Franz Lemmermeyer

# Error-correcting Codes

February 16, 2005

 $\mathbf{2}$ 

# Table of Contents

Oh	my	God – It's full of codes!	1
1.	Mo	tivation	5
	1.1	ISBN - An Error-detecting Code	5
	1.2	Codes with Variable Word Length	8
	1.3	Hamming	9
	1.4	Repetition Codes	10
	1.5	A Simple Linear Code	12
	1.6	Hamming's Square Codes	13
	1.7	Hamming's [7, 4]-code	14
	1.8	Where are they?	16
2.	Bas	sic Concepts	17
	2.1	Codes	17
	2.2	Hamming Distance	18
	2.3	Vector Spaces	21
	2.4	Subspaces	23
	2.5	Linear Codes	26
	2.6	Generator and Parity Matrix	27
	2.7	Hamming Codes	30
	2.8	Perfect Codes	31
3.	Wo	rking with Linear Codes	37
	3.1	Quotient Spaces	37
	3.2	Decoding Linear Codes	39
	3.3	More on Parity Check Matrices	41
	3.4	Probability	42
	3.5	New Codes from Old	52
	3.6	The Quality of Codes	54
4.	Fin	ite Fields	57
	4.1	Simple Examples of Finite Fields	58
	4.2	Construction of Finite Fields	60
	4.3	Galois Theory of Finite Fields	62
		•	

	4.4	The Quadratic Reciprocity Law	66							
5.	Cyclic Linear Codes									
	5.1	Cyclic Codes	71							
	5.2	The Algebra behind Cyclic Codes	72							
	5.3	Reed-Muller Codes	83							
	5.4	Quadratic Residue Codes	84							
	5.5	BCH Codes	85							
	5.6	Reed-Solomon Codes	85							
	5.7	Algebraic Geometry Codes	88							
6	Oth	er Classes of Codes	01							
0.	6 1	Reed-Muller Codes	91 91							
	6.2	Reed-Solomon Codes	91							
	6.3	Algebraic Geometry Codes	92							
7	Sph	ere Packings and Codes	03							
7.	$_{7.1}^{\mathbf{Sph}}$	ere Packings and Codes	93 93							
7.	<b>Sph</b> 7.1 7 2	ere Packings and Codes Lattices	93 93 93							
7.	Sph 7.1 7.2 7.3	ere Packings and Codes Lattices Codes and Lattices Sphere Packings	93 93 93 94							
7.	Sph 7.1 7.2 7.3 7 4	ere Packings and Codes Lattices Codes and Lattices Sphere Packings The Leech Lattice	93 93 93 94 94							
7.	Sph 7.1 7.2 7.3 7.4 7.5	ere Packings and Codes         Lattices         Codes and Lattices         Sphere Packings         The Leech Lattice         Finite Simple Groups	93 93 93 94 94 94							
7.	Sph 7.1 7.2 7.3 7.4 7.5	ere Packings and Codes Lattices Codes and Lattices Sphere Packings The Leech Lattice Finite Simple Groups	<ul> <li>93</li> <li>93</li> <li>93</li> <li>94</li> <li>94</li> <li>94</li> <li>94</li> </ul>							
7. A.	Sph 7.1 7.2 7.3 7.4 7.5 Alg	ere Packings and Codes         Lattices         Codes and Lattices         Sphere Packings         The Leech Lattice         Finite Simple Groups	<ul> <li>93</li> <li>93</li> <li>93</li> <li>94</li> <li>94</li> <li>94</li> <li>95</li> <li>95</li> </ul>							
7. A.	Sph 7.1 7.2 7.3 7.4 7.5 Alg A.1	ere Packings and Codes         Lattices         Codes and Lattices         Sphere Packings         The Leech Lattice         Finite Simple Groups         ebraic Structures         Groups	<ul> <li>93</li> <li>93</li> <li>93</li> <li>94</li> <li>94</li> <li>94</li> <li>94</li> <li>95</li> <li>95</li> <li>95</li> <li>95</li> </ul>							
7. A.	Sph 7.1 7.2 7.3 7.4 7.5 Alg A.1 A.2	ere Packings and Codes         Lattices         Codes and Lattices         Sphere Packings         The Leech Lattice         Finite Simple Groups         ebraic Structures         Groups         Rings	<ul> <li>93</li> <li>93</li> <li>93</li> <li>94</li> <li>94</li> <li>94</li> <li>95</li> <li>95</li> <li>95</li> <li>95</li> </ul>							
7. A.	Sph 7.1 7.2 7.3 7.4 7.5 Alg A.1 A.2 A.3	ere Packings and Codes         Lattices         Codes and Lattices         Sphere Packings         The Leech Lattice         Finite Simple Groups         ebraic Structures         Groups         Rings         Fields	<ul> <li>93</li> <li>93</li> <li>93</li> <li>94</li> <li>94</li> <li>94</li> <li>95</li> <li>95</li> <li>96</li> <li>96</li> </ul>							
7. A.	Sph 7.1 7.2 7.3 7.4 7.5 Alg A.1 A.2 A.3 A.4	ere Packings and Codes         Lattices         Codes and Lattices         Sphere Packings         The Leech Lattice         Finite Simple Groups         ebraic Structures         Groups         Rings         Fields         Vector Spaces	<ul> <li>93</li> <li>93</li> <li>94</li> <li>94</li> <li>94</li> <li>95</li> <li>95</li> <li>96</li> <li>96</li> <li>96</li> </ul>							
7. A.	Sph 7.1 7.2 7.3 7.4 7.5 Alg A.1 A.2 A.3 A.4 A.5	ere Packings and Codes         Lattices         Codes and Lattices         Sphere Packings         The Leech Lattice         Finite Simple Groups         ebraic Structures         Groups         Rings         Fields         Vector Spaces         Algebras	<ul> <li>93</li> <li>93</li> <li>93</li> <li>94</li> <li>94</li> <li>94</li> <li>95</li> <li>95</li> <li>96</li> <li>96</li> <li>97</li> </ul>							

# 1. Motivation

#### 1.1 ISBN - An Error-detecting Code

Almost all the books published today have an International Standard Book Number (ISBN), which consists of four parts; the book 'Codes and Curves' by Judy Walker has the ISBN 0-8218-2626-X. The first string (here just the digit 0) is the country code: the 0 tells us that this book was published in the USA. Here's a short list of the most common country codes:

code	language	country
0	English	UK, US, Australia, NZ, Canada
1	English	South Africa, Zimbabwe
2	French	France, Belgium, Canada, Switzerland
3	German	Germany, Austria, Switzerland
4	Japanese	Japan
5	Russian	Russia, states of the former USSR
7	Chinese	China
88	Italian	Italy, Switzerland

The fact that Switzerland has three different country codes suggests the country code is more of a language code (Switzerland has three official languages).

The second string in an ISBN identifies the publishing company: the 8218 informs us that the book was published by the American Mathematical Society.

The third string is the number that is given to each book the corresponding company publishes. Assuming that the AMS started with book number 0001, Walker's book should be the 2626th book that was published by the AMS since the introduction of the ISBN.

The last digit (the X looks more like a letter than a digit, but wait and see) finally is the one that is of interest to us: it is a check digit. It is computed from the other digits by a simple rule: multiply the digits of your ISBN by 1, 2, 3, ..., and form the sum: the check digits is the remainder of this sum modulo 11. For the ISBN 0-387-94225 we find  $1 \cdot 0 + 2 \cdot 3 + 3 \cdot 8 + 4 \cdot 7 + 5 \cdot 9 + 6 \cdot 4 + 7 \cdot 2 + 8 \cdot 2 + 9 \cdot 5 \equiv 4 \mod 11$ , hence the complete ISBN is 0-387-94225-4.

If the remainder modulo 11 turns out to be 10, we will use X (the letter for 10 used by the Romans) to denote the check digit; this happens for Walker's book.

**Exercise 1.1.** Verify that the check digit for Walker's book is X.

**Exercise 1.2.** Serge Lang's Algebraic Number Theory has two ISBN's: 0-387-94225-4 and 3-540-94225-4. The first one refers to Springer Verlag New York, the second one to Springer Verlag Berlin-Heidelberg. Why do you think Springer Verlag New York was given the code 0-387 instead of just 0-540?

So this is how computing the check digits works. What is it good for? Well, the ISBN was invented at a time when most errors were still made by humans, and not by computers. Suppose you wanted to buy Lang's Algebraic Number Theory; to prevent you from getting Long's Algebraic Numbers instead, the friendly clerk at the book store would include the ISBN in the order and so identify the book beyond doubt. (Nowadays everything's different; yet: if you order Long's Algebraic Numbers at www.half.com, I'll pay you a coffee if you don't get Lang's book instead. If anyone is interested in Lang's book, I have three of them ...). Now when humans write down ten digit numbers, they tend to make mistakes. In order to detect these errors, the check digit was introduced. For example, if you're given the ISBN 0-387-94325-4, you can verify that the formula for the check digit does not give you the right result. Assuming you have done your math correctly, you would assume that there's something wrong with one of the digits (possibly the check digit itself – you never know).

The idea behind the formula for computing the check digit of an ISBN is that check digits computed that way can detect the most common errors made by humans: misreading a single digit, or switching two consecutive digits.

**Exercise 1.3.** Show that the check digit of an ISBN can detect single errors. (Assume  $d_1d_2 \cdots d_{10}$  is a valid ISBN with check digit  $d_{11}$ . Replace  $d_j$  by a digit  $d_j + r$  for some  $r \neq 0$  and show that this would necessarily change the check digit.)

**Exercise 1.4.** Show that the check digit of an ISBN detects inversions (switching two adjacent digits). Show that ISBN's may not detect the switching of digits that are not adjacent.

**Exercise 1.5.** Show that the check digit of an ISBN may not detect two single errors by writing down two valid ISBNs that differ in exactly two positions.

#### Other Check Digit Systems

You can see check digits at work when you buy something online with a credit card: just mistype a digit of your credit card number deliberately, and the order form will inform you immediately that there seems to be something wrong with your number: there's an error-detecting code behind credit card numbers, which can detect (but in general not correct) certain errors.

Here are a few more examples; determine which of these check digits can detect single errors or inversions, and analyze exactly which errors can't be detected:

- American Express Traveler's Checks have an ID number at the bottom; the sum of all digits including the check digit is  $\equiv 0 \mod 9$ . USPS money orders use the same system.
- Airline Tickets have a Stock Control Number with Check digit, which is just the remainder of the 10-digit number modulo 7. The same system is used by FedEx, UPS, Avis, and National Car Rental agencies.
- Bank Checks have 9-digit numbers at the bottom left; if you multiply them alternatively by 7, 3, 9, then the sum should be  $\equiv 0 \mod 9$ .
- Universal Product Codes (UPC) can be found on almost any product you can buy these days. If you multiply the digits alternatively by 1 and 3, starting with the check digit on the right, then the sum should be ≡ 0 mod 10.
- The ZIP bar code can be found on many bulk-mail envelopes (e.g. on business reply cards). Each block of 5 lines represents a digit; the correspondence is

The sum of all digits (ZIP+4 or ZIP+4+ last two digits of address) including the check digit should be  $\equiv 0 \mod 10$ .

The ZIP code has actually error-correcting capability: observe that each digit is represented by a string of 5 binary digits exactly two of which are nonzero. Thus if you come across a string of 5 binary digits with 4 zeros, there must have been an error, and using the check digit you can correct it!

#### Here's an example:

#### 

The first and the last bar define the border; the 50 bars in between encode 10 numbers: ZIP +4 + check digit. For decoding, we partition the bars between the two border bars into blocks of 5; the first block is ||||||, representing 00011, that is, 1. The whole ZIP code is 10117-9652, and the check digit is 8. In fact,  $1+0+1+1+7+9+6+5+2+8 = 40 \equiv 0 \mod 10$ .

The way in which the ZIP code encodes the ten digits is extremeley interesting. We already observed that each codeword has exactly two 1s and three 0s; since there are exactly  $\frac{5!}{3!2!} = \binom{5}{2} = 10$  5-letter words made out of two 1s and three 0s, the codewords exhaust all such 5-letter words. What this means is that if you permute any codeword (say by switching the first and the third digit), you will get another codeword. Letting *C* denote the ZIP code and Aut(*C*) the automorphism group of *C* (the group of all permutation of the five digits that preserve codewords), this means that Aut(*C*) =  $S_5$ , the symmetric group of degree 5. It has 5! = 120 elements, so the ZIP code is highly symmetric!

• Check digits are not always appended to the end of the number. Washington State driver's licences are strings of 12 alpha-numeric characters, the tenth being the check digit. Vehicle Identification Numbers usually have 17 characters, the ninth one being the check digit.

#### Examples.

- The number 67021200988 is an ID on a USPS money order.
- The ID 448721117 is a valid American Express ID.
- The stock control number 2853643659 has check digit 6.

**Exercise 1.6.** Verify the check digits of a bank check, a UPC, and a ZIP bar code of your choice.

### 1.2 Codes with Variable Word Length

All the codes we introduced so far (except for natural languages) have fixed word length, and we shall restrict our attention to such codes for the rest of the semester. It should be remarked, though, that codes with variable word length are important and useful.

The best known code of this type is the Morse code. The fact that the most frequent letters in English, namely E and T, are encoded with single digits is supposed to keep the coded messages as short as possible. One problem with the Morse code is that you need spaces between words; without them, the code 01000110 (or  $\cdot - \cdots - - \cdot$  using the common way Morse code is written) encodes both LEG and RUN (see Table , which is taken from [7, p. 208]).

In the same vein, there's a thing called Huffman coding that encodes ASCII text using a code of variable length, again making codewords for common letters as small as possible. Huffman codes, however, have an additional property: they are uniquely decodable because no codeword is a prefix of another.

Observe that neither Morse nor Huffman codes are error-correcting. While there exist error-correcting codes with variable word length, we will not discuss them in this course.

1. Motivation

character	probability	ASCII	Morse	Huffman
space	0.1859	01000001	space	000
А	0.0642	10000010	01	0100
В	0.0127	10000100	1000	0111111
$\mathbf{C}$	0.0218	10000111	1010	11111
D	0.0317	10001000	100	01011
$\mathbf{E}$	0.1031	10001011	0	101
$\mathbf{F}$	0.0208	10001101	0010	001100
G	0.0152	10001110	110	011101
Η	0.0467	10010000	0000	1110
Ι	0.0575	10010011	00	1000
$\mathbf{J}$	0.0008	10010101	0111	0111001110
Κ	0.0049	10010110	101	01110010
$\mathbf{L}$	0.0321	10011001	0100	01010
Μ	0.0198	10011010	11	001101
Ν	0.0574	10011100	10	1001
Ο	0.0632	10011111	111	0110
Р	0.0152	10100000	0110	011110
$\mathbf{Q}$	0.0008	10100011	1101	0111001101
$\mathbf{R}$	0.0484	10100101	010	1101
$\mathbf{S}$	0.0514	10100110	000	1100
Т	0.0796	10101001	1	0010
$\mathbf{U}$	0.0228	10101010	001	11110
$\mathbf{V}$	0.0083	10101100	0001	0111000
W	0.0175	10101111	011	001110
Х	0.0013	10110001	1001	0111001100
Υ	0.0164	10110010	1011	001111
$\mathbf{Z}$	0.0005	10110100	1100	0111001111

 Table 1.1. Codes for the English Alphabet

#### 1.3 Hamming

The main reference for historical material is [14].

Hamming's undergraduate and graduate training was in pure mathematics, his Ph.D. thesis was on differential equations, and he was part of the Los Alamos atomic bomb project during WWII. He left Los Alamos for the Bell Laboratories in 1946:

I was a pure mathematician – I felt somewhat lost at the place. Every once in a while I got terribly discouraged at the department being mostly electrical engineering.

When Hamming arrived at the Bell Labs, they had a Model V computer there: it occupied 1000 square feet, weighed 10 tons, and could solve systems of 13 linear equations in less than 4 hours.

Hamming had only weekend access to the Model V; there were no supervising personnel present over weekends, so whenever the computer discovered an error in the input, it would simply abandon the job and take the next one. Two weekends in a row I came in and found that all my stuff had been dumped and nothing was done. I was really aroused and annoyed because I wanted those answers and two weekends had been lost. And so I said "Damn it, if the machine can detect an error, why can't it locate the position of the error and correct it?"

The fact that we know about this little incident already tells you that Hamming succeeded in teaching computers not only to detect but also to correct errors. Actually he managed to come up with quite a good code, but before we can explain why his code is good we have to look at bad codes first.

There are a lot more problems here than just the construction of errorcorrecting codes; it is not always desirable to have codes that correct as many errors as possible, because this could be just too time consuming to be practical. What we need are estimates of how good codes have to be given a specific task. Here's what Hamming had to say about this:

The Relay Computer, operating with a self-checking code, stops whenever an error is detected. Under normal operating conditions this amounts to two or three stops per day. However, if we imagine a comparable electronic computing machine operating  $10^5$  times the speed [of the Relay computer] and with elements  $10^3$  times more reliable than relays, we find two to three hundred stops per day.

So not only did Hamming envisage the advent of fast computers, he also predicted Microsoft Operating systems.

## 1.4 Repetition Codes

The simplest possible error correcting code (and one of the worst) is the repetition code. Suppose you want to transmit the following string of digits: 011010001. In order to be able to correct errors, you may send each single digit three times. If no error is made during transmission, the result will be 000111111000111000000000111. Now assume that instead you receive the string 00011111000011000010000111. You can immediately tell that an error must have occurred since this is not a codeword: code words consist of blocks of three equal digits, and here you have two blocks (the third 110, and the seventh 010) that don't have this form. In order to correct the message, you therefore replace 110 by 111 and 010 by 000, and under the assumption that only one error has been made in each of these blocks this must be the original message.

More generally, messages having k bits can be encoded by repeating them r times, creating codewords of length n = kr; the corresponding codes is called the repetition code of type [n, k]. The code described above has type [3, 1]. The repetition code of type [4, 2] has the codewords 0000, 0011, 1100, and 1111.

**Exercise 1.7.** Encode the message 101 using the repetition code of type [3, 1].

**Exercise 1.8.** You receive the message 000110111101 that was encoded using the [3, 1]-repetition code. Locate the errors and correct them.

**Exercise 1.9.** Encode the message 101 with the [4, 1]-repetition code.

**Example.** Let us compute the automorphism group  $\operatorname{Aut}(C)$  of the [4,2]-repetition code  $C = \{0000, 0011, 1100, 1111\}$ . By writing down all possible permutations of 4 objects that preserve C it is straightforward to see that  $\operatorname{Aut}(C)$  is a nonabelian group with 8 elements.

One automorphism of C is given by  $R = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$ . A simple calculation shows that  $R^2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$  transposes the first and the second, as well as the third and the fourth coefficients, which implies that  $R^2$  has order 4, and that  $R^4 = \text{id}$ .

This shows that Aut(C) has a cyclic subgroup of order 4. But we also have the transposition  $S = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$  which is different from id, R,  $R^2$  and  $R^3$ . Thus Aut(C) contains at least the 8 elements of  $G = \{S^a \circ R^b : 0 \le a \le 1, 0 \le b \le 3\}$ . How can we show that there are no others? The first step is proving that G is actually a group. This follows almost immediately from the fact that  $R \circ S = S \circ R^3$ , which allows us to write any product of elements of G in the form  $S^a \circ R^b$ . For example, the product  $(S \circ R) \circ S$  is transformed into  $(S \circ R) \circ S = S \circ (R \circ S) = S \circ (S \circ R^3) = (S \circ S) \circ R^3 = \text{id} \circ R^3 = R^3$ . Here we have used associativity, which holds since we are dealing with composition of permutations, and the composition of maps is always associative.

So now we know that G is a group. Why does this help us? Because this implies that  $G \subseteq \operatorname{Aut}(C) \subseteq S_4$ , where  $S_4$  is the full group of permutations on 4 objects. Since  $S_4$  has 4! = 24 elements, and since the order of a subgroup divides the order of a group, we conclude that  $8 \mid \#\operatorname{Aut}(C) \mid 24$ . Thus either  $\operatorname{Aut}(C) = G$  or  $\operatorname{Aut}(C) = S_4$ . But the last possibility cannot occur: we need only write down a single permutation that does not conserve codewords, for example the transposition of the middle bits, which turns 0011 into 0101, hence is not an automorphism.

Thus  $\operatorname{Aut}(C) = G$ ; in fact, G is isomorphic to the dihedral group  $D_4$  of order 8, because the generators R and S satisfy the relations  $R^4 = S^2 = \operatorname{id}$ ,  $SRS = R^3$  of the dihedral group.

Now  $D_4$  is the symmetry group of a square, and it is actually possible to make this isomorphism concrete: draw a square and label the corners clockwise with 1, 3, 2 and 4.



Then any symmetry of the square induces a permutation of the indices 1, 2, 3, 4, and it is easily checked that each symmetry of the square gives an automorphism of C. For example, the 90°-degree rotation (counterclockwise) of the square corresponds to the permutation R, whereas S is the reflection along the diagonal 34.

**Exercise 1.10.** Show that the automorphism group of the [n, 1]-repetition code is  $S_n$ .

The [3, 1]-repetition code seems like a very good code – it can correct single errors in each block. The reason I called it bad is that there are much better ways of encoding messages. What does better mean? Losely speaking, you have bought the ability to correct single errors by transmitting 3k bits in stead of k bits. If the same thing can be done with a factor of 2 instead of 3 then you can transmit information 1.5 times faster with the same amount of error correcting capability; note, however, that while repeating each bit twice gives you a code that can be transmitted 1.5 times faster than repeating it three times, you also lose the capability of correcting errors.

What we have here is the 'basic conflict' in coding theory: on the one hand, we would like to be able to correct as many errors as possible, on the other hand we want to minimize the number of extra bits we need for error detection and correction.

A simple measure for the quality of a code is the information rate: for a code with w codewords, each of which has length n, the information rate is fiven by  $\frac{\log_2 w}{n}$ . Observe that  $n = \log_2 2^n$ , and that  $2^n$  is the number of all words of length n over  $\mathbb{F}_2$ .

**Exercise 1.11.** Show that the repetition code of type [3,1] has information rate  $\frac{1}{3}$ . What is the information rate of the general repetition code of type [k, r]?

The [3, 1] repetition code has higher information rate than the [4, 1] repetition code, and both have the ability to correct single errors, so the [3, 1]-code is superior. The [2, 1]-repetition code has higher information rate than the code of type [3, 1], but the [2, 1]-repetition code cannot correct errors.

### 1.5 A Simple Linear Code

Here's a simple example of how to achieve this. Instead of encoding the message bit by bit, we take blocks of three bits and encode them as follows:

bits	codeword	bits	codeword
000	000000	011	011011
100	100011	101	101101
010	010101	110	110110
001	001110	111	111000

**Exercise 1.12.** You received the following message encoded by the code described above. Locate the errors and correct them.

011010, 001110, 100001, 011110, 101111.

**Exercise 1.13.** Determine the automorphism group of this code. (Hint: Observe that automorphisms must preserve the weight of codewords, that is, the number of 1's per word.)

The reason why you were able to correct single errors is the following: every codeword differs in at least three places from any other code word. This means that if you change a codeword at a single position, the original word is the only one that differs in at most one position from the word containing an error.

A very nice property of this code is the fact that one need not store the whole dictionary of bits and codewords to work with the code because this is an example of a *linear code*: if v and w are codewords, then so is v + w, where the sum of the vectors is computed coordinatewise, using the addition rules 0 + 0 = 0, 1 + 0 = 0 + 1 = 1, and 1 + 1 = 0. These rules form the addition law of the field with two elements  $\mathbb{F}_2 = \{0, 1\}$ ; the set of all vectors of length 6 form a 6-dimensional vector space over  $\mathbb{F}_2$ , and the codewords form a subspace of dimension 3.

This fact will allow us to do the encoding and decoding using simple linear algebra, which means it can be done very efficiently.

#### 1.6 Hamming's Square Codes

Partition your message into blocks of 4 bits each; write these 4 bits in a square array, and compute the sums of rows and columns modulo 2. For the message 1101 this results in the following array:

1101	$\longmapsto$	1	$1 \mid$		1	1	$\begin{vmatrix} 0 \\ 1 \end{vmatrix}$
		0 1	$\longmapsto$	0	T	1	
		0	T		1	0	1

The codeword is then the word formed out of the three rows, that is, 110011101. For decoding, write the received message as a  $3 \times 3$ -array, and write the upper left  $2 \times 2$ -minor as a 4-bit word again.

What happens if a single error appears in a transmission? Suppose e.g. that you receive the message

$$\begin{array}{cccc} 1 & 0 & 0 \\ 0 & 1 & 1 \\ \hline 1 & 0 & 1 \end{array}$$

Then the parity checks in the first row and the second column don't match, so you know that the element in the first row and the second column must be wrong, and you correct it to a 1. **Exercise 1.14.** The following message was encoded using Hamming's code above. Detect where errors have been made, and correct them. 111110011, 011111110, 100110011.

**Exercise 1.15.** Suppose a single error occurs in one of the check digits. Can this be detected, or will this error lead to 'removing' an error in the original message that wasn't even there?

**Exercise 1.16.** Replace the  $2 \times 2$ -matrices by  $n \times n$ -matrices and develop an error-correcting code by generalizing the  $2 \times 2$ -code above and show that these codes can correct single errors. Determine whether these  $n \times n$ -square codes are linear or not.

**Exercise 1.17.** The information rate of Hamming's  $n \times n$  square code is  $\frac{n^2}{(n+1)^2}$ .

**Exercise 1.18.** Show that Hamming's  $n \times n$  square code with the  $(n+1)^2$ -th digit (the corner check symbol) deleted still can correct single errors; compute its information rate.

# 1.7 Hamming's [7, 4]-code

This code will turn out to be the best single error correcting code with four message digits. To a message of 4 bits, 3 check digits are added in the following way: the bits of the original message are placed at the slots 3, 5, 6 and 7, the three check digits at 1, 2 and 4. The first check digit is the parity of the bits in positions 3, 5 and 7 (these numbers are those odd numbers where the message bits are, or in more fancy terms, have a binary expansion ending in 1), the second check digit is the parity of the bits in positions 3, 6 and 7 (their binary expansions have a 1 in the twos place), and the third check digit is the parity of the bits in positions 5, 6 and 7 (their binary expansions have a 1 in the fours place).

Example: consider the message 1010; the 7-digit codeword is then

$$d_1 d_2 d_3 d_4 d_5 d_6 d_7 = 1011010.$$

The digits  $d_3$ ,  $d_5$ ,  $d_6$  and  $d_7$  come from the original word, whereas  $d_1$ ,  $d_2$  and  $d_4$  are computed as follows:

$$d_1 = d_3 + d_5 + d_7,$$
  

$$d_2 = d_3 + d_6 + d_7,$$
  

$$d_4 = d_5 + d_6 + d_7.$$

In the case at hand we find  $d_1 = 1 + 0 + 0 = 1$ ;  $d_2 = 1 + 1 + 0 = 0$ , and  $d_3 = 0 + 1 + 0 = 1$ .

1. Motivation

Note that the formulas are not arbitrary: the indices 1, 3, 5, 7 in the first formula are those with a 1 in the one's place in their binary expansion (001, 011, 101, 111), those in the second formula have a 1 in the twos place (010, 011, 110, 111), and those in the third have a 1 in the fours place (100, 101, 110, 111).

If you receive a 7-digit number  $d_1d_2d_3d_4d_5d_6d_7$ , then this is a codeword if and only if

$$d_1 + d_3 + d_5 + d_7 = c_1 = 0,$$
  

$$d_2 + d_3 + d_6 + d_7 = c_2 = 0,$$
  

$$d_4 + d_5 + d_6 + d_7 = c_4 = 0.$$

(These are the same formulas as above in light of -1 = 1.) The resulting 3-digit word  $c_4c_2c_1 = 000$  is called the checking number of the word, and it is 000 if and only if the word is a codeword. The really amazing thing is this: if the word you receive contains a single error, then the checking number gives you the position (in binary representation) in which the error was made!

In fact, assume you receive the word 1011000. You perform the parity checks

so the checking number is  $c_4c_2c_1 = 110$ , which is the binary expansion of 4 + 2 + 0 = 6, indicating that there's an error in the sixth digit.

**Exercise 1.19.** List all codewords of Hamming's [7,4]-code.

**Exercise 1.20.** Using a case-by-case analysis, show that the checking number of Hamming's [7, 4]-code always locates single errors correctly.

**Exercise 1.21.** Write the digits of the number 3.14159 as 4-digit binary numbers, and encode each of them using Hamming's [7, 4]-code.

**Exercise 1.22.** The following message was encoded using Hamming's [7,4]-code. Locate all errors and correct them:

1001011, 0101111, 1101001, 1110010.

If I haven't done anything stupid, these are binary expansions of the decimal digits of e.

**Exercise 1.23.** Show that Hamming's [7,4]-code is linear.

**Exercise 1.24.** Compute the information rate of Hamming's [7,4]-code.

**Exercise 1.25.** Determine the automorphism group of Hamming's [7,4]-code. (This one is hard: Aut(C)  $\simeq$  GL<sub>3</sub>( $\mathbb{F}_2$ ), a group of order 168.)

#### 1.8 Where are they?

They're everywhere. Strictly speaking languages are error-correcting codes; this allows you to recognize the word 'intbrnatuonal' despite the two typos. Seen as error-correcting codes, natural languages are rather bad because there are codewords very close to each other, like 'sea', 'see', 'bee', 'bed', 'bad' etc.

One of the oldest error detecting codes was used for sending telegrams: the end of the message would contain the number of words sent so the receiver would know when a word got lost.

Whenever your computer reads something from your hard disk, there are errors being made and automatically corrected. A very common kind of code is named after Reed and Solomon; these are used for magnetic tapes, CDs, wireless phones, DVDs, and digital TV. If you scratch the surface of a CD or drill a small hole into it (if you've got a CD by the Backstreet Boys this would be a perfect opportunity to put it to good use), you still will be able to hear the music because the missing information is recomputed from the bits that have survived. If information is transmitted on the internet, some pieces will simply get lost; the fact that the internet can function at all is because of error-correcting codes. There are in fact new codes being invented to take care of the most common error in channels of this kind: losing a packet of information (big files are chopped down into small pieces, and each one is being sent around the world; one piece might find its way to your computer via Stockholm, another will come via Paris, and yet another one gets lost because of computer problems in Toronto). When the packets are put together again on your computer, it is possible to tell exactly where the missing piece should be located (of course each packet contains information about the two adjacent packages, otherwise watching videos on the net wouldn't be much fun), and a good code would enable you to recompute the missing piece instead of sending a request for the lost package back to the computer from which you downloaded the file. Errors such as the above for which the exact location of the error is known are called *erasures* in the literature.

# 2. Basic Concepts

In this chapter we introduce the basic concepts of codes and recall the necessary background from linear algebra that is needed for describing and studying linear codes.

# 2.1 Codes

The most basic algebraic structure we shall be using is the finite field with two elements,  $\mathbb{F}_2 = \{0, 1\}$ . The addition and multiplication tables are the following:

+	0	1	•	0	1
0	0	1	0	0	0
1	1	0	1	0	1

More generally, there are finite fields with p elements for any prime p, but since computers prefer 0 and 1, they're not as important in practice. We shall later construct fields with  $p^n$  elements for any prime p and every  $n \ge 1$ ; again, we shall be mainly interested in finite fields with  $2^n$  elements.

For the moment, however, our finite fields will be  $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$  for primes p. For a given integer  $n \geq 1$ , the set  $\mathbb{F}_p^n$  consists of all 'vectors'  $(x_1, \ldots, x_n)$  with  $x_i \in \mathbb{F}_p$ .

**Definition 2.1.** A code of length n is a subset of  $\mathbb{F}_p^n$ . The size of a code  $C \subseteq \mathbb{F}_p^n$  is the number #C of codewords.

If we take  $C = \mathbb{F}_p^n$ , we have a code with large size but no error-detecting capability since every word is a codeword. If we take  $C = \{0\}$ , error-correction is no problem at all but there is only one possible message since C has size 1.

**Definition 2.2.** The information rate of  $C \subseteq \mathbb{F}_p^n$  is  $\frac{1}{n} \log_p(\#C)$ .

Let us compute the information rate of the repetition code of length n = 3 over  $\mathbb{F}_2$ ; the codewords are (0,0,0) and (1,1,1), so #C = 2 and  $\log_2 \#C = 1$ , hence the information rate is  $\frac{1}{3}$ .

**Exercise 2.1.** Compute the information rates of the repetition codes of length n, of Hamming's square code, and of Hamming's [7, 4]-code.

**Definition 2.3.** A code  $C \subseteq \mathbb{F}_p^n$  with information rate k/n is called a code of type  $[n, k]_p$  (we will soon add a third number to the type).

# 2.2 Hamming Distance

We have already mentioned that English is not a very good error correcting codes because it has many words that are quite similar. We can make this precise by introducing the Hamming distance  $d_H$  between two words.

**Definition 2.4.** A word of length n over the alphabet  $\mathbb{F}_p$  is a vector  $(x_1, x_2, \cdots, x_n) \in \mathbb{F}_p^n$ .

Let  $x = (x_1, x_2, \dots, x_n)$  and  $y = (y_1, y_2, \dots, y_n)$  be two words of length n; then their Hamming distance is defined by

$$d_H(x, y) = \#\{i : x_i \neq y_i\},\$$

*i.e.*, it is the number of positions in which the two words differ.

Thus, in English we have  $d_H(\text{bad}, \text{bed}) = 1$ ,  $d_H(\text{bed}, \text{bee}) = 1$ , as well as  $d_H(\text{bad}, \text{bee}) = 2$ .

The most basic properties of the Hamming distance are

**Proposition 2.5.** For words  $x, y, z \in \mathbb{F}_2^n$ , we have

- $d_H(x,y) \ge 0$  with equality if and only if x = y;
- $d_H(x, y) = d_H(y, x);$
- $d_H(x,z) \le d_H(x,y) + d_H(y,z).$

*Proof.* Clearly  $d_H(x, y) \ge 0$  since the Hamming distance is the cardinality of a set. Moreover,  $d_H(x, y) = 0$  means that x and y differ in 0 positions, hence x = y. The symmetry of  $d_H$  is also obvious, and the triangle inequality is also easily seen to hold: if x differs from y in r positions, and if y differs from z in s positions, then x differs from z in at most r + s positions (possibly in less than r + s positions if x and y resp. y and z differ in the same position).

In fact, assume that x and y differ in the first a positions, and that y and z differ in b out of them; moreover, assume that y and z also differ in c positions at which x and y are equal:

#### 2. Basic Concepts

Then  $d_H(x, y) = a$ ,  $d_H(y, z) = b + c$ , and  $d_H(x, z) = a - b + c$  (if we replace  $F_2$  by  $\mathbb{F}_p$ , the this equality must be replaced by  $d_H(x, z) \ge a - b + c$  since two wrongs only make a right in the case where there are only 0s and 1s), hence clearly  $d_H(x, y) + d_H(y, z) = a + b + c \ge a - b + c = d_H(x, z)$ .  $\Box$ 

What the proposition above tells us is that the Hamming distance  $d_H$  defines a metric on  $\mathbb{F}_2^n$ . A function  $X \times X \longrightarrow \mathbb{R}$  is called a metric on a set X if it has the following properties:

- $d(x, y) \ge 0$  with equality if and only if x = y;
- $d(x,y) = d_H(y,x);$
- $d(x,z) \le d(x,y) + d(y,z)$ .

The third property is called the triangle inequality, since applied to the case  $X = \mathbb{R}^2$  of the Euclidean plane with the standard Euclidean metric  $d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ , where  $x = (x_1, x_2)$  and  $y = (y_1, y_2)$ , it says that any side of a triangle is less than (or equal to, if the triangle is degenerate) the sum of the other two sides.

**Definition 2.6.** The minimum distance d(C) of a code C is the smallest Hamming distance between codewords:  $d(C) = \min\{d_H(x, y) : x, y \in C\}$ . A code over  $\mathbb{F}_q$  with length n and information rate k/n is said to have type  $[n, k, d]_q$ .

The trivial code  $C = \mathbb{F}_p^n$  of length n over  $\mathbb{F}_p$  has d(C) = 1. The repetition code of length 3 over  $\mathbb{F}_2$  has minimum distance 3.

**Exercise 2.2.** Compute d(C) for each of the following codes:

- 1.  $C = \{101, 111, 011\};$
- 2.  $C = \{000, 001, 010, 011\};$
- 3.  $C = \{0000, 1001, 0110, 1111\}.$

Moreover, determine their automorphism groups  $\operatorname{Aut}(C)$ .

**Exercise 2.3.** Compute the minimum distance for the repetition code of length  $n \ge 1$ , Hamming's square code, and his [7, 4]-code.

**Proposition 2.7.** A code C with minimum distance d = d(C) can detect d-1 errors and correct  $\lfloor \frac{d-1}{2} \rfloor$  errors. Moreover, these bounds are best possible.

*Proof.* If a word w is obtained from a codeword c by making at most d-1 errors, then w cannot be a codeword since  $d_H(w, c) \leq d-1$  and the minimum distance is d. Thus the code detects these errors.

On the other hand, if d(C) = d, then there exist codewords c, c' with Hamming distance d, that is, we can get c' from c by making exactly d errors at the right positions; the code cannot detect such an error. 20 Franz Lemmermeyer

For proving the claims about error-correcting, let's first look at the case when d = 2r + 1 is odd. If at most r errors are made in the transmission of the codeword c, then we can retrieve c by choosing the unique codeword with minimal Hamming distance to the received message. If c, c' are codewords with d(c, c') = 2r + 1, then making r + 1 errors at the right positions in c will make this method decode the message as c'.

**Exercise 2.4.** Complete the proof of Prop. 2.7 (the case d = 2r).

**Definition 2.8.** The weight wt(x) of an element  $x \in \mathbb{F}_p^n$  is the number of its nonzero coordinates. The minimum weight of a code is the minimal weight of its nonzero codewords.

Observe that the Hamming distance of x and y is the weight of their difference x - y.

**Exercise 2.5.** Encode messages consisting of strings w of 4 bits by first repeating the word and then adding a parity check digit, i.e. by writing wwx, where  $x \equiv wt(w) \mod 2$ . Determine the type of this code.

**Remark.** There are a couple of things that will not play a big role in this course, one of them being the automorphism group  $\operatorname{Aut}(C)$  of a code (it is, however, important for the deeper parts of the theory of error-correcting codes). The reason I'm mentioning them is a) because it's fun to play around with this notion even if we can't connect it to deep theorems, and b) because it is an example (Galois theory is another) where modern mathematics attaches groups to the objects it wants to study. Most introductory textbooks don't even mention  $\operatorname{Aut}(C)$ .

Another notion that we won't have time to really go into is the *weight* enumerator of a code. Weight enumerators are polynomials attached to codes, and the simplest is the Hamming weight enumerator defined by

$$W_C(X,Y) = \sum_{i=0}^n a_i X^{n-i} Y^i,$$

where  $a_i$  denotes the number of codewords of weight *i*. Thus the polynomial  $W_C(X, Y)$  contains information about the weight distribution of codewords.

- 1. The [n, 1]-repetition code has two codewords, namely  $0 \dots 0$  and  $1 \dots 1$ , hence  $a_0 = 1$ ,  $a_1 = \dots = a_{n-1} = 0$ , and  $a_n = 1$ . Thus  $W_C(X, Y) = X^n + Y^n$ .
- 2. The [4, 2]-repetition code  $C = \{0000, 0011, 1100, 1111\}$  has weight enumerator  $W_C(X, Y) = X^4 + 2X^2Y^2 + Y^4$ .

**Exercise 2.6.** The weight enumerator for the [4, 2]-repetition code is related to the weight enumerator for the [2, 1]-repetition code. How? Can you generalize?

**Exercise 2.7.** Compute the weight enumerator for Hamming's [7,4]-code.

**Exercise 2.8.** Compute the weight enumerators for the codes in Exercise 2.2.

#### 2.3 Vector Spaces

For the axioms of vector spaces see Appendix A.4. As an exercise, prove that  $\lambda 0 = 0$ , where 0 is the neutral element of (V, +), and that 0v = 0, where the 0 on the left hand side is the neutral element of (F, +).

Examples of vector spaces are

- $F^n = \underbrace{F \times \ldots \times F}_{n \text{ times}}$ ; its elements are vectors  $(x_1, \ldots, x_n)$  with  $x_i \in F$ .
- the set  $F[X]_n$  of polynomials of degree  $\leq n$ ; the vectors are polynomials  $a_0 + a_1 X + \ldots + a_n X^n$ ;
- the polynomial ring F[X], whose elements are all the polynomials in one variable X with coefficients in F.

A subset  $U \subseteq V$  of an *F*-vector space *V* is called a subspace if *U* is a vector space itself. For  $m \leq n$ , it is easily seen that  $F[X]_m$  is a subspace of  $F[X]_n$ .

If there are elements  $v_1, \ldots, v_n \in V$  such that every  $v \in V$  can be written in the form

$$v = \lambda_1 v_1 + \ldots + \lambda_n v_n, \tag{2.1}$$

then V is said to be finitely generated. If the representation (2.1) is unique, the set  $\{v_1, \ldots, v_n\}$  is called a basis, and  $n = \dim_K V$  is called the dimension of V.

For vector spaces V over finite fields F, the dimension gives us information about the number of elements in V:

**Lemma 2.9.** Let  $\mathbb{F}_q$  be a finite field with q elements, and let V be an  $\mathbb{F}_q$ -vector space of dimension n. Then V has  $q^n$  vectors.

*Proof.* Every vector  $x \in V$  has a unique representation  $x = a_1x_1 + \ldots + a_nx_n$ , where  $x_1, \ldots, x_n$  is a basis of V and where  $a_i \in \mathbb{F}_q$ .

For general vector spaces, one has to work a bit to show that all bases have the same cardinality. For vector spaces over finite fields, we can give a very simple proof based on a counting argument:

**Corollary 2.10.** If V is a finite dimensional vector space over a finite field F, then all bases of V have the same cardinality.

*Proof.* Let  $x_1, \ldots, x_m$  and  $y_1, \ldots, y_n$  be bases of V. Applying Lemma 2.9 to the first basis shows that V has  $q^m$  elements, applying it to the second basis shows that V has  $q^n$  elements. Thus  $q^m = q^n$ , hence m = n.

So bases of vector spaces are useful - but do they always exist?

**Theorem 2.11.** Every finitely generated F-vector space has a basis.

*Proof.* Let V be generated by  $v_1, \ldots, v_n$ , and assume that no proper subset of the  $v_i$  generates V. If  $\{v_1, \ldots, v_n\}$  is not a basis, there must be an element  $v \in V$  with two different representations, say

$$v = \lambda_1 v_1 + \ldots + \lambda_n v_n = \mu_1 v_1 + \ldots + \mu_n v_n.$$

This implies

$$0 = \kappa_1 v_1 + \ldots + \kappa_n v_n,$$

where  $\kappa_i = \lambda_i - \mu_i$ , and where not all the  $\kappa_i$  vanish.

Assume without loss of generality that  $\kappa_n \neq 0$ . Then

$$v_n = -\kappa_n^{-1}(\kappa_1 v_1 + \ldots + \kappa_{n-1} v_{n-1}),$$

but this implies that V is generated by  $v_1, \ldots, v_{n-1}$  contrary to our assumption.

This has a couple of simple consequences that are valid in general (i.e. without assuming that the field F is finite):

**Corollary 2.12.** If  $U \subseteq V$  are finite dimensional *F*-vector spaces over a finite field *F*, then dim  $U \leq \dim V$ , with equality if and only if U = V.

*Proof.* Let dim U = m, dim V = n, and q = #F; then U has  $q^m$  elements, V has  $q^n$  elements, and the claims follow.

For proving that vectors  $x_1, \ldots, x_n$  form a basis of a vector space V, you have to prove two things: first, that every  $v \in V$  can be written as a linear combination of the  $x_i$  (we say that the  $x_i$  span V), and second, that this representation is unique. To show this, it is sufficient to prove that the zero vector 0 has a unique representation, because if any vector v can be written in two different ways as a linear combination of the  $x_i$ , then so can 0 = x - x. Since 0 has an obvious representation (namely  $0 = 0x_1 + \ldots + 0x_n$ ), it is sufficient to show that the  $x_i$  are linearly independent: this means that whenever  $0 = a_1x_1 + \ldots + a_nx_n$ , all the  $a_i$  must vanish.

#### 2.4 Subspaces

Given subspaces U and W of V, we can form the following subspaces:

- $U \cap W = \{v \in V : v \in U, v \in W\}$  (intersection);
- $U + W = \{u + w \in V : u \in U, w \in W\}$  (sum of U and W).

Given any pair of vector spaces U and W, we can form their direct sum

$$U \oplus W = \{(u, w) : u \in U, w \in W\}.$$

Note that  $U \oplus W$  is not a subspace of V even if U and W are: it's something completely new.

Exercise 2.9. Show that these sets really are vector spaces. Prove that

 $\begin{array}{rcl} \displaystyle \dim U \cap W & \leq & \min\{\dim U, \dim W\}, \\ \displaystyle \max\{\dim U, \dim W\} & \leq & \dim U + V & \leq & \dim U + \dim V, \\ \displaystyle and & & \dim U \oplus W & = & \dim U + \dim W. \end{array}$ 

Prove that U+U = U, but  $U \oplus U \neq U$ . More generally, show that U+W = U if and only if  $W \subseteq U$ . Find an example that shows that  $U \cup W$  is not always a subspace of V.

A map  $f: U \longrightarrow V$  between two *F*-vector spaces is called a linear map if it respects the vector space structure, that is, if  $f(\lambda_1 u_1 + \lambda_2 u_2) = \lambda_1 f(u_1) + \lambda_2 f(u_2)$  for all  $\lambda_1, \lambda_2 \in F$  and all  $u_1, u_2 \in U$ .

Every linear map comes attached with two vector spaces, the kernel and the image.

**Exercise 2.10.** Show that, for a linear map  $f: U \longrightarrow V$ , the sets

$$\ker f = \{u \in U : f(u) = 0\} \quad and$$
$$\inf f = \{f(u) : u \in U\}$$

are vector spaces. In fact, ker f is a subspace of U and im f a subspace of V.

If  $f: U \longrightarrow V$  is an injective linear map between vector spaces over finite fields, and if U and V have the same (finite) dimension, then this lemma implies that f is an isomorphism. Actually, this is true even for infinite fields:

**Proposition 2.13.** If  $f: U \longrightarrow V$  is an injective linear map between vector spaces of the same finite dimension, then f is an isomorphism.

*Proof.* We only prove this for vector spaces over finite fields. Then the fact that f is injective implies that #f(U) = #U; thus  $\#U = \#f(U) \leq \#V = \#U$  since  $f(U) \subseteq V$  and #V = #U by assumption, we must have equality throughout.

The following lemma is often helpful for computing dimensions:

**Lemma 2.14.** If  $f : U \longrightarrow V$  is a linear map between finite dimensional vector spaces, then dim  $U = \dim \ker f + \dim \inf f$ .

*Proof.* Let  $u_1, \ldots, u_r$  be a basis of ker f, and let  $v_1, \ldots, v_s$  be a basis of im f. Since the  $v_i$  are images of vectors from U, we can write  $v_1 = f(u_{r+1})$ ,  $\ldots, v_s = f(u_{r+s})$  with  $u_{r+i} \in U$ .

We claim that  $\{u_1, \ldots, u_r, u_{r+1}, \ldots, u_{r+s}\}$  is a basis of U. There are two things we have to prove:

- 1. The  $u_i$  are independent. To see this, assume that we have  $a_1u_1 + \ldots + a_{r+s}u_{r+s} = 0$  with  $a_i \in F$ . Applying f and observing that  $f(u_1) = \ldots = f(u_r) = 0$  we get  $0 = f(0) = f(a_{r+1}u_{r+1}) + \ldots + f(a_{r+s}u_{r+s}) = a_{r+1}v_1 + \ldots + a_{r+s}v_s$ . Since the  $v_i$  are linearly independent, we deduce that  $a_{r+1} = \ldots = a_{r+s} = 0$ . This implies  $a_1u_1 + \ldots + a_ru_r = 0$ , and since the  $u_i$   $(1 \le i \le r)$  form a basis, they are independent. Thus  $a_i = 0$  for  $1 \le i \le r + s$ , proving the claim that the  $u_i$  are independent.
- 2. The  $u_i$  generate the vector space U. Given any  $u \in U$ , we have to write it as a linear combination of the  $u_i$ . We can write  $f(u) = a_{r+1}v_1 + \ldots + a_{r+s}v_s$ ; but then  $f(u - a_{r+1}u_{r+1} - \ldots - a_{r+s}u_{r+s}) = 0$ , so this element is in the kernel and therefore can be written as a linear combination of  $u_i$   $(1 \le i \le r)$ ; this implies the claim.

This completes the proof.

**Proposition 2.15.** If U, W are subspaces of an F-vector space V, then so are  $U \cap W$  and  $U + W = \{u + w : u \in U, w \in W\}$ . If V has finite dimension, then  $\dim(U + W) = \dim U + \dim W - \dim(U \cap W)$ .

*Proof.* We define a linear map  $f: U \oplus W \longrightarrow U + W$  by sending the vector  $(u, w) \in U \oplus W$  to  $u + w \in V$ . This map is surjective since every vector in U + W has the form u + w for  $u \in U$  and  $w \in W$ , hence comes from some (u, w). The kernel of f consists of all (u, w) such that u + w = 0. We claim that ker  $f = \{(u, -u) : u \in U \cap W\}$ .

Clearly such vectors are in the kernel since f(u, -u) = u - u = 0. Assume therefore that f(u, w) = 0; then u + w = 0 shows that  $w \in U$  and w = -u, proving the claim.

Clearly dim ker  $f = \dim U \cap W$ , and invoking Lemma 2.14 completes the proof.

Assume that we have a vector space V of dimension n and the standard scalar product  $x \cdot y = x_1y_1 + \ldots x_ny_n$ , where  $x = (x_1, \ldots, x_n)$  and  $y = (y_1, \ldots, y_n)$ , we can assign a vector space  $U^{\perp}$  to any subspace U of V in the following way: we put

$$U^{\perp} = \{ x \in V : x \cdot u = 0 \text{ for all } u \in U \}$$

#### 2. Linear Algebra

and call it the *orthogonal space* to U.

In linear algebra, where you consider vector spaces over the field  $\mathbb{R}$  of real numbers, for any subspace U you always have  $V = U \oplus U^{\perp}$ . This does not hold anymore over finite fields. Consider e.g.  $V = \mathbb{F}_2^3$  and  $U = \{(0,0,0), (1,1,0)\}$ . Then  $U^{\perp} = \{(0,0,0), (0,0,1), (1,1,0), (1,1,1)\}$  as you can easily verify, and in particular  $U \subset U^{\perp}$ !

**Exercise 2.11.** Assume that  $U \subseteq \mathbb{F}_2^n$  is an even subspace (every vector has even weight). Show that  $U \subseteq U^{\perp}$ .

**Proposition 2.16.** Let U and V be as above, and assume that V has finite dimension. Then dim  $V = \dim U + \dim U^{\perp}$ .

*Proof.* Let  $n = \dim V$  and  $k = \dim U$ , and take a basis  $u_1 = (u_{11}, \ldots, u_{1n})$ ,  $\ldots, u_k = (u_{k1}, \ldots, u_{kn})$ . A vector  $v = (v_1, \ldots, v_n)$  will lie in  $U^{\perp}$  if and only if  $u \cdot v = 0$  for all  $u \in U$ .

We claim that the last condition is equivalent to  $u_i \cdot v = 0$  for  $1 \leq i \leq k$ ; in fact, if  $u \cdot v = 0$  for all  $u \in U$  then certainly  $u_i \cdot v = 0$  since  $u_i \in U$ . Conversely, if  $u_i \cdot v = 0$  for a basis  $\{u_1, \ldots, u_k\}$ , and if  $u \in U$ , then  $u = a_1u_1 + \ldots a_nu_n$ , hence  $u \cdot v = a_1u_1 \cdot v + \ldots + a_nu_n \cdot v = 0$ .

Thus  $v \in U^{\perp}$  if and only if we have  $u_i \cdot v = 0$  for  $1 \leq i \leq k$ , that is, if and only if

$$u_{11}v_1 + \ldots + u_{1n}v_n = 0,$$
  
$$\vdots$$
  
$$u_{k1}v_1 + \ldots + u_{kn}v_n = 0.$$

This is a system of k linear equations in n unknowns  $v_n$ , and since the matric  $(a_{ij})$  has rank k, the space of solutions has dimension n - k.

In fact, after some elementary row and column transformations we may assume that the basis vectors  $u_1, \ldots, u_k$ , when concatenated into a  $k \times n$ matrix, look like this:

$\binom{1}{}$	• • •	0	$u_{1,k+1}$	• • •	$u_{1,n}$	
1:	·	÷	÷		÷	
$\setminus 0$		1	$u_{k,k+1}$		$u_{k,n}$	

Thus  $v = (v_1, \ldots, v_n)$  is orthogonal to all  $u_i$  if and only if the components  $v_i$  satisfy

$$v_i + u_{i,k+1}v_{k+1} + \ldots + u_{i,n}v_n = 0.$$
(2.2)

Clearly we can choose the  $v_{k+1}, \ldots, v_n$  freely (there are  $q^{n-k}$  choices), and the  $v_1, \ldots, v_k$  are then determined by the equations (2.2). Thus  $\#U^{\perp} = q^{n-k}$ , hence dim  $U^{\perp} = n - k$ .

**Proposition 2.17.** If U is a subspace of the finite dimensional vector space V, then  $U^{\perp \perp} = U$ .

*Proof.* We have  $U^{\perp} = \{v \in V : v \cdot u = 0 \text{ for all } u \in U\}$  and  $U^{\perp \perp} = \{v \in V : v \cdot w = 0 \text{ for all } w \in U^{\perp}\}$ . Since  $u \cdot w = 0$  for all  $u \in U$  and  $w \in U^{\perp}$ , we have  $U \subseteq U^{\perp \perp}$ . By Proposition 2.16, we know  $\dim U^{\perp} = \dim V - \dim U$  and  $\dim U^{\perp \perp} = \dim V - \dim U^{\perp} = \dim U$ . Thus  $U^{\perp \perp} \subseteq U$  are vector spaces of the same dimension, hence they must be equal.  $\Box$ 

# 2.5 Linear Codes

**Definition 2.18.** A linear code of length n over  $\mathbb{F}_p$  is a subspace of  $\mathbb{F}_p^n$ . The dimension k of C is the dimension of C as an  $F_p$ -vector space.

This makes only sense if we agree to identify codewords like 10010 with vectors (1, 0, 0, 1, 0). From now on we will use both notations simultaneously.

**Exercise 2.12.** Check that the information rate of a code of length n is k/n, where k is the dimension of C.

**Exercise 2.13.** The parity check code adds a check digit to any string of n bits by demanding that the sum of all coordinates (including the check digits) is 0. Show that the parity check code is a linear code in  $\mathbb{F}_2^{n+1}$  and determine its parameters.

**Proposition 2.19.** The minimum distance of a linear code is its minimum weight.

*Proof.* Let d be the minimum distance of a code C; then there are codewords  $x, y \in C$  with  $d_H(x, y) = d$ . Since C is linear, we have  $x - y \in C$ , and  $d = d_H(x, y) = wt(x - y)$  shows that there is a codeword with weight d. If there was a codeword  $z \in C$  with weight < d, then  $d_H(0, z) < d$ , contradicting the assumption that d(C) = d (observe that  $0 \in C$  since C is linear).  $\Box$ 

**Exercise 2.14.** Check whether the following two codes are linear, compute their minimum distance and information rate, and determine their type:

$$C_1 = \{(0,0,0,0), (1,0,0,0), (0,1,0,0), (1,1,0,0)\}$$
  

$$C_2 = \{(0,0,0,0), (1,0,1,0), (0,1,0,1), (1,0,0,1)\}$$

Moreover, compute their automorphism groups  $\operatorname{Aut}(C_i)$ .

**Exercise 2.15.** Show that there is no  $[4,2]_2$  code with minimum distance 3.

**Exercise 2.16.** Find all codes of type  $[4, 2]_2$  with minimum distance 2.

**Definition 2.20.** A linear code  $C \subseteq \mathbb{F}_q^n$  is called degenerate if C is contained in a hyperplane, i.e., if  $C \subseteq H_i$  for some  $H_i = \{(x_1, \ldots, x_n) \in \mathbb{F}_p^n : x_i = 0\}$ .

Degenerate codes are uninteresting in practice.

**Exercise 2.17.** Let  $C \subseteq \mathbb{F}_p^n$  be a linear code. Then  $C' \subseteq \mathbb{F}_p^{n+1}$  defined by

$$C' = \{(x_1, \dots, x_n, 0) : (x_1, \dots, x_n) \in C\}$$

is a degenerate linear code.

**Definition 2.21.** A code C is called even if all of its codewords have even weight:  $wt(c) \equiv 0 \mod 2$  for all  $c \in C$ .

**Exercise 2.18.** Let  $C \subseteq \mathbb{F}_2^n$  be a code; the parity extension of C is

$$C^{+} = \{ (x_1, \dots, x_{n+1}) \in \mathbb{F}_2^{n+1} : (x_1, \dots, x_n) \in C, x_{n+1} = x_1 + \dots + x_n \}.$$

Show that  $C^+$  is a linear code if C is, and that  $C^+$  is degenerate if and only if C is even.

To every *linear* code  $C \subseteq \mathbb{F}_q^n$  we can associate a 'partner': since C is a subspace of  $\mathbb{F}_q^n$ , so is  $C^{\perp}$ , and  $C^{\perp}$  is called the *dual code* of C. (Actually it should have been called the orthogonal code, but ...)

**Exercise 2.19.** Let C be a code of type  $[n,k]_q$ ; show that  $C^{\perp}$  is a code of type  $[n,n-k]_q$ .

Let us compute  $C^{\perp}$  for the repetition code of type [4, 2]. We have  $C = \{0000, 0011, 1100, 1111\}$ . The dual code has type [4, 2], hence also consists of exactly 4 codewords. Now  $(x_1, x_2, x_3, x_4) \in C^{\perp}$  is such a codeword if and only if  $x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4 = 0$  for all  $(y_1, y_2, y_3, y_4) \in C$ . This gives us a system of 4 linear equations for 4 unknowns that we can solve, but as a matter of fact it is plain to see that  $C \subseteq C^{\perp}$ : all the codewords of C are orthogonal to themselves. Since both C and  $C^{\perp}$  have dimension 4, we must have  $C = C^{\perp}$ .

**Definition 2.22.** Linear codes C with  $C = C^{\perp}$  are called self dual.

The [3, 1]-repetition code C is not self dual (of course not, since  $C^{\perp}$  must have dimension 3-1=2): we have  $C = \{000, 111\}$ , and a little computation shows that  $C^{\perp} = \{000, 011, 101, 110\}$ . Note that d(C) = 3 and  $d(C^{\perp}) = 2$ .

#### 2.6 Generator and Parity Matrix

In general, a code is determined by its codewords; if the number of codewords is large, tables aren't very instructive. For linear codes, there is a much better and very concise way of describing the code:

**Definition 2.23.** Let C be a linear code of type  $[n, k]_q$ . A  $k \times n$  matrix whose rows form a basis of C in  $\mathbb{F}_q^n$  is called a generator matrix.

Observe that a  $k \times n$ -matrix has k rows and n columns (Rows fiRst, Columns seCond). Some books use the transposed matrix  $G^t$  as a generator matrix; in that case it's the columns that form a basis of the codes.

**Examples.** The linear code from Section 1.5 has generator matrix

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

Hamming's [7, 4]-code has generator matrix

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

Of course we have a lot of freedom in picking generator matrices since vector spaces have many different bases. How did I find a generator matrix for Hamming's [7, 4]-code whose  $4 \times 4$ -submatrix on the left is the unit matrix? Well, I did a few calculations (we shall later see that, with a few twists, we can always make a generator matrix look like that; it has the huge advantage that we can see immediately that the rows form a basis, but it has other advantages as well): I wanted the first row to start with 1000; can I choose the remaining three digits  $x_5$ ,  $x_6$ ,  $x_7$  in such a way that this becomes a code word? I can do this if and only if the  $x_i$  satisfy the linear system of equations

$$x_3 + x_5 + x_7 = 1,$$
  

$$x_3 + x_6 + x_7 = 0,$$
  

$$x_5 + x_6 + x_7 = 0.$$

Since  $x_3 = 0$  in our case, subtracting the last from the second equation gives  $x_5 = 0$ , then the first shows  $x_7 = 1$ , and now the second gives  $x_6 = 1$ . Since this is a solution of the system, 1000011 is a code word, and can be used as the first row.

**Exercise 2.20.** Do similar computations for the second, third and fourth row of the generator matrix of Hamming's [7, 4]-code.

**Exercise 2.21.** Determine a generator matrix of the [4,2]-repetition code. Do the same for the general [rk,k]-repetition code.

**Exercise 2.22.** Determine a generator matrix for Hamming's  $n \times n$ -square code. If you can't see how to do it in general, solve the problem for n = 2, 3, 4.

Now  $k \times n$ -matrices give us linear maps  $F^k \longrightarrow F^n$  via  $x \longmapsto xG$ . In fact, this is how you encode messages: given a code  $C \subseteq \mathbb{F}_q^n$ , choose a basis of the subspace C, form the corresponding generator matrix G, and encode each message block  $m = (m_1, \ldots, m_k)$  as  $c = (c_1, \ldots, c_n) = cG$ .

2. Basic Concepts

**Exercise 2.23.** In the linear code from Section 1.5 and in Hamming's [7, 4]-code, check that a message is encoded by matrix multiplication  $x \mapsto xG$ , where G is the generator matrix given above.

Since a code by definition is a vector space, and since vector spaces have many different bases, there are also many possible generator matrices for a given code, and each of these matrices will describe a different way in which the actual encoding is done. From now on we will think of a linear code as being given by a generator matrix.

**Definition 2.24.** Two codes  $C, C' \subseteq \mathbb{F}_2^n$  are called equivalent if there is a permutation  $\pi$  of the set  $\{1, 2, \ldots, n\}$  such that  $C' = \{(x_{\pi(1)}, \ldots, x_{\pi(n)}) : (x_1, \ldots, x_n) \in C\}$ .

Thus codes are equivalent if they are equal up to a permutation of coordinates. Take the ZIP bar code for example: if we swap the last two bits, we get an equivalent code (that happens to be identical in this case). The [4, 2]-repetition code has code words 0000, 0011, 1100, 1111; switching the second and the third digit we get the equivalent code 0000, 0101, 1010, 1111. This last code has a generator matrix  $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$ ; note that the original [4, 2]repetition code does not have a generator matrix whose left square part is the identity matrix.

**Exercise 2.24.** Show that equivalent codes have the same type, weight enumerator, and automorphism group.

#### The Parity Matrix

**Definition 2.25.** A parity matrix H for a code C is a generator matrix of the dual code  $C^{\perp}$ .

Note that  $C^{\perp}$  has type [n, n-k], so H is a  $(n-k) \times n$ -matrix.

**Exercise 2.25.** Compute a parity matrix for the linear code in Section 1.5 and for Hamming's [7, 4]-code.

Let x be any message; if G is a generator matrix for the code C, then x is encoded as xG. This code word is perpendicular to any code word  $yH \in C^{\perp}$ , i.e., we must have  $xG \cdot yH = 0$ . Since the dot product  $u \cdot v$  is just matrix multiplication  $uv^T$ , this means that  $0 = xG(yH)^T = xGH^Ty^T$ . But this holds for all vectors  $x \in F^k$  and  $y \in F^{n-k}$ , hence  $GH^T = 0$  must be the zero matrix. Note that G has type  $k \times n$ ,  $H^T$  has type  $n \times (n-k)$ , hence  $GH^T$ has type  $k \times (n-k)$ .

Using the parity matrix it is easy to test whether a received message is a codeword or not:

**Proposition 2.26.** Let  $C \subseteq \mathbb{F}_q^n$  be a linear code of type  $[n, k]_q$  with generator matrix G and parity matrix H. Then  $c \in C$  if and only if  $cH^T = 0$ .

*Proof.* Assume that c is a codeword. Then c = xG for some vector  $x \in \mathbb{F}_q^k$ , hence  $cH^T = xGH^T = 0$  by the discussion above. Conversely, if  $cH^T = 0$ , then c is perpendicular to every vector in  $C^{\perp}$ , hence  $c \in (C^{\perp})^{\perp} = C$ .

At this point we have reduced encoding and error detection to some simple (and fast) matrix computations. For error correction, on the other hand, we have to work harder. But before we do that, let's first look at a generalization of Hamming's [7, 4]-code.

# 2.7 Hamming Codes

Let  $r \geq 1$  be a natural number and put  $n = 2^r - 1$ . Let H be the  $r \times n$ matrix whose columns are just the nonzero vectors in  $\mathbb{F}_2^r$ . The code C = $\operatorname{Ham}(r)$  whose parity check matrix is H is called the Hamming code of length n. For this definition to make sense we have to check that the rows of Hare linearly independent. But this is clear: among the columns of H (we write these columns horizontally for typographical reasons) are  $(1, 0, \ldots, 0)$ ,  $(0, 1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 1)$ . Assume we have a relation  $a_1H_1 + \ldots + a_rH_r = 0$  between the rows  $H_r$ ; then this relation must hold 'columnwise', that is,  $a_1H_{1i} + \ldots + a_rH_{ri} = 0$  for every  $1 \leq i \leq n$ . If the vector  $(1, 0, \ldots, 0)$ is in the *i*-th column, then this relation gives  $a_1 = 0$ . Similarly, the other columns listed above show that  $a_2 = \ldots = a_r = 0$ , so the rows are linearly independent.

**Examples.** If r = 2, then n = 3 and  $H = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ . We find G = (1, 1, 1) and dim Ham(2) = 1. If r = 3, then n = 7 and dim Ham(3) = 4; actually

$$H = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix},$$

and C is equivalent to Hamming's  $[7, 4]_2$  code.

The order in which we list the rows in the parity matrix is not really important: permuting these rows does not destroy the property  $GH^T = 0$ . Sometimes, however, it is an advantage to have a specific order; in our case, we can choose the 'three relations' of Hamming's [7, 4]-code Ham(3) as the rows of H:

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$
 (2.3)

**Theorem 2.27.** The Hamming code C = Ham(r) of length  $n = 2^r - 1$  is a binary code of type  $[n, n - r, 3]_2$ .

*Proof.* By definition, the parity matrix of C is a  $n \times r$ -matrix, so  $C^{\perp}$  has type [n, r]. Since dim  $C + \dim C^{\perp} = n$ , we conclude that C has type [n, n - r].

It remains to compute the minimum distance d(C). We start by showing that there is no codeword of weight 1. Assume there is, say  $x = (x_1, \ldots, x_n)$ with  $x_i = 1$ ,  $x_j = 0$  for  $j \neq i$ . This is a codeword if and only if  $xH^T = 0$ , where H is the parity matrix. For our x, we see that  $xH^T$  is the *i*-th row (!) of  $H^T$ , that is, the *i*-th column of H. But by definition of H, the columns are nonzero.

Next let us show that there are no codewords of weight 2. Assume that x is such a codeword, and that e.g.  $x_i = x_j = 1$ . Then  $xH^T$  is the sum of the *i*-th and *j*-th columns of H, so if  $xH^T = 0$ , these columns must add up to 0. But over  $\mathbb{F}_2$ , this implies that the *i*-th and *j*-th columns coincide, which contradicts the construction of H.

By now we know that  $d(C) \geq 3$ . We now show that there is a codeword of weight 3, completing the proof. To this end, observe that the three vectors  $(0, \ldots, 0, 0, 1)$ ,  $(0, \ldots, 0, 1, 0)$  and  $(0, \ldots, 0, 1, 1)$  are columns of H, say the first, second and third (after rearranging columns if necessary). But then  $x = (1, 1, 1, 0, \ldots, 0)$  is orthogonal to every row of H, hence  $xH^T = 0$ , and so x is a codeword of weight 3.

It is immediate from this result that the information rate of Hamming codes converges rapidly to 1 as  $r \longrightarrow \infty$ : in fact, the information rate is  $\frac{n-r}{r} = 1 - \frac{r}{n}$ , and  $r/n = r/(2^r - 1)$  converges to 0 quite fast. This does not imply that Hamming codes are 'better' for large r, for the following reason: transmitting information over a noisy channel will result in errors being made. If the probability that an error occurs in each bit is constant, say  $p \ll 1$ , then the probability that more than 1 error is made in a block of length n grows quite fast with n, and eventually most of the transmitted messages will have more than 1 error, making the Hamming code (which is 1-error correcting) for large n inefficient despite its high information rate.

# 2.8 Perfect Codes

We have mentioned a couple of times now that good codes are codes such that certain invariants (e.g. the information rate, or the ability to correct t errors) are both large. Here we will make a couple of simple observations that show that there is a limit to what we can do.

Let us start with defining perfect codes (this will take a while). Given a vector  $x \in \mathbb{F}_q^n$ , we can define the (Hamming) sphere of radius r and center x as the set of all  $y \in \mathbb{F}_q^n$  such that  $d(x, y) \leq r$ . Clearly, if C is a code with minimum distance d = 2t + 1, then spheres of radius t around codewords are disjoint: if not, then some  $z \in \mathbb{F}_q^n$  would be contained in two spheres around  $x \neq y$ , and then  $d(x, y) \leq d(x, z) + d(z, y) \leq 2t < d(C)$ , contradicting the fact that d(C) is the minimum distance.

#### 32 Franz Lemmermeyer

How many points in a sphere?

**Lemma 2.28.** A sphere of radius r in  $\mathbb{F}_q^n$  (where  $0 \le r \le n$ ) contains exactly

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \ldots + \binom{n}{r}(q-1)^r$$

points.

*Proof.* Fix  $x \in \mathbb{F}_q^n$  and look at the vectors  $y \in \mathbb{F}_q^n$  within Hamming distance r. Let's count the number of points y that differ in exactly  $m \leq r$  positions from x. For each of these m positions, we have q-1 choices of picking a coordinate different from the corresponding coordinate of x, and there are  $\binom{n}{m}$  ways of choosing these positions (the order of this selection clearly does not matter). Thus there are  $\binom{n}{m}(q-1)^m$  vectors y that differ at exactly m positions from x, and this implies the claim.

This rather simple observation leads to a bound for the quality of codes, the sphere packing bound or Hamming bound:

**Proposition 2.29.** For any code  $C \subseteq \mathbb{F}_q^n$  with minimum distance d(C) = 2t + 1, the inequality

$$#C \cdot \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \ldots + \binom{n}{t}(q-1)^t \right\} \leq q^n \quad (2.4)$$

holds.

*Proof.* We have seen above that spheres of radius t around codewords are disjoint. By putting such spheres around each of the #C codewords, the number of points per sphere multiplied by the number of spheres cannot exceed the number of points in  $\mathbb{F}_q^n$ : this is exactly what the inequality says.

In particular, if C is a binary code of type [n, k, 2t + 1], then

$$2^k \cdot \left\{ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{t} \right\} \leq 2^n.$$

**Definition 2.30.** A code  $C \subseteq \mathbb{F}_q^n$  is called perfect if there is an integer t such that for every  $x \in \mathbb{F}_q^n$  there is a unique code word  $c \in C$  with  $d_H(x, c) \leq t$ .

We now give a couple of

#### Examples

1. Consider the [2,1]-repetition code  $C = \{00, 11\}$ . The following table gives the 4 words  $v \in \mathbb{F}_2^2$  and the codewords c within distance 1 from v, and the diagram illustrates the 1-balls around the two codewords:

2. Working with Linear Codes



Clearly this code is not perfect.

2. Now consider the [3, 1]-repetition code  $C = \{000, 111\}$ . The corresponding table of codewords inside Hamming 1-balls looks like this:

v	$c: d(v, c) \le 1$	v	$c:d(v,c)\leq 1$
000	000	100	000
001	000	101	111
010	000	110	111
011	111	111	111

Thus this code is 1-perfect.

Assume that  $C \subseteq \mathbb{F}_q^n$  is perfect for some  $t \ge 0$  (we will also say that C is *t*-perfect). Then the Hamming balls with radius *t* around a codeword *c* (all words with Hamming distance  $\le t$  from *c*) contain a unique codeword, namely *c*. Therefore these Hamming balls cover  $\mathbb{F}_q^n$ , and therefore we have equality in (2.4)! And conversely, equality in (2.4) implies that the code is *t*-perfect.

**Proposition 2.31.** A t-perfect code C has minimum distance d(C) = 2t+1.

*Proof.* Let x and y be different codewords; then the t-balls around x and y are disjoint, hence  $d(x, y) \ge 2t + 1$ .

Now let c be any codeword, and let z be any vector with d(c, z) = t + 1. Then z is not inside the t-ball around c, and since C is perfect, it must be inside the t-ball around some other codeword c'. But then  $d(z, c') \leq t$ , hence  $d(c, c') \leq d(c, z) + d(z, c') \leq 2t + 1$ . Thus d(C) = 2t + 1.

The word perfect should not lure you into thinking that these codes are 'best possible'; in fact, the following exercise shows that the trivial code and some repetition codes are perfect, and neither of these are good codes, let alone close to best possible:

**Exercise 2.26.** The trivial code  $C = \mathbb{F}_q^n$  is 0-perfect. The repetition code of odd length n = 2t + 1 over  $\mathbb{F}_q$  is t-perfect.

Perfect repetition codes are called the trivial perfect codes. There are also nontrivial perfect codes:

**Proposition 2.32.** Hamming codes are perfect.

*Proof.* We have  $n = 2^r - 1$ , k = n - r, and d(C) = 3, i.e., t = 1. The sphere packing bound is given by

$$2^{n-r}(1+n) \le 2^n$$

in this case, and since  $1 + n = 2^r$ , we have equality.

The following (nontrivial) result classifies perfect codes:

**Theorem 2.33.** Let C be a t-perfect binary linear code. If t = 1, then C is (equivalent to) a Hamming code; if t > 1, then C is either equivalent to the [2t + 1, 1]-repetition code, or to the binary Golay code  $G_{23}$ .

The Golay code  $G_{23}$  has type [23, 12, 7] and is 3-perfect; a generator matrix for  $G_{23}$  is given by  $(I_{12}|P)$  with

	(	1	0	1	0	0	0	1	1	1	0	1
	[	1	1	0	1	0	0	0	1	1	1	0
		0	1	1	0	1	0	0	0	1	1	1
		1	0	1	1	0	1	0	0	0	1	1
		1	1	0	1	1	0	1	0	0	0	1
D		1	1	1	0	1	1	0	1	0	0	0
F =		0	1	1	1	0	1	1	0	1	0	0
		0	0	1	1	1	0	1	1	0	1	0
		0	0	0	1	1	1	0	1	1	0	1
		1	0	0	0	1	1	1	0	1	1	0
		0	1	0	0	0	1	1	1	0	1	1
	ſ	1	1	1	1	1	1	1	1	1	1	1 /

The Golay code owes its existence to the curiousity

$$\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 1 + 23 + 253 + 1771 = 2048 = 2^{11}.$$

The definition of the Golay code by writing down a generator matrix is of course not very satisfying. How are we supposed to check that  $d(G_{23}) = 7$ ? We will later see that both Hamming codes and Golay's code are cyclic codes and can be described elegantly using ideals in certain polynomial rings.

#### The Hat Puzzle

Consider the following problem: seven players enter a room, and a hat is placed on each player's head. The color of each hat is determined by tossing a coin, and each person can see the other players' hats but not his own. No communication is allowed between the players except for fixing a strategy before entering the room. Once they've had a chance to look at the other hats, all player must simulaneously guess their own hat's color or pass. The team

wins if at least one player guesses correctly and no one guesses incorrectly. What is the best strategy?

If we don't require that the guesses be made simultaneously, there is a quite simple strategy: the team agrees on a player A who guesses his hat is red or blue according as he sees an even or an odd number of red hats. The other players then determine their hat colors as follows: if player A sees an even number of red hats, and if another player sees an odd number of red hats not counting the hat of A, then his own hat must be red.

If we require simultaneity, then it seems at first that there is no way to win in more than 50 % of the cases since each player has a 50 % chance of guessing correctly or not. A strategy to achieve this is to pick one player who makes a guess, while all the others pass.

As a matter of fact, the players can do much better if n > 2. In fact, assume for now that there are only three players. Since the colors are distributed randomly, the chance that there are two hats of one and another head of the other color is 75 %, and a strategy to win in 3 out of 4 cases is the following: each player looks at the other players' hats; if he sees hats of different colors, he passes; if he sees two hats of the same color, he guesses his hat has the opposite color.

This really works: there are 8 different situations, and this is what happens:

hats	1	2	3	team
BBB	R	R	R	loses
BBR	_	_	R	wins
BRB	_	R	_	wins
RBB	R	_	_	wins
BRR	В	—	_	wins
RBR	_	В	_	wins
RRB	_	—	В	wins
RRR	В	В	В	loses

If you look at each player individually, then you will notice that e.g. player 1 still is wrong 50 % of the time; the clever trick of the strategy is to make many players guess wrong at the same time and ensure that they are making correct predictions when the other players pass.

What has this got to do with coding theory? If you replace the B's with 0s and the R's with 1s, then in the left column there are the 8 vectors of  $\mathbb{F}_2^3$ . Moreover, the strategy adopted by the players guarantees that they are wrong only when the vector is 000 or 111, that is, a codeword of the [3, 1]-repetition code. What makes the strategy work is the fact that this code is perfect!

In fact, consider the game with  $n = 2^r - 1$  players (we know that  $\operatorname{Ham}(r)$  is a perfect code  $\subseteq \mathbb{F}_2^n$  for all  $r \ge 1$ ). The strategy is as follows: each player has a number from 1 through n; let  $c = (c_1 \cdots c_n)$  denote the word that

describes the colors of the players' hats (of course, nobody except the host knows c). Player i knows the word  $c + e_i$ , where  $e_i$  is the word having  $c_i$  in position i and zeros everywhere else (in other words: it agrees with c except that we have put  $c_i = 0$ ).

Let *H* denote the  $r \times n$ -matrix whose *i*-th column is the binary expansion of *i*; for n = 7 we have

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

Now player *i* computes the syndrome of the word  $c+e_i$ . If  $syn(c+e_i) = 0$ , he guesses that the color of his hat is 1, if  $syn(c+e_i) = (i)_2$  (the binary expansion of *i*), he guesses that the color of his hat is 0, and he passes otherwise.

Here's what's going to happen: If c is a codeword, then  $\operatorname{syn}(c) = cH^T = 0$ , hence  $\operatorname{syn}(c + e_i) = \operatorname{syn}(e_i)$ , which is  $c_i$  times the *i*-th column of the parity check matrix. Thus  $\operatorname{syn}(c+e_i) = c_i \cdot (i)_2$ , namely  $c_i$  times the binary expansion of *i*. Thus if  $c_i = 0$ , then  $\operatorname{syn}(c + e_i) = 0$ , and player *i* guesses that his hat has color 1; if  $c_i = 1$ , then player *i* guesses his hat has color 0. Thus in this case, every player guesses the wrong color.

Now assume that c is not a codeword. Then player i passes except when  $\operatorname{syn}(c) = c_i \cdot (i)_2$ ; in this case, since  $\operatorname{syn}(c) \neq 0$ , we must have  $c_i = 1$ , and this is exactly what he guesses if he follows the strategy. Can it happen that all players are passing? No: since the Hamming code is perfect, every non-codeword c has Hamming distance 1 to a unique codeword, say to  $c + e_j$ , where  $e_j$  is the vector with 1 in position j and zeros elsewhere. According to the strategy, player j will guess, and he will guess the correct answer.

**Remark.** This problem made it into the New York Times on April 10, 2001; it was first discussed in the PhD thesis of Todd Ebert from UC Irvine in 1998 and spread like wildfire when he later offered it as an extra credit problem for his students.
# 3. Working with Linear Codes

In principle, decoding codes is easy: when you receive a word c', look for the codeword c with smallest Hamming distance to c' and declare that cwas the message that was sent. In practice, however, this does not work: reading music from your CD would take forever if decoding was done this way: what is needed is a *fast* way to decode linear codes (encoding is just a matrix multiplication, which is very fast). The way to proceed is to exploit the vector space structure of codes to replace the search for codewords by something more efficient. For doing this we introduce quotient spaces.

### 3.1 Quotient Spaces

Given an abelian group G and a subgroup H, you have learned in algebra how to form the quotient group G/H: as a set, we have

$$G/H = \{g + H : g \in G\},\$$

where

$$q + H = \{g + h : h \in H\}$$

is called a coset. We have g+H = g'+H if and only if  $g-g' \in H$ . We can make the set G/H of cosets into an abelian group by putting (g+H) + (g'+H) =(g+g')+H: the group axioms are easily verified. The idea is that computing in G/H is done as if in G, except that everything in H is regarded as being 0. The main example of quotient groups are the residue class groups  $\mathbb{Z}/m\mathbb{Z}$ , where  $G = \mathbb{Z}$  is the additive group of integers, and where  $H = m\mathbb{Z}$  is the subgroup of multiples of m.

Since  $\mathbb{Z}$  is actually a ring (and  $m\mathbb{Z}$  an ideal), we can even define a multiplication on the quotient  $\mathbb{Z}/m\mathbb{Z}$  and get the residue class rings  $\mathbb{Z}/m\mathbb{Z}$ .

In the case of vector spaces we do something similar: we take G and H to be additive groups of vector spaces, and then make the quotient group into a vector space by defining scalar multiplication.

Let V be a finite dimensional F-vector space and U a subspace. We call two vectors  $v, v' \in V$  equivalent if  $v - v' \in U$ . The coset  $v + U = \{v' \in V : v - v' \in U\}$  is the equivalence class of v, and the set  $V/U = \{v + U : v \in V\}$  of all cosets is called the quotient space of V by U. **Example.** Consider the subspace  $U = \{0000, 1011, 0101, 1110\}$  of  $V = \mathbb{F}_2^4$ . One coset is U = 0 + U itself; for finding another coset, choose any vector in  $\mathbb{F}_2^4 \setminus U$ , for example 1000, and form 1000 + U; next pick a vector not in U or 1000 + U, like 0100 etc. Eventually you will find the complete list of cosets

 $\begin{array}{l} 0000+U=\{0000,1011,0101,1110\}\\ 1000+U=\{1000,0011,1101,0110\},\\ 0100+U=\{0100,1111,0001,1010\},\\ 0010+U=\{0010,1001,0111,1100\}. \end{array}$ 

As you can see, each coset has the same number of elements. This is not accidental:

**Proposition 3.1.** Let  $U \subseteq V$  be vector spaces over  $\mathbb{F}_q$ ; then

- 1. each vector  $v \in V$  is contained in some coset;
- 2. cosets are either equal or disjoint;
- 3. each coset has  $q^{\dim U}$  elements.

*Proof.* The first claim is trivial since  $v \in v + U$  (note that  $0 \in U$  since U is a vector space).

Now let v + U and v' + U be cosets. If these cosets are not disjoint, then there must exist an element  $x \in V$  such that  $x \in v + U$  and  $x \in v' + U$ . By definition, this means that  $v - x \in U$  and  $v' - x \in U$ . Since U is a vector space, this implies  $v - v' = (v - x) - (v' - x) \in U$ , which in turn shows that v + U = v' + U, proving the second claim.

Finally, given a  $v \in V$  we can define a map  $U \longrightarrow v + U$  by putting  $f(u) = v + u \in v + U$ . This map is injective: if f(u) = f(u'), then v + u = v + u', hence u = u'. Moreover, f is bijective since every element in v + U has the form v + u for some  $u \in U$ , hence is the image of u. Thus f is a bijection, and this implies that #U = #(v + U). Since  $\#U = q^{\dim U}$ , this proves the last claim.

At this point, V/U is just a set. We can give this quotient space the structure of an *F*-vector space in the following way: we define addition of vectors by (v + U) + (w + U) = (v + w) + U and scalar multiplication by  $\lambda(v + U) = \lambda v + U$ . Since we defined operations on equivalence classes using representatives, we have to check that everything's well defined.

Assume therefore that v + U = v' + U and w + U = w' + U. Then

**Exercise 3.1.** Show that scalar multiplication is well defined.

**Exercise 3.2.** Check that V/U satisfies the axioms of an F-vector space.

**Exercise 3.3.** Let  $V = \mathbb{R}^2$  and let U be the first diagonal, i.e., the subspace of V generated by (1, 1). Compute and simplify the following expressions:

- 1. ((3,0) + U) + ((-1,2) + U) = ;
- 2. 7((1,2) + U) = .

**Exercise 3.4.** Show that  $\dim V/U = \dim V - \dim U$  whenever V is a finite dimensional vector space and U some subspace.

# 3.2 Decoding Linear Codes

A decoding method introduced by Slepian in 1960 uses the quotient space structure on V/C.

### **Slepian Decoding**

Suppose someone sends a codeword v, and that a possibly corrupted word v' is received. We want to find the codeword with minimal Hamming distance to v'. Here's how to do it: the vector v' defines a coset  $v' + C \in V/C$ . Let e be a codeword with minimal weight in v' + C; then v' + C = e + C, hence w = v' - e is a codeword, and actually a code word with minimal distance to v': for if w' were a codeword with even smaller distance to v', then e' = v' - w' would have smaller weight than e: contradiction. Observe, however, that there might be other codewords with the same distance to v' as w.

**Definition 3.2.** A word  $e \in x + C$  with minimal weight is called a coset leader of x + C.

Thus in principle decoding is easy: when you receive v', look for the coset leader e of v' + W, and declare that v' - e was the codeword that was sent. If v' happens to be a codeword, then of course v' + C = 0 + C, hence 0 is the coset leader, and v' is regarded as being uncorrupted.

As an example, consider the code  $C = \{0000, 1011, 0101, 1110\}$  with generator matrix  $G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$ . We have already computed the complete list of cosets

 $\begin{array}{l} 0000+C=\{0000,1011,0101,1110\},\\ 1000+C=\{1000,0011,1101,0110\},\\ 0100+C=\{0100,1111,0001,1010\},\\ 0010+C=\{0010,1001,0111,1100\}. \end{array}$ 

The representatives of the cosets we have chosen have already minimal weight, hence they are coset leaders. An array of cosets such as the one above, where each vector is the sum of the coset leader and the corresponding vector at the top of its column, is called a *Slepian array*.

Assume you receive the word v' = 1111. This is not a codeword; its coset has coset leader e = 0100, hence v = v' - e = 1011 is the codeword with minimal distance from v'; if the cosets form a Slepian array as above, this is just the vector on top of the column that 1111 is in.

**Exercise 3.5.** Which single errors does this code correct?

The problem with this method of decoding is first that table look-ups are slow if the table is large: we need something more efficient.

## Syndrome Decoding

Let C be a binary linear code with parity check matrix H; then the vector  $syn(x) = xH^T$  is called the syndrome of the word x. These syndromes will now be used to simplify the decoding process. We start with a simple observation:

**Lemma 3.3.** Two words  $x, y \in \mathbb{F}_2^n$  belong to the same coset v + C if and only if  $\operatorname{syn}(x) = \operatorname{syn}(y)$ .

*Proof.* We know that

$$\begin{aligned} x, y \in v + C & \iff \quad x - y \in C \\ & \iff \quad (x - y)H^T = 0 \\ & \iff \quad xH^T = yH^T, \end{aligned}$$

and this proves the claim.

Thus each word in a coset has the same syndrome, namely that of the coset leader. This leads to the following algorithm for decoding linear codes: first, we precompute the Slepian array for our code, but store only the list of coset leaders and their syndromes.

- 1. Calculate the syndrome  $syn(x) = xH^T$  of the received word  $x \in \mathbb{F}_2^n$ ;
- 2. Scan the list of syndromes and find the coset leader e with the syndrome syn(e) = syn(x);
- 3. Decode x as c = x e.

This is essentially the same as Slepian decoding, except that we don't have to store (and search through) the whole Slepian array with  $2^n$  entries but only the  $2^{n-k}$  coset leaders.

3. Working with Linear Codes 3.3 More on Parity Check Matrices

## 3.3 More on Parity Check Matrices

In this section we uncover relations between parity check matrices H and the minimum distance, and give a simple way of computing H from standard generator matrices. First we study the effect of multiplying a vector x by  $H^T$ :

**Lemma 3.4.** Let C be a linear code of type [n, k], and H an arbitrary parity check matrix for C. Let  $H = (h_1, \ldots, h_n)$ , that is, let  $h_i$  denote the *i*-th column of H. If c is a codeword and you receive x, let  $e = (e_1, \ldots, e_n) = x - c$  denote the error. Then  $\operatorname{syn}(x) = \operatorname{syn}(e) = e_1(h_1)^T + \ldots + e_n(h_n)^T$ .

*Proof.* The fact that syn(x) = syn(e) follows from Lemma 3.3. Now

$$syn(e) = eH^{T} = (e_{1}, \dots, e_{n}) \begin{pmatrix} h_{1,1} & \dots & h_{n-k,1} \\ \vdots & \ddots & \vdots \\ h_{1,n} & \dots & h_{n-k,n} \end{pmatrix}$$
$$= (e_{1}h_{1,1} + \dots + e_{n}h_{1,n}, \dots, e_{1}h_{n-k,1} + \dots + e_{n}h_{n-k,n})$$
$$= e_{1}(h_{1,1}, \dots, h_{n-k,1}) + \dots + e_{n}(h_{1,n}, \dots, h_{n-k,n})$$
$$= e_{1}h_{1}^{T} + \dots + e_{n}h_{n}^{T}$$

as claimed.

Let's consider a binary linear code; if wt(e) = 0, no error has been made. If wt(e) = 1, say  $e_i = 1$  and  $e_j = 0$  for  $j \neq i$ , then  $syn(e) = h_i^T$ , and the syndrome is just the transpose of the *i*-th column of *H*. If wt(e) = 2, then the syndrome of *e* is the transpose of the sum of the *i*-th and the *j*-th columns of *H* etc.

### **Decoding Hamming Codes**

Let  $\operatorname{Ham}(r)$  denote the Hamming code of type [n, n-r, 3], where  $n = 2^r - 1$ , and let G and H be generator matrix and parity check matrix of  $\operatorname{Ham}(r)$ . In order to get a decoding algorithm that gives us the simple method we know from Hamming's [7, 4]-code we pick H in a special way: we choose the columns in such a way that the *i*-th column  $(h_1, \ldots, h_r)$  represents the binary expansion of *i*, namely  $(i)_2 = h_r 2^{r-1} + \ldots + h_2 2 + h_1$ . In the special case r = 3, this gives us the parity check matrix we constructed in (2.3).

Now assume we receive a word  $x \in \mathbb{F}_2^n$ . We compute the syndrome  $\operatorname{syn}(x) = xH^T$ ; if  $\operatorname{syn}(x) = 0$ , then x is a codeword, and we assume that no error has been made.

If  $\operatorname{syn}(x) \neq 0$ , assume that  $\operatorname{wt}(e) = 1$  (Hamming codes are 1-errorcorrecting, so we can't do any better). Then  $\operatorname{syn}(x) = \operatorname{syn}(e) = eH^T$  is the transpose of the *i*-th column; since the *i*-th column is the binary expansion of *i*, all we have to do is compute  $\operatorname{syn}(x)$ , interpret is as a binary expansion for some number *i*, and correct *x* by changing the *i*-th bit.

### Minimum Distance

We will use Lemma 3.4 to give a method for computing the minimum distance of a code from its parity check matrix:

**Proposition 3.5.** The minimum distance of a linear code  $C \subseteq \mathbb{F}_q^n$  is the smallest number  $\delta$  of dependent columns of a parity check matrix H of C.

For an example, consider the parity check matrix of  $\operatorname{Ham}(r)$ : it consists of all the different nonzero vectors in  $\mathbb{F}_2^r$ . Clearly there are no two dependent columns since none is the zero vector and since no two coincide. But the columns  $(100\cdots 0)^T$ ,  $(010\cdots 0)^T$  and  $(110\cdots 0)^T$  clearly are dependent, hence the minimum distance equals 3.

*Proof.* First assume that  $c = (c_1, \ldots, c_n) \in C$  is a codeword of weight d = d(C). Then by Lemma 3.4,  $0 = cH^T = c_1h_1^T + \ldots + c_nh_n^T$  is the transpose of a linear combination of d columns of the parity check matrix H (involving those d columns for which  $e_i \neq 0$ ). Thus these d columns are linearly dependent, and we have  $\delta \leq d$ .

Conversely, let  $h_{i_1}, \ldots, h_{i_{\delta}}$  be a minimal set of dependent columns, say  $\lambda_{i_1}h_{i_1} + \ldots + \lambda_{i_{\delta}}h_{i_{\delta}} = 0$ ; let  $c \in \mathbb{F}_2^n$  be the word with  $\lambda_{i_j}$  in positions  $i_j$   $(j = 1, \ldots, d)$  and 0s everywhere else; observe that wt $(c) = \delta$ . We know that  $cH^T = c_1h_1^T + \ldots c_nh_n^T$ ; in this sum, only those terms with  $c_i \neq 0$  survive, and we find  $cH^T = \lambda_{i_1}h_{i_1} + \ldots + \lambda_{i_{\delta}}h_{i_{\delta}} = 0$ . Thus  $cH^T = 0$ , and c is a codeword of weight  $\delta$ , proving that  $\delta \geq d$ .

One more useful thing: we say a generator matrix G of a code C of type [n,k] is in standard form if  $G = (I_k|A)$  for the  $k \times k$  unit matrix  $I_k$  and some  $(n-k) \times k$ -matrix A.

**Example.**  $G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} = (I_2|A)$  with  $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$  is a standard generator matrix for the code  $C = \{0000, 1011, 0101, 1110\}$  of type [4, 2].

**Proposition 3.6.** If  $G = (I_k|A)$  is a standard generator matrix of a linear code C of type [n, k], then  $H = (-A^T|I_{n-k})$  is a parity check matrix for C.

*Proof.* The matrix H has the right dimension and its rows are independent (because of the presence of  $I_{n-k}$ ). Thus it is sufficient to check that  $GH^T = 0$ , which is a routine computation.

# 3.4 Probability

In this section we will make some simple remarks on the probability that messages received are decoded correctly, or that all errors are detected. For this end, we have to make assumptions about the probability that certain

#### 3. Working with Linear Codes

errors occur, and we shall restrict our attention to the case where this probability does not depend on time and is the same for every bit. Thus given a codeword  $c \in \mathbb{F}_2^n$ , we shall assume that the probability that the *i*-th bit is transmitted incorrectly is a constant p; we call p the error probability of a symmetric channel. Of course  $0 \le p \le 1$ , but in fact we may assume that  $0 \le p \le \frac{1}{2}$ : if the probability of an error per bit is  $> \frac{1}{2}$ , we simply send messages with 0 and 1 replaced (i.e. we make a deliberate error in every bit) guaranteeing that the bits received have error probability  $\le \frac{1}{2}$ .

Now assume that a codeword of length n is transmitted. The probability that an individual bit is received correctly is 1-p, hence the probability that no error occurs is  $(1-p)^n$ . Next, the probability that a single error occurs in a specified location is  $p(1-p)^{n-1}$ , and the probability that i errors occur in i specified locations is  $p^i(1-p)^{n-i}$ .

**Theorem 3.7.** Let  $C \subseteq \mathbb{F}_q^n$  be a linear code, and let  $a_i$  denote the number of codewords with weight *i*. Then the probability that a message received contains an error that is not detected by C is given by

$$\sum_{i=1}^{n} a_i p^i (1-p)^{n-i} (q-1)^{-i}.$$

In particular, if C is a binary code, that is, if q = 2, this probability is

$$\sum_{i=1}^{n} a_i p^i (1-p)^{n-i} = W_C (1-p,p) - (1-p)^n,$$

where  $W_C(X, Y)$  is the weight enumerator of C.

*Proof.* Let c be a codeword, and assume that e is the error made in the transmission, that is, put e = x - c where x is the received message. Since C is linear, x will be a codeword if and only if e is. Thus possible errors will go undetected if e is a codeword different from 0.

Assume that e has weight i = wt(e). The probability of making i errors in i specified locations is  $p^i(1-p)^{n-i}$ , and the number of choices for these ilocations in  $\binom{n}{i}$ . Thus  $\binom{n}{i}p^i(1-p)^{n-i}$  is the probability that the error e has weight i.

Now there are  $\binom{n}{i}(q-1)^i$  words of weight *i* since there are  $\binom{n}{i}$  choices for the nonzero positions, and q-1 possible choices for each of these positions. Thus the probability that the error word *e* of weight *i* is a codeword is  $a_i/\binom{n}{i}(q-1)^i$ .

Now the probability that e is a codeword of weight i is the probability that e has weight i times the probability that a word of weight i is a codeword, namely

$$\operatorname{prob}\left[\operatorname{wt}(e) = i, e \in C\right] = \frac{a_i}{\binom{n}{i}(q-1)^i} \cdot \binom{n}{i} p^i (1-p)^{n-i}$$
$$= a_i p^i (1-p)^{n-i} (q-1)^{-i}.$$

Thus the probability that e is a *nonzero* codeword is the sum of these expressions from i = 1 to i = n, and this proves the claim.

Observe that this expression can also be written as

$$\sum_{i=1}^{n} a_i p^i (1-p)^{n-i} (q-1)^{-i} = W_C \left(1-p, \frac{p}{q-1}\right) - (1-p)^n,$$

where  $W_C(X, Y)$  is the weight enumerator of the code C; the correction term  $(1-p)^n$  is due to the summation starting at i = 1. Thus the probability that errors go undetected can be computed from the weight enumerator.

This result emphasizes the importance of weight enumerators. There is more to them, however, than just probability questions: given a linear code of type [n, k, d], we cannot determine the minimum distance of the dual code  $C^{\perp}$ . However, given the weight enumerator  $w_C(X, Y)$  of C, we can compute the weight enumerator of the dual code! Note that the type of a code can be computed from its weight enumerator.

Actually, the relation between the weight enumerators of dual code is a rather deep result (when compared with the proofs we did so far). In any case, here's

**Theorem 3.8.** For a binary linear code C, the MacWilliams identity

$$w_{C^{\perp}}(X,Y) = \frac{1}{\#C}w_C(X+Y,X-Y)$$

relates the weight enumerators of C and  $C^{\perp}$ .

Before we start proving this identity, let us see what it says. Let  $a_i$  denote the number of codewords of C with weight i, and let  $b_i$  denote the corresponding number for  $C^{\perp}$ . Then

$$w_C(X,Y) = X^n + a_1 X^{n-1} Y + \dots + a_n Y^n,$$
  
$$w_{C^{\perp}}(X,Y) = X^n + b_1 X^{n-1} Y + \dots + b_n Y^n$$

since  $a_0 = b_0 = 1$ . Let us now compute  $w_C(X + Y, X - Y)$ . We find

$$w_{C}(X+Y,X-Y) = (X+Y)^{n} + a_{1}(X+Y)^{n-1}(X-Y) + a_{2}(X+Y)^{n-2}(X-Y)^{2} + \dots + a_{n-1}(X+Y)(X-Y)^{n-1} + a_{n}(X-Y)^{n} = X^{n}(1+a_{1}+a_{2}+\dots+a_{n}) + X^{n-1}Y(n+(n-2)a_{1}+(n-4)a_{2} + \dots + (2-n)a_{n-1}-na_{n}) + \dots$$

Thus the coefficient of  $X^n$  in  $w_C(X+Y, X-Y)$  equals  $1+a_1+a_2+\ldots+a_n$ , which is just the number of codewords of C; in other words, this coefficient is  $\#C \cdot b_1$  as predicted by the MacWilliams identity.

For the proof, we need a few lemmas.

**Lemma 3.9.** Let C be a binary linear code of type [n,k]. Then for any  $y \in \mathbb{F}_2^n \setminus C^{\perp}$ , the sets  $C_0 = \{x \in C : x \cdot y = 0\}$  and  $C_1 = \{x \in C : x \cdot y = 1\}$  have equal cardinality.

Of course this is not true if  $y \in C^{\perp}$  because then  $x \cdot y = 0$  for all  $x \in C$ .

*Proof.* Let c be any codeword with cy = 1 (if cy = 0 for all codewords, then  $y \in C^{\perp}$  contradicting the assumption). Then f(x) = x + c defines a map  $f: C_0 \longrightarrow C_1$  because  $x \in C_0$  implies  $f(x) \cdot y = (x + c) \cdot y = x \cdot y + c \cdot y = 0 + 1 = 1$ . Assume that f(x) = f(x'); then x + c = x' + c, hence x = x', so f is clearly injective. Moreover, f is surjective since, given  $x \in C_1$ , we have x = f(x+c) due to the fact that  $x+c \in C_0$  and 2c = 0. Thus f is a bijection, and the claim follows.

**Lemma 3.10.** Let C be a binary linear code of type [n,k]. Then for any  $y \in \mathbb{F}_2^n$  we have

$$\sum_{x \in C} (-1)^{x \cdot y} = \begin{cases} 2^k & \text{if } y \in C^\perp; \\ 0 & \text{if } y \notin C^\perp; \end{cases}$$

This formula is a very special case of what is called an orthogonality relation in the theory of characters (a part of group theory belonging to representation theory). The function  $\chi_y$  defined by  $\chi_y(x) = (-1)^{x \cdot y}$  is a character on  $\mathbb{F}_2^n$  with values in  $\mathbb{F}_2$ .

*Proof.* If  $y \in C^{\perp}$ , then  $x \cdot y = 0$  for any  $x \in C$ , hence  $\sum_{x \in C} (-1)^{x \cdot y} = \sum_{x \in C} 1 = 2^k$ . If  $y \notin C^{\perp}$ , then  $x \cdot y = 0$  as often as  $x \cdot y = 1$  by Lemma 3.9, hence  $\sum_{x \in C} (-1)^{x \cdot y} = 0$ .

The last lemma we need is

**Lemma 3.11.** For an arbitrary  $x \in \mathbb{F}_2^n$  we have the following polynomial identity:

$$\sum_{y \in \mathbb{F}_2^n} (-1)^{x \cdot y} z^{\operatorname{wt}(y)} = (1-z)^{\operatorname{wt}(x)} (1+z)^{n - \operatorname{wt}(x)}.$$

*Proof.* Write  $x = (x_1, \ldots, x_n)$  and  $y = (y_1, \ldots, y_n)$ ; then

$$\sum_{y \in \mathbb{F}_2^n} (-1)^{x \cdot y} z^{\operatorname{wt}(y)} = \sum_{y_1=0}^1 \sum_{y_2=0}^1 \cdots \sum_{y_n=0}^1 (-1)^{x_1 y_1 + \dots + x_n y_n} z^{y_1 + \dots + y_n}$$
$$= \sum_{y_1=0}^1 \cdots \sum_{y_n=0}^1 \left( \prod_{i=1}^n (-1)^{x_i y_i} z^{y_i} \right)$$
$$= \prod_{i=1}^n \left( \sum_{j=0}^1 (-1)^{j x_i} z^j \right).$$

In fact, if we multiply out the expression

Error-Correcting Codes

$$\prod_{i=1}^{n} \left( (-1)^0 \cdot x_i z^0 + (-1)^{1 \cdot x_i} z^1 \right),$$

then we get exactly the sum over the products we started with. Finally,

$$\sum_{y \in \mathbb{F}_2^n} (-1)^{x \cdot y} z^{\operatorname{wt}(y)} = (1-z)^{\operatorname{wt}(x)} (1+z)^{n-\operatorname{wt}(x)}$$
$$\sum_{j=0}^1 (-1)^{jx_i} z^j = \begin{cases} 1+z & \text{if } x_i = 0, \\ 1-z & \text{if } x_i = 1. \end{cases} \square$$

since

Let us now introduce the weight enumerator in one variable z: if we divide  $W_C(X,Y)$  through by  $X^n$  and substitute Y/X = z, then we find

$$X^{-n}W_C(X,Y) = 1 + a_1z + a_2z^2 + \ldots + a_nz^n = w_C(z).$$

Conversely, given  $w_C(z)$  we can compute  $W_C(X, Y)$  by substituting z = Y/Xand clearing denominators.

Proof of Thm. 3.7. The idea is to express the polynomial

$$f(z) = \sum_{x \in C} \left( \sum_{y \in \mathbb{F}_2^n} (-1)^{x \cdot y} z^{\operatorname{wt}(y)} \right)$$

in two different ways.

First, using Lemma 3.11 we get

$$f(z) = \sum_{x \in C} (1-z)^{\operatorname{wt}(x)} (1+z)^{n-\operatorname{wt}(x)}$$
$$= (1+z)^n \sum_{x \in C} \left(\frac{1-z}{1+z}\right)^{\operatorname{wt}(x)}$$
$$= (1+z)^n w_C \left(\frac{1-z}{1+z}\right).$$

On the other hand we can write

$$f(z) = \sum_{y \in \mathbb{F}_2^n} z^{\operatorname{wt}(y)} \left( \sum_{x \in C} (-1)^{x \cdot y} \right)$$
$$= \sum_{y \in C^{\perp}} z^{\operatorname{wt}(y)} 2^k \qquad \text{by Lemma 3.9}$$
$$= 2^k w_{C^{\perp}}(z).$$

Equating the two expressions and substituting z = Y/X establishes the MacWilliams identity.

#### 3. Working with Linear Codes

Finally, let us compute the weight enumerators of Hamming codes  $\operatorname{Ham}(r)$ . We claim

**Theorem 3.12.** Let C = Ham(r) and  $n = 2^r - 1$ . Then

$$W_C(X,Y) = \frac{1}{n+1}(X+Y)^n + \frac{n}{n+1}(X+Y)^{(n-1)/2}(X-Y)^{(n+1)/2}.$$

We will give two proofs: the first one computes the weight enumerator directly and proceeds in several stages; the second proof is testimony to the fact that sometimes computing the weight enumerator via the dual code plus MacWilliams is *much* easier.

The first step in the direct proof is the following recursion formula for the numbers  $a_i$  of codewords with weight i of Ham(r):

**Lemma 3.13.** For  $2 \le i \le n$  we have

$$ia_i + a_{i-1} + (n-i+2)a_{i-2} = \binom{n}{i-1}.$$
 (3.1)

Moreover, the equation continues to hold for i = 1 if we put  $a_{-1} = 0$ ; it is valid for i = n + 1 if  $a_{n+1} = 0$ .

*Proof.* Consider subsets I of  $\{1, 2, ..., n\}$  with i-1 elements. For each subset I, we define a word  $b_I = (b_1, ..., b_n)$  by demanding that

$$b_k = \begin{cases} 1 & \text{if } k \in I; \\ 0 & \text{otherwise.} \end{cases}$$

Let  $\operatorname{syn}(b_I) = (b_1, \ldots, b_n) \cdot H^T$  denote the syndrome of  $b_I$ . Now we distinguish three types of subsets I according to these syndromes: recall that  $\operatorname{syn}(x)$  is the transpose of a (possibly empty) sum of columns of the parity check matrix; in the case of Hamming codes  $\operatorname{Ham}(r)$ , every nonzero vector of length r is the transpose of a column of H because H contains all nonzero columns.

- 1. Subsets I such that  $syn(b_I) = 0$ . This means that  $b_I$  is actually a codeword, and since  $wt(b_I) = i 1$ , we find that the number of subsets I of this kind equals  $a_{i-1}$ .
- 2. Subsets I such that  $\operatorname{syn}(b_I) = h_j^T$  for some index  $j \in I$ . In this case, consider the set  $J = I \setminus \{j\}$ . Then  $\operatorname{syn}(b_J) = 0$ , so  $b_J \in C$  is a codeword of weight i 2. Given such a codeword, writing a 1 at any of the n (i-2) positions where  $b_J$  has 0s gives a word  $b_I$ . Moreover, the words  $b_I$  constructed from some J are different from the words  $b_{I'}$  constructed from a different J': this is because  $d(b_J, b_I) = 1$  and  $d(b_{J'}, d_{I'}) = 1$ , therefore  $b_I = b_{I'}$  would imply that  $d(b_J, b_{j'}) \leq 2$ , hence J = J' since Hamming codes have minimum distance 3.

What this means is that all sets I of this type are constructed from the  $a_{i-2}$  codewords c of weight i-2 by writing 1 in one of the n-i+2 positions where c has 0s. Thus the number of such subsets is  $(n-i+2)a_{i-2}$ .

48Franz Lemmermeyer

3. Subsets I such that  $syn(b_I) = h_j^T$  for some index  $j \notin I$ . In this case, consider the set  $J = I \cup \{j\}$ . Then  $b_J$  has weight *i*, and there are  $a_i$  such words. The words  $b_I$  are constructed from the words  $b_J$  by eliminating one of the 1s (there are i of them) in some  $b_J$ . Different Js lead to different Is (same argument as above:  $d(C) \geq 3$ ), hence there are  $ia_i$ different sets I of this type.

Here's what we have done: we have counted the  $\binom{n}{i-1}$  sets in a different way; comparing results gives us  $\binom{n}{i-1} = ia_i + a_{i-1} + (n-i+2)a_{i-2}$ , which is what we wanted to prove.

The last claim follows from  $a_0 = 1$ ,  $a_1 = 0$  and  $\binom{n}{0} = 1$ ; moreover,  $a_n = 1$ since  $(1 \cdots 1) \in \text{Ham}(r)$ , and  $a_{n-1} = 0$ : In fact,  $(1 \cdots 1)H^T$  is the transpose of the sum of all columns of H, that is, of all the (nonzero) elements of the group  $\mathbb{F}_2^r$ , which is the zero element by Lagrange's theorem. Thus  $(1 \cdots 1) \in \text{Ham}(r)$ . Moreover, if c is a word of weight n-1, then  $cH^T$  is the transpose of the sum of all (nonzero) elements in  $\mathbb{F}_2^r$  except one, which – again by Lagrange's theorem – is equal to the transpose of the column missing from this sum; in particular,  $cH^T \neq 0$ , so c is not a codeword. 

The next step is transforming the recurrence formula for the  $a_i$  into a differential equation for the weight enumerator:

**Lemma 3.14.** The function  $A(z) = a_0 + a_1 z + a_2 z^2 + ... + a_n z^n$  satisfies the differential equation

$$(1-z)^2 A'(z) + (1+nz)A(z) = (1+z)^n.$$
(3.2)

*Proof.* Multiply (3.1) through by  $z^{i-1}$  and form the sum from i = 1 to n+1. The right hand side is

$$\sum_{n=1}^{n+1} \binom{n}{i-1} z^{i-1} = (1+z)^n.$$

On the left hand side we have

$$\sum_{i=1}^{n+1} [ia_i z^{i-1} + a_{i-1} z^{i-1} + (n-i+2)a_{i-2} z^{i-1}].$$

Clearly 
$$\sum_{i=1}^{n+1} i a_i z^{i-1} = A'(z)$$
 and  $\sum_{i=1}^{n+1} a_{i-1} z^{i-1} = A(z)$ . Next,  
 $\sum_{i=1}^{n+1} n a_{i-2} z^{i-1} = nz \sum_{i=1}^{n+1} a_{i-2} z^{i-2} = nz [A(z) - z^n],$ 

and finally

3. Working with Linear Codes

$$\sum_{i=1}^{n+1} (i-2)a_{i-2}z^{i-1} = z^2 \sum_{i=1}^{n+1} (i-2)a_{i-2}z^{i-3} = z^2 [A'(z) - nz^{n-2}].$$

Combining everything gives us the desired differential equation.

Now we know from the theory of first order differential equations that equations such as (3.2) combined with a initial condition (in our case A(0) = $a_0 = 1$ ) have a unique solution; thus we may simply verify that

$$A(z) = \frac{1}{n+1}(1+z)^n + \frac{n}{n+1}(1+z)^{(n-1)/2}(1-z)^{(n+1)/2}$$

solves this problem, and then write this polynomial in z as a homogeneous polynomial in X and Y. Checking that A(z) does it is no problem, but, for what it's worth, here's an actual solution of the differential equation:<sup>1</sup> Write the ODE in the form M dz + N dA = 0, where

$$M = (1 + nz)A(z) - (1 + z)^n$$
 and  $N = (1 - z^2)A'(z)$ .

This form is not exact since  $\frac{\partial M}{\partial A} = 1 + nz \neq -2z = \frac{\partial N}{\partial z}$ . We now try to find an integrating factor, that is, a function  $\mu(z)$  such that  $\frac{\partial \mu M}{\partial A} = \frac{\partial \mu N}{\partial z}$ . The standard approach shows that

$$\begin{split} \log \mu(z) &= \int \frac{1}{N} \Big\{ \frac{\partial M}{\partial A} - \frac{\partial N}{\partial z} \Big\} \, \mathrm{d}z \quad = \int \frac{1 + (n+2)z}{1 - z^2} \, \mathrm{d}z \\ &= \int \Big\{ \frac{n+3}{2(1-z)} - \frac{n+1}{2(1+z)} \Big\} \, \mathrm{d}z \\ &= -\frac{n+3}{2} \log(1-z) - \frac{n+1}{2} \log(1+z), \end{split}$$

hence

$$\mu(z) = (1+z)^{-\frac{n+1}{2}}(1-z)^{-\frac{n+3}{2}}.$$

Observe that we do not have to worry about integration constants since any nonzero multiple of the function  $\mu$  will do. Now the form  $\mu M dz + \mu N dA = 0$ is exact, so by Poincarés Lemma it is closed, hence there exists a function F(A, z) such that  $\frac{\partial F}{\partial z} = \mu M$  and  $\frac{\partial F}{\partial A} = \mu N$ . From

$$\frac{\partial F}{\partial A} = \mu M = (1+z)^{-\frac{n-1}{2}} (1-z)^{-\frac{n+1}{2}}$$

we get

$$F(A,z) = (1+z)^{-\frac{n-1}{2}}(1-z)^{-\frac{n+1}{2}}A + G(z)$$

49

<sup>&</sup>lt;sup>1</sup> Provided by Eli de Palma.

for some function G depending only on z. Using  $\frac{\partial F}{\partial z}=\mu M$  we get

$$G'(z) = -(1+z)^{\frac{n-1}{2}}(1-z)^{-\frac{n+3}{2}} = -\left(\frac{1+z}{1-z}\right)^{\frac{n-1}{2}}\frac{1}{(1-z)^2},$$

so the substitution  $u = \frac{1+z}{1-z}$  gives, using  $\frac{du}{dz} = \frac{2}{(1-z)^2}$ ,

$$G'(u) = -\frac{1}{2}u^{\frac{n-1}{2}} du$$
, hence  $G(u) = -\frac{1}{n+1}u^{\frac{n+1}{2}}$ 

and therefore

$$G(z) = -\frac{1}{n+1}(1+z)^{\frac{n+1}{2}}(1-z)^{-\frac{n+1}{2}}.$$

Thus we have

$$F(A,z) = (1+z)^{-\frac{n-1}{2}}(1-z)^{-\frac{n+1}{2}}A - \frac{1}{n+1}(1+z)^{\frac{n+1}{2}}(1-z)^{-\frac{n+1}{2}};$$

the original ODE says dF = 0, so F = c is a constant, and solving for A gives

$$A(z) = \frac{1}{n+1}(1+z)^n + c \cdot (1+z)^{\frac{n-1}{2}}(1-z)^{\frac{n+1}{2}}.$$
(3.3)

The initial condition A(0) = 1 implies that  $c = \frac{n}{n+1}$ .

Here's a simpler solution involving an educated guess: solving the homogeneous equation  $(1 + nz)A(z) + (1 - z^2)A'(z) = 0$  is straight forward: we find

$$\frac{A'(z)}{A(z)} = -\frac{1+nz}{1-z^2} = \frac{n-1}{2(1+z)} - \frac{n+1}{2(1-z)}$$

hence

$$\log A(z) = \frac{n-1}{2}\log(1+z) - \frac{n+1}{2}\log(1-z),$$

and thus

$$A(z) = (1+z)^{\frac{n-1}{2}}(1-z)^{\frac{n+1}{2}}.$$

Since  $A(z) = \frac{1}{n+1}(1+z)^n$  is a particular solution of the ODE, all solutions are given by (3.3).

OK – let us look at the case r = 3. Here  $n = 2^3 - 1 = 7$ , hence

#### 3. Working with Linear Codes

$$\begin{split} A(z) &= \frac{1}{8}(1+z)^7 + \frac{7}{8}(1+z)^3(1-z)^4 \\ &= \frac{1}{8}[1+7z+21z^2+35z^3+35z^4+21z^5+7z^6+z^7] \\ &\quad +\frac{7}{8}(1-z)(1-3z^2+3z^4-z^6) \\ &= \frac{1}{8}[(1+7)+(7-7)z+(21-21)z^2+(35+21)z^3+(35+21)z^4 \\ &\quad +(21-21)z^5+(7-7)z^6+(1+7)z^7] \\ &= 1+7z^3+7z^4+z^7, \end{split}$$

which gives us the weight enumerator as

$$W_C(X,Y) = X^7 + 7X^4Y^3 + 7X^3Y^4 + Y^7$$

as expected.

Now let us give a second way of computing the weight enumerator of Hamming's codes C = Ham(r). It is much simpler than the first one because we use MacWilliams to reduce it to the calculation of  $W_{C^{\perp}}(X, Y)$ , which in turn is almost trivial once the following lemma is proved:

**Lemma 3.15.** Let C = Ham(r); then every nonzero codeword of  $C^{\perp}$  has weight  $2^{r-1}$ .

Once we know this, we have

$$W_{C^{\perp}}(X,Y) = X^n + nX^m Y^{n-m},$$

where  $m = 2^{r-1} = \frac{n+1}{2}$ . Thus

$$W_C(X,Y) = 2^{-r} \left[ (X+Y)^n + n(X+Y)^m (X-Y)^{n-m} \right]$$
  
=  $\frac{1}{n+1} (X+Y)^n + \frac{n}{n+1} (X+Y)^{\frac{n-1}{2}} (X-Y)^{\frac{n+1}{2}}$ 

Proof of Lemma 3.15. Let  $h_1, \ldots, h_r$  denote the rows of the parity check matrix of C = Ham(r): this is the generator matrix of  $C^{\perp}$ , that is, the matrix whose columns are just the nonzero vectors in  $\mathbb{F}_2^r$ . A nonzero codeword  $c \in C^{\perp}$  is just a nontrivial linear combination of these rows by the definition of the generator matrix:  $c = \lambda_1 h_1 + \ldots + \lambda_r h_r$ . Note that we consider a single word  $c \neq 0$ , so the  $\lambda_i$  are fixed numbers.

We want to count the number n(c) of 0s in c. To this end, write  $h_i = (h_{i1}, \ldots, h_{in})$ , and observe that the *i*th bit in c equals  $\lambda_1 h_{1i} + \ldots + \lambda_r h_{ri}$ . Thus the *i*th bit of c is  $\lambda_1 x_1 + \ldots + \lambda r x_r$ , where  $(x_1, \ldots, x_r)$  is the transpose of the *i*th column of H. Thus n(c) equals the number of nonzero vectors in

$$X = \{(x_1, \ldots, x_r) : \lambda_1 x_1 + \ldots + \lambda r x_r = 0\}.$$

In fact, any nonzero vector in X is the transpose of some column in H.

Now X is easily seen to be a vector space, namely the kernel of the linear map  $\lambda : \mathbb{F}_2^r \longrightarrow \mathbb{F}_2$  defined by  $\lambda(x_1, \ldots, x_r) = \lambda_1 x_1 + \ldots + \lambda r x_r$ . Since not all the  $\lambda_i$  vanish,  $\lambda$  is surjective, hence dim  $X = \dim \ker \lambda = \dim F_2^r - 1 = r - 1$ , and in particular X contains  $2^{r-1} - 1$  non-zero vectors.

Thus  $n(c) = 2^{r-1} - 1$  and  $wt(c) = n - n(c) = 2^{r-1}$ .

# 3.5 New Codes from Old

**Definition 3.16.** Let C be a code of length n over  $\mathbb{F}_2$ .

The parity check extension  $C^+$  of C is the code of length n+1 defined by

 $C^{+} = \{(x_1, \dots, x_{n+1}) \in \mathbb{F}_2^{n+1} : x_1 + \dots + x_n + x_{n+1} = 0\}.$ 

The truncation  $C^-$  is the code of length n-1 defined by

$$C^{-} = \{(x_1, \dots, x_{n-1}) : (x_1, \dots, x_n) \in C \text{ for some } x_n \in \mathbb{F}_2\}.$$

The shortening C' of C is the code of length n-1 given by

$$C' = \{(x_1, \dots, x_{n-1}) : (x_1, \dots, x_{n-1}, 0)\} \in C.\}$$

**Exercise 3.6.** If C is linear, then so are  $C^+$ ,  $C^-$  and C'. What can you say about the type of these codes?

We now show how to compute the weight enumerator of  $C^+$  from the weight enumerator of C.

**Proposition 3.17.** Let C be a binary linear code with weight enumerator  $W_C(X,Y) = X^n + a_1 X^{n-1} Y + \ldots + a_n Y^n$ . Then the weight enumerator of the parity extension code is

$$W_{C^+}(X,Y) = X^{n+1} + b_1 X^n Y + \ldots + b_{n+1} Y^{n+1},$$

where  $b_{2i+1} = 0$  and  $b_{2i} = a_{2i} + a_{2i-1}$  for all *i*.

*Proof.* We know that  $c_{n+1}+c_n+\ldots+c_1=0$  for codewords  $c = (c_1c_2\ldots c_{n+1})$ . Since there must be an even number of 1s in c for the sum of the bits to vanish, we conclude that there are no words of odd weight:  $b_{2i+1} = 0$ .

Now consider  $b_{2i}$ : a word of weight 2i either comes from a codeword in C of weight 2i (by adding a 0) or from a codeword in C of weight 2i + 1 (by adding a 1). Thus  $b_{2i} = a_{2i} + a_{2i-1}$  as claimed.

Corollary 3.18. We have

$$W_{C^+}(X,Y) = \frac{1}{2} \Big[ (X+Y)W_C(X,Y) + (X-Y)W_C(X,-Y) \Big].$$

**Definition 3.19.** The parity check extension of Hamming's [7, 4]-code is called  $e_8$ .

**Exercise 3.7.** Show that  $e_8$  and the repetition code of length 2 (each block of 4 bits is sent twice) are linear codes on  $\mathbb{F}_2^8$ ; compute their generator matrices, verify that the have the same information rate, and check that they have type [8, 4, 4] and [8, 4, 2], respectively.

A code is called *self dual* if  $C = C^{\perp}$ , and *isotropic* if  $C \subseteq C^{\perp}$ .

**Exercise 3.8.** Show that  $e_8$  is a self dual code.

**Proposition 3.20.** Let  $\operatorname{Ham}^*(r)$  denote the parity check extension of the Hamming code with type  $[2^r - 1, 2^r - 1 - r, 3]$ . Then  $\operatorname{Ham}^*(r)$  has type  $[2^r, 2^r - 1 - r, 4]$ 

Proof. All codewords of a parity check extension code are even, therefore the minimum distance, which is the minimal positive weight of codewords, is even, too. Thus  $d(\operatorname{Ham}^*(r)) \geq 4$ . On the other hand, adding a parity bit to any word in  $\operatorname{Ham}(r)$  with weight 3 (and there are such words since  $d(\operatorname{Ham}(r)) = 3$ ) gives a codeword of weight 4, so the minimum distance is  $\leq 4$ . The dimension of codes does not increase by parity extension (because the number of codewords remains unchanged), hence  $\operatorname{Ham}^*(r)$  has type  $[2^r, 2^r - 1 - r, 4]$ .

A simple way of constructing a code out of two codes  $C_1$  and  $C_2$  would be concatenating them:

$$C_1 \oplus C_2 = \{ (x|y) : x \in C_1, y \in C_2 \}.$$

This is, however, not really an interesting code because of the following result.

**Proposition 3.21.** If  $C_1$  and  $C_2$  are binary linear codes of type  $[n_i, k_i, d_i]$ (i = 1, 2), then  $C_1 \oplus C_2$  is linear of type  $[n_1 + n_2, k_1 + k_2, \min\{d(C_1), d(C_2)\}]$ . More exactly, if  $w_1$  and  $w_2$  are their weight enumerators, then the weight enumerator w of  $C_1 \oplus C_2$  is given by  $w(X, Y) = w_1(X, Y) \cdot w_2(X, Y)$ .

*Proof.* The inequality  $d(C_1 \oplus C_2) \leq \min\{d(C_1), d(C_2)\}$  is obvious: assume that  $d(C_1) \leq d(C_2)$  and that  $c_1 \in C_1$  is a codeword with minimal weight  $\operatorname{wt}(c_1) = d(C_1)$ . Then the weight of  $(c_1|0) \in C_1 \oplus C_2$  is  $d(C_1)$ , and this implies the claim.

On the other hand,  $\operatorname{wt}(c_1|c_2) = \operatorname{wt}(c_1) + \operatorname{wt}(c_2) \ge \operatorname{wt}(c_1)$ , so the weight of codewords cannot decrease by concatenating. This gives the inequality in the other direction.

It remains to prove the claim about the weight enumerators (and of course it would have sufficed to prove just this claim). Let  $(c_1|c_2) \in C_1 \oplus C_2$ , and let  $a_i$  and  $b_i$  denote the number of codewords of weight i in  $C_1$  and  $C_2$ , respectively. Then the number of codewords  $(c_1|c_2)$  of weight is  $a_0b_i+a_1b_{i-1}+a_2b_{i-2}+\ldots+a_{i-1}b_1+a_ib_0$ . 54

On the other hand, putting  $m = n_1$  and  $n = n_2$  we have

$$w_1(X,Y) = (a_0 X^m + a_1 X^{m-1} Y + \dots + a_m Y^m),$$
  
$$w_2(X,Y) = (b_0 X^n + b_1 X^{n-1} Y + \dots + b_n Y^n),$$

so the coefficient of  $X^i$  in the product  $w_1w_2$  is  $a_0b_i + a_1b_{i-1} + \ldots + a_ib_0$ . This proves the claim.

A slightly better idea of how to combine two codes is due to Plotkin: let  $C_1, C_2 \subseteq \mathbb{F}_2^n$  be linear codes with the same word length. We define  $C_1 * C_2$  as the set of words (u|u+v) of length 2n, where the first n bits form a codeword  $u \in C_1$ , and where the second n bits are the sum of u and a codeword  $v \in C_2$ . Note that, in general,  $C_1 * C_2$  and  $C_2 * C_1$  are not equivalent (the codes  $C_1 \oplus C_2$  and  $C_2 \oplus C_1$ , are always equivalent, altough they're not always equal).

**Theorem 3.22.** Let  $C_1$  and  $C_2$  be binary linear codes of type  $[n, k_1, d_1]$ and  $[n, k_2, d_2]$ , respectively. Then  $C_1 * C_2$  is a linear code of type  $[2n, k_1 + k_2, \min\{2d_1, d_2\}]$ .

*Proof.* The proof that  $C_1 * C_2$  is linear is left as an exercise.

Consider the map  $f: C_1 \oplus C_2 \longrightarrow C_1 * C_2$  defined by  $(u, v) \longmapsto (u|u+v)$ . This f is linear: we have f((u, v) + (u', v')) = f((u+u', v+v')) = (u+u'|u+u'+v+v') = (u|u+v) + (u'|u''+v') = f((u,v)) + f((u',v')) and  $f(\lambda(u,v)) = f((\lambda u, \lambda v)) = (\lambda u|\lambda u + \lambda v) = \lambda(u|u+v) = \lambda f(u,v)$ . Next, ker  $f = \{(u,v) \in C_1 \oplus C_2 : u = 0, u+v = 0\} = \{(0,0)\}$ , so f is injective. Finally, f is clearly surjective. Thus  $C_1 \oplus C_2 \simeq C_1 * C_2$ , and in particular both spaces have the same dimension  $k_1 + k_2$ .

For proving the claims about  $d(C_1 * C_2)$  we assume that u and v are codewords of minimal positive weight in  $C_1$  and  $C_2$ , respectively, i.e., wt $(u) = d_1$  and wt $(v) = d_2$ . Then wt $(0|0 + v) = d_2$  and wt $(u|u + 0) = 2d_1$ , hence  $d(c_1 * C_2) \leq \min\{2d_1, d_2\}$ .

Now assume that (u|u + v) has minimal positive weight in  $C_1 * C_2$ . If v = 0, then this weight is  $2 \operatorname{wt}(u) \ge 2d_1$ . If  $v \ne 0$ , we have  $\operatorname{wt}(u|u + v) = \operatorname{wt}(u) + \operatorname{wt}(u + v) \ge \operatorname{wt}(v)$ : in fact,  $\operatorname{wt}(u) = d(0, u)$  and  $\operatorname{wt}(u + v) = d(u, v)$ , hence  $\operatorname{wt}(v) = d(0, v) \le d(0, u) + d(u, v) = \operatorname{wt}(u) + \operatorname{wt}(u + v)$ . This proves that  $d(C_1 * C_2) \ge d(C_2)$  if  $v \ne 0$ . Taken together these inequalities show that  $d(c_1 * C_2) \ge \min\{2d_1, d_2\}$ , and since we have already shown the inverse inequality, the proof of the theorem is complete.  $\Box$ 

# 3.6 The Quality of Codes

Our next goal is proving that for codes in  $\mathbb{F}_p^n$ , the information rate and the minimum distance cannot both be large:

3. Basic Concepts

**Proposition 3.23** (Singleton Bound). Let C be a linear code of length n, dimension k, and minimum distance d. Then  $d \le n - k + 1$ .

*Proof.* The subset  $W \subseteq \mathbb{F}_p^n$  defined by

 $W = \{ (x_1, \dots, x_n) \in \mathbb{F}_p^n : x_d = x_{d+1} = \dots = x_n = 0 \}$ 

is actually a subspace of dimension d-1 of  $\mathbb{F}_p^n$ . Since every vector in W has weight  $\leq d-1$ , the only codeword in W is 0, and we find  $W \cap C = \{0\}$ . Thus  $\dim W + \dim C = \dim(W+C) \leq \dim \mathbb{F}_p^n = n$ , and the claim follows.  $\Box$ 

**Exercise 3.9.** The Singleton bound cannot be improved trivially: show that the trivial code  $C = \mathbb{F}_q^k$  has type  $[k, k, 1]_q$  and attains the bound.

**Exercise 3.10.** The Singleton bound is not always attained: show that there does not exist a code of type  $[q^2 + q + 1, q^2 + q - 2, 4]_q$ .

For every  $k \in \mathbb{N}$  put

 $m_q(k) = \sup\{n \in \mathbb{N} : \text{ there is a linear code of type } [n, k, n+1-k]_q\}.$ 

It is a major problem in coding theory to determine  $m_q(k)$  for all  $k \in \mathbb{N}$ .

A non-degenerate linear code of type  $[n, k, n + 1 - k]_q$  is called a MDS (maximal distance separable) code.

Main Conjecture on MDS Codes. For  $2 \le k < q$  we have  $m_q(k) = q+1$ , except when

- q is even and k = 3: then  $m_q(3) = q + 2$ ;
- k = q 1: then  $m_q(q 1) = q + 2$ .

Error-Correcting Codes

# 4. Finite Fields

So far, we were dealing mostly with the field  $\mathbb{F}_2$ , and occasionally have come across the field  $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$  of p elements, where p is prime. These fields are not as suitable as  $\mathbb{F}_2$  in practice because computers prefer to work with 0 and 1 only. In this chapter we will construct new finite fields, in particular finite fields whose number of elements is a power of 2.

Before we can do so, we have to review a few basic facts about the characteristic of rings and fields. Let R be an arbitrary ring. If there exists an integer n such that  $n \cdot 1 = \underbrace{1 + \ldots + 1}_{n \text{ terms}} = 0$ , then the smallest such integer is

called the *characteristic* of the ring; if no such n exists, we say that R has characteristic 0.

If R is a finite ring, the characteristic of R is always nonzero: in fact, consider the ring elements  $1 \cdot 1, 2 \cdot 1, 3 \cdot 1, \ldots$ ; since R is finite, eventually elements must repeat, that is, there are integers m < m' such that  $m \cdot 1 = m' \cdot 1$ . But then  $n \cdot 1 = 0$  for n = m' - m.

The rings  $\mathbb{Z}/n\mathbb{Z}$  have characteristic n, so every integer  $n \ge 1$  occurs as the characteristic of a ring. This is not true for fields:

**Proposition 4.1.** Let R be an integral domain; then  $p = \operatorname{char} R$  is 0 or a prime.

*Proof.* Assume that  $p \cdot 1 = 0$  and p = mn. Then  $(m \cdot 1)(n \cdot 1) = p \cdot 1 = 0$ ; but since R is an integral domain, a product can be 0 only if one of its factors is 0, so we can conclude that  $m \cdot 1 = 0$  or  $n \cdot 1 = 0$ . Since p was chosen to be the minimal integer with  $p \cdot 1 = 0$ , this implies  $m \ge p$  or  $n \ge p$ , that is, m = p or n = p. In other words: all factorizations of p are trivial, hence p is prime.  $\Box$ 

Given any prime p, the ring  $\mathbb{Z}/p\mathbb{Z}$  of residue classes modulo p is a finite field with p elements and characteristic p. Thus all primes occur as the characteristic of some finite field.

There are other differences between finite rings and fields that one has to be aware of, in particular since we are so used to working over  $\mathbb{Z}$ ,  $\mathbb{Q}$  or  $\mathbb{F}_p$  that we sometimes fall into traps.

**Lemma 4.2.** If E is a subfield of F, then E and F have the same zero and identity elements.

Isn't this obvious? Well, it is not: consider the ring  $R = M_2(F)$  of  $2 \times 2$ -matrices over some field F. The set of all matrices  $S = \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix}$  forms a subring of R with zero  $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$  and identity  $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ .

Proof of Lemma 4.2. Let 0 be the zero element of F and 0' that of E. Then 0' + 0' = 0' in E (and therefore in F). Moreover, we have 0 + 0' = 0' in F, so comparing yields 0 + 0' = 0' + 0', and cancelling 0' gives 0 = 0'. Observe that this proof works even when E and F are arbitrary rings.

Similarly, let 1 and 1' denote the identities in F and in E, respectively. Then  $1' \cdot 1' = 1'$  in E and therefore in F, as well as  $1 \cdot 1' = 1'$  in F. Comparing these equations gives  $1 \cdot 1' = 1' \cdot 1'$ ; but in arbitrary rings, we cannot simply cancel 1'. If F is a field, however, then  $1' \neq 0' = 0$  by the field axioms, and cancelling the nonzero factor 1' yields 1 = 1'.

Observe also that polynomials of degree n over a ring may have more than n roots: the polynomial  $X^2 - 1$  over  $(\mathbb{Z}/8)\mathbb{Z}[X]$  has the four different roots  $x \equiv 1, 3, 5, 7 \mod 8$ . Over fields, such accidents cannot happen: If  $\alpha \in k$  is a a root of  $f \in k[X]$ , then  $f(X) = (X - \alpha)g(X)$  for some polynomial of degree n - 1 by long division, and since k is a field, thus has no zero divisors, a root of f must be a root of  $X - \alpha$  or of g; induction does the rest.

Note that  $X^2 - 1 = (X - 1)(X + 1)$  over  $(\mathbb{Z}/8)\mathbb{Z}[X]$ , and that 3 is a root of the right hand side even though it is not a root of one of the factors.

### 4.1 Simple Examples of Finite Fields

### The Field with 4 Elements

The smallest fields are those with 2 and 3 elements; is there a field with 4 elements? Assume F is such a field; then F contains a neutral element of addition, which we denote by 0, and a neutral element of multiplication, which we denote by 1. Let us denote the remaining two elements by x and y; we need to make  $F = \{0, 1, x, y\}$  into an additive group, and  $\{1, x, y\}$  into a multiplicative group, and make sure that distributivity holds. Experimenting a little bit with the different possibilities eventually produces the following addition and multiplication tables:

+	0	1	x	y	·	0	1	x	y
0	0	1	x	y	0	0	0	0	0
1	1	0	y	x	1	0	1	x	y
x	x	y	0	1	x	0	x	y	1
y	y	x	1	0	y	0	y	1	x

Associativity and distributivity can now be checked by hand, and we have found a field of cardinality 4. Note that  $\mathbb{F}_4 \neq \mathbb{Z}/4\mathbb{Z}$ : in the ring  $\mathbb{Z}/4\mathbb{Z}$ , we have  $1 + 1 = 2 \neq 0$  whereas 1 + 1 = 0 in  $\mathbb{F}_4$ . Moreover,  $\mathbb{F}_4$  has  $\mathbb{F}_2 = \{0, 1\}$  as a subfield, whereas  $\{0, 1\}$  is not even a subring of  $\mathbb{Z}/4\mathbb{Z}$ .

### **Quadratic Extensions**

Recall that a field is a commutative ring in which every nonzero element has an inverse. A field with only finitely many elements is called a finite field. The only finite fields we have seen so far are the fields  $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$  for primes p. In this section, we shall meet a few more.

A basic method for constructing fields is the following:

**Proposition 4.3.** Let K be a field of characteristic  $\neq 2$  and assume  $x \in K$  is a nonsquare. Define addition and multiplication on the set  $L = K \times K = \{(a,b) : a, b \in K\}$  by

$$(a,b) + (c,d) = (a+c,b+d); \quad (a,b) \cdot (c,d) = (ac+bdx,ad+bc).$$

Then L is a field, and  $\iota : K \hookrightarrow L : a \longmapsto (a,0)$  is an injective ring homomorphism, that is, the set  $\{(a,0) : a \in K\}$  is a subfield of L isomorphic to K.

*Proof.* (L, +) is clearly an additive group with neutral element 0 = (0, 0). Moreover,  $L \setminus \{0\}$  is a multiplicative group with neutral element 1 = (1, 0): given any  $(a, b) \neq 0$ , we claim that its inverse is given by (c, d) with  $c = \frac{a}{a^2 - xb^2}$ and  $d = -\frac{b}{a^2 - xb^2}$ . In order for these expressions to make sense, we have to show that  $a^2 - xb^2 \neq 0$ . So assume that  $a^2 - xb^2 = 0$ ; if b = 0, then  $a^2 = 0$ and hence a = 0 (since K is a field): contradiction, since then (a, b) = 0. Thus  $b \neq 0$ , hence  $x = (a/b)^2$  is a square in K contradicting our assumption.

Finally, we have to check that the given element really is the desired inverse:  $(a, b)(\frac{a}{a^2-xb^2}, \frac{-b}{a^2-xb^2}) = 1$ . Proving that the other field axioms hold is easy, as is checking that  $\iota$  is

Proving that the other field axioms hold is easy, as is checking that  $\iota$  is an injective ring homomorphism.

From now on, the element (a, b) in the field L just constructed will be denoted by  $a + b\sqrt{x}$ , and we write  $L = K(\sqrt{x})$ .

Observe that the field  $\mathbb{F}_2$  does not contain any nonsquare. Yet we can construct a quadratic extension of  $\mathbb{F}_2$  by observing that  $y^2 + y = 1$  does not have a root in  $\mathbb{F}_2$ :

**Proposition 4.4.** Let K be a field of characteristic 2, and assume that there is an  $x \in K^{\times}$  such that the polynomial  $y^2 + y - x \in K[y]$  does not have any roots in K. Define addition and multiplication on  $L = K \times K$  by

 $(a,b) + (c,d) = (a+c,b+d), \quad (a,b) \cdot (c,d) = (ac+bdx, ad+bc+bd).$ 

Then L is a field, and  $\iota: K \hookrightarrow L: a \longmapsto (a, 0)$  is an injective ring homomorphism.

Before we prove this result, let us explain how we arrived at these formulas. The idea is to think of L as the set of elements a + by with  $y^2 + y = x$ . Then  $(a+by)(c+dy) = ac + (ad+bc)y + bdy^2 = (ac+bdx) + (ad+bc-bd)y$ , and since -1 = +1 in K, this explains our definitions. Proof. Clearly, 0 = (0, 0) and 1 = (1, 0) are the neutral elements with respect to addition and multiplication. Moreover, -(a, b) = (-a, -b) = (a, b), so every element is its own additive inverse. Let us now prove that  $(a, b) \neq (0, 0)$ has a multiplicative inverse. We want 1 = (1, 0) = (a, b)(c, d) = (ac+bdx, ad+<math>bc+bd), that is, 1 = ac+bdx and 0 = ad+bc+bd. This is a system of linear equations with unknowns c and d; its determinant is  $D = a^2 + ab + b^2x$ . If b = 0, then  $a \neq 0$  since  $(a, b) \neq (0, 0)$ , hence  $D \neq 0$ ; if  $b \neq 0$ , putting  $y = ab^{-1} \in K$  gives  $D = b^2(y^2 + y + x)$ , which is  $\neq 0$  since the bracket has no roots  $y \in K$ . Thus  $D \neq 0$ , hence the system has a unique solution (c, d). Checking commutativity and associativity is simple, though tedious.

Now assume that  $K = \mathbb{Z}/p\mathbb{Z}$ ; if  $x \in K$  is a quadratic nonresidue, then the proposition above tells us that  $L = K(\sqrt{x})$  is a field. Since L has exactly  $p^2$  elements, L is a finite field with  $p^2$  elements; in the mathematical literature, finite fields with q elements are denoted by  $\mathbb{F}_q$ .

How many fields with  $p^2$  elements are there? At first sight, we have constructed  $\frac{p-1}{2}$  such fields above, since this is the number of quadratic nonresidues modulo p. It turns out, however, that these fields are isomorphic: if x and y are quadratic nonresidues, then  $y = xz^2$  for some nonzero  $z \in \mathbb{Z}/p\mathbb{Z}$ ; but then the element  $a + b\sqrt{y} \in K(\sqrt{y})$  is nothing but  $a + bz\sqrt{x} \in K(\sqrt{x})$ . As a matter of fact, we will see below that for every prime power  $q = p^n$ there exists exactly one finite field with q elements.

# 4.2 Construction of Finite Fields

Let F be a finite field, and let p denote its characteristic. The subfield of F generated by 1 is the finite field with p elements, and so we can view F as a vector space over  $\mathbb{F}_p$ . Let  $n = \dim F$  denote its dimension. Then every element of F has a unique representation of the form (2.1), where  $\lambda_1, \ldots, \lambda_n$  run through  $\mathbb{F}_p$ . This implies that F has exactly  $p^n$  elements.

**Proposition 4.5.** If F is a finite field of characteristic p, then F has exactly  $p^n$  elements, where  $n \ge 1$  is some natural number.

We now prove the converse:

**Theorem 4.6.** For every prime power  $q = p^n$ , there is (up to isomorphism) exactly one finite field with q elements.

Before we can do so, let us recall some algebra.

#### Ideals

**Exercise 4.1.** Check that R[X] is a ring if R is a ring.

4. Finite Fields

The most important structure associated with rings is probably that of an ideal. An ideal I in a ring R is a subring of R with the additional property that RI = I, i.e., that  $ri \in I$  for all  $r \in R$  and  $i \in I$ .

**Exercise 4.2.** Let  $R = \mathbb{Z}$ ; then any subring S of R has the form  $S = m\mathbb{Z}$  (integral multiples of m) for  $m \in \mathbb{Z}$ . Prove this, and show that all these subrings are ideals.

**Exercise 4.3.** Show that  $\mathbb{Z}$  is a subring of  $\mathbb{Q}$ , but not an ideal in  $\mathbb{Q}$ .

As for vector spaces, we define kernel and image of ring homomorphisms  $f: R \longrightarrow S$  by

$$\ker f = \{r \in R : f(u) = 0\} \subseteq R \text{ and}$$
$$\inf f = \{f(r) : r \in R\} \subseteq S.$$

**Exercise 4.4.** If  $f : R \longrightarrow S$  is a ring homomorphism, then ker f is an ideal in R.

**Exercise 4.5.** Show that im f is always a subring, but not always an ideal of S for ring homomorphisms  $f : R \longrightarrow S$ . (Hint: you don't know that many subrings that aren't ideals).

The reason why ideals are so important is because they allow us to define quotient structures. If R is a ring and I an ideal in R, then we can form the quotient ring R/I. Its elements are the cosets  $r+I = \{r+i : i \in I\}$  for  $r \in R$ , and we have r + I = r' + I if  $r - r' \in I$ . Addition and multiplication in R/I are defined by (r + I) + (r' + I) = (r + r') + I and (r + I)(r' + I) = rr' + I, and these are well defined since I is an ideal (and not just a subring).

**Exercise 4.6.** Given any ring and an element  $a \in R$ , the set  $(a) = \{ra : r \in R\}$  is an ideal in R.

Ideals of the form (a) are called principal ideals. Not every ideal is principal: in the ring  $\mathbb{Z}[X]$ , the ideal (p, X) (the set of all  $\mathbb{Z}$ -linear combinations of p and X) is an ideal, but cannot be generated by a single element.

Proof of Theorem 4.6. We will only prove the existence, not the uniqueness of the field with  $p^n$  elements. All we have to do is show that for every prime p and every  $n \ge 1$ , there is an irreducible polynomial f of degree n in  $\mathbb{F}_p[X]$ , because then  $\mathbb{F}_p[X]/(f)$  is a field with  $p^n$  elements. In fact, since f is irreducible and  $\mathbb{F}_p[X]$  is a PID, the ideal (f) is maximal, hence  $\mathbb{F}_p[X]/(f)$  is a field.

For those who want to check the existence of multiplication of inverses by hand (this is the only field axiom whose verification is not straight forward): assume that  $r+(f) \neq 0+(f)$ . Let  $d = \gcd(r, f)$  (note that d is a polynomial); since  $d \mid f$  and f is irreducible, we conclude that d = f or d = 1 (up to unit factors, that is, constants). But d = f is impossible since  $d \mid r$  and  $\deg r <$  $\deg f$ . Thus d = 1, and by Bezout there are polynomials  $u, v \in \mathbb{F}_p[X]$  such that 1 = ur + vf. But then 1 + (f) = ur + (f), hence (r + (f))(u + (f)) = 1 + (f), and u + (f) is the multiplicative inverse of r + (f).

Moreover,  $\mathbb{F}_p[X]/(f)$  has exactly  $p^n$  elements because  $\mathbb{F}_p[X]/(f) = \{a_0 + a_1X + \ldots + a_{n-1}X^{n-1} : a_i \in \mathbb{F}_p\}$ ; given any  $g \in \mathbb{F}_p[X]$ , write g = qf + r with deg r < n = deg f; then g + (f) = r + (f), so every element in  $\mathbb{F}_p[X]/(f)$  can be represented as r + (f) for some  $r \in \mathbb{F}_p[X]$  with deg r < n. Moreover, this representation is unique, for if  $g_1 + (f) = g_2 + (f)$  for polynomials  $g_1, g_2 \in \mathbb{F}_p[X]$  with deg  $g_1$ , deg  $g_2 < n$ , then  $g_1 - g_2 \in (f)$ , that is, the polynomial  $g_1 - g_2$ .

It remains to show that, for every  $n \ge 1$ , there exist irreducible polynomials  $f \in \mathbb{F}_p[X]$  of degree n. This is obvious for n = 1: any polynomial X - a with  $a \in \mathbb{F}_p$  is irreducible.

Before giving the general case, let us study n = 2: there are exactly  $p^2$  monic polynomials  $X^2 + aX + b$  with  $a, b \in \mathbb{F}_p$ . How many of the are reducible? Exactly those that can be written in the form (X - r)(X - s) for  $r, s \in \mathbb{F}_p$ . There are p of these with r = s; if  $r \neq s$ , then (X - r)(X - s) = (X - s)(X - r), so the  $p^2 - p$  pairs (r, s) with  $r \neq s$  give  $\frac{1}{2}(p^2 - p)$  different reducible polynomials. Thus there are  $p^2 - p - \frac{1}{2}(p^2 - p) = \frac{1}{2}(p^2 - p)$  monic irreducible quadratic polynomials in  $\mathbb{F}_p[X]$ ; since  $p^2 - p > 0$  for all primes p, we have shown that there is at least one, which is good enough for our purposes.

The general case uses the same idea: you count the number of all polynomials as well as polynomials having at least two factors, and then show that there must be some irreducible polynomials. Providing the details requires some effort, though.  $\hfill \Box$ 

**Exercise 4.7.** Show that the polynomial  $x^2 + x + 1$  is irreducible in  $\mathbb{F}_2[x]$ . Use it to construct the field  $\mathbb{F}_4$  as  $\mathbb{F}_4 \simeq \mathbb{F}_2[x]/(x^2 + x + 1)$ ; determine addition and multiplication tables. Compute  $\operatorname{Frob}(x)$ , where  $\operatorname{Frob}: x \longrightarrow x^2$  denotes the Frobenius automorphism.

**Exercise 4.8.** Construct the field  $\mathbb{F}_9$ , and determine addition and multiplication tables.

**Exercise 4.9.** Find all irreducible quadratic polynomials in  $\mathbb{F}_p[X]$  for p = 3, 5, 7.

# 4.3 Galois Theory of Finite Fields

Let F be a finite field of characteristic p; then  $F = \mathbb{F}_q$  with  $q = p^n$  for some  $n \ge 1$ . In any such field, the map

$$\phi: x \longmapsto x^p \tag{4.1}$$

is a ring homomorphism: in fact,  $\phi(x+y) = (x+y)^p = x^p + y^p = \phi(x) + \phi(y)$ since the binomial coefficients  $\binom{p}{k} = 0$  for  $k = 1, \ldots, p-1$ . Moreover,  $\phi(xy) = \phi(x)\phi(y)$  and  $\phi(1) = 1$ .

#### 4. Finite Fields

**Binomial Expansion** We have  $(x+y)^n = (x+y)(x+y)\cdots(x+y)$ , and you multiply by choosing x or y from each bracket and adding up these products. How often do you get the product  $x^{n-i}y^i$ ? As often as you can choose i times the y; this can be done in  $\binom{n}{i}$  times, so

$$(x+y)^n = x^n + \binom{n}{1}x^{n-1}y + \ldots + \binom{n}{n-1}xy^{n-1} + y^n$$

It is easy to show that  $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ . If n = p is a prime, then the numerator p! is divisible by p, but the denominator is not except when i = 0 or i = p.

Let us study the homomorphism Frob :  $\mathbb{F}_q \longrightarrow \mathbb{F}_q$  in more detail. First, Frob is injective: in fact,  $x \in \ker$  Frob if and only if  $x^p = 0$ , which holds if and only if x = 0. Since injective maps between finite sets of the same cardinality are bijective, Frob is an isomorphism. We shall call (4.1) the Frobenius automorphism of  $\mathbb{F}_q/\mathbb{F}_p$ .

### **Lemma 4.7.** The Frobenius automorphism of $\mathbb{F}_q/\mathbb{F}_p$ acts trivially on $\mathbb{F}_p$ .

*Proof.* If  $x \in \mathbb{F}_p^{\times}$ , then  $x^{p-1} = 1$  by Lagrange's Theorem, hence  $\operatorname{Frob}(x) = x^p = x$ . Since  $\operatorname{Frob}(0) = 0$ , the Frobenius automorphism fixes every element of  $\mathbb{F}_p$ .

Let us recall Lagrange's theorem (essentially Fermat's little theorem for arbitrary finite abelian groups):

**Theorem 4.8.** If G is a finite (multiplicative) abelian group with n elements, then  $g^n = 1$  for every  $g \in G$ .

*Proof.* We mimic the proof of Fermat's little theorem: write  $G = \{g_1, \ldots, g_n\}$ and multiply each  $g_i$  by g; write  $g_i \cdot g = h_i$  for some  $h_i \in G$ . We claim that  $G = \{h_1, \ldots, h_n\}$ . To prove this it is sufficient to show that the  $h_i$  are pairwise different. But  $h_i = h_j$  implies  $g_i g = g_j g$ ; since we can multiply through by the inverse  $g^{-1}$  of g we get  $g_i = g_j$ , which implies i = j.

Thus the  $h_i$  are just a permutation of the  $g_i$ , hence  $\prod h_i = \prod g_j$ . Now multiplying the equations

$$g_1 \cdot g = h_1$$
$$g_2 \cdot g = h_2$$
$$\vdots$$
$$g_n \cdot g = h_n$$

we get  $(\prod g_i)g^n = \prod h_i$ , and cancelling  $\prod h_i = \prod g_j$  gives  $g^n = 1$ .

If F is a finite field with  $q = p^n$  elements, then  $G = F^{\times} = F \setminus \{0\}$  is a multiplicative abelian group with  $p^n - 1$  elements, so by Lagrange's theorem we have  $a^{p^n-1} = 1$  for all  $a \in F^{\times}$ , and therefore  $a^q = a^{p^n} = a$  for all  $a \in F$ .

This means that the  $a \in F$  are roots of the polynomial  $f(X) = X^q - X$ ; since f can have at most  $q = \deg f$  roots over any field, we conclude that  $X^q - X = \prod_{a \in F} (X - a).$ 

**Proposition 4.9.** The roots of the polynomial  $X^q - X$ , where  $q = p^n$  is a prime power, are exactly the q elements of the finite field  $F = \mathbb{F}_q$ .

In particular,  $X^p - X = \prod_{a \in F_p} (X - a)$ , or  $X^{p-1} - 1 = \prod_{a \in F_p^{\times}} (X - a)$ . Comparing the constant terms shows that -1 = (p - 1)!: Wilson's theorem.

We now define the Galois group  $\operatorname{Gal}(E/F)$  of a field extension E/F as the group of all automorphisms of E that leave F elementwise fixed. This is in fact a group: if  $\sigma$  and  $\tau$  are such automorphisms, then so is  $\tau \circ \sigma$ , and the identity map is an automorphism. Since  $\sigma$  is an automorphism, so is its inverse map  $\sigma^{-1}$ ; this is also a homomorphism: if  $x = \sigma(x')$  and  $y = \sigma(y')$ , then

$$\begin{aligned} \sigma^{-1}(x+y) &= \sigma^{-1}(\sigma(x')+\sigma(y')) &= \sigma^{-1}(\sigma(x'+y')) \\ &= x'+y' = \sigma^{-1}(x) + \sigma^{-1}(y). \end{aligned}$$

Thus the Galois group of E/F is a group.

**Example.** Let  $E = \mathbb{C}$  and  $F = \mathbb{R}$ ; any automorphism  $\phi$  maps a + bi to  $\phi(a + bi) = \phi(a) + \phi(b)\phi(i) = a + b\phi(i)$  because  $\phi$  fixes the base field  $\mathbb{R}$ . Moreover,  $\phi(i)^2 = \phi(i^2) = \phi(-1) = -1$ , so  $\phi(i) = i$  or  $\phi(i) = -i$ . In the first case,  $\phi$  is the identity, in the second case it is complex conjugation  $a + bi \mapsto a - bi$ . Thus  $\operatorname{Gal}(\mathbb{C}/\mathbb{R})$  has  $(\mathbb{C}:\mathbb{R}) = 2$  elements.

**Example.** It can be shown that  $\operatorname{Gal}(\mathbb{Q}(\sqrt[3]{2})/\mathbb{Q})$  consists only of the identity element: any automorphism  $\phi$  maps  $a + b\omega + c\omega^2$  (with  $\omega = \sqrt[3]{2}$ ) to  $a + b\phi(\omega) + c\phi(\omega)^2$ , so all we have to figure out is the possible images  $\phi(\omega)$ . But clearly  $\phi(\omega)^3 = \phi(\omega^3) = \phi(2) = 2$ , hence  $\phi(\omega) = \omega$ ,  $\rho\omega$ , or  $\rho^2\omega$ , where  $\rho = \frac{1}{2}(-1 + \sqrt{-3})$  is a cube root of unity (we have  $\rho^3 = 1$ ; check it if you haven't seen this before). However, it can be shown that  $\rho\omega$  is not contained in the field  $\mathbb{Q}(\sqrt[3]{2})$ , so our only choice is  $\phi(\omega) = \omega$ , and  $\phi$  is the identity.

We now determine the structure of Galois groups in the special case of extensions  $\mathbb{F}_q/\mathbb{F}_p$  of finite fields. Before we do so, let's have another look at the example  $\mathbb{C}/\mathbb{R}$ :  $\mathbb{C}$  is generated by a root i of  $x^2+1$ , and the automorphisms of  $\mathbb{C}/\mathbb{R}$  mapped i to i or -i, in other words: they permuted the roots of  $x^2+1$ . We now generalize this fact:

**Lemma 4.10.** Let F be any field,  $f \in F[X]$  an irreducible polynomial, and E = F[X]/(f). Then any  $\chi \in \text{Gal}(E/F)$  permutes the roots of f.

*Proof.* First of all, f does have roots in E: in fact, evaluating f at the element  $x = X + (f) \in E$  gives f(x) = f(X) + (f) = 0 + (f).

Next, if  $f(\gamma) = 0$ , then writing  $f(X) = a_0 + a_1 X + \ldots + a_n X^n$  gives  $f(\chi(\gamma)) = \chi(f(\gamma))$  since

#### 4. Finite Fields

$$f(\chi(\gamma)) = a_0 + a_1 \chi(\gamma) + a_2 \chi(\gamma)^2 + \dots + a_n \chi(\gamma)^n$$
  
=  $a_0 + a_1 \chi(\gamma) + a_2 \chi(\gamma^2) + \dots + a_n \chi(\gamma^n)$   
=  $\chi(a_0 + a_1 \gamma + a_2 \gamma^2 + \dots + a_n \gamma^n)$   
=  $\chi(f(\gamma)),$ 

where we have used that  $\chi$  acts trivially on the coefficients of f (which are elements of F). But then  $f(\chi(\gamma) = \chi(f(\gamma)) = \chi(0) = 0$ , so  $\chi(\gamma)$  is also a root of f.

Now we claim:

### **Proposition 4.11.** We have $\operatorname{Gal}(\mathbb{F}_q/\mathbb{F}_p) \simeq \mathbb{Z}/n\mathbb{Z}$ with $n = (\mathbb{F}_q : \mathbb{F}_p)$ .

*Proof.* Let Frob :  $x \mapsto x^p$  denote the Frobenius automorphism. We first claim that Frob<sup>n</sup> is the identity map. In fact,  $\operatorname{Frob}^n(x) = \operatorname{Frob}^{n-1}(\operatorname{Frob}(x)) = \dots = x^{p^n} = x^q = 1$  by Lagrange's theorem.

Next id = Frob<sup>0</sup>, Frob, ..., Frob<sup>n-1</sup> are automorphisms; we claim that they are pairwise different. Assume that Frob<sup>r</sup> = Frob<sup>s</sup> for r < s < n. With m = s - r we find id = Frob<sup>r+n-r</sup> = Frob<sup>s+n-r</sup> = Frob<sup>m</sup>; observe that 0 < m < n. Now Frob<sup>m</sup> = id implies that  $x^{p^m} = x$  for all  $x \in \mathbb{F}_q$ ; but then  $x^{p^m} - x$  has  $q = p^n$  roots, contradicting the fact that polynomials f over fields have at most deg f roots.

Thus Frob generates a (cyclic) subgroup of order n in  $\operatorname{Gal}(\mathbb{F}_q/\mathbb{F}_p)$ . It remains to show that  $\#\operatorname{Gal}(\mathbb{F}_q/\mathbb{F}_p) \leq n$ .

To this end, write  $\mathbb{F}_q = \mathbb{F}_p[X]/(f)$  for some irreducible polynomial  $f \in \mathbb{F}_p[X]$  with deg f = n. Let  $\chi \in \operatorname{Gal}(\mathbb{F}_q/\mathbb{F}_p)$ .

Now let  $\gamma = X + (f)$ ; then  $\gamma$  is a root of f, and so are  $\operatorname{Frob}(\gamma)$ ,  $\operatorname{Frob}^2(\gamma), \ldots, \operatorname{Frob}^{n-1}(\gamma)$ . We claim that these roots are different. If not, then  $\operatorname{Frob}^k(\gamma) = \gamma$  for some k < n. Form the polynomial  $g(X) = (X - \gamma)(X - \operatorname{Frob}(\gamma)) \cdots (X - \operatorname{Frob}^{k-1}(\gamma))$ . Clearly g is fixed by  $\operatorname{Frob}$ , hence  $g(X) \in \mathbb{F}_p[X]$ . Moreover, all the roots of g are roots of f, so  $g \mid f$ : but since f is irreducible, this implies g = f, hence k = n.

Now since  $\chi$  maps roots to roots, there is some  $k \leq n$  such that  $\chi(\gamma) = \operatorname{Frob}^k(\gamma)$ ; we claim that  $\chi = \operatorname{Frob}^k$ . To this end, it is sufficient to show that any automorphism is uniquely determined by its action on  $\gamma$  (remember the example  $E = \mathbb{C}$ , where Frob was determined by its action on i).

This will follow if we can prove that  $\gamma$  generates the whole field: if we know  $\operatorname{Frob}(\gamma)$ , we know  $\operatorname{Frob}(a_0 + a_1\gamma + a_2\gamma^2 + \ldots)$ , so it is sufficient to show that any field element is a linear combination of the powers of  $\gamma$ . But this is clear, since  $\gamma = X + (f)$ , and arbitrary field elements have the form  $a_0 + a_1X + a_2X^2 + \ldots + (f) = a_0 + a_1\gamma + a_2\gamma_2 + \ldots$ 

The main theorem of Galois theory gives an inclusion-reversing bijection between the subextensions of  $\mathbb{F}_q/\mathbb{F}_p$  and the subgroups of  $\operatorname{Gal}(\mathbb{F}_q/\mathbb{F}_p)$ . It allows to replace questions concerning subfields (usually difficult) by questions involving groups (these have a much simpler structure than fields). **Theorem 4.12.** There is a bijection between subfields of  $\mathbb{F}_q/\mathbb{F}_p$  and subgroups of  $\operatorname{Gal}(\mathbb{F}_q/\mathbb{F}_p)$ . The subfield E corresponds to the subgroup  $\operatorname{Gal}(\mathbb{F}_q/E)$ of all automorphisms fixing E elementwise.

## 4.4 The Quadratic Reciprocity Law

Let p be an odd prime; an integer  $a \in \mathbb{Z}$  not divisible by p is called a quadratic residue modulo p if and only if the element  $a \cdot 1 \in \mathbb{F}_p^{\times}$  is a square. The nonzero squares in  $\mathbb{F}_7$  are 1, 2, 4, whereas 3, 5, 6 are nonsquares. Legendre introduced a symbol for  $a \in \mathbb{F}_p$  by defining

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a = x^2 \text{ for some } x \in \mathbb{F}_p^{\times}, \\ 0 & \text{if } a = 0, \\ -1 & \text{if } a \neq x^2 \text{ for any } x \in \mathbb{F}_p. \end{cases}$$

Let q be an odd prime different from p. The last thing you would expect is that there is a connection between squares modulo p and squares modulo q; the quadratic reciprocity law claims that there is such a connection, and a rather simple one at that:

**Theorem 4.13.** Let p and q be distinct odd primes. Then

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}.$$

There are many elementary proofs based on Gauss's Lemma; these proofs offer no insight at all into *why* the reciprocity law is true. The following proof based on Gauss's fifth proof (he published six of them and two more were discovered in his papers after his death) uses Gauss sums and shows that the reciprocity law is essentially a consequence of the action of the Frobenius on Gauss sums.

We will also need Euler's criterion saying that

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}}$$

for  $a \in \mathbb{F}_p$ .

For defining a Gauss sum we fix an odd prime p and a finite field  $\mathbb{F}_p$ , as well as an odd prime  $q \neq p$ . Then  $q^{p-1} \equiv 1 \mod p$  by Fermat's Little Theorem, hence implies that  $F = \mathbb{F}_{q^{p-1}}$  contains a primitive p-th root of unity  $\alpha$ : in fact, Since  $F^{\times}$  is cyclic, there is a generator  $\gamma \in \mathbb{F}^{\times}$ , and  $\alpha = \gamma^{(q^{p-1}-1)/p}$  has the properties

- $\alpha \in F$  (obviously)
- $\alpha^p = 1, \alpha \neq 1$  (check!).

### 4. Finite Fields

Now we define the Gauss sum

$$\tau = \sum_{a=1}^{p-1} \left(\frac{a}{p}\right) \alpha^a. \tag{4.2}$$

By definition it is an element of the field F of characteristic q.

**Example.** Take p = 3 and let q = 7; then  $q \equiv 1 \mod 3$  implies that already  $\mathbb{F}_7$  contains a cube root of 1, namely  $\alpha = 2$ . Thus  $\tau \in \mathbb{F}_7$ , and we find  $\tau = (\frac{1}{3}) \cdot 2 + (\frac{2}{3}) \cdot 2^2 = 2 - 4 = -2$ , as well as  $\tau^2 = 4 = -3$ .

Now let us look at q = 5; here  $\mathbb{F}_5$  does not contain a cube root of unity (check that if  $a^3 \equiv 1 \mod 5$  then  $a \equiv 1 \mod 5$ ), and we have to consider the quadratic extension field  $\mathbb{F}_{25} = \mathbb{F}_5(\sqrt{-3}) = \mathbb{F}_5[X]/(X^2+3)$ . Write  $\alpha = 3X - 3+(f)$  with  $f = X^2+3$ ; then  $\alpha^2 = 9X^2 - 18X + 9 + (f) = -X^2 + 2X - 1 + (f) = 2X + 2 + (f)$ , and  $\alpha^3 = (3X - 3)(2X + 2) + (f) = X^2 - 1 + (f) = 1 + (f)$ . Thus  $\tau = (\frac{1}{3}) \cdot \alpha + (\frac{2}{3}) \cdot \alpha^2 = \alpha - \alpha^2 = 3X - 3 - (2X + 2) + (f) = X + (f)$ and  $\tau^2 = X^2 + (f) = -3 + (f)$ .

In particular,  $\tau$  does not always live in the base field. In order to find out how far away from  $\mathbb{F}_p$  the Gauss sum is, we compute the action of the Frobenius in  $\mathbb{F}_q$  on  $\tau$ . We find

$$\tau^{q} = \left(\sum_{a=1}^{p-1} \left(\frac{a}{p}\right) \alpha^{a}\right)^{q} = \sum_{a=1}^{p-1} \left(\frac{a}{p}\right)^{q} \alpha^{aq} = \sum_{a=1}^{p-1} \left(\frac{a}{p}\right) \alpha^{aq}$$
$$= \left(\frac{q}{p}\right) \sum_{a=1}^{p-1} \left(\frac{aq}{p}\right) \alpha^{aq} = \left(\frac{q}{p}\right) \sum_{b=1}^{p-1} \left(\frac{b}{p}\right) \alpha^{b} = \left(\frac{q}{p}\right) \tau.$$

Here we have used that b = aq runs through  $\mathbb{F}_p^{\times}$  if a does, and we have proved **Proposition 4.14.** We have  $\operatorname{Frob}(\tau) = (\frac{q}{p})\tau$ .

Thus we always have  $\operatorname{Frob}(\tau) = \pm \tau$ , which implies that  $\operatorname{Frob}(\tau^2) = \tau$ : thus  $\tau^2$  is always an element of the base field! In fact we always have  $\tau^2 = p^*$ , where  $p^* = \pm p$ , or more exactly  $p^* = (-1)^{(p-1)/2}p$ . In particular, if  $p^*$  is a square in  $\mathbb{F}_q$ , then  $\tau$  is an element of the base field  $\mathbb{F}_q$ , and if  $p^*$  is a nonsquare,  $\tau$  lives in the field with  $q^2$  elements.

**Proposition 4.15.** The Gauss sum  $\tau$  in (4.2) satisfies  $\tau^2 = p^*$ , where  $p^* = (-1)^{(p-1)/2}p$ .

*Proof.* We will have to compute  $\tau^2$ , so let's do it.

$$\tau^{2} = \left(\sum_{a=1}^{p-1} \left(\frac{a}{p}\right) \alpha^{a}\right) \left(\sum_{b=1}^{p-1} \left(\frac{b}{p}\right) \alpha^{b}\right)$$
$$= \sum_{a=1}^{p-1} \sum_{b=1}^{p-1} \left(\frac{ab}{p}\right) \alpha^{a+b} = \sum_{a=1}^{p-1} \sum_{c=1}^{p-1} \left(\frac{c}{p}\right) \alpha^{a+ac},$$

where we have made the substitution c = ab; note that, for fixed  $a \neq 0$ , the product c = ab runs through  $\mathbb{F}_p^{\times}$  if b does, and that  $(a^2/p) = +1$  since  $a^2$  is a nonzero square. The reason for this substitution becomes clear immediately:

$$\tau^{2} = \sum_{a=1}^{p-1} \sum_{c=1}^{p-1} \left(\frac{c}{p}\right) (\alpha^{1+c})^{a} = \sum_{c=1}^{p-1} \left(\frac{c}{p}\right) \sum_{a=1}^{p-1} (\alpha^{1+c})^{a}.$$

Now let us have a good look at the last sum. Putting

$$S = \sum_{a=0}^{p-1} (\alpha^{1+c})^a$$

(note the additional term for a = 0) we have

$$\alpha^{1+c} \cdot S = \sum_{a=0}^{p-1} (\alpha^{1+c})^{a+1} = \sum_{b=0}^{p-1} (\alpha^{1+c})^b = S$$

by the simple substitution b = a + 1. This implies S = 0 whenever  $\alpha^{1+c} \neq 1$ , that is, whenever  $c \neq -1$ . If c = -1, on the other hand, then

$$S = \sum_{a=0}^{p-1} (\alpha^{1+c})^a = \sum_{a=0}^{p-1} 1 = p.$$

This implies that

$$\sum_{a=1}^{p-1} (\alpha^{1+c})^a = \begin{cases} p-1 & \text{if } c = -1, \\ -1 & \text{if } c \neq -1. \end{cases}$$

Thus

$$\tau^{2} = \left(\frac{-1}{p}\right)(p-1) - \sum_{c=1}^{p-2} \left(\frac{c}{p}\right) = \left(\frac{-1}{p}\right)p = p^{*}.$$

Here we have used that the sum  $\sum_{c=1}^{p-1} \left(\frac{c}{p}\right) = 0$  since there are as many quadratic residues as there are nonresidues. A different argument reminiscent of what we did above for S is the following: put  $T = \sum_{c=1}^{p-1} \left(\frac{c}{p}\right)$ ; let g denote a quadratic nonresidue modulo p. Then  $-T = \left(\frac{g}{p}\right)T = \sum_{c=1}^{p-1} \left(\frac{gc}{p}\right) = \sum_{d=1}^{p-1} \left(\frac{d}{p}\right) = T$  since d = gc runs through  $\mathbb{F}_p^{\times}$  as c does. Thus 2T = 0, hence T = 0 since we are working over a field with odd characteristic.

The quadratic reciprocity law now follows from computing the action of the Frobenius in  $\mathbb{F}_q$  on the Gauss sum  $\tau$  in another way:

Frob
$$(\tau) = \tau^q = \tau \tau^{q-1} = \tau (p^*)^{\frac{q-1}{2}} = \left(\frac{p^*}{q}\right) \tau.$$

### 4. Finite Fields

**Proposition 4.16.** We have  $\operatorname{Frob}(\tau) = \left(\frac{p^*}{q}\right)\tau$ .

Comparing Propositions 4.14 and 4.16 (note that  $\tau \neq 0$  since  $\tau^2 = p^* \neq 0$  in  $\mathbb{F}_q$ ) shows that

$$\left(\frac{p^*}{q}\right) = \left(\frac{q}{p}\right)$$

which immediately implies the quadratic reciprocity law.

**Exercise 4.10.** We have  $\mathbb{F}_9 = \mathbb{F}_3[i]$  with  $i^2 = -1(=2)$ ; determine how the Frobenius of  $\mathbb{F}_9/\mathbb{F}_3$  acts on 1+i and 1-i and check that  $f(X) = (X - (1+i))(X - (1-i)) \in \mathbb{F}_3[X]$ .

Error-Correcting Codes

# 5. Cyclic Linear Codes

We've come a long way: starting with codes as subsets C of  $\mathbb{F}_q^n$ , we added structure by considering only linear codes, that is, we demanded that C be a vector space over  $\mathbb{F}_q$ . This restricted the number of codes, but it simplified working with them: all we need is a generator and a parity check matrix instead of a list of all codewords.

It might seem quite surprising that we can impose even stronger conditions on codes and still get something interesting. In fact, the class of cyclic codes that we are going to study includes the family of Hamming codes; more exactly, Hamming codes – while not being cyclic except for the [7, 4]code – are equivalent to cyclic codes, that is, become cyclic upon a certain permutation of the coordinates.

So what are cyclic codes?

# 5.1 Cyclic Codes

**Definition 5.1.** A code  $C \subseteq \mathbb{F}_q^n$  is cyclic if C is linear and if, for every codeword  $(c_0c_1 \dots c_{n-1}) \in C$ , the word  $(c_{n-1}c_0c_1 \dots c_{n-2})$  is also a codeword.

This means that any cyclic permutation of a codeword is a codeword again. Note that the cyclic shift

$$\pi: (c_0c_1\ldots c_{n-1})\longmapsto (c_{n-1}c_0c_1\ldots c_{n-2})$$

is an automorphism of the cyclic code C; since  $\pi^n$  is the identity, cyclic codes  $C \subseteq \mathbb{F}_q^n$  have an automorphism group  $\operatorname{Aut}(C)$  with a cyclic subgroup of order n.

**Example.** The code  $C = \{000, 110, 101, 011\}$  is cyclic.

The order of the automorphism group of Hamming's [7, 4]-code Ham $(3) \subseteq \mathbb{F}_2^7$  is  $168 = 2^3 \cdot 3 \cdot 7$ ; since 168 is divisible by 7, we might suspect that Ham(3) is cyclic. In fact it is:

**Exercise 5.1.** Show that Hamming's [7,4]-code is cyclic.

What we will do now may seem a bit strange at first, but is actually the key to exploiting the additional structure of cyclic codes: we will write codewords not as vectors but as polynomials!

To each codeword  $c = (c_0 c_1 \dots c_{n-1}) \in C \subset \mathbb{F}_q$  we associate the polynomial

$$c(X) = c_0 + c_1 X + c_2 X^2 + \ldots + c_{n-1} X^{n-1} \in \mathbb{F}_q[X]$$
(5.1)

We can do this for any linear code, and adding codewords is the same as adding polynomials: we work coordinatewise anyway. If we multiply a codeword (5.1) by X, then we get

$$c(X)X = c_0 X + c_1 X^2 + \ldots + c_{n-1} X^n$$
(5.2)

Unfortunately, this isn't a codeword, although it looks almost like the cyclic shift of c. Fortunately, all we have to do is make sure that we can write  $c_{n-1}$  instead of  $c_{n-1}X^n$ . Of course we know how to do this: we work modulo  $X^n - 1$ . In fact, if we multiply  $c(X) \in R_n = \mathbb{F}_q[X]/(X^n - 1)$  by X, then we can simplify (5.2) to

$$c(X)X \equiv c_{n-1} + c_0X + \ldots + c_{n-2}X^{n-1}$$

which is the polynomial corresponding to a cyclic permutation of c.

Note that  $R_n$  is a ring, but not a field in general, since the polynomial  $X^n - 1$  is irreducible only if n = 1, and then  $R_1 = \mathbb{F}_q$ . On the other hand,  $R_n$  has more structure than just the ring structure: it is, as we shall see, an  $\mathbb{F}_q$ -vector space. Thus every subspace C of  $R_n$  is a linear code, and we will see that such a C is cyclic if, in addition, C is an ideal!

$\mathbb{F}_q^n$	$\mathbb{F}_q[X]/(X^n-1)$
linear subspace	linear subspace
cyclic shift	multiplication by $X$
cyclic code	ideal

# 5.2 The Algebra behind Cyclic Codes

One important tool in studying cyclic codes is the following proposition which realizes the vector space  $\mathbb{F}_q^n$  as a quotient of the ring of polynomials over  $\mathbb{F}_q$ .

**Proposition 5.2.** We have  $\mathbb{F}_q^n \simeq \mathbb{F}_q[X]/(X^n - 1)$  as vector spaces.

Assume now that R contains a field F, and let I be an ideal in R (the ring  $\mathbb{Z}$  does not contain a field, so this is not always satisfied). Let us see why R/I is a vector space in this case. Since R/I is a ring, it certainly is an additive group. Multiplication of scalars is defined by  $\lambda(r+I) = \lambda r + I$ , where  $\lambda \in F \subseteq R$ ; checking that the axioms for a vector space are satisfied is easy.

That finally explains why both  $\mathbb{F}_q^n$  and  $\mathbb{F}_q[X]/(X^n-1)$  are both  $\mathbb{F}_q$ -vector spaces: the ring  $\mathbb{F}_q[X]$  contains the field  $\mathbb{F}_q$ .
Proof of Prop. 5.2. We claim that the map

$$\phi: (a_0, a_1, \dots, a_{n-1}) \longmapsto a_0 + a_1 X + \dots + a_{n-1} X^{n-1} + (X^n - 1)$$

defines an isomorphism  $\mathbb{F}_q^n \longrightarrow \mathbb{F}_q[X]/(X^n - 1)$  between vector spaces.

Checking that  $\phi$  is a linear map is trivial. We have to prove that ker  $\phi = 0$ and im  $\phi = \mathbb{F}_q[X]/(X^n - 1)$ .

Assume that  $(a_0, \ldots, a_{n-1}) \in \ker \phi$ . Then  $a_0 + a_1 X + \ldots + a_{n-1} X^{n-1} \in (X^n - 1)$ ; but then  $f(X) = a_0 + a_1 X + \ldots + a_{n-1} X^{n-1}$  is a multiple of  $X^n - 1$ , and since deg  $f < \deg(X^n - 1) = n$ , this implies f = 0, i.e.  $(a_0, \ldots, a_{n-1}) = 0$ . Since  $\phi$  is obviously surjective, this completes the proof.  $\Box$ 

Ok, so  $R_n$  is, as a vector space, nothing but  $\mathbb{F}_q^n$ . Why did we bother defining it at all? Here's the catch:

**Proposition 5.3.** Let C be a linear subspace of the  $\mathbb{F}_q$ -vector space  $R_n$ . Then C is a cyclic code if and only if C is an ideal in the ring  $R_n$ .

*Proof.* Assume that C is cyclic and let  $c(X) = c_0 + c_1 X + \ldots + c_{n-1} X^{n-1} \in C$ . Then  $X \cdot c(X) \in C$  since multiplication by X is just a cyclic shift. Induction shows that multiplication by  $X^k$  sends c(X) to codewords. Finally, C is closed under multiplication by scalars  $a_k$ , and collecting everything we see that  $(a_0 + a_1 X + \ldots + a_m X^m)c(X) \in C$ . Thus C is an ideal.

Now assume that C is an ideal; since multiplication by  $X \in R_n$  preserves C, we deduce that C is cyclic.

In every ring R, we have the principal ideals (r) for  $r \in R$ . In our case, every polynomial  $f \in \mathbb{F}_q[X]$  generates an ideal C = (f), hence a cyclic code. Note however, that different f do not necessarily give rise to different codes. In fact, if f and g are polynomials in  $\mathbb{F}_q[X]$ , then it might happen that the ideals (f) and (g) are the same. In the ring  $\mathbb{Z}$ , for example, the ideals (m)and (-m) are equal. More generally:

**Exercise 5.2.** If R is a ring and  $u \in R^{\times}$  is a unit, then (r) = (ru).

Here are some examples. First, put  $R_n = \mathbb{F}_q[X]/(X^n - 1)$ . Then consider the ideal  $I = (X^2 + X + 1)$  in  $R_3$ . What elements does I contain? Of course, every ideal contains 0, and clearly  $X^2 + X + 1 \in I$ . To find other elements, we multiply  $X^2 + X + 1$  by ring elements, that is, polynomials. Constant don't produce anything new, so look at the product  $X(X^2 + X_1)$ ; since we are working modulo  $X^3 - 1$ , we find  $X(X^2 + X_1) = X^3 + X^2 + X = X^2 + X + 1$ . Similarly, multiplication by X + 1 gives  $(X+1)(X^2 + X+1) = 2(X^2 + X + 1)$ , and it can be proved easily by induction that  $f(X^2 + X + 1)$  is always a constant multiple of  $X^2 + X + 1 \mod X^3 - 1$ .

Thus the ideal  $(X^2 + X + 1)$  contains only two elements over  $\mathbb{F}_2$ , namely 0 and  $X^2 + X + 1$ . The corresponding cyclic code in  $\mathbb{F}_2^3$  is {000, 111}.

**Exercise 5.3.** Compute the cyclic code in  $\mathbb{F}_2^3$  generated by  $X + 1 \in R_3$ .

**Exercise 5.4.** Show that the code generated by  $X^2 + X + 1 \in R_4$  is the trivial code  $\mathbb{F}_2^4$ . Conclude that  $X^2 + X + 1$  is a unit in  $R_4$  and determine its multiplicative inverse.

While  $\mathbb{F}_q^n$  is just a boring old vector space, the quotient  $R_n = \mathbb{F}_q[X]/(X^n - 1)$  is actually a genuine ring. Can we determine its ideals?

We can determine all the ideals in  $\mathbb{Z}$ : they are (0), (1), (2), (3), ...; in particular, all ideals in  $\mathbb{Z}$  are principal (generated by single elements), which is why  $\mathbb{Z}$  is called a principal ideal ring.

For a proof, consider an ideal I in  $\mathbb{Z}$ . If I = (0), then we are done; if not, then I contains a nonzero element m. Since I is closed under multiplication by the ring element -1, both m and -m are in I, and in particular I contains a natural number. Let n denote the smallest positive natural number in I; we claim that I = (n).

In fact, assume that  $a \in I$ ; we have to show that a = qn is an integral multiple of n. To this end, write a = qn + r for some integer  $0 \le r < n$ . Then  $a \in I$  and  $n \in I$  imply that  $r = a - qn \in I$  (since I is an ideal), but the minimality of n shows that r = 0, hence a = qn and I = (n).

We now do the very same thing for  $R_n$ ; the analogy between the ring of integers and the ring of polynomials  $\mathbb{F}_q[X]$  suggests that minimality with respect to the absolut value has to be replaced with minimality of the degree, and this does in fact work:

**Theorem 5.4.** Assume that  $p \nmid n$ . Then the nonzero ideals of  $R_n = \mathbb{F}_q[X]/(X^n - 1)$  have the form (g) with  $g(X) \mid (X^n - 1)$ .

*Proof.* We first show that every ideal I in  $R_n$  is principal. If I = (0), this is clear, so assume that  $I \neq (0)$ . Then I contains a polynomial  $\neq 0$ ; let g denote a polynomial in I with minimal degree  $\geq 0$ . We claim that I = (g). In fact, take any  $a \in I$  and write a = qg + r for some polynomial r with deg  $r < \deg g$ . As before,  $r = a - qg \in I$ , and the minimality of deg  $g \geq 0$  implies that r = 0. Thus a = qg and I = (g) as claimed.

We now show that any g chosen this way divides  $X^n - 1$ . To this end, write  $X^n - 1 = qg + r$  with deg  $r < \deg g$ . Since  $X^n - 1 = 0 \in I$  and  $g \in I$ , we find  $r = a - qg \in I$ , and again the minimality of deg g shows that r = 0; thus  $g \mid X^n - 1$ .

**Example.** Let us determine all cyclic codes in  $R_3$  over  $\mathbb{F}_2$ . From what we have shown we know that cyclic codes correspond to ideals in  $R_3$ , which in turn correspond to ideals (g) with  $g \mid X^3 - 1$ . Now  $X^3 - 1 = (X+1)(X^2 + X + 1)$  in  $\mathbb{F}_2[X]$ , and both factors are irreducible. Thus there are four cyclic codes:

- g = 1: this gives  $C = \mathbb{F}_2^3$ .
- g = X + 1: then  $C = \{000, 011, 101, 110\}$  is the parity check code (the set of all words of even weight).

- 5. Cyclic Linear Codes 5.2 The Algebra behind Cyclic Codes
  - $g = X^2 + X + 1$ : then, as we have seen,  $C = \{000, 111\}$  is the triple repetition code.
  - $g = X^3 1$ : then g = 0, hence  $C = \{000\}$ .

**Exercise 5.5.** Compute all cyclic codes in  $R_4$  over  $\mathbb{F}_2$ .

If I is an ideal in  $R_n$ , then any polynomial  $g \in I$  with minimal nonnegative degree is called a *generator polynomial*.

Our next goal is to compute the dimension as well as generator and parity check matrices of cyclic codes.

**Proposition 5.5.** Let C be a cyclic code in  $R_n$  with generator polynomial  $g = g_0 + g_1 X + \ldots + g_r X^r$ . Then dim C = n - r, and

$$G = \begin{pmatrix} g_0 & g_1 & \dots & g_r & 0 & \dots & 0\\ 0 & g_0 & \dots & g_{r-1} & g_r & \dots & 0\\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & 0 & \dots & g_0 & g_1 & \dots & g_r \end{pmatrix}$$

is a generator matrix.

*Proof.* First we claim that  $g_0 \neq 0$ : in fact, if  $g_0 = 0$ , then  $X^{n-1}g(X) = X^{-1}g(X) =: f(X)$  is a codeword, and  $f \in I$  has degree deg  $f = \deg g - 1$  contradicting the minimality of deg g.

Using the fact that  $g_0 \neq 0$  we see that the rows of G are linearly independent (suppose that a linear combination of the rows is 0; a look at the first column shows that the first coefficient is 0, the second row then shows that the second coefficient is 0 etc.). Moreover, the rows correspond to the codewords  $g(X), Xg(X), \ldots, X^{n-r-1}g(X)$ , and it remains to see why these polynomials span  $R_4$  as an  $\mathbb{F}_q$ -vector space.

Remember that any codeword  $a \in C$  has the form a = qg (as an equation in  $\mathbb{F}_q[X]$ , no reduction mod  $X^n - 1$  necessary). Since deg a < n, we find deg q < n - r, hence

$$q(X)g(X) = (q_0 + q_1X + \dots + q_{n-r-1}X^{n-r-1})g(X)$$
  
=  $q_0 + q_1 \cdot Xg(X) + \dots + q_{n-r-1} \cdot X^{n-r-1}g(X).$ 

This is the desired linear combination!

Observe that the generator matrix does not have the standard form (it does not begin with a unit matrix). There is a simple remedy for this: for each i = 0, 1, ..., k - 1 write  $X^{n-k+i} = q_i(X)g(X) + r_i(X)$ , where deg  $r_i < \deg g = n - k$ . Then  $X^{n-k+i} - r_i(X)$  is a multiple of g, hence a codeword. Thus the matrix  $G_1$  whose rows are (vectors corresponding to) the polynomials  $X^{n-k+i} - r_i(X)$  has the form  $(R|I_k)$  for some  $(n-k) \times k$ matrix R (corresponding to the polynomials  $-r_i(X)$ ) and the unit matrix  $I_k$ 

(corresponding to the  $X^{n-k+i}$ ). The rows are independent codewords, and the dimension of the matrix is right, so  $G_1$  is a generator matrix for C.

Suppose that  $X^n - 1 = g(X)h(X)$ ; then h is called the check polynomial of the cyclic code with generator polynomial g. Here's why:

**Proposition 5.6.** Let C be a cyclic code in  $R_n$  with generator polynomial g. Write  $X^n - 1 = g(X)h(X)$ ; then c is a codeword if and only if  $c(X)h(X) \equiv 0 \mod X^n - 1$ .

*Proof.* Assume that  $c \in C$ . Then c = qg, hence ch = qgh = 0 in  $R_n$  since  $gh = X^n - 1 \equiv 0 \mod (X^n - 1)$ .

Conversely, assume ch = 0 in  $R_n$ . Write c = qg + r with  $\deg r < \deg g$ ; then 0 = ch = qgh + rh = rh and  $\deg rh < \deg g + \deg h < n$ . Thus rh = 0in  $\mathbb{F}_q[X]$ , hence r = 0 and c = qg, which shows that c is a codeword.

Next we unveil the connection between check polynomials and check matrices:

**Theorem 5.7.** Let C be a cyclic code of type [n, k] with check polynomial  $h(X) = h_0 + h_1 X + \ldots + h_k X^k$  (actually  $h_k = 1$ ). Then

	$h_k$	$h_{k-1}$		$h_0$	0		0
	0	$h_k$		$h_1$	$h_0$		0
H =	:	:	•.	۰.	•.	۰.	:
	ŀ	•	•	•	•	•	·
	$\left( 0 \right)$	0		$h_k$	$h_{k-1}$		$h_0$

is a check matrix for C, and  $C^{\perp}$  is a cyclic code with generator polynomial  $\overline{h}(X) = X^k h(\frac{1}{X}) = h_k + h_{k-1}X + \ldots + h_0 X^k$ .

*Proof.* We know that a word  $c = c_0 + c_1 X + \ldots + c_{n-1} X^{n-1}$  is a codeword if and only if ch = 0. In order for ch to vanish, in particular the coefficients of  $X^k$ ,  $X^{k+1}$ , ...,  $X^n$  must be 0. The coefficient of  $X^k$  in ch is

$$c_0h_k + c_1h_{k-1} + \ldots + c_kh_0$$

the coefficient of  $X^{k+1}$  is

$$c_1h_k + c_2h_{k-1} + \ldots + c_{k+1}h_0,$$

..., and the coefficient of  $X^n$  is

$$c_{n-k-1}h_k + c_{n-k}h_{k-1} + \ldots + c_{n-1}h_0$$

Thus codewords  $c \in C$  are orthogonal to the vector  $h_k h_{k-1} \cdots h_0 00 \cdots 0$  and its cyclic shifts, which means the rows of the matrix H are all codewords of  $H^{\perp}$ . Since h is monic, we have  $h_k = 1$ , and it is clear that the rows of Hare independent. Since H has n - k rows, H generates  $C^{\perp}$  and is therefore a check matrix for C. Next we claim that  $\overline{h} \mid X^n - 1$ . From  $\overline{h}(X) = X^k h(\frac{1}{X})$  and  $h(\frac{1}{X})g(\frac{1}{X}) = (\frac{1}{X})^n - 1$  we get  $\overline{h}(X)\overline{g}(X) = X^k h(\frac{1}{X})X^{n-k}g(\frac{1}{X}) = X^n(X^{-n} - 1) = 1 - X^n$ , hence  $\overline{h} \mid X^n - 1$ . Thus  $\overline{h}$  generates a cyclic code with generator matrix H, in other words,  $\overline{h}$  is the generator polynomial of the dual code  $C^{\perp}$  of C.  $\Box$ 

**Proposition 5.8.** The binary cyclic code C in  $R_n$  with generator polynomial X + 1 is the parity code consisting of all words of even weight.

*Proof.* We have dim  $C = n - \deg(X + 1) = n - 1$ , so  $C^{\perp}$  is a 1-dimensional code with generator polynomial  $1 + X + X^2 + \ldots + X^{n-1}$ ; thus  $C^{\perp}$  consists of two words, the all-zeros word and the all-ones word, which implies that C is the parity code.

**Exercise 5.6.** Show that if (X - 1) | g and  $g | X^n - 1$ , then g generates an even cyclic code.

**Exercise 5.7.** Assume that  $g \mid X^n - 1$  and g is not divisible by X - 1. Assume that g generates the code C of type [n, k]. Describe the code generated by (x - 1)g(X).

#### Encoding with Cyclic Codes

Given a cyclic code  $C \subseteq \mathbb{F}_q^n$  of type [n, k] with generator polynomial g of degree n - k, we can encode messages  $m \in \mathbb{F}_q^k$  by writing down a generator matrix G as above and computing c = mG. What happens if we do that?

Write  $m = (m_0, \ldots, m_{k-1})$ ; the product

	$g_0$	$g_1$		$g_{n-k-1}$	0		0 )
$\mathcal{O}$ (	0	$g_0$		$g_{r-1}$	$g_{n-k-1}$	• • •	0
$mG = (m_0, \ldots, m_{k-1})$	:	÷	·	·	·	·	÷
	$\int 0$	0		$g_0$	$g_1$	• • •	$g_{n-k-1}$

is a vector of length n, namely

~

$$mG = (m_0g_0, m_0g_1 + m_1g_0, \dots, m_0g_{n-k-1} + m_1g_{n-k-2} + \dots + m_{n-k-1}g_0, m_1g_{n-k-1} + m_2g_{n-k-2} + \dots + m_{n-k}g_0, \dots, m_kg_{n-k-1}).$$

On the other hand, if we think of m and g as polynomials, then their product is

$$mg = m_0g_0 + (m_0g_1 + m_1g_0)X + \dots + (m_0g_{n-k-1} + m_1g_{n-k-2} + \dots + m_{n-k-1}g_0)X^{n-k-1} + \dots + m_kg_{n-k-1}X^{n-1}.$$

Thus the encoded message c = mG corresponds to the poynomial c(X) = m(X)g(X).

**Example.** Consider the generator polynomial  $g(X) = X^3 + X + 1$  in  $R_7$  (note that  $g \mid X^7 - 1$ ). The dimension of the code C generated by g is dim  $C = n - \deg g = 7 - 3 = 4$ .

Let us encode the message m = 1010. We transform it into a polynomial  $m(X) = X^2 + 1$  and multiply by g(X) to get  $c(X) = m(X)g(X) = X^5 + 2X^3 + X^2 + X + 1$ , giving the codeword 1110010 (note that 2 = 0). We get the same result when we multiply m by the generator matrix

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

Observe that the corresponding parity check matrix is

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

Since the columns of H consists of all nonzero vectors in  $\mathbb{F}_2^3$ , this cyclic code is equivalent to Hamming's [7, 4] code.

In order to get back the original message, we first test whether the word we received is a codeword and compute

$$(1110010)H^T = (000),$$

so the word is a codeword. Since the generator matrix G does not have standard form, it is not obvious how to retrieve the message from the codeword c. Note, however, that columns 1, 5, 6 and 7 'almost' form a unit matrix; in particular, the bits in positions 1, 6 and 7 are the message bits  $m_1$ ,  $m_3$  and  $m_4$ , while  $m_2$  can be computed from  $m_2 + m_4 = c_5$ .

Alternatively, we can check whether c = (1110010) is a codeword by checking whether its product with h(X) is divisible by  $X^7 - 1$ ; but since c(X)h(X) is divisible by  $X^7 - 1 = g(X)h(X)$  if and only if c(X) is divisible by g(X), we might just as well do long division by g: we have to do this anyway for decoding the message, and we find

$$X^{5} + X^{2} + X + 1 = (X^{2} + 1)g(X),$$

and so retrieve the message (1010).

Let us now play the same game with the generator matrix  $G_1$ . We find

$$G_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and  $mG_1 = (0011010)$ . Note that, since  $G_1$  has the unit matrix  $I_4$  as its tail, the message bits are the last 4 bits of the codeword. On the other hand,  $X^3 \cdot m(X) \equiv X^2 + 1 \mod g(X)$ , so the message is represented by  $X^3 \cdot m(X) - (X^2 + 1) = X^5 + X^3 + X^2$  corresponding to (0011010).

For checking whether c is a codeword we can compute the parity check matrix  $H_1$  corresponding to  $G_1$ ; since  $G_1 = (R|I_k)$ , we have  $H_1 = (I_{n-k}| - R^T)$ , that is,

$$H_1 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Checking that  $cH^T = (000)$  is straight forward. Alternatively we could simply divide c(X) by g(X): if c is a codeword, the remainder must be 0.

**Important Observation.** For encoding cyclic codes, one need not compute a generator matrix: messages can be encoded with the help of a generator polynomial alone.

#### Decoding with Cyclic Codes

Now transmit the codeword c(X); assume that the receiver gets the word w(X) = c(X) + e(X), where e(X) is the 'error polynomial'. The syndrome polynomial is the unique polynomial of degree  $< \deg g$  such that  $s(x) \equiv w(X) \mod g(X)$ . Note that since c(X) is a multiple of g, we have  $s(X) \equiv w(X) = c(X) + e(X) \equiv e(X) \mod g(X)$ , so the syndrome polynomial only depends on the error, and codewords have syndrome 0.

There is a parity check matrix  $H_1$  corresponding to the generator matrix  $G_1 = (R|I_k)$ , namely  $H_1 = (I_{n-k}| - R^T)$ ; of course +1 = -1 over fields with characteristic 2 (e.g.  $\mathbb{F}_q$  with q a power of 2).

We now show how to compute the syndrome  $syn(w) = wH_1^T$  of some word w using only polynomial arithmetic. Let w(X) be the polynomial corresponding to w; then

$$s(X) = w_0 + w_1 X + \dots + w_{n-k-1} X^{n-k-1} + w_{n-k} r_0(X) + \dots + w_{n-1} r_{k-1}(X)$$
  
=  $w_0 + w_1 X + \dots + w_{n-k-1} X^{n-k-1} + w_{n-k} (X^{n-k} - q_0(X)g(X))$   
+  $\dots + w_{n-1} (X^{n-1} - q_{k-1}(X)g(X))$   
=  $r(X) - (w_{n-k}q_0(X)g(X) + \dots + w_{n-1}q_{k-1}(X)g(X))$   
=  $r(X) - h(X)g(X).$ 

Thus  $s(X) \equiv w(X) \mod g(X)$  can be computed as the remainder of long division of w(X) by g(X).

Moreover, if we know the syndrome s(X) of w, we can compute the syndrome of Xw(X) as follows: we have

$$w(X) = q(X)g(X) + s(X),$$

hence

$$Xw(X) = Xq(X)q(X) + Xs(X)$$

Note, however, that Xs(X) might have degree  $\geq \deg g$ , so we have to be careful. In fact, if  $\deg s < \deg g - 1$ , then the syndrome  $s_1(X)$  of  $X \cdot w(X)$  is just  $X \cdot s(X)$ ; if, on the other hand,  $\deg s = \deg g = n - k$ , then the coefficient  $s_{n-k-1}$  of s(X) is nonzero, and we have  $s_1(X) = X \cdot s(X) - s_{n-k-1}g(X)$ . This formula is valid also if  $s_{n-k-1} = 0$ , and can be applied inductively to find  $s_2(X)$  from  $s_1(X)$  etc., that is, we can compute the syndromes  $s_i(X)$  for all words  $X^i w(X)$  from s(X) alone.

**Proposition 5.9.** Let C be a cyclic code in  $R_n$  with generator polynomial g. If s(X) = syn(w), then  $syn(X \cdot w) = X \cdot s(X) - s_{n-k-1}g(X)$ .

Now let us talk about decoding and error correction. Assume C is a code of type [n, k, d] with d = 2t + 1, so C is t-error correcting. Let  $H_1 = (I_{n-k}|A)$ be the parity check matrix. If you send a codeword c and an error e of weight at most t occurs, you receive the word w = c + e, and we know that  $\operatorname{syn}(w) =$  $\operatorname{syn}(e)$ . Now let  $e^* = (s|0)$  denote the word in  $\mathbb{F}_q^n$  you get from gluing zeros to the syndrome  $s = \operatorname{syn}(w)$ ; it is easy to see that  $\operatorname{syn}(e^*) = \operatorname{syn}(e)$ . Now  $\operatorname{suppose}$  (this is not true in general) that  $\operatorname{wt}(s) \leq t$ ; then  $\operatorname{wt}(e^*) \leq t$ , and since e and  $e^*$  are in the same coset mod C, they must be equal (their difference is a codeword of weight  $\leq 2t$ ). Thus, in this case, we can determine e as  $e = e^* = (s|0)$ .

Now assume that C is cyclic with generator polynomial g(X). Let e be an error pattern with a cyclic run of at least k zeros. Then there is some i with  $0 \le i \le n-1$  such that the *i*-fold cyclic shift of e is a vector whose nonzero components all lie within the first n-k coordinates. For this i, wt $(s_i) \le t$ , where  $s_i$  is the syndrome of  $X^i w(X)$ .

Thus to correct and decode a message satisfying the above assimptions, compute the polynomials  $s_i(X)$ ; if wt  $s_i \leq t$ , then  $X^i e(X) = (s_i|0)$ , and you can compute e(X); if not, then the error was too clever to be trapped.

**Error Trapping** Let C be a cyclic code with minimum distance t. Assume you receive the message w(X).

- 1. Compute the syndrome polynomial  $s(X) \equiv w(X) \mod g(X)$ .
- 2. For i = 0, ..., n-1 compute the cyclic shifts  $s_i(X) \equiv X^i s(X) \mod g(X)$ until you find a syndrome  $s_j$  such that  $\operatorname{wt}(s_j) \leq t$ . The error word of minimal weight is then  $e(X) = X^{n-j} s_j(X) \mod (X^n - 1)$ .

**Example.** Let  $g(X) = X^3 + X + 1$  and w = (1011011); then  $w(X) = 1 + X^2 + X^3 + X^5 + X^6$ ,  $syn(w) = X^2$  since  $w(X) = (X^3 + X^2 + X + 1)g(X) + X^2$ , so assuming a single error, the correct codeword was (1001011).

**Big Example.** Let  $g(X) = 1 + X^4 + X^6 + X^7 + X^8$  be a generator of a cyclic code of type [15, 7, 5]; then t = 2, so C is 2-error correcting. Take w = (110011101100010); then

$$w(X) = (X + X^{2} + X^{4} + X^{5})g(X) + (1 + X^{2} + X^{5} + X^{7}),$$

and we compute the syndromes  $s_i(X)$ :

```
0
  10100101
1
  11011001
2
  11100111
3
  11111000
4
  01111100
5
  00111110
6
  00011111
7
  10000100
```

Thus the error word is

$$X^{15-7}(s_7|0) = X^8(10000100000000) = (000000001000010),$$

and we get r - e = (110011100100000).

#### Minimal Distance of Cyclic Codes

There is no simple relation between the generating polynomial and the minimal distance of a cyclic code. There is, however, a lower bound which sometimes yields good results.

**Theorem 5.10.** Let  $g \mid (X^n - 1) \in \mathbb{F}_q[X]$  be a generator polynomial for a cyclic code C, and let  $\alpha \in \mathbb{F}_{q^n}$  be a primitive n-th root of unity in some extension field (this means that n is the minimal integer > 0 with  $\alpha^n = 1$ ). If the powers  $\alpha^b$ ,  $\alpha^{b+1}$ ,  $\alpha^{b+2}$ , ...,  $\alpha^{b+r-1}$  are roots of g, then  $d(C) \ge r+1$ .

*Proof.* Assume that  $c := \sum_{i=0}^{n-1} c_i x^i \in C$  is a codeword of weight  $s \geq r$ , and let  $c_{t_1}, \ldots, c_{t_s}$  be the nonzero coordinates of c. Since c(X) is a multiple of g(X), and since  $\alpha^b, \ldots, \alpha^{b+r-1}$  are roots of g, we find  $c(\alpha^b) = c(\alpha^{b+1}) = \ldots = c(\alpha^{b+r-1}) = 0$ . Thus

$$\begin{array}{rcrcrcrcrc} c_{t_1}\alpha^{bt_1} & + & \dots & + & c_{t_s}\alpha^{bt_s} & = & 0\\ c_{t_1}\alpha^{(b+1)t_1} & + & \dots & + & c_{t_s}\alpha^{(b+1)t_s} & = & 0\\ \vdots & & & \vdots & & \vdots\\ c_{t_1}\alpha^{(b+r-1)t_1} & + & \dots & + & c_{t_s}\alpha^{(b+r-1)t_s} & = & 0. \end{array}$$

With

$$H := \begin{pmatrix} \alpha^{bt_1} & \dots & \alpha^{bt_s} \\ \alpha^{(b+1)t_1} & \dots & \alpha^{(b+1)t_s} \\ \vdots & \vdots & \vdots \\ \alpha^{(b+s-1)t_1} & \dots & \alpha^{(b+s-1)t_s} \end{pmatrix}$$

we get in particular  $H(c_{t_1}, \ldots, c_{t_s})^T = 0$ , that is,  $c \in \ker H$ . On the other hand the determinant of H is given by

$$\det(H) = \alpha^{bt_1} \alpha^{bt_2} \cdot \ldots \cdot \alpha^{bt_s} \det \begin{pmatrix} 1 & \ldots & 1 \\ \alpha^{t_1} & \ldots & \alpha^{t_s} \\ (\alpha^{t_1})^2 & \cdots & (\alpha^{t_s})^2 \\ \vdots & \vdots & \vdots \\ (\alpha^{t_1})^{s-1} & \ldots & (\alpha^{t_s})^{s-1} \end{pmatrix}$$
$$= \alpha^{bt_1} \alpha^{bt_2} \cdot \ldots \cdot \alpha^{bt_s} \prod_{i < j} (\alpha^{t_j} - \alpha^{t_i})$$

(this is a Vandermonde determinant). Since  $\alpha$  is a primitive *n*-th root of unity, we have  $\alpha^{t_i} \neq \alpha^{t_j}$  for  $i \neq j$ , hence  $\det(H) \neq 0$ . But then ker H = 0, and therefore c = 0. Thus the only codeword of weight  $\leq r$  is the all-zeros word.

**Example.** We know  $X^7 - 1 = (X+1)(X^3 + X + 1)(X^3 + X^2 + 1) \in \mathbb{F}_2[X]$  from the homework. Let  $g = X^3 + X + 1$  and  $\mathbb{F}_8 = \mathbb{F}_2[X]/(X^3 + X + 1)$ . Then every root  $\alpha$  of g is a generator of  $\mathbb{F}_8^{\times}$ , also from the homework. Since  $g \in \mathbb{F}_2[X]$ , the Frobenius  $\operatorname{Frob}(x) = x^2$  maps roots of g onto roots of g, hence  $\alpha$  and  $\alpha^2 = \operatorname{Frob}(\alpha)$  are roots of g. Theorem 5.10 shows that the code generated by g has minimum distance  $\geq 3.$  Actually, CisHamming's[7, 4]-code, whose minimum distance is 3.

**Example.** Now consider  $X^{23} - 1 = (X + 1)g_1g_2 \in \mathbb{F}_2[X]$  with

$$g_1 = X^{11} + X^9 + X^7 + X^6 + X^5 + X + 1,$$
  
$$g_2 = X^{11} + X^{10} + X^6 + X^5 + X^4 + X^2 + 1$$

Any root  $\alpha$  of  $g_1$  is a primitive 23rd root of unity. Applying the Frobenius to  $\alpha$  we find that  $g_1$  has the roots

$$\alpha, \alpha^{2}, \alpha^{4}, \alpha^{8}, \alpha^{16}, \alpha^{32} = \alpha^{9}, \alpha^{18}, \alpha^{36} = \alpha^{13}, \alpha^{3}, \alpha^{6}, \alpha^{12};$$

In particular,  $g_1$  has the 'consecutive' roots  $\alpha, \alpha^2, \alpha^3, \alpha^4$ , hence C has minimum distance  $\geq 5$ . It can be shown that  $C = G_{23}$  is Golay's [23, 12, 7] code. Thus the bound from Theorem 5.10 is not always best possible.

Cyclic Linear Codes

#### **Cyclic Redundancy Checks**

Cyclic Redundancy Codes (CRC) are used for error detection (not correction) and use methods very similar to cyclic codes. Here's how it works: pick a generator polynomial  $g(X) \in \mathbb{F}_2[X]$  of degree k; to each message of n + 1bits add k parity bits as follows: write  $m = (m_n \dots m_0)$  as a polynomial  $m(X) = m_0 + m_1 X + \dots + m_n X^n$ ; then compute  $r(X) \equiv X^k m(X) \mod g(X)$ with deg  $r < \deg g$ . Write r(X) as a vector and add these to the message as parity bits by sending  $X^k m(X) + r(X)$ .

Example: g(X) = X + 1,  $m = (m_n, \ldots, m_0)$ ; then  $Xm(X) = m_0X + m_1X^2 + \ldots + m_nX^{n+1} \equiv m_0 + m_1 + \ldots + m_n \mod X + 1$ , so we append the parity bit  $b = m_0 + m_1 + \ldots + m_n$  to the message and send  $(m_n, \ldots, m_0, b)$ . We know that this is a 1-error detecting code.

Here's another one: take  $g(X) = X^2 + 1$ ; then  $X^2m(X) = m_0X^2 + m_1X^3 + \dots + m_nX^{n+2} \equiv (m_0 + m_2 + m_4 + \dots) + (m_1 + m_3 + m_5 + \dots)X$ , and the two check bits appended to the message m are the sum of the bits in the even and in the odd positions. Note that if an error occurs in two consecutive bits, then the both check bits be wrong, so we can detect such errors.

How are errors detected in general? Note that the word  $X^k m(X) + r(X)$  that was sent is divisible by g since  $X^k m(X) \equiv r(X) \mod g(X)$  and -1 = 1 in  $\mathbb{F}_2$ . Thus if the polynomial corresponding to the received method is not divisible by g, then there must have been errors in the transmission.

The most frequently used generator polynomials for CRCs are

- CRC-12:  $X^{12} + X^{11} + X^3 + X^2 + X + 1;$
- CRC-16:  $X^{16} + X^{15} + X^2 + 1;$
- CRC-CCITT:  $X^{16} + X^{12} + X^5 + 1$

as well as a polynomial of degree 32 for CRC-32.

#### 5.3 Reed-Muller Codes

Reed-Muller Codes (one of those was used for transmitting pictures from Mars during the Mariner 9 mission in 1972) are a family of codes RM(r,m) can be defined by induction:

- 1.  $RM(0,m) = \{0 \cdots 0, 1 \cdots 1\}$  is the code of length  $2^m$  consisting of the all-zeros and the all-ones word.
- 2.  $RM(m,m) = \mathbb{F}_2^{2^m}$ .
- 3. RM(r,m) = RM(r,m-1) \* RM(r-1,m-1) for 0 < r < m, where \* denotes the Plotkin construction from Chapter 3.

Using induction, it is not very hard to prove

**Theorem 5.11.** The code RM(r,m) is a linear code of length  $2^m$ , dimension  $\binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{r}$  and minimum distance  $2^{m-r}$ .

Some Reed-Muller codes are old friends:

**Theorem 5.12.** The code RM(1,m) is the dual code of  $Ham^+(m)$ , the parity extension code of Hamming's code Ham(m).

Moreover, dual codes of Reed-Muller codes are Reed-Muller again:

**Theorem 5.13.** We have  $RM(r,m)^{\perp} = RM(m-r-1,m)$ .

### 5.4 Quadratic Residue Codes

Quadratic residue codes are codes of length p defined over  $\mathbb{F}_l$ , where p and l are primes such that (l/p) = +1 (l is a quadratic residue modulo p). Thus over  $\mathbb{F}_2$ , QR codes have length p with  $p \equiv \pm 1 \mod 8$ .

QR codes are cyclic codes; the conditions above guarantee (this was proved by Gauss in his Disquisitiones) that

$$X^{p} - 1 = (X - 1)q(X)n(X), \quad \text{where}$$
$$q(X) = \prod_{(r/p)=+1} (X - \alpha^{r}),$$
$$n(X) = \prod_{(n/p)=-1} (X - \alpha^{n}).$$

Here  $\alpha$  is a primitive *p*-th root of unity in a suitable extension of  $\mathbb{F}_l$ . It is easy to see that  $q(X) \in \mathbb{F}_l[X]$ : we just apply the Frobenius  $\operatorname{Frob}(x) = x^l$  to q(X); since Frob maps squares to squares, it permutes the  $\alpha^r$ , hence fixes the resulting polynomial.

**Theorem 5.14.** The minimum distance  $d_p$  of the QR code satisfies  $d_p \ge \sqrt{p}$  if  $p \equiv 1 \mod 4$  and  $d_p^2 - d_p + 1 \ge \sqrt{p}$  if  $p \equiv 3 \mod 4$ .

code	n	k	$d \ge$	defined by
$\overline{Q}_p$	p	$\frac{p-1}{2}$	$d_p + 1$	cyclic, $g = (X - 1)q(X)$
$Q_p$	p	$\frac{p+1}{2}$	$d_p$	cyclic, $g = q(X)$
$Q_p^+$	p+1	$\frac{p+1}{2}$	$d_p + 1$	parity extension of $Q_p$

We have already stated that the QR code  $Q_{23}$  is Golay's [23, 12, 7] code, and that  $Q_{23}^+$  is the extended Golay code of type [24, 12, 8], corresponding to the third row in the table below.

5. Cyclic Linear Codes

The state of the art of determining the minimum distance of extended quadratic residue codes  $Q_p^+$  over  $\mathbb{F}_2$  as of June 2000:

n	k	d	n	k	d	n	k	d
8	4	4	74	37	14	 138	69	14 - 22
18	9	6	80	40	16	152	76	20
24	12	8	90	45	18	168	84	16 - 24
32	16	8	98	49	16	192	96	16 - 28
42	21	10	104	52	20	194	97	16 - 28
48	24	12	114	57	12 - 16	200	100	16 - 32
72	36	12	128	64	20			

QR codes have large automorphism groups: the group PSL(2, p) is a subgroup of  $Aut(Q_p)$  over  $\mathbb{F}_2$ . Here PSL(2, p) denotes the projective special linear group with entries from  $\mathbb{F}_p$ : this is a group of order  $\frac{1}{2}(p-1)p(p+1)$ . Note that # PSL(2,7) = 168.

### 5.5 BCH Codes

These codes are named after Bose, Chandhuri, and Hocquenghem. The idea is to construct cyclic codes with large minimum distance by constructing a generator polynomial in such a way that it has a string of roots of unity as roots.

Let *C* be a cyclic code in  $\mathbb{F}_q^n$  and assume that gcd(n,q) = 1. Let *m* denote the order of  $q \mod n$  and let  $\alpha$  be a generator of the multiplicative group of  $\mathbb{F}_{q^m}$ . Then *C* is a BCH code of designed distance  $\delta$  if the generating polynomial g(X) of *C* is the product of the distinct minimal polynomials of the  $\delta - 1$  consecutive powers  $\alpha^b$ ,  $\alpha^{b+1}$ , ...,  $\alpha^{b+\delta-2}$ . By Theorem 5.10, the minimum distance of a BCH code of designed distance  $\delta$  is  $\geq \delta$ .

### 5.6 Reed-Solomon Codes

Reed-Solomon Codes were invented by Reed and Solomon in 1960. These codes are relatively easy to encode, but decoding was sort of a problem despite improvements by Berlekamp and others. NASA used Reed-Solomon codes for some of their missions to Mars; the problem of decoding was taken care of by computers on earth, so fact that decoding was hard wasn't much of an obstacle. In recent years, better ways of decoding have been developed, and nowadays Reed-Solomon codes are used for CDs, wireless communication, DVD or digital TV, to mention a few.

The quality of Reed-Solomon codes can be seen from the fact that CD-ROMs can correct up to 4000 consecutive errors! To reach such a large bursterror correcting capability, one uses the technique of interleaving.

Here's the recipe for constructing Reed-Solomon codes. Take a root  $\alpha$  of  $X^3 + X + 1 \in \mathbb{F}_2[X]$ ; we know that  $\alpha$  generates  $\mathbb{F}_8^{\times}$ . Form the polynomial

$$g(X) = (X + \alpha^5)(X + \alpha^6) = X^2 + \alpha X + \alpha^4 \in \mathbb{F}_8[X].$$

The cyclic code  $C \subset \mathbb{F}_8^7$  generated by g has type [7, 5, 3] (the fact that  $d \geq 3$  follows from Theorem 5.10). Now every element in  $\mathbb{F}_8 = \mathbb{F}_2[X]/(g)$  can be written as a vector in  $\mathbb{F}_2^3$ ; thus we can interpret C as a code of length  $21 = 3 \cdot 7$  and dimension  $15 = 3 \cdot 5$  over  $\mathbb{F}_2$  with generator matrix

/	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	
	0	0	0	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	
	0	0	0	0	0	0	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	0	
	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	1	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	1/	

Observe that this code is also described by  $g(X) = X^2 + \alpha X + \alpha^4$ .

It turns out that its minimal distance is still 3, which is rather bad. However, by adding a parity bit in  $\mathbb{F}_2$  to every triple of bits makes C into a code of length 28 and dimension 15 whose minimum distance turns out to be 6; the first row of its generator matrix is

#### 

More generally, if you start with a code generated by a suitable polynomial of degree r in  $\mathbb{F}_q^{q-1}$ , where  $q = 2^f$ , we get a binary linear code of type [fn, f(n-r), d] with  $d \ge r+1$ . Adding a parity bit to each f-tuple gives a code of type [(f+1)n, f(n-r), d'] with  $d' \ge 2(r+1)$ .

This description of Reed-Solomon codes shows that these are cyclic codes. There is a completely different description that motivated the invention of algebraic geometry codes by Goppa.

Take a message  $m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}_2^k$  and form the polynomial  $m(X) = m_0 + m_1 X + \ldots + m_{k-1} X^{k-1} \in \mathbb{F}_2[X]$ . Suppose we evaluate this polynomial at k different values  $x_i$  and record the results  $(x_0, y_0), \ldots, (x_{k-1}, y_{k-1})$  where  $y_i = m(x_i)$ . If we transmit these k points instead of m, and if no error occurs, then we can recompute m since there is a unique polynomial of degree

Cyclic Codes

 $\leq k-1$  going through k different points. If we fix the k x-coordinates, all we have to do is transmit the k y-coordinates, which, over  $\mathbb{F}_2$ , are just bits: in other words: transmitting the points is as cheap as transmitting the message itself.

In order to get an error-correcting code we evaluate m at n = k + 2s values for some  $s \ge 1$ ; assume that at most s errors were made. Then there must exist a polynomial of degree  $\le k - 1$  passing through at least k + s of the n points  $(x_0, y_0), \ldots, (x_n, y_n)$ . We compute this polynomial, and its coefficients form the original message m.

Of course 'computing this polynomial' is much easier said than done, but over the last few decades an algorithm has been developed that does this very efficiently. It consists of several steps:

- Calculating syndromes by evaluating the received message at several points;
- Berlekamp's algorithm finds an 'error locator polynomial' and the number v of errors made (assuming  $v \leq s$ ); this involves solving s linear equations in s unknowns.
- Chien search finds the roots of L from L and v;
- Forney's algorithm computes the symbol error values from the syndromes, L, and its roots. This again involves solving s linear equations in s unknowns.
- Combining the pieces above to reconstruct the original mesage.

As you can see, this looks somewhat nontrivial. In fact, there are several nice ideas involved in the algorithms, but luckily there's nothing too fancy going on: arithmetic of polynomials over finite fields, linear algebra, and Lagrange interpolation.

Let us look more closely at the following special case. Fix  $\mathbb{F}_q$  and let  $\alpha_1, \ldots, \alpha_{q-1}$  denote the nonzero elements of  $\mathbb{F}_q$ . Let  $L_r$  denote the set of polynomials in  $\mathbb{F}_q[X]$  with degree  $\leq r$ ; clearly  $L_r$  is an  $\mathbb{F}_q$ -vector space of dimension r + 1. For each integer k with  $1 \leq k \leq q - 1$  we define the Reed-Solomon code

$$RS(k,q) = \{ (f(\alpha_1), \dots, f(\alpha_{q-1})) : f \in L_{k-1} \}.$$

Then  $RS(k,q) \subseteq \mathbb{F}_q^{q-1}$ , so RS(k,q) has length q-1. Moreover, evaluation of  $f \in L_{k-1}$  defines a map

$$\varepsilon: L_{k-1} \longrightarrow RS(k,q): f \longmapsto (f(\alpha_1), \dots, f(\alpha_{q-1}))$$

which is easily seen to be an injective linear map; thus dim RS(k,q) = k - 1.

Finally, assume that  $f \in L_{k-1}$  gives rise to a codeword  $c = \varepsilon(f)$  of minimum weight d > 0. Then f has at least n - d zeros, so  $k - 1 = \deg f \ge n - d$ ; this implies  $d \ge q - k$ . By the Singleton bound (Proposition 3.23), we have equality.

**Proposition 5.15.** The Reed-Solomon code RS(k, d) is a linear code of type [q-1, k, q-k].

Finally, let me quote from an article on 'The Decoding of Algebraic Geometry Codes' from 1995:

At the beginning of the invention of RS codes, the engineering community had the impression that these codes would never be used in practice, since it used advanced mathematics such as Galois fields.

This changed with the development of fast decoding algorithms for RS codes, and the authors express their hope that something similar is going to happen for algebraic geometry codes, which also have the reputation of using 'too advanced mathematics'.

### 5.7 Algebraic Geometry Codes

This family of codes is almost impossible to describe accurately without introducing a lot of new notions. I will simplify the description in order to get by with just a few new concepts. The best first glance into these codes is offered by Judy Walker's booklet [15]; there are almost no proofs, however. Pretzel's book [12] is a very well written introduction, although I personally don't like the way he avoids projective planes through three affine charts.

#### **Classical Goppa Codes**

Let  $L = \{\alpha_1, \ldots, \alpha_n\}$  be a set of n distinct elements of  $\mathbb{F}_{q^m}$ , and let  $g \in \mathbb{F}_{q^m}[X]$  be a polynomial such that  $g(\alpha_i) \neq 0$  for all  $1 \leq i \leq n$ . The classical Goppa code  $\Gamma(L, g)$  is defined as the set of all  $c = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$  such that

$$\sum_{i=1}^{n} \frac{c_i}{X - \alpha_i} \equiv 0 \mod g.$$

This congruence means that, when the sum is written as a fraction of two polynomials, the numerator is divisible by g (note that the denominator is coprime to g since it doesn't have a root in common with g).

It can be shown that Goppa codes  $\Gamma(L, g)$  with  $L = \{\beta^{-i} : 0 \le i \le n-1\}$ and  $g(X) = X^{d-1}$ , where  $\beta$  is a primitive *n*-th root of unity, are BCH codes with designed distance *d*.

#### **Goppa's Geometric Codes**

Goppa's idea was to interpret Reed-Solomon codes in the following way: the points  $\alpha_1, \ldots, \alpha_{q-1}$  are the nonzero rational points on the line y = 0 over

Cyclic Codes

 $\mathbb{F}_q$ , and polynomials of degree  $\leq k-1$  are a certain vector space of nice functions on the line. In order to get a good minimum distance we would like to increase q, but unfortunately there are only  $q \mathbb{F}_q$ -rational points on the line y = 0.

Goppas insight was that we get more rational points when we replace lines by algebraic curves. Such curves exist in spaces of arbitrary high dimension, but we shall restrict our attention to plane algebraic curves: these are the sets of zeros of equations  $f \in F[X, Y]$ , that is,

$$\mathcal{X} = \{ (x, y) \in K \times K : f(x, y) = 0 \}.$$

Over  $F = \mathbb{R}$ , the curve described by  $f(X, Y) = X^2 + Y^2 - 1$  is just the unit circle. We will assume that our curves are (absolutely) irreducible, thus  $f = X^2 - Y^2$  is not regarded as a curve since f = (X - Y)(X + Y).

Given such a curve  $\mathcal{X}$  defined by an irreducible f, we can form the function field  $F(\mathcal{X})$  as the field of fractions of the integral domain  $F[\mathcal{X}] := F[X, Y]/(f)$ (note that f is irreducible, thus prime, but not necessarily maximal). This function field  $F(\mathcal{X})$  corresponds to the vector space of polynomials in the linear case; we now have to come up with an analogue of polynomials of bounded degree. To this end, consider a set  $L = \{P_1, \ldots, P_n\}$  of F-rational points on  $\mathcal{X}$ ; we associate to such a set a divisor  $D = P_1 + \ldots + P_r$ , a formal 'sum' of these points; the support of D is the set L.

Now to every  $f \in F(\mathcal{X})$  we associate a divisor div  $(f) = P_1 + \ldots P_r - Q_1 - \ldots - Q_s$ , where the  $P_i$  are poles and the  $Q_i$  are zeros of f, counted with multiplicity. It is a basic theorem that when these things are counted correctly, then r = s. If f is a polynomial of degree k, then the  $Q_1, \ldots, Q_k$  are the usual zeros, and  $P_1 = \ldots = P_k = \infty$  is the point at infinity. Now we define the Riemann-Roch space

$$\mathcal{L}(D) = \{ f \in F(\mathcal{X}) : \operatorname{div}(f) + D \ge 0 \} \cup \{ 0 \}$$

as the vector space of all functions such that the formal sum div (f) + D has nonnegative coefficients.

Now to a smooth curve  $\mathcal{X}$ , a set  $L = \{P_1, \ldots, P_n\}$  of *F*-rational points, and a divisor *D* with support disjoint from *L* we associate a linear code

$$C(\mathcal{X}, L, D) = \{ (f(P_1), \dots, f(P_n)) : f \in \mathcal{L}(D) \} \subseteq \mathbb{F}_q^n.$$

These algebraic geometry codes have high minimum distance if they have many  $\mathbb{F}_q$ -rational points. This fact has led to an industry of people trying to construct curves  $\mathcal{X}$  with a maximal number of  $\mathbb{F}_q$ -rational points. If  $\mathcal{X}$  is given by

$$y^2 - x^3 - ax^2 - bx - c, \quad a, b, c \in \mathbb{F}_q,$$

where the polynomial  $x^3 + ax^2 + bx + c$  is supposed not to have multiple roots, then  $\mathcal{X}$  is called an elliptic curve, and the number of  $\mathbb{F}_q$ -rational points on  $\mathcal{X}$ is bounded by the Hasse bound

Error-Correcting Codes

$$|\#\mathcal{X} - (q+1)| \le 2\sqrt{q}.$$

There exist similar bounds for general smooth algebraic curves; these were first proved by Weil and are equivalent to the Riemann conjecture for the zeta functions of these curves.

# 6. Other Classes of Codes

#### 6.1 Reed-Muller Codes

Reed-Muller Codes (one of those was used for transmitting pictures from Mars during the Mariner 9 mission in 1972) are a family of codes RM(r,m) can be defined by induction:

- 1.  $RM(0,m) = \{0 \cdots 0, 1 \cdots 1\}$  is the code of length  $2^m$  consisting of the all-zeros and the all-ones word.
- 2.  $RM(m,m) = \mathbb{F}_2^{2^m}$ .
- 3. RM(r,m) = RM(r,m-1) \* RM(r-1,m-1) for 0 < r < m, where \* denotes the Plotkin construction from Chapter 3.

Using induction, it is not very hard to prove

**Theorem 6.1.** The code RM(r,m) is a linear code of length  $2^m$ , dimension  $\binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{r}$  and minimum distance  $2^{m-r}$ .

Some Reed-Muller codes are old friends:

**Theorem 6.2.** The code RM(1,m) is the dual code of  $\operatorname{Ham}^+(m)$ , the parity extension code of Hamming's code  $\operatorname{Ham}(m)$ .

Moreover, dual codes of Reed-Muller codes are Reed-Muller again:

**Theorem 6.3.** We have  $RM(r, m)^{\perp} = RM(m - r - 1, m)$ .

### 6.2 Reed-Solomon Codes

Reed-Solomon Codes were invented by Reed and Solomon in 1960. These codes are relatively easy to encode, but decoding was sort of a problem despite improvements by Berlekamp and others. NASA used Reed-Solomon codes for some of their missions to Mars; the problem of decoding was taken care of by computers on earth, so fact that decoding was hard wasn't much of an obstacle. In recent years, better ways of decoding have been developed, and nowadays Reed-Solomon codes are used for CDs, wireless communication, DVD or digital TV, to mention a few.

The quality of Reed-Solomon codes can be seen from the fact that CD-ROMs can correct up to 4000 consecutive errors!

Error-Correcting Codes

Efficient Encoding

Efficient Decoding

6.3 Algebraic Geometry Codes

# 7. Sphere Packings and Codes

### 7.1 Lattices

**Definition 7.1.** A (full) lattice  $\Lambda$  is  $\mathbb{R}^n$  is a discrete subgroup of  $\mathbb{R}^n$  (containing a basis of  $\mathbb{R}^n$ ).

All our lattices will be full lattices.

The subgroup property means that, given two lattice points  $x, y \in \Lambda$ , we also have  $x \pm y \in \Lambda$ ; discrete means that there is a positive  $\varepsilon > 0$  such that every ball in  $\mathbb{R}^n$  with radius  $\varepsilon$  contains at most one lattice point. It can be shown that every lattice has a 'basis'  $x_1, \ldots, x_n$  such that every  $x \in \Lambda$  can be written uniquely as

$$x = a_1 x_1 + \ldots + a_n x_n, quad, a_i \in \mathbb{Z}.$$

In other words:  $\Lambda = x_1 \mathbb{Z} \oplus \ldots \oplus x_n \mathbb{Z}$ .

The simplest lattice in  $\mathbb{R}^n$  is probably the lattice  $\Lambda = \mathbb{Z}^n$  generated by the standard basis  $e_1 = (1, 0, \dots, 0), \dots, e_n - (0, \dots, 0, 1)$ . For every lattice  $\Lambda$  and every positive real number  $\lambda$ , we can define the lattice  $\lambda \Lambda = \{\lambda x : x \in \Lambda\}$ ; showing that  $\lambda \Lambda$  is a discrete subgroup of  $\mathbb{R}^n$  is left as an exercise.

In the vector space  $\mathbb{R}^n$  we have the standard scalar product  $x \cdot y = xy^t = x_1y_1 + \ldots + x_ny_n$ . Every lattice  $\Lambda \subset \mathbb{R}^n$  has a dual lattice

$$\Lambda^* = \{ x \in \mathbb{R}^n : x \cdot y \in \mathbb{Z} \text{ for all } y \in \Lambda \}.$$

**Definition 7.2.** A lattice  $\Lambda$  is called an integral lattice if  $\Lambda \subseteq \Lambda^*$ . We call  $\Lambda$  unimodular if  $\Lambda = \Lambda^*$ .

**Definition 7.3.** An integral lattice  $\Lambda$  is called even if  $x \cdot x \equiv 0 \mod 2$  for all  $x \in \Lambda$ .

Determinants of lattices ...

# 7.2 Codes and Lattices

Both  $\Lambda = \mathbb{Z}^n$  and  $\mathbb{F}_2^n$  are additive groups, and we have a group homomorphism

 $\rho: \mathbb{Z}^n \longrightarrow \mathbb{F}_2^n; (a_1, \ldots, a_n) \longmapsto (a_1 \mod 2, \ldots, a_n \mod 2).$ 

Consider a code  $C \subseteq \mathbb{F}_2^n$ ; its preimage

$$\rho^{-1}(C) = \{ x \in \mathbb{Z}^n : \rho(x) \in C \}$$

is a lattice in  $\mathbb{R}^n$  lying 'between'  $\mathbb{Z}^n$  and  $2\mathbb{Z}^n$  in the sense that  $2\mathbb{Z}^n \subseteq \rho^{-1}(C) \subseteq \mathbb{Z}^n$ .

**Lemma 7.4.** If dim C = k, then  $(\mathbb{Z}^n : \rho^{-1}(C)) = 2^{n-k}$ .

Put  $\Gamma_C = \frac{1}{\sqrt{2}} \rho^{-1}(C)$ .

**Theorem 7.5.** Let  $C \subseteq \mathbb{F}_2^n$  be a code and  $\Gamma_C$  its associated lattice.

1. We have  $C \subseteq C^{\perp}$  if and only if  $\Gamma_C$  is integral;

2. C is self dual if and only if  $\Gamma_C$  is unimodular;

3. C is doubly even if and only if  $\Gamma_C$  is even.

Proof.

### 7.3 Sphere Packings

packing density; Minkowski-Hlawka bound

# 7.4 The Leech Lattice

Let  $\mathcal{G}_{24}$  be the extended Golay code. The lattice  $\Lambda = \rho^{-1}(\mathcal{G}_{24})$  is an even lattice in  $\mathbb{R}^{24}$ .

## 7.5 Finite Simple Groups

# A. Algebraic Structures

### A.1 Groups

A group G is a set of elements on which a composition  $\circ$  is defined such that the following conditions are satisfied:

- closure:  $a \circ b \in G$  whenever  $a, b \in G$ ;
- associativity:  $(a \circ b) \circ c = a \circ (b \circ c)$  for all  $a, b, c \in G$ ;
- neutral element: there is an  $e \in G$  such that  $a \circ e = e \circ a = a$  for every  $a \in G$ ;
- inverse element: for all  $a \in G$  there is a  $b \in G$  such that  $a \circ b = e$ .

The neutral element is usually denoted by 0 or 1 according as G is written additively ( $\circ = +$ ) or multiplicatively ( $\circ = \cdot$ ).

If  $a \circ b = b \circ a$  for all  $a, b \in G$ , then G is called commutative or abelian.

Examples for groups are  $(G, \circ) = (\mathbb{Z}, +); (\mathbb{Q}, +); (\mathbb{Q}^{\times}, \cdot);$  nonexamples are  $(\mathbb{N}, +); (\mathbb{N}, \cdot); (\mathbb{R}, \cdot).$ 

Other important examples are the symmetry groups of polygons (the dihedral groups; these are nonabelian groups) or the rotation groups of Platonic solids. One of the most illustrious groups is the rotation group of the icosahedron, the alternating group  $A_5$  of order 60. It is the smallest nontrivial simple group, responsible for the fact that most polynomials of degree 5 cannot be solved by radicals, and it tends to pop up everywhere (see Ebeling [5] for a connection with codes).

Large classes of important groups are constructed using matrices. The  $n \times n$ -matrices with nonzero determinant and entries from the finite Field  $\mathbb{F}_p$  form a group  $\operatorname{GL}(n, \mathbb{F}_p)$ , the general linear group. The determinant induces a homomorphism det :  $\operatorname{GL}(n, \mathbb{F}_p) \longrightarrow \mathbb{F}_p^{\times}$ , and its kernel is the (normal) subgroup  $\operatorname{SL}(n, \mathbb{F}_p)$  of all matrices in  $\operatorname{GL}(n, \mathbb{F}_p)$  with determinant 1.

### A.2 Rings

In a ring, we have two compositions that we denote by + and  $\cdot$ . A triple  $(R, +, \cdot)$  is called a ring if the following conditions are satisfied:

- 96 Franz Lemmermeyer
  - additive group: (R, +) is an abelian group;
  - closure of multiplication:  $ab \in R$  whenever  $a, b \in R$ ;
  - associativity: (ab)c = a(bc) for all  $a, b, c \in R$ ;
  - distributivity: a(b+c) = ab + ac for all  $a, b, c \in R$ . The ring is called
  - commutative if ab = ba for all  $a, b \in R$ ;
  - unitary if it has a neutral element with respect to multiplication;
  - an integral domain if it has no zero divisors, that is, if ab = 0 implies a = 0 or b = 0.

Examples of rings (with the usual addition and multiplication) are  $\mathbb{Z}$ ,  $\mathbb{R}$ ,  $\mathbb{Z}[X]$ ; nonexamples are  $\mathbb{N}$  or  $\mathbb{Z}[X]_n$ . The rings  $\mathbb{Z}/m\mathbb{Z}$  with composite m have zero divisors: if m = ab, then  $ab \equiv 0 \mod m$  but  $a, b \not\equiv 0 \mod m$ . The ring  $2\mathbb{Z}$  of even integers is not unitary. The ring of  $2 \times 2$ -matrices with entries in  $\mathbb{Z}$  is not commutative and has zero divisors.

### A.3 Fields

A field  $(F, +, \cdot)$  is a ring with the additional property that  $(F^{\times}, \cdot)$  is an abelian group (that is we can divide through by any nonzero element), where  $F^{\times} = F \setminus \{0\}$ . If  $F^{\times}$  is a nonabelian group, then F is called a skew field (slightly outdated) or a division algebra.

Examples of fields are  $\mathbb{Q}$  or  $\mathbb{R}$ , nonexamples are  $\mathbb{Z}$  or  $\mathbb{R}[X]$ .

### A.4 Vector Spaces

Let F be a field. A set V is called a vector space over F if (V, +) is an additive group (its elements are called vectors), and if there is a map  $F \times V \longrightarrow V$  called scalar multiplication such that for all  $\lambda, \mu \in F$  and all  $v, w \in V$  we have:

- 1v = v;
- $\lambda(v+w) = \lambda v + \lambda w;$
- $(\lambda + \mu)v = \lambda v + \mu v;$
- $\lambda(\mu v) = (\lambda \mu) v.$

Examples of vector spaces over  $\mathbb{F}$  are  $\mathbb{F}^n$  (vectors with *n* coordinates and entries in *F*), the set  $F[X]_n$  of polynomials of degree  $\leq n$  and coefficients in *F*, or the polynomial ring F[X].

### A.5 Algebras

An F-algebra A consists of an F-vector space A such that elements of A can be multiplied:

- $\lambda(ab) = (\lambda a)b$  for  $\lambda \in F$ ,  $a, b \in A$ ;
- (a+b)c = ac + bc for all  $a, b, c \in R$ ;

The algebra is called

- associative if (ab)c = a(bc) for all  $a, b, c \in A$ ;
- unitary if A has a multiplicative identity;
- commutative if ab = ba;
- (left) division algebra if every nonzero  $a \in A$  has a (left) multiplicative inverse.

Examples of algebras are the polynomial rings F[X], Hamilton's quaternions, or the matrix algebras  $M_n(F)$  over a field F (the set of  $n \times n$  matrices with entries in F).

# References

- J. Baylis, Error-Correcting Codes. A mathematical introduction, Chapman & Hall 1998
   A very nice book that I discovered only recently. Very detailed, many examples, and on an elementary level. Unfortunately, it doesn't even talk about finite fields except Z/pZ.
- 2. E. Berlekamp, *Algebraic Coding Theory*, New York 1968 A classic I haven't read yet. Berlekamp is a household name in computational algebra, mainly connected with polynomials and their factorization.
- 3. J. Bierbrauer, *Introduction to Coding Theory*, Lecture Notes 2000 An introduction for undergraduates from yet another guy from Heidelberg who ended up in the US.
- 4. I.F. Blake (ed.), Algebraic coding theory. History and development, 1973 A collection of reprints of original publications on coding theory, including the seminal papers by Hamming, Golay, Slepian, MacWilliams, Reed, Muller, ...
- 5. W. Ebeling, Lattices and Codes, Vieweg 1994 An excellent presentation of the connections between lattices and codes. For understanding the whole book, you will need a solid background in complex analysis and number theory. There will be a new edition going into print in May 2002. 95
- 6. G. van der Geer, J.H. van Lint, Introduction to Coding Theory and Algebraic Geometry, DMV Seminar 12, Basel 1988
  A concise introduction into algebraic geometry codes. We will cover part I, which is coding theory proper. Part II covers algebraic geometry codes.
- R. Hill, A first course in coding theory, Oxford 1986
   A very nice introduction with lots of examples; I put it on reserve in the library. 8
- 8. J.H. van Lint, Introduction to Coding Theory, 3rd ed. Springer Verlag 1999

One of the standard references for coding theory.

- 100 Franz Lemmermeyer
- J.H. van Lint, The Mathematics of the Compact Disc, Mitt. DMV 4 (1998), 25–29
   An article that appeared in the German equivalent of the Notices of the AMS.
- V. Pless, Introduction to the theory of error-correcting codes, Wiley 1982 A somewhat concise but valuable introduction; there exist newer editions. One of the few books that discuss automorphism groups of codes.
- 11. O. Pretzel, *Error-correcting codes and finite fields*, Oxford 1992 A detailed introduction with many examples. On reserve in the library.
- O. Pretzel, Codes and Algebraic Curves, Oxford 1998 Another book discussing the link between algebraic curves and coding theory. 88
- 13. C.A. Rogers, *Packing and Covering*, Cambridge Univ. Press 1964 My favorite introduction to packings; unfortunately out of print and hard to find.
- T.M. Thompson, From error-correcting codes through sphere packings to simple groups, Math. Assoc. America, 1983 THE history of error-correcting codes. 9
- 15. J. Walker, Codes and Curves, AMS 2000
  A very nice introduction to codes and curves, without the difficult proofs. If you like the online notes, buy the book — it's about \$ 20. 88