1 BCH Codes

For any positive integers $m \geq 3$ and $t \leq 2^{m-1}$, there exists a binary BCH code with the following parameters [3]

- Block length $n = 2^m - 1$
- Number of parity-check digits $n - k \leq m\delta$, with δ, the correcting capacity of the code and k the number of information bits
- Minimum distance $d_{min} \geq 2\delta + 1$

We denote this code by BCH$[n, k, \delta]$. Let α be the primitive element in GF(2^m), the generator polynomial $g(x)$ of the BCH$[n, k, \delta]$ code is given by:

$$g(x) = LCM\{\phi_1(x), \phi_2(x), \cdots, \phi_{2\delta}(x))\}$$

with $\phi_i(x)$ being the minimal polynomial of α^i.

Depending on the parameters of the HQC scheme, we construct shortened BCH codes such that $k = 256$ from the two following BCH codes BCH-1 and BCH-2 (codes from [3]):

code	n	k	δ
BCH-1	1023	513	57
BCH-2	1023	483	60

We obtain the following shortened codes

code	n	k	δ
BCH-S1	766	256	57
BCH-S2	796	256	60

The shortened codes are obtained by substracting 257 (and 227) from BCH-1 (from BCH-2), thus, we have the following BCH codes:

- BCH-S1[$766 = 1023 - 257$, $256 = 513 - 257$, 57]
- BCH-S2[$796 = 1023 - 227$, $256 = 483 - 227$, 60]

We notice that shortening the BCH code does not affect the correcting capacity.

In our case, we will be working in GF(2^{10}), for that we use the primitive polynomial of degree $1 + X^3 + X^{10}$ to build this field (polynomial from [3]). We precomputed the generator polynomials for the two codes that we will be using in our implementation (BCH-S1 and BCH-S2) and we included their Hexadecimal formats in the file `parameters.h`.

2 BCH Decoding

We give a brief reminder on decoding BCH codes following [3]. Consider the BCH code defined by $[n, k, \delta]$, with $n = 2^m - 1$ ($m \geq 0$ of positive integer) and suppose that a code word $v(x) = v_0 + v_1 x + \cdots + v_{n-1}x^{n-1}$ is transmitted and that during transmission, error occurred in the following received vector:

$$r(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-1}x^{n-1}$$

We have that the location of errors are given by the error polynomial $e(x) = e_0 + e_1 x + e_2 x^2 + \cdots + e_{n-1} x^{n-1}$, if $e_i = 1$, then there is an error occurred at that location. Then we can write

$$r(x) = v(x) + e(x)$$

We define the set of syndromes $S_1, S_2, \cdots, S_{2\delta}$ as $S_i = r(\alpha^i)$, with α being the primitive element in $\mathrm{GF}(2^m)$. We have that $r(\alpha^i) = e(\alpha^i)$, since $v(\alpha^i) = 0$ (v is a code word). Suppose that $e(x)$ has t errors at locations $j_1, \cdots, j_t$, then

$$e(x) = x^{j_1} + x^{j_2} + \cdots + x^{j_t},$$

we obtain the following set of equations, where $\alpha^{j_1}, \alpha^{j_2}, \cdots, \alpha^{j_t}$ are unknown:

$$\begin{aligned}
S_1 &= \alpha^{j_1} + \alpha^{j_2} + \cdots + \alpha^{j_t} \\
S_2 &= (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + \cdots + (\alpha^{j_t})^2 \\
S_3 &= (\alpha^{j_1})^3 + (\alpha^{j_2})^3 + \cdots + (\alpha^{j_t})^3 \\
&\vdots \\
S_{2\delta} &= (\alpha^{j_1})^{2\delta} + (\alpha^{j_2})^{2\delta} + \cdots + (\alpha^{j_t})^{2\delta}
\end{aligned}$$

The goal of a BCH decoding algorithm is to solve this system of equations. We define the error location numbers by $\beta_i = \alpha^{j_i}$, which indicate the location of the error. The equations above, can be expressed as follows:

$$\begin{aligned}
S_1 &= \beta_1 + \beta_2 + \cdots + \beta_t \\
S_2 &= \beta_1^2 + \beta_2^2 + \cdots + \beta_t^2 \\
S_3 &= \beta_1^3 + \beta_2^3 + \cdots + \beta_t^3 \\
&\vdots \\
S_{2\delta} &= \beta_1^{2\delta} + \beta_2^{2\delta} + \cdots + \beta_t^{2\delta}
\end{aligned}$$

we define the error location polynomial as:

$$\begin{aligned}
\sigma(x) &= (1 + \beta_1 x)(1 + \beta_2 x) \cdots (1 + \beta_t x) \\
&= \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_t x^t
\end{aligned}$$

We can see that, the roots of $\sigma(x)$ are $\beta_1^{-1}, \beta_2^{-1}, \cdots, \beta_t^{-1}$ which are the inverses of the error location numbers.

We can summarize the decoding procedure of a $\mathrm{BCH}[n, k, \delta]$ code by the following steps:

1. The first step is the computation of $2 \times \delta$ syndromes using the received polynomial
2. The second step is the computation of the error-location polynomial $\sigma(x)$ from the $2 \times \delta$ syndromes computed in the first step (in our implementation we will use the Simplified Berlekamp's Algorithm [2])
3. The third step is to find the error-location numbers by calculating the roots of the polynomial $\sigma(x)$ and returning their inverse (in our implementation we will be using the Chien search algorithm [1])
4. The fourth step is the correction of errors in the received polynomial

Remark 1 *As mentioned before, in our implementation, we deal with shortened BCH code. We notice that we will be using the same decoding procedure described above.*

2.1 Syndromes computations

```
// bch.h
void syndrome_gen(syndrome_set* synd_set, gf_tables* tables, vector_u32* v)
```

The syndromes are computed by evaluating the received polynomial stored in the vector v at the 2× PARAM DELTA consecutive roots of the generator polynomial $\alpha^i, i = 1, 2, \cdots, 2 * \mathsf{PARAM\ DELTA}$. Let us denote by $p(x)$ the polynomial in the vector v, thus the syndromes are

$$p(\alpha), p(\alpha^2), \cdots, p(\alpha^{2\times\mathsf{PARAM\ DELTA}})$$

and they are stored as $GF(2^{10})$ elements in the structure synd set which is the output the function.

2.2 Computing the Error-Location Polynomial

```
// bch.h
void get_error_location_poly(sigma_poly* sigma, gf_tables* tables, syndrome_set* synd_set);
```

This function implements the simplified Berlekamp's algorithm for finding the error location polynomial for binary **BCH** codes.

The algorithm has initial conditions: Tableau à faire (format Lin Costello) Following XX, we define

- t: the correcting capacity of the BCH code
- $S_1, S_2, \cdots, S_{2t}$: the set of syndromes

1. If $d_\mu = 0$

$$\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x)$$
$$l_{\mu+1} = l_\mu$$

2. If $d_\mu \neq 0$, find another row ρ prior to μth row such that:

   - $d_\mu \neq 0$ (the discrepancy of the rwo is not equal to zero)
   - $2 * \rho - l_\rho$ is maximum

   Then compute

$$\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x) + d_\mu d_\rho^{-1} x^{2(\mu-\rho)} \sigma^{(\rho)}(x)$$

   and set

$$l_{\mu+1} = max(l_\mu, l_\rho + 2(\mu - \rho)$$

3. In either case the new value of discrepancy is

$$d_{\mu+1} = S_{2\mu+3} + \sigma_1^{(\mu+1)} S_{2\mu+2} + \cdots + \sigma_{l_{\mu+1}}^{(\mu+1)} S_{2\mu+3-l_{\mu+1}}.$$

4. Increment μ and compute $\mu - l_\mu$
5. Repeat steps 1 to 4, until $\sigma^{(t)}(x)$ is computed

2.3 Finding the Error-Location Numbers

```
// bch.h
void chien_search(uint16_t* error_pos, uint16_t* size, gf_tables* tables, sigma_poly* elp);
```

2.4 Error correction

```
// bch.h
void error_poly_gen(vector_u32* error_poly, uint16_t* error_pos, uint16_t size)
```

References

[1] Robert Chien. Cyclic decoding procedures for bose-chaudhuri-hocquenghem codes. *IEEE Transactions on information theory*, 10(4):357–363, 1964.

[2] Laurie L Joiner and John J Komo. Decoding binary bch codes. In *Southeastcon'95. Visualize the Future., Proceedings., IEEE*, pages 67–73. IEEE, 1995.

[3] Shu Lin and Daniel Costello. Error control coding: Fundamentals and applications. 1983.