Introduction to Coding Theory
MA3202

Jürgen Bierbrauer

December 21, 2000

Contents

1 Bitstrings and binary operations 5

2 The Hamming distance 9

3 Binary codes 11

4 Error-correcting codes in general 15

5 The binary symmetric channel 17

6 The sphere-packing bound 19

7 Binary linear codes 23

8 Block coding 29

9 The effect of coding 33

10 Duality 35

11 Binary Hamming and Simplex codes 39

12 Principle of duality 43

13 Prime fields 47

14 Finite fields 51

15 General linear codes 55

16 Duality and orthogonal arrays **59**

17 The game of SETS **65**

18 Reed-Solomon codes **69**

19 Shortening and Puncturing **75**

20 Syndrome decoding **81**

21 Concatenation **85**

22 Universal hashing **89**

23 Cyclic codes **93**

24 Shannon entropy **97**

25 The Gilbert-Varshamov bound **101**

26 3-dimensional codes, projective planes **105**

27 The end of triviality **113**

28 The last chapter **115**

Chapter 1

Bitstrings and binary operations

The object of Coding Theory is the transmission of messages over noisy channels. Below is the standard picture visualizing the situation:

At first we need to understand what the elements of this picture mean: what is a **message,** a **channel,** what is **noise**? Along the way we will encounter more basic notions. In this first chapter some of these will be explained. We start with the message:

If there are 8 possible messages to be sent, say, then we can represent each message as a bitstring of length 3, like 000 or 011 or 110. We will generally

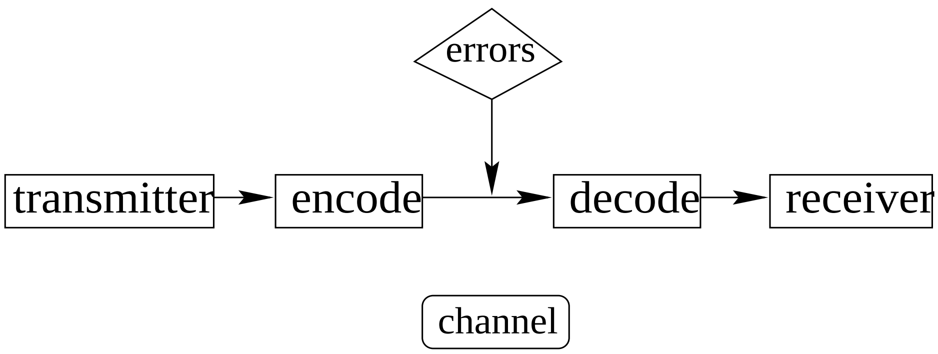

Figure 1.1: Information transmission over a noisy channel

assume this has been done and define a **message** to be a bitstring. So here are some examples of bitstrings and their lengths:

bit string	length
000	3
110	3
110011	6
0000011111	10

Assume now we have to send one of 8 messages (the bitstrings of length 3), for example message 011. If it should happen along the way (in the channel) that the second bit is flipped (the second coordinate is in error), then 001 will be received.

We want to express this situation in mathematical terminology.

1.1 Definition. $\mathbb{F}_2 = \{0, 1\}$ *with addition*

$$0 + 0 = 0,\ 0 + 1 = 1,\ 1 + 0 = 1,\ 1 + 1 = 0$$

Here the letter **F** stands for **field.** It indicates that this tiny little structure of only 2 elements has something in common with the field of real numbers and the complex number field: each of these structures satisfies the same set of axioms. Each of them is a field. We will come back to this topic later.

The idea behind the addition in $\mathbb{F}_2$ is to model the errors in information transmission: a coordinate of a message is in error if and only if 1 is added to the entry in this coordinate. This explains why we must have $1 + 1 = 0$: if a bit is flipped twice then no error occurs.

Addition in $\mathbb{F}_2$ is also known as XORing, in particular in the computer science literature. Here **XOR** stands for **exclusive or,** a logical operation, where 1 stands for **true** and 0 for **false.** The relation $1+1 = 0$ (true or true = false) is what makes it "exclusive": in order for the result to be true one of the two ingredients has to be true, but not both (the ordinary **or** operation would have $1 + 1 = 1$).

Another motivation for our binary addition comes from arithmetic: if we distinguish even and odd integers, the following familiar rules hold:

$$even+even = even,\ even+odd = odd,\ odd+even = odd,\ odd+odd = even.$$

With $even = 0$ and $odd = 1$ these are exactly the rules of binary addition.

Addition in $\mathbb{F}_2$ describes what happens in each coordinate. Calculation with bitstrings is formalized as follows:

1.2 Definition. *$\mathbb{F}_2^n$ consists of bitstrings $x = (x_1, x_2, \dots, x_n)$ of length n, where $x_i \in \mathbb{F}_2$. Addition in $\mathbb{F}_2^n$ is coordinatewise.*

For example, let $x = 001 \in \mathbb{F}_2^3$ and $y = 101 \in \mathbb{F}_2^3$. Then $x + y = 100$. With this terminology it is easier to express the relation between messages, errors and received messages. Let $x = 001100$ be the message sent. Assume errors occur in coordinates 2 and 4. We can express this by adding the **error vector** $e = 010100$. The received message will then be

$$y = x + e = 011000.$$

Let us do this the other way around: if $x = 010101$ was sent and $y = 010111$ was received, then an error occurred in coordinate 5. This means $e = 000010$.

Return to the situation where we send one of 8 messages (the elements of $\mathbb{F}_2^3$), for example $x = 011$. No matter what the error vector e is, the received message $y = x + e$ is again one of the legitimate messages. There is no way for the receiver to suspect that an error occurred, let alone to correct it. So how can we hope to correct errors? Here is the easiest of all error-correcting systems: encode each 0 of the message to be sent as a block 000 and analogously each 1 by 111. If the original message is $x = 011$, the encoded message is now $x' = 000111111$. The receiver knows what encoding scheme has been used. He will therefore divide the received message in blocks of length 3. If such a block is 000 or 111 then decoding is obvious: $000 \mapsto 0$, $111 \mapsto 1$. Assume a block of the received message is 101. The receiver knows that at least one transmission error must have happened. As a small number of errors is more probable than many errors the decoding will be $101 \mapsto 1$ (by majority decision).

The initial picture makes sense now. We have seen that messages can be encoded by the transmitter and decoded by the receiver such that the following holds: if not more than error occurs during transmission (in the channel), then the error will automatically be corrected. What we have used is known as the **repetition code** of length 3.

1. We learned how to calculate with bitstrings.
2. $\mathbb{F}_2^n$ consists of the bitstrings of length n.
3. The received message is the binary sum of the sent message and the error vector.
4. The repetition code of length 3 corrects one bit error.

Problems

1. Compute the sum of 11001 and 01110
2. Assume 000000 was sent and two errors occurred. List all possible received messages.
3. Let $x = 1101$ be the message to be sent. Encode x using the repetition code of length 3.
4. Assume the repetition code of length 3 is used and 000110111101 is the received encoded message. What is the result of the receiver's decoding?

Chapter 2

The Hamming distance

2.1 Definition. *Let* $x = (x_1, x_2, \ldots, x_n)$ *and* $y = (y_1, y_2, \ldots, y_n)$ *be bitstrings in* $\mathbb{F}_2^n$. *The* **distance** *(or* **Hamming distance***) between* x *and* y *is defined as*

$$d(x, y) = \textit{number of coordinates } i \textit{ where } x_i \neq y_i.$$

Here are some examples:

$$d(0000, 1111) = 4,\ d(00110, 00101) = 2,\ d(111111, 001100) = 4.$$

Expressed in the context of messages and errors we have that $d(x, y)$ is the minimum number of errors transforming x into y. In fact, consider the second example above: $x = 00110$ and $y = 00101$ differ in the two last coordinates. This means that $d(x, y) = 2$ and $x + 00011 = y$.

Things get even easier when we use the weight.

2.2 Definition. *The* **weight** $wt(x)$ *of the bitstring* $x \in \mathbb{F}_2^n$ *is the number of nonzero coordinates in* x.

Here are some examples:

$$wt(0000) = 0,\ wt(1111) = 4,\ wt(00110) = 2,\ wt(001101) = 3.$$

The weight of a bitstring is its distance from the all-0 bitstring. If the all-0 bitstring is sent and w errors occur during transmission, then the received message has weight w. If x is sent, e is the error vector and $y = x + e$ is received, then $d(x, y) = wt(e)$.

The Hamming distance is also called the Hamming metric. The general notion of a metric is generally used in mathematics. Here is the definition:

2.3 Definition. *Let X be a set. For every pair $x \in X, y \in X$ let a real number $d(x, y)$ be given (the* **distance***) from x to y). The function d is called a* **metric** *if the following are satisfied:*

- $d(x, y) \geq 0$ *for all* x, y.
- $d(y, x) = d(x, y)$ *for all* x, y.
- $d(x, y) = 0$ *if and only if* $x = y$.
- $d(x, z) \leq d(x, y) + d(y, z)$ *for all* x, y, z.

The last requirement is also the most important. It is known as the **triangle inequality.** A famous metric is the Euclidean metric in Euclidean space. If for example $x = (x_1, x_2)$ and $y = (y_1, y_2)$ are two points in the plane, then their Euclidean distance is $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

2.4 Theorem. *The Hamming distance is a metric on $\mathbb{F}_2^n$.*

Most of the properties of Definition 2.3 are obvious. Only the triangle inequality is a little interesting. This is left as an exercise.

1. The Hamming distance between two bitstrings of same length is the number of coordinates where they differ.
2. The Hamming distance is a metric
3. The weight is the distance from the all-0 vector.
4. $d(x, y) = wt(x + y)$.

Problems

1. Compute $d(11001, 01110)$ and $d(0000, 0110)$.
2. Find $wt(00110)$ and $wt(10111)$.
3. Prove that the Hamming distance is a metric
4. Assume x is sent and $y = x + e$ is received. What can we say about $d(x, y)$ and about $wt(e)$ if not more than 3 errors have occurred?

Chapter 3

Binary codes

We saw that no errors can be detected or corrected if all elements of $\mathbb{F}_2^n$ are used as messages. The obvious idea is to use only a certain subset. Such subsets will be called codes.

Let us send bitstrings of length 6. Instead of using all elements of $\mathbb{F}_2^6$ as (encoded) messages, we use only the following subset:

$$\boxed{\begin{array}{c} 000000 \\ 100110 \\ 010101 \\ 110011 \\ 001011 \\ 101101 \\ 011110 \\ 111000 \end{array}}$$

Such a subset of bitstrings of length 6 is also called a **binary code** of length 6. Its elements are **code words.** In our example we have 8 codewords.

The most important property of this code is the following: any two different codewords are at distance ≥ 3. We say that 3 is the **minimum distance** of the code. Please check that for yourself. The parameters of this binary code are then recorded as $(6, 8, 3)_2$: we have a binary code (the subscript 2 indicates that), of length 6, consisting of 8 codewords, with minimum distance of 3.

The idea of error-correction is the following: transmitter and receiver agree on the code to be used. Only codewords will be sent. If only one error

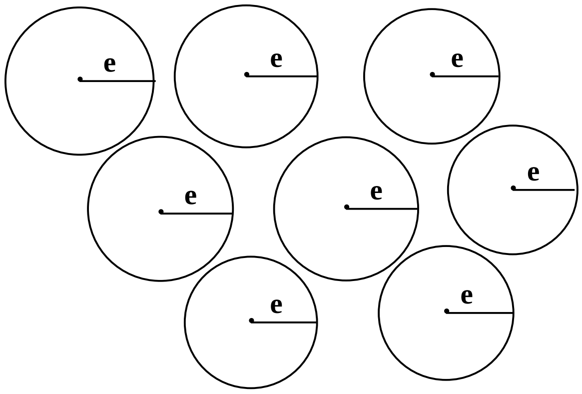

Figure 3.1: Non-overlapping balls centered at codewords

occurs in the channel, then the received word will be in a ball of radius 1 around the codewords. These balls of radius 1 do not overlap: if a bitstring has distance 1 from some codeword, then it has larger distance from any other codeword. In other words, the receiver will decode any vector at distance ≤ 1 from some codeword as that codeword. In the picture: the whole ball of radius 1 is decoded as the center of the ball. If not more than 1 error occurred, then this error will be corrected.

3.1 Definition. *A binary code* $(n, M, d)_2$ *is a set of* M *bitstrings in* $\mathbb{F}_2^n$ *such that any two different elements of the code (codewords) are at distance* $\geq d$. *We call* n *the* **length** *and* d *the* **minimum distance** *of the code.*

A code of minimum distance 5 can correct 2 errors, minimum distance 7 can correct 3 errors and so on. A fundamental problem of coding theory is the following: Given n and d, find the largest M such that there is a code $(n, M, d)_2$.

1. A code $(n, M, d)_2$ is a collection of M bitstrings of length n such that any two different of these codewords are at Hamming distance at least d.

2. A code can correct e errors provided its minimum distance is $d \geq 2e + 1$.

3. We saw a code $(6, 8, 3)_2$.

Problems

1. If we want to correct 8 bit errors, what would the minimum distance of the code have to be?

2. Using our code $(6, 8, 3)_2$, decode the following received vectors:

$$111100, \ 111011, \ 000001, \ 011110.$$

3. Does a code $(5, 6, 3)_2$ exist?

Chapter 4

Error-correcting codes in general

The notion of a **binary** code is too narrow, although it is most frequently used in information transmission. Here is the general concept of an (error-correcting) code:

4.1 Definition. *Let $\mathcal{A}$ be a finite set of q elements (the* **alphabet***). A q-ary code $\mathcal{C}$ of* **length** *n is a family of n-tuples with entries in $\mathcal{A}$:*

$$\mathcal{C} \subseteq \mathcal{A}^n.$$

For example, let $q = 3$ and $\mathcal{A} = \{0, 1, 2\}$. Then $\mathcal{A}^4$ consists of the 3^4 tuples of length 4 with entries $0, 1, 2$, like for example

$$0000, \ 0102 \ 2221 \text{ or } 2100.$$

A $3-$ary code (also called **ternary**) of length 4 consists of a collection of such ternary 4-tuples.

4.2 Definition. *Let $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ be bitstrings in $\mathcal{A}^n$. The* **distance** *(or* **Hamming distance***) between x and y is defined as*

$$d(x, y) = \textit{number of coordinates } i \textit{ where } x_i \neq y_i.$$

This is the same as in Definition 2.1. As before the minimum distance of a code is the minimum of the distances between different code words.

4.3 Definition. *A q-ary code $(n, M, d)_q$ is a set of M strings in $\mathcal{A}^n$ (where $|\mathcal{A}| = q$) such that any two different elements of the code (codewords) are at distance $\geq d$. We call n the* **length** *and d the* **minimum distance** *of the code.*

Here are the words of a ternary code $(4, 27, 2)_3$:

0000	1002	2001	0102	1101	2100	0201	1200	2202
0012	1011	2010	0111	1110	2112	0210	1212	2211
0021	1020	2022	0120	1122	2121	0222	1221	2220

As in the binary case, in order to correct e errors we need a code of minimum distance at least $2e + 1$. Our code $(4, 27, 2)_3$ will not suffice to correct one error. For example, if 0000 was sent and 0010 received (only 1 error occurred), the received vector has distance 1 not only from 0000 but also from 0012 and from 0210 and from 2010.

A basic problem of coding theory is the following: given q, n, d, find the maximum number M such that a code $(n, M, d)_q$ exists.

1. A q-ary code $(n, M, d)_q$ is a collection of M q-ary strings of length n such that any two different of these codewords are at Hamming distance at least d.
2. A code can correct e errors provided its minimum distance is $d \geq 2e + 1$.
3. We saw a code $(4, 27, 2)_3$.

Problems

1. Find the smallest length n such that an $(n, 27, 2)_3$ exists.
2. Prove the following: if there is an $(n, M, d)_q$, then there is an $(n + 1, M, d)_q$.
3. Prove: If there is an $(n, M, d)_{q-1}$, then there is an $(n, M, d)_q$.

Chapter 5

The binary symmetric channel

So how do we model the **noise** mentioned in Chapter 1? In the case of the binary alphabet the easiest and most widely used model is the binary symmetric channel (BSC). It assumes that there is a certain fixed probability, a number p, where $0 < p < 1$ for a bit transmission error. Clearly a value $p > 1/2$ does not make sense (why?) and $p = 1/2$ would mean that pure noise is received. We can therefore assume $p < 1/2$, and for all practical purposes p will be rather small.

The probability that a sent 0 will be received as 0 is $1 - p$. Likewise the probability that a sent 1 will be received as 1 as $1-p$ as well. The probability that a sent 0 is reveived as a 1 is p, just as the probability that a sent 1 is received as 0.

This model of a channel is called **symmetric** because 0 and 1 play symmetric roles. It is also called a **memoryless** channel because the probability of a bit error is independent of the prehistory. Can you think of situations when this model will not be appropriate?

Let us do some combinatorial counting, based on the BSC: assume a word (a bitstring) of length n is sent through the channel. The probability that no bit error occurs is $(1 - p)^n$. The probability that exactly one error occurs is $np(1-p)^{n-1}$ (there are n choices where the error could happen; the probability of a certain error pattern involving just 1 error is $p(1-p)^{n-1}$. The probability of certain error pattern with precisely k errors is $p^k(1 - p)^{n-k}$. The number of such error patterns is the number of possibilities of choosing k elements out of n. This is $\binom{n}{k}$. The probability that precisely k errors

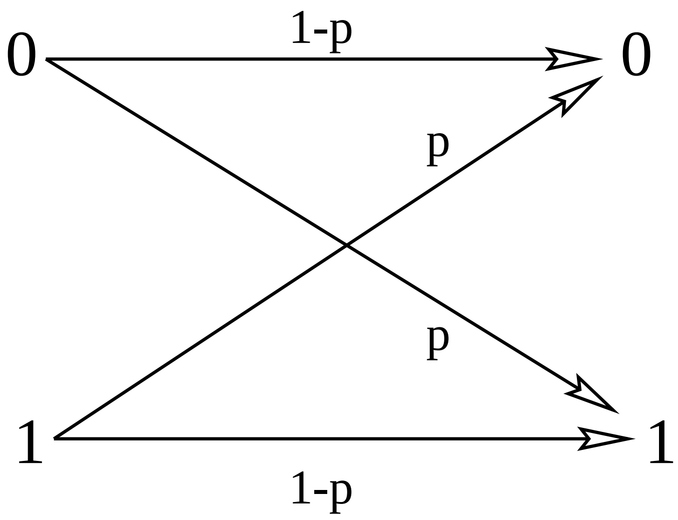

Figure 5.1: The BSC

occur is therefore $\binom{n}{k} p^k (1-p)^{n-k}$. If we want to compute the probability that **at most** k errors happen, we have to sum up these probabilities: the probability that not more than k bit errors occur is $\sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$.

1. The binary symmetric channel is the simplest model for noise.
2. If a bitstring of length n is sent via the BSC, the probability that at most k bit errors occur is $\sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$.

Problems

1. Compute the probability that no more than 1 error occurs in the transmission of a bitstring of length 10, when the bit error probability is $p = 10^{-3}$, or $p = 10^{-4}$, or $p = 10^{-5}$.

Chapter 6

The sphere-packing bound

As all mathematical optimization problems, the problems of finding the maximum number M such that a code $(n, M, d)_q$ exists, splits in two parts: we have to construct good codes (with high M), and we have to prove upper bounds showing that a higher value of M is impossible. In this chapter we will prove our first upper bound. It is a direct consequence of the basic idea of error correction.

As a preparation we need a counting argument (this is a typical problem of elementary combinatorics), of the same type as the argument from Chapter 5. Given a vector $x \in \mathcal{A}^n$ (recall that $\mathcal{A}$ is our alphabet of size q), how many vectors are there of distance $\leq i$ from x? We call these vectors the **ball of radius** i with center x.

6.1 Definition. *Consider the space $\mathcal{A}^n$ of vectors of length n, with the Hamming metric. This is also called a* **Hamming space.** *The number of vectors at distance $\leq i$ from a given vector is denoted $V_q(i, n)$. We call $V_q(i, n)$ the* **volume** *of a ball of radius i.*

Fix some distance j and count the vectors at distance precisely j from the given vector. There are $\binom{n}{j}$ choices for the set of coordinates where the entries are different. Once this set is fixed, there are $q-1$ possibilities for the possible entries in each of these j coordinates. We count $\binom{n}{j}(q-1)^j$. The volume $V_q(i, n)$ is obtained by adding up these numbers, for $j \leq i$. We have seen the following:

6.2 Proposition. *The volume of the ball of radius i is*

$$V_q(i,n) = \sum_{j=0}^{i} \binom{n}{j} (q-1)^j.$$

We are close to our bound. Let $\mathcal{C}$ be a code $(n,M,d)_q$. Let $e = \lfloor (d-1)/2 \rfloor$ (the largest integer less or equal $(d-1)/2$). We have chosen e such that $2e+1 < d$. Our standard argument shows that the balls of radius e centered at the codewords must be disjoint. As each such ball has $V_q(e,n)$ vectors, we must have that $MV_q(e,n) \leq q^n$ (counting all vectors in these ball we cannot get more vectors than the whole space contains). We have our bound.

6.3 Theorem (sphere packing bound). *Each q-ary code $\mathcal{C}$ of length n and minimum distance d satisfies*

$$|\mathcal{C}| \leq \frac{q^n}{V_v(e,n)}.$$

Here $e = \lfloor (d-1)/2 \rfloor$.

Codes for which equality holds in Theorem 6.3 are known as **perfect codes.** The parameters of perfect codes have all been classified.

Our first code was a $(6,8,3)_2$. What is the maximum M for a code $(6,M,3)_2$? Here $e = 1$. We have $V_2(1,6) = 7$. The sphere packing bound says $7M \leq 64$, hence $M \leq 9$. We claim that a code $(6,9,3)_2$ cannot exist. The reader is asked to provide a proof along the lines sketched in the Problems section.

Here is a nice parameter situation: consider a possible code $(11,729,5)_3$. Observe that $729 = 3^6$. We have $e = 2$ and $V_3(2,11) = 1 + 11 \cdot 2 + \binom{11}{2} \cdot 4 = 1 + 22 + 220 = 243 = 3^5$ (quite a coincidence). It follows that a ternary code $(11,M,5)_3$ has $M \leq 3^{11}/3^5 = 3^6 = 729$. We see that the parameters $(11,3^6,5)_3$ are extremal. If such a code exists, then it is perfect. A perfect code $(11,729,5)_3$ does indeed exist. It is uniquely determined by its parameters and known as the **ternary Golay code,** named after a Swiss engineer who described it in 1949.

Here is how this code can be used in a betting system: in most European countries it is popular to bet on the results of football matches (**football** is not to be confused with American football. In this country the game is known by the bizarre name of **soccer**). Each match has one of three possible results: 1= home team wins, 2=guest wins, or 0= a draw. This is why we use ternary codes. There are 11 or more matches in the pool. If there are 11 matches we can use the 729 codewords of the ternary Golay code as bets. As every vector is at distance ≤ 2 from a codeword, it is guaranteed that no matter what the results of the 11 matches are one of our bets has at least 9 results correct. Assume there are 13 matches in the pool. Then we may choose two matches whose results seem to be safe and use the Golay code for the remaining 11 matches.

Incidentally, the ternary Golay code is older than Golay. The Finnish journalist Virttakallio published it shortly after the war in a Finnish football journal. It is no coincidence that this discovery was made in Finland. This country has a rich tradition in coding theory and related questions.

1. The **sphere packing bound** is our first general bound on codes.
2. Codes meeting it with equality are called **perfect**
3. The **ternary Golay code** $(11, 729, 5)_3$ is perfect.

Problems

1. Construct a code $(4, 2, 3)_2$ (this is really trivial).
2. Show that there is no $(4, 3, 3)_2$.
3. Construct a code $(5, 4, 3)_2$ (hint: use our code $(6, 8, 3)_2$)).
4. Show that there is no $(5, 5, 3)_2$.
5. Using the preceding problem, show that there is no $(6, 9, 3)_2$.
6. What does the sphere-packing bound tell us about the length n of a binary code $(n, 2^7, 5)_2$?

Chapter 7

Binary linear codes

Our binary code $(6, 8, 3)_2$ has an additional structure, which greatly simplifies its description. Consider the following three of its codewords:

$$\boxed{\begin{array}{c}100110\\010101\\001011\end{array}}$$

Call them z_1, z_2, z_3. Consider all **linear combinations** of z_1, z_2 and z_3, that is all vectors of the form $\lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3$, where $\lambda_i \in \mathbb{F}_2$. These linear combinations are different and they are just exactly the words of our code. We call $\{z_1, z_2, z_3\}$ a **basis** of our code and say that our code is **linear.** Another way of seeing this is by the following observation: the sum of any two codewords is a codeword again: the code is closed under sums. We take this as a definition:

7.1 Definition. *A binary code is* **linear** *if it is closed under addition.*

It is a basic fact from linear algebra that each linear code (abstractly: each linear space) has a basis. Linear algebra applies to arbitrary fields. Most people are familiar with fields like the rational numbers, the real numbers and the complex numbers, but we can apply the basics of linear algebra to finite fields like $\mathbb{F}_2$ as well. This leads to the following basic fact:

7.2 Theorem. *A binary linear code $(n, M, d)_2$ has $M = 2^k$ for some k. The number k is the* **dimension** *of the code. There is a* **basis** *$\{z_1, z_2, \dots, z_k\}$ of k codewords. This means that each codeword is a linear combination of the z_i.*

There is a general tendency to restrict attention to linear codes. One reason is that these are much easier to describe and to work with than codes in general. For example, a binary linear code of dimension k has $M = 2^k$ codewords, but it is uniquely described by a basis, which has only k elements. This is a much more compact representation of the code than a list of all its words.

7.3 Definition. *The parameters of a k-dimensional binary linear code of length n and minimum distance d are written*

$$[n, k, d]_2$$

(recall $M = 2^k$).

The basic problem of binary linear codes can be described as follows: determine the maximum k such that a code $[n, k, d]_2$ exists. Our code $(6, 8, 3)_2$ is a linear code. Its parameters are therefore $[6, 3, 3]_2$. The compact representation described above leads to the notion of a generator matrix.

7.4 Definition. *Let $\mathcal{C}$ be a linear code $[n, k, d]_2$. A* **generator matrix** *G of $\mathcal{C}$ is a (k, n)-matrix whose rows form a basis of $\mathcal{C}$.*

If we know a generator matrix G, then we know the code. The codewords are just all linear combinations of the rows of G, in other words: the code is the rowspace of G.

Here is another binary linear code. Please check it is linear. The minimum distance is 3.

0000000	1100110
1101000	0100101
1010100	1000011
0110010	0001110
1110001	1001101
0111100	0101011
1011010	0010111
0011001	1111111

As there are $16 = 2^4$ codewords, the dimension is $k = 4$. The parameters are therefore $[7, 4, 3]_2$. It is known as the **binary Hamming code.** How can we be sure the minimum distance is really 3? We would have to check $\binom{16}{2} = 120$ pairs of codewords. Here is a simplification, valid for linear codes:

7.5 Proposition. *For a binary linear code, the minimum distance equals the minimum of the weights of nonzero codewords (observe that the all-*0 *word is automatically contained in each linear code).*

Proof. We know that each weight also is a distance. It suffices to show that the distance between two codewords also is the weight of some codeword. Let x, y be different codewords at distance $d(x, y)$. Addition of the some vector to both words does not change the distance. We add x. This yields $d(x, y) = d(0, y + x) = wt(y + x)$. As the code is linear, $x + y$ is a codeword again. As $x \neq y$, we have $x + y \neq 0$. □

Because of Proposition 7.5 it suffices to check that each of the 15 nonzero codewords has weight ≥ 3 to check that indeed $d = 3$. Here is a generator matrix:

1	0	0	1	1	0	1
0	1	0	1	0	1	1
0	0	1	0	1	1	1
0	0	0	1	1	1	0

How can we be sure? At first we make sure that every row of the matrix is a nonzero codeword. One method of controlling that we have a generator matrix is to make sure that each codeword is indeed a linear combination of rows, equivalently that different linear combinations of rows yield different codewords. We can speed up by using basic facts of linear algebra. One such fact is the following: we have a generator matrix if and only if the matrix has rank equal the number of rows ($= k$), if and only if there is a (k, k)-submatrix with nonzero determinant. In our case the last check is fastest: the first four columns form a triangular matrix with ones on the diagonal, hence of determinant $= 1$. We repeat:

7.6 Proposition. *Let $\mathcal{C}$ be a binary $[n, k]_2$-code (the minimum distance is irrelevant here). Let G be a matrix whose rows are codewords of $\mathcal{C}$. The following are equivalent:*

1. *G is a generator matrix.*

2. *G has rank k (remember that the* **rank** *of a matrix is a basic term from linear algebra).*

3. *There is a (k,k)-submatrix of nonzero determinant (recall that the* **determinant** *of a matrix is another basic notion from linear algebra).*

4. *The rows of G are* **linearly independent** *(yet another such basic notion).*

> 1. A binary code is **linear** (or a **subspace** of $\mathbb{F}_2^n$) if it is closed under addition.
> 2. A binary linear code (subspace) has 2^k codewords, k is the **dimension**.
> 3. The parameters of binary linear codes: $[n,k,d]_2$.
> 4. Each binary linear code has a **basis,** consisting of k codewords.
> 5. A (k,n)-matrix whose rows from a basis is a **generator matrix.**

Problems

1. Show that our code $[7,4,3]_2$ is perfect.

2. Try to decide if an $[8,4,4]_2$ exists.

3. Give an example showing that the basis of a code is not uniquely determined.

4. Determine the parameters of the binary linear code generated by the rows of the matrix

1	0	0	1	1	0	1
0	1	0	1	0	1	1
0	0	1	0	1	1	1

5. Compute the parameters $[n,k,d]_2$ of the binary linear code generated by

	1	1	0	0	0	0	1	1	0	1
G=	0	0	1	1	0	0	1	0	1	1
	0	0	0	0	1	1	0	1	1	1

Find a nonzero codeword of minimum weight.

Chapter 8

Block coding

Binary linear codes are particularly handy for block coding. We use a code $\mathcal{C}$ with parameters $[n,k,d]_2$. As there are 2^k codewords we can find a bijective function (the encoding function) $\alpha : \mathbb{F}_2^k \longrightarrow \mathcal{C}$. There is no reason not to choose this function as a linear function ($\alpha(x+y) = \alpha(x) + \alpha(y)$). Here is a possible encoding function for our code $[7,4,3]$:

0000	$\longrightarrow$	0000000
1000	$\longrightarrow$	1101000
0100	$\longrightarrow$	1010100
0010	$\longrightarrow$	0110010
0001	$\longrightarrow$	1110001
1100	$\longrightarrow$	0111100
1010	$\longrightarrow$	1011010
1001	$\longrightarrow$	0011001
0110	$\longrightarrow$	1100110
0101	$\longrightarrow$	0100101
0011	$\longrightarrow$	1000011
1110	$\longrightarrow$	0001110
1101	$\longrightarrow$	1001101
1011	$\longrightarrow$	0101011
0111	$\longrightarrow$	0010111
1111	$\longrightarrow$	1111111

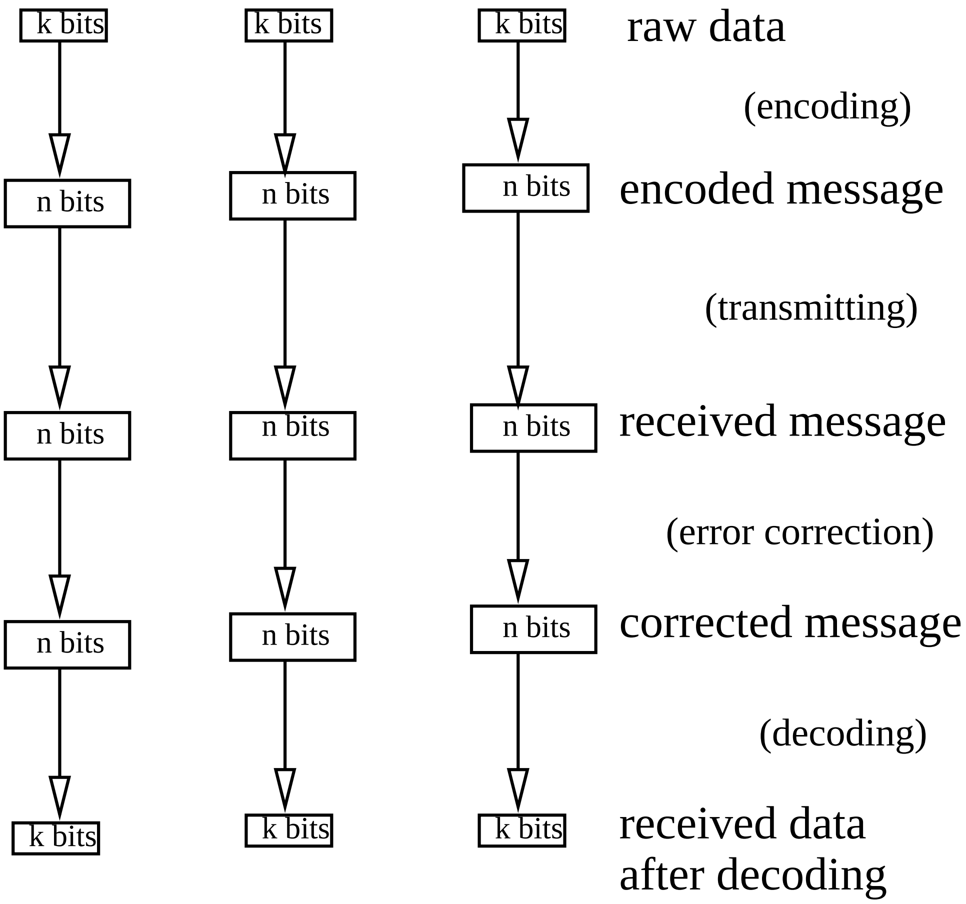

Figure 8.1: Block coding

We can describe how block coding works: the incoming data is divided into consecutive blocks of length k. Each such block is therefore a bitstring of length k, an element of $\mathbb{F}_2^k$. Apply the encoding function α to each block. The result is an encoded block of length n. These are transmitted via the channel. Error correction (recall our basic coding idea: $e = \lfloor (d-1)/2 \rfloor$ bit errors per encoded block can be corrected) produces blocks of length n, which hopefully are the same as the blocks fed into the channel. Finally, the inverse α^{-1} of the encoding function is applied to each block of length n. The result is a block of length k, which hopefully is identical to the corresponding block of raw data. The last two stages of the procedure (error correction and decoding) are also collectively known as decoding.

We understand the repetition code of length 3 from Chapter 1 as a part of the general picture: the code has parameters $[3, 1, 3]_2$ and consists of the codewords 000 and 111. The encoding function maps $0 \mapsto 000, 1 \mapsto 111$. This is about as elementary as it gets.

In block coding the length of bitstrings is increased from k to n. That means that k/n of the bits sent through the channel represent pure information, whereas the rest is redundancy that we have cleverly introduced for the purposes of error correction. We want to keep that ratio k/n as large as possible in order not to waste time and money.

8.1 Definition. *The ratio $R = k/n$ of a binary linear code $[n, k, d]_2$ is its* **information rate,** *the ratio d/n is the* **relative distance.**

Problems

1. Follow through all stages of block coding for the input string 00101110 when transmission errors occur in coordinates 3,7 and 14.

Chapter 9

The effect of coding

Let us compare the error probabilities with and without coding in a special case, based on the BSC (Chapter 5). The bit error probability is a (small) number p. Use block coding with blocks of length $k = 4$. We want to compute (or bound) the probability P that a given 4-tuple is **not** received correctly after coding and decoding.

Without coding ($n = k = 4$) the probability of correct transmission is $(1-p)^4 = 1 - 4p + 6p^2 - 4p^3 + p^4 \approx 1 - 4p$, so $P \approx 4p$.

Using our code $[7,4,3]_2$ the message will be correctly decoded if not more than 1 bit error occurs. The probability of correct transmission is therefore

$$\geq (1-p)^7 + 7p(1-p)^6 \approx (1 - 7p + 21p^2) + (7p - 42p^2) = 1 - 21p^2.$$

In this case we have $P \leq 21p^2$. We pay for this smaller error probability with an information rate $R = 4/7$ (meaning that 3/7 of transmission time is redundant).

Compare this to our very first code, the repetition code of length 3. We leave it as an exercise to check that, seen as a block code with $k = 4, n = 12$, its error probability is $P \leq 12p^2$. This is a decrease by about a factor 2 with respect to the $[7,4,3]_2$, but the information rate has gone down to 1/3.

There is a code $[11,4,5]_2$. If we use it, the probability of correct transmission is

$$(1-p)^{11} + 11p(1-p)^{10} + \binom{11}{2} p^2(1-p)^9$$

$$\approx (1 - 11p + 55p^2 - \binom{11}{3} p^3) + (11p - 110p^2 + 495p^3) + (55p^2 - 495p^3)$$

$$\approx 1 - \binom{11}{3} p^3.$$

The error probability is $P \leq 165p^3$.

Let $p = 10^{-4}$. Without coding the block error probability is $P \approx 4 \cdot 10^{-4}$. With an information rate of $R = 4/7$ we reach $P \leq 2.1 \cdot 10^{-7}$, and with an information rate of $R = 4/11$ this can be pressed down to $P \leq 1.65 \cdot 10^{-10}$. The repetition code is suddenly looking rather stupid in comparison.

Problems

1. Compute the block error probability of the repetition code of length 3, seen as a code encoding blocks of length 4 into messages of length 12.

2. Compute information rate and block error probability for a code $[9, 5, 3]_2$ (it exists).

3. Compute information rate and block error probability for a code $[23, 12, 7]_2$ (the **binary Golay code**).

Chapter 10

Duality

As is often the case in mathematics, duality is a great benefit also for coding theory. Everybody knows the dot product from physics. We can use it in our binary context as well.

10.1 Definition. *The* **dot product** *is defined on* $\mathbb{F}_2^n$ *by*

$$(x_1, x_2, \ldots, x_n) \cdot (y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} x_i y_i$$

Let $x, y \in \mathbb{F}_2^n$. We say that x and y are **orthogonal** ($x \perp y$) if $x \cdot y = 0$. It is obvious that for every subset $U \subseteq \mathbb{F}_q^n$ the set

$$U^\perp = \{x \mid x \in \mathbb{F}_2^n, x \cdot u = 0 \text{ for all } u \in U\}$$

is a subspace of $\mathbb{F}_2^n$. The following rules hold for all $x, y, z \in \mathbb{F}_2^n$ and $\lambda \in \mathbb{F}_2$:

$$\begin{array}{rcl} x \cdot y & = & y \cdot x \\ (x+y) \cdot z & = & x \cdot z + y \cdot z \\ (\lambda x) \cdot y & = & x \cdot (\lambda y) = \lambda(x \cdot y). \end{array}$$

The dot-product is an example of what are called **symmetric bilinear forms** in general (another topic of linear algebra). Here is a basic fact:

10.2 Lemma. *Let* $U \subseteq \mathbb{F}_2^n$ *be a linear subspace. Then*

$$dim(U) + dim(U^\perp) = n.$$

In particular we see that when $\mathcal{C}$ is a linear k-dimensional binary code of length n, then the **dual code** $\mathcal{C}^\perp$ has dimension $n-k$.

10.3 Lemma. *Let $U \subseteq \mathbb{F}_q^n$ be a linear subspace. Then*

$$U^{\perp\perp} = U$$

Proof. By definition $U^{\perp\perp} \supseteq U$. It follows from Lemma 10.2 that these spaces have the same dimension. □

Consider the $\mathbf{0} = 000\ldots$ (the 0-vector) and $\mathbf{1} = 111\ldots$ the all-1-vector. We have that $U = \{\mathbf{0}, \mathbf{1}\}$ is a 1-dimensional subspace of $\mathbb{F}_2^n$. This is the **repetition code.** Its dual $U^\perp$ will therefore have dimension $n-1$. What is this dual? The whole space is orthogonal to the 0-word. The vector $(x_1, x_2, \ldots, x_n)$ is orthogonal to the all-1-vector if

$$x_1 + x_2 + \cdots + x_n = 0.$$

10.4 Definition. *The binary* **sum zero code** *of length n consists of all binary n-tuples whose entries sum to 0. It is a linear code $[n, n-1, 2]_2$. Its dual is the repetition code.*

Please check that the minimum distance is really 2 (equivalently: it is impossible that two binary vectors at distance 1 both have sum 0).

Is it possible that a binary space is orthogonal to itself ($U \subseteq U^\perp$)? Yes, of course. For example, a bitstring is orthogonal to itself if its weight is even (see the Problems section). This is very different from real spaces, where a nonzero vector can never be orthogonal to itself.

Whenever we have a code $[n, k, d]_2$, we can consider its dual. The dimension will be $n-k$, but it is unclear what the minimum distance is going to be. As an example consider our code $[7, 4, 3]_2$ (the binary Hamming code $\mathcal{H}$ from Chapter 7). Consider the first three rows of the generator matrix G given in that chapter. They are orthogonal to all rows of G (check that). It follows that these three words generate a 3-dimensional code, which is orthogonal to $\mathcal{H}$. As $\mathcal{H}$ has length 7 and dimension 4, $\mathcal{H}^\perp$ has dimension 3. It follows that the first 3 rows of $\mathcal{H}$ form a generator matrix of $\mathcal{H}^\perp$. In particular $\mathcal{H}^\perp \subset \mathcal{H}$. In the Problems section of Chapter 7 it was shown that $\mathcal{H}^\perp$ is a code $[7, 3, 4]_2$.

10.5 Proposition. *Let $\mathcal{H}$ be the binary Hamming code $[7, 4, 3]_2$.*
Then $\mathcal{H}^\perp \subset \mathcal{H}$ is a code $[7, 3, 4]_2$.

Duality gives us another method to describe a code. Assume a generator matrix H of $\mathcal{C}^\perp$ is given. Then I have a fast method to decide if a bitstring $x \in \mathbb{F}_2^n$ belongs to $\mathcal{C}$. This will be the case if and only if x is orthogonal to all the rows of H. In many instances this description will be handier than the description via a generator matrix.

10.6 Definition. *A generator matrix of the dual code $\mathcal{C}^\perp$ is called a* **check matrix** *for $\mathcal{C}$.*

Here is a little trick (motivated by Gauß elimination from linear algebra) which often simplifies the determination of the check matrix.

10.7 Lemma. *Let a generator matrix for a code $[n,k]_2$ be written in the form $G = (I|P)$. Then a check matrix is $H = (P^T|I)$. Here I in G is the (k,k)-unit matrix, the I in H is the $(n-k, n-k)$-unit matrix and P^T is the transpose of P.*

Proof. We have to show that each row of G is orthogonal to each row of H. How can we express this in matrix notation? Recall the **matrix product:** the entry in row i and column j of the product matrix AB is the dot product of row i of A and column j of B. This means we have to show that $GH^T = 0$ (the 0-matrix). In fact

$$GH^T = (I|P)(P^T|I)^T = P + P = 0.$$

□

Consider our code $[6,3,3]_2$ from Chapter 3. It has a generator matrix $G = \left(\begin{array}{c|c} 100 & 110 \\ 010 & 101 \\ 001 & 011 \end{array}\right)$. This has the form of Lemma 10.7. We obtain as check matrix $H = \left(\begin{array}{c|c} 110 & 100 \\ 101 & 010 \\ 011 & 001 \end{array}\right)$. Observe that the submatrix P of G has to be transposed to obtain H. In our example P is symmetric ($P = P^T$).

1. Duality is based on the **dot product**
2. The dual of an $[n, k]_2$-code has dimension $n - k$.
3. A **check matrix** H of $\mathcal{C}$ is a generator matrix of the dual $\mathcal{C}^\perp$.
4. The sum-zero code $[n, n-1, 2]_2$ is the dual of the repetition code $[n, 1, n]_2$.

Problems

1. When is the all-1-word orthogonal to itself?
2. A code is **self-dual** if it equals its dual. Is there a self-dual $[6, 3, 3]_2$?
3. Find an $(8, 4)$-matrix in standard form (starting with the unit matrix I) which generates a self-dual code ($\mathcal{C}^\perp = \mathcal{C}$) with parameters $[8, 4, 4]_2$.

Chapter 11

Binary Hamming and Simplex codes

We learned how to describe binary linear codes using either a generator matrix or a check matrix. It is natural to concentrate on the **rows** of these matrices (they are codewords of the code to be described or of its dual). However, it will turn out very profitable to change the point of view and read these matrices **columnwise.**

For every r, let M_r be a matrix whose columns run through all nonzero bitstrings of length r. Clearly M_r is an $(r, 2^r - 1)$-matrix. The order in which the columns are written does not really matter. We will write them in order of increasing weight, and for each weight we order the columns lexicographically:

$$M_3 = \begin{pmatrix} 1001101 \\ 0101011 \\ 0010111 \end{pmatrix}.$$

$$M_4 = \begin{pmatrix} 100011100011101 \\ 010010011011011 \\ 001001010110111 \\ 000100101101111 \end{pmatrix}.$$

We call the code generated by M_r the **Simplex code** $\mathcal{S}_r(2)$. As M_r starts with the unit matrix, it has rank r. The Simplex code has therefore dimension r. How about the minimum distance? Each row of M_r has weight 2^{r-1}. Imagine we write the all-0 vector as last column. It is then clear that each combination of rows of M_r (hence each nonzero codeword of the Simplex code) has equally many zeroes and ones (2^{r-1} each). It follows that each nonzero codeword of the Simplex code has weight 2^{r-1}.

11.1 Theorem. *Let $\mathcal{S}_r(2)$ (the binary r-dimensional* **Simplex code***) be the code whose generator matrix has as columns all nonzero bitstrings of length r. Then $\mathcal{S}_r(2)$ is a code $[2^r-1, r, 2^{r-1}]_2$. Moreover each nonzero codeword has weight 2^{r-1}.*

We have codes $[7,3,4]_2$, $[15,4,8]_2$, $[31,5,16]_2, \dots$ The smallest of these codes is a good old friend already.

Consider the duals of the Simplex codes. The dimension is $2^r-1-r = 2^r-(r+1)$ by Lemma 10.2. How about the minimum distance? This is surprisingly easy to decide. Let $x \in \mathbb{F}_2^{2^r-1}$. This time M_r is a check matrix. This means that x is a codeword if the sum of the columns of M_r corresponding to entries 1 in x vanishes. If $wt(x)=1$, then M_r must have a 0-column, contradiction. If $wt(x)=2$, then two of the columns of M_r are identical, which is not the case. Can we have $wt(x)=3$? This means that the sum of some 3 columns vanishes, equivalently that the sum of two nonzero tuples is a third nonzero tuple. This happens all the time. For example, columns number $1,2$ and 4 in M_r sum to 0. This means that $(1,1,0,1,0,0,0)$ is a codeword. We have seen that our code has minimum distance $d=3$.

11.2 Theorem. *Let $\mathcal{H}_r(2) = \mathcal{S}_r(2)^\perp$, known as a binary* **Hamming code.** *Then $\mathcal{H}_r(2)$ is a code $[2^r-1, 2^r-(r+1), 3]_2$.*

This yields codes $[7,4,3]_2$, $[15,11,3]_2$, $[31,26,3]_2, \dots$ Again we are already familiar with the smallest member of the family.

11.3 Theorem. *The binary Hamming codes are perfect codes.*

Proof. Recall from Chapter 6 that a code is perfect if it meets the sphere packing bound with equality. We have $d=3$. The volume of the ball of radius 1 is $V_2(1,n) = n+1 = 2^r$. As $2^n/|\mathcal{H}_r(2)| = 2^r$, we have equality. $\square$

Hamming decoding

An important question which we have not touched at yet is the **decoding algorithm.** How does the receiver compute the code word that was sent, assuming not more than e errors occurred (here $2e < d$) ?The general idea (see Chapter 3) states that we should search in the vicinity of the received word y (at distance $\leq e$). If we find a codeword x in this ball (we are guaranteed not to find more than one), then we decode $y \mapsto x$. How is this search process done in practice? As data comes streaming in we probably want to decode it in real time, so we must use a fast algorithm.

In the case of the Hamming codes this is particularly easy. We illustrate with the familiar Hamming code $[7, 4, 3]_2$. Here is the check matrix again:

1	0	0	1	1	0	1
0	1	0	1	0	1	1
0	0	1	0	1	1	1

Denote the columns of H by $s_1, s_2, \ldots, s_7$. Assume the codeword x was sent. That x is a code word is equivalent to $xH^t = 0$ (in words: x is orthogonal to the rows of H). For example, take $x = (1, 0, 1, 1, 0, 1, 0)$. As $d = 3$, we assume at most one error occurred. If this error occurred in coordinate number 3, say, then

$$y = (1, 0, 1, 1, 0, 1, 0) + (0, 0, 1, 0, 0, 0, 0) = (1, 0, 0, 1, 0, 1, 0)$$

is the received vector. Here is how the receiver proceeds:
he or she computes $yH^t = (0, 0, 1)^t$. As $y = x + z$, where z is the error vector, the receiver knows that $yH^t = xH^t + zH^t = zH^t$. If zH^t is column number j of H, then the error occurred in that column. The correct code word is obtained by adding 1 in coordinate j to x. In our example $(0, 0, 1) = s_3$. It follows $x = y + (0, 0, 1, 0, 0, 0, 0)$.

1. Write a matrix M_r with r rows and $2^r - 1$ columns, whose columns are all the nonzero binary r-tuples.
2. Then G is a generator matrix of the Simplex code $\mathcal{S}_r(2)$ (of dimension r) and a check matrix of the Hamming code $\mathcal{H}_r(2)$.
3. $\mathcal{H}_r(2)$ is a perfect code with minimum distance 3.
4. Every nonzero codeword of $\mathcal{S}_r(2)$ has weight 2^{r-1}.
5. There is a particularly easy decoding algorithm for the Hamming codes.

Problems

1. Using matrix M_3, find at least 5 different codewords in the Hamming code $[7, 4, 3]_2$.
2. Use M_4 and find at least 5 different codewords in $\mathcal{H}_r(2)$.
3. Use the binary Hamming code $[7, 4, 3]_2$. Decode the following received vectors:
   $y_1 = (1, 1, 0, 1, 1, 0, 0), y_2 = (1, 1, 1, 1, 1, 1, 1), y_3 = (1, 1, 1, 0, 0, 0, 0)$.

Chapter 12

The principle of duality for binary linear codes

We have seen the promised principle of duality in action already, when we considered the Simplex and Hamming codes in the preceding chapter. Let us go over the argument again, which led us to conclude that the Hamming codes have minimum distance $d = 3$.

Let H be a check matrix of a binary linear code $\mathcal{C}$ with parameters $[n, k, d]_2$ (then H is a $(n-k, n)$-matrix of rank $n-k$). We want to read off d from H. Consider a nonzero bitstring $x = (x_1, x_2, \ldots, x_n)$. Then $x \in \mathcal{C}$ if and only if x is orthogonal to all rows of H, equivalently if the linear combination of the columns of H with the x_i as coefficients vanishes. If x has weight w, then the sum of some w columns of H vanishes. In particular d (the minimum weight of $\mathcal{C}$) is the smallest number such that some d columns of H sum to the 0-vector.

12.1 Definition. *A binary matrix H is $t-$**wise independent** if no t or less columns of H sum to* 0.

If H is t-wise independent, then $\mathcal{C}$ does not contain a nonzero vector of weight $\leq t$. If H is not t-wise independent, then $\mathcal{C}$ does contain a nonzero vector of weight $\leq t$. This gives us one form of the principle of duality.

12.2 Theorem (Principle of duality). *Let $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a linear code $[n, k]_2$. Then the following are equivalent:*

- *$\mathcal{C}$ has minimum distance $\geq d$.*

- *A check matrix of $\mathcal{C}$ is $(d-1)$-wise independent.*

Consider

$$H = \left(\begin{array}{c} 0100011100011101 \\ 0010010011011011 \\ 0001001010110111 \\ 0000100101101111 \\ \hline 1111111111111111 \end{array} \right).$$

The 16 columns of H run through all binary 4-tuples (including the 0-tuple), followed by a last entry of 1. The first 5 columns show that H has rank 5. It is therefore the check matrix of a code $[16, 11, d]_2$. What is d? As there is no 0-column in H we have $d > 1$. As no two columns in H are equal, we have $d > 2$. The sum of three different columns of H cannot vanish, because of the last entry $(1 + 1 + 1 = 1)$. We have $d \geq 4$. Finally, we find plenty of cases where 4 columns of H sum to 0. For example, columns number $1, 2, 3$ and 12 sum to the 0-column. This shows

$$x = (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0) \in \mathcal{C}.$$

We have seen that the code $\mathcal{C}$ with H as check matrix has parameters $[16, 11, 4]_2$.

Instead of using the check matrix H we can just as well express the situation basing ourselves on the dual code $\mathcal{C}^{\perp}$ itself (the code generated by the rows of H). Imagine an array C whose rows are all the codewords of $\mathcal{C}^{\perp}$. Assume some 4 columns of H sum to 0. As you run your fingers down these 4 columns you will not encounter all possible bitstrings of length 4 but only at most half of them: if I know the first three bits, then the last one is uniquely determined. Assume on the other hand that H is 3-wise independent. If I consider the projection onto any set of 3 columns, then every 3-tuple will be encountered, even the same number of times.

Our matrix M_3 is a generator matrix of the Simplex code $[7, 3, 4]_2$ and a check matrix of the Hamming code $[7, 4, 3]_2$. Let us write down all linear combinations of the rows of M_3, hence all codewords of the Simplex code:

$$C = \begin{array}{|c|c|c|c|c|c|c|}\hline 0&0&0&0&0&0&0\\\hline 1&0&0&1&1&0&1\\\hline 0&1&0&1&0&1&1\\\hline 0&0&1&0&1&1&1\\\hline 1&1&0&0&1&1&0\\\hline 1&0&1&1&0&1&0\\\hline 0&1&1&1&1&0&0\\\hline 1&1&1&0&0&0&1\\\hline\end{array}$$

Observe that as promised all the nonzero rows have weight 4. The fact that the Hamming code has $d = 3$, equivalently that its check matrix M_3 is 2-wise independent, now translates into the following fact: for any $2 = 3 - 1$ columns if we run our fingers down this pair of columns in C, then every pair of entries (every binary pair) appears the same number of times, twice. This leads to an important combinatorial notion:

12.3 Definition. *A binary array with n columns is a* **binary orthogonal array** *of* **strength** *t if in the projection onto any set of t columns each t-tuple of entries occurs the same number λ of times. We write the parameters of such an orthogonal array as*

$$OA_\lambda(t, n, 2).$$

In particular the number of rows is then $\lambda 2^t$. The array C above is an $OA_2(2, 7, 2)$. We record the Principle of Duality in the following form:

12.4 Theorem. *[binary Principle of Duality] Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear subspace. Then the minimum distance of $\mathcal{C}$ and the (maximal) strength of $\mathcal{C}^\perp$ are related by*

$$d(\mathcal{C}) = t(\mathcal{C}^\perp) + 1.$$

So the minimum distance of a binary linear code is precisely one larger than the strength of its dual. For example, the dual of a code $[8, 4, 4]_2$ is a code of dimension 4 (hence with $2^4 = 16$ codewords) with strength $d - 1 = 4 - 1 = 3$. This means in clear that if we fix any three coordinates and if we fix the 3-tuple in the projection, there will be precisely $\lambda = 16/8 = 2$ codewords of our dual code having this 3-tuple in these 3 coordinates.

The notion of an orthogonal array originated not in information theory but in statictics. It plays a central role in the design of experiments (see the recent book [6]) and has many more applications in theoretical computer science and cryptology. The reader is invited to generalize Definition 12.3 from the binary to the general q-ary case.

Observe that any set of n-tuples has a minimum distance and a strength. One speaks of codes if one is primarily interested in the minimum distance, of orthogonal arrays if the strength is in the center of the interest. The principle of duality shows that these two notions are related by duality in the binary linear case. That means we have another large field of applications for our codes.

1. The **strength** is the most important parameter of an **orthogonal array**.

2. The **Principle of duality for binary linear codes** states that the minimum distance of $\mathcal{C}$ is one more than the strength of its dual.

3. An equivalent formulation of the Principle of duality is the following: the minimum distance of $\mathcal{C}$ is the smallest number $t \geq 1$ such that some t columns of a check matrix sum to 0.

Problems

1. Define q-ary orthogonal arrays for arbitrary q.

2. Show that a binary orthogonal array of strength $t > 1$ also has strength $t - 1$.

Chapter 13

Prime fields

We have defined codes in general, but considered linear codes only in the binary case. This is not general enough. The tactical advantage when we consider **linear** codes is that linear algebra helps us in finding and describing these structures. Linear algebra works over any **field.** In the binary case the underlying field is $\mathbb{F}_2 = \{0,1\}$, the smallest of all. Many fruitful code constructions rely on codes defined over a large field and involve "going down" to codes over a small field. Also, there are constructions that work best over larger fields. There is simply no reasonable way to get around linear codes over general finite fields.

The basic properties of a field K are the following: addition and multiplication both define **commutative** groups (in particular $a+b=b+a$ and $a\cdot b=b\cdot a$ for all $a,b\in K$), the collaboration of addition and multiplication is defined by the **distributive law** ($a(b+c)=ab+ac$). Most important is the following requirement: every nonzero element of K possesses a multiplicative inverse, in formulas: for every $0\neq a\in K$ there is some $a^{-1}\in K$ such that $a\cdot a^{-1}=1$. The existence of the inverse has the following important consequence: if $a\neq 0$ and $b\neq 0$, then $a\cdot b\neq 0$ (non-existence of 0-divisors). For finite fields these conditions (existence of the inverse and non-existence of 0-divisors) are equivalent. Let us prove one direction:

13.1 Lemma. *If multiplicative inverses exist, then there are no 0-divisors.*

Proof. Assume $ab=0$ and $b\neq 0$. We want to show that necessarily $a=0$. Multiply both sides by b^{-1}. This yields

$$abb^{-1}=a\cdot 1=a=0\cdot b^{-1}=0.$$

□

Some finite fields are easy to construct and to work with:

13.2 Theorem. *Let p be prime. Then $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ is a field (a* **prime field***).*

Let us make sure we know what calculation mod an integer n means. Loosing speaking, we calculate just as in the integers, but we identify n with 0.

13.3 Example. *Let us calculate mod 3. We have only 3 different numbers mod 3. These are $0, 1, 2$. This is so because $3 = 0, 4 = 3 + 1 = 1$ and so forth. As $1 + 2 = 3 = 0$ we have that 1 and 2 are additive inverses. This can also be written as $2 = -1$ and $1 = -2$. Also $2^2 = 4 = 1$.*

13.4 Example. *Now calculate mod 4. There are 4 different numbers mod 4. These are $0, 1, 2, 3$. We have for example $2 + 3 = 5 = 1$ and $2 + 2 = 4 = 0$, hence $2 = -2$. What the multiplication is concerned: $2 \cdot 3 = 6 = 2, 3^2 = 9 = 1$ and $2 \cdot 2 = 4 = 0$. The last formula shows that 2 is a divisor of 0. In particular the numbers mod 4 do not form a field.*

Why is Theorem 13.2 true? As addition and multiplication in $\mathbb{F}_p$ are inherited from the integers, it is clear that distributivity holds and that addition and multiplication are commutative. The only problem is the existence of the inverse (equivalently the absence of 0-divisors. A number is $\neq 0$ mod p if it is not divisible by p. The non-existence of 0-divisors is therefore equivalent to the following statement: if integers a and b both are not divisible by p, then $a \cdot b$ is not divisible by p. This is clearly true. It is one way of expressing the defining property of a **prime,** and it certainly is not true for composite numbers (see the example for $n = 4$ above).

13.5 Example. *As $p = 101$ is a prime, there is a field $\mathbb{F}_{101} = \mathbb{Z}/101\mathbb{Z}$ of 101 elements. These are the numbers $0, 1, 2, \ldots, 100$. As $2 \cdot 50 = 100 = -1$ we have $2^{-1} = -50 = 51$. In fact, $2 \cdot 51 = 102 = 1$. As $10^2 = 100 = -1$, we have $10^{-1} = -10 = 91$.*

If it is not clear what the underlying prime is or if we calculate mod an arbitrary integer n, the notation of **congruences** is used: $a \equiv b \pmod{n}$ means that integers a and b are the same when calculating mod n, equivalently that n divides $b - a$. For example we have $33 \equiv 17 \pmod{4}$ and of course these are $\equiv 1 \pmod 4$. The fact that $30 \equiv 12 \equiv 0 \pmod 2$ simply expresses the fact that both 30 and 12 are even numbers (calculation mod 2 is the distinction between even and odd numbers).

1. For every prime p there is a field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ of p elements.
2. We calculate in $\mathbb{F}_p$ just as we calculate with integers, with the simplification that $p = 0$.
3. In case $p = 2$ this yields precisely the field $\mathbb{F}_2$ which we have used all the time

Problems

1. Show that $\mathbb{Z}/6\mathbb{Z}$ is **not** a field (of 6 elements).
2. Find the multiplicative inverse of 16 mod 101.
3. Find $3^{100} \pmod{17}$.
4. Determine all elements $a \in \mathbb{F}_p$ satisfying $a^2 = 1$.

Chapter 14

Finite fields

We have seen in the preceding chapter that $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ is a finite field when p is prime. Are there any other finite fields? It is easy to see the following: if n is the number of elements (the **order**) of a field K, then $n = p^r$ must be a power of a prime. Also, each field of order p^r must contain the prime field $\mathbb{F}_p$. The prime p is then called the **characteristic** of K. We do not want to go all too deeply into field theory. The following theorem will be accepted without proof:

14.1 Theorem. *For every prime-power p^r there is exactly one field $\mathbb{F}_{p^r}$ of that order.*

We need to know how to construct these fields. As $\mathbb{F}_{p^r}$ contains the prime field $\mathbb{F}_p$, it is an r-dimensional vector space over $\mathbb{F}_p$. We can think of $\mathbb{F}_{p^r}$ as the space $\mathbb{F}_p^r$ of r-tuples with entries in the prime field, with an additional multiplicative structure.

In this section we want to describe how finite fields are constructed and use as example the smallest case $\mathbb{F}_4$. The first (and in general most difficult) step in the construction of $\mathbb{F}_{p^r}$ is to find a polynomial $X^r + \ldots$ of degree r with coefficients in $\mathbb{F}_p$, which is **irreducible.**

In our example we simply write down all binary polynomials of degree $r = 2$. Fortunately, there are only four such polynomials:

$$X^2,\ X^2+1,\ X^2+X \text{ and } X^2+X+1.$$

A polynomial of degree r is irreducible if it cannot be written as the product of two polynomials of smaller degree. Irreducible polynomials play a role, which is analogous to the primes in the context of integers. In our example we see that $X^2 = X \cdot X$, $X^2 + 1 = (X+1)^2$ and $X^2 + X = X(X+1)$ are certainly not irreducible. The only remaining candidate is $X^2 + X + 1$. This is an irreducible polynomial.

Assume now we have found an irreducible polynomial

$$f(X) = X^r + a_{r-1}X^{r-1} + \cdots + a_1 X + a_0.$$

The elements of $\mathbb{F}_{p^r}$ are all polynomials of degree $< r$. Addition and multiplication is as usual with polynomials. The important simplification is that $f(X) = 0$. We say we calculate **mod** the polynomial $f(X)$. When doing that it is customary to assign a neutral Greek name to the (former) variable X, say ϵ. The defining equation is therefore $f(\epsilon) = 0$. This means that whenever a power of ϵ appears with an exponent $\geq r$, we reduce the exponent by using the defining equation

$$\epsilon^r = -a_{r-1}\epsilon^{r-1} - \cdots - a_1\epsilon - a_0.$$

Back to our example $\mathbb{F}_4$. We have the irreducible polynomial $f(X) = X^2 + X + 1$ (with coefficients in $\mathbb{F}_2$). As new name for X when calculating mod $f(X)$ we choose ω. The defining equation is therefore

$$\omega^2 = \omega + 1$$

(observe that in characteristic 2 we have $+ = -$). This suffices to describe the field structure completely. We have $\mathbb{F}_4 = \{0, 1, \omega, \omega + 1\}$ and addition is addition of polynomials. The defining equation tells as what ω^2 is. We can go on: $\omega^3 = \omega(\omega+1) = \omega^2 + \omega = (\omega+1) + \omega = 1$. As $\omega^3 = 1$ we say that ω has **order** 3. We have written all nonzero field elements as powers of ω. This defines the multiplicative structure.

In general it is not always true that all field elements are powers of X. If this is the case one speaks of a **primitive** polynomial and calls the corresponding field elements **primitive** elements. Every finite field can be described by a primitive polynomial.

We want to accept that fields $\mathbb{F}_q$ exist (and are uniquely determined) for every prime-power q. It will be assumed that the reader is completely familiar with prime fields and with $\mathbb{F}_4$. Should we encounter any other finite field it will be constructed following the recipe given in the present chapter.

1. For every prime-power $q = p^r$ there is a finite field $\mathbb{F}_q$ with q elements.
2. It is uniquely determined and contains the prime field $\mathbb{F}_p$.
3. $\mathbb{F}_{p^r}$ is constructed with the help of an irreducible polynomial of degree r with coefficients in $\mathbb{F}_p$.
4. The field $\mathbb{F}_4$ has been described in detail.
5. $\mathbb{F}_4 = \{0, 1, \omega, \omega^2\}$, where $1 + \omega + \omega^2 = 0$ and $\omega^3 = 1$.

Problems

1. Find all irreducible polynomials of degree 3 with coefficients in $\mathbb{F}_2$.
2. Construct $\mathbb{F}_8$.
3. How many elements $a \in \mathbb{F}_p$ satisfy $a^2 = 1$?
4. Is it true that $\mathbb{F}_4 \subset \mathbb{F}_8$?
5. Determine the sum of all elements in $\mathbb{F}_q$.
6. Construct the field of complex numbers as a quadratic extension of the field $\mathbb{R}$ of real numbers. What is the natural choice of a quadratic irreducible polynomial?

Chapter 15

General linear codes

15.1 Definition. *Let $\mathbb{F}_q$ be the field with q elements. A q-ary linear code of length n and dimension k is a linear subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$ of vector space dimension k. If its minimum distance is d, then we record the parameters of $\mathcal{C}$ as $[n, k, d]_q$.*

Observe that a q-ary linear code of dimension k has q^k code words. It contains the all-0-word $0 = (0, 0, \ldots, 0)$.

For fixed q we wish to construct codes $[n, k, d]_q$ with large d, large k and small n. Recall that a code with minimum distance d allows the correction of e transmission errors, when $2e < d$.

All our basic facts concerning binary linear codes carry over to general linear codes. The first benefit we drew from linearity in the binary case was Proposition 7.5, the reduction from minimum distance to minimum weight.

15.2 Proposition. *The minimum distance d of a linear code $[n, k, d]_q$ equals its minimum weight. Here the weight of a codeword is the number of coordinates with a nonzero entry and the minimum is taken over the nonzero words of the code.*

Proof. This is true because each distance between two different codewords is also the weight of some nonzero codeword: $d(x, y) = wt(x - y)$, and $x - y$ is a codeword because the code is linear. $\square$

A **generator matrix** G of an $[n, k, d]_q$-code $\mathcal{C}$ is a (k, n)-matrix whose rows form a basis of $\mathcal{C}$. The code $\mathcal{C}$ is recovered from G by taking all linear combinations of rows. If we compare with the binary case, the sums that we used there have to be replaced by **linear combinations** in the general case. Here is a review of some important terms from linear algebra:

15.3 Definition. *Let V be a vector space defined over a field K and $v_1, v_2, \ldots, v_m \in V$. A* **linear combination** *of $v_1, \ldots, v_m$ is an expression of the form*

$$\lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_m v_m,$$

where the $\lambda_i \in K$. If there is a linear combination such that $\sum_{i=1}^m \lambda_i v_i = 0$ and not all the $\lambda_i = 0$, then the set $\{v_1, \ldots, v_m\}$ is called **linearly dependent.** *It is* **linearly independent** *if it is not linearly dependent.*

Recall also that a **basis** of a vector space is a maximal set of linearly independent vectors, and that the **dimension** of V is the number of elements in a basis. Our linear codes are subspaces of $\mathbb{F}_q^n$, the set of all n-tuples. The n-tuples form a vector space of dimension n. The characterization of generator matrices given in Proposition 7.6 for the binary case looks as follows in the general case:

15.4 Proposition. *Let $\mathcal{C}$ be an $[n, k]_q$-code and G a matrix whose rows are codewords of $\mathcal{C}$. The following are equivalent:*

1. *G is a generator matrix.*

2. *G has rank k.*

3. *There is a (k, k)-submatrix of nonzero determinant*

4. *The rows of G are* **linearly independent**

We illustrate with the following code of length 4, with entries in the field $\mathbb{F}_4 = \{0, 1, \omega, \omega^2\}$.

0	0	0	0
1	1	1	1
ω	ω	ω	ω
ω^2	ω^2	ω^2	ω^2
0	1	ω	ω^2
1	0	ω^2	ω
ω	ω^2	0	1
ω^2	ω	1	0
0	ω	ω^2	1
1	ω^2	ω	0
ω	0	1	ω^2
ω^2	1	0	ω
0	ω^2	1	ω
1	ω	0	ω^2
ω	1	ω^2	0
ω^2	0	ω	1

It is a linear code (check that linear combinations of codewords are codewords again) and has $16 = 4^2$ elements. This means that its dimension is 2. We can represent it in more compact form by a generator matrix, and hence by a basis. A basis consists of any 2 linearly independent codewords. If we choose the fifth row $z_1 = (0, 1, \omega, \omega^2)$ as one basis element, all scalar multiples of z_1 will be linearly dependent of it, so these cannot be used as second basis element. These are $0z_1 = (0, 0, 0, 0)$, z_1 itself, $\omega z_1 = (0, \omega, \omega^2, 1)$ and $\omega^2 z_1 = (0, \omega^2, 1, \omega)$. We have a choice between the remaining $16 - 4 = 12$ codewords to obtain the second row of a generator matrix. Here is a generator matrix of our code:

$$G = \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline 0 & 1 & \omega & \omega^2 \\ \hline \end{array}$$

The list of all 16 codewords shows that $d = 3$. We have a quaternary linear code $[4, 2, 3]_4$.

In case your linear algebra education is more biased towards solving systems of linear equations and you are not too familiar with terms like "basis" and "dimension" here is how to proceed:

Let k elements of $\mathbb{F}_q^n$ be given. You want to find the dimension of the code they generate.

1. Write your tuples as rows of a (k, n)-matrix G in some order.
2. Use row operations to bring the matrix in row-echelon form. Call the resulting matrix G'.
3. Row operations will not change the row space (the code). The dimension of the code is the number of nonzero rows of G'. These nonzero rows form a basis of the code.

> 1. q-ary linear codes $[n, k, d]_q$ have been defined
> 2. We want large k, large d and small n.
> 3. Minimum distance equals minimum nonzero weight.
> 4. The **generator matrix** is an (k, n)-matrix whose rows form a basis of the code.

Problems

1. Find a generator matrix of a ternary ($q = 3$) linear code $[4, 3, 2]_3$.

Chapter 16

Duality and orthogonal arrays

There is no reason not to use the dot product in $\mathbb{F}_q^n$.

16.1 Definition (dot product). *The dot product is defined on $\mathbb{F}_q^n$ by*

$$(x_1, x_2, \ldots, x_n) \cdot (y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} x_i y_i$$

If $\mathcal{C} \subset \mathbb{F}_q^n$ is a subspace of dimension k, then its dual $\mathcal{C}^\perp$ has dimension $n-k$. A generator matrix H of $\mathcal{C}^\perp$ is called a **check matrix** of $\mathcal{C}$. Also, $\mathcal{C}^{\perp\perp} = \mathcal{C}$.

16.2 Example. *The* **repetition code** $[n, 1, n]_q$ *is the code with the all-1-vector* $(1, 1, \ldots, 1)$ *as basis. Its dual therefore has dimension* $n-1$*. It is the* **sum-0-code** *whose codewords are the* $(x_1, x_2, \ldots, x_n)$ *such that* $\sum_{i=1}^n x_i = 0$*. The sum-0-code is an* $[n, n-1, 2]_q$*-code. Consider the case* $q = 3, n = 4$*. The sum-zero code is an* $[4, 3, 2]_3$*-code. This is the code we saw in Chapter 4. A generator matrix is*

$$G = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 0 & 0 \\ \hline 0 & 1 & 2 & 0 \\ \hline 0 & 0 & 1 & 2 \\ \hline \end{array}$$

All this is not any different from the binary case. The same is true of the principle of duality.

16.3 Definition. *A matrix H with entries in the field $\mathbb{F}_q$ is $t-$***wise independent** *if any t columns of H are linearly independent (thus the projection onto any t columns of H has full rank t).*

We have seen the following important relation in the binary case already. It is one way of expressing the principle of duality.

16.4 Theorem. *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear subspace. Then $\mathcal{C}$ has minimum distance $\geq d$ if and only if a (every) check matrix of $\mathcal{C}$ is $(d-1)-$wise independent.*

We can express this in terms of the dual code instead of in terms of its generator matrix H. Clearly H is $t-$wise independent if and only if in the restriction of $\mathcal{C}^\perp$ to any t columns any t-tuple occurs the same number of times. This leads to the combinatorial notion of an **orthogonal array** of strength t (see Chapter 12). Recall the defining property of an OA of strength t; in the projection onto any set of t columns each possible t-tuple occurs the same λ of times. The number λ is called the **index** of the array. The parameters of a q-ary OA of strength t with n columns and index λ are recorded as

$$OA_\lambda(t, n, q).$$

The number of rows is λq^t.

16.5 Theorem (Principle of Duality). *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear subspace. Then the minimum distance of $\mathcal{C}$ and the strength of $\mathcal{C}^\perp$ are related by*

$$d(\mathcal{C}) = t(\mathcal{C}^\perp) + 1.$$

Back to the code $[4, 2, 3]_4$ given in the preceding chapter. As it has only 16 codewords its strength is at most 2. We can check that when we run our fingers down any two columns, each pair of entries occurs precisely once. This means that the strength is indeed two. As an orthogonal array the parameters are therefore

$$OA_1(2, 4, 4).$$

The principle of duality gives a fancy proof for the fact that the sum-zero code has $d = 2$, as follows: the repetition code has as generator matrix $(1, 1, \ldots, 1)$. This matrix is 1-wise independent (as it has no 0 entries) but of course not 2-wise independent. Theorem 16.4 states that the dual of the repetition code has minimum distance $1 + 1 = 2$.

The principle of duality also allows us to construct Hamming codes and Simplex codes in the general q-ary case. The binary case is in Chapter 11.

For every prime-power q and dimension r write a matrix $M_r(q)$ with entries in $\mathbb{F}_q$ such that the columns are representatives for the 1-dimensional subspaces in $\mathbb{F}_q^r$. This means that

- There is no 0-column.
- No two columns are multiples of each other.
- The matrix is maximal with these properties, meaning that for every nonzero r-tuple there is a nonzero multiple of it which is a column of $M_r(q)$.

The number of columns is then $(q^r - 1)/(q - 1)$.

$$M_3(3) = \begin{pmatrix} 1001111001111 \\ 0101200111122 \\ 0010012121212 \end{pmatrix}.$$

This matrix has 13 columns of length 3. For each nonzero ternary triple either this triple or its negative is a column of $M_3(3)$.

Let us use $M_r(q)$ as check matrix of a code, a q-ary Hamming code. Clearly the length is $(q^r-1)/(q-1)$ and the dimension is $(q^r-1)/(q-1)-r$. By definition $M_r(q)$ is 1-wise independent (no 0-column) and 2-wise independent (no two columns are multiples of each other). This means that the code generated by $M_r(q)$ has strength 2 (this is the Simplex code) and, by the principle of duality, that the code with $M_r(q)$ as check matrix has minimum distance 3.

16.6 Theorem (Hamming codes). *Let $\mathcal{H}_r(q)$ be the code with check matrix an $(r, (q^r - 1)/(q - 1))$-matrix $M_r(q)$ whose columns are representatives of the 1-dimensional subspaces. Then $\mathcal{H}_r(q)$ is an $[\frac{q^r-1}{q-1}, \frac{q^r-1}{q-1} - r, 3]_q$-code. The Hamming code is a perfect code.*

Proof. It remains to show that the Hamming codes are perfect. We have seen that the minimum distance of $\mathcal{H}_r(q)$ is $d = 3$. In the terminology of the sphere packing bound Theorem 6.3 we have $e = 1$. The volume of the ball of radius 1 is $V_q(1, n) = 1 + (q - 1)n$. As $n = (q^r - 1)/(q - 1)$ we get $V_q(1, n) = q^r$, hence $|\mathcal{H}_r(q)| \cdot V_q(1, n) = q^n$. $\square$

The Simplex code $\mathcal{S}_r(q) = \mathcal{H}_r(q)^\perp$ has parameters $[\frac{q^r-1}{q-1}, r, q^{r-1}]_q$.

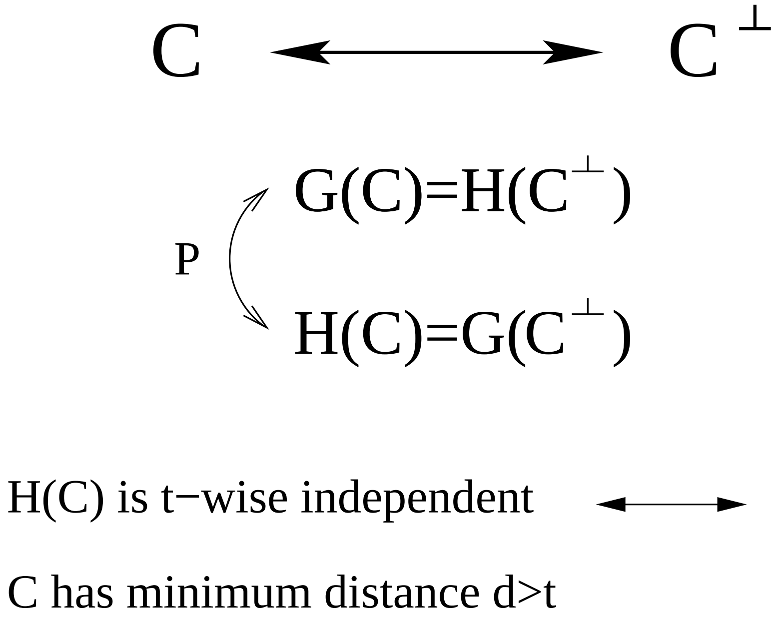

Figure 16.1: Principle of duality

The principle of duality, although rather elementary, is very important. One could almost say it distinguishes coding theorists from the rest of the world. If you really understand it you may (almost) call yourself a coding theorist. We saw that duality motivates the definition of orthogonal arrays. Again, technically this is easy to understand. The theory of linear orthogonal arrays is the theory of dual linear codes, hence of linear codes. However, the strength invited applications which are very different from the application of the minimum distance we considered so far.

As it is so important to understand the principle of duality, we sum up and visualize: In order to determine the minimum distance d of a code C we have to determine the maximum t such that a check matrix H of C is t-wise independent. Then $d = t + 1$. If we know a generator matrix G of C, we can bring it in standard form at first (using Gauß elimination), then apply P-transformation, meaning the trick described in Lemma 10.7 for the binary case and in the second problem of the present chapter for the general case.

1. The **dual** $\mathcal{C}^\perp$ of a linear code $\mathcal{C}$ is defined with respect to the usual **dot product.** $\mathcal{C}$ and $\mathcal{C}^\perp$ have complementary dimensions.
2. A **generator matrix** of $\mathcal{C}$ has rows forming a basis of $\mathcal{C}$.
3. A **check matrix** of $\mathcal{C}$ is a generator matrix of $\mathcal{C}^\perp$.
4. The **strength** is the most important parameter of an **orthogonal array**.
5. The **Principle of duality for linear codes** states that the minimum distance of $\mathcal{C}$ is one more than the strength of its dual.
6. Two trivial families of codes, the **sum-zero codes** and the **repetition codes,** are related by duality.
7. Pick a prime-power q and a natural number r.
8. Write a matrix $M_r(q)$ with r rows and $(q^r-1)/(q-1)$ columns, whose columns are representatives for the 1-dimensional subspaces of $\mathbb{F}_q^r$.
9. Then $M_r(q)$ is a generator matrix of the **Simplex code** $\mathcal{S}_r(q)$ (of dimension r) and a check matrix of the **Hamming code** $\mathcal{H}_r(q)$.
10. $\mathcal{H}_r(q)$ is a perfect code with minimum distance 3.
11. Every nonzero codeword of $\mathcal{S}_r(q)$ has weight q^{r-1}.

Problems

1. Prove that the dual of the linear repetition code of length n is the sum zero-code.

2. Prove the following generalization of Lemma 10.7: Let a generator matrix for a code $[n,k]_q$ be written in the form $G=(I|P)$. Then a check matrix is $H=(-P^T|I)$. Here I in G is the (k,k)-unit matrix,

the I in H is the $(n-k, n-k)$-unit matrix and P^T is the transpose of P.

3. Consider the following quaternary matrix H :

$$\begin{array}{|ccccccccccccccccccccc|}
\hline
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & \omega & \overline{\omega} & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & \omega & \omega & \omega & \overline{\omega} & \overline{\omega} & \overline{\omega} \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & \omega & \overline{\omega} & 1 & \omega & \overline{\omega} & 1 & \omega & \overline{\omega} & 1 & \omega & \overline{\omega} & 1 & \omega & \overline{\omega} \\
\hline
\end{array}$$

Determine the strength of the orthogonal array generated by H and the parameters of the quaternary code with H as check matrix. Here $\mathbb{F}_4 = \{0, 1, \omega, \omega^2 = \overline{\omega}\}$. What is the name of this code?

4. Determine the parameters of the binary code with the following check matrix (this is easy):

1	1	0	0	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	0	1	1
0	0	0	0	1	1	1	1	1	0	1

More interesting is the code with this matrix as generator matrix. Determine its parameters.

5. Determine the parameters of the code with the following check matrix:

0	0	1	1	ω	ω
0	ω	0	ω^2	ω	ω^2
1	1	1	1	1	1

Compare the result with the Singleton bound.

6. Same problem as before. The check matrix is

1	0	0	1	ω^2	ω
0	1	0	1	ω	ω^2
0	0	1	1	1	1

Prove your result.

7. What is the smallest length n such that a code $[n, 7, 3]_3$ exists. (Hint: think of the Hamming codes). Show your argument.

Chapter 17

The game of SETS and some ternary codes

The game of SETS consists of 81 cards. Each card assigns a unique value to each of four attributes. These attributes are

$$NUMBER, SHAPE, COLOUR \text{ and } SHADING.$$

More precisely: each card has one, two or three symbols on it (attribute NUMBER has values one, two, three); these symbols all have the same shape, oval, diamond or squiggle (attribute SHAPE has values oval, diamond, squiggle), they have all the same colour, purple, red or green (attribute COLOUR has values purple, red, green), and the same shading, either outlined, filled or striped (attribute SHADING has values outlined, filled, striped). We can therefore think of a card as a 4-tuple of attributes. For example, there is a card (two,squiggle,green,filled). This is represented graphically as a card with two green filled squiggles on it. The other way around: the purple card with one outlined oval will be represented as the 4-tuple (one,oval,purple,outlined). We see the representation of the cards as 4-tuples as a first step in a process of simplification and algebraization. The next step is prompted by the observation that the concrete values of the attributes are irrelevant for the combinatorial questions pertaining to the game. If we agree that purple is colour number 0, red gets represented by 1 and green by 2, then the value of the attribute COLOUR is represented by the number in the third coordinate of the 4-tuple. We do the same for each coordinate (attribute), using the same ordering of the values as above. The cards considered above can now be written in a shorter way:

$$(two, squiggle, green, filled) = (1, 2, 2, 1)$$

$$(one, oval, purple, outlined) = (0, 0, 0, 0)$$

We interpret the cards as the quadruples with entries $0, 1$ or 2 in each coordinate, in other words: each card represents an element of $\mathbb{F}_3^4$.

The SETS

The SETS constitute the fundamental idea of the game. Three cards form a SET if the following holds: for each attribute the values of our three cards are either all the same or all different. For example, the following three cards form a SET:

(two,	squiggle,	green,	filled)
(three,	squiggle,	green,	outlined)
(one,	squiggle,	green,	striped)

The all have the same value in coordinates two and three (SHAPE and COLOUR) and all different values in coordinates one and four (NUMBER and SHADING). Using the addition in $\mathbb{F}_3^4$ the defining property of SETS can be expressed in a natural way. When three cards form a SET, then by definition the values of each particular attribute are either all the same ($0, 0, 0$ or $1, 1, 1$ or $2, 2, 2$) or all different ($0, 1, 2$ after reordering). What do all these triples have in common? Their sum is 0 (recall that we calculate mod 3). In fact, three numbers from $\mathbb{F}_3$ sum to 0 if and only if they are all the same or all different. This gives us the desired expression for the defining property of sets:

17.1 Lemma. *Three different elements x, y, z of $\mathbb{F}_3^4$ form a SET if and only if $x + y + z = (0, 0, 0, 0)$.*

We call $(0, 0, 0, 0)$ the zero (card or tuple) and write shortly 0 for it. An important property of SETS is now clearly visible: any two cards are contained in precisely one SET. In fact, consider for example the cards $(0, 1, 0, 2)$ and $(1, 2, 0, 2)$. Which card will complement this to a set? Each coordinate entry of the third card has to be such that the sum of the entries of all cards vanishes. This determines the third card. It is in our example $(2, 0, 0, 2)$.

We ask the following natural question: what is the maximum number of cards, which do not contain a SET? Let us call such a collection of cards **SET-free.** An equivalent property is that no three of our cards should add to $0 = (0,0,0,0)$. Using a little trick we can express this in terms of linear independence: Let $x = (x_1, x_2, x_3, x_4)$ be a card. Consider the corresponding quintuple $x' = (x_1, x_2, x_3, x_4, 1) = (x, 1)$. If $x+y+z = 0$, then also $x'+y'+z' = 0$ as the entries in the first coordinate add up to 0. In particular x', y', z' are linearly dependent. Assume conversely that x', y', z' are linearly dependent (and different). The last coordinate shows that the coefficients of this linear combination are three nonzero elements, which sum to 0. It follows that these elements are the same (either all +1 or all -1). We obtain $x+y+z = 0$. This yields the following translation: A collection of cards x is SET-free if and only the corresponding collection of 5-tuples $x' = (x, 1)$ have the property that any three are linearly independent. We have an equivalent formulation of the original question: What is the maximum size of a collection of vectors in $\mathbb{F}_3^5$ such that all have last coefficient 1 and no three are linearly dependent? Here is an example:

0111111111	0222222222
1011112222	2022221111
0001021221	0002012112
0011220102	0022110201
1111111111	1111111111

In terms of the original question the column vectors of this matrix without the bottom 1 form a SET-free collection of 20 cards. It can be shown that this is maximal (there is no family of 21 such cards). In terms of coding theory we have a ternary matrix, which is 3-wise independent. By the principle of duality it is a check matrix of a linear code $[20, 15, 4]_3$.

Problems

1. How many SETS are there? Show your counting method.

2. Consider the ternary $(4, 10)$-matrix A in the upper left corner of the matrix above:

$$A = \begin{pmatrix} 0111111111 \\ 1011112222 \\ 0001021221 \\ 0011220102 \end{pmatrix}$$

   Determine the parameters of the code generated by A and the parameters of its dual.

3. Consider the following ternary matrix:

1	0	0	0	0	0	0	2	1	2	1	2
0	1	0	0	0	0	1	0	2	1	1	2
0	0	1	0	0	0	2	2	0	1	1	1
0	0	0	1	0	0	1	1	1	0	1	1
0	0	0	0	1	0	1	2	2	2	0	1
0	0	0	0	0	1	2	1	2	2	1	0

   Determine the parameters of the ternary code with this check matrix. What are the parameters after truncation? Compare with the Singleton bound.

Chapter 18

Singleton bound and Reed-Solomon codes

Here is our second general bound on codes. Consider a code $(n, M, d)_q$ and arrange the codewords in an array (with M rows and n columns). Here comes the trick: Pick some $d-1$ coordinates, say the last ones. Forget these coordinates (in other words: project to the first $n-d+1$ coordinates). These shorter strings are still different (if two of them would agree, then the codewords they were derived from would have distance $\leq d-1$, which is not possible by the definition of d). It follows that M cannot be larger than the number of all q-ary $(n-d+1)$-tuples. We have proved the following bound:

18.1 Theorem (Singleton bound). *Let $\mathcal{C}$ be a q-ary code of length n. Then the following holds:*

$$|\mathcal{C}| \leq q^{n-d+1}.$$

Codes satisfying this bound with equality are traditionally called **MDS-codes**. *Here the letters* **MDS** *stand for* **maximum distance separable.**

In the linear case this simplifies:

18.2 Corollary. *Each linear code $[n, k, d]_q$ satisfies $k + d \leq n + 1$.*

Next we construct a very important family of MDS-codes. Before this can be done we have to recall the concept of a **polynomial.** A polynomial (in one variable) with coefficients in the field K is an expression of the form

$$f(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0.$$

If $a_n \neq 0$, we call n the **degree** of $f(X)$. Each polynomial defines a polynomial function ($u \in K$ is mapped to the evaluation $f(u)$). Call $u \in K$ a **root** (or a **zero**) of $f(X)$ if $f(u) = 0$. It is a basic fact that a polynomial of degree n cannot have more than n roots. The reason is that whenever $f(u) = 0$ we can write $f(X)$ as a product $f(X) = (X - u)g(X)$, where $g(X)$ has degree $n - 1$.

We describe an array whose rows are the codewords. Let the rows be parametrized by the polynomials $p(X)$ with coefficients in $\mathbb{F}_q$, of degree $< k$ (these form a vector space of dimension k with the **monomials** $X^i, i = 0, 1, \ldots, k-1$ as basis). The columns are parametrized by the elements $u \in \mathbb{F}_q$. The entry in row $p(X)$ and column u is the evaluation $p(u)$. Obviously this defines a linear q-ary code of length q (because linear combinations of polynomials of degree $< k$ are again polynomials of degree $< k$). Assume $k < q$. The number of roots of a nonzero polynomial of degree $< k$ is $\leq k-1$. It follows that the minimum distance is $\geq q - (k-1) = q - k + 1$. As there are polynomials of degree $k - 1$ with $k - 1$ different roots we have equality.

18.3 Definition. *For $k < q$ the words of the* **Reed-Solomon code** $\mathcal{RS}(k, q)$ *of dimension k over $\mathbb{F}_q$ are parametrized by the polynomials $p(X)$ with coefficients in $\mathbb{F}_q$ of degree $< k$, the coordinates by $u \in \mathbb{F}_q$. The corresponding entry is $p(u)$. The code $\mathcal{RS}(k, q)$ has length q, dimension k and minimum distance $q - k + 1$. It meets the Singleton bound Theorem 18.1 with equality.*

In order to show that the dual of a Reed-Solomon code is a Reed-Solomon code we need a little algebraic lemma:

18.4 Lemma. *If i is not a multiple of $q - 1$, then* $\sum_{u \in F_q} u^i = 0.$

Proof. Put $s = \sum_{u \in F_q} u^i$. For every $0 \neq u_0 \in \mathbb{F}_q$ we have $s u_0^i = s$ (as uu_0 runs through all of $\mathbb{F}_q$ when u does), hence $s(u_0^i - 1) = 0$. As i is not a multiple of $q - 1$, there is an u_0 such that $u_0^i \neq 1$. It follows $s = 0$. □

18.5 Proposition.

$$\mathcal{RS}(k, q)^\perp = \mathcal{RS}(q - k, q)$$

Proof. Observe that $\mathcal{RS}(k, q)$ and $\mathcal{RS}(q - k, q)$ have complementary dimensions, so it suffices to show they are orthogonal. $\mathcal{RS}(k, q)$ is generated by the $X^i, i < k$, the code $\mathcal{RS}(q - k, q)$ is generated by the $X^j, j < q - k$. The dot product of the corresponding codewords is $\sum_u u^{i+j}$, which is $= 0$ by Lemma 18.4. □

18.6 Theorem. *The Reed-Solomon code $\mathcal{RS}(k,q)$ is an $OA_1(k,q,q)$.*

Proof. Recall that $\mathcal{RS}(k,q)$ has minimum distance $q-k+1$, and that its dual is $\mathcal{RS}(q-k,q)$. As this code has minimum distance $q-(q-k)+1=k+1$ we conclude that the strength of $\mathcal{RS}(k,q)$ is k. As it has q^k elements the claim follows. □

The Reed-Solomon code $\mathcal{RS}(2,4)$ has been displayed in Chapter 15. Observe that q-ary Reed Solomon codes have length q, so we need a relatively large field to obtain interesting codes. Examples for Reed-Solomon codes are

$$[5,3,3]_5,\ [7,3,5]_7,\ [8,4,5]_8,\ [8,5,4]_8,\ [16,8,9]_{16},\dots$$

As the **monomials** $X^i, i=0,1,\dots,k-1$ form a basis of the space of polynomials of degree $<k$, a generator matrix of $\mathcal{RS}(k,q)$ has rows indexed by i, columns indexed by the $u \in \mathbb{F}_q$, with entries u^i.

Consider 8-ary Reed-Solomon codes. We need the field $\mathbb{F}_8$. Let us just follow the general construction prodecure: as $8=2^3$ we need an irreducible polynomial $f(X)$ of degree 3 with coefficients in $\mathbb{F}_2$. The constant term cannot vanish (the polynomial would be divisble by X otherwise). This leaves us with 4 candidates:

$$X^3+1,\ X^3+X^2+1,\ X^3+X+1 \text{ and } X^3+X^2+X+1.$$

The first and the last of these candidates are reducible as they satisfy $f(1)=0$. These polynomials are therefore divisible by $X-1=X+1$. The two remaining candidates are irreducible. We choose

$$f(X)=X^3+X^2+1.$$

Denote by ϵ the image of X in $\mathbb{F}_8$. The elements of $\mathbb{F}_8$ are the polynomials of degree ≤ 2 in ϵ :

$$\mathbb{F}_8=\{0,1,\epsilon,\epsilon+1,\epsilon^2,\epsilon^2+1,\epsilon^2+\epsilon,\epsilon^2+\epsilon+1\}$$

This defines the addition. The multiplication is determined by the basic equation (equivalent to the choice of $f(X)$)

$$\epsilon^3=\epsilon^2+1.$$

We can multiply any two elements of $\mathbb{F}_8$. The basic equation allows us to get rid of all exponents > 2. Consider the powers of ϵ :

$$\epsilon^3 = \epsilon^2 + 1,\ \epsilon^4 = \epsilon^3 + \epsilon = \epsilon^2 + \epsilon + 1.$$

In the same manner we can determine all powers of ϵ :

$$\epsilon^5 = \epsilon^3 + \epsilon^2 + \epsilon = \epsilon + 1,\ \epsilon^6 = \epsilon^2 + \epsilon \text{ and } \epsilon^7 = 1.$$

We see once again that knowledge of the irreducible polynomial is equivalent to knowledge of the field.

To write down generator matrices of the 8-ary Reed-Solomon codes let $0, 1, \epsilon, \dots, \epsilon^6$ parametrize the columns, in this order. Rows are parametrized by the monomials $1, X, X^2, \dots$. The following generator matrix for $\mathcal{RS}(4,8)$ is obtained:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \epsilon & \epsilon^2 & \epsilon^3 & \epsilon^4 & \epsilon^5 & \epsilon^6 \\ 0 & 1 & \epsilon^2 & \epsilon^4 & \epsilon^6 & \epsilon & \epsilon^3 & \epsilon^5 \\ 0 & 1 & \epsilon^3 & \epsilon^6 & \epsilon^2 & \epsilon^5 & \epsilon & \epsilon^4 \end{pmatrix}$$

We know that $\mathcal{RS}(4,8)$ is a self-dual code $[8,4,5]_8$. Also, the Reed-Solomon codes of different dimensions are contained in each other. The first three rows of G generate $\mathcal{RS}(3,8)$, the first two rows generate $\mathcal{RS}(2,8)$ and $\mathcal{RS}(1,8)$ is the repetition code.

1. Pick a prime-power q and a dimension $k \leq q$. The **Reed-Solomon code** $\mathcal{RS}(k,q)$ is defined as follows:
2. The coordinates are parametrized by the elements $u \in \mathbb{F}_q$.
3. The codewords are parametrized by the polynomials $p(X)$ with coefficients in $\mathbb{F}_q$ of degree $< k$.
4. The codeword parametrized by $p(X)$ has as entries the evaluations $p(u)$.
5. $\mathcal{RS}(k,q)$ is a code $[q, k, q-k+1]_q$ and an orthogonal array $OA_1(k,q,q)$.

Reed-Solomon codes

generator matrix:

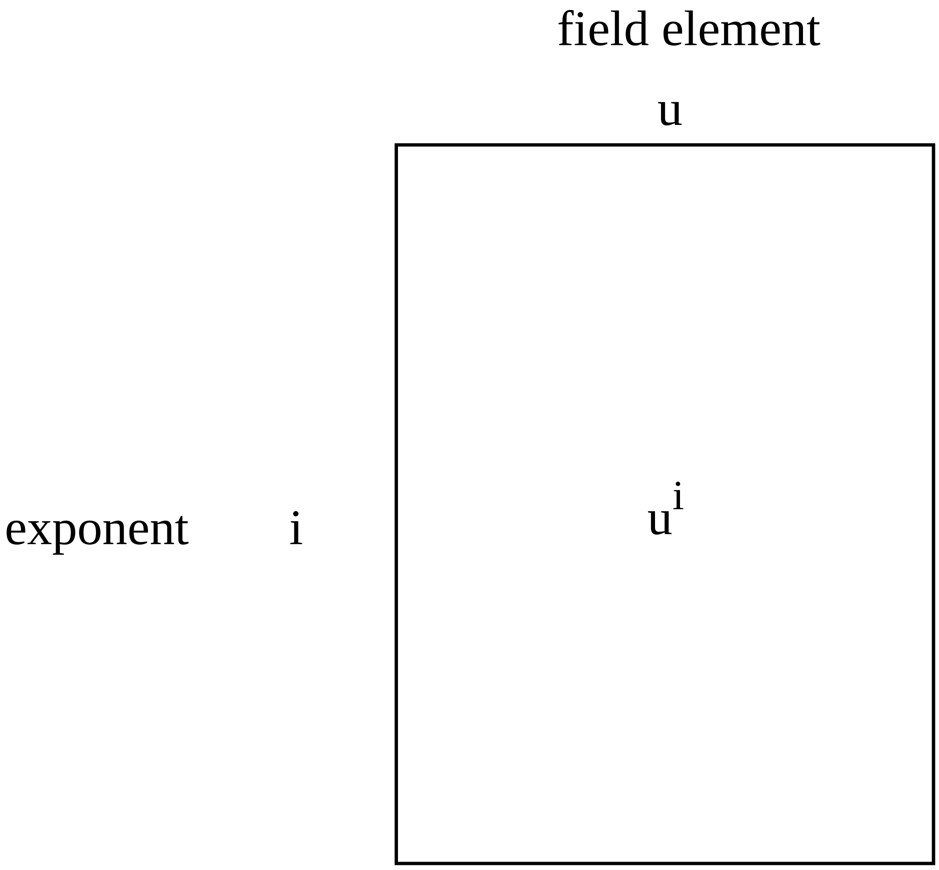

Figure 18.1: Reed-Solomon codes

Problems

1. Write generator matrices of $\mathcal{RS}(2,5)$ and of $\mathcal{RS}(3,5)$

Chapter 19

Recursive constructions I: shortening and puncturing

So far we have concentrated on constructing codes from scratch, so to speak. One idea led to the Hamming codes, another one to the Reed-Solomon codes. Now we want to start studying recursive mechanisms. Here is the most obvious question: which codes can we find inside a given code?

Let $\mathcal{C}$ be a code $[n, k, d]_q$. Fix a coordinate, say n. The **shortening** $\mathcal{D}$ of $\mathcal{C}$ is defined by

$$\mathcal{D} = \{(x_1, x_2, \ldots, x_{n-1}) \mid (x_1, x_2, \ldots, x_{n-1}, 0) \in \mathcal{C}\}.$$

That means we consider the subcode of $\mathcal{C}$ consisting of all codewords that end in 0. Then we throw away this superfluous last coordinate. What can we say about the parameters of $\mathcal{D}$? Clearly $\mathcal{D}$ is linear. As it is defined by one linear equation, the dimension is $k-1$ (the dimension will be k if all words of $\mathcal{C}$ should be 0 at the last coordinate). Also, $\mathcal{D}$ is a subcode of $\mathcal{C}$. It follows that the minimum weight (=minimum distance) of $\mathcal{D}$ cannot be smaller than d. This describes our first recursive construction:

19.1 Theorem (shortening). *If there is a code $[n, k, d]_q$, then shortening yields a subcode $[n-1, k-1, d]_q$.*

Here is another elementary idea. This procedure is called **projection** in algebra. In the coding community it is known as **puncturing:** define

$$\mathcal{E} = \{(x_1, x_2, \ldots, x_{n-1}) \mid \text{ there is } x \in \mathbb{F}_q \text{ such that}$$

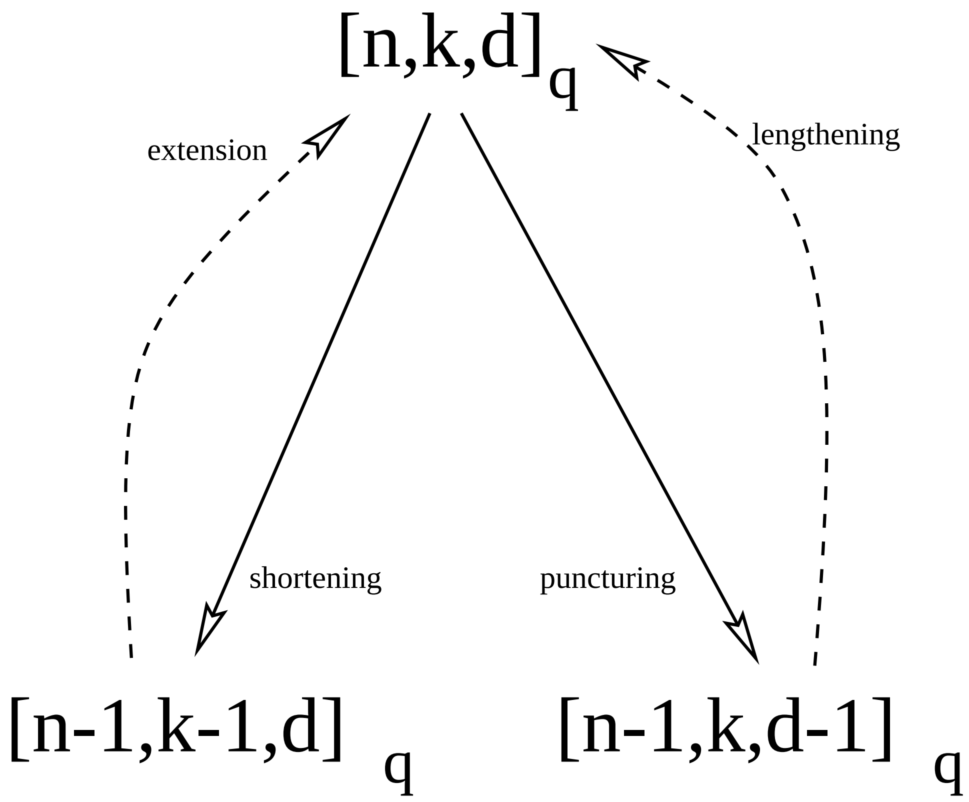

Figure 19.1: Shortening and puncturing

$$(x_1, x_2, \ldots, x_{n-1}, x) \in \mathcal{C}\}.$$

This means we project the codewords of $\mathcal{C}$ onto the first $n-1$ coordinates. The number of codewords of $\mathcal{E}$ is the same as in $\mathcal{C}$ (if two different words in $\mathcal{C}$ would agree in all but one coordinate, their distance would be 1). In particular $\mathcal{E}$ has dimension k. This time we cannot expect the minimum distance to remain unchanged. However, as only one coordinate has been removed, the minimum distance will decrease by at most 1.

19.2 Theorem (shortening). *If there is a code $[n, k, d]_q$, where $d > 1$, then puncturing (projection) yields a code $[n-1, k, d-1]_q$.*

For example, given a code $[24, 12, 8]_2$ (this is known as the **binary extended Golay code**) we can construct by repeated shortening and puncturing

$$[23,12,7]_2, [23,11,8]_2,\ [22,12,6]_2,\ [22,11,7]_2,\ [22,10,8]_2,\ [21,11,6]_2 \dots$$

This raises the interesting question when one of these constructions can be reversed. Given a code $[n-1,k-1,d]_q$, can we find a code $[n,k,d]_q$ such that the original code is obtained by shortening it? This process is called **extension.** Clearly this cannot always be possible. The process of shortening shows that parameters $[n,k,d]_q$ are stronger than $[n-1,k-1,d]_q$. In the case of puncturing the situation is analogous. The inverse of puncturing is called **lengthening** and is clearly not always possible. By the way, the terminology in the literature is rather chaotic.

Here is a famous case when lengthening works: let $\mathcal{C}$ be an $[n,k,d]_2$, where d is odd. Append a new coordinate, where the entry is chosen such that the new lengthened codeword has even weight. This means: the entry in the last coordinate is 0 if the weight of the old codeword is even, the entry in the last coordinate is 1 if the old weight was odd. Clearly the new code is linear and has the same dimension as the old one. Also, in the lengthened code all weights are even. In particular, weight d does not occur. We have a code $[n+1,k,d+1]_2$. The entry in the extra coordinate is known as the **parity check bit.**

19.3 Theorem (parity check). *Let $\mathcal{C}$ be a binary code $[n,k,d]_2$, where d is odd. Appending a parity check bit yields a code $[n+1,k,d+1]_2$.*

In particular a code $[n,k,d]_2$ with odd d exists if and only if a code $[n+1,k,d+1]_2$ exists. In the theory of binary linear codes one can therefore concentrate on even minimum weight. However, this is a specialty of the binary case. We have seen the binary Hamming code $[7,4,3]_2$ in Chapters 7, 8 and 11. A generator matrix of $\mathcal{H}_3(2)$ was given in Chapter 7:

1	0	0	1	1	0	1
0	1	0	1	0	1	1
0	0	1	0	1	1	1
0	0	0	1	1	1	0

After addition of a parity check bit we obtain

1	0	0	1	1	0	1	0
0	1	0	1	0	1	1	0
0	0	1	0	1	1	1	0
0	0	0	1	1	1	0	1

This is a generator matrix of the so-called **extended Hamming-code** $[8,4,4]_2$. We check that any two rows are orthogonal. This means that the extended Hamming code is orthogonal to itself. As its dimension is just half the length (see Lemma 10.2) we conclude that it equals its dual. Such codes are called **self-dual**.

19.4 Theorem. *The extended binary Hamming code* $[8,4,4]_2$ *is self-dual* ($\mathcal{C} = \mathcal{C}^\perp$).

We can apply the parity check to all the binary Hamming codes $[2^r - 1, 2^r - (r+1), 3]_2$. This yields the extended binary Hammingcodes:

19.5 Theorem. *The* **extended binary Hamming codes** *have parameters*

$$[2^r, 2^r - (r+1), 4]_2$$

Examples are

$$[8,4,4]_2\ [16,11,4]_2,\ [32,26,4]_2.$$

1. Let $\mathcal{C}$ be a linear code $[n,k,d]_q$.
2. 1-step **shortening** produces $[n-1,k-1,d]_q$,
   1-step **projection** yields $[n-1,k,d-1]_q$.
3. We speak of a 1-step **extension** if $[n+1,k+1,d]_q$ can be obtained, of a 1-step **lengthening** if $[n+1,k,d+1]_q$ can be constructed.
4. The **parity check** works only in the binary case. It constructs $[n+1,k,d+1]_2$ from $[n,k,d]_2$ when d is odd.
5. The **extended binary Hamming codes** have parameters $[2^r, 2^r-(r+1), 4]_2$.

Problems

1. Let $\mathcal{C}$ be a binary linear code with odd minimum distance d and $\mathcal{D} \subset \mathcal{C}$ be the set of code words of even weight. Prove that $\mathcal{D}$ is a subspace of codimension 1 in $\mathcal{C}$.

2. Given a linear code $[192, 11, 92]_2$ (it exists), make a list of all parameters of codes of lengths 188 and above that can be constructed using iterated shortening and projection.

3. Prove the following: if a code is MDS, then every code derived from it by shortening or projection is MDS as well.

4. Determine the parameters of the 3-dimensional binary code generated by

$$G = \begin{pmatrix} 111000000111100 \\ 000111000110011 \\ 000111111111100 \end{pmatrix}$$

   Add a parity check bit to this generator matrix. What are the parameters of the resulting extended code?

5. Let $X = \{1, 2, \dots, 8\}$. Define a binary matrix M of size $(8, 24)$ as follows: the rows of M are indexed by the elements $x \in X$. The columns of M are indexed by unordered pairs $\{i, j\} \subset X$, where $\{i, j\}$ is none of $\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}$ (these forbidden pairs form what is known as a **one-factor**). The entry in row x and column $\{i, j\}$ is 0 is $x = i$ or $x = j$, it is 1 otherwise. Observe that every column of M has weight 6 and every row has weight 18.
   Prove that M has rank 7 and that its row-space is a code $[24, 7, 10]_2$. This description is from [9].

6. Given a linear code $[24, 7, 10]_2$, which parameters of 6-or 7-dimensional codes can be constructed using iterated shortening and projection?

7. Consider the matrix

$$A = \begin{pmatrix} 0111111111 \\ 1011112222 \\ 0001021221 \\ 0011220102 \end{pmatrix}$$

from Chapter 17, which generates a code $[10, 4, 6]_3$. Find generator matrices for the codes derived by shortening and by projection.

8. Let $\mathcal{C}$ be a code $[n, k, d]_q$ and $x = (x_1, x_2, \ldots, x_w, 0, 0, \ldots) \in \mathcal{C}^\perp$ a codeword of weight w in the dual code. Apply shortening to $\mathcal{C}$ with respect to the first w coordinates. We know that the resulting code has parameters (at least) $[n - w, k - w, d]_q$.

   Show that this shortened has in fact better parameters. This construction is known as **construction** Y_1.

9. We know from Chapter 16 the Simplex codes and their duals, the Hamming codes. Determine the parameters of the codes obtained by applying construction Y_1 to the Simplex codes.

Chapter 20

Syndrome decoding

A weak point in the mechanism of error correction is the decoding algorithm. Recall the general idea of block coding (Chapter 8). As the received vectors y stream in they need to be decoded fast, preferably in real time. The naive method of decoding would be to search through all $\sum_{i=0}^{e}(q-1)^i\binom{n}{i}$ vectors at distance $\leq e$ from y until a codeword is found (we use a code correcting e errors, hence of minimum distance $> 2e$). As we saw in Chapter 9, we want to use strong codes with large e. The length n will then inevitably be large as well. However, the searching process will then last all too long. In Chapter 11 we saw a faster decoding method in the case of binary Hamming codes. It is left as an exercise to show that this decoding method generalizes to q-ary Hamming codes. In the present chapter we present a general method of decoding, which uses the linearity of the codes and is easy to understand.

Assume the linear code $\mathcal{C}$ $[n,k,d]_q$ is described by check matrix H. Let x be the codeword, which was sent, and $y = x + err$ the received word, where err is the error vector. In Chapter 11 we used the vector yH^T. This vector is known as the **syndrome**:

20.1 Definition. *Let H be a check matrix of a code $\mathcal{C}$ with parameters $[n,k,d]_q$ and $y \in \mathbb{F}_q^n$. The* **syndrome** *of y is*

$$\sigma(y) = yH^T \in \mathbb{F}_q^{n-k}.$$

Observe that the entries of $\sigma(y)$ are the dot products of y and the rows of H. In particular $y \in \mathcal{C}$ if and only if $\sigma(y) = 0$ is the 0-vector.

When will two vectors have the same syndrome, $\sigma(y) = \sigma(y')$? This means

$0 = \sigma(y) - \sigma(y') = \sigma(y - y')$, in other words $y - y' \in \mathcal{C}$. The sets of vectors with same syndrome are called **cosets** of $\mathcal{C}$.

20.2 Definition. *A* **coset** *of the linear code* $\mathcal{C}$ *is a set of vectors of the form* $\mathcal{C} + y$. *Two vectors are in the same coset if they have the same syndrome. The number of different cosets is* q^{n-k} *(if* $\mathcal{C}$ *is an* $[n, k]_q$*-code).*

What does this have to do with decoding? If $y = x + err$ is the received vector and we decode it (hopefully) as $y \mapsto x = y - err$, then $\sigma(y) = \sigma(err)$. We are therefore looking for a vector in the same coset as y with small weight. These vectors are our candidates for err.

Here is how syndrome decoding works: in each of the q^{n-k} cosets of $\mathcal{C}$ find a vector of smallest weight. Call this element the **coset leader.** If y is received, the following steps are executed:

1. Compute the syndrome $\sigma(y)$.
2. In the list find the coset leader ϵ of the corresponding coset.
3. Decode $y \mapsto y - \epsilon$.

Let us illustrate with the extended binary Hamming code $[8, 4, 4]_2$. Consider the matrix

$$G = H = \left(\begin{array}{c|c} 1000 & 0111 \\ 0100 & 1011 \\ 0010 & 1101 \\ 0001 & 1110 \end{array} \right)$$

Check again our basic facts: the first 4 columns show that G generates a 4-dimensional code $\mathcal{C}$. As any two rows are orthogonal, we have $\mathcal{C} \subseteq \mathcal{C}^{\perp}$, the dimension shows $\mathcal{C} = \mathcal{C}^{\perp}$: a self-dual code. We can therefore consider G also as a check matrix H for the same code. $\mathcal{C}$ is contained in the sum 0 code (all rows of G have weight 4), and it is clear that there is no codeword of weight 2. This shows that $\mathcal{C}$ is an $[8, 4, 4]_2$-code. Observe that the columns of H consist of all 4-tuples of weights 1 or 3. Observe also that $\mathcal{C}$ contains the all-1-vector (the sum of all rows of G). It follows that there can be no code-words of weight 6 in $\mathcal{C}$.

The number of cosets is 16. One coset is $\mathcal{C}$ itself, with coset leader the 0 word. Each vector of weight 1 is the unique coset leader in its coset (as there are no words of weight 2 in $\mathcal{C}$). This gives us 8 more cosets and their leaders. The corresponding syndromes are columns of H, hence all vectors of weights 1 or 3 in $\mathbb{F}_2^4$. If a coset contains a vector of weight 2, then this is the minimum weight in this coset. The 6 vectors of weight 2 with support in the second half of the coordinates are in pairwise different cosets. This gives us 6 more cosets. The corresponding syndromes are the vectors of weight 2. Only one coset is missing yet. It corresponds to the syndrome $(1, 1, 1, 1)$. We can choose $(1, 0, 0, 0, 1, 0, 0, 0)$ as representative. This gives us the following list of coset leaders and syndromes.

coset leader	syndrome
00000000	0000
10000000	1000
01000000	0100
00100000	0010
00010000	0001
00001000	0111
00000100	1011
00000010	1101
00000001	1110
00001100	1100
00001010	1010
00001001	1001
00000110	0110
00000101	0101
00000011	0011
10001000	1111

Our code corrects one error. This translates to the fact that each vector of weight ≤ 1 is the uniquely determined leader of its coset. As we choose leaders also from cosets which do not contain a unique leader, an attempt is made to decode also in these cases when more than one error occurred.

Assume $y = 11000011$ is received. Compute $\sigma(y) = 1111$. The corresponding coset leader is 10001000. We decode $y \mapsto 01001011$.

1. Let H be a check matrix of the $[n,k,d]_q$−code $\mathcal{C}$.
2. The **syndrome** of $y \in \mathbb{F}_q^n$ is $\sigma(y) = yH^T$.
3. $y, y' \in \mathbb{F}_q^n$ are in the same **coset** of $\mathcal{C}$ if $y - y' \in \mathcal{C}$, equivalently if $\sigma(y) = \sigma(y')$.
4. A **leader** of a coset is a vector of smallest weight.
5. **syndrome decoding:**
6. Establish a list of leaders of all q^{n-k} cosets.
7. Decode $y \mapsto x = y - z$, where z is the leader of the coset containing y.

Problems

1. Use the decoding scheme for $[8,4,4]_2$ given in this chapter to decode the following received vectors: $y = 00110011$, $y = 11111010$, $y = 10001001$.
2. Find coset leaders for the sum-0-codes $[n, n-1, 2]_q$.
3. Find coset leaders for the Hamming code $[7,4,3]_2$.
4. Find a check matrix of a quaternary code $[6,3,4]_4$. Find coset leaders for all cosets.

Chapter 21

Recursive constructions II:Concatenation

The best general family of codes we know thus far are the Reed-Solomon codes. Recall that q-ary Reed-Solomon codes have length q. In particular these codes are interesting only over relatively large fields. On the other hand, if we are searching for q-ary codes of length $n \leq q$ there is no reason not to use Reed-Solomon codes and what results from shortening or projecting. All these codes $[n, k, n-k+1]_q$ are MDS-codes. How about larger lengths? We promised several times to use codes defined over larger fields and certain methods of "going down" to construct good codes over smaller fields. This was one of our motivations to consider codes over large fields. Concatenation is one such general method.

Let $Q = q^r$. The field $\mathbb{F}_Q$ contains $\mathbb{F}_q$ as a subfield. Start from an arbitrary linear Q-ary code $\mathcal{C}$ with parameters $[N, K, D]_Q$. Each codeword is an N-tuple $(x_1, x_2, \ldots, x_N)$, where $x_i \in \mathbb{F}_Q$. Now $\mathbb{F}_Q$ is a vectorspace of dimension r over $\mathbb{F}_q$. Every r-dimensional code over $\mathbb{F}_q$ is an r-dimensional $\mathbb{F}_q$-vectorspace as well. Here is the idea of concatenation: fix a code $\mathcal{D}$ with parameters $[n, r, d]_q$ and a one-to-one mapping, a vectorspace isomorphism $\alpha : \mathbb{F}_Q \longrightarrow \mathcal{D}$. In order to obtain the new q-ary concatenated code we replace each field element $u \in \mathbb{F}_Q$ by its image $\alpha(u) \in \mathbb{F}_q^r$. The images of codewords of $\mathcal{C}$ (Q-ary N-tuples) are q-ary nN-tuples.

21.1 Definition (concatenation). *Let $Q = q^r$ and $\mathcal{C}$ a linear Q-ary code $[N, K, D]_Q$ (the* **outer code***). Let further $\mathcal{D}$ be a linear $[n, r, d]_q$ (the* **inner code***). Choose an $\mathbb{F}_q$-isomorphism $\alpha : \mathbb{F}_Q \longrightarrow \mathcal{D} \subseteq \mathbb{F}_q^n$. The words of the*

concatenated code *are in bijection with the words of* $\mathcal{C}$. *If* $(x_1, x_2, \dots, x_n) \in \mathcal{C}$, *then the corresponding word of the concatenated code is* $(\alpha(x_1), \dots, \alpha(x_N))$.

The length of the concatenated code is nN. It has as many codewords as the outer code $\mathcal{C}$. This number is $Q^K = q^{rK}$. The dimension of the concatenated code is therefore rK. Let $\alpha(x), x \in \mathcal{C}$ be a nonzero codeword. As $\mathcal{C}$ has minimum weight D there are at least D coordinates such that $x_i \neq 0$. For these coordinates $\alpha(x_i)$ is a nonzero word of the inner code $\mathcal{D}$, and has therefore weight $\geq d$. It follows that $\alpha(x)$ has weight $\geq dD$.

21.2 Theorem. *The concatenated code of an outer code* $[N, K, D]_Q$, *where* $Q = q^r$, *and an inner code* $[n, r, d]_q$ *is a linear* q*-ary code* $[nN, rK, dD]_q$.

Use concatenation to derive binary codes from quaternary. The inner codes must then be 2-dimensional binary codes. Use the Reed-Solomon code $[4, 2, 3]_4$ as outer code. As inner code use the sum-0-code $[3, 2, 2]_2$. Its codewords are $000, 110, 101, 011$. We can choose

$$\alpha : 0 \mapsto 000,\ 1 \mapsto 110,\ \omega \mapsto 101,\ \omega^2 \mapsto 011.$$

Consider the codeword of $\mathcal{RS}(2, 4)$ parametrized by the polynomial X. It is $(0, 1, \omega, \omega^2)$. Its image under α is 000110101011 of weight 6. The concatenated code is a $[12, 4, 6]_2$-code.

Assume we want to use our $[8, 4, 4]_2$ as inner code. The outer code will then have to be a 16-ary code. The Reed-Solomon codes yield $[16, k, 17-k]_{16}$ for all $k \leq 16$. Concatenation yields binary codes $[128, 4k, 68-4k]_2$ for all $k \leq 16$. We see that we can construct binary codes of arbitrary lengths by using concatenation with a Reed-Solomon code defined over a large extension field as outer code and a suitable short inner code.

Let us see how to obtain a generator matrix of the concatenated code. Let G be a generator matrix of the inner code. Denote its rows by $z_1, z_2, \dots, z_r$. We need another fact from field theory, see Chapter 14: when $\mathbb{F}_{q^r}$ is constructed from the base field $\mathbb{F}_q$, using an irreducible polynomial $f(X)$ of degree r and ϵ is $X \pmod{f(X)}$, then $1, \epsilon, \dots, \epsilon^{r-1}$ is a basis of $\mathbb{F}_{q^r}$ as a vector space over $\mathbb{F}_q$. Recall that ϵ^r can be written as a linear combination of these smaller powers of ϵ.

We can choose α such that

$$\alpha : 1 \rightarrow z_1, \epsilon \rightarrow z_2, \dots, \epsilon^{r-1} \rightarrow z_r.$$

Let now M be a generator matrix of the inner code, with rows $R_1, R_2, \ldots, R_K$. The rows of a generator matrix of the concatenated code are the images under α of

$$R_1, \epsilon R_1, \ldots, \epsilon^{r-1} R_1, \ldots, R_K, \epsilon R_K, \ldots, \epsilon^{r-1} R_K.$$

Consider our first example. Our "large" field $\mathbb{F}_4$ has basis $1, \omega$. The rows of a generator matrix of the inner code $[3,2,2]_2$ can be chosen as $z_1 = 110$ and $z_2 = 101$. According to our recipe this gives us

$$\alpha : 1 \to 110, \ \omega \to 101.$$

The rest is determined by linear combinations: $\alpha(0) = 000$ of course and $\alpha : \omega^2 = 1 + \omega \to 110 + 101 = 011$. This is what we had before. We know how a generator matrix of the outer $[4,2,3]_4$ can be determined. We have $R_1 = (1,1,1,1), R_2 = (0,1,\omega,\omega^2)$. This yields the following generator matrix for the concatenated $[12,4,6]_2$:

$$\begin{array}{rcl} R_1 & \to & 110110110110 \\ \omega R_1 & \to & 101101101101 \\ R_2 & \to & 000110101011 \\ \omega R_2 & \to & 000101011110 \end{array}$$

1. Let $\mathcal{C}$ be a $[N,K,D]_Q$-code, where $Q = q^r$.
2. Let $\mathcal{D}$ be an $[n,r,d]_q$-code.
3. Concatenation produces an $[nN, rK, dD]_q$-code.
4. We can construct good codes of arbitrary length over small fields by using concatenation with Reed-Solomon codes over extension fields as outer codes.
5. Show how to construct a code $[27,10,10]_3$ by concatenation. What do you use as outer and inner codes?

Problems

1. Use $\mathcal{RS}(4,8)$ and $[7,3,4]_2$ in a concatenation scheme. What are the parameters of the resulting concatenated code?

2. We have already met the following matrix in the Problems section of Chapter 16. It generates a code $[6,3,4]_4$, which is known as the **hexacode.** We want to apply concatenation with the hexacode as outer code. List at least two parameters of binary codes that can be used as inner codes, and the parameters of the resulting concatenated codes.

1	0	0	1	ω^2	ω
0	1	0	1	ω	ω^2
0	0	1	1	1	1

3. Use concatenation with Reed-Solomon codes over $\mathbb{F}_{2^r}$ as outer codes and binary sum-0-codes as inner codes. Determine the parameters of the resulting binary codes.

4. Consider concatenation with Reed-Solomon codes as outer codes and a ternary $[10,4,6]_3$ as inner code. What are the resulting parameters?

5. Construct a 48-dimensional binary linear code by applying concatenation to a Reed-Solomon code as outer code and $[23,12,7]_2$ as inner code. What are the parameters of this concatenated code?

6. Show how to construct a code $[128,12,56]_2$ by concatenation. What do you use as outer and inner codes?

7. Show how to construct a code $[27,10,10]_3$ by concatenation. What do you use as outer and inner codes?

Chapter 22

Universal hashing

So far we have concentrated on one important area of applications for codes, transmission of messages over noisy channels. It has been mentioned early on in this text that the applications of the concept of codes are manifold and certainly not limited to this historically first area. It is time to take a step in this direction.

An important concept in theoretical computer science are **hash functions.** The idea is to find a function $f : X \longrightarrow Y$ from a "large" set X to a "small" set Y with one of the following properties:

- Given $x \in X$ it is computationally infeasible to find $x' \neq x$ such that $f(x') = f(x)$.

- It is computationally infeasible to find $x \neq x'$ such that $f(x') = f(x)$.

Applications in cryptography are described in [15]. To give just one example, when a long document is to be signed electronically one applies at first a hash function to produce a much shorter string (a digest of the original document) and then applies the signature scheme to this hash value. The notion of a hash function is very problematic. For one thing it is clear that numerous collisions (pairs x, x' such that $f(x') = f(x)$)) will happen. We want to make it computationally impossible for an opponent to find any. How can one guarantee that this is the case? The answer is: one cannot. In fact, whenever a new hash function is being proposed it just takes a couple of years to break it, that is to describe a method which results in collisions.

Universal hashing was designed to find a way out of this dilemma, at least for certain applications. Just like for hash functions themselves there are numerous variants. Let us fix notation:

22.1 Definition. *An $(n, M)_q$-array is an array (a matrix) with n rows and M columns, where each entry is taken from an alphabet $\mathcal{A}$ of size q. We also interpret each row of the array as a mapping $f : X \longrightarrow \mathcal{A}$, where $|X| = M$ (the elements of X parametrize the columns). In this perspective we speak of an $(n, M)_q$-***hash family***.*

The idea behind universal hashing is to use a carefully chosen family of hash functions instead of a single hash function. Whenever a hash function has to be applied it will be chosen **at random** from the hash family. The expected behaviour of this method is determined by structural properties of the hash family. As terms like **probability** and **sample space** come in naturally in this context, we review these basic notions of probability theory.

22.2 Definition. *Let Ω be a finite set. For every $x \in \Omega$ let a nonnegative number $p(x)$ be given such that*

$$\sum_{x \in \Omega} p(x) = 1.$$

We see p as a mapping with domain Ω and real values. Then (Ω, p) is a (finite) **probability space** *or* **sample space.** *Call $p(x)$ the* **probability** *of x.*

The probability of a subset $S \subseteq \Omega$ is defined as $p(S) = \sum_{s \in S} p(s)$.

22.3 Definition. *A sample space Ω is* **uniformly distributed** *if each element of Ω has the same probability $1/|\Omega|$.*

As we intend to choose the rows of our arrays (the functions from our hash family) at random, it is natural to consider the rows as elements of a sample space, with uniform distribution: each of the n rows (functions) has probability $1/n$. The idea that collisions should be unlikely translates into the following requirement: whenever two different columns (elements $x \neq x'$ of the ground set) are fixed and a row f is chosen at random, the probability that $f(x') = f(x)$) should be small. This leads to the following definition:

22.4 Definition. *An $(n, M)_q$-hash family is ϵ-***universal** *for some $0 < \epsilon < 1$ if for any two distinct elements (columns) $x', x \in X$ the number ν of functions (rows) f such that $f(x') = f(x)$ satisfies $\frac{\nu}{n} \leq \epsilon$, in other words: the probability of collision is $\leq \epsilon$.*

Here is a toy example:

01234	01234	01234	01234	01234
01234	12340	23401	34012	40123
01234	23401	40123	12340	34012
01234	34012	12340	40123	23401
01234	40123	34012	23401	12340

We see this as a $(5, 25)_5$-hash family, or as a family of 5 functions from a 25-set to a 5-set. It can be checked that each pair of columns contains at most 1 collision. For example, a collision of the first column and some other column is equivalent with an entry 0 in that other column. As each column except the first contains at most entry 0 we have checked pairs involving the first column. Our array is $\frac{1}{5}$-universal.

It is now easy to interpret this notion from the point of view of the Hamming distance. Pick two columns. A collision is a coordinate (row) where they agree. The Hamming distance is the number of coordinates where there is **no collision.** Let d be the Hamming distance between the two columns. The probability of collision is then $\frac{n-d}{n} = 1 - d/n$. This shows how to interpret the notion of collision from our coding point of view: see the columns as words of a q-ary code of length n and minimum distance d. The array is then ϵ -universal, where $\epsilon = 1 - d/n$. We have established the following general equivalence:

22.5 Theorem. *The following are equivalent:*

- *An ϵ -universal $(n, M)_q$ hash family, and*
- *An $(n, M, d)_q$-code, where $\epsilon = 1 - d/n$.*

In fact, in our toy example we have simply chosen the columns as the codewords of the Reed-Solomon code $\mathcal{RS}(2, 5)$. As $d = 4$, we have $\epsilon = 1 - 4/5 = 1/5$.

The notion of an ϵ-universal hash family really is nothing new for us, but it gives a different interpretation of codes. In this application very small alphabets are not interesting at all. Also, as the ground set X has to be very large, the length n cannot be small. The probability ϵ of collision has to be small. This means that d/n has to be close to 1. It follows that the dimension k of our code (in the linear case) must be relatively small. As $M = q^k$ we see again that q cannot be all that small. Fortunately we know already a source of good long codes over large alphabets: the Reed-Solomon codes. If they should not suffice we can use concatenation of Reed-Solomon codes.

Assume we want to hash documents of 2^{20} bits (about 1 million bits). As we want to be able to hash any document of this length, we have a ground set of size $2^{2^{20}}$. The hashed value should have a length of only 32 bits. This means that the alphabet has size $q = 2^{32}$. Can we use a q-ary Reed-Solomon code $\mathcal{RS}(k, q)$? We need $q^k \geq 2^{2^{20}}$, or $k \geq 2^{15}$. The probability of collision is then $\epsilon < 2^{15}/2^{32} = 2^{-17}$. This describes an $2^{-17}-$universal $(2^{32}, 2^{2^{20}})_{2^{32}}-$hash family. When using Reed-Solomon codes we will always have $n = q$.

Here is an application of $\epsilon-$universal hash families for the purposes of error detection. Assume we send the long message x to the receiver via a rather good channel. Let y be the received message. In most of the cases no errors will occur, hence $y = x$. In order to be able to detect errors (not to correct them) the following method can be used: we choose a function f at random from our $\epsilon-$universal family and send f as well as $f(x)$ to the receiver, using a public channel which will certainly not commit errors. This will not cost all too much as these messages are much shorter than x (in the example above x has 2^{20} bits, whereas f and $f(x)$ are bitstrings of length 32 each). If $f(y) \neq f(x)$, then the receiver knows that a transmission error has happened. He will ask for retransmission in this case. If $f(y) = f(x)$, the message will be accepted as correct. The probability of an error is then limited by ϵ.

Problems

1. Determine the parameters of Reed-Solomon codes when interpreted as $\epsilon-$universal hash families.

2. Describe an $(8, 16)_2$ hash family, which is $\frac{1}{2}$-universal.

3. Find a $\frac{2}{5}$-universal $(10, 81)_3$-hash family.

4. Does a $\frac{1}{2}$-universal $(6, 9)_2$-hash family exist?

5. Construct a $(256, 2^{17})_{256}$ hash family, which is $\frac{1}{16}$-universal.

6. Explain why a code $[4000, 5, 3996]_{4096}$ exists and determine the parameters of the universal hash family derived from it.

Chapter 23

Cyclic codes and the binary Golay code

A fruitful source of good codes is the theory of **cyclic codes.** A code is called cyclic if the cyclic shift of any codeword is a codeword again. As an example consider the binary 23-tuple

$$z_1 = (1,1,0,0,0,1,1,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0)$$

In order to obtain its first cyclic shift we write at first the final entry 0, followed by the 23 first entries:

$$z_2 = (0,1,1,0,0,0,1,1,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,)$$

The next cyclic shift is

$$z_3 = (0,0,1,1,0,0,0,1,1,1,0,1,0,1,0,0,0,0,0,0,0,0,0)$$

and so on. Denote by $\mathcal{C}$ the cyclic binary linear code generated by z_1. It is clear that the 12 first cyclic shifts $z_1, z_2, \ldots, z_{12}$ are linearly independent (why?), but z_{13} is a linear combination of $z_1, z_2, \ldots, z_{12}$. It follows that $\mathcal{C}$ has dimension $k = 12$ and that a matrix with $z_1, z_2, \ldots, z_{12}$ as rows is a generator matrix of $\mathcal{C}$.

23.1 Definition. *The* **binary Golay code** $\mathcal{G}_{23}$ *is the binary linear cyclic code generated by* z_1 *above and its cyclic shifts.*

Here is the generator matrix G :

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

It is a fact that $\mathcal{G}_{23}$ has minimum distance $d = 7$. We cannot prove that here in a systematic way.

23.2 Theorem. *The binary Golay code $\mathcal{G}_{23}$ is a $[23, 12, 7]$-code. It is a perfect code.*

Addition of a parity check bit to each row of G yields a generator matrix of the so-called extended binary Golay code $[24, 12, 8]_2$. It is easily checked that this code $\mathcal{G}_{24}$ is self-dual: $\mathcal{G}_{24} = \mathcal{G}_{24}^{\perp}$. In fact, because of the cyclicity it suffices to check that the first row of the generator matrix is orthogonal to all rows.

All rows of the generator matrix of $\mathcal{G}_{24}$ have weight 8. This suffices to prove that all words of $\mathcal{G}_{24}$ have weights divisible by 4 (see the Problems section). The presence of the all-1-word $\mathbf{1}$ (see the Problems section again) shows that the only possible weights of codewords of $\mathcal{G}_{24}$ are $0, 8, 12, 16, 24$ and that there are as many words of weight 16 as there are words of weight 8. Define A_i to be the number of codewords of weight i in $\mathcal{G}_{24}$. The collection of numbers A_i is known as the **weight distribution.** You are challenged to find the weight distribution of $\mathcal{G}_{24}$. In particular it turns out that $A_8 = 759$.

This can serve as a motivation for a basic structure of modern discrete mathematics, **combinatorial designs.**

23.3 Definition. *Let V be a set of v objects and $\mathcal{B}$ a family of subsets of V, where each such subset (a* **block***) has k elements. We say that $\mathcal{B}$ is a t-design with* **replication number** *λ if the following holds:*

Each set $T \subset V$ of t elements is contained in precisely λ blocks. The parameters of such a t-design are written $t - (v, k, \lambda)$.

We can derive a 5-design from our code $\mathcal{G}_{24}$, as follows: As underlying set use the 24 coordinates. Identify each word of weight i with a subset of cardinality i : those coordinates where the entry is 1. Take as blocks the 759 subsets corresponding to the words of weight 8. The last problem in the Problems section shows that this defines a design $5-(24,8,1)$.

23.4 Theorem. *The* 759 *octads of the extended binary Golay code* $[24,12,8]_2$ *are the blocks of a design* $5-(24,8,1)$. *This is known as the* **large Witt design.**

We have seen that the cyclic code $\mathcal{G}_{23}$ and its lengthening $\mathcal{G}_{24}$ are highly interesting codes. In fact cyclic codes are one of the most widely used families of codes. The traditional theory of cyclic codes uses notation and results from ring theory (rings are an important type of algebraic structure). A different approach uses deeper knowledge of finite fields. We do not want to pursue this here.

A particularly important family of designs are the **projective geometries.** We describe the projective planes $PG(2,q)$: start from an arbitrary finite field $\mathbb{F}_q$. Consider the space $V=\mathbb{F}_q^3$, a 3-dimensional space. Define

- **points:** the one-dimensional subspaces of V,
- **lines:** the two-dimensional subspaces of V.

As V has q^3-1 nonzero vector, and each 1-dimensional space has $q-1$ nonzero vectors, the total number of points is $\frac{q^3-1}{q-1}=q^2+q+1$. The same counting method shows that each line has $\frac{q^2-1}{q-1}=q+1$ points.

The most important property of this geometry is the following: any two different points are in precisely one common line. This fact is clear as any two 1-dimensional spaces generate a 2-dimensional space. We conclude that $PG(2,q)$, the classical projective plane of order q, is a design $2-(q^2+q+1,q+1,1)$.

Design theory is an important branch of modern combinatorial theory.

Problems

1. Show that the all-1-word belongs to the extended binary Golay code $[24, 12, 8]_2$.

2. Let $\mathcal{C}$ be a binary linear code, $\mathcal{C} \subseteq \mathcal{C}^\perp$. Assume there is a basis of $\mathcal{C}$ all of whose elements have weights divisible by 4. Show that all weights of codewords are multiples of 4.

3. Determine the weight distribution of $\mathcal{G}_{24}$.

4. Use the value $A_8 = 759$ to show the following: for each set of 5 of the 24 coordinates of $\mathcal{G}_{24}$ there is exactly one codeword of weight 8 (a so-called **octad**) with entries 1 in those 5 coordinates.

5. Find a picture of the **Fano plane** $PG(2, 2)$.

6. For every prime-power q there is a design $2-(q^2, q, 1)$ (the **affine plane** of order q). Determine the number of blocks.

7. The SETS from Chapter 17 form a 2-design. Find its parameters.

8. The extended binary Golay code $[24, 12, 8]_2$ has $A_{12} = 2576$ (2576 codewords have weight 12). The corresponding subsets are called **dodecads.** The 2576 dodecads form the blocks of a design $2-(24, 12, \lambda)$. What is λ? In fact, this is a 5-design? What is λ now?

Chapter 24

Shannon entropy and the basics of information theory

Recall the definition of a sample space from Chapter 22. Such a (finite) sample space is defined by a probability distribution, non-negative numbers p_i such that $\sum_i p_i = 1$. The entropy is a function, which measures the amount of information contained in a sample space.

24.1 Definition. *Let $p_i,\ i = 1, \dots, n$ be a probability distribution. The* **Shannon entropy** *H is defined by*

$$H(p_1, p_2, \dots, p_n) = -\sum_{i=1}^{n} p_i \cdot log(p_i).$$

Here, as always in information theory, the logarithm is taken base 2. We put $0 \cdot log(0) = 0$. Imagine a source, which at every unit of time emits one of n symbols, called letters, according to the given probability distribution. That is, the probability for letter i to be chosen is p_i. We claim that the entropy function is a measure for the average amount of information per letter generated by this source. The unit of information is the **bit,** the amount of information given by the choice between two equiprobable events: $H(\frac{1}{2}, \frac{1}{2}) = 1$.

24.2 Theorem. *Let $H_n : \mathbb{R}_+^n \longrightarrow \mathbb{R}_+ (n = 2, 3, \dots)$ be a family of functions satisfying the following properties:*

- *Each H_n is continuous.*

- *$H_2(1/2, 1/2) = 1$ (one bit).*
- *There is a continuous monotonely increasing function $A(x) : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ such that $A(n) = H_n(1/n, \ldots, 1/n)$.*
- $$H_n(p_1, p_2, \ldots, p_n) = H_2(p_1, 1-p_1) + (1-p_1) \cdot H_{n-1}(\frac{p_2}{1-p_1}, \ldots, \frac{p_n}{1-p_1}).$$

  Here the numbers p_i describe a probability distribution.

Then $H_n(p_1, \ldots, p_n)$ is the entropy function.

This theorem justifies why we consider the entropy function from Definition 24.1 as the measure of information we were looking for. The proof is not hard, but we don't want to do that here.

The whole idea behind the definition of entropy is in the last property of Theorem 24.2. It says that the average amount of information contained in the choice between certain probabilities according to a given distribution is not changed when the information is revealed in various steps. For example,

$$H(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}) = \frac{1}{2} + \frac{1}{4}2 + \frac{1}{8}3 + \frac{1}{8}3 = 1.75.$$

It is clear that this entropy should be larger than $H(\frac{1}{2}, \frac{1}{2}) = 1$. Also, it should be expected that the maximum entropy given by a sample space of n points is obtained by the uniform distribution, and hence is $H(\frac{1}{n}, \frac{1}{n}, \ldots) = log(n)$. In case $n = 4$ this yields a maximum entropy of $log(4) = 2$. Our entropy in the above example is indeed $1.75 < 2$.

The case of a sample space of just 2 points plays a particular role in coding theory.

24.3 Definition. *The* **binary entropy function** *is defined by*

$$h(x) = -xlog(x) - (1-x)log(1-x)$$

for $0 \leq x \leq 1/2$.

Observe that $h(x) = H(x, 1-x)$ is indeed a special case of the Shannon entropy. We have $h(0) = 1, h(1/2) = 1$, and h is increasing on the interval $[0, \frac{1}{2}]$.

There is a close relationship between the binary entropy function and binomials. As a tool we use the Stirling formula, which is frequently used in calculus and statistics.

24.4 Lemma. *The* **Stirling formula** *for factorials states that*

$$n! \sim \sqrt{2\pi n}(n/e)^n,$$

meaning that the factor approaches 1 for $n \to \infty$.

A proof is in [4]. Let now $n, m \to \infty$ such that $\frac{m}{n} \to p$. We want to check how the binomial coefficient $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ behaves. By the Stirling formula we have

$$lim(\frac{1}{n}log(\binom{n}{m})) = lim(log(n) - \frac{m}{n}log(m) - \frac{n-m}{n}log(n-m)) =$$

$$lim(-\frac{m}{n}log(\frac{m}{n}) - \frac{n-m}{n}log(\frac{n-m}{n})) = h(p).$$

We have seen the following:

24.5 Theorem. *Let $n, m \to \infty$ such that $\frac{m}{n} \to p$. Then*

$$lim(\frac{1}{n}log(\binom{n}{m})) = h(p),$$

in other words

$$\binom{n}{m} \sim 2^{nh(p)}.$$

1. The entropy function (Shannon entropy) $H(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{n} p_i \cdot log(p_i)$ (where log is base 2) is a measure for the average amount of information per symbol of a source, which generates n symbols with respective probabilities p_i.
2. The binary entropy function is $h(x) = H(x, 1-x)$.
3. If $\frac{m}{n} \to p$, then $\binom{n}{m} \sim 2^{nh(p)}$.

Problems

1. What is the relationship between $H(p_1, p_2, p_3, p_4)$ and $H(\frac{p_1}{2}, \frac{p_1}{2}, \frac{p_2}{2}, \frac{p_2}{2}, \frac{p_3}{2}, \frac{p_3}{2}, \frac{p_4}{2}, \frac{p_4}{2})$?

2. What is larger (contains more information), $H(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$ or $H(\frac{1}{3}, \frac{1}{3}, \frac{2}{9}, \frac{1}{9})$? Compute both entropies and compare.

3. Compare the following experiments:

   - **A:** Three possible events occur with probabilities $0.5, 0.25, 0.25$
   - **B:** Three possible events occur each with probability $1/3$.

   Which of these experiments yields a larger amount of information? Compute the entropies and compare.

4. Same type of problem as before, with

   - **A:** Five possible events occur with probabilities $\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}$.
   - **B:** Three possible events occur with probabilities $\frac{1}{2}, \frac{1}{3}, \frac{1}{6}$.

Chapter 25

The Gilbert-Varshamov bound

The Gilbert-Varshamov bound proves the existence of certain good linear codes without actually constructing these codes. In this sense it is non-constructive. It is based on a simple counting argument.

Recall the volume $V_q(i,n) = \sum_{j=0}^{i} \binom{n}{j}(q-1)^j$ of the ball of radius i as introduced in Chapter 6. Let $\mathcal{C}$ be a code $[n, k-1, d]_q$. As $\mathcal{C}$ has q^{k-1} elements it follows that if $q^{k-1}V(d-1,n) < q^n$, then there is a vector $v \in \mathbb{F}_q^n$, which has distance $\geq d$ from every code-word $\in \mathcal{C}$. The code generated by $\mathcal{C}$ and v has dimension k and minimum distance d. In order to construct $\mathcal{C}$ we can use the same argument, based on a code with dimension $k-2$. The condition is then automatically satisfied. We see that we can start from the 0-code and apply the same argument, constructing codes of increasing dimension as long as the condition $q^{k-1}V(d-1,n) < q^n$ is satisfied. We have proved the following:

25.1 Theorem (Gilbert-Varshamov bound). *If* $V_q(d-1,n) < q^{n-k+1}$, *then a* q*-ary linear code with parameters* $[n,k,d]$ *exists.*

We can use the Gilbert-Varshamov bound to obtain an **asymptotic** existence theorem. What does that mean?

Let $\mathcal{C}$ be a code $[n,k,d]_q$. Recall $\delta = d/n$, the **relative minimum distance** and $R = k/n$ the **information rate,** from Chapter 8 and Chapter 9. It was the basic idea of block coding to encode k-tuples as n-tuples in such a way that error-correction is possible (because d is large). We are therefore interested in families of codes such that d is relatively large with respect to n (hence δ is bounded away from 0) while keeping R as close to 1 as possible.

25.2 Definition. *A pair (δ, R) is* **asymptotically reachable** *by linear q-ary codes if there is an infinite family of such codes with length $n_i \longrightarrow \infty$, relative minimum distances δ_i and information rates R_i such that $lim_i \delta_i \geq \delta$ and $lim_i R_i \geq R$.*

The asymptotic existence problem for linear q-ary codes is then the following: let $\delta \leq 1$ and $R \leq 1$. Decide if the pair (δ, R) is asymptotically reachable or not. When δ is given we want R as large as possible. When δ increases the maximum R has to decrease. One of the problems shows that rate $R > 0$ can be reached only if $\delta < (q-1)/q$. This is an upper bound (a pessimistic statement). It excludes pairs (δ, R) (such a pair cannot be reached if $\delta \geq (q-1)/q$ and $R > 0$). The Singleton bound from Chapter 18 yields another such bound. In fact, the Singleton bound states $k + d \leq n + 1$. Divide by $n : R + \delta \leq 1 + \frac{1}{n}$.

25.3 Theorem (asymptotic Singleton bound). *If (δ, R) is asymptotically reachable by q-ary linear codes, then $R + \delta \leq 1$.*

Let us derive an existence theorem (an optimistic statement) from the GV-bound: the GV-bound guarantees the existence of a code $[n, k, d]_q$ if

$$\sum_{i=0}^{d-1} \binom{n}{d-1} < q^{n-k+1}.$$

In order to obtain an asymptotic statement take log_q on both sides, divide by n and take the limit. It is clear (why?) that the largest term for $i = d-1$ is dominating on the left side: we can replace the left side by this term and obtain the same asymptotic contribution. We know from the preceding chapter that $\binom{n}{d-1} \sim 2^{nh(\delta)}$. This yields a contribution $log_q(2)h(\delta)$. The second factor $(q-1)^{d-1}$ yields an asymptotic contribution $\delta log_q(q-1)$, the right side yields $1 - R$. We have proved the following **asymptotic Gilbert-Varshamov bound:**

25.4 Theorem. *If $\delta \leq (q-1)/q$ and*

$$R \leq 1 - log_q(2)h(\delta) - \delta log_q(q-1),$$

then (δ, R) is asymptotically reachable.

1. The **Gilbert-Varshamov bound** is an elementary nonconstructive existence theorem for linear codes.
2. Most of the bounds on codes derived earlier yield asymptotic upper bounds (nonexistence theorems) in terms of the **relative minimum distance** $\delta = d/n$ and the **information rate** $R = k/n$ of linear codes $[n,k,d]_q$.
3. The Gilbert-Varshamov bound yields a nonconstructive asymptotic existence theorem.

Problems

1. Determine the maximum dimension k such that the GV-bound guarantees the existence of a code $[8,k,4]_2$.
2. Determine the maximum dimension k such that the GV-bound guarantees the existence of a code $[6,k,4]_3$.
3. Prove the following variant of the Gilbert-Varshamov theorem: If $V_q(d-2,n-1) < q^{n-k}$, then $[n,k,d]_q$ exists. Moreover every code $[n-1,k-1,d]_q$ can be embedded in a code $[n,k,d]_q$. Hint:start from the dual code.
4. Let $\mathcal{C}$ be a code $[n,k,d]_q$. Prove that $\frac{d}{n} \leq (q-1)q^{k-1}/(q^k-1)$. This can be done by a simple counting argument. Conclude that if (δ, R) is asymptotically reachable by linear codes and $R > 0$, then $\delta \leq (q-1)/q$.
5. Let $h(x)$ be the binary entropy function. Determine the derivative $h'(x)$.
6. The q-ary entropy function is defined as

$$H_q(\delta) = log_q(2)h(\delta) + \delta log_q(q-1).$$

In particular $H_2 = h$. The right side of the asymptotiv GV-bound is $1 - H_q(\delta)$.

Prove that H_q is increasing for $0 \leq \delta \leq (q-1)/q$. Determine $H_q((q-1)/q)$.

7. Determine the maximum dimension k such that the Gilbert-Varshamov bound guarantees the existence of a code $[10, k, 5]_4$. Show your calculation.

Chapter 26

3-dimensional codes and projective planes

In this chapter we want to learn how to interpret linear codes geometrically. Starting point is a generator matrix of the code. We limit ourselves to the case of 3-dimensional codes. The underlying geometry is the projective plane $PG(2,q)$, see Chapter 23. Recall the basic data: we call **points** of $PG(2,q)$ the 1-dimensional subspaces of $\mathbb{F}_q^3$, the 2-dimensional subspaces are called **lines.** There are q^2+q+1 points and equally many lines. The lines form the blocks of a design $2-(q^2+q+1, q+1, 1)$ (in particular each line has $q+1$ points). The smallest projective plane is $PG(2,2)$ (7 points, 7 lines, each line has 3 points, each point is on 3 lines, each pair of points is on precisely one line). This binary projective plane $PG(2,2)$ is also known as the **Fano plane.**

Each nonzero vector (x,y,z) of $PG(2,q)$ generates a 1-dimensional subspace, hence a point. We can label the points by these triples. The difference between triples and points is that triples, which are scalar multiples of each other, define the same point. The Fano plane is depicted in Figure 26.1.

As an example consider the binary code with generator matrix

$$G = \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ \hline \end{array}$$

Here is the link to geometry: we consider the **columns** of G as points of $PG(2,2)$. In this case each point of the Fano plane occurs precisely once as

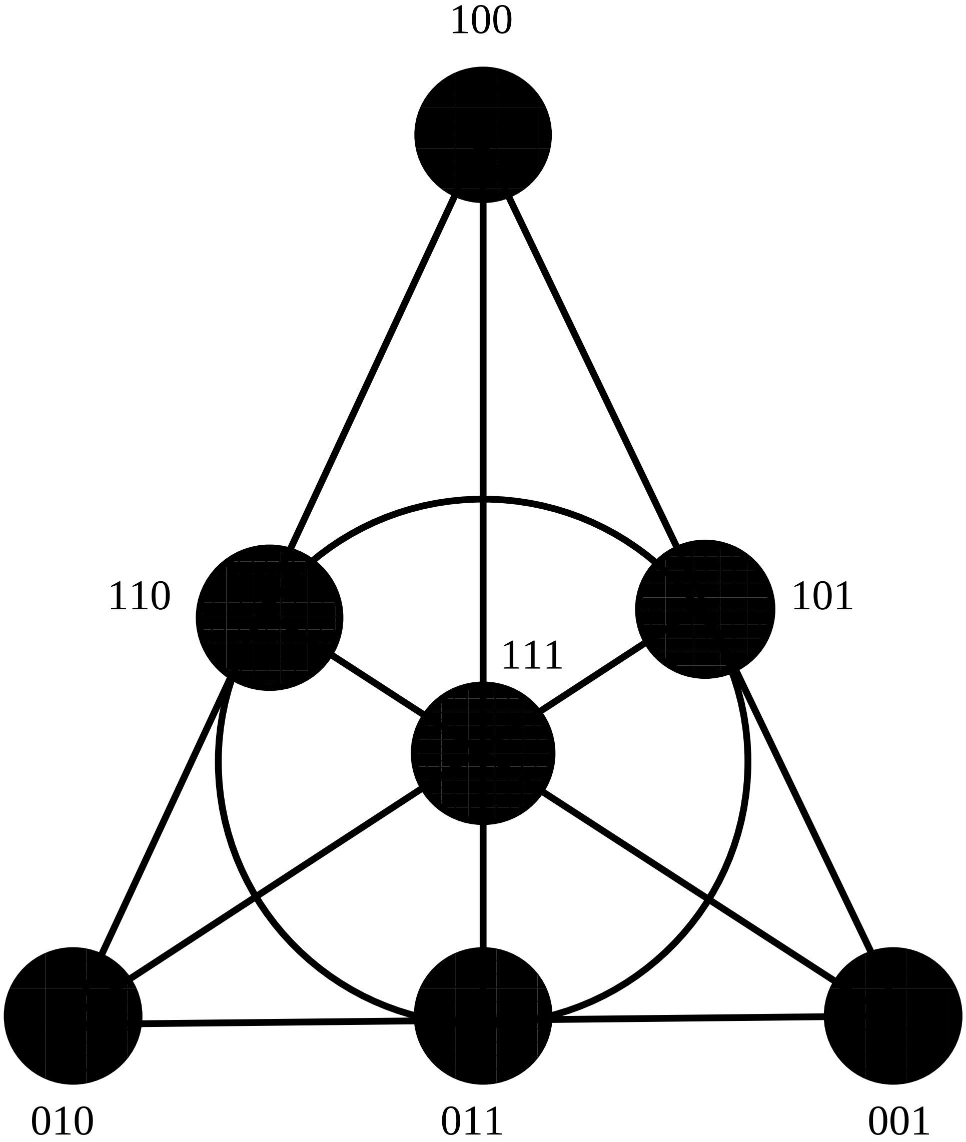

Figure 26.1: **The Fano plane**

a column. A geometer would identify this code $\mathcal{C}$ (or this generator matrix) with the Fano plane itself. What is the minimum distance d of this code $[7,3,d]_2$, or, more generally, what are the weights of codewords?

Denote the rows of G as z_1, z_2, z_3. The codewords are the linear combinations $\lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3$. Fix a coordinate. The corresponding column of G is a point P of $PG(2,2)$. When will our codeword have entry 0 in column P? Let $P = (x,y,z)$. Entry 0 means $\lambda_1 x + \lambda_2 y + \lambda_3 z = 0$, or $(\lambda_1, \lambda_2, \lambda_3) \cdot P = 0$.

Let us sum up: each coordinate is indexed by a point P of $PG(2,2)$ (the corresponding column of the generator matrix). Each nonzero codeword is given by a nonzero triple $(\lambda_1, \lambda_2, \lambda_3)$ (the codeword is $\lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3$). This codeword has entry 0 in this coordinate if and only if the dot product $(\lambda_1, \lambda_2, \lambda_3) \cdot P = 0$. The dot product represents one nontrivial linear equation on $P = (x,y,z)$. These points P are therefore precisely the points of a 2-dimensional subspace, in other words of a **line.**

Sum up again: each nonzero codeword defines a line l (scalar multiples define the same line). Each coordinate is indexed by a point P. The corresponding entry of the codeword vanishes ($= 0$) if $P \in l$. The weight of the codeword is therefore the number of points, **which are not on** l.

This description is true for any 3-dimensional code. In our example the 7 coordinates are described by the 7 points of the Fano plane. As each line has 3 points, there are 4 points outside each line. We have seen that our code is a $[7,3,4]_2$-code, more precisely: each nonzero word of our code has weight 4.

We know this code. It is the Simplex code $\mathcal{S}_3(2)$ from Chapter 11, for general q see Chapter 16.

In general there is no reason why points of $PG(2,q)$ should not occur several times as columns of a generator matrix. We should therefore describe every 3-dimensional code by a weight function w defined on the points of $PG(2,q)$, where $w(P)$ tells us how many columns of a generator matrix give point P. The length n of the code is then $n = \sum_P w(P)$. More importantly we understand how to express the minimum distance d (and more generally all weights of codewords) geometrically: For each line l define the weight of l by

$$w(l) = \sum_{P \in l} w(P).$$

The weights of codewords are then the numbers $n - w(l)$, where l varies through the lines, and the minimum distance is therefore $d = n - Max\{w(l)\}$.

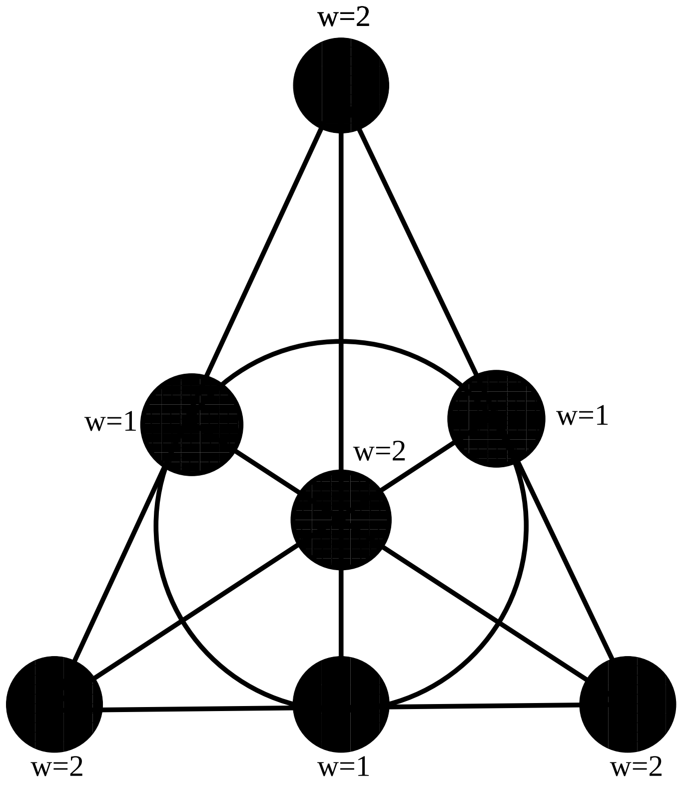

Figure 26.2: **A binary code [11,3,6]**

26.1 Example. *The properties of the Simplex codes are particularly easy to understand in this geometric language. The 3-dimensional Simplex code by definition has length q^2+q+1. Each point of $PG(2,q)$ appears as column once ($w(P)=1$). This means that every line has weight $q+1$ and that every nonzero codeword has weight $(q^2+q+1)-(q+1)=q^2$. In particular the code parameters are $[q^2+q+1,3,q^2]_q$.*

Here is another binary example. We start from the Fano plane and use the weights given in Figure 26.2. The length of the code is the sum of all weights: $n=11$. Also, we see that the lines have weights ≤ 5. This shows that we have a code $[11,3,6]_2$. A corresponding generator matrix is obtained by writing each point P as often as a column as $w(P)$ indicates ($w(P)=0$ means that this point does not occur, $w(P)=1$ means it occurs once, $\ldots$). In our case we obtain the following generator matrix

1	1	0	0	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	0	1	1
0	0	0	0	1	1	1	1	1	0	1

3-dimensional codes and congestion-free networks

Consider the following scenario: There are n network sites, to be connected by broadcast media. A **link** or **bus** is a subset of the sites. In order to avoid congestion it is required that every two sites appear together on at least one link. Each site is equipped with a limited number of communication ports and hence can appear on at most some fixed number r of the links. Also, each link cannot accomodate more than k sites. This construction problem for network topologies has been studied extensively. For an introduction, recent literature and combinatorial-geometric links see [3].

Assume there is a code $[n, 3, n-k]_q$. We write a generator matrix and identify the columns with points $P \in PG(2,q)$, each point appearing with multiplicity $w(P)$. Identify the coordinates with the sites. We construct q^2+q+1 links, determined by the one-dimensional subspaces $\langle v \rangle$. The link defined by $\langle v \rangle$ consists of the coordinates (sites), where v has vanishing entry. As $d = n-k$, each link has at most k sites, as required. Clearly, each site is on at most $q+1 = r$ links and any two sites are on at least one common link. It can even be shown that this construction is best possible when $r-1$ is a prime-power, at least in an asymptotic sense. As an illustration consider Figure 26.2. This represents a network for 11 sites such that each site is on precisely 3 links, each link has at most 5 sites and any two sites occur in a common link.

1. The general notion underlying the geometric method for 3-dimensional codes is that of **projective planes** $PG(2,q)$ (of **order** q).
2. The **points** of $PG(2,q)$ are the 1-dimensional subspaces of $\mathbb{F}_q^3$.
3. The **lines** of $PG(2,q)$ are the 2-dimensional subspaces of $\mathbb{F}_q^3$.

1. The **geometric method:** view the columns of a generator matrix G as a family of points from $PG(2,q)$.
2. Write $w(P)$ if point P occurs $w(P)$ times as a column of G.
3. w is a function defined on the points of $PG(2,q)$ with non-negative integer values.
4. The length is $n = \sum_P w(P)$.
5. For every line l let $w(l) = \sum_{P \in l} w(P)$.
6. The minimum distance is $d = n - max_l w(l)$.

Problems

1. Construct a code $[14, 3, 8]_2$ by the geometrical method.

2. Describe the construction of codes $[q^2, 3, q^2 - q]_q$ by the geometric method.

3. Under which condition on the weight function w will the code really have dimension 3?

4. The **projective line** $PG(1, q)$ consists of the $q + 1$ one-dimensional subspaces (**points**) of $\mathbb{F}_q^2$. Describe the parameters of codes $[n, 2, d]_q$ using geometric language.

5. Given d, determine the smallest n such that a code $[n, 2, d]_q$ exists.

6. Write the generator matrix of a code $[10, 2, 8]_7$. and show that the parameters are as claimed.

7. Find a description of 4-dimensional codes, which is analogous to our geometric description of 3-dimensional codes.

8. Find a generator matrix of a code $[17, 3, 9]_2$.

9. Imagine you have 28 points in $PG(2, 8)$ such that no line contains more than 4 of these points (they exist). Use these points as columns of a generator matrix.

   What are the parameters of the code? Give a brief explanation.

Chapter 27

The end of triviality

Consider our basic coding-theoretic problem of determining, for which values of the parameters linear codes $[n, k, d]_q$ will exist. We ask what is the smallest dimension k and what is the smallest distance d such that the answer is completely known for every field $\mathbb{F}_q$. Dimension 1 is really trivial. In dimension 2 it is relatively easy to give a complete answer, see the Problems section of the preceding chapter. It may come as a surprise that dimension 3 is not trivial in that sense. In fact, there are many constellations where it is not known if a code $[n, 3, d]_q$ does exist. Even worse, there is little hope this problem will ever be completely solved. The geometric description helps us understand why this is the case.

Concentrate on 3-dimensional codes, which are described by weight functions in the sense of the preceding chapter such that only weights 0 or 1 occur. These weight functions can be obviously identified with subsets of $PG(2, q)$. The corresponding codes are called **projective** codes. By the geometric description a projective code $[n, 3, n-\nu]_q$ is equivalent to a set of n points in $PG(2, q)$ such that no line contains more than ν of these points. For given ν we want to maximize n. This is a famous geometric problem and it is nowhere near solution.

How about minimum distance d? Clearly $d = 2$ is trivial. We have the sum-zero codes $[n, n-1, 2]_q$. The maximum n for which a code $[n, n-k, 3]_q$ exists is $n = (q^k - 1)/(q-1)$ (see Chapter 16, the Hamming codes). It follows that distance 3 still is trivial. Also, because of the parity check, $d = 4$ is trivial in the binary case (see Chapter 19, the extended binary Hamming codes).

Once again, how does that look like in practice? Consider the smallest n such that $[n, 7, 4]_2$ exists. Because of puncturing and the parity check this is equivalent to $[n-1,7,3]_2$. By the principle of duality this means that we need a check matrix all of whose columns are nonzero and different. These columns are elements of $\mathbb{F}_2^{n-8}$. We need $2^{n-8}-1 \geq n-1$. The smallest n for which this is true is $n=12$. This shows that $[12,7,4]_2$ exists, whereas $[11,7,4]_2$ does not exist.

The problem of parameters $[n,k,4]_q$ is highly nontrivial for every $q>2$. In order to understand why this is so the geometric description is helpful again. The dual of a code $\mathcal{C}$ with parameters $[n,k,4]_q$ is a code $[n,n-k]_q$ of strength 3, in other words the columns of a check matrix H have the property that no three are linearly dependent. We have studied the geometric description only in the case of dimension 3. Clearly this can be generalized to arbitrary dimension. Again the 1-dimensional subspaces are the points and the 2-dimensional subspaces are the lines of these **projective geometries.** Our check matrix H gives us n points such that no three are on a line. Such sets are called **caps.** Finding the largest cardinality n of caps in projective spaces is an old and hard problem, and it is equivalent to determine when codes $[n,k,4]_q$ exist. In the ternary case the first open problem occurs when $n-k=7$. A code $[112,105,4]_3$ is known (in geometric language this is *the double of the Hill cap in projective* 6-*space*), the existence of $[113,106,4]_3$ is in doubt. Over $\mathbb{F}_4$ the smallest open case occurs when $n-k=6:[126,120,4]_4$ is known (the *Glynn cap*), $[127,121,4]_4$ is in doubt.

Problems

1. Prove that a code is projective if and only if its dual has minimum distance >2.

Chapter 28

The last chapter

The origins of coding theory are linked to the problem of reliable communication over noisy channels. This is a typical problem of the discipline now called **Information Theory.** Both disciplines, **Coding Theory** and **Information Theory,** originated with Claude Shannon's famous 1948 paper [12]. It contains the celebrated **channel coding theorem,** which states roughly that good long codes are guaranteed to exist, without giving a clue how to construct them. Closely related is the development of **Cryptography**. Its aim is to ensure reliable communication in the presence of ill-willed opponents. These problems are rather different. In the coding theory scenario we have to overcome a technical problem (the shortcomings of a communication channel), whereas in Cryptography we have to beat an opponent. Nonetheless the mathematical tools used in these two areas have a large intersection. Historically the development of both disciplines was boosted by the efforts of World War II. Another famous paper of Claude Shannon, [13] from 1949, is perceived as the origin of modern cryptography.

Progress in coding theory was slow but steady. An important development was the theory of **cyclic codes,** which is traditionally couched in the language of ring theory. This attracted the attention of pure mathematicians. A different, more direct, approach is in [2]. Cyclic codes are particularly useful because they admit a fast decoding algorithm.

The ties between coding theory and several areas of pure mathematics have grown stronger all the time. The most important insight goes back to the early 1980s. It is the discovery, by Goppa and Manin, of a close link between linear codes and **algebraic curves** (in algebraic language **function fields**). Algebraic curves are objects of the mainstream mathematical disciplines **number theory** and **algebraic geometry.** They have a long and rich history. The observation by Goppa&Manin makes it possible to use these number-theoretic tools for the construction of codes. The corresponding codes are known as **algebraic-geometric codes** (AG-codes).

Clearly AG-codes need some background. This topic was out of reach for our course. Cyclic codes need just a little bit more knowledge on finite fields than we covered. We could have introduced to the theory of cyclic codes, but it would have eaten up just too much time and space. A graduate level course in coding theory typically devotes half of the time to the study of cyclic codes.

There is a close link between coding theory and **combinatorics.** We have seen that **block designs** can be derived from codes (the Witt design in Chapter 23). Block designs are important objects of modern discrete mathematics. For more information see the **CRC Handbook of Combinatorial Designs** [5].

The only general bounds we used are the Singleton bound and the Hamming bound. There is in fact a multitude of bounds, each of which is better than all the others in certain parameter ranges. A general algebraic mechanism for the derivation of bounds is related to **orthogonal polynomials** (in the case of codes these are the **Kravchouk polynomials**) and **linear programming.** There is a general **linear programming bound.** Both the Singleton bound and the Hamming bound are special cases of the LP bound.

The study of self-dual codes (we have seen the extended binary Golay code $[24, 12, 8]_2$, the extended Hamming code $[8, 4, 4]_2$ amd the extended ternary Golay code $[12, 6, 6]_3$) uses **invariant theory,** a classical subdiscipline of **group theory.**

There is a very close relationship between coding theory and **Galois geometry.** The basic objects of Galois geometry are **affine and projective spaces** (for an introduction see [8]). Linear codes can be equivalently described as sets of points in projective spaces. In fact the theory of linear codes and Galois geometry are two ways of studying the same questions. In many situations the geometric language is more appropriate than the algebraic approach. We have seen the special case of this mechanism when the codes are 3-dimensional.

In this text we stressed duality. The dual of a linear code with minimum distance d is an orthogonal array of strength $d-1$. **Orthogonal arrays** were originally defined in the framework of design of experiments, in **statistics.**

The same is true of block designs. Observe that the defining conditions of an orthogonal array and of block designs both are uniformity conditions. They look very similar. Orthogonal arrays can be interpreted as families of **random variables** (=functions defined on sample spaces, see Chapter 22), which satisfy certain statistical independence conditions. Such families of random variables are heavily used not only in statistics but also in the theory of **algorithms.** Whenever we construct a good linear code we also obtain such a statistical object.

This leads us to applications of codes. Traditionally coding-theorists are biased towards the information-theoretic application that we have encountered so often in this text. This fixation is no longer justified. We have just mentioned applications in statistics and algorithms. A plethora of applications have surfaced in theoretical computer science, in particular in cryptography. We have seen universal hash families in Chapter 22. A rather surprising application concerns quasi-Monte Carlo algorithms (related to uniformly distributed point sets in Euclidean space), in particular numerical integration and the pricing of exotic options.

If you construct a linear code and want to compare it with what is known, there is A.E.Brouwer's interactive data base of parameters $[n, k, d]_q$ of linear codes

`http://www.win.tue.nl/~aeb/voorlincod.html`

For $q \leq 9, n$ not too large and $k \leq n$ it gives you two numbers: the largest d that is known (the lower bound) and the largest d which may theoretically be possible.

An early classic among the textbooks is Berlekamp's book [1], which presents in particular an excellent introduction to the traditional theory of cyclic codes. The 1977 book [10] by McWilliams and Sloane is considered as something like the bible of traditional coding theory. A more recent introduction is van Lint's book [16], a relatively short and dense text which helped a great deal in attracting pure mathematicians to the area. Stichtenoth's book [14] is a self-contained introduction to the theory of algebraic function fields in one variable (equivalently: algebraic curves) and to the codes derived from them. Among the undergraduate textbooks we mention Pless [11] and Hill [7]. Hill's book is the first book of its kind which presents an introduction to the geometric approach to codes.

Bibliography

[1] E R Berlekamp: *Algebraic Coding Theory*, McGraw-Hill, New York 1968.

[2] J.Bierbrauer: The theory of cyclic codes and a generalization to additive codes, submitted for publication in *Designs, Codes and Cryptography.*

[3] Charles J. Colbourn: *Projective planes and congestion-free networks,* submitted for publication.

[4] W.Feller: An Introduction to Probability Theory and Its Applications I, Wiley 1950.

[5] The CRC Handbook of Combinatorial Designs, C.J. Colbourn and J.H.Dinitz (eds.), CRC Press 1996.

[6] Hedayat, N.J.A.Sloane and J.Stufken: *Orthogonal Arrays: Theory and Applications,* Springer 1999.

[7] R.Hill: *A first course in coding theory,* Clarendon 1986.

[8] J.W.P. Hirschfeld: *Projective Geometries over Finite Fields,* Oxford University Press ^{2}1998.

[9] D.B. Jaffe: *Looking inside codes,* manuscript.

[10] F. J. MacWilliams, N. J. A. Sloane. *The Theory of Error-Correcting Codes,* North-Holland, 1977.

[11] Vera Pless: *The Theory of Error-Correcting Codes,* Wiley 1989.

[12] C.E. Shannon: *A mathematical theory of communication, Bell System Technical Journal* **27**(1948),379-423,623-656.

[13] C.E. Shannon: *Communication Theory of Secrecy Systems, Bell System Technical Journal* **28**(1949),656-715.

[14] Henning Stichtenoth: *Algebraic Function Fields and Codes,* Springer 1993.

[15] D.R.Stinson: *Cryptography - Theory and Practice,* CRC Press 1995.

[16] J.H.van Lint: *Introduction to Coding Theory,* Springer Verlag 1982.