequence number of each bit as 1 2 3 ...

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	0	0	0	1	1	0	0	1

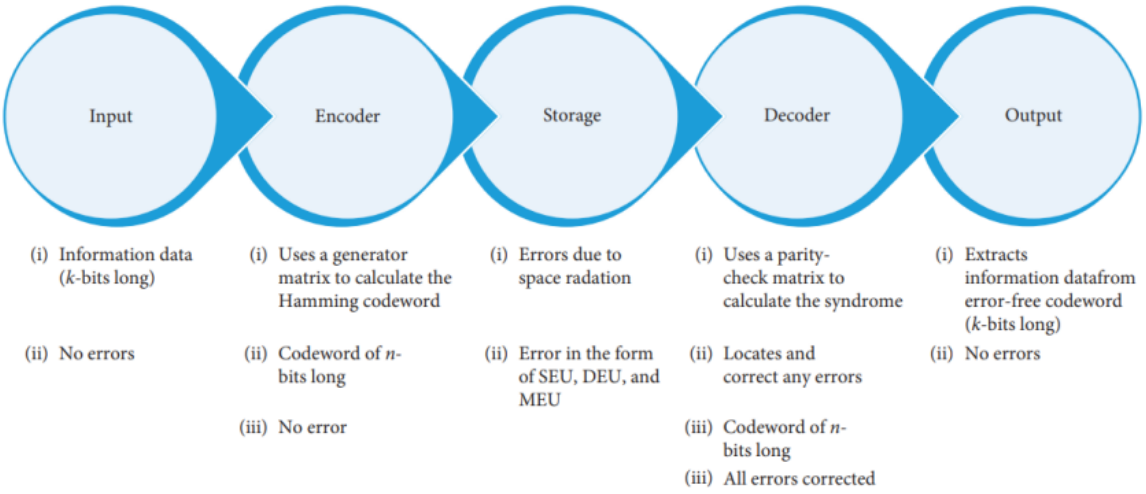


Figure 1: General Layout of Hamming Code

Check the Parity Bits: In this process we identified the parity bits. It will be a power of 2's (1, 2, 4, 8 all are powers of 2).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	0	0	0	1	1	0	0	1

Find the value of each Parity bit (i.e. 1, 2, 4, and 8):

VALUE OF PARITY BIT 1: BY CHECK 1 AND SKIP 1										
1#	?	0	0	1	0	1	0	1	=1	No. of 1's are odd
VALUE OF PARITY BIT 2: BY CHECK 2 AND SKIP 2										
2#	?	0	1	1	0	1	0	1	=0	No. of 1's are even
VALUE OF PARITY BIT 4: BY CHECK 4 AND SKIP 4										
4#	?	0	1	1	1	0	0	1	=0	No. of 1's are even
VALUE OF PARITY BIT 8: BY CHECK 8 AND SKIP 8										
8#	?	0	0	1	1	0	0	1	=1	No. of 1's are odd

Match the value of parity bits with the value of received signals:

PARITY BIT	ACTUAL VALUE	DERIVED VALUE	CORRECT PARITY NO.	INCORRECT PARITY NO.
1	1	1	1	-
2	1	0	-	2
4	1	0	-	4
8	0	1	-	8

Now we find that the value of parity bit 1 is matched with the value of received signal and the value of parity bits 2, 4, 8 are not matched with the value of received signal. Therefore, it is decided that the parity bit 1 is correct and parity bits 2, 4, 8 are incorrect.

Find out the incorrect bit: For finding the incorrect bit we add all the incorrect parity bits i.e. $2 + 4 + 8 = 14$.

The result 14 means the value of 14 bit is corrupted.

7.2.2 Error Correction

The value of 14 bit will be changed according to following rules:

- 0 will be changed to 1
- 1 will be changed to 0

The value of 14 bit is 0 as per the following:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	0	0	0	1	1	0	0	1

Value of 14 bit is converted as 1 instead of 0:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	0	0	0	1	1	0	1	1
P	P		P				P							

Now again repeat the above process:

VALUE OF PARITY BIT 1: BY CHECK 1 AND SKIP 1										
1#	?	0	0	1	0	1	0	1	=1	No. of 1's are odd
VALUE OF PARITY BIT 2: BY CHECK 2 AND SKIP 2										
2#	?	0	1	1	0	1	1	1	=1	No. of 1's are odd
VALUE OF PARITY BIT 4: BY CHECK 4 AND SKIP 4										
4#	?	0	1	1	1	0	1	1	=1	No. of 1's are odd
VALUE OF PARITY BIT 8: BY CHECK 8 AND SKIP 8										
8#	?	0	0	1	1	0	1	1	=0	No. of 1's are even

Match the value of parity bits with the value of received signals:

PARITY BIT	ACTUAL VALUE	DERIVED VALUE	CORRECT PARITY NO.	INCORRECT PARITY NO.
1	1	1	1	-
2	1	1	2	-
4	1	1	4	-
8	0	0	8	-

Now we find that the value of all parity bits 1, 2, 4, 8 are matched with the value of received signal. Therefore, the error of the signal has been removed. In this way we can detect and correct the error of received signals.

7.3 Hamming Code Matrix

Hamming codes are linear codes with parity-check matrices of dimension $r \times 2^r - 1$ s.t. the columns of H are exactly the unique, non-zero vectors of length r. Hamming Codes are denoted as $Ham(r, q)$ with q prime.Binary Hamming codes are denoted similarly as $Ham(r, 2)$. The length of the code-words in a binary Hamming code will be $n = 2^r - 1$. Listing the columns of the parity-check in a different order results in a simple permutation of the rows and columns of the generator matrix which will generate an equivalent Hamming code (as shown in an earlier example). Hamming codes always achieve their sphere-packing bounds making every Hamming code also a perfect code.

From this definition it is easy to construct the parity-check matrix representing $Ham(3, 2)$ by simply From this definition it is easy to construct the parity-check matrix representing listing every possible column vector of length 3:

$$H = \begin{bmatrix} 1011100 \\ 1101010 \\ 0111001 \end{bmatrix}, thus\, G = \begin{bmatrix} 1000110 \\ 0100011 \\ 0010101 \\ 0001111 \end{bmatrix} \; and\, n = 2^3 - 1 = 7$$

Another example is a $Ham(4,2)$ code:

$$H = \begin{bmatrix} 100010011011111 \\ 010011000111011 \\ 001001110010111 \\ 000100101101101 \end{bmatrix}, \; thus\, G = \left[\begin{array}{c|c} & 1100 \\ & 0110 \\ & 0011 \\ & 1010 \\ & 1001 \\ \ldots I_{11} \ldots & 0101 \\ & 1110 \\ & 1101 \\ & 1101 \\ & 1011 \\ & 0111 \\ & 1111 \end{array}\right] \; and\, n = 2^4 - 1 = 15$$

In this previous example only the four columns with exactly one entry of 1 are linearly independent, which implies that $d = 3$ and this code can detect up to one error in each code-word.

7.4 Conclusion

- We can detect the error on received sequence of information by checking the value of parity bits.
- It is found that the derived values of parity bits are not matched with the value of received signal parity bits. It shows that the signal has some error and if the values are matched it means a signal has no error.
- After detecting the error we can find which bit has an error by adding the sequence numbers of incorrect parity bits, the result shows the sequence number of incorrect bits. After finding the sequence number of incorrect bits we can easily correct the error.

8 Reed-Solomon Codes

Reed-Solomon codes are a subclass of non-binary BCH codes on $GF(q^r)$, where $r = 1$.

$$\implies GF(q^r) = GF(q)$$

suppose we have to assure to correct up to t errors, then we have to choose $2t$ consecutive powers of the primitive element α in $GF(q)$. We can build $GF(q)$ as an extension of a Galois Field $GF(p)$, s.t. $GF(q) = GF(p^m)$, with p prime. Once we have a fixed p, evaluating $GF(q) = GF(p^m) = GF(p)[x]/p(x)$, with $p(x)$ a primitive polynomial of degree m. In this way we can consider elements of $GF(q)$ as polynomials and, by choosing $p(x)$ a primitive polynomial in $GF(p)$ of degree m, we are sure that $\alpha = x$ is the primitive polynomial in $GF(q)$. For sake of computational simplicity it is usual to take $p = 2$.

Once we have chosen $2t$ consecutive powers of the primitive element, i.e. powers of $x \in GF(2^m)$, we have to compute all their minimal polynomials. A minimal polynomial of a on $GF(q)$ is, by definition, the minimum degree monic polynomial with coefficients in $GF(q)$, s.t. $a \in GF(q^m)$ is a zero of it. As $m = 1$, the required polynomial is trivially $x - a$.

$\therefore$ for each consecutive power of α, the minimum polynomial will be $x - \alpha^i$ and the generator polynomial can be easily evaluated by multiplying all of these elementary factors:

$$\prod_{i=1}^{2t} (x - \alpha^{\xi_0 + i})$$

where ξ_0 is an arbitrary offset.

Reed-Solomon Codes are denoted as $RS(n,k)$ with s symbol bits. Each code-word consists of k data symbols of s bits each for a total code-word length of n. If $2t = n - k$, then the code-word is capable of correcting t incorrect symbols.

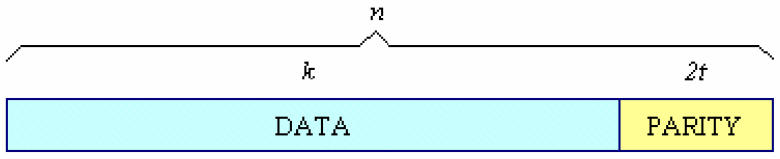


Figure 2: Reed-Solomon Code-word Diagram

These codes are minimum distance codes, i.e. $RS(n,k)$ has got the smallest possible Hamming Distance: $d_{min}(RS(n,k)) = n-k+1$. This is easy to see because:

$$d_{min}(RS(n,k)) \geq 2t+1$$

$$d_{min}(RS(n,k)) \geq 2t+1 = n-k+1 \qquad \because 2t = n-k$$

By the properties of Hamming Code Distance,

$$d_{min}(RS(n.k)) \leq n-k+1$$

Putting together the two inequalities:

$$d_{min}(RS(n,k)) = n-k+1$$

which is the smallest possible.

8.1 $RS(255,223)$ code

One of the most commonly used instances of Reed-Solomon codes is $RS(255,223)$. This code maps 223 length words in 255 length code words.

- $k = 223$
- $n = 255$

Since in Reed-Solomon codes $n = q-1$, where q is the numerosity of the employed Galois Field, whose elements represent the code symbols, we deduce that the finite field we need is $GF(q) = GF(256)$. Choosing the finite field as an extension of $GF(2)$, we get $GF(256) = GF(2^8)$:

- $m = 8$

m is the degree of the prime polynomial $p(x)$ which characterizes $GF(p^m) = GF(2^8) = GF(2)[x]/p(x)$. Every element of $GF(256)$ is uniquely associated to a polynomial with coefficients in $GF(2)$ with smaller degree than 8 and it can be, thus, represented as a binary vector of length 8. As primitive polynomial we will use $p(x) = 1 + x^2 + x^3 + x^4 + x^8$.

- $t = 16$

t is the greatest number of symbol errors $RS(255,223)$ can correct:

$$t = \left\lfloor \frac{d_{min}}{2} \right\rfloor = \left\lfloor \frac{n-k+1}{2} \right\rfloor = \left\lfloor \frac{255-223+1}{2} \right\rfloor = \left\lfloor \frac{33}{2} \right\rfloor = \lfloor 16.5 \rfloor = 16$$

Since the code corrects symbols, which are made up by 8 bits each, the capability of Reed-Solomon codes to correct errors is much higher than it can seem: if we use a binary transmission on the channel, this can corrupt up to $16 \times 8 = 128$ bits, but, unless more than 16 symbols are involved, the word can still be properly corrected.

8.2 Applications in Real Life

Reed-Solomon Codes are used to protect data from defects in the storage or transmission media. The Reed-Solomon code is crucial to the success of the audio CD, allowing CD players to interpolate data lost due to scratches or dust on the disks. The code allows the CD player to correct over 2mm of damaged or missing disk surface. This means that almost any "skips" the user hears when listening to a CD are almost certainly related to tracking errors by the laser itself.

Cellular telephones employ these codes to overcome interference from high- strength radio transmitters. Digital televisions, high-speed DSL modems, the Space Communications Protocol and common bar code scanners also rely on these powerful codes.

Protocols used in computer networks almost universally apply redundancy to data transmissions to protect against errors. One of the most common networking protocols used in Windows networks is Transmission Control Protocol or TCP.

9 Network Communication

9.1 Overview and Purpose

In coding theory, network communication refers to the transmission of data over a network using error-correcting codes. Error-correcting codes are utilized to detect and correct errors that may occur during data transmission, ensuring reliable and accurate communication.

9.2 How?

There are various types of error-correcting codes used in network communication, such as Hamming codes, Reed-Solomon codes, Turbo codes, and LDPC (Low-Density Parity-Check) codes. These codes differ in their error detection and correction capabilities, complexity, and performance characteristics, allowing for efficient and robust network communication in different scenarios.

9.3 In Brief

The process of network communication using coding theory involves encoding the data at the source, transmitting the code-word over the channel, and decoding the code-word at the destination to recover the original data. The use of error-correcting codes helps ensure that the transmitted data is reliable and accurate, even in the presence of errors introduced by the channel.

9.3.1 Methodology

The methodology for network communication using error-correcting codes follows these steps:

- **Code Selection**: Choose an appropriate error-correcting code based on the requirements of the network communication system. Consider factors such as the expected error rate, available bandwidth, computational complexity, and desired level of error correction.
- **Encoding**: The source data is encoded using the selected error-correcting code. The encoding process adds redundant bits to the original data, forming a code-word. The redundant bits allow for error detection and correction during transmission.
- **Modulation and Channel Coding**: The encoded data is modulated to convert it into a form suitable for transmission over the channel. This step involves mapping the code-word to physical signals that can be transmitted through the chosen communication medium. Channel coding may also be applied at this stage to further enhance error correction capabilities, especially in the presence of specific channel impairments.
- **Transmission**: The modulated and channel-coded data is transmitted over the network channel. This can involve sending the data over wired or wireless connections, fiber optics, or other communication mediums.

- **Reception**: At the receiving end, the transmitted data is received from the channel. This may involve amplification, filtering, and other signal processing techniques to improve the quality of the received signal.
- **Decoding**: The received data undergoes decoding, where the error-correcting code is used to detect and correct any errors that occurred during transmission. The decoder analyzes the received code-word and attempts to recover the original data. Various decoding algorithms exist depending on the specific error-correcting code used.
- **Error Detection and Correction**: The decoding process identifies errors in the received code-word. If errors are detected, appropriate error correction techniques are applied to recover the original data. This can involve error correction algorithms that use the redundant bits to determine the most likely original data, such as syndrome decoding, maximum likelihood decoding, or iterative decoding.
- **Extraction**: The corrected codeword is extracted, and the original data is retrieved from it. The extracted data is then made available for further processing or display by the destination device or application.

The methodology for network communication using error-correcting codes is iterative, with data being encoded, transmitted, received, and decoded in a continuous cycle to ensure reliable and accurate communication. The specific techniques and algorithms employed depend on the chosen error-correcting code and the characteristics of the communication system.

9.4 Components

Network communication in coding theory typically involves the following components:

- **Source**: The source generates the data to be transmitted. This can be any device or application that produces data, such as a computer, sensor, or user input.
- **Encoder**: The encoder takes the data from the source and applies an error-correcting code to generate a codeword. The codeword contains additional redundant bits that help detect and correct errors.
- **Channel**: The channel represents the communication medium over which the codeword is transmitted. This can include wired or wireless networks, optical fibers, or any other means of data transmission.
- **Decoder**: The decoder receives the transmitted codeword from the channel and performs error detection and correction. It uses the error-correcting code to identify and correct any errors that may have occurred during transmission.
- **Destination**: The destination receives the decoded codeword from the decoder and extracts the original data. It can be a device or application that needs the transmitted information for further processing or display.

References

- Coding Theory: AS-404, Dr. Khusboo Verma, Ass. Prof, FoET, Lucknow University
- A. K. Singh, "Error detection and correction by hamming code," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, India, 2016, pp. 35-37, doi: 10.1109/ICGTSPICC.2016.7955265.
- Reed-Solomon Codes, Department of Information Engineering, University of Padova
- Introduction to algebraic coding, University of California San Diego
- Hillier, Caleb Balyan, V.. (2019). Error Detection and Correction On-Board Nanosatellites Using Hamming Codes. Journal of Electrical and Computer Engineering. 2019. 1-15. 10.1155/2019/3905094