Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Lektor i realfag 8.-13.	Vårsemesteret, 2021 Åpen
Forfatter: Celina Santos Osmundsen	Celina Santos Osmundsen ..... (signatur forfatter)
Fagansvarlig: Veileder(e): David Ploog	
Tittel på masteroppgaven: Feilkorrigering av koder og Reed-Muller koder Engelsk tittel: Error correcting codes and Reed-Muller codes	
Studiepoeng: 30	
Emneord: Coding theory Hamming codes, Error correcting codes Reed-Muller Introduce code theory to pupils	Sidetall: 20 + vedlegg/annet: 1 Stavanger, 15.juni 2021 dato/år

Error-Correcting Codes and Reed-Muller codes

Celina Santos Osmundsen

A thesis presented for the degree of
Lektor i realfag 8.-13.

Universitetet
i Stavanger

Teknisk og Naturvitskapeleg Fakultet
Universitetet i Stavanger
Norge
15. juni 2021

Abstract

In this thesis, we will go through the basics of coding theory. By introducing the basic concepts of coding, we will look at different types of error-correcting codes. We will look into the properties of block codes and linear codes. The different families of codes have different properties but the same goal, to detect and correct errors. Matrices can be used to make good codes. Examples of good codes by matrices can be the Hamming code and the Reed-Muller. When we have been through all of the theory we see how it can be introduced in the classroom and linking it to the curriculum aims. The reason why I chose this thesis is because the prerequisites is close to where the mathematical level are for the oldest pupils I can teach when I finish my masters. For me, it was important that I could see the relevance from the thesis to my work afterwards. It was also interesting to see how and if I can use this in school later on.

Contents

Chapter 1

Coding theory

Coding theory is divided into several categories. One of these is channel coding. In this thesis, we will look into channel coding, more specifically, error-correcting codes.

In communication, we have information exchanged between two parties. In coding theory, these are called the sender or source who will send the information to a receiver. The sending is called the transmission and will happen through a medium such as a telephone, software or the internet.

The information is sent through a channel. The input alphabet X is sent through a medium, and the result is an output alphabet Y. We also look at the transition probability function P . When the message is sent, the probability for $p(y|x)$ means that for every x in X and y in Y the sent symbol x was transmitted, and the symbol y received.

When we send a codeword or symbol, we call it a transmission. Through the transmission of information, bits or parts of the message can be transmitted wrong. When the information is transmitted wrong, the information sent is not the same information as the one received. From this, we can conclude with noise in the channel. If the received word is the same as the one sent, there is no noise, and the channel is noiseless.

The Binary Symmetric Channel, or the BSC, is the channel where both X and Y is equal to $\{0, 1\}$ and the probability P is given by $p(0|1) = p(1|0) = p$ and $p(0|0) = p(1|1) = 1 - p$ for some $0 \leq p \leq 1$. If a symbol is received correctly, the probability is $1 - p$, if it is received incorrectly, the probability is p, if the symbol is received incorrectly, an error has occurred. If the probability of receiving the correct symbol is less than $\frac{1}{2}$, the channel will be more reliable.

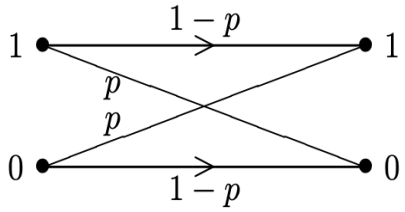

Illustration of the BSC

As mentioned earlier, in coding, there are alphabets. An alphabet is denoted as Q, and within each alphabet, there are q distinct symbols. If we have a binary code, the alphabet will contain 0 and 1, meaning the $q = 2$. If we say we send a bit, a bit corresponds to a bit. When we send messages with letters, this is called a string or bitstrings which corresponds to vectors.

Altogether, Shannon's results proves the existence of good codes, but the proof is probabilistic: it does not say anything about how to find the good codes. Also, we cannot let the block length go to infinity in practice. From this, it becomes a relevant problem to find practical, good codes.

1.1 Redundancy

Redundancy to a code is when the transmitted code has more symbols than necessary. The redundancy is to protect the original code better against errors in the channels. The redundancy is used as a reassurance to be sure that the information can be transmitted almost correctly and still be interpreted as the original information. Especially when the words are long, redundancy is used to prevent errors.

An example can be if one wants to transmit a 1 over the BSC giving it an error probability p. Instead of sending the symbol once, we can make a $k - fold$ repetition code where we send the symbol k-times instead of once. When the code is transmitted we can see from the symbols received which ever symbol occurs most is most likely the symbol we wanted to receive. In this case, we can see how the redundancy only is for reassurance. To find the redundancy, we know that it will be $0 < r < 1$. When we have a $k - fold$ repetition code, we see that the information rate is $rate = \frac{1}{k}$. If a code C is $\mathbb{F}_2^3$ we have length $n = 3$, and we are in the binary system. To find the rate of C, we need to: $rate(C) = \frac{log_2 M}{3}$ so we have the $rate(C) = \frac{log_2 2}{3}$, and from this we get $rate(C) = \frac{1}{3}$. We can choose to have a large information rate and low redundancy or a large redundancy and small information rate in coding theory.

1.2 Claude Shannon

Shannon's *noisy-channel coding theorem* states that arbitrarily good codes exist in a specific and technical sense. For simplicity, we assume a BSC. This means that only bits, 0 or 1, are sent and received and that the probability of an error is the same number $p < 0,5$. The *capacity* of such a channel is $C = 1 + p\log_2(p) + (1-p)\log_2(1-p)$. Then the theorem states that there exists a sequence of codes C_n of length n and information rate $rate < C$ such that the error probability converges to 0.

In practical terms, code words are n-bit strings such that $k < n$ are the information part and $n-k$ the redundancy, giving information rate $rate = k/n$. Shannon's theorem says that among binary codes with $rate = k/n < C$, there exist codes such that the probability of transmissions errors is low, i.e. that the code can correct many errors (has a large minimal distance, in the language introduced below).

1.3 Focus

The focus in this thesis is Reed-Muller codes. To understand how the Reed-Muller code is constructed and used, it will first go through some of the basics for the thesis by looking into fields, block codes and linear codes. From there we can go on to how to do calculations with the Reed-Muller code, and how to use the code to find codewords. This thesis will also look into some of the properties of the Reed-Muller code. Show that the Reed-Muller code is both linear and a repetition code. The thesis will also show how to find and calculate dual of the Reed-Muller code and its dimension. The thesis mainly follows the lecture notes given by Henk von Tilborg called *CODING THEORY, a first course.*

1.4 Motivation

The reason why I chose this thesis is because the prerequisites is close to where the mathematical level are for the oldest pupils I can teach when I finish my masters. For me, it was important that I could see the relevance from the thesis to my work afterwards. It was also interesting to see how and if I can use this in school later on.

Chapter 2

Basic notations

To understand the construction of the Reed-Muller code and the Reed-Muller codes properties, some basic concepts about codes and coding theory one should know. Some of the theory presented in this chapter is finding the minimum distance of a code, what the minimum distance tells us and the error correction capability. The construction of the Hamming Code and what the Hamming code is will also be introduced.

2.1 Finite fields $\mathbb{F}_p$

Finite fields are algebraic structures, important examples of finite fields are $\mathbb{Q}, \mathbb{R}, \mathbb{C}$. In these fields, we do all four operations of arithmetic: addition, subtraction, multiplication and division. When we work in these types of fields, all regular rules can be applied. We use finite fields to limit what we are working with, to get to what we would like. Finite fields are also used in data technology, where the structured data is finite, and memory is finite. Here we also limit the fields so that the computers or software can do what it is supposed to without doing it for eternity.

Most codes consist of vectors taken from a vector space over a finite field, and many codes are constructed by using algebraic methods. Finite fields of order q exist if and only if q is a prime power. An example of this can be $\mathbb{F}_{3^2}$, where $q = p^k$ is $q = 3^2$.

An example of what a finite field could be is $\mathbb{Z}_{11}$. The finite field $\mathbb{Z}_{11}$ is made up of the set

$$\mathbb{Z}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}.$$

The set could also be written as $\mathbb{Z}_{11} = \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$. The sets are the because of residue classes. A residue class is when a number is divided by a given modulo and give the same remainder. Each of the numbers in the set corresponds to a number in the second set. By adding or subtracting the number of the given modulo, we can find the number from the first set in the second set. The corresponding numbers from the sets will give the same answers by the same operations in mod 11.

If we use any of the operations on the set $\mathbb{Z}_{11}$ we have to remember that the answer we want must lie within the set we defined. We do this by adding or subtracting the number 11 until we get a number with the set. So for basic addition we see that it is easy to do the operations.

$$8 + 6 = 14 \equiv 3 \mod 11$$

Here we clearly see that the number is in our set.

If we take the numbers 8 and 6 and subtract 6 from 8 in $\mathbb{Z}_{11}$ we get:

$$8 - 6 = 2 \mod 11$$

However

$$6 - 8 = -2 \equiv 9 \mod 11$$

Here we also need to adjust the answer to be within the defined set.

Doing multiplication in $\mathbb{Z}_{11}$:

$$6 \cdot 8 = 48 \equiv 4 \mod 11$$

When we need to do division in a set, it is not straightforward as the operations previously mentioned. To do division in a set, we need to understand Euclid's Algorithm. Euclid's Algorithm: Let $m, n \in \mathbb{Z}$. Then there exist $a, b \in \mathbb{Z}$. Such that $am+bn = gcd(m, n)$, To apply the algorithm, we need to do it in steps. The first step is to find the $gcd(m, n)$ by continued division and remainders. $m = 231, n = 39$

We start by dividing m by n and finding the remainder of this.

$$231 = 5 \cdot 39 + 36$$

In the second line, we use n and divide it by the previous remainder.

$$39 = 1 \cdot 36 + 3$$

Lastly, we use the remainder of the first line and divide it by the remainder of the second line.

$$36 = 12 \cdot 3 + 0$$

We can see that the last non-zero remainder is 3, and 3 will therefore be the greatest common divisor of n and m, 231 and 39.

$$gcd(231, 39) = 3$$

The second step is to find the a and b in the expression $am + bn = gcd(m, n)$. $a \cdot 39 + b \cdot 231 = gcd(m, n) = 3$. To find the a and b, we need to go backwards from the continued division in part 1. From the first part, we write it as a linear combination of n *and* m.

We start by using the last line from the division, where we find the $gcd(m, n)$.

$$3 = 39 - 1 \cdot 36$$

By substituting the remainder for the starting expression in step 1 we get a new expression.

$$3 = 39 - (231 - 5 \cdot 39)$$

By ordering the expression we can get it in the form we want $am + bn = gcd(m, n)$

$$3 = 6 \cdot 39 - 1 \cdot 231$$

This gives us the a=-1, and $b = 6$

When we want to divide in our set $\mathbb{Z}_{11}$ we know that $m = p$, and $1 \leq n < p$. So we can have the following equation:

$$\frac{5}{8} \in \mathbb{Z}_{11}$$

To solve this we first have to find $\frac{1}{8} \in \mathbb{Z}_{11}$. Then we have to apply Euclids Algorithm. We set $p = 11$ and $n = 8$.

$$11 = 1 \cdot 8 + 3$$

$$8 = 2 \cdot 3 + 2$$

$$3 = 1 \cdot 2 + 1$$

$$2 = 2 \cdot 1 + 0$$

$$gcd(11, 8) = 1$$

Now that we have the $gcd(11, 8)$ we need to use the $ap+bn = gcd(p, n)$ and substitute backwards.

$$1 = 3 - 2$$

$$1 = 3 - (8 - 2 \cdot 3)$$

$$1 = 3 \cdot 3 - 1 \cdot 8$$

$$1 = 3(11 - 1 \cdot 8) - 1 \cdot 8$$

$$1 = 3 \cdot 11 - 4 \cdot 8$$

Here we have the equation equal to the expression $ap + bn = gcd(p, n)$ giving us the $a = 3$ and $b = -4$. Since there is a multiple of p in our expression we know this becomes 0 in $\mathbb{Z}_{11}$. We are left with $a \cdot 0 + bn = gcd(p, n)$, $bn = 1$

$$bn = 1 \mod 11$$

$$\frac{1}{n} = b \mod 11$$

We see that our $b = -4$, but in $\mod 11$ these becomes $b = 7$.

$$\frac{1}{8} = 7 \mod 11$$

We can now look at the problem we started with and replace our $\frac{1}{8}$ with 7.

$$\frac{5}{8} \mod 11$$

$$5 \cdot 7 = 35 \equiv 2 \mod 11$$

The general method to check if $\mathbb{F}_{p^k}$ is a field is to see that all prime powers

$$\mathbb{F}_{p^k} = \mathbb{F}_1[x]/|f(x)| \; with f(x) \in \mathbb{F}_p[x] \text{ irreducible polynomial with } \deg f(x) = k$$

By the following example we can see that $\mathbb{Z}_4$ is not a field, however $\mathbb{Z}_{2^2}$ is a field.

Example 1 $\mathbb{F}_4$

$$\mathbb{F}_4 = \mathbb{F}_2[x]/(x^2 + x + 1) = \{0, 1, x, x + 1\},$$

$x^2 + x + 1$ is irreducible over the field $\mathbb{F}_2$, but $x^2 + 1$ is in fact reducible over $\mathbb{F}_2$. This is because the function $x^2 + x + 1$ does not have any roots in $\mathbb{F}_2$ and the degree is 2. $x^2 + 1$ can be reduced into $(x + 1)(x + 1)$

When we have a field:

$$\mathbb{F}_{p^k} = \mathbb{F}_p[x] \; / \; (f(x))$$

with $deg(f) = k$, $f(x)$=irreducible polynomial.

We can construct $\mathbb{C}$ using the field $\mathbb{R}[x]$ and divide it by an irreducible polynomial in $R[x]$, but reducible in $\mathbb{C}$.

$$\mathbb{C} = \mathbb{R}[x]/(x^2 + 1)$$

$$x^2 + 1 \in \mathbb{R}[x] \text{ is irreducible.}$$

$\mathbb{C}$ is a quadratic field extension.

2.2 Codes

A code is a subset c of a larger set S. When we transmit a code $c \in C$ the receiver will receive a code $s \in S$ and try to decode the message to a codeword.

The number of codewords in a code is called the cardinality. The cardinality of a code depends on the alphabet of the code. If we have an $[n, k, d]$code, we can use the k and the q and calculate $q^k = M$, where M is the cardinality of the code. The cardinality can be written as $|C| = q^k = M$. If a code is said to have a cardinality equal to one, the code is trivial. So a trivial code must be non-zero and contain exactly one codeword, $C = \{c\}$.

2.3 Block codes

Block codes are codes where all codewords are of the same fixed length n. An example of a code that is not a block code is the Morse code, where the codewords are of varied lengths. Block codes are usually just called codes and will be addressed as this throughout the thesis. The codewords are represented by an n-dimensional vector over the finite field $\mathbb{F}_q^n$. The code C is a subset of the field $\mathbb{F}_q^n$.

An important property of a code is the Hamming distance. The Hamming distance tells us the number of coordinates two given vectors differ. If we have an alphabet Q and fixed length n

$$\mathbb{Q}^n$$

The Hamming distance is given by:

$$d(x, y) = |\{1 \leq i \leq n \mid x_i \neq y_i\}|$$

We get that the $d(x, x)$ is equal to zero because the two vectors are the same and for that reason, the distance will be 0. Also, $d(x, y) = d(y, x)$ because it is not relevant which one is the first vector when we look at places the two differs. From the properties we can introduce the triangle inequality

$$d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z \in \mathbb{Q}^n$$

From the triangle inequality we can see that the Hamming distance is a distance function.

We also have another important property called the minimum distance d of a non-trivial code C given by:

$$d = min\{d(x, y) \mid x \in C, y \in C, x \neq y\}$$

Both the Hamming distance and the minimum distance of a code C shows closeness between two codewords.

$$\underline{u} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$
$$\underline{v} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$d(\underline{u}, \underline{v}) = 2$, The vectors $\underline{u}$ and $\underline{v}$ differ in the first and third place giving the distance=2. Given a vector space, the distance will be the number of places, coordinates, the vectors differ. The smaller this number is, the closer the receiver is to get the original message. When we have the minimum distance of a code, it is simple to find the error-correcting capability e.

$$e = \left\lfloor \frac{d-1}{2} \right\rfloor$$

If we have a code C with the minimum distance d, we can detect $d - 1$ errors and correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors. If we have a code with a small distance, the correction and detection of an error is harder than with a large distance. If we have a larger distance, we can assume that the error we have is closer to one word than another in C. It is also easier to see where the error has occurred.

The Hamming bound is another way to approach the minimum distance. We think of the codewords as disjoint balls with radius = e. $B_r(x)$ is a ball of radius r around x, $\{y \in \mathbb{Q}^n \mid d(y, x) \leq$

$r\}$. The cardinality of $B_r(x)$ is found when we know all the words that are at distance i from x. From the n coordinates from x we choose exactly i, and for each of these we replace them with $q-1$ alphabet symbols. Then we have $\binom{n}{i}(q-1)^i$ words at the distance i from x giving us:

$$|B_r(x)| = \sum_{i=0}^{r} \binom{n}{i}(q-1)^i$$

Since we now know that balls around all the codewords in C are disjoint, this leaves q^n distinct words in $\mathbb{Q}$. So we can use the following theorem:

Let C be a code, then the covering radius ρ is given by

$$|C| \sum_{i=0}^{e} \binom{n}{i}(q-1)^i \leq q^n$$

Let the distance from codeword x to the code C be $d(x, C)$, $d(x, C) = min\{d(x, c) \mid c \in C\}$, so to find the furthest removed code we get it by the covering radius. The covering radius ρ a code C is given by

$$\rho = max\{d(x, y)|x \in \mathbb{Q}\}$$

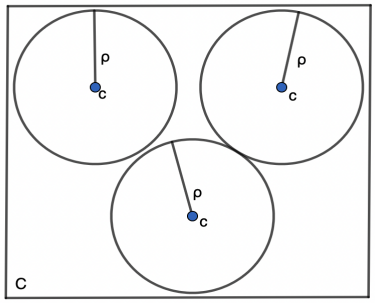

Illustration of the cover radius

A perfect code is an e-error-correcting code C with the covering radius $\rho = e$. This means every element in $\mathbb{Q}^n$ is at most e from a unique codeword.

2.4 Linear codes

A linear code is a code for which any linear combination of codewords is also a codeword. Linear code C of the fixed length n is any linear subspace V_n^q. From linear algebra, we have $V \subset \mathbb{F}_q^n$ is a subvector space. If $0 \in V$, and $v_1, v_2 \in V$ and by that we get $v_1 + v_2 \in V$

For a linear code the minimum distance of the code, C is equal to the minimum weight of all the non-zero code vectors. $d(x, y) = d(x - y, 0) = w(x - y)$

The dimension k of a code C is the part of the codeword that is not redundancy $k = n - r$.If a code C have a dimension k and minimum distance d, we call it an $[n, k, d]$code. We can also have a q-ary (n, M, d) code C, where the cardinality is M. For $[n, k, d]$ code C we know it is linear, and the cardinality is q^k.

Good linear codes can be made by matrices.

A generator matrix G of an [n,k,d]code C is a $k \times n$ matrix. The basis of C is made up by the k rows. The rows of G generate C.

$$C = \{\underline{a}G \mid \underline{a} \in V_k(q)\}$$

$$[n, 1, n]$$

The generator matrix of the q-ary [n,k,d] repetition code is given by.

$$G = (1\ 1\ 1\ \cdots\ 1)$$

If we have an even weight linear code we can use the parameters [n,n-1,2] so this generator is given by a matrix.

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & 1 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & 1 & \cdots & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}$$

If we have a code C the first $k \times k$ matrix is the identity matrix. What is left after is the redundancy, which makes the error-correcting possible. So we can say that the first part of the code is the I_k which is the identity matrix, and the second part is P. Giving us $G = (I_k\ P)$.

$$[6,3,3] = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Here it is clear that the 3×3 matrix in the front is the identity matrix and the rest is the redundancy.

We also have a parity check matrix H of an [n,k,d] code C is an $(n-k) \times n$ matrix.

$$\underline{c} \in C \Leftrightarrow H\underline{c}^T = \underline{0}^T$$

This means that C is the solution space of $n-k$ linearly independent equations $H\underline{x}^T = 0^T$. So when we have the generator matrix G we can take $H = (-P^T\ \ I_{n-k})$ and then we have the parity check matrix.

Let $(\underline{x},\underline{y})$ is the inner product of the $\sum_{i=1}^{n} x_i y_i$ in the vector space $V_n(q)$. We say that the vectors are orthogonal when the inner product is zero. However, we can have vectors that are orthogonal to themselves without being the zero-vector. To check if a product is orthogonal, we use the dot product. If we take the vector $\underline{v} = (1\ 0\ 2\ 1)$ in $V_4(3)$. To check if it is orthogonal with itself.

$$\underline{v} \cdot \underline{v} = (1\ 0\ 2\ 1) \cdot (1\ 0\ 2\ 1)$$

$$= (1 \cdot 1) + (0 \cdot 0) + (2 \cdot 2) + (1 \cdot 1) = 0$$

Another example in $\mathbb{F}_2$ where the vector $\underline{u} = (1,1)$ is orthogonal to itself.

The [n,k,d] code C also have a dual code $C^\perp$ is defined as:

$$C^\perp = \{\underline{x} \in V_n(q) \mid (\underline{x}, \underline{c} = 0 \text{ for all } \underline{c} \in C\}$$

The code $C^\perp$ is a linear subspace of the dimension $n-k$, so the $C^\perp = [n, n-k, d^\perp]$ code. We can check that the $C^\perp$ is the generator matrix of the parity check matrix H and that the parity check matrix of $C^\perp$ is the generator matrix G of C, by using $GH^T = 0$. If a code is in both C and $C^\perp$ the code is self-orthogonal.

2.5 Hamming codes

$$\underline{c} = C \Leftrightarrow H\underline{c}^T = \underline{0}^T$$

If a linear code C has word $\underline{c}$ if and only if coordinates of $\underline{c}$ give a dependency relation between the columns of the parity check matrix H of C. The d of the Hamming code is $d \geq 2$ iff the parity check matrix do not contain the all-zero column. If the parity check matrix contains two columns that are

linearly dependent, we have $d \geq 3$. The distance of the Hamming code C is $\geq d$ iff there are $d-1$ linearly dependent columns in the parity check matrix.

The q-ary Hamming code of length $n = (q^r - 1)/(q-1)$ and the r is defined by the parity check matrix that consists of all the pairwise linearly independent columns. [7,4,3]

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

These columns in the matrix give us the binary representation of the numbers 1 up to 7.

An example where we can use the $\underline{c} = C \Leftrightarrow H\underline{c}^T = \underline{0}^T$

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

Here it is easy to see that this has the identity matrix in front. So to make the parity check matrix we transpose the part that is not the identity matrix, and set the identity matrix, but we need to change the identity to be a 4×4 identity matrix instead to get the right dimension.

$$H = [-P^T \ I_{n-k}] = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Choose an arbitrary codeword from the code C and transpose it.

$$\underline{c}^T = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$H\underline{c}^T = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

For all non-zero codewords $H\underline{c}^T = \underline{0}^T$ holds.

Chapter 3

Reed-Muller codes

Reed-Muller is a class of linear codes. It dates back to 1954, when D. E. Muller wrote a paper about the code. Within the same year, I. S. Reed found a decoding algorithm for the code. The Reed-Muller codes are one of the oldest and simplest family of codes. It has been used in computer science and electrical engineering.

When we work with Reed-Muller we only consider binary polynomials with squarefree monomials,

$$f = f(x_1, x_2, x_3, ..., x_m)$$

where m is some fixed integer. The Reed-Muller code can also be denoted as a binary field with length 2^m, $\mathbb{F}_2^m$. In $\mathbb{F}_2$: we get that $x^2 = x$, because we have $0^2 = 0$ and $1^2 = 1$ we could have an example of two polynomials that is two different functions, but in $\mathbb{F}_2$ these two are the same function. $f(x, y, z) = g(x, y, z)$

Example of when this happens can be:

$$f(x, y, z) = x^2y + yz + z^3$$

and

$$g(x, y, z) = xy + xz + z$$

Although they are different polynomials, they are the same function. A polynomial contains several monomials, meaning power products, but in Reed-Muller codes, they are all without any squares. The polynomial can have several terms, but none of the terms will have any powers larger than 1 nor coefficients other than 0 or 1, however, the degree of the polynomial can be up to m.

$$monomial = x_1x_2...x_m$$

Here we have a monomial, a term made up of several variables multiplied together.

$$polynomial = 2{x_1}^3{x_4}^2x_5 + {x_2}^2x_3 + 3$$

Here we have a polynomial containing several terms made up of several variables multiplied together in several terms.

Definition 1 *The r-th order Reed-Muller code* $\mathcal{RM}(m, m)$ *of length* $n = 2^n$

$$\mathcal{RM}(r, m) = \{(f(\underline{u}_0), f(\underline{u}_2), ..., f(\underline{u}_{n-1})) \mid degree(f) \leq r\}$$

The degree of a Reed-Muller code is decided by the degree of an element in $f(x_1, x_2, ..., x_m)$ of $\mathcal{RM}(r, m)$. The degree of the $\mathcal{RM}(r, m)$ is from the degree of f as a polynomial in m variables. If f contains a term which is a product of r variables, but no term that is a product of $\leq r+1$ variables, f is said to have *degree* r at most. The degree of f is clear to see when $1 \leq r \leq m$. We can not

have monomials with squares, and therefore, we cannot get a larger degree than r. The maximum degree of polynomials without squares in m is $x_1x_2...x_m$

The Reed-Muller code $\mathcal{RM}(2,3)$ will at most have the degree 2. This is because of the binary polynomial with squarefree monomials. The function will only have binary coefficient and exponents. When we have $r = 2$ the degree cannot be more than 2, simply because we can not make a term with a higher degree than two, when the function is binary and, the r is 2.

3.1 Properties of Reed-Muller codes

To find the number of codewords in a Reed-Muller code, we need to know what the m is. When we have the m the number of codewords is:

$$\mathbb{F}_2^m = 2^m$$

To find the dimension of any Reed-Muller code, we need to add all the binomial coefficients of the $\mathcal{RM}(r,m)$ code.

A polynomial $f(x_1,\ldots,x_m)$ in m variables without squares of degree r is a sum of r monomials with all exponents 1. There are $\binom{0,m}{+}\binom{1,m}{)} + \ldots + \binom{r,m}{}$ of those.

By setting $n = 5$ we start by finding the values we get for each i, when $m = 5$. We find the number of each

$$\binom{5}{0} = 1$$

$$\binom{5}{1} = 5$$

$$\binom{5}{2} = 10$$

$$\binom{5}{3} = 10$$

$$\binom{5}{4} = 5$$

$$\binom{5}{5} = 1$$

When adding them together we get

$$dim\mathcal{RM}(5,5) = 1+5+5+10+10+1 = 32 = 2^5 = 2^m$$

. We can also obtain the dimension of $\mathcal{RM}(3,5)$ by adding the binomial coefficients from $r = 0$ up until $r = 3$. So $dim\mathcal{RM}(3,5) = 1+5+10+10 = 26$ 2^m is the number of all subsets of a set with m elements, for example the set $x_1, x_2, ..., x_m$ $\mathcal{RM}(rm)$. If all the codewords are 0, the subset is empty.

We can count the number of binary polynomials with squarefree monomials by using the binomial coefficient. When

$$\sum_{l=0}^{m} \binom{m}{l} = 2^m$$

The number of squarefree binary polynomials in $x_1, x_2, .., x_m$ with degree $\leq r$ is given by

$$2^{1+\binom{m}{1}+\binom{m}{2}+\ldots+\binom{m}{r}} = 2^{\sum_{i=1}^{r}\binom{m}{i}} = 2^{2^m} = 2^n$$

We can conclude that the Reed-Muller codes include each vector in V_n. So there are exactly 2^n vectors in V_n. We have already seen how to get the $n = 2^m$. To get the number of all polynomials we need to use the dimension of the $\mathcal{RM}(r,m)$ code, and use it as an exponent of 2 to get the number of polynomials.

3.2 Reed-Muller code is a linear repetition code

To prove that the Reed-Muller code is in fact a linear code we need to check:

$$c_1 = f \in \mathbb{F}_2[x_1, x_2, ..., x_m],$$

and

$$c_2 = g \in \mathbb{F}_2[x_1, x_2, ..., x_m]$$

where deg f $\leq$ r, and deg g $\leq$ r

$$c_1 + c_2 = f + q$$

Both c_1 and c_2 are given by polynomials. We know that the degree of f and the degree of g are less than r+1 because they are contained in the $\mathcal{RM}(r, m)$. For the code to be linear, we want $c_1 + c_2 = f + g$ to also be in $\mathcal{RM}$. All the binary polynomials with squarefree monomials in the m variables $x_1, ..., x_m$, therefore the $deq(f + g) \leq max(deg(f), deg(g) \leq r$. When we get the sum of the degree of the polynomials, it does not go up, but it can go down.

The linear subspace of $\mathbb{F}_2$ which has a basis of vectors $(\underline{u_0}, \underline{u_1}, ..., \underline{u_m})$. We know from the definition of *the* $r - th$ *order*$\mathcal{RM}(r, m)$ that the $\mathcal{RM}(0, m)$ is the repetitive code of length $n = 2^m$. From the definition, we can confirm that the Reed-Muller code is a linear repetition code.

3.3 Examples of Reed-Muller codes

An example of a Reed-Muller code $\mathcal{RM}(r, 3)$. We first need to understand how to construct it. We start by taking a function $f(x_1, x_2, x_3)$

When working with $\mathcal{RM}$, its possible to write the codes in lexicographical order or reverse lexicographical order, these codes will, however be in lexicographical order. That is so the most significant coordinate will be the first.

To show how to construct the $\mathcal{RM}$(1,3), $\mathcal{RM}$(2,3), $\mathcal{RM}$(3,3).

To compute the three Reed-Muller given above, we first have to find the length.

$$n = \mathbb{F}_2^3 = 2^3$$

We start by setting the vectors in order where they are presented in the binary form from 0 to $m - 1$.

$$\begin{pmatrix} 0 & 0 & 0 \end{pmatrix} = \underline{u}_0$$
$$\begin{pmatrix} 0 & 0 & 1 \end{pmatrix} = \underline{u}_1$$
$$\begin{pmatrix} 0 & 1 & 0 \end{pmatrix} = \underline{u}_2$$
$$\begin{pmatrix} 0 & 1 & 1 \end{pmatrix} = \underline{u}_3$$
$$\begin{pmatrix} 1 & 0 & 0 \end{pmatrix} = \underline{u}_4$$
$$\begin{pmatrix} 1 & 0 & 1 \end{pmatrix} = \underline{u}_5$$
$$\begin{pmatrix} 1 & 1 & 0 \end{pmatrix} = \underline{u}_6$$
$$\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \underline{u}_7$$

We change the rows into columns and add the repetition vector 1 of length 2^3.

$$\underline{v}_0 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$
$$\underline{v}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$
$$\underline{v}_2 = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$
$$\underline{v}_3 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

We then, by setting the vectors together, get the new matrix. This matrix is the $\mathcal{RM}(1,3)$. This matrix includes the linear terms of the code.

$$\mathcal{RM}(1,3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

When we have the $\mathcal{RM}(1,3)$ to get the next rows of the matrix, we have to make the terms of second degree. We need to add $\underline{v}_1$ with $\underline{v}_2$, $\underline{v}_1$ with $\underline{v}_3$, and $\underline{v}_2$ with $\underline{v}_3$

$$\mathcal{RM}(2,3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

So to get the full $\mathcal{RM}(1,3)$ matrix, we lastly have to add up the three vector $\underline{v}_1, \underline{v}_2, \underline{v}_3$.

$$\mathcal{RM}(3,3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Another way to show the codewords is by the a table. Here is the $\mathcal{RM}(4,4)$:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_1x_2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
x_1x_3	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
x_1x_4	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
x_2x_3	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
x_2x_4	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
x_3x_4	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
$x_1x_2x_3$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
$x_1x_2x_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
$x_1x_3x_4$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
$x_2x_3x_4$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
$x_1x_2x_3x_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table of $\mathcal{RM}(4,4)$

When we have this table we can also try finding other codewords like $(x_1)(x_2-1)(x_3-1)(x_4-1)$. Where you have to change all the variables, here the codeword is:

$$(x_1)(x_2-1)(x_3-1)(x_4-1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3.4 Proof of minimum distance of Reed-Muller code

To find the minimum distance of a Reed-Muller code, we first need to understand the construction of $(\underline{u}, \underline{u}+\underline{v})$.

If we construct a new code by using C_1 a binary $[n, k_1, d_1]$ code and C_2 another binary $[n, k_2, d_2]$ code, we can define a code C

$$C = \{(\underline{u}, \underline{u}+\underline{v}) \mid \underline{u}\ in\ C_1, \underline{v}\ in\ C_2\}$$

By constructing C we have the parameters $[2n, k_1+k_2, d]$.

The length of the code is easy to see, the length of the C_1 first and then the same length as C_1+C_2, which is the same length, which we can see from how the codes are defined, with the same n. Therefore by adding them after each other, we get $n+n=2n$.
We get the cardinality of C by adding k_1+k_2 because these are the powers of the alphabet. When we find the cardinality of c_1 we use the k_1 as a power of the alphabet q, and the for c_2 we use k_2. The cardinality for the code C is found by multiplying the $q^{k_1} with q^{k_2}, this gives q^{k_1+k_2}$. The cardinality is k_1+k_2.

The distance, however is the minimum distance of $2d_1$ and d_2, $\min\{2d_1, d_2\}$. To show this we need to use $(\underline{u_1}, \underline{u_1}+\underline{v_1})$ and $(\underline{u_2}, \underline{u_2}+\underline{v_2})$, two codewords in C. By first setting $\underline{v_1}=\underline{v_2}$, we get that the minimum distance between the codewords would be d_1 both in the first part and the second part of the code. Because it happens twice, we need to double the distance. Therefore we get $2d_1$. We get the second distance when we set $\underline{v_1} \neq \underline{v_2}$. There are three different cases we need to take into count. The first one is where $\underline{u_1}$ and $\underline{u_2}$ cancel each other out, so we are only left with $\underline{v_1}$ and $\underline{v_2}$. In this case, we get the same distance as in C_2, which is d_2. We could also get that $\underline{u_1}$ and $\underline{u_2}$ differ, or $\underline{u_1}+\underline{v_1}$ and $\underline{u_2}+\underline{v_2}$ differ. The minimum distance for $(\underline{u_1}, \underline{u_1}+\underline{v_1})$ and $(\underline{u_2}, \underline{u_2}+\underline{v_2})$ is at least d_2.

Therefore the minimum distance of C=min$\{\ 2d_1, d_2\ \}$

To find the minimum distance of a Reed-Muller code with the parameters $[n, \sum_{l=0}^{r}\binom{m}{l}, 2^{m-r}]$, with the length $n=2^{m-r}$.

If we have a polynomial in $f(x_1, x_2, ..., x_m)$ in $\mathcal{RM}(r, m)$ we split the terms into terms containing the x_1 and those which do not. So we are left with $f(x_1, x_2, ..., x_m) = p(x_2, x_3, ..., x_m) + x_1 q(x_2, x_3, ..., x_m)$. $p(x_2, x_3, ..., x_m)$ is in $\mathcal{RM}(r, m-1)$, the number of variables go down by 1, simply because we remove the x_1. WE also have one less variable for the q we get $\mathcal{RM}(r-1, m-1)$. The degree of the polynomial $p(x_1, x_2, ..., x_m)$ can at most be r, because the term with the highest degree in f can be in p as well. However, when we have q we exclude x_1 from our function. So if a term in f has the highest degree of the function, it will have a lower degree in q because we factorise out the $x_1$term. To find the minimum distance of p, we need to use the standard formula 2^{m-r}, but we have $m-1$ number of variables so the minimum distance of p=2^m-1-r. To find the minimum distance for q, we use the same formula, but we need to subtract 1 from both m and r, leaving us with. $2^{(m-1)-(r-1)} = 2^{m-r}$. From here, we have the same rule as the one for constructed codes. $\min\{\ 2d_p, d_q\ \}$. We see that:

$$min\{2d_p, d_q\} = min\{2 \cdot 2^{m-1-r}, 2^{m-r}\} = min\{2^{m-r}, 2^{m-r}\} = 2^{m-r}$$

From notes: For m=1, this is trivial.

$$\mathcal{RM}(0,1) = \{11, 00\},\ d = 2^{m-r} = 2^{1-0} = 2$$

$$\mathcal{RM}(1,1) = \{11, 01, 10, 00\},\ d = 2^{m-r} = 2^{1-1} = 1$$

Example 2 *Start by setting m=4, and using the polynomial with squarefree monomials.*

$$f(x_1, x_2, x_3, x_4) = 1 + x_1 + x_2x_4 + x_1x_3x_4 + x_2x_3$$

By grouping all terms into terms including x_1 and those who don't include x_1 we get:

$$1 + x_2x_4 + x_2x_3 + x_1(1 + x_3x_4)$$

We can now split up the polynomial into two new functions all the terms not containing x_1 is set in p, and then the terms including x_1 is set in q. So p will now be a function of x_2, x_3, x_4 and $x_1 q$ is a function q of x_2, x_3, x_4. This gives us:

$$f(x_1, x_2, x_3, x_4) = p(x_2, x_3, x_4) + x_1 q(1 + x_3, x_4)$$

The minimum distance of p $\mathcal{RM}(r, m-1)$, where we know the degree can at most be r, we get the following:

$$minimum\ distance\ of\ p = 2^{(m-1)-(r-1)} = 2^{m-r}$$

The minimum distance of q $\mathcal{RM}(r-1, m-1)$, where we know the degree cannot be r, we get:

$$minimum\ distance\ of\ q = 2^{(m-1)-(r-1)}$$

So $f(x_1, x_2, x_3, x_4) = p(x_2, x_3, x_4) + x_1 q(x_2, x_3, x_4)$
From $\mathcal{RM}(1, 4)$

$$\mathcal{RM}(1,4) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

When we use $f(x_1, x_2, x_3, x_4)$ *we get that*

$$p = 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1$$

$$q = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0$$

$$f = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0$$

$$d_1 = 2^{m-r-1}$$

,

$$d_2 = 2^{m-r}$$

$$d = (2d_1, d_2) = (2 \cdot 2^{m-r-1}, 2^{m-r}) = 2^{m-r}$$

3.5 Dual of Reed-Muller code

To find the dual of a Reed-Muller code, we use the dimension of the $\mathcal{RM}(r, m)$ code. As well as the length n. To find the dimension of the field, we use the same formula as for the [n,k,d] but change the upper limit to m so that we include the entire field.

$$n = dim\ \mathbb{F}_2^n = \sum_{i=0}^{n} \binom{n}{i}$$

$$\mathcal{RM}(r, m) = C \in \mathbb{F}_2^n$$

$$k = dim\ C = \sum_{l=0}^{r} \binom{m}{l}$$

$$\mathcal{RM}$$

$$dim\ C^{\perp} = n - k$$

To find the dimension of $C^{\perp}$ we can use the two sums we found and subtract the k from the n, similar to how we find the dimension of a regular linear code.

$$dim\ C^{\perp} = \sum_{l=0}^{m} \binom{m}{l} - \sum_{l=0}^{r} \binom{m}{l}$$

We change $l = 0$ to $l = r + 1$, when we do this we can use the same expression of the sum.

$$= \sum_{l=r+1}^{m} \binom{m}{l}$$

$$= \sum_{l'=0}^{m-(r+1)} \binom{m}{l + (r+1)}$$

By substituting the l with l' we can see how the dim $C^{\perp}$ is in the same form as the dimension of C.

$$l = l' + r + 1$$

$$l' = l - (r+1)$$

$$\sum_{l=0}^{m-r-1} \binom{m}{l + r + 1}$$

Each element f in $\mathcal{RM}(m-r-1, m)$ is orthogonal to each element g in$\mathcal{RM}(r, m)$. If we multiply f and g the product will be in $\mathcal{RM}(m-1, m)$. Since $m-1$ is less than m, $\mathcal{RM}(m-1, m)$ is an even weight code. This implies that the functions are orthogonal to each other. $C^{\perp} = C$ only happens when the $k = \frac{n}{2}$ and is then called a self dual.

Chapter 4

Introducing coding theory in school

In 2020 there was a new curriculum for all pupils in the Norwegian school system. The renewal of the curriculum takes a step back from specific goals and gives more room for the pupils to be engaged in their own learning. We have looked into algebra during this thesis, which occurs several places in *Kunnskapsløftet 2020*, or *LK20*. The theory from the thesis can be incorporated in several of the different grade level's curricula.

The necessity of the pupils understanding the terminology of coding theory can be debatable. The pupils know how to work with the operations in mathematics without knowing the mathematical concept behind it, such as working with the clock. Pupils can use modular arithmetic and work with the clock in modulo 12 and 24 without knowing anything about sets, groups or fields.

The 8^{th}-grade curriculum includes several curriculum aims regarding algebra. From the 8^{th} grade curriculum, we can take a closer look at the following curriculum aim.

Explore algebraic calculation rules

This curriculum aim gives the teacher an advantage in working towards this aim because it does not state nor specify an approach. The teacher can choose what and how to work with the curriculum aim. It is also an underlying goal in *Kunnskapsløftet 2020* that the pupils will learn to be critical thinkers and think in a more scientific and exploring manner. Here it is possible to introduce modular arithmetic. The pupils can learn what a set is and how to understand sets, as well as finite fields.

An obstacle with fields in the 8^{th}grade is that exponents are new to the pupils. They are not familiar with working with exponents. The pupils will most likely be able to do the operations need to work with fields and sets, but some pupils would see the working with codes in a theoretical form to be too challenging. If the pupils would work with the fields and sets on paper, the student with little motivation would most likely end up doing nothing. If a teacher chooses to introduce coding theory and coding to a class in the 8^{th} grade, there are several tools one can use. To introduce coding in a more practical way the teacher can bring in ASCII. With ASCII tables the teacher can give the students challenge and introduce games. The pupils can then start coding and decoding without knowing the coding theory. This can also encourage the pupils to learn more about coding without seeing it as working with math in the traditional way. Now that most schools use computers or iPads in the classroom there are several apps and websites where they can start coding without knowing its concepts. If pupils use apps to code, the pupils can also see when or if they do a mistake almost immediately and fix it, instead of not seeing the mistake on paper and feeling defeated when they have to start over with the calculations.

Bibliography

- Utdanningsdirektoratet (2020) *Kompetansemål og vurdering* available from: https://www.udir.no/lk20/mat01-05/kompetansemaal-og-vurdering/kv16?lang=nob (downloaded June 2021)
- Utdanningdirektoratet (2020) *Kompetansemål og vurdering* available from:https://www.udir.no/lk20/mat01-05/kompetansemaal-og-vurdering/kv14?lang=nob (downloaded June 2021)
- Utdanningdirektoratet (2020) *Opplæringens verdigrunnlag* available from:https://www.udir.no/lk20/overordnet-del/opplaringens-verdigrunnlag/ (downloaded June 2021)
- van Lint, J. H. (1973) *Lecture Notes in Mathematics*. Published: Berlin, Springer
- van Tilborg, H. C. A (1993) *Coding Theory - A first course* Available from:http://docplayer.net/7371482-Coding-theory-a-first-course-henk-c-a-van-tilborg.html (downloaded: January 2021)