Cyclic Codes for Error Detection

Sunaina Pati, Chennai Mathematical Institute, Chennai

April 5, 2025

Abstract

This talk is based on Peterson and Brown's seminal paper *"Cyclic Codes for Error Detection."* I will introduce cyclic codes, a class of linear codes with the defining property that any cyclic shift of a codeword is also a codeword. The talk will cover their algebraic structure, key properties, and how these features enable efficient encoding and error detection. I will also explore BCH codes, their decoding capabilities, and how cyclic codes can be designed to detect single, double, triple, and burst errors. Examples and theorems from the paper will be discussed in detail to highlight both theoretical insights and practical applications.

Contents

1 Introduction

Suppose two parties, Alice and Bob, wish to communicate over a channel that may be unreliable—that is, parts of the message may get corrupted or altered during transmission. To ensure robust communication, we want the recipient to be able to detect such corruption, and ideally, recover the original message even in the presence of errors.

We assume that the channel allows the transmission of strings over a fixed finite alphabet. This alphabet could be a finite field (e.g., $\mathbb{F}_q$), binary symbols (bits), or even characters from the English alphabet.

This brings us to two central goals in the design of such communication systems:

- **Error detection**: Identifying whether the received message has been corrupted.
- **Error correction**: Reconstructing the original message despite some level of corruption.

In many practical scenarios, the message that Alice wishes to send may be written in one alphabet—say, English—while the communication channel supports a different alphabet—such as binary strings. In such cases, we need a systematic way to convert messages from the source alphabet to the channel alphabet. This is typically done by encoding the message into *blocks* of symbols over the channel alphabet.

For instance, when converting English text to binary, we might use ASCII encoding. Each character is represented as a fixed-length binary string, i.e., a block of bits. These binary blocks are then transmitted over the channel.

Naturally, not every possible block of bits corresponds to a meaningful message. In general, only a subset of all possible strings is used for encoding valid messages. This leads us to the formal notion of a *block code*.

Definition. Let Σ be the fixed finite alphabet used by the communication channel. A *block code* $\mathcal{C}$ of length n over Σ is a subset of Σ^n:

$$\mathcal{C} \subseteq \Sigma^n.$$

Elements of $\mathcal{C}$ are called *codewords*.

Definition. For any two strings x and y of the same length, the *Hamming distance* between them is defined as the number of positions at which the corresponding symbols differ. That is,

$$d(x, y) = |\{i \mid x_i \neq y_i\}|.$$

Turns out Hamming distance is actually a distance!

Theorem 1.1. *The Hamming distance defines a metric on* Σ^n.

Proof. Let $x, y, z \in \Sigma^n$. We verify the four properties of a metric:

1. **Non-negativity:** By definition, $d(x, y)$ counts the number of differing coordinates between x and y. Since this is a count, $d(x, y) \geq 0$.
2. **Identity of indiscernibles:** If $x = y$, then all coordinates agree, so $d(x, y) = 0$. Conversely, if $d(x, y) = 0$, then there are no positions i with $x_i \neq y_i$, so $x = y$.
3. **Symmetry:** The number of positions where $x_i \neq y_i$ is the same as the number where $y_i \neq x_i$, so
$$d(x, y) = d(y, x).$$

4. **Triangle inequality:** Let us consider each coordinate $i \in \{1, 2, \ldots, n\}$. At each position, define the indicator function:

$$\delta(a, b) = \begin{cases} 1 & \text{if } a \neq b, \\ 0 & \text{if } a = b. \end{cases}$$

Then,

$$d(x, z) = \sum_{i=1}^{n} \delta(x_i, z_i).$$

For each i, observe that

$$\delta(x_i, z_i) \leq \delta(x_i, y_i) + \delta(y_i, z_i),$$

because if $x_i = z_i$, the left-hand side is 0 and the inequality holds; and if $x_i \neq z_i$, then either $x_i \neq y_i$ or $y_i \neq z_i$ (or both), so the right-hand side is at least 1. Summing over all i, we get

$$d(x, z) \leq d(x, y) + d(y, z).$$

Hence, all the properties of a metric are satisfied. □

Definition. The *distance* of a code $\mathcal{C}$ is the minimum Hamming distance between any two distinct codewords in $\mathcal{C}$. Formally,

$$d(\mathcal{C}) = \min_{\substack{x,y \in \mathcal{C} \\ x \neq y}} d(x, y).$$

In a sense, the distance of a code quantifies how many changes are required to transform one valid message (codeword) into another. For example, if a code has distance $d = 5$, this means that there exist two distinct codewords x and y that differ in exactly 5 positions.

Now, suppose Alice sends the codeword x, but due to noise in the channel, exactly those 5 positions are altered. Then Bob would receive y instead of x, and since y is also a valid codeword, he would completely misinterpret the message.

This highlights a key idea: *to minimize the chance of such confusion, we want codes to have large distance*. A larger distance makes it less likely that noise will transform one valid codeword into another. This leads to an important observation.

Claim 1.1. *Let d be the distance of a code $\mathcal{C}$. Then the code is $d-1$-detectable, or if the channel corrupts atmost $d-1$ letters of the message, then the other party can detect that the code word has been corrupted.*

Proof. By definition, the minimum distance d of a code C is the smallest Hamming distance between any two distinct codewords in C. This means that to transform one codeword into another, at least d symbol changes are required.

Suppose a codeword $c \in C$ is transmitted, and at most $d-1$ symbol errors occur during transmission. Let r be the received word. Then the Hamming distance between c and r is at most $d-1$.

However, since any other codeword in C is at distance at least d from c, the received word r cannot be a different valid codeword. Therefore, either $r = c$, or $r \notin C$.

Thus, if $r \in C$, no errors occurred; otherwise, if errors occurred, $r \notin C$, and the receiver can detect that an error happened.

Hence, the code can detect up to $d-1$ errors. □

Lemma 1.1. *If the communication channel corrupts at most t symbols during transmission, then any code with minimum distance at least $t+1$ can* detect *such errors. That is, Bob can recognize when the received message has been corrupted.*

Theorem 1.2 (Error Correction Bound)**.** *Let $C \subseteq \Sigma^n$ be a code with minimum Hamming distance d. Then C can correct up to t errors if and only if $d \geq 2t + 1$.*

Proof. Assume a codeword $x \in C$ is transmitted, and during transmission, at most t symbols are corrupted, resulting in a received word $r \in \Sigma^n$. That is,

$$d(x, r) \leq t.$$

To decode correctly, the receiver must identify the original codeword x from r. A standard decoding strategy is to choose the codeword $y \in C$ that minimizes the Hamming distance $d(r, y)$. For this strategy to work reliably, we want x to be the *unique* codeword within distance t of r.

(Only if direction) Suppose C can correct up to t errors. Then for any two distinct codewords $x, y \in C$, the decoding balls of radius t around them must be disjoint:

$$B_t(x) \cap B_t(y) = \emptyset.$$

Otherwise, if $r \in B_t(x) \cap B_t(y)$, then both $d(x, r) \leq t$ and $d(y, r) \leq t$, so r could be decoded ambiguously.

Now, by the triangle inequality for the Hamming distance:

$$d(x, y) \leq d(x, r) + d(r, y) \leq t + t = 2t,$$

which would contradict the assumption that the minimum distance of the code is d. Therefore, to avoid such ambiguity, we must have:

$$d(x, y) > 2t \quad \text{for all } x \neq y \in C.$$

Hence, $d \geq 2t + 1$.

(If direction) Now suppose the minimum distance $d \geq 2t + 1$. Let $x \in C$ be the transmitted codeword and $r \in \Sigma^n$ the received word with $d(x, r) \leq t$. We claim that x is the unique codeword within distance t of r.

Suppose for contradiction that there exists another codeword $y \in C$, $y \neq x$, such that $d(y, r) \leq t$. Then:

$$d(x, y) \leq d(x, r) + d(r, y) \leq t + t = 2t,$$

which contradicts the assumption that $d(x, y) \geq d \geq 2t + 1$. Therefore, such a y cannot exist.

Hence, for every received word within distance t of a codeword x, that codeword is uniquely closest, and the code can correct up to t errors.

□

Suppose Alice sends a codeword x, and during transmission, it gets corrupted and Bob receives a word y. Given that at most t symbols were altered in the channel, we want to show that Bob can still correctly recover the original message.

Since Bob knows that at most t errors could have occurred, he considers all codewords within Hamming distance t of the received word y.[1] Clearly, the original codeword x lies within this ball, since $d(x, y) \leq t$.

Now, if x is the *only* codeword within this ball, then Bob can unambiguously conclude that Alice must have sent x.

[1] This is often visualized as placing a Hamming ball of radius t around y.

We claim that this is indeed the case, provided the code has minimum distance at least $2t + 1$. Suppose, for contradiction, there exists another codeword $x' \neq x$ such that $d(x', y) \leq t$. Then by the triangle inequality:

$$d(x, x') \leq d(x, y) + d(y, x') \leq t + t = 2t,$$

which contradicts the assumption that the minimum distance between any two codewords is at least $2t + 1$.

In other words, to move from one valid codeword to another through corruption, at least $2t + 1$ symbol changes are necessary. So if only t symbols are corrupted, the received word is guaranteed to be closer (in Hamming distance) to the original codeword than to any other.

However, in practice, it may not be efficient to check *all* codewords within distance t of y to perform decoding. In fact, even determining whether a given string is a codeword may itself be computationally difficult.

This highlights two crucial properties we want from our codes in addition to large minimum distance:

- **Efficient codeword detection**: There should be an efficient way to verify whether a given string is a valid codeword.
- **Efficient decoding**: Given a corrupted word (close to some codeword), we should be able to efficiently recover the original codeword.

1.1 Example: Repetition Code

Let us consider a simple code over the binary alphabet $\Sigma = \{0, 1\}$. Define the *repetition code* of length $n = 3$ as:

$$\mathcal{C} = \{000, 111\}$$

This code encodes the bit 0 as 000 and 1 as 111. It is a $(3, 1)$ block code: each message bit is encoded as a 3-bit codeword.

1.1.1 Distance of the Code

The Hamming distance between the two codewords is:

$$d(000, 111) = 3$$

Hence, the minimum distance of the code is $d = 3$. Therefore:

- It can detect up to $d - 1 = 2$ errors.
- It can correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor = 1$ error.

1.1.2 Error Correction Example

Suppose Alice wants to send the bit 1, so she transmits the codeword 111.

Case 1: No error. Bob receives 111, which is a valid codeword, and decodes it as 1.

Case 2: One-bit error. Suppose the second bit is flipped: $111 \to 101$. Bob receives 101. Now he checks distances:

$$d(101, 000) = 2, \quad d(101, 111) = 1$$

Bob decodes it as $111 \Rightarrow 1$. **Correctly decoded.**

Case 3: Two-bit error. Suppose $111 \to 001$. Now:

$$d(001, 000) = 1, \quad d(001, 111) = 2$$

Bob decodes it as $000 \Rightarrow 0$. **Incorrect decoding.**

However, since $001 \notin \mathcal{C}$, Bob can at least detect that an error has occurred.

2 Linear Codes

Recall that a block code $\mathcal{C}$ is an arbitrary subset of Σ^n. Such codes may have no algebraic structure, making it computationally hard to verify whether a given string is a codeword. This motivates the study of *linear codes*, which impose additional structure that allows for efficient encoding and decoding.

To define linear codes formally, we assume that the alphabet Σ is a finite field $\mathbb{F}_q$. Then the ambient space $\mathbb{F}_q^n$ is a vector space of dimension n over $\mathbb{F}_q$. A linear code is simply a subspace of this vector space.

Definition. A linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is called an $[n, k, d]_q$ linear code if:

- $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$,
- The minimum (Hamming) distance between any two distinct codewords in $\mathcal{C}$ is d.

Equivalently, $\mathcal{C}$ is closed under linear combinations: if $x, y \in \mathcal{C}$ and $\alpha, \beta \in \mathbb{F}_q$, then $\alpha x + \beta y \in \mathcal{C}$.

2.1 Interpretation of Parameters

An $[n, k, d]_q$ code encodes k symbols (message length) as n symbols (codeword length), thereby introducing redundancy to allow for error correction. The parameters have the following interpretation:

- k: Number of message symbols.
- n: Length of the codeword.
- d: Minimum distance between any two codewords — the code can detect up to $d - 1$ errors and correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.

In practice, we aim to maximize k (rate) and d (error protection), but these are in tension with each other.

2.2 Example: Repetition Code

Let $\mathbb{F}_2 = \{0, 1\}$ be the binary field, and consider the code

$$\mathcal{C} = \{(0, 0, 0), (1, 1, 1)\}.$$

This is a $[3, 1, 3]_2$ linear code. It is a 1-dimensional subspace of $\mathbb{F}_2^3$ (generated by the vector $(1, 1, 1)$), and the minimum distance between the two codewords is 3.

2.3 Weight of a Vector

Definition. The *Hamming weight* of a vector $x \in \mathbb{F}_q^n$ is defined as the number of nonzero entries in x:

$$\mathrm{wt}(x) = |\{i \in \{1, 2, \ldots, n\} \mid x_i \neq 0\}|.$$

In the case of binary codes ($\mathbb{F}_2$), this simply counts the number of 1s in x. In linear codes, the minimum distance of the code is equal to the minimum weight among all nonzero codewords:

$$d = \min\{\mathrm{wt}(x) \mid x \in \mathcal{C},\ x \neq 0\}.$$

This equivalence holds because, in a linear code, the difference of any two codewords is also a codeword.

2.4 Minimum Distance and Weight

Claim 2.1. *If $\mathcal{C}$ is a linear code, then*

$$d(\mathcal{C}) = \min_{x \in \mathcal{C} \setminus \{0\}} \mathrm{wt}(x).$$

Proof. By definition, the minimum distance of $\mathcal{C}$ is

$$d(\mathcal{C}) = \min_{\substack{x,y \in \mathcal{C} \\ x \neq y}} d(x, y).$$

Since $\mathcal{C}$ is linear, $x - y \in \mathcal{C}$ for all $x, y \in \mathcal{C}$. Also, $d(x, y) = \mathrm{wt}(x - y)$. So the above becomes

$$d(\mathcal{C}) = \min_{z \in \mathcal{C} \setminus \{0\}} \mathrm{wt}(z).$$

□

2.5 Membership Detection Using Parity-Check Matrix

Suppose we are given a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ of dimension k, and we want to test whether a given vector $x \in \mathbb{F}_q^n$ is a codeword (i.e., belongs to $\mathcal{C}$). Since $\mathcal{C}$ is a subspace, it has a basis $\{b_1, b_2, \ldots, b_k\}$, where each $b_i \in \mathbb{F}_q^n$.

Our goal is to construct a matrix H such that

$$Hx = 0 \quad \text{if and only if} \quad x \in \mathcal{C}.$$

That is, H defines a linear map whose kernel is exactly $\mathcal{C}$. This matrix H is called the *parity-check matrix* of the code.

2.5.1 Constructing the Matrix H

We first extend the basis $\{b_1, \ldots, b_k\}$ of $\mathcal{C}$ to a full basis $\{b_1, \ldots, b_k, b_{k+1}, \ldots, b_n\}$ of $\mathbb{F}_q^n$. Then every vector $v \in \mathbb{F}_q^n$ can be uniquely written as:

$$v = \alpha_1 b_1 + \cdots + \alpha_k b_k + \alpha_{k+1} b_{k+1} + \cdots + \alpha_n b_n.$$

Define a linear map H that:

- maps $b_1, \ldots, b_k$ to 0,
- and maps $b_{k+1}, \ldots, b_n$ to themselves (i.e., acts as identity on them).

Then:

$$Hv = \alpha_{k+1} b_{k+1} + \cdots + \alpha_n b_n.$$

Clearly, $Hv = 0$ if and only if $\alpha_{k+1} = \cdots = \alpha_n = 0$, which happens if and only if $v \in \mathcal{C}$.

2.5.2 Computing the Matrix Form of H

Let us now compute the matrix representation of H. Suppose the extended basis $\{b_1, \ldots, b_n\}$ is such that it aligns with the standard basis $\{e_1, \ldots, e_n\}$ of $\mathbb{F}_q^n$. Then the transformation H sends:

$$(\alpha_1, \ldots, \alpha_n) \mapsto (0, \ldots, 0, \alpha_{k+1}, \ldots, \alpha_n),$$

and its matrix is simply:

$$\hat{H} = \begin{bmatrix} \mathbf{0}_{(n-k)\times k} & I_{(n-k)\times(n-k)} \end{bmatrix},$$

where the zero block is of size $(n-k) \times k$ and I is the identity matrix of size $(n-k) \times (n-k)$.

But in general, our basis $\{b_1, \ldots, b_n\}$ won't be the standard basis. So we define a matrix $B \in \mathbb{F}_q^{n \times n}$ whose columns are the basis vectors:

$$B = \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_n \\ | & | & & | \end{bmatrix}.$$

Then B sends $e_i \mapsto b_i$, so its inverse B^{-1} sends $b_i \mapsto e_i$.

Now consider the transformation $\hat{H}B^{-1}$. It acts as follows:

$$\hat{H}B^{-1} b_i = \hat{H} e_i = \begin{cases} 0 & \text{if } i \le k, \\ e_i & \text{if } i > k. \end{cases}$$

Therefore, $H := \hat{H}B^{-1}$ is the matrix we are looking for — it satisfies $Hx = 0$ if and only if $x \in \mathcal{C}$.

2.5.3 Conclusion

The matrix $H = \hat{H}B^{-1}$ is the **parity-check matrix** of the code. It defines a linear transformation whose kernel is exactly $\mathcal{C}$. Given any vector $x \in \mathbb{F}_q^n$, we can efficiently test whether $x \in \mathcal{C}$ by simply checking whether $Hx = 0$.

2.6 Generator Matrix

A linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ of dimension k can be specified by a basis $\{g_1, g_2, \ldots, g_k\}$. Arranging these basis vectors as rows of a $k \times n$ matrix G gives the *generator matrix* of the code.

Definition. A generator matrix G of an $[n, k, d]_q$ code is a $k \times n$ matrix over $\mathbb{F}_q$ such that

$$\mathcal{C} = \{mG \mid m \in \mathbb{F}_q^k\}.$$

Each message $m \in \mathbb{F}_q^k$ is encoded as a codeword $c = mG$. Encoding is thus a simple matrix multiplication.

2.7 Example: $[7, 4, 3]$ Hamming Code

The $[7, 4, 3]$ Hamming code over $\mathbb{F}_2$ is a classic example of a linear code that encodes 4-bit messages into 7-bit codewords and can correct any single-bit error.

A generator matrix G for this code is:

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

A message $m = (m_1, m_2, m_3, m_4)$ is encoded as $c = mG$.

The parity-check matrix H satisfies $GH^\top = 0$ and is given by:

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

Each column of H is a nonzero 3-bit binary vector — in fact, the set of all nonzero vectors in $\mathbb{F}_2^3$. This ensures the minimum distance is 3, so the code can correct 1 error.

3 Properties of Hamming Codes

Definition (Binary Hamming Code)**.** For an integer $r \geq 2$, the *binary Hamming code* $\mathcal{H}_r$ is a linear code over $\mathbb{F}_2$ with parameters

$$[n = 2^r - 1,\ k = 2^r - 1 - r,\ d = 3],$$

constructed using a parity-check matrix $H \in \mathbb{F}_2^{r \times n}$ whose columns are all non-zero binary vectors of length r, each appearing exactly once.

Theorem 3.1 (Minimum Distance)**.** *The minimum distance d of the Hamming code $\mathcal{H}_r$ is 3.*

Proof. Let $H \in \mathbb{F}_2^{r \times n}$ be the parity-check matrix of $\mathcal{H}_r$. Since all columns of H are non-zero and distinct, any two columns are linearly independent.

Suppose $c \in \mathcal{H}_r \setminus \{0\}$. Then c is a non-zero codeword, and $Hc^\top = 0$. The weight $w(c)$ of c is the number of positions where c is non-zero.

We show that there are no codewords of weight 1 or 2:

- Weight 1: Suppose c has weight 1, then $c = e_i$ for some i, so $Hc^\top = He_i^\top = h_i \neq 0$, contradiction.
- Weight 2: Suppose $c = e_i + e_j$, then $Hc^\top = h_i + h_j \neq 0$ because $h_i \neq h_j$, so again $c \notin \mathcal{H}_r$.

But any three columns of H may sum to zero. So there exists at least one codeword of weight 3. Thus, the minimum weight and hence the minimum distance is 3. □

Theorem 3.2 (Error Correction Capability)**.** *The Hamming code $\mathcal{H}_r$ can correct all single-bit errors.*

Proof. Since the minimum distance is $d = 3$, the code can correct up to $t = \left\lfloor \frac{d-1}{2} \right\rfloor = 1$ error. □

Theorem 3.3 (Perfectness of Hamming Codes)**.** *The binary Hamming code $\mathcal{H}_r$ is a perfect 1-error-correcting code. That is, the set of Hamming balls of radius 1 centered at codewords of $\mathcal{H}_r$ covers the entire space $\mathbb{F}_2^n$.*

Proof. The number of vectors in $\mathbb{F}_2^n$ is 2^n. Each Hamming ball of radius 1 has:

$$\sum_{i=0}^{1} \binom{n}{i} = \binom{n}{0} + \binom{n}{1} = 1 + n = 1 + (2^r - 1) = 2^r$$

vectors. The number of codewords is $2^k = 2^{n-r}$. So the total number of vectors covered by all Hamming balls of radius 1 centered at codewords is:

$$2^{n-r} \cdot 2^r = 2^n.$$

Hence, the union of these balls covers all of $\mathbb{F}_2^n$, and the code is perfect. □

Remark. No redundancy is wasted in a perfect code. The Hamming codes are the only nontrivial perfect single-error-correcting binary linear codes.

4 Cyclic Codes

Definition. A *cyclic code* $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a linear code with the property that for every codeword $(c_0, c_1, \ldots, c_{n-1}) \in \mathcal{C}$, the cyclic shift

$$(c_{n-1}, c_0, c_1, \ldots, c_{n-2})$$

is also in $\mathcal{C}$.

That is, the space of codewords is invariant under cyclic shifts. So, if one codeword belongs to the code, then all of its cyclic shifts also belong to the code.

We can identify each vector $(c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ with the polynomial

$$c(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} \in \mathbb{F}_q[x]/(x^n - 1).$$

Under this identification, cyclic shifts correspond to multiplication by x modulo $x^n - 1$. That is, a cyclic code corresponds to an ideal in the ring $\mathbb{F}_q[x]/(x^n - 1)$.

This allows efficient encoding and decoding algorithms via polynomial arithmetic. Thank you Field theory!

4.1 Codewords as Polynomials

Given a vector $(c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$, we associate to it a polynomial

$$c(X) = c_0 + c_1 X + c_2 X^2 + \cdots + c_{n-1} X^{n-1}.$$

This allows us to interpret the vector space $\mathbb{F}_q^n$ as the additive group of the ring $\mathbb{F}_q[X]/(X^n - 1)$, since both are n-dimensional over $\mathbb{F}_q$ and isomorphic as additive groups.

What makes this ring view more powerful is the additional multiplicative structure. Suppose $c = (c_0, \ldots, c_{n-1})$ is a codeword. Then a right cyclic shift gives us the vector

$$c' = (c_{n-1}, c_0, \ldots, c_{n-2}).$$

As a polynomial, $c(X) = c_0 + c_1X + \cdots + c_{n-1}X^{n-1}$, and the shifted version becomes

$$c'(X) = X \cdot c(X) \mod (X^n - 1).$$

Indeed, multiplying by X shifts each term to a higher degree, and the X^n term is identified with 1 in the quotient ring $\mathbb{F}_q[X]/(X^n - 1)$.

Remark. Cyclic shifts correspond to multiplication by powers of X modulo $X^n - 1$. Thus, the ring $\mathbb{F}_q[X]/(X^n - 1)$ provides a natural algebraic framework for studying cyclic codes.

We now state the fundmental result of cyclic code!

Theorem 4.1. *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear code. Then $\mathcal{C}$ is a cyclic code if and only if $\mathcal{C}$ corresponds to an ideal in the quotient ring $R = \mathbb{F}_q[X]/(X^n - 1)$.*

Proof. We interpret codewords in $\mathbb{F}_q^n$ as polynomials in $\mathbb{F}_q[X]$ of degree less than n, by associating a vector

$$(c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n \quad \text{with the polynomial} \quad c(X) = c_0 + c_1X + \cdots + c_{n-1}X^{n-1}.$$

All operations are considered modulo $X^n - 1$, so the set of all such polynomials forms the ring $R = \mathbb{F}_q[X]/(X^n - 1)$.

($\Rightarrow$) Suppose $\mathcal{C}$ is a cyclic code. Then by definition, $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a linear subspace that is closed under cyclic shifts.

Since $\mathcal{C}$ is closed under multiplication by X, and by linearity, it is closed under multiplication by all polynomials in $\mathbb{F}_q[X]$ modulo $X^n - 1$, it follows that for any $a(X) \in R$, we have:

$$a(X)c(X) \in \mathcal{C}.$$

Thus, $\mathcal{C}$ is closed under multiplication by arbitrary elements of R, and is therefore an ideal in R.

($\Leftarrow$) Conversely, suppose $\mathcal{C} \subseteq R$ is an ideal. Then, by the definition of an ideal, for any $c(X) \in \mathcal{C}$ and any $a(X) \in R$, the product $a(X)c(X) \in \mathcal{C}$. Since $X \in R$, multiplying by powers of X corresponds to performing cyclic shifts modulo $X^n - 1$. Thus, $X^i c(X) \in \mathcal{C}$ for all $i \geq 0$, showing that $\mathcal{C}$ is closed under cyclic shifts.

Hence, $\mathcal{C}$ is a cyclic code. □

Now let us study the structure of ideals in $R = \mathbb{F}_q[X]/(X^n - 1)$. Since $X^n - 1$ is a monic polynomial of degree n, and $\mathbb{F}_q[X]$ is a principal ideal domain (PID), all ideals in R are principal. Suppose we factor:

$$X^n - 1 = g_1(X)g_2(X) \cdots g_k(X)$$

into irreducible polynomials over $\mathbb{F}_q$. Then every ideal of R is of the form $\langle g(X) \rangle$ where $g(X)$ divides $X^n - 1$.

Corollary 4.1. *Every cyclic code $\mathcal{C} \subseteq \mathbb{F}_q^n$ corresponds to a principal ideal $\langle g(X) \rangle \subseteq \mathbb{F}_q[X]/(X^n - 1)$, where $g(X)$ divides $X^n - 1$.*

The polynomial $g(X)$ is called the *generator polynomial* of the cyclic code.

4.2 Dimension of a Cyclic Code

Theorem 4.2. *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a cyclic code, and let $g(X) \in \mathbb{F}_q[X]$ be its generator polynomial. Suppose* $\deg g(X) = d$. *Then:*

1. *The number of codewords in $\mathcal{C}$ is $|\mathcal{C}| = q^{n-d}$.*
2. *The dimension of the code is* $\dim \mathcal{C} = n - \deg g(X)$.

Proof. Since $\mathcal{C}$ is a cyclic code of length n, we can identify codewords with polynomials in the quotient ring

$$R = \mathbb{F}_q[X]/(X^n - 1).$$

Under this identification, $\mathcal{C} \subseteq R$ corresponds to an ideal in the ring R. Every ideal of R is principal, and therefore generated by a unique monic divisor $g(X)$ of $X^n - 1$. That is,

$$\mathcal{C} = \langle g(X) \rangle = \{g(X)h(X) \mid h(X) \in \mathbb{F}_q[X], \deg(h) < n - d\}.$$

This is because multiplication in R is modulo $X^n - 1$, so we only consider polynomials of degree less than n, and

$$\deg(g(X)h(X)) < d + (n - d) = n.$$

Thus, each codeword is uniquely determined by a polynomial $h(X) \in \mathbb{F}_q[X]$ of degree less than $n - d$, and the total number of such polynomials is:

$$q^{n-d},$$

since each coefficient in $h(X) = h_0 + h_1 X + \cdots + h_{n-d-1}X^{n-d-1}$ can be chosen freely from $\mathbb{F}_q$.

Hence, the number of codewords is $|\mathcal{C}| = q^{n-d}$, and since $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$, its dimension is

$$\dim \mathcal{C} = \log_q |\mathcal{C}| = n - d = n - \deg(g(X)).$$

□

4.2.1 Parity Check Matrix

The parity check operation becomes especially elegant for cyclic codes. Since $\mathcal{C} = (g(X))$, a word $c(X) \in R$ is in the code if and only if $g(X) \mid c(X)$. Thus, the parity check is just:

$$c(X) \in \mathcal{C} \iff c(X) \mod g(X) = 0.$$

While one can explicitly write a parity check matrix for cyclic codes, this modular description is often more convenient computationally.

5 BCH Codes

BCH codes (named after Bose, Ray-Chaudhuri, and Hocquenghem) are a particularly important class of cyclic codes. The key idea is to define the generator polynomial *via its roots* rather than its coefficients.

We continue working in $R = \mathbb{F}_q[X]/(X^n - 1)$, identifying cyclic codes with ideals in this ring. We also assume $\gcd(n, q) = 1$, ensuring that $X^n - 1$ has no repeated roots.

Let ζ be a primitive n-th root of unity in some extension field of $\mathbb{F}_q$. Choose integer $d \geq 1$, and define the BCH generator polynomial to be the minimal polynomial over $\mathbb{F}_q$ that has $\zeta, \zeta^2, \ldots, \zeta^d$ as roots:

$$g(X) = \text{lcm}\left(\text{MinimalPoly}_{\mathbb{F}_q}(\zeta), \ldots, \text{MinimalPoly}_{\mathbb{F}_q}(\zeta^d)\right).$$

Then the BCH code is defined as the ideal $\mathcal{C} = (g(X)) \subseteq R$.

This construction guarantees:

- The minimum distance of the code is at least $d + 1$ (to prove soon).
- The dimension of the code is $n - \deg g(X)$.

5.1 Parity Check Matrix for BCH Codes

Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a BCH code of length n, and let $g(X) \in \mathbb{F}_q[X]$ be its generator polynomial. Assume that $g(X)$ is constructed so that it has $\zeta, \zeta^2, \ldots, \zeta^d$ as roots, where ζ is a primitive n-th root of unity in an extension field $\mathbb{F}_{q^m} \supseteq \mathbb{F}_q$. Then:

- Every codeword $c(X) \in \mathcal{C}$ satisfies $g(X) \mid c(X)$.
- Since $g(X)$ has $\zeta, \zeta^2, \ldots, \zeta^d$ as roots, we have $c(\zeta^i) = 0$ for $1 \leq i \leq d$.

Let us write a codeword $c(X) \in \mathcal{C}$ as

$$c(X) = c_0 + c_1 X + c_2 X^2 + \cdots + c_{n-1} X^{n-1},$$

which corresponds to the vector $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$.

Evaluating $c(X)$ at ζ^i gives:

$$c(\zeta^i) = \sum_{j=0}^{n-1} c_j (\zeta^i)^j = 0, \quad \text{for } i = 1, 2, \ldots, d.$$

This system of equations can be written as a matrix equation:

$$\begin{bmatrix} 1 & \zeta & \zeta^2 & \cdots & \zeta^{n-1} \\ 1 & \zeta^2 & (\zeta^2)^2 & \cdots & (\zeta^2)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^d & (\zeta^d)^2 & \cdots & (\zeta^d)^{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

We denote this matrix by H, called the **parity-check matrix** of the BCH code. That is,

$$H = \begin{bmatrix} 1 & \zeta & \zeta^2 & \cdots & \zeta^{n-1} \\ 1 & \zeta^2 & (\zeta^2)^2 & \cdots & (\zeta^2)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^d & (\zeta^d)^2 & \cdots & (\zeta^d)^{n-1} \end{bmatrix} \in \mathbb{F}_{q^m}^{d \times n}.$$

Each row of H corresponds to evaluating codewords at one of the roots ζ^i, and the condition $H\mathbf{c}^\top = 0$ ensures that each $c(X) \in \mathcal{C}$ vanishes at these roots.

5.2 Distance of a BCH Code

Suppose $g(X)$ is the generating polynomial that has $\zeta, \zeta^2, \ldots, \zeta^d$ as roots. What can we say about the minimum distance of the code $\mathcal{C} = (g(X))$?

Theorem 5.1. *A BCH code constructed by requiring $\zeta^1, \ldots, \zeta^d$ to be roots has minimum distance at least $d+1$.*

Proof. Suppose not. Then there exists a nonzero codeword $c(X)$ of weight at most d. That is, $c(X)$ has at most d nonzero coefficients. Let the nonzero coefficients be at positions $i_1 < i_2 < \cdots < i_d$, with corresponding values $c_{i_1}, c_{i_2}, \ldots, c_{i_d}$.

Then, for each $1 \leq k \leq d$, we must have:

$$\sum_{j=1}^{d} c_{i_j} (\zeta^k)^{i_j} = 0.$$

This gives us a homogeneous system:

$$\begin{bmatrix} (\zeta^{i_1}) & (\zeta^{i_2}) & \cdots & (\zeta^{i_d}) \\ (\zeta^{i_1})^2 & (\zeta^{i_2})^2 & \cdots & (\zeta^{i_d})^2 \\ \vdots & \vdots & \ddots & \vdots \\ (\zeta^{i_1})^d & (\zeta^{i_2})^d & \cdots & (\zeta^{i_d})^d \end{bmatrix} \begin{bmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

This is a Vandermonde matrix in $(\zeta^{i_1}, \zeta^{i_2}, \ldots, \zeta^{i_d})$. It is well known that a Vandermonde matrix is invertible if the entries are distinct (i.e., if $\zeta^{i_j} \neq \zeta^{i_k}$ for $j \neq k$), and so the only solution to this system is the zero vector. This contradicts our assumption that the codeword had nonzero entries.

Hence, no codeword of weight $\leq d$ exists, and the minimum weight is at least $d+1$. □

6 The Decoding Problem

Suppose Alice sends a codeword $c(X)$ (a polynomial of degree at most $n-1$) and Bob receives

$$r(X) = c(X) + e(X),$$

where $e(X)$ is the error polynomial with at most $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ nonzero terms (i.e., errors in at most t positions).

We aim to recover $c(X)$ from $r(X)$ efficiently, assuming the number of errors is at most t.

6.1 Locator and Correction Polynomials

Let

$$M = \{i \mid e_i \neq 0\}, \quad |M| = t.$$

Define the following two polynomials:

- **Locator Polynomial**:
$$u(Y) = \prod_{i \in M} (1 - \zeta^i Y),$$
whose roots (in Y) are at $Y = \zeta^{-i}$ for $i \in M$.
- **Correction Polynomial**:
$$v(Y) = \sum_{i \in M} e_i \zeta^i Y \prod_{\substack{j \in M \\ j \neq i}} (1 - \zeta^j Y).$$

6.1.1 Why These Polynomials Help

We have:

$$u(\zeta^{-i}) = 0 \iff i \in M.$$

So evaluating u at all ζ^{-i} determines the error positions.

To find the actual error values, we compute the the formal derivative of $u(Y)$. To compute the formal derivative $u'(Y)$, we apply the product rule:

$$\frac{d}{dY}\left[\prod_{i\in M}(1-\zeta^i Y)\right] = \sum_{i\in M}\left(\frac{d}{dY}(1-\zeta^i Y)\cdot\prod_{\substack{j\in M\\ j\neq i}}(1-\zeta^j Y)\right).$$

Since $\frac{d}{dY}(1-\zeta^i Y) = -\zeta^i$, we get:

$$u'(Y) = -\sum_{i\in M}\zeta^i\prod_{\substack{j\in M\\ j\neq i}}(1-\zeta^j Y).$$

Evaluating v and u' at $Y = \zeta^{-i}$ gives:

$$v(\zeta^{-i}) = e_i\prod_{\substack{j\in M\\ j\neq i}}(1-\zeta^{j-i}),$$
$$u'(\zeta^{-i}) = -\zeta^i\prod_{\substack{j\in M\\ j\neq i}}(1-\zeta^{j-i}),$$

hence,

$$\frac{v(\zeta^{-i})}{u'(\zeta^{-i})} = -\frac{e_i}{\zeta^i}.$$

Thus, we can recover the value e_i once we know u and v.

6.2 Computing the Polynomials

All Bob has is the received word $r(X)$ and the knowledge that $g(X)$ vanishes at $\zeta^1, \ldots, \zeta^{d-1}$. So, we can compute:

$$S_k = r(\zeta^k) = c(\zeta^k) + e(\zeta^k) = e(\zeta^k),$$

for $1 \leq k < d$, since $c(\zeta^k) = 0$.

Let us define the rational function

$$w(Y) = \frac{v(Y)}{u(Y)} = \sum_{i\in M}\frac{e_i\zeta^i Y}{1-\zeta^i Y}.$$

We use a formal geometric series expansion (justified below):

$$\begin{aligned}\frac{1}{1-\zeta^i Y} &= \sum_{k=0}^{\infty} (\zeta^i Y)^k, \\ w(Y) &= \sum_{i \in M} e_i \zeta^i Y \sum_{k=0}^{\infty} (\zeta^i Y)^k \\ &= \sum_{k=0}^{\infty} Y^{k+1} \sum_{i \in M} e_i \zeta^{i(k+1)} \\ &= \sum_{k=1}^{\infty} Y^k e(\zeta^k).\end{aligned}$$

Thus,

$$w(Y) \equiv \sum_{k=1}^{d-1} Y^k e(\zeta^k) \mod Y^d.$$

6.2.1 Justifying the Geometric Expansion

We are working modulo Y^d, so we only care about the polynomial truncated to degree $< d$. Since each $(1-\zeta^i Y)$ divides $1-(\zeta^i Y)^d = 1-\zeta^{id}Y^d = 1$ in $\mathbb{F}_q[Y]/(Y^d)$, it follows that $(1-\zeta^i Y)$ is invertible modulo Y^d, with inverse:

$$(1-\zeta^i Y)^{-1} \equiv \sum_{k=0}^{d-1} (\zeta^i Y)^k \mod Y^d.$$

Hence, the expansion is justified modulo Y^d.

6.3 Recovering $u(Y)$ and $v(Y)$

Let

$$u(Y) = 1 + u_1 Y + \cdots + u_t Y^t, \quad v(Y) = v_1 Y + \cdots + v_t Y^t.$$

From $w(Y) \equiv v(Y)/u(Y) \mod Y^d$, we can clear the denominator and write:

$$v(Y) \equiv w(Y) \cdot u(Y) \mod Y^d.$$

Expanding the right-hand side, we write:

$$w(Y) \cdot u(Y) = \left(\sum_{j=1}^{d-1} S_j Y^j \right) \left(1 + u_1 Y + u_2 Y^2 + \cdots + u_t Y^t\right).$$

This product is computed modulo Y^d, and we collect the coefficients of $Y^1, Y^2, \ldots, Y^{d-1}$. Each coefficient yields a linear equation in the unknowns $u_1, \ldots, u_t$ and $v_1, \ldots, v_t$. For example, the coefficient of Y^1 gives:

$$v_1 = S_1,$$

the coefficient of Y^2 gives:

$$v_2 = S_2 + S_1 u_1,$$

and so on.

In total, we obtain $d-1$ linear equations in $2t$ unknowns.

Matching coefficients of Y^1 through Y^{d-1} gives a system of $d-1$ linear equations in the $2t$ unknowns $u_1, \ldots, u_t, v_1, \ldots, v_t$. Since $2t \leq d-1$, this system can be solved efficiently.

Since we assume $2t \leq d-1$, this system is either determined or underdetermined but solvable, and can be handled efficiently using linear algebra over $\mathbb{F}_q$.

Once we solve for the error-locator polynomial $u(Y)$, we can find the error positions by finding its roots. Then, we can compute the error values and recover the transmitted codeword.

Algorithm 1 Syndrome Decoding for BCH Codes

Require: Received word $r = (r_0, \ldots, r_{n-1}) \in \mathbb{F}_q^n$, designed to correct up to t errors.
Ensure: Corrected codeword $c \in \mathbb{F}_q^n$
Let ζ be a primitive n-th root of unity in some extension of $\mathbb{F}_q$.
Compute S_j:
for $j = 1$ to $d-1$ **do**
$S_j \leftarrow \sum_{i=0}^{n-1} r_i \zeta^{ij}$
end for
Define: $w(Y) \leftarrow S_1 Y + S_2 Y^2 + \cdots + S_{d-1} Y^{d-1}$
Solve Key Equation: Find polynomials $u(Y) = 1 + u_1 Y + \cdots + u_t Y^t$ and $v(Y) = v_1 Y + \cdots + v_t Y^t$ such that

$$v(Y) \equiv w(Y) \cdot u(Y) \mod Y^d$$

This gives a system of $d-1$ linear equations in $2t$ unknowns.
Solve this system using Gaussian elimination
Find Error Locations:
Find all roots of $u(Y)$ in the field extension.
for each root α of $u(Y)$ **do**
Find $i \in \{0, 1, \ldots, n-1\}$ such that $\alpha = \zeta^{-i}$
Add i to list of error positions I
end for
Compute Error Values
Compute formal derivative $u'(Y)$ of $u(Y)$.
for each $i_k \in I$ **do**

$$e_{i_k} \leftarrow -\frac{v(\zeta^{-i_k})}{u'(\zeta^{-i_k})}$$

end for
Correct Errors:
for each $i_k \in I$ **do**
$r_{i_k} \leftarrow r_{i_k} - e_{i_k}$
end for
return $c \leftarrow r$

7 Error Detection Using Generator Polynomials

7.1 Detection of a Single Error

Suppose that a single-bit error occurs in position i during transmission. The received polynomial becomes:

$$r(X) = c(X) + X^i.$$

Since $g(X)$ divides $c(X)$, it follows that:

$$r(X) \bmod g(X) = X^i \bmod g(X).$$

Hence, whether or not the error is detectable reduces to whether $X^i \bmod g(X)$ is nonzero.

Theorem 7.1 (Error Detection Criterion)**.** *A single-bit error in a codeword is detectable if and only if $g(X)$ does not divide X^i for any i, $0 \leq i < n$.*

Proof. If $g(X)$ divides X^i, then $X^i \equiv 0 \mod g(X)$, so $r(X) \equiv c(X) \mod g(X)$, and the syndrome is zero. Thus, the receiver cannot detect the error because $r(X)$ appears to be a valid codeword.

Conversely, if $g(X)$ does not divide X^i, then $r(X) \bmod g(X) \neq 0$, and the error is detected due to a non-zero syndrome. Therefore, detectability of single errors is equivalent to $g(X)$ not dividing any monomial X^i. □

Theorem 7.2 (Sufficient Condition)**.** *If the generator polynomial $g(X)$ has more than one nonzero term (i.e., it is not a monomial), then $g(X)$ does not divide any monomial X^i for $i \geq 0$. Hence, all single-bit errors are detectable.*

Proof. Suppose, for contradiction, that $g(X)$ divides X^i for some $i \geq 0$. Then $X^i = g(X)q(X)$ for some polynomial $q(X)$.

But X^i is a monomial, so the product $g(X)q(X)$ must also be a monomial. This implies that both $g(X)$ and $q(X)$ must be monomials (since any polynomial with more than one term multiplied by anything gives multiple terms).

However, $g(X)$ was assumed to have more than one nonzero term, contradicting this implication. Hence, $g(X)$ cannot divide any X^i, and thus all single-bit errors are detectable. □

Theorem 7.3. *Let $f(X) \in \mathbb{F}_2[X]$ be a polynomial divisible by $1 + X$. Then $f(X)$ has an even number of nonzero terms.*

Proof. We work over $\mathbb{F}_2$, the field with two elements.

Suppose $f(X) = a_0 + a_1X + \cdots + a_nX^n \in \mathbb{F}_2[X]$, and assume $(1 + X) \mid f(X)$. Then $f(1) = 0$, because:

$$1 + X \mid f(X) \quad \Rightarrow \quad f(1) = 0.$$

Now evaluate $f(1)$:

$$f(1) = a_0 + a_1 + \cdots + a_n.$$

Since we are working in $\mathbb{F}_2$, addition is modulo 2, so $f(1)$ equals the number of nonzero coefficients modulo 2.

Therefore, $f(1) = 0$ implies that the number of nonzero coefficients in $f(X)$ is even.

Hence, $f(X)$ has an even number of terms. □

It follows that the code generated by the polynomial $P(X) = 1 + X$ detects not only any single error, but in fact any *odd* number of errors. The reason is that any codeword must have an even number of nonzero coefficients, i.e., an even Hamming weight. Thus, the check symbol in such a code is simply an overall parity check, chosen to ensure that the total number of ones in the code polynomial is even.

A polynomial $P(X) \in \mathbb{F}_2[X]$ is said to have *exponent* e if e is the least positive integer such that

$$P(X) \mid (X^e + 1)$$

in $\mathbb{F}_2[X]$. That is, $P(X)$ divides $X^e + 1$ and does not divide $X^k + 1$ for any positive integer $k < e$.

Theorem 7.4 (Single and Double Error Detection)**.** *Let $P(X)$ be a binary polynomial of exponent e. Then a linear cyclic code of length $n \leq e$ generated by $P(X)$ detects all single and double errors.*

Proof. A codeword is a multiple of $P(X)$: $c(X) = q(X)P(X)$. Suppose a single error occurs at position i, resulting in the received polynomial

$$r(X) = c(X) + X^i.$$

Then $r(X) \bmod P(X) = X^i \bmod P(X)$. If $P(X) \nmid X^i$, the error is detected. Since $P(X) \mid (1 + X^e)$ and $i < e$, X^i cannot be divisible by $P(X)$, so all single errors are detected.

For two errors at positions i and j with $i \neq j$, the received polynomial is

$$r(X) = c(X) + X^i + X^j.$$

Again, unless $P(X) \mid X^i + X^j$, the error will be detected. Since $n \leq e$, $X^i + X^j$ is not divisible by $P(X)$ for any distinct $i, j < n$, so the code detects all double errors as well. □

Theorem 7.5 (Detection of Single, Double, and Triple Errors)**.** *Let $P_1(X)$ be a binary polynomial of exponent e, and define $P(X) = (1 + X)P_1(X)$. Then a cyclic code of length $n \leq e$ generated by $P(X)$ detects all single, double, and triple errors.*

Proof. The factor $1 + X$ ensures that the code detects all error patterns of odd weight (e.g., weight 1 and 3). From Theorem 1, the factor $P_1(X)$ detects all weight 2 errors provided $n \leq e$.

Now, if the received polynomial contains a single error, say X^i, it is not divisible by $1 + X$ and thus not divisible by $P(X)$, hence detected.

For double errors $X^i + X^j$, the weight is 2 and the same argument from the previous theorem applies.

For triple errors, the total weight is odd, and thus the error polynomial is not divisible by $1 + X$. Therefore, it cannot be divisible by $P(X)$ either. Hence, all errors of weight ≤ 3 are detected. □

7.2 Computing the Exponent

To compute the exponent e of a polynomial $P(X)$, one checks successive powers of X to find the smallest e such that

$$1 + X^e \equiv 0 \pmod{P(X)} \quad \text{or equivalently} \quad P(X) \mid (1 + X^e).$$

This can be done via long division or modular exponentiation in $\mathbb{F}_2[X]$.

Algorithm 2 Compute the Exponent of a Polynomial $P(X) \in \mathbb{F}_2[X]$

Require: A nonzero polynomial $P(X) \in \mathbb{F}_2[X]$
Ensure: The exponent e, the smallest positive integer such that $P(X) \mid (X^e + 1)$
1: Let $e \leftarrow 1$
2: Let $f(X) \leftarrow X \mod P(X)$
3: **while** $f(X) \not\equiv 1 \mod P(X)$ **do**
4: $\quad f(X) \leftarrow f(X) \cdot X \mod P(X)$
5: $\quad e \leftarrow e + 1$
6: **end while**
7: **return** e

8 Burst Error Detection and Correction

Definition. A **burst error of length** b is any pattern of errors for which the number of symbols between the *first* and *last* erroneous symbols (inclusive) is b. That is, the burst spans a contiguous block of b positions, and errors may occur at one or more positions within this block.

Example 8.1. Consider a transmitted codeword of length 10:

$$\text{Transmitted:} \quad \underline{1\,0\,0\,1\,1\,0\,1\,0\,1\,1}$$

Suppose an error occurs in the following positions:

$$\text{Received:} \quad \underline{1\,1\,0\,1\,0\,1\,1\,0\,1\,1}$$

Errors occurred at positions 2, 4, 5, and 6. The first error is at position 2 and the last at position 6. Therefore, the total span is $6 - 2 + 1 = 5$. Thus, this is a burst error of length $b = 5$, even though not every bit in this span is erroneous.

Theorem 8.1. *Let $P(X)$ be a generator polynomial of degree r. Then a cyclic code generated by $P(X)$ detects all burst errors of length $\leq r$.*

Proof. Let $E(X)$ be the error polynomial corresponding to a burst of length $\leq r$. Then $E(X) = X^k Q(X)$ where $\deg(Q) < r$. Since $\deg(Q) < \deg(P)$, the polynomial $P(X)$ cannot divide $E(X)$,so the error is detected. □

8.1 General Burst Error Detection in Cyclic Codes

Theorem 8.2 (Burst Error Detection)**.** *Let $g(X)$ be a generator polynomial of a cyclic code of length n and dimension k. Then the code detects all burst errors of length $\leq n - k$.*

Proof. Suppose an error polynomial $e(X)$ represents a burst of length $\leq n - k$. Then $e(X) = X^i Q(X)$ with $\deg Q < n - k$. Since $\deg Q < \deg g(X)$, $g(X)$ does not divide $e(X)$ and the error is detected. □

Theorem 8.3 (Fraction of Undetected Bursts)**.** *Let $C \subseteq \mathbb{F}_2^n$ be a cyclic (n, k)-code. Then, the fraction of burst errors of length $b > n - k$ that go undetected by the code is:*

$$\begin{cases} 2^{-(n-k)} & \text{if } b > n - k + 1, \\ 2^{-(n-k-1)} & \text{if } b = n - k + 1. \end{cases}$$

Proof. Let the error pattern be of the form $E(X) = X^i E_1(X)$, where $E_1(X)$ is a polynomial of degree $b-1$. Since $E_1(X)$ must have both the terms X^0 and X^{b-1}, there are exactly $b-2$ coefficients corresponding to X^j where $0 < j < b-1$ that can be either 0 or 1. Therefore, there are 2^{b-2} distinct possible polynomials $E_1(X)$.

An error is undetected if and only if $E_1(X)$ has $P(X)$ as a factor:

$$E_1(X) = P(X)Q(X),$$

where $\deg(P) = n-k$. Then the degree of $Q(X)$ must be:

$$\deg(Q(X)) = b-1-(n-k).$$

- If $b-1 = n-k$, then $Q(X) = 1$, and there is only one such polynomial:

  $$E_1(X) = P(X),$$

  which gives a single undetected error pattern. Hence, the fraction of undetected bursts is:

  $$\frac{1}{2^{b-2}} = 2^{-(n-k-1)}.$$

- If $b-1 > n-k$, then $Q(X)$ has degree $b-1-(n-k)$. Since $Q(X)$ must also contain both X^0 and $X^{b-1-(n-k)}$, it has $b-2-(n-k)$ intermediate coefficients that can vary freely. Thus, the number of such $Q(X)$ is:

  $$2^{b-2-(n-k)},$$

  and so the fraction of undetected bursts is:

  $$\frac{2^{b-2-(n-k)}}{2^{b-2}} = 2^{-(n-k)}.$$

□

8.2 Detection of Multiple Bursts

Theorem 8.4. *The cyclic code generated by $P(X) = (1+X)P_1(X)$ detects any combination of two burst-errors of length two or less, provided the code length n is greater than e, the exponent of $P_1(X)$.*

Proof. There are four types of error patterns involving at most two bursts of length at most two:

1. $E(X) = X^i + X^j$ — two single-bit errors.
2. $E(X) = (X^i + X^{i+1}) + X^j$ — one burst of length two and one single-bit error.
3. $E(X) = X^i + (X^j + X^{j+1})$ — again, one burst of length two and one single-bit error.
4. $E(X) = (X^i + X^{i+1}) + (X^j + X^{j+1})$ — two bursts of length two.

Observation:

- Error patterns of types (2) and (3) have an *odd number of 1s*, hence they are detected by the $1+X$ factor in $P(X)$, since a codeword divisible by $1+X$ must have an even number of nonzero terms.

- In type (4), we have:

$$E(X) = (X^i + X^{i+1}) + (X^j + X^{j+1}) = (1 + X)(X^i + X^j).$$

  Here, the factor $1 + X$ cancels out with the same factor in $P(X)$. So the divisibility of $E(X)$ by $P(X)$ now depends on whether $X^i + X^j$ is divisible by $P_1(X)$.

- The same issue arises in case (1), where $E(X) = X^i + X^j$. To ensure detection, we must guarantee that $X^i + X^j$ is not divisible by $P_1(X)$.

From above theorem we know that if the code length $n > e$, then $X^i + X^j$ is not divisible by $P_1(X)$ for any distinct i, j, hence such patterns will be detected.

□

9 Fire Codes

Named after Philip Fire, fire codes are a class of cyclic codes designed for detecting and correcting multiple burst errors.

9.1 Definition

Definition (Fire Code)**.** A **Fire code** is a binary cyclic code of length n with generator polynomial

$$g(X) = (X^c + 1)P(X),$$

where $P(X)$ is a primitive polynomial over $\mathbb{F}_2$ of degree r and $c \geq 1$ is a fixed integer. The total length of the code is $n \leq 2^r - 1$ (the exponent of $P(X)$).

The factor $X^c + 1$ enforces a constraint on the relative position of burst errors, enabling the detection of multiple bursts separated by a certain gap.

9.2 Parameters and Capabilities

- Code length: $n \leq \exp(P)$
- Detects: All bursts of length $\leq b$ occurring in two separate places if they are separated by at least c bits

9.3 Burst Error Detection Capability

Theorem 9.1. *Let $P(X) = (X^c + 1)P_1(X)$, where:*

- *$P_1(X)$ is irreducible,*
- *the degree of $P_1(X)$ is at least as great as the length of the shorter burst,*
- *and the code length n is no greater than $\operatorname{lcm}(c, e)$, where e is the exponent to which $P_1(X)$ belongs.*

Then the cyclic code generated by $P(X)$ will detect any combination of two burst errors

$$E(X) = X^i E_1(X) + X^j E_2(X),$$

provided that the sum of the lengths of the bursts is at most $c + 1$.

Proof. Suppose $E(X) = X^i E_1(X) + X^j E_2(X)$ is an undetected error pattern, i.e., $E(X) \in (P(X))$. This means that:

$$P(X) \mid E(X).$$

Let the lengths of $E_1(X)$ and $E_2(X)$ be ℓ_1 and ℓ_2, respectively. Then:

$$\deg(E_1(X)) < \ell_1, \quad \deg(E_2(X)) < \ell_2.$$

We assume that:

$$\ell_1 + \ell_2 \leq c + 1.$$

Now note that $X^c + 1$ divides $P(X)$, so any polynomial divisible by $P(X)$ is also divisible by $X^c + 1$. Therefore, we must have:

$$(X^c + 1) \mid E(X).$$

But since $\ell_1 + \ell_2 \leq c + 1$, and $E_1(X), E_2(X)$ are supported in intervals of size at most ℓ_1, ℓ_2, their non-zero terms do not "wrap around" modulo $X^n - 1$, and the total span of $E(X)$ is at most $c + 1$. Hence, $E(X)$ cannot be a multiple of $X^c + 1$ unless it has a specific symmetric structure.

Moreover, since $P_1(X)$ is irreducible and has degree at least $\min(\ell_1, \ell_2)$, it cannot divide $E_1(X)$ or $E_2(X)$ unless one of them is zero or the burst has a very specific pattern (which would contradict minimality of the burst or irreducibility of $P_1(X)$).

Finally, the condition that $n \leq \text{lcm}(c, e)$, where e is the exponent of $P_1(X)$, ensures that the roots of $P_1(X)$ are appropriately spaced and not aligned with roots of $X^c + 1$ modulo $X^n - 1$, preventing coincidental cancellation.

Thus, under these assumptions, $P(X) \nmid E(X)$, and the code detects the error. □

Remark. The minimum gap c ensures that the two bursts do not interfere in a way that could make their combination divisible by $X^c + 1$. The use of the primitive $P(X)$ ensures that any burst of length $\leq b$ is detected individually.

10 References

References

[1] *CNT notes*, Piyush Kurur, Computation Number Theory 2007, Scribes by Ramprasad Saptharishi, lecture notes

[2] *Cyclic Codes for Error Detection*,W. W. PETERSONt, MEMBER, IRE, AND D. T. BROWNt, MEMBER, IRE, paper