Chapter 2

Linear Codes

In Chapter 1, we used $0, 1, 2, \ldots$ purely as symbols in an alphabet A, and our code could be any subset of A^n. To make progress we now want to do arithmetic with our symbols, so our alphabet must be a field F. Then we can regard our words in A^n as vectors in F^n, which is a vector space over F. Moreover, we shall require that our code C is a subspace of F^n.

2.1 Finite Fields

A field is a set with two binary operations, which obey the standard rules of arithmetic.

Definition 2.1. A non-empty set F with addition $F \times F \to F$, mapping (a, b) to $a + b$, and multiplication $F \times F \to F$, mapping (a, b) to ab, is called a **field** if the following axioms hold.

i) Associativity of addition: For every a, b, and c in F, $(a + b) + c = a + (b + c)$

ii) Additive identity: There exists 0 in F such that for every $a \in F$, $a + 0 = a = 0 + a$

iii) Additive inverse: For every $a \in F$, there exists $b \in F$ such that $a + b = 0 = b + a$

iv) Commutative addition: For every a and b in F, $a + b = b + a$

v) Associativity of multiplication: For every a, b, and c in F, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

vi) Multiplicative identity: There exists 1 in F such that, for every $a \in F$, $1 \cdot a = a = a \cdot 1$

vii) Multiplicative inverse: For every $a \in F$, $a \neq 0$, there exists $b \in F$ such that $a \cdot b = 1 = b \cdot a$

viii) Commutative multiplication: For every a and b in F, $a \cdot b = b \cdot a$

ix) Distributivity: For every a, b, and c in F, $(a+b) \cdot c = a \cdot c + b \cdot c$ and $a \cdot (b+c) = a \cdot b + a \cdot c$

x) Multiplicative and additive inverses are different: $0 \neq 1$

Axioms i - iv make $(F, +)$ into an abelian (or commutative) group; axioms v - viii make $(F - \{0\}, \cdot)$ into another abelian group; axioms i - vi & ix make $(F, +, \cdot)$ a ring. Note that the definitions of addition and multiplication as maps $F \times F \to F$ imply that these operations are closed. It is easy to show that the identities 0 and 1 are unique, and that for each $a \in F$, the additive inverse $-a$ and the multiplicative inverse a^{-1} are unique. We also have some convenient notations: for m a non-negative integer, we write $m \cdot a$ for adding, and a^m for multiplying, m copies of a. It's all very familiar.

However, the fields you know best are the reals $\mathbb{R}$, the complex numbers $\mathbb{C}$, and the rationals $\mathbb{Q}$, and these are no good as alphabets, because they are infinite.

To find finite fields, we start from arithmetic modulo n. Here the set $\mathbb{Z}/n$ (or $\mathbb{Z}_n$, or $\mathbb{Z}/n\mathbb{Z}$) is the congruence classes $\{[0]_n, [1]_n, \ldots, [n-1]_n\}$ (or $\{\overline{0}, \overline{1}, \ldots, \overline{n-1}\}$), and we add or multiply by using any representative of that class. It is easy to check that every axiom except for vii will hold for any n. But vii holds (that is, $\mathbb{Z}/n$ has multiplicative inverses for all non-zero elements) if and only if n is prime.

For n prime we have a field, and to make this clear (and also to be able to use n for block-length, as in Chapter 1) we shall write $\mathbb{F}_p$ instead of $\mathbb{Z}/n$. Once we have specified p, we can write simply $0, 1, 2, \ldots p-1$ rather than $[0]_p, [1]_p, \ldots, [p-1]_p$ (or $\overline{0}, \overline{1}, \ldots, \overline{p-1}$).

Are there finite fields with a non-prime number q of elements? The answer is yes if and only if q is a prime power, $q = p^r, r \in \mathbb{Z}_+$. But of course $\mathbb{F}_{3^2} = \mathbb{F}_9 \neq \mathbb{Z}/9$, because $\mathbb{Z}/9$ is not a field. We shall construct and use such non-prime fields $\mathbb{F}_q$ in Chapter 6.

The notation $\mathbb{F}_q$ is justified, because it can be shown that any two fields with the same number of elements are isomorphic. For general statements we shall call our field $\mathbb{F}_q$, but *until we reach Chapter 6 q will always be prime* (that is, $r = 1$). This allows us to keep 'p' for the symbol-error probability.

2.2 Finite Vector Spaces

Just as $\mathbb{R}^n$ is a vector space over $\mathbb{R}$, $\mathbb{F}_q^n = (\mathbb{F}_q)^n$ is a vector space over $\mathbb{F}_q$. Everything you have learned about vector spaces (and most of your ideas about $\mathbb{R}^n$) will still work. These notes gives only a quick, informal reminder of the main definitions and ideas from linear algebra which we shall use. Formal definitions, results and examples will be written in terms of finite fields, and spaces and codes over them. The main difference is that we can now count the vectors in a space: to start with, $|\mathbb{F}_q^n| = q^n$. We can even write a complete list of them.

We shall still sometimes call our vectors 'words', and some of them will be our codewords. Because words (in English) are horizontal, we will usually write vectors as rows (rather than columns): $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$, where the x_i are in $\mathbb{F}_q$. You need to know which field $\mathbb{F}_q$ you are working over, because all arithmetic must be done mod q.

Example 15. The vectors $\mathbf{x} = (0, 1, 2, 0)$ and $\mathbf{y} = (1, 1, 1, 1)$ could be vectors in $\mathbb{F}_3^4$, but they could also be vectors in $\mathbb{F}_7^4$. In $\mathbb{F}_3^4$, we would have $\mathbf{x} + \mathbf{y} = (1, 2, 0, 1)$, and $2\mathbf{x} = (0, 2, 1, 0)$. But in $\mathbb{F}_7^4$, it would be $\mathbf{x} + \mathbf{y} = (1, 2, 3, 1)$, $2\mathbf{x} = (0, 2, 4, 0)$ and $4\mathbf{x} = (0, 4, 1, 0)$. △

In Chapter 1, a code could be any subset of A^n. But we now make a more restrictive definition.

Definition 2.2. A **linear code** is a subspace of the vector space $\mathbb{F}_q^n$, for some finite field $\mathbb{F}_q$ and non-negative integer n.

Recall that a **subspace** of a vector space is a subset which is closed under vector addition and scalar multiplication. Thus, if we are given a subset of $\mathbb{F}_q^n$ as a list, it is straightforward (if tedious) to check whether it is a linear code or not.

Example 16. Let $\mathbf{x} = (0, 1, 2, 0)$, and $\mathbf{y} = (1, 1, 1, 1,)$, $\mathbf{z} = (0, 2, 1, 0)$ and $\mathbf{0} = (0, 0, 0, 0)$ be vectors in $\mathbb{F}_3^4$. Then $C_1 = \{\mathbf{x}, \mathbf{y}\}$ is not a linear code since $\mathbf{x} + \mathbf{y} = (1, 2, 0, 1) \notin C_1$. But $C_2 = \{\mathbf{x}, \mathbf{z}, \mathbf{0}\}$ is a linear code, as adding any combination of these vectors (which also includes multiplying them by 0, 1 or 2) gives one of them. △

If the subset S is not closed, we can keep adding in vectors as necessary until it is. The resulting space is the **span** of the set, written $\langle S \rangle$, and is by construction a linear code.

Example 17. The span of C_1 is the linear code $C_3 = \langle C_1 \rangle = \langle \{\mathbf{x}, \mathbf{y}\} \rangle =$

$\{(0, 0, 0, 0), (0, 1, 2, 0), (0, 2, 1, 0), (1, 1, 1, 1), (1, 2, 0, 1), (1, 0, 2, 1), (2, 2, 2, 2), (2, 0, 1, 2), (2, 1, 0, 2)\}$.

We could also notice that $C_2 = \{0\mathbf{x}, 1\mathbf{x}, 2\mathbf{x}\} = \langle \{\mathbf{x}\} \rangle$, so in fact $\langle C_2 \rangle = C_2$. △

If $\langle S \rangle = C$ then we say S is a **spanning set** for C. There are usually many possible spanning set for a subspace.

A set of vectors $S = \{\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_k}\}$ in $\mathbb{F}_q^n$ is **linearly independent** if and only if no non-trivial linear combination of them equals the zero-vector $\mathbf{0}$; that is, for $\lambda_i \in \mathbb{F}_q$, iff:

$$\lambda_1 \mathbf{x_1} + \lambda_2 \mathbf{x_2} + \ldots + \lambda_k \mathbf{x_k} = \mathbf{0} \implies \lambda_1 = \lambda_2 = \ldots = \lambda_k = 0.$$

A linearly independent spanning set for a linear code C is a **basis** for C. While there may still be many possible bases, these will all have the same number of vectors, and this number is the **dimension** of C, written $\dim(C)$.

Example 18. The spanning sets of the code listed above, $C_3 \subseteq \mathbb{F}_3^4$, include
$S_1 = \{(0, 1, 2, 0), (1, 1, 1, 1)\}$,
$S_2 = \{(0, 1, 2, 0), (2, 2, 2, 2)\}$, and
$S_3 = \{(0, 1, 2, 0), (1, 1, 1, 1), (2, 0, 1, 2)\}$.

S_3 is not a basis, because $1(0, 1, 2, 0) + 2(1, 1, 1, 1) + 2(2, 0, 1, 2) = (6, 3, 6, 6) = (0, 0, 0, 0)$. But both S_1 and S_2 are linearly independent sets, and thus bases for C_3. So $\dim(C_3) = 2$.

To prove the linear independence of S_1, we can note that if $\lambda_1(0, 1, 2, 0) + \lambda_2(1, 1, 1, 1) = (0, 0, 0, 0)$, then by the first position $\lambda_2 = 0$, and so then by the second position $\lambda_1 = 0$ also.

△

Example 19. For the whole space $\mathbb{F}_q^n$, the 'standard basis' is $B = \{\mathbf{e_1}, \mathbf{e_2}, \ldots \mathbf{e_k}\}$, where e_i has 1 in the i^{th} position and 0 elsewhere. △

A basis B for a linear code C is "just right" for making every vector $\mathbf{c} \in C$ as a linear combination of vectors from B: a spanning set can make each $\mathbf{c}$ *at least* one way, a linearly independent set can make each $\mathbf{c}$ *at most* one way, but a basis can make each $\mathbf{c}$ *exactly* one way. We use this property to prove the following:

Proposition 2.3. *If a linear code $C \subseteq \mathbb{F}_q^n$ has dimension k, then $|C| = q^k$.*

Proof. Let $B = \{\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_k}\}$ be any basis for C. There is a one-to-one correspondence between codewords $\mathbf{c} \in C$ and linear combinations $\lambda_1\mathbf{x_1} + \lambda_2\mathbf{x_2} + \ldots + \lambda_k\mathbf{x_k}$, with $\lambda_i \in \mathbb{F}_q$. Each λ_i can take any of q values. □

Example 20. For $C_3 \subseteq \mathbb{F}_3^4$ listed above, $\dim(C_3) = 2$, and $|C_3| = 9 = 3^2$. △

We can now update our (n, M, d) notation for the parameters of a code:

Definition 2.4. A q-ary $[n, k, d]$ code is a linear code, a subspace of $\mathbb{F}_q^n$ of dimension k with minimum distance d.

The square or round brackets prevent ambiguity: any q-ary $[n, k, d]$ code is also a q-ary (n, q^k, d) code, but not vice-versa. From now on, almost all codes will be linear, so I will write "code" for "linear code"

2.3 Array Decoding

The zero element 0 plays a very special role in $\mathbb{F}_q$, and so does the zero vector $\mathbf{0}$ in $\mathbb{F}_q^n$. We are also interested in how many entries of a general vector $\mathbf{x} \in \mathbb{F}_q^n$ are (or are not) zero.

Definition 2.5. For $\mathbf{x} \in \mathbb{F}_q^n$, the **weight** of $\mathbf{x}$, written $w(\mathbf{x})$, is the number of non-zero entries in $\mathbf{x}$.

Weights are closely related to Hamming distances. To show this, we must first define the difference between two vectors, using several properties of our vector space. Notice that by axioms vi and iii any field has a -1, the additive inverse of 1. (For q prime we could also write this as $q - 1$.) Then since we can multiply vectors by scalars, we can write $-\mathbf{y}$ for $-1 \cdot \mathbf{y}$, and $\mathbf{x} - \mathbf{y}$ for $\mathbf{x} + (-\mathbf{y})$.

Lemma 2.6. *For $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{F}_q^n$, we have $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y})$.*

Proof. The vector $\mathbf{x} - \mathbf{y}$ has non-zero entries exactly where $\mathbf{x}$ and $\mathbf{y}$ differ. □

So any Hamming distance can be written as a weight. But also, since $w(\mathbf{x}) = w(\mathbf{x} - \mathbf{0}) = d(\mathbf{x}, \mathbf{0})$, any weight can be written as a Hamming distance. This allows us to prove the following useful fact:

Proposition 2.7. *For the code $C \subseteq \mathbb{F}_q^n$, $d(C)$ is the minimum weight of any non-zero codeword in C.*

Proof. First we define two sets of non-negative integers:

$$W = \{w(\mathbf{x}) \mid \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\} \text{ and } D = \{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}.$$

Then the proposition says that $\min(D) = \min(W)$. We can show more: that $D = W$. For any $w(\mathbf{x}) \in W$, we know that both $\mathbf{x}$ and $\mathbf{0}$ are in C, so $w(\mathbf{x}) = d(\mathbf{x}, \mathbf{0}) \in D$. Conversely, for any $d(\mathbf{x}, \mathbf{y}) \in D$, we know $\mathbf{x}$ and $\mathbf{y}$ are both in C. Because C is a subspace, $\mathbf{x} - \mathbf{y}$ must also be in C, so $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y}) \in W$. □

This proposition means that to find $d(C)$ for a linear code with q^k words, we need to consider only q^k weights , rather than $\binom{q^k}{2} = \frac{q^k(q^k-1)}{2}$ distances.

Example 21. For code C_3 listed in Section 2.2, we can see that $d(C) = 2$, without finding $\binom{9}{2} = 36$ distances. But note that C_3 can also be written as $\langle\{(1,1,1,1),(1,2,0,1)\}\rangle$; $d(C)$ is not always obvious from a basis. △

For linear codes we have a much better way to talk about errors.

Definition 2.8. Suppose that a codeword $\mathbf{c} \in C \subseteq \mathbb{F}_q^n$ is sent, and $\mathbf{y} \in \mathbb{F}_q^n$ is received. Then the **error-vector** is $\mathbf{e} = \mathbf{y} - \mathbf{c}$.

We can think of the channel as adding $\mathbf{e}$ to $\mathbf{c}$, and the decoding process aims to subtract it again. But of course the receiver does not know which error-vector was added, and can only choose a *likely* error-vector to subtract. Which error-vectors are likely? We know that $d(\mathbf{y}, \mathbf{c}) = w(\mathbf{e})$; this is the number of symbol-errors which $\mathbf{c}$ has suffered. This allows us to re-write Propositions 1.9 and 1.11 for linear codes.

Proposition 2.9. *Let the code $C \subseteq \mathbb{F}_q^n$, be sent over a q-ary symmetric channel with symbol-error probability p. For any $\mathbf{c} \in C$, $\mathbf{y} \in \mathbb{F}_q^n$, and $\mathbf{e} = \mathbf{y} - \mathbf{c}$,*

$$P(\mathbf{y} \text{ received} \mid \mathbf{c} \text{ sent}) = P(\mathbf{e} \text{ added in channel}) = \left(\frac{p}{q-1}\right)^{w(\mathbf{e})} (1-p)^{n-w(\mathbf{e})}.$$

If also $p < (q-1)/q$, and each codeword $c \in C$ is equally likely to be sent, then for any given $\mathbf{y}$ $P(\mathbf{c}$ sent $\mid \mathbf{y}$ received) increases as $w(\mathbf{e})$ decreases.

The most likely error-vectors are those of least weight, or in other words those involving the fewest symbol-errors. So we should still use nearest-neighbour decoding: for a linear code, given a received word $\mathbf{y}$ we must find a codeword $\mathbf{c}$ such that $w(\mathbf{e}) = d(\mathbf{y}, \mathbf{c})$ is as small as possible; as before this will be one of the most likely codewords to have been sent. We could do this by calculating $\mathbf{e}_i = \mathbf{y} - \mathbf{c}_i$ for each $c_i \in C$, and then comparing all the $w(\mathbf{e}_i)$. But it is much more efficient to make an array, as follows. (This is sometimes called a 'Slepian' or 'Standard' array.)

Algorithm: Array Decoding
Let C be a code of dimension k in $\mathbb{F}_q^n$. We construct an array as follows:

1. Write the q^k codewords as the top row, with $\mathbf{0}$ in the first column.
2. Consider the vectors of $\mathbb{F}_q^n$ which are not yet in the array, and choose one of lowest available weight, $\mathbf{e}$.
3. Write $\mathbf{e}$ into the first column, and then complete this new row by adding $\mathbf{e}$ to each codeword in the top row.
4. If the array has q^{n-k} rows, then STOP. Otherwise, go to 2.

Now, decode any received word $\mathbf{y}$ to the codeword at the top of its column.

Example 22. Let C be the $[4, 2, 2]$ code $\langle (1, 1, 0, 0), (0, 0, 1, 1) \rangle \subseteq \mathbb{F}_2^4$. Then one possible array is:

$(0,0,0,0)$	$(1,1,0,0)$	$(0,0,1,1)$	$(1,1,1,1)$
$(1,0,0,0)$	$(0,1,0,0)$	$(1,0,1,1)$	$(0,1,1,1)$
$(0,0,1,0)$	$(1,1,1,0)$	$(0,0,0,1)$	$(1,1,0,1)$
$(1,0,1,0)$	$(0,1,1,0)$	$(1,0,0,1)$	$(0,1,0,1)$

This array decodes (0,0,0,1) to (0,0,1,1), and (0,1,1,0) to (1,1,0,0). In each case the word is decoded to a nearest neighbour, though this nearest neighbour is not unique. △

For this method to be well defined, we require that every possible vector in $\mathbb{F}_q^n$ appears exactly once in the array. (See Q22) We should also prove the following:

Proposition 2.10. *Array decoding is nearest-neighbour decoding.*

Proof. Suppose that our received word $\mathbf{y}$ is in the same row as $\mathbf{e}_1$ in the first column, and in the same column as codeword $\mathbf{c}_1$ in the top row. So we decode $\mathbf{y}$ to $\mathbf{c}_1$; that is, we assume that error-vector $\mathbf{e}_1$ was added in the channel, and now subtract it.

$\mathbf{0}$	$\mathbf{c}_1$
$\mathbf{e}_1$	$\mathbf{y}$

We can show by contradiction that $\mathbf{c}_1$ is a nearest neighbour of $\mathbf{y}$. Suppose not, that is, there is some $\mathbf{c}_2$ such that $\mathbf{y} = \mathbf{c}_2 + \mathbf{e}_2$, with $w(\mathbf{e}_2) < w(\mathbf{e}_1)$. Then we have $\mathbf{c}_2 + \mathbf{e}_2 = \mathbf{y} = \mathbf{c}_1 + \mathbf{e}_1$, so $\mathbf{e}_2 = \mathbf{e}_1 + (\mathbf{c}_1 - \mathbf{c}_2)$. Since $\mathbf{c}_1 - \mathbf{c}_2$ must be a codeword in the top row, $\mathbf{e}_2$ is in the same row as $\mathbf{e}_1$, so it is not in any row higher up. Thus when $\mathbf{e}_1$ was picked by step 2. of the algorithm, $\mathbf{e}_2$ was not yet in the array, it was available, so $\mathbf{e}_1$ did not have least possible weight. Contradiction. □

As we know, nearest-neighbour decoding does not always find the right codeword. If we use an array, what is the probability that, after transmission and array decoding, the receivers will have the correct word, the one that was sent? Effectively, decoding with an array subtracts one of the vectors $\mathbf{e}$ in the first column from the received word y. Thus decoding will be successful if and only if the channel added one of these vectors.

Proposition 2.11. *Let C be a code $\subseteq \mathbb{F}_q^n$, sent over a q-ary symmetric channel with symbol-error probability p. Suppose the decoding array has α_i vectors of weight i in its first column. Then for any codeword c sent, the chance that it is successfully decoded is*

$$\sum_{i=0}^{n} \alpha_i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i}.$$

Example 23. For the array above, with $q = 2$, we have $\alpha_0 = 1,\ \alpha_1 = 2,\ \alpha_2 = 1,\ \alpha_3 = 0,\ \alpha_4 = 0$. So the chance of successful decoding is $(1-p)^4 + 2p(1-p)^3 + p^2((1-p)^2$. $\triangle$

Proof. The chance of successful decoding is the chance that one of the error-vectors in the first column occurred; since these are disjoint possibilities, we add their individual probabilities. (We include the zero error-vector - that is, the possibility that the codeword is received correctly.) $\square$

Steps 1 and 2 of the algorithm involve choice, so that many different arrays could be made for the same code. In general, some of these arrays would decode some received words differently. However, for a perfect code, all arrays will perform identical decoding. These ideas are explored in more detail in the homework (Q25-27).