### Finite Fields

From now on, we look only at channels whose input and output alphabets are the same, each consisting of the elements of some finite field.

A finite field consists of a finite collection of "numbers" that behave like real and complex numbers. Specifically,

- Addition and multiplication are defined, and they are commutative and associative, and multiplication is distributive over addition.
- There are numbers called 0 and 1, such that \( a + 0 = a \) and \( a \cdot 1 = a \) for all \( a \).
- Subtraction and division (except by 0) can be done, and these operations are the inverses of addition and multiplication.

### The Finite Field \( \mathbb{F}_2 \)

The smallest finite field, called \( \mathbb{F}_2 \) or \( GF(2) \), has just two elements, 0 and 1. Addition and multiplication are defined as follows:

\[
\begin{align*}
0 + 0 &= 0 & 0 \cdot 0 &= 0 \\
0 + 1 &= 1 & 0 \cdot 1 &= 0 \\
1 + 0 &= 1 & 1 \cdot 0 &= 0 \\
1 + 1 &= 0 & 1 \cdot 1 &= 1
\end{align*}
\]

This can also be seen as arithmetic modulo 2 (called \( \mathbb{Z}_2 \)).

Viewed as logical operations, addition is the same as 'exclusive-or', and multiplication is the same as 'and'.

Note: In \( \mathbb{F}_2 \), \( -a = a \), and hence \( a + b = a + b \).

### Other Finite Fields

There is a finite field with \( p \) elements for every prime \( p \). This field is the same as \( \mathbb{Z}_p \), in which arithmetic on \( 0, \ldots, p - 1 \) is done module \( p \).

For example, \( \mathbb{F}_3 = F_3 \) works as follows:

\[
\begin{align*}
0 + 0 &= 0 & 0 \cdot 0 &= 0 \\
0 + 1 &= 1 & 0 \cdot 1 &= 0 \\
0 + 2 &= 2 & 0 \cdot 2 &= 0 \\
1 + 0 &= 1 & 1 \cdot 0 &= 0 \\
1 + 1 &= 2 & 1 \cdot 1 &= 1 \\
1 + 2 &= 0 & 1 \cdot 2 &= 2 \\
2 + 0 &= 2 & 2 \cdot 0 &= 0 \\
2 + 1 &= 0 & 2 \cdot 1 &= 2 \\
2 + 2 &= 1 & 2 \cdot 2 &= 1
\end{align*}
\]

There’s also a finite field for every integer power of a prime, with \( p^e \) elements. These fields are not the same as \( \mathbb{Z}_{p^e} \), which is not a field when \( e > 1 \). (See J&J, Section 6.1.)

### Vector Spaces Over a Finite Field

Just as we can define vectors over real numbers, we can define vectors over a finite field. We get to add such vectors, and multiply them by a scalar from the finite field.

We can think of these vectors as \( n \)-tuples of field elements. For instance, with vectors of length five over \( \mathbb{F}_2 \):

\[
\begin{align*}
(1,0,0,1,1) + (0,1,0,0,1) &= (1,1,0,1,0) \\
1 \cdot (1,0,0,1,1) &= (1,0,0,1,1) \\
0 \cdot (1,0,0,1,1) &= (0,0,0,0,0)
\end{align*}
\]

Most properties of real vector spaces continue to hold — eg, the existence of basis vectors.

We refer to the vector space of all \( n \)-tuples from \( F_q \) as \( F_q^n \). We will use boldface letters such as \( \mathbf{u} \) and \( \mathbf{v} \) to refer such vectors.
**Linear Codes**

We can view $F_q^n$ as the input and output alphabet of the $n$th extension of a channel with input and output alphabet $F_q$.

A code, $C$, for this extension of the channel is a subset of $F_q^n$.

$C$ is a **linear code** if the following conditions hold:

1) If $u$ and $v$ are codewords of $C$, then $u + v$ is also a codeword of $C$.

2) If $u$ is a codeword of $C$ and $a$ is in $F_q$, then $au$ is also a codeword of $C$.

In other words, $C$ must be a subspace of $F_q^n$.

Note that the all-zero codeword must be in $C$, since $0 = 0u$ for any $u$.

Note: For binary codes (over $F_2$), condition (2) will always hold if condition (1) does, since $1u = u$ and $0u = 0 = u + u$.

---

**Linear Codes From Basis Vectors**

We can construct a linear code by choosing $k$ linearly-independent basis vectors from $F_q^n$.

We'll call the basis vectors $u_1, \ldots, u_k$. We define the set of codewords to be all those vectors that can be written in the form

$$a_1u_1 + a_2u_2 + \cdots + a_ku_k$$

where $a_1, \ldots, a_n$ are elements of $F_q$.

The codewords obtained with different $a_1, \ldots, a_k$ are all different. (Otherwise $u_1, \ldots, u_k$ wouldn't be linearly-independent.)

There are therefore $q^k$ codewords.

We can encode a block consisting of $k$ symbols, $a_1, \ldots, a_k$, from $F_q$ as a codeword of length $n$ using the formula above.

This is referred to as an $[n, k]$ code.

---

**Linear Codes From Linear Equations**

Another way to define a linear code for $F_q^n$ is to provide a set of simultaneous equations that must be satisfied for $v$ to be a codeword.

These equations have the form $b \cdot v = 0$, i.e.

$$b_1v_1 + b_2v_2 + \cdots + b_nv_n = 0$$

The set of solutions is a linear code because

1) $b \cdot u = 0$ and $b \cdot v = 0$ implies $b \cdot (u + v) = 0$.

2) $b \cdot v = 0$ implies $b \cdot (av) = 0$.

If we have $n - k$ such equations, and they are independent, the code will have $q^k$ codewords.

---

**A $[3, 1]$ Code Over $F_3$**

As a simple example, consider the code for $F_3^3$ defined by the following equations that must be satisfied by a codeword $v$:

$$v_1 + v_2 = 0$$

$$v_2 + 2v_3 = 0$$

There should be three codewords in this code.

One of them is $(1, 2, 2)$, since in $F_3$ (which is $\mathbb{Z}_3$), $1 + 2 = 0$ and $2 + 2 \cdot 2 = 2 + 1 = 0$.

We can take this codeword as a basis vector, and find the other two codewords as the multiples of it:

$$0(1, 2, 2) = (0, 0, 0), \quad 2(1, 2, 2) = (2, 1, 1)$$
The Repetition Codes Over $F_2$

A repetition code over $F_2^2$ has only two codewords — one has all 0s, the other all 1s.

This is a linear $[n,1]$ code, with $(1, \ldots, 1)$ as the basis vector.

The code is also defined by the following $n-1$ equations satisfied by a codeword $v$:

\[ v_1 + v_2 = 0, \ v_2 + v_3 = 0, \ldots, \ v_{n-1} + v_n = 0 \]

The Single Parity-Check Codes

An $[n,n-1]$ code over $F_2$ can be defined by the following single equation satisfied by a codeword $v$:

\[ v_1 + v_2 + \cdots + v_n = 0 \]

In other words, the parity of all the bits in a codeword must be even.

This code can also be defined using $n-1$ basis vectors. One choice of basis vectors when $n = 5$ is as follows:

\[ (1,0,0,0,1) \]
\[ (0,1,0,0,1) \]
\[ (0,0,1,0,1) \]
\[ (0,0,0,1,1) \]


Recall the following code from lecture 9b:

\[
\{ \ 00000, \ 00111, \ 11001, \ 11110 \ \}
\]

Is this a linear code? We need to check that all sums of codewords are also codewords:

\[
\begin{align*}
00111 + 11001 &= 11110 \\
00111 + 11110 &= 11001 \\
11001 + 11110 &= 00111 \\
\end{align*}
\]

We can generate this code using 00111 and 11001 as basis vectors. We then get the four codewords as follows:

\[
\begin{align*}
0 \cdot 00111 + 0 \cdot 11001 &= 00000 \\
0 \cdot 00111 + 1 \cdot 11001 &= 11001 \\
1 \cdot 00111 + 0 \cdot 11001 &= 00111 \\
1 \cdot 00111 + 1 \cdot 11001 &= 11110 \\
\end{align*}
\]

The [7,4] Binary Hamming code

The [7,4] Hamming code is defined over $F_2$ by the following equations that are satisfied by a codeword $u$:

\[
\begin{align*}
u_4 + u_5 + u_6 + u_7 &= 0 \\
u_2 + u_3 + u_6 + u_7 &= 0 \\
u_1 + u_3 + u_5 + u_7 &= 0 \\
\end{align*}
\]

Since these equations are independent, there should be 16 codewords.

We can also define the code in terms of the following four basis vectors:

\[
\begin{align*}
1001100, \ 0101010, \ 1110000, \ 1101001 \\
\end{align*}
\]

We will see later that this code is capable of correcting any single error.