This is a brief summary of the main concepts that we cover regarding error-correcting codes.

When data is transmitted over a communication channel (say the internet) errors may happen. We need a way to encode data in a way that allows us to recognize an error and even better to correct it. These kinds of encoding are done using error correcting codes.

The idea is that we encode $x$ into a codeword $C(x)$ that adds some redundant information so that if $y$ is a distortion of $C(x)$, then $x$ can still be recovered from $y$.

See the example below for the simple example of the 4-copy 3 code.

1 General concepts

We take $\Sigma$ to be the alphabet, which is just a finite set of symbols. Typically, we consider the binary alphabet $\Sigma = \{0, 1\}$.

$\Sigma^n$ is the set of strings over the alphabet $\Sigma$ of length $n$. For example, in the case of the binary alphabet, $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$.

The Hamming distance between two strings $x$ and $y$ in $\Sigma^n$ is the number of positions where $x$ and $y$ differ. The notation is $\Delta(x, y)$. For example $\Delta(0111, 1010) = 3$. The Hamming distance satisfies the triangle inequality: for all $u, v, t$ in $\Sigma^n$, $\Delta(u, v) \leq \Delta(u, t) + \Delta(t, v)$. 
Definition 1. A code $C$ is, in the most general sense, just a subset of $\Sigma^n$. The elements of $C$ are called codewords. To be useful, we want the distance between any two codewords to be relatively large. The main parameters of a code are:

- $q$ - the size of the alphabet $\Sigma$. (By default, $q = 2$).
- $n$ - the length of any codeword; this is also called the block length of a code.
- $M$ - the number of codewords;
- $k = \log_q M$ - this is called the information length of the code.
- $d$ - the min. distance between any codewords.
- $k/n$ is the rate of the code. The rate is less than 1, and the closer it is to 1, the better, because there is less redundancy added.
- $d/n$ is the relative distance.

Another useful view is to consider that a code is a mapping $C : \Sigma^k \rightarrow \Sigma^n$. The connection with the above general definition is done by taking the set of codewords (in the general definition) to be the image of $C$ (in the "function" definition). The elements $x$ of $\Sigma^k$ are called messages, and $C(x)$ is the codeword corresponding to the message $x$.

The notation for a code $C$ having the above parameters is $(n, k, d)_q$ code. Another notation is $(n, M, d)_q$ code. Note that the two notations differ in the second component and that $M = q^k$.

Codewords are used to store or to transmit information. Let us suppose, for concreteness, that we are using them to send information over some channel. Say we want to transmit the message $x$. What we actually send is not $x$ itself but $C(x)$. Some bits of $C(x)$ may be corrupted during transport and the received word is $y$, which may differ in several positions from $C(x)$. We want to be able to reconstruct $x$.

This situation is illustrated in the following diagram:

\[
x \xrightarrow{\text{Enc}} C(x) \xrightarrow{\text{Channel}} y \xrightarrow{\text{Dec}} x
\]
Example. The 4-copy-3 code.

In this example, \( \Sigma = \{0, 1\} \), \( k = 4 \). So, the set of messages consists of all 4-bit long binary strings. The encoding is done by repeating each bit of the message 3 times. If the channel flips at most 1 bit, then the original message can be reconstructed. For example if \( x = 0100 \), then \( C(x) = 000111000000 \) (the spaces are inserted to facilitate reading). If the receiver gets \( y = 000111010000 \), then by taking the majority in each block of 3 bits, we can reobtain \( C(x) \) and then \( x \).

Note that in this example, \( d = 3 \), i.e., every two codewords differ in at least 3 positions.

Key properties of a code with distance \( d \).

Let us suppose that \( d \) is odd, and we take \( t = (d - 1)/2 \). (If \( d \) is even, then \( t = (d - 2)/2 \). But almost always, \( d \) is odd.)

(a) If \( \Delta(C(x), y) \leq d - 1 \), (i.e., at most \( d - 1 \) errors happened) then we can detect that \( y \) is not a codeword, and thus we detect that there were errors during transmission. For example, if we modify the above example, and double each bit (instead of tripling it), then \( d = 2 \), and we can detect errors but we cannot correct it.

(b) If \( \Delta(C(x), y) \leq t \), then we can correct \( y \) because \( C(x) \) is the closest codeword to \( y \) and is unique with this property. The reason is unique, is because we cannot have 2 codewords \( c_1 \) and \( c_2 \) at distance at most \( t \) from \( y \), because otherwise the distance between \( c_1 \) and \( c_2 \) would be at most \( 2t \) which is less than \( d \), so this is impossible.

In general, to correct a received word \( y \), we look for the codeword closest to \( y \). This is correction under the maximum likelihood principle: this is the codeword most likely to be the correct one.

In the above example \( d = 3 \), and so \( t = 1 \). Therefore, we can correct one error.

For a string \( x \), and a positive integer \( r \), the ball centered at \( x \) of radius \( r \), denoted \( B(x, r) \) is the set of strings \( y \) at distance at most \( r \) from \( x \).

The number of elements in \( B(x, r) \), where \( x \) is a string of length \( n \) over an alphabet
with $q$ symbols, is called the volume of the ball and is denoted by $Vol(n, r)$. We have
\[ Vol(n, r) = \sum_{j=0}^{r} \binom{n}{j} (q-1)^j. \]

For a code we want $d$ to be large, so that we can correct many errors, but we also want $M$ to be large, so that we can encode many things with the codewords. These two wishes are clearly in conflict: if we take many codewords, (i.e., $M$ is large), then inevitably, some of them will in fact be close (and, therefore, $d$ is small).

**Bounds on the number of codewords**

The following bounds are known:

- **Singleton bound**
  \[ M \leq q^{n-d+1}. \]
  A code that achieves equality in the above relation is called an *MDS code* (maximum distance separable code). Recall that $M = q^k$, so the Singleton bound can be re-written as $q^k \leq q^{n-d+1}$, which implies $d \leq n - k + 1$. In an MDS code, $d = n - k + 1$, so for given $n$ and $k$, it has the largest $d$ possible.

- **Hamming bound** (recall $d = 2t + 1$)
  \[ M \leq \frac{q^n}{\sum_{j=0}^{t} \binom{n}{j} (q-1)^j} = \frac{q^n}{Vol_q(n, t)}. \]
  A code that achieves equality in the above relation is called a *perfect code*. In a perfect code, the set of $n$-long strings can be partitioned into balls of radius $t$.

- **The Gilbert-Varshamov bound** goes the other way.
  It says that for fixed $n, d, q$, there is an $(n, M, d)_q$ code with
  \[ M \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j} (q-1)^j} = \frac{q^n}{Vol_q(n, d-1)}. \]
  The code constructed in the proof of the Gilbert-Varshamov bound is not suitable for real applications. Codes used in practice may have a smaller or a larger $M$ than the one in the Gilbert-Varshamov.
2 Basic notions in Linear Algebra that are useful for codes

Let $n$ be a natural number. We denote by $V_n$, the set of $n$-dimensional vectors, where an $n$-dimensional vector is a sequence of $n$ “numbers” (also called scalars). The “numbers” are the elements of a finite field $F$, which gives the components of the vectors in $V_n$. (A finite fields is a finite set endowed with two arithmetic operations, $+$ and $\times$, that have all the nice properties of normal addition and multiplication on real numbers that we are used to from elementary school. We’ll see more about finite fields later). By default, we will consider that the field is $\mathbb{Z}_2$, formed with the residues modulo 2, 0 and 1, and with the two operations being addition modulo 2 and multiplication modulo 2.

The vectors will be column vectors, i.e., a vector $v$ has the form

$$v = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

If we want a row vector, we need to transpose $v$, so $v^T = (a_1, a_2, \ldots, a_n)$.

Operations with vectors

Vectors can be added:

$$v = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \quad w = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \quad v + w = \begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_n + b_n \end{pmatrix}$$

and we can multiply a vector with a scalar:

$$v = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \quad \alpha v = \begin{pmatrix} \alpha a_1 \\ \alpha a_2 \\ \vdots \\ \alpha a_n \end{pmatrix}$$

Another operation is the inner product of two vectors, whose result is a scalar (not a
Two vectors $v, w$ are orthogonal (or, perpendicular) if $v \cdot w = 0$. Notation: $v \perp w$.

**Independence**

The vectors $v_1, v_2, \ldots, v_k$ are dependent if one of them (say, $v_1$) is a linear combination of the other ones, i.e.,

$$v_1 = \alpha_2 v_2 + \ldots + \alpha_k v_k.$$ 

This is the same as saying that there are scalars $\alpha_1, \ldots, \alpha_k$, not all 0, such that $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k = 0$.

The vectors $v_1, v_2, \ldots, v_k$ are independent if they are not dependent. So the sentence “$v_1, v_2, \ldots, v_k$ are independent” means that if $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k = 0$, then it must be the case that all scalars $\alpha_1, \ldots, \alpha_k$ are zero.

**Basis, dimension of a linear space**

$V_n$ has dimension $n$. This means that there are $n$ fixed vectors (called a basis) such that every vector in $V_n$ is a linear combination of the $n$ vectors in the basis.

For example, we can take the so-called standard basis $e_1, \ldots, e_n$, where each $e_i^T = (0, 0, \ldots, 1, 0, \ldots 0)$ (the 1 is in the $i$-th position, and the other positions are all 0).

Now, let us take $k$ vectors $v_1, \ldots, v_k$ from $V_n$. We define the span of $v_1, \ldots, v_k$ (denoted $\text{span}(v_1, \ldots, v_k)$) to be the set of all vectors that are linear combinations of $v_1, \ldots, v_k$. Formally,

$$\text{span}(v_1, \ldots, v_k) = \{ v \mid v = \alpha_1 v_1 + \ldots + \alpha_k v_k \text{ for some scalars } \alpha_1, \ldots, \alpha_k \}.$$ 

The linear space $W = \text{span}(v_1, \ldots, v_k)$ is a subspace of $V_n$, and if the vectors $v_1, \ldots, v_k$ are independent, then $W$ has dimension $k$. It has a dual (also called complementary) space
$U$ consisting of vectors that are orthogonal to every vector in $W$. The dual space $U$ has dimension $n - k$, in other words $\dim(W) + \dim(U) = n$. Since $U$ has dimension $n - k$ it has a basis $u_1, \ldots, u_{n-k}$ and so $U$ is the span of the vectors $u_1, \ldots, u_{n-k}$.

A useful fact:

**Fact 2.** $v \in \text{span}(v_1, \ldots, v_k) \iff v \perp \text{each } u_1, \ldots, u_{n-k}$.

**Matrix multiplication**

A $n_1$-by-$n_2$ matrix is a table with $n_1$ rows and $n_2$ columns, where the entries are scalars. We can view the rows as $n_2$-dim. vectors, and the columns as $n_1$-dim vectors.

If $A$ is an $n_1$-by-$n_2$ matrix, and $B$ is an $n_2$-by-$n_3$ matrix (so the number of columns of $A$ is equal to the number of rows of $B$) then the matrices $A$ and $B$ can be multiplied, and $AB = C$, where $C$ is an $n_1$-by-$n_3$ matrix and the $(i, j)$ entry of $C$ is the inner product (the $i$-th row of $A$) (j-th column of $B$).

Two important special cases is when we multiply a vector to a matrix $A$, to the left or to the right: $v^T A$, or $Av$. Note that $v^T A$ is the linear combination of the rows of $A$ with the coefficients from $v$, and $Av$ is the linear combination of the columns of $A$, again with the coefficients from $v$.

Thus, if $v^T = (v_1, \ldots, v_k)$ and suppose $A$ has $k$ rows. Then

$$v^T A = v_1 \times \text{row 1 of } A + v_2 \times \text{row 2 of } A + \ldots + v_k \times \text{row } k \text{ of } A.$$ 

If $A$ has $k$ columns, then

$$Av = v_1 \times \text{col 1 of } A + v_2 \times \text{col 2 of } A + \ldots + v_k \times \text{col } k \text{ of } A.$$ 

### 3 Linear codes

Linear codes represent an important family of codes.

**Definition 3.** A $(k, n)$-linear code $C$ is a linear subspace of dimension $k$ of $V_n$.

Recall that $V_n$ is the set of $n$-vectors with elements in a field $\mathbb{F}$

The definition implies that:
For any $c_1$ and $c_2$ in $C \Rightarrow c_1 + c_2 \in C$,

- $c \in C$, $\alpha \in F \Rightarrow \alpha c \in C$.

Most codes used in practice are linear.

**The generating matrix**

One way to present a linear code is to give their generating matrix $G$.

The matrix $G$ has $k$ rows and $n$ columns, for some natural numbers $k$ and $n$. The requirement is that the rows are linearly independent vectors (i.e., no linear combination of them is the 0 vector).

The code $C$ generated by the matrix $G$ is the set of vectors of length $n$ over the field $F$, given by

$$C = \{ v^T G \mid v \in F^k \}. \quad \text{(1)}$$

(Recall that the notation $v^T$ means $v$ transposed. In our course, a vector $v$ is a column vector, so $v^T$ is a row vector.)

In other words, we take all the vectors $v$ of length $k$ over the field $F$, we make all the products $v^T G$ and this produces all the codewords of $C$. In short, the code $C$ is the span of the rows of $G$, i.e., the set of all linear combinations of the rows of $G$.

So,

$$C = \text{span}(G).$$

(span$(G)$ means the span of the rows of $G$).

Using linear algebra terminology, the rows of $G$ are $k$ linearly independent in $F^n$ and thus they form the basis of a subspace of $F^n$ of dimension $k$. The code $C$ consists of this subspace, i.e., $C$ consists of all vectors that are linear combinations of the vectors in the base.

Example 1:
Here the field is $\mathbb{Z}_2$, \( k = 2 \), \( n = 6 \) and thus there are 4 vectors in $\mathbb{Z}_2$ of length 2, namely \( v_1^T = (0, 0), v_2^T = (0, 1), v_3^T = (1, 0), v_4^T = (1, 1) \). The codewords of the code $C$ generated by $G$ are:

\[
\begin{align*}
 v_1^T G &= (0, 0, 0, 0, 0, 0) \\
v_2^T G &= (0, 1, 0, 1, 0, 1) \\
v_3^T G &= (1, 0, 1, 0, 1, 0) \\
v_4^T G &= (1, 1, 1, 1, 1, 1).
\end{align*}
\]

The notation for a linear code is $[n, k, d]_q$, where

- $n$ - the length of the codeword (viewed this time as a vector of length $n$ over $\mathbb{F}$), which is the number of columns in $G$,

- $k$ - the length of the vectors $v$, or, if you prefer, the number of rows in $G$.

- $d$ - the min. distance between any two codewords,

- $q$ - size of $\mathbb{F}$.

As we have seen, the number of codewords is $M = q^k$.

**An important property of linear codes.**

The weight of a word is the number of non-zero symbols in it. For example, the weight of $(1, 0, 1, 1, 0, 0, 0)$ is 3.

\[\text{If } C \text{ is a linear code, then the distance } d \text{ is equal to the minimum weight of a non-zero codeword.}\]

**Proof.** Let $d$ be the distance of $C$ and let $w$ be the minimum weight of a non-zero codeword in $C$.

There are two codewords $c_1$ and $c_2$ that are at distance $d$, i.e., $\Delta(c_1, c_2) = d$. Then $c_1 - c_2$ is also a codeword, it is non-zero, and its weight is $d$. So $d \geq w$. 

9
Conversely, let $c$ be a codeword with weight $w$. Then the distance between $c$ and the codeword 0 is $w$, so $w \geq d$.

We conclude that $d = w$. \hfill \square

Example 2: The 2-copy-on-4-bit code has the following generating matrix:

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Example 3: The Hamming code for 4-bit strings (which is a $(7,4,3)_2$ code) has the following generating matrix:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

**Parity-check matrix**

There is another way to present a linear code, via the so-called *parity-check matrix* $H$. This is a matrix such that the code $C$ is equal to the null space of $H$. In other words,

$$C = \{v \mid Hv = 0\}. \quad (2)$$

It is nice if we can take the $n$-by-$k$ generating matrix $G$ to be of the standard form

$$G = [I_k, P], \quad (3)$$

where $I_k$ is the identity matrix of order $k$, and $P$ is some matrix of type $k$-by-$(n - k)$. A code generated by a matrix in the standard form is called a *systematic code*.

By doing elementary row operations (changing a row with a combination of the rows), we can transform $G$ into a matrix having the standard form which defines an equivalent code (i.e., it has the same $n, k, d$).
Given a matrix $G$ in the standard form, we define the matrix

$$H = [-P^T, I_{n-k}], \quad (4)$$

and one can check that this is the \textit{parity check matrix}.

Indeed,

$$G \times H^T = (I_k, P) \times \begin{pmatrix} -P \\ I_{n-k} \end{pmatrix} = I_k \times (-P) + P \times I_{n-k} = -P + P = 0.$$

Note that the $(i,j)$ entry of $G \times H^T$ is the $i$-th row of $G$ times the $j$-th row of $H$. So the fact that $G \times H^T = 0$ implies that (any row of $G$) $\times$ (any row of $H$) $= 0$. Since any codeword $c$ is in the span of the rows of $G$, it means that $c \times$ (any row of $H$) $= 0$, so $Hc = 0$.

Conversely, it can be shown that if $Hc = 0$, then $c$ must be in the span of the rows of $G$, i.e., $c$ must be a codeword.

\textit{Proof:} Since $Hc = 0$, it follows that $c$ is orthogonal to each row of $H$. So $c$ belongs to the linear space that is dual to span($H$). Note that since span($H$) has dimension $n - k$ (by the form of $H$), its dual has dimension $k$. Also the dual of span($H$) contains span($G$) and since span($G$) has dimension $k$, the dual of span($H$) must be span ($G$).

To summarize: A linear $[n,k,d]$ code $C$ has the following properties:

1. it is the span of the rows of a $k$-by-$n$ matrix $G$, where $G$ is the generating matrix.  
   This is equation (1).

2. it is the null space of an $(n - k)$-by-$n$ matrix $H$, where $H$ is the parity check matrix.  
   This is equation (2).

3. $d = \min$ weight of a non-zero codeword.

4. If $G$ is in the standard form, then it is easy to get $H$, and vice-versa. See (3) and (4).
For the $G$ from Example 1, we have

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

For the Hadamard code given in Example 3, the parity check matrix is

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

Note that the columns of $H$ represent the numbers 3, 5, 6, 7, 1, 2, 4 (written in binary going downwards), which are all the non-zero numbers modulo 8.

This allows us to show that $d = 3$. Indeed, let us show that $d$ is not 2. Suppose there are 2 codewords $c_1$ and $c_2$ and $\Delta(c_1, c_2) = 2$. Then $c_1 + c_2$ would also be a codeword and it would have 2 ones and 5 zeros. Since $H(c_1 + c_2) = 0$ (because $c_1 + c_2$ is a codeword), this would imply that $H$ has two columns whose sum is 0, which means that $H$ would have two columns that are identical, which is false. So $d > 2$. On the other hand if we take $c_1$ - row 1 of $G$ and $c_2$ = row 2 of $G$ + row 3 of $G$, then $\Delta(c_1, c_2) = 3$. So, $d = 3$. So, the Hamming code can correct $t = (d - 1)/2 = 1$ error. This is the same as for the “4-copy-3” code, but the Hamming code has better rate.

The Hamming code has a very efficient correction procedure.

Example: Consider the message $m^T = (1, 0, 1, 1)$. The corresponding codeword is $c = m^T \cdot G = (1, 0, 1, 1, 0, 1, 0)$. We can check that $H \cdot c = 0$.

Suppose the receiver gets $c' = (1, 0, 1, 1, 1, 1, 0)$.

Then

$$Hc' = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$
Since this is not \[
\begin{pmatrix}
1 \\
0 \\
0
\end{pmatrix},
\]
the receiver knows that there was an error. But it’s even better: the vector \[
\begin{pmatrix}
1 \\
0 \\
0
\end{pmatrix}
\] is the 5-th column of \(H\), and this indicates that the error is in the 5-th position of \(c'\). The receiver will flip the 5-th bit of \(c'\) and re-obtain the correct \(c\). Next, he takes the first 4 bits of \(c\) and re-obtains \(m\).

Why does this work? Note that \(c' = c + e\), where \(e = (0, 0, 0, 1, 0, 0)\) is the error vector. Observe that

\[
He' = H(c + e) = Hc + He = He = 5\text{-th col of } H.
\]

So, this is why \(He'\) is the column where the error is.

3.1 Another view on the parity check matrix

Let \(C\) be a linear code. Two vectors \(u, v\) are orthogonal if the inner product \(u \cdot v = 0\).

**Definition 4.** A vector \(v\) is orthogonal to \(C\) if \(v\) is orthogonal to every vector of \(C\).

It can be shown that \(v\) is orthogonal on \(C\) iff \(v\) is orthogonal to every vector in a base of \(C\).

We denote \(C^\perp\) = the set of all vectors \(v\) orthogonal to \(C\). It can be shown that \(C^\perp\) is also a linear subspace of \(\mathbb{F}^n\). Thus, \(C^\perp\) is also a linear code which is called the dual code of \(C\).

**Dimension formula:** \(\dim(C) + \dim(C^\perp) = n\).

So if \(\dim(C) = k\), then \(\dim(C^\perp) = n - k\) (where \(n\) is the dimension of the codes space, i.e., the length of a codeword). A generating matrix for \(C^\perp\) is a \((n - k)\)-by-\(n\) matrix \(H\) whose rows are \(n - k\) vectors that make a base of \(C^\perp\). It can be shown that \(v \in C\) iff \(Hv = 0\). Thus \(H\) is the generating matrix of \(C^\perp\) and the parity check matrix of \(C\).
3.2 Error-correction with the syndrome method.

A vector $v$ is a codeword of a linear code if and only if $Hv = 0$, where $H$ is the parity check matrix.

This gives a way to detect words. If the received word is $c'$ and $Hc' \neq 0$, then $c'$ is not a codeword, and, therefore, there were errors during transmission.

The parity check matrix also simplifies a bit the process of error-correction using the algorithm based on syndromes.

Sketch of the method:

In principle, the error correction can be done as follows (note, that this is a brute-force approach).

We have $c' = c + e$, where $e$ is the error vector, having weight at most $t$ (because we assume that at most $t$ positions are wrong).

To make the correction, we can look for a codeword $c$ and an error vector $e$ such that $c' = c + e$. This works because it is not possible to find two such pairs, i.e., it is not possible to find a pair $(c_1, e_1)$ and another one $(e_2, e_2)$ such that $c' = c_1 + e_1$ and also $c' = c_2 + e_2$. The reason is that otherwise we would have $c_1 - c_2 = e_2 - e_1$. But $c_1 - c_2$ is a codeword and so it has at least $d$ non-zeroes, and $e_2 - e_1$ has weight at most $2t < d$, so this is impossible (we have used the fact that the min weight of any codeword is $d$).

This method works, but it is slow.

The syndrome method improves this situation, i.e., it is faster. For a vector $r$, let $S(r) = Hr$, and we call $S(r)$ the syndrome of $r$.

Note that if the received word $c' = c + e$, then $Hc' = Hc + He = He$, so the received word and the error vector have the same syndrome.

Thus, it is enough to build a table in which we list all the error vectors and their syndromes. When we receive a word $c'$, we calculate $S(c')$, then we lookup in the table for the error vector $e$ with the same syndrome, and we correct $c'$ to $c = c' - e$.

Example.

Consider the code $C$, which is a $[n = 7, k = 3]$ ternary code (numbers are modulo 3)
given by parity-check matrix:

\[
H = \begin{pmatrix}
1 & 2 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 \\
2 & 2 & 2 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

The generating matrix is

\[
G = \begin{pmatrix}
1 & 0 & 0 & 2 & 2 & 0 & 1 \\
0 & 1 & 0 & 1 & 2 & 0 & 1 \\
0 & 0 & 1 & 0 & 2 & 2 & 1
\end{pmatrix}
\]

Let’s find the distance \( d \) of the code. No two columns of \( H \) are dependent, so \( d > 2 \). But \( c = (2, 1, 0, 2, 0, 0, 0) \) is a codeword (it is \( 2 \times \) (first row of \( G \)) + \( 1 \times \) (second row of \( G \))), and it has weight = 3. So \( d = 3 \) and therefore the code can correct \( t = 1 \) error.

Therefore the error vectors are all vectors that have weight 1. We build the table with error vectors and their syndromes.

<table>
<thead>
<tr>
<th>( e )</th>
<th>( S(e) = He )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>1102</td>
</tr>
<tr>
<td>0100000</td>
<td>2102</td>
</tr>
<tr>
<td>0010000</td>
<td>0112</td>
</tr>
<tr>
<td>0001000</td>
<td>1000</td>
</tr>
<tr>
<td>0000100</td>
<td>0100</td>
</tr>
<tr>
<td>0000010</td>
<td>0010</td>
</tr>
<tr>
<td>0000001</td>
<td>0001</td>
</tr>
<tr>
<td>2000000</td>
<td>2201</td>
</tr>
<tr>
<td>0200000</td>
<td>1201</td>
</tr>
<tr>
<td>0020000</td>
<td>0221</td>
</tr>
<tr>
<td>0002000</td>
<td>2000</td>
</tr>
<tr>
<td>0000200</td>
<td>0200</td>
</tr>
<tr>
<td>0000020</td>
<td>0020</td>
</tr>
<tr>
<td>0000002</td>
<td>0002</td>
</tr>
</tbody>
</table>
• Suppose we receive the word \( c' = (1, 0, 1, 0, 1, 2, 2)^T \).

• We compute its syndrome \( S(c') = Hc' = (1, 0, 0, 0)^T \).

• We look up in the table and find the \( e \) with the above syndrome. We get \( e = (0, 0, 0, 1, 0, 0, 0)^T \).

• So the correct codeword is \( c = c' - e = (1, 0, 1, 2, 2)^T \), and the message is \( m = (1, 0, 1) \).

3.3 Cyclic codes

**Definition 5.** A linear code \( C \) is cyclic if:
\[
(c_1, c_2, \ldots, c_n) \in C \implies (c_n, c_1, \ldots, c_{n-1}) \in C.
\]

For example if \( C \) is a cyclic code and \((1, 1, 0, 1) \in C\), then
\[
(1, 1, 1, 0) \in C \\
(0, 1, 1, 1) \in C \\
(1, 0, 1, 1) \in C
\]

Cyclic codes have properties that are useful in the design of codes with a guaranteed \( d \), and with efficient detecting/correction algorithms. BCH codes, Reed-Muller codes, and Reed-Solomon codes, widely used in practice, are of this type.

Of course a cyclic code being linear can be presented via a generating matrix \( G \), or a parity-check matrix \( H \). But they can also be presented using some polynomials.

**Describing cyclic codes with polynomials**

1. Start with a field. Let’s take as an example \( \mathbb{Z}_2 \).

2. Consider the set of polynomials with coefficients in that field. In our example, this is called \( \mathbb{Z}_2[X] \).

3. For codes with codeword length \( n \), take the above polynomials and do addition and multiplication modulo the polynomial \( X^n - 1 \). For ex \( n = 7 \), so we work in \( \mathbb{Z}_2[X]/(X^7 - 1) \) (this is just another way of saying that we work mod \( X^7 - 1 \)).
4. Take \( g(X) \) a divisor of \( X^n - 1 \). This is called the generating polynomial.

For example, \( g(X) = (1 + X^2 + X^3 + X^4) \).

Note that \( X^7 - 1 = (1 + X^2 + X^3 + X^4)(1 + X^2 + X^3) \), so, as it should be, \( g(X) \) is a divisor of \( X^7 - 1 \).

*Observation:* when we work mod \( (X^7 - 1) \), polynomials have degree at most 6. This is so because \( X^7 = 1, X^8 = X, \) etc.

5. The codewords are (the coefficients of) all polynomials that are multiples of \( g(X) \) in \( \mathbb{Z}_2[X]/(X^n - 1) \).

In our example, these are all polynomials in the set \( \{ g(X)(aX^2 + bX + c \mid a, b, c \in \mathbb{Z}_2) \} \) (because they cannot have degree larger than 6).

Example of codewords: \( g(X), Xg(X), X^2g(X) \) which are respectively

\( g(X) = (1, 0, 1, 1, 1, 0, 0) \) (we took the coefficients)

\( Xg(X) = (0, 1, 0, 1, 1, 0) \)

\( X^2g(X) = (0, 0, 1, 0, 1, 1, 1) \)

Note that all the other codewords are linear combinations of three vectors above.

6. Thus, the code is linear with generating matrix:

\[
G = \begin{pmatrix}
1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1
\end{pmatrix}
= \begin{pmatrix}
g(X) \\
Xg(X) \\
X^2g(X)
\end{pmatrix}
\]

Note that from left-to-right, in each row, we write the coefficients in increasing order of the degree.

7. In general, if \( g(X) = a_0 + a_1X + \ldots + a_sX^s \), the generating matrix \( G \) is given by

\[
G = \begin{pmatrix}
a_0 & a_1 & \ldots & a_s & 0 & \ldots & 0 \\
0 & a_0 & \ldots & a_{s-1} & a_s & \ldots & 0 \\
\vdots \\
0 & 0 & \ldots & a_0 & a_1 & \ldots & a_s
\end{pmatrix}
= \begin{pmatrix}
g(X) \\
Xg(X) \\
\vdots \\
X^{n-s-1}g(X)
\end{pmatrix}
\]
8. If the degree of \( g = s \), the generating matrix \( G \) is of type \( (n-s) \)-by-\( n \). So the parameter \( k \) of the code is \( k = n - s \). 

9. The polynomial \( h(X) = \frac{X^n - 1}{g(X)} \) is called the parity check polynomial of the code \( C \) generated by \( g(X) \). 
   
   In our example, \( h(X) = 1 + X^2 + X^3 \). 

10. \( c(X) \) is a codeword \( \iff \) \( h(X)c(X) = 0 \pmod{X^n - 1} \). 
    
    Proof: if \( c(X) \) is a codeword then \( c(x) = g(X)p(X) \pmod{X^n - 1} \) for some polynomial \( p(X) \). Then 
    
    \[
    h(X)c(X) = h(X)g(X)p(X) = (X^n - 1)p(X) = 0 \pmod{X^n - 1}.
    \]
    
    Conversely, if \( c(X)h(X) = 0 \pmod{X^n - 1} \), then \( c(X)h(X) = p(X)(X^n - 1) \), for some polynomial \( p \), so 
    
    \[
    c(X) = p(X)(X^n - 1)/h(X) = p(X)g(X)
    \]
    
    which means that \( c(X) \) is a codeword. 

11. If \( h(X) = b_\ell x^\ell + b_{\ell-1} x^{\ell-1} + \ldots + b_0 \), the parity check matrix \( H \) is given by 
    
    \[
    H = \begin{pmatrix}
     b_\ell & b_{\ell-1} & \cdots & b_0 & 0 & \cdots & 0 \\
     0 & b_\ell & \cdots & b_1 & b_0 & \cdots & 0 \\
     \vdots & & & & & & \\
     0 & 0 & \cdots & b_\ell & b_{\ell-1} & \cdots & b_0
    \end{pmatrix}
    \]
    
    Note that from left-to-right, in each row, we write the coefficients in decreasing order of the degree. 
    
    It can be checked that every row of \( H \) is orthogonal on any row of \( G \) (that is if we take any two rows in \( H \) and respectively in \( G \), their inner product is 0). The reason for this is that \( g(X)h(X) \) is equal to \( X^n - 1 \). To take just one example, the (first row of \( H \)) \cdot (first row of \( G \)) is \( b_\ell a_0 + b_{\ell-1} a_1 + \ldots + b_0 a_\ell \) which is the coefficient of \( X^\ell \) in the product \( h(X)g(X) \) and this coefficient must be zero (because \( h(X)g(X) \) has non-zero coefficients only for \( X^n \) and for 1).
In our example,

\[ H = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1
\end{pmatrix} \]

3.4 BCH Codes

These codes were invented by Bose and Ray Chaudhuri and independently by Hocquenghem in 1959. They are a particular type of cyclic codes. One reason they are important is that they have efficient decoding algorithms, and they can be designed to have a guaranteed large value of \( d \).

They are based on the \textit{BCH bound} given in the following theorem.

\textbf{Theorem 6. (BCH bound).} Let \( g \) be the generating polynomial of a cyclic code \( C \) with parameters \([n, k, d]\), where \( g \) has coefficients over a finite field \( \mathbb{F} \) with \( q = p^m \) elements.

Let \( \alpha \) be a primitive \( n \)-th root of unity. (This means that \( \alpha^n = 1 \) and \( \alpha^m \neq 1 \), for \( m < n \)).

If \( g \) has as roots \( \delta \) consecutive powers of \( \alpha \), then \( d \geq \delta + 1 \).

As mentioned, this allows us to construct codes with a value of \( d \) guaranteed to be larger than a desired value, which is very important for a code.

We sketch the idea of the proof with an example.

Suppose \( g(\alpha) = g(\alpha^2) = g(\alpha^3) = 0 \), so \( g \) has \( \delta = 3 \) consecutive powers of \( \alpha \) as roots. We show that \( d \geq 4 \).

Let \( c = (c_0, c_1, \ldots, c_{n-1}) \) be a non-zero codeword of minimal weight (so equal to \( d \)) and let \( m(X) = c_0 + c_1 + \ldots + c_{n-1}X^{n-1} \) be the associated polynomial. Since \( m(X) \) is a multiple of \( g(X) \), we know that \( m(\alpha) = m(\alpha^2) = m(\alpha^3) = 0 \).

Suppose \( (c_0, c_1, \ldots, c_{n-1}) \) has weight 3, say \( c_2, c_5, c_7 \) are not zero and the other \( c_i \)’s are zero.
Then
\[
\begin{pmatrix}
\alpha^2 & \alpha^5 & \alpha^7 \\
\alpha^4 & \alpha^{10} & \alpha^{14} \\
\alpha^6 & \alpha^{15} & \alpha^{21}
\end{pmatrix}
\begin{pmatrix}
c_2 \\
c_5 \\
c_7
\end{pmatrix}
= \begin{pmatrix}
m(\alpha) \\
m(\alpha^2) \\
m(\alpha^3)
\end{pmatrix}
= \begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix}
\]

But the matrix
\[
A = \begin{pmatrix}
\alpha^2 & \alpha^5 & \alpha^7 \\
\alpha^4 & \alpha^{10} & \alpha^{14} \\
\alpha^6 & \alpha^{15} & \alpha^{21}
\end{pmatrix}
\]

has non-zero determinant (it is the well-known Vandermonde determinant), which implies $c_2 = c_5 = c_7 = 0$ (because $Ax = 0$ has only solution $x = (0,0,0)^T$). This means that $c$ is the zero codeword, which is a contradiction. In the same way, $d$ cannot be 2, so it has to be at least 4.

**Construction of BCH Codes**

1. Work in a finite field $\mathbb{F}$. We know that $|\mathbb{F}| = q = p^m$, for some prime number $p$.
2. Take $n$, not a multiple of $p$, and then take the polynomial $X^n - 1$.
3. Factor $X^n - 1 = f_1(X)f_2(X)\ldots f_t(X)$ into irreducible polynomials over $\mathbb{F}[X]$.
4. Take $\alpha$ a primitive $n$-th root of unity (Recall that this means that $\alpha^n = 1$ and $\alpha^m \neq 1$, for $m < n$. $\alpha$ may exist in an extension $\mathbb{F}'$ of $\mathbb{F}$ similar to $\mathbb{C}$ an extension of $\mathbb{R}$; See Remark 8 below).
5. Then $\alpha^0 = 1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ are distinct roots of $X^n - 1$ and $X^n - 1 = (X - 1)(X - \alpha)(X - \alpha^2)\ldots(X - \alpha^{n-1})$.

Some of these factors when multiplied form $f_1$, some other ones form $f_2$, etc.
6. Let $q_j(X)$ be the polynomial from the list $f_1, f_2, \ldots, f_t$ for which $\alpha^j$ is a root.

For integers $s \geq -1$ and $d$, the BCH code of designed distance $d$ is the code generated by the polynomial

$g(X) = \text{least common multiple of } q_{s+1}(X), q_{s+2}(X), \ldots, q_{s+d-1}(X)$. 

Key property of \( g \): its roots include \( d-1 \) consecutive powers of \( \alpha \), namely \( \alpha^{s+1}, \alpha^{s+2}, \ldots, \alpha^{s+d-1} \).

So, by the BCH bound, the code has distance at least \( d \).

Example:

Take \( F = \mathbb{Z}_2 \), \( n = 7 \).

\[
(X^7 - 1) = (X - 1)(X^3 + X^2 + 1)(X^3 + X + 1).
\]

Next we need to find a 7-th root of unity. We do this with the following observation.

**Fact 7.** If \( \alpha \) is a root of \( X^3 + X + 1 \), then it is a 7-th root of unity.

**Proof sketch** - showing that \( \alpha \) is a 7-th root of unity.

We need to show that \( \alpha^7 = 1 \), but \( \alpha^j \neq 1 \), for all \( j < 7 \).

Because \( \alpha^3 + \alpha + 1 = 0 \), it follows that \( \alpha^3 = -\alpha - 1 = \alpha + 1 \) (taking into account that \( -1 = 1 \mod(2) \)). So \( \alpha^7 = \alpha \cdot \alpha^3 \cdot \alpha^3 = \alpha(\alpha + 1)(\alpha + 1) = \alpha(\alpha^2 + 2\alpha + 1) = \alpha(\alpha^2 + 1) = \alpha^3 + \alpha = 1 \).

Next, \( \alpha^6 = (\alpha^3)^2 = (\alpha + 1)^2 = \alpha^2 + 2\alpha + 1 = \alpha^2 + 1 \neq 1 \) (because \( \alpha^2 \neq 0 \)).

In a similar way, one can show that none of \( \alpha^5, \alpha^4, \alpha^3, \alpha^2, \alpha \) is equal to 1.

*End sketch of proof.*

Next we show that \( \alpha^2 \) and \( \alpha^4 \) are also roots of \( X^3 + X + 1 \).

Since \( \alpha^3 = \alpha + 1 \), by squaring, we get \( \alpha^6 = \alpha^2 + 2\alpha + 1 = \alpha^2 + 1 \). So \( \alpha^6 + \alpha^2 + 1 = 0 \), which says that indeed \( \alpha^2 \) is a root of \( X^3 + X + 1 \). In the same way, it follows that \( \alpha^4 \) is also a root of \( X^3 + X + 1 \).

So, \( X^3 + X + 1 = (X - \alpha)(X - \alpha^2)(X - \alpha^4) \).

Clearly \( \alpha^0 = 1 \) is a root of \( X - 1 \).

It means that the other three powers of \( \alpha \), namely \( \alpha^3, \alpha^5, \alpha^6 \) must be roots of \( X^3 + X^2 + 1 \).

So, \( X^3 + X^2 + 1 = (X - \alpha^3)(X - \alpha^5)(X - \alpha^6) \).

Thus:

\[
q_0(x) = X - 1,
\]

\[21\]
\[ q_1(X) = q_2(x) = q_4(X) = X^3 + X + 1, \]
\[ q_3(X) = q_5(x) = q_6(X) = X^3 + X^2 + 1. \]

The BCH code for \( s = -1 \) and \( d = 3 \) is generated by
\[ g(X) = \text{lcm}(q_0(X), q_1(X)) = (X - 1)(X^3 + X + 1) = X^4 + X^3 + X^2 + 1. \]

We get a code with \( n = 7, d = 3, k = n - \text{deg}(g) = 7 - 4 = 3. \)

We can do better with \( s = 0 \) and \( d = 3. \) We get
\[ g_1(X) = \text{lcm}(q_1(X), q_2(X)) = X^3 + X + 1, \]
a polynomial of degree 3, so now the distance of the code is at least 3 (as before), but \( k = n - \text{deg}(g_1) = 7 - 3 = 4. \)

**Remark 8.** How to find an extension \( F' \) of \( F \) where a primitive \( n \)-th root of unity \( \alpha \) exists.

If for some \( t, n \) divides \( q^t - 1 \), then \( F' \) and \( \alpha \) exist. We can obtain them as follows.

We take an irreducible polynomial \( h \) of degree \( t \) (it can be shown that such a polynomial exists), and define the extension \( F' = F[X]/h. \) Then \( F' \) has size \( q^t \), and (like any finite field) has a generator \( g. \) This means that \( F'^* = F' \setminus \{0\} = \{g^0, g^1, \ldots, g^{q^t-2}\}. \) If \( n \times m = q^t - 1, \)
then we can take \( \alpha = g^m. \)

For instance, say \( q = 2 \) and we want \( n = 21. \)

We need to find \( t \) so that \( n \) divides \( q^t - 1. \) We can take \( t = 6 \) (because \( n = 21 \) divides \( 2^6 - 1 = 63. \))

Now we take the field \( F \) to be \( \text{GF}(2^6). \) \( F^* \) is cyclic, and so there is some generator \( \beta \) such that \( F^* = \{1, \beta, \beta^2, \ldots, \beta^{62}\}. \)

Now we can take \( \alpha = \beta^3 \) and \( \alpha \) is a 21-th primitive root of 1, as desired. Indeed \( \alpha^{21} = \beta^{3 \cdot 21} = \beta^{63} = 1, \)
and it can be checked that \( \alpha \) to smaller powers is not 1.

**Decoding BCH codes**

One important advantage of BCH codes is that they have efficient error-correction algorithms (for ex. Peterson-Gorenstein-Zierler algorithm). We will sketch the method for the particular case of correcting 1 error.

1. Let \( C \) be a BCH code with designated \( d \geq 3 \), so \( t = 1 \) (thus, we can correct 1 error).
2. Let \( g \) be a generating polynomial for \( C \), \( \alpha \) a primitive \( n \)-th root of unity, and suppose \( g(\alpha^{k+1}) = g(\alpha^{k+2}) = 0 \)(two consecutive powers of \( \alpha \) are roots of \( g \)).
3. Take
\[
H = \begin{pmatrix}
1 & \alpha^{k+1} & \alpha^{2(k+1)} & \ldots & \alpha^{(n-1)(k+1)} \\
1 & \alpha^{k+2} & \alpha^{2(k+2)} & \ldots & \alpha^{(n-1)(k+2)}
\end{pmatrix}
\]

4. Consider a codeword \( C = (c_0, \ldots, c_{n-1}) \) and the corresponding polynomial
\[
m(X) = c_0 + \ldots + c_{n-1}X^{n-1},
\]
which we know is a multiple of \( g \). So, \( m(\alpha^{k+1}) = m(\alpha^{k+2}) = 0 \).

5. So
\[
H \begin{pmatrix}
c_0 \\
\vdots \\
c_{n-1}
\end{pmatrix} = \begin{pmatrix}
m(\alpha^{k+1}) \\
m(\alpha^{k+2})
\end{pmatrix} = \begin{pmatrix}
0 \\
0
\end{pmatrix}
\]

6. Suppose the received word is \( c' = c + e \). Here \( e \) is the error vector, which has a single non-zero entry.

7. The correcting algorithm on input \( c' \) does the following:
   (a) Compute
   \[
   H \cdot c' = \begin{pmatrix}
s_1 \\
s_2
\end{pmatrix}
   \]
   (b) If \( s_1 = 0 \), there is no error, and we stop.
   (c) Else, compute \( s_2/s_1 \). It is \( \alpha^{j-1} \), for some \( j \), and this means that the error is in the \( j \)-th position.

   If we work in the field \( \mathbb{Z}_2 \), we flip the \( j \)-th position in \( c' \), and we have obtained the correct \( c \).

   If we work in a larger field, we compute \( e_j = \frac{s_1}{\alpha^{(j-1)(k+1)}} \). This is the \( j \)-th entry in the error vector \( e \), the other entries are all 0.

8. This is why it works:
\[
c \text{ is the codeword.}
\]
\[
e = (0, \ldots, e_j, \ldots, 0) \text{ is the error vector, which we want to find.}
\]
\[
c' = c + e.
\]
\[HC' = H(c + e) = Hc + He = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \]

We see that
\[s_1 = e_j \alpha^{(j-1)(k+1)} \text{ and } s_2 = e_j \alpha^{(j-1)(k+2)}.\]

Then \(s_2/s_1 = \alpha^{j-1}\) and \(s_1/\alpha^{(j-1)(k+1)} = e_j.\)

### 3.5 Reed-Solomon Codes

- a particular type of BCH codes
- very widely used: data storage (CDs, DVDs, hard drives), wireless (cell phones, wireless communication), satellite and digital TV, modems, DSL, etc.
- Some standards are RS codes: \(RS^*(204, 188, 17)_{256}\) is ITU J.83(A), \(RS^*(128, 122, 7)_{256}\) is ITU J.83(B)
- \(RS^*(255, 223, 33)_{256}\) also common in practice.

#### Construction of Reed-Solomon codes

- \(F\), finite field with \(q\) elements.
- must have \(n = q - 1\), so \(q\) has to be pretty large.
- \(F\) has a primitive \(n\)-th root of unity \(\alpha\) (because \(F\) has a generator, and a generator is an \(n\)-th primitive root of unity). So \(\alpha^n = 1\) and \(\alpha^j \neq 1\) for \(j < n\). Note that \(\alpha\) is in \(F\), we do not need to go to an extension of \(F\) to find \(\alpha\).
- Choose \(d, 1 \leq d \leq n\), and take \(g(X) = (X - \alpha)(X - \alpha^2)\ldots(X - \alpha^{d-1}).\)
- The code generated by the polynomial \(g\) is a Reed-Solomon code.
- The parameters: \(n\), the distance is \(d\), and \(k = n - (\text{degree } g) = n - d + 1\). So RS codes are MDS.
Example: RS(7,3,5)$_8$.

The field is $GF(8)$, $q = 8$, $n = q - 1 = 7$, $d = 5$, $k = 7 - 5 + 1 = 3$.

The generating polynomial is

$$g(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4) = X^4 + \alpha^3 X^3 + X^2 + \alpha X + \alpha^3,$$

where $\alpha$ is a primitive 7th root of unity.

The generating matrix is

$$G = \begin{pmatrix}
\alpha^3 & \alpha & 1 & \alpha^3 & 1 & 0 & 0 \\
0 & \alpha^3 & \alpha & 1 & \alpha^3 & 1 & 0 \\
0 & 0 & \alpha^3 & \alpha & 1 & \alpha^3 & 1
\end{pmatrix}$$

Recall (look at notes on finite fields) that the elements of $GF(8)$ are polynomials of degree $\leq 2$ with coefficients in $\mathbb{Z}_2$ and $+$ and $\ast$ are done modulo $Y^3 + Y + 1$ (which is a particular irreducible polynomial of degree 3 in $\mathbb{Z}_2[X]$).

We have the following equivalent representations:

<table>
<thead>
<tr>
<th>$\alpha$ powers</th>
<th>polynomial</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$Y$</td>
<td>010</td>
</tr>
<tr>
<td>$\alpha^2$</td>
<td>$Y^2$</td>
<td>100</td>
</tr>
<tr>
<td>$\alpha^3$</td>
<td>$Y + 1$</td>
<td>011</td>
</tr>
<tr>
<td>$\alpha^4$</td>
<td>$Y^2 + Y$</td>
<td>110</td>
</tr>
<tr>
<td>$\alpha^5$</td>
<td>$Y^2 + Y + 1$</td>
<td>111</td>
</tr>
<tr>
<td>$\alpha^6$</td>
<td>$Y^2 + 1$</td>
<td>101</td>
</tr>
<tr>
<td>$\alpha^7$</td>
<td>1</td>
<td>001</td>
</tr>
</tbody>
</table>

So, in the generating matrix $G$, we can move from the $\alpha$ powers representation above to the binary representation which is more familiar. For example the first row

$$\begin{pmatrix}
\alpha^3 & \alpha & 1 \\
\end{pmatrix}$$

becomes

$$\begin{pmatrix}
011 & 010 & 001 & 011 & 001 & 000 & 000
\end{pmatrix}$$
Encoding. One example: \( m \) of length \( k = 3 \) over \( \text{GF}(8) \),

\[
m = \alpha^4 0 \alpha^4
\]

which in binary is

\[
m = 110 000 110
\]

The code is

\[
m \cdot G = (\alpha^7, \alpha^5, \alpha^4 + \alpha^7, \alpha^7 + \alpha^5, \alpha^4 + \alpha^5, \alpha^7, \alpha^4)
\]

which in binary is

\[
(001 111 111 110 000 001 110).
\]

Why are RS codes popular

They handle well errors that occur in bursts.

Consider \( \text{GF}(2^8) \).

Each element of the field is a byte.

Say \( n = 2^8 - 1 = 255 \) and \( d = 33 = 2 \times 16 + 1 \). So we can correct 16 errors, which means 16 bytes. But if the error is 121 consecutive bits, they must fall in 16 bytes, so they can be corrected.