Goppa Codes

Somil Sarode

July 9, 2026

Motivation

- Communication channels introduce errors.
- Error-correcting codes add redundancy to recover data.

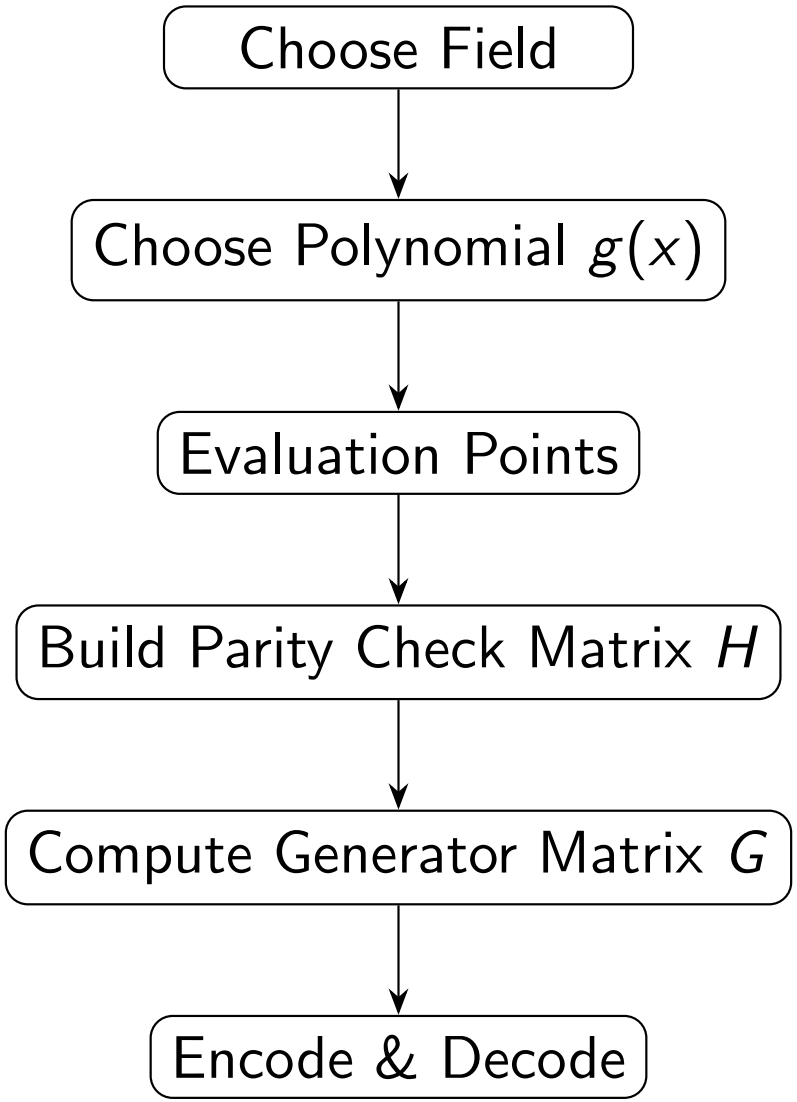
A Little History

- Introduced by Valerii Denisovich Goppa in 1970.
- Binary Goppa codes became famous a few years later for a different reason: in 1978, Robert McEliece used them to build one of the first public-key cryptosystems.
- That cryptosystem is still unbroken today, and a Goppa-code-based scheme (Classic McEliece) is one of the standardized post-quantum encryption methods.

How Goppa Codes Differ from Other Codes

- Goppa codes use a polynomial $g(x)$ and evaluation set L such that no item of L is a root of $g(x)$.
- Reed–Solomon: Message turns into a polynomial $M(x)$, and you would send values of $M(x)$ evaluated at certain points.
- Goppa: Message stays as a vector m, and you send a codeword $c = mG$.
- $g(x)$ is never transmitted. It is only used to build the generator matrix G.

Roadmap


Choose Field
Choose Polynomial $g(x)$
Evaluation Points
Build Parity Check Matrix H
Compute Generator Matrix G
Encode & Decode

Step 1: Choose the Finite Field

We use $GF(2^3)$.

- Eight field elements: $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^6\}$, where α is a root of the field's defining polynomial
- Arithmetic performed modulo an irreducible polynomial, e.g. $x^3 + x + 1$
- Addition is XOR on coefficient vectors; multiplication reduces mod the defining polynomial
- This field will supply both the coefficients of $g(x)$ and the evaluation points in L

Step 2: Choose the Goppa Polynomial

Choose $g(x) = x^2 + x + \alpha^3$, where $\alpha^3 = \alpha + 1$.

- Coefficients come from the extension field $GF(2^3)$, not just $\{0, 1\}$
- Degree $t = 2$
- Crucially, g must have no roots in $GF(2^3)$ – this is why it can't be the same polynomial used to define the field itself (that one has α as a root, by construction)
- Irreducibility (no repeated roots) guarantees the code corrects up to t errors

Step 3: Select Evaluation Points

Choose $L = \{\alpha_1, \ldots, \alpha_n\}$ such that $g(\alpha_i) \neq 0$.

- These become the coordinates of the code, so $n = |L|$ is the codeword length
- Typically L is taken to be all (or most) elements of $GF(2^m)$
- Any α_i with $g(\alpha_i) = 0$ must be excluded, since it would make a column of H undefined
- Here we can take L to be all $n = 8$ elements of $GF(2^3)$ (none are roots of our chosen g)

Step 4: Construct the Parity-Check Matrix

For each evaluation point, $\frac{1}{x-\alpha_i}$ mod $g(x)$ produces one column of H.

- H has $t = 2$ rows over the extension field $GF(2^3)$
- Each field-element entry is expanded into 3 binary rows, giving a 6×8 binary parity-check matrix
- A valid codeword satisfies $Hc^T = 0$.
- H is what actually gets used for both encoding (via G) and decoding

Step 5: Compute the Generator Matrix

Find a basis for the null space of H. $GH^T = 0$.

- The rows of G generate every valid codeword: $c = mG$ for any message m
- If the binary-expanded H has $mt = 6$ independent rows, G has $k = n - mt = 2$ rows
- Typically obtained via Gaussian elimination on H, putting it in the form $[A \mid I]$ so that $G = [I \mid A^T]$

Encoding

Given message $m = (1, 0)$, compute $c = mG$.

- Here $k = 2$ message bits are encoded into $n = 8$ codeword bits, giving rate $R = k/n = 1/4$
- Encoding is just a matrix-vector (vector-matrix) multiplication over $GF(2)$
- The result is transmitted through the communication channel

Transmission and Errors

Suppose one bit flips during transmission. $c = (10100111)$ becomes $r = (10100011)$.

- We can write $r = c + e$, where e is the error vector (here e has a single 1)
- Real channels may flip several bits at once; this code can correct up to t such errors
- The decoder never sees e directly — it must infer it from r alone

Decoding

Compute the syndrome $s = Hr^T$.

- The syndrome identifies the error pattern, allowing us to correct the received word
- Since $Hc^T = 0$, we have $s = Hr^T = H(c + e)^T = He^T$: the syndrome depends only on the error, not the codeword
- Algorithms such as Patterson's algorithm use s to build an error locator polynomial, whose roots point to the flipped bit positions
- Once the error positions are known, flipping those bits in r recovers c, and hence m

Applications

- Reliable communication – deep-space and satellite links, where retransmission is costly or impossible
- Data storage – protecting against bit errors in memory and storage media
- Code-based cryptography – security relies on the hardness of general decoding, not number theory
- McEliece cryptosystem – uses a scrambled generator matrix as a public key; decoding without the trapdoor is believed hard even for quantum computers
- Classic McEliece, built on binary Goppa codes, is a NIST-standardized post-quantum encryption scheme

Summary

1. Choose a finite field.
2. Choose a Goppa polynomial.
3. Select evaluation points.
4. Build H.
5. Compute G.
6. Encode, transmit, decode.

Questions?