Part II

Linear codes

CHAPTER 2: LINEAR CODES

WHY LINEAR CODES

Most of the important codes are special types of so-called linear codes. Linear codes are of very large importance because they have:
very concise description,
very nice properties,
very easy encoding
and, in general,
an easy to describe decoding.

Many practically important linear codes have also an efficient decoding.

GALOI FIELDS \( GF(q) \) – where \( q \) is a prime.

It is the set \( \{0, 1, \ldots, q - 1\} \) with two operations:

- addition modulo \( q \) — \( + \mod q \)
- multiplication modulo \( q \) — \( \times \mod q \)

Example — \( GF(3) \)

\[
2 + 2 = 1 \\
2 \times 2 = 1
\]

Example — \( GF(7) \)

\[
5 + 5 = 3 \\
5 \times 5 = 4
\]

Example — \( GF(11) \)

\[
7 + 8 = 4 \\
7 \times 8 = 1
\]

Comment. To design linear codes also Galoi fields \( GF(p^k) \), \( k > 1 \) can be used, but their structure and operations are defined in a more complex way, see the Appendix.
EXERCISE

Which of the following binary codes are linear?
- \( C_1 = \{00, 01, 10, 11\} \)
- \( C_2 = \{000, 011, 101, 110\} \)
- \( C_3 = \{00000, 01101, 10110, 11011\} \)
- \( C_4 = \{101, 111, 011\} \)
- \( C_5 = \{000, 001, 010, 011\} \)
- \( C_6 = \{0000, 1001, 1101, 1110\} \)

How to create a linear code?

**Notation?** If \( S \) is a set of vectors of a vector space, then let \( \langle S \rangle \) be the set of all linear combinations of vectors from \( S \).

**Theorem** For any subset \( S \) of a linear space, \( \langle S \rangle \) is a linear space that consists of the following words:
- the zero word,
- all words in \( S \),
- all sums of two or more words in \( S \).

**Example**

\[ S = \{0100, 0011, 1100\} \]

\[ \langle S \rangle = \{0000, 0100, 0011, 1100, 0111, 1011, 1000, 1111\} \]

BASIC PROPERTIES of LINEAR CODES II

If \( C \) is a linear \([n,k]\)-code, then it has a basis \( \Gamma \) consisting of \( k \) codewords and each codeword of \( C \) is a linear combination of the codewords from its basis \( \Gamma \).

**Example**

Code

\[ C_4 = \{0000000, 1111111, 1000101, 1100010, 0110001, 1011000, 0101100, 0010110, 0001110, 0111010, 1001110, 0100111, 1010011, 1101001, 1110100\} \]

has, as one of its bases, the basis

\[ \{1111111, 1000101, 1100010, 0110001\} \]

How many different bases has a linear code?

**Theorem** A binary linear code of dimension \( k \) has

\[ \frac{1}{k!} \prod_{i=0}^{k-1} (2^k - 2^i) \]

bases.

BASIC PROPERTIES of LINEAR CODES I

**Notation:** Let \( w(x) \) (weight of \( x \)) to denote the number of non-zero entries of \( x \).

**Lemma** If \( x, y \in F_2^n \), then \( h(x, y) = w(x - y) \).

**Proof** \( x - y \) has non-zero entries in exactly those positions where \( x \) and \( y \) differ.

**Theorem** Let \( C \) be a linear code and let weight of \( C \), notation \( w(C) \), be the smallest of the weights of non-zero codewords of \( C \). Then \( h(C) = w(C) \).

**Proof** There are \( x, y \in C \) such that \( h(C) = h(x, y) \). Hence \( h(C) = w(x - y) \geq w(C) \).

On the other hand, for some \( x \in C \)

\[ w(C) = w(x) = h(x, 0) \geq h(C) \]

**Consequence**

- If \( C \) is a code with \( m \) codewords and it is not linear, then in order to determine \( h(C) \) one has to make in general \( \binom{m}{2} = \Theta(m^2) \) comparisons in the worst case.
- If \( C \) is a code with \( m \) codewords, then in order to compute \( h(C) \), \( m - 1 \) comparisons are enough.

ADVANTAGES and DISADVANTAGES of LINEAR CODES I.

**Advantages** - big.

- Minimal distance \( h(C) \) is easy to compute if \( C \) is a linear code.
- Linear codes have simple specifications.
- To specify a non-linear code usually all codewords have to be listed.
- To specify a linear \([n,k]\)-code it is enough to list \( k \) codewords (of a basis).

**Definition** A \( k \times n \) matrix whose rows form a basis of a linear \([n,k]\)-code (subspace) \( C \) is said to be the generator matrix of \( C \).

**Example** One of the generator matrices of the binary code

\[ C_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \]

is the matrix \( \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \)

and one of the generator matrices of the code

\[ C_4 \text{ is } \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \]

- There are simple encoding/decoding procedures for linear codes.
Disadvantages of linear codes are small:

1. Linear $q$-codes are not defined unless $q$ is a power of a prime.
2. The restriction to linear codes might be a restriction to weaker codes than sometimes desired.
UNIQUENESS of ENCODING

with linear codes

**Theorem** If \( G = \{ w_i \}_{i=1}^k \) is a generator matrix of a binary linear code \( C \) of length \( n \) and dimension \( k \), then the set of codewords/vectors

\[
v = uG
\]

ranges over all \( 2^k \) codewords of \( C \) as \( u \) ranges over all \( 2^k \) datawords of length \( k \). Therefore

\[
C = \{ uG \mid u \in \{0,1\}^k \}
\]

Moreover

\[
u_1 G = u_2 G
\]

if and only if

\[
u_1 = u_2.
\]

**Proof** If \( u_1 G - u_2 G = 0 \), then

\[
0 = \sum_{i=1}^k u_{1,i} w_i - \sum_{i=1}^k u_{2,i} w_i = \sum_{i=1}^k (u_{1,i} - u_{2,i}) w_i
\]

And, therefore, since \( w_i \) are linearly independent, \( u_1 = u_2 \).

LINEAR CODES as SYSTEMATIC CODES

Since to each linear \([n, k]\)-code \( C \) there is a generator matrix of the form \( G = [I_k | A] \) encoding of a dataword \( w \) with \( G \) has the form

\[
wG = w \cdot wA
\]

and therefore with such an encoding we can see the code \( C \) as being a systematic code.

Each linear code can therefore be seen, at a proper encoding, as being systematic code.

DECODING of LINEAR CODES - BASICS

Decoding problem: If a codeword: \( x = x_1 \ldots x_n \) is sent and the word \( y = y_1 \ldots y_n \) is received, then \( e = y - x = e_1 \ldots e_n \) is said to be the error vector. The decoder must decide, from \( y \), which \( x \) was sent, or, equivalently, which error \( e \) occurred.

To describe main Decoding method some technicalities have to be introduced.

**Definition** Suppose \( C \) is an \([n, k]\)-code over \( F_q^n \) and \( u \in F_q^k \). Then the set

\[
u + C = \{ u + x \mid x \in C \}
\]

is called a coset (\( u \)-coset) of \( C \) in \( F_q^n \).

**Example** Let \( C = \{0000, 1011, 0101, 1110\} \).

**Cosets:**

- \( 0000 + C = C \),
- \( 1000 + C = \{1000, 0011, 1101, 0110\} \),
- \( 0100 + C = \{0100, 1111, 0001, 1010\} = 0001 + C \),
- \( 0010 + C = \{0010, 1001, 0111, 1100\} \).

Are there some other cosets in this case?

**Theorem** Suppose \( C \) is a linear \([n, k]\)-code over \( F_q^n \). Then

\( a) \) every vector of \( F_q^n \) is in some coset of \( C \),
\( b) \) every coset contains exactly \( q^k \) elements,
\( c) \) two cosets are either disjoint or identical.

DECODING of LINEAR CODES - TECHNICALITIES

Decoding problem: If a codeword: \( x = x_1 \ldots x_n \) is sent and the word \( y = y_1 \ldots y_n \) is received, then \( e = y - x = e_1 \ldots e_n \) is said to be the error vector. The decoder must decide, from \( y \), which \( x \) was sent, or, equivalently, which error \( e \) occurred.

To describe main Decoding method some technicalities have to be introduced.

**Definition** Suppose \( C \) is an \([n, k]\)-code over \( F_q^n \) and \( u \in F_q^k \). Then the set

\[
u + C = \{ u + x \mid x \in C \}
\]

is called a coset (\( u \)-coset) of \( C \) in \( F_q^n \).

**Example** Let \( C = \{0000, 1011, 0101, 1110\} \).

**Cosets:**

- \( 0000 + C = C \),
- \( 1000 + C = \{1000, 0011, 1101, 0110\} \),
- \( 0100 + C = \{0100, 1111, 0001, 1010\} = 0001 + C \),
- \( 0010 + C = \{0010, 1001, 0111, 1100\} \).

Are there some other cosets in this case?

**Theorem** Suppose \( C \) is a linear \([n, k]\)-code over \( F_q^n \). Then

\( a) \) every vector of \( F_q^n \) is in some coset of \( C \),
\( b) \) every coset contains exactly \( q^k \) elements,
\( c) \) two cosets are either disjoint or identical.
2. Linear codes

Each vector having minimum weight in a coset is called a coset leader.

1. Design a (Slepian) standard array for an \([n, k]\)-code \(C\) that is a \(q^{n-k} \times q^k\) array of the form:

<table>
<thead>
<tr>
<th>codewords</th>
<th>coset leader</th>
<th>codeword 2</th>
<th>...</th>
<th>codeword 2^*</th>
</tr>
</thead>
<tbody>
<tr>
<td>coset leader</td>
<td>+</td>
<td>...</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>coset leader</td>
<td>+</td>
<td>...</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Example

<table>
<thead>
<tr>
<th>0000</th>
<th>1011</th>
<th>0101</th>
<th>1110</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0011</td>
<td>1101</td>
<td>0110</td>
</tr>
<tr>
<td>0100</td>
<td>1111</td>
<td>0001</td>
<td>1010</td>
</tr>
<tr>
<td>0010</td>
<td>1001</td>
<td>0111</td>
<td>1100</td>
</tr>
</tbody>
</table>

A word \(y\) is decoded as codeword of the first row of the column in which \(y\) occurs. Error vectors which will be corrected are precisely coset leaders!

In practice, this decoding method is too slow and requires too much memory.

Suppose a binary linear code is used only for error detection.

The decoder will fail to detect errors which have occurred if the received word \(y\) is a codeword different from the codeword \(x\) which was sent, i.e. if the error vector \(e = y \oplus x\) is itself a non-zero codeword.

The probability \(P_{\text{undetect}}(C)\) that an incorrect codeword is received is given by the following result.

**Theorem** Let \(C\) be a binary \([n, k]\)-code, and let \(A_i\) denote the number of codewords of \(C\) of weight \(i\). Then, if \(C\) is used for error detection, the probability of an incorrect message being received is

\[
P_{\text{undetect}}(C) = \sum_{i=0}^{n} A_i p^i (1 - p)^{n-i}.
\]

**Example** In the case of the \([4,2]\) code from the last example

\[
A_1 = 1, A_2 = 2
\]

\[
P_{\text{undetect}}(C) = p^2 (1 - p)^2 + 2p^3 (1 - p) = p^2 - p^4.
\]

For \(p = 0.01\)

\[
P_{\text{undetect}}(C) = 0.00009999.
\]

### PROBABILITY of GOOD ERROR CORRECTION

What is the probability that a received word will be decoded correctly - that is as the codeword that was sent (for binary linear codes and binary symmetric channel)?

Probability of an error in the case of a given error vector of weight \(i\) is

\[
p(1 - p)^{n-i}.
\]

Therefore, it holds.

**Theorem** Let \(C\) be a binary \([n, k]\)-code, and for \(i = 0, 1, \ldots, n\) let \(\alpha_i\) be the number of coset leaders of weight \(i\). The probability \(P_{\text{corr}}(C)\) that a received vector when decoded by means of a standard array is the codeword which was sent is given by

\[
P_{\text{corr}}(C) = \sum_{i=0}^{n} \alpha_i p^i (1 - p)^{n-i}.
\]

**Example** For the \([4,2]\)-code of the last example

\[
\alpha_0 = 1, \alpha_1 = 3, \alpha_2 = \alpha_3 = \alpha_4 = 0.
\]

Hence

\[
P_{\text{corr}}(C) = (1 - p)^4 + 3p(1 - p)^3 = (1 - p)^3 (1 + 2p).
\]

If \(p = 0.01\), then \(P_{\text{corr}} = 0.9897\)

### DUAL CODES

**Inner product** of two vectors (words)

\[
u = u_1 \ldots u_n, \quad v = v_1 \ldots v_n
\]

in \(F_q^n\) is an element of \(GF(q)\) defined (using modulo \(q\) operations) by

\[
u \cdot v = u_1 v_1 + \ldots + u_n v_n.
\]

**Example**

\[
\begin{align*}
\text{In } F_4^1 & : 1001 \cdot 1001 = 0 \\
\text{In } F_4^2 & : 2001 \cdot 1210 = 2 \\
& 1212 \cdot 2121 = 2
\end{align*}
\]

If \(u \cdot v = 0\) then words (vectors) \(u\) and \(v\) are called orthogonal.

**Properties**

If \(u, v, w \in F_q^n, \lambda, \mu \in GF(q)\), then

\[
u \cdot v = v \cdot u, (\lambda u + \mu v) \cdot w = \lambda (u \cdot w) + \mu (v \cdot w).
\]

Given a linear \([n, k]\)-code \(C\), then the dual code of \(C\), denoted by \(C^\perp\), is defined by

\[
C^\perp = \{ v \in F_q^n | \forall u \in C \implies u \cdot v = 0 \}
\]

**Lemma** Suppose \(C\) is an \([n, k]\)-code having a generator matrix \(G\). Then for \(v \in F_q^n\)

\[
\forall v \in C^\perp \iff v G^\top = 0,
\]

where \(G^\top\) denotes the transpose of the matrix \(G\).
For understanding of the role the parity checks play for linear codes, it is important to understand relation between orthogonality and special parity checks.

If binary words \( x \) and \( y \) are orthogonal, then the word \( y \) has even number of ones (1's) in the positions determined by ones (1's) in the word \( x \).

This implies that if words \( x \) and \( y \) are orthogonal, then \( x \) is a parity check word for \( y \) and \( y \) is a parity check word for \( x \).

Exercise: Let the word 100001 be orthogonal to a set \( S \) of binary words of length 6. What can we say about the words in \( S \)?

**Example**

For the \([n, 1]\)-repetition code \( C \), with the generator matrix

\[
G = (1, 1, \ldots, 1)
\]

the dual code \( C^\perp \) is \([n, n - 1]\)-code with the generator matrix \( G^\perp \), described by

\[
G^\perp = \begin{pmatrix}
1 & 1 & 0 & 0 & \ldots & 0 \\
1 & 0 & 1 & 0 & \ldots & 0 \\
\vdots & & & & \ddots & \vdots \\
1 & 0 & 0 & 0 & \ldots & 1
\end{pmatrix}
\]

**Parity Check Matrices I**

Example If

\[
C_5 = \begin{pmatrix}
0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1
\end{pmatrix}
\]

then \( C_5^\perp = C_5 \).

If

\[
C_6 = \begin{pmatrix}
0 & 0 & 0 \\
1 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1
\end{pmatrix}
\]

then \( C_6^\perp = \begin{pmatrix}
0 & 0 & 0 \\
1 & 1 & 1
\end{pmatrix} \).

Theorem Suppose \( C \) is a linear \([n, k]\)-code over \( F_q^n \), then the dual code \( C^\perp \) is a linear \([n, n - k]\)-code.

**Definition** A parity-check matrix \( H \) for an \([n, k]\)-code \( C \) is a generator matrix of \( C^\perp \).

Example Parity-check matrix for

\[
C_5 = \begin{pmatrix}
0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1
\end{pmatrix}
\]

is \( \begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1
\end{pmatrix} \) and for

\[
C_6 \text{ is } \begin{pmatrix}
1 & 1 & 1
\end{pmatrix}
\]

The rows of a parity check matrix are parity checks on codewords. They actually say that certain linear combinations of elements of every codeword are zeros modulo 2.
**HAMMING CODES**

An important family of simple linear codes that are easy to encode and decode, are so-called Hamming codes.

**Definition** Let \( r \) be an integer and \( H \) be an \( r \times (2^r - 1) \) matrix columns of which are all non-zero distinct words from \( F_2^r \). The code having \( H \) as its parity-check matrix is called binary Hamming code and denoted by \( \text{Ham}(r, 2) \).

**Example**

\[
\text{Ham}(2, 2) : H = \begin{bmatrix}
1 & 1 & 0 \\
1 & 0 & 1
\end{bmatrix} \Rightarrow G = \begin{bmatrix}
1 & 1 & 1 \\
1 & 0 & 1
\end{bmatrix}
\]

\[
\text{Ham}(3, 2) = H = \begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0
\end{bmatrix} \Rightarrow G = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

**Theorem** Hamming code \( \text{Ham}(r, 2) \)
- is \( [2^r - 1, 2^r - r - 1] \)-code,
- has minimum distance 3,
- and is a perfect code.

**Properties of binary Hamming codes**
- Coset leaders are precisely words of weight \( \leq 1 \).
- The syndrome of the word \( 0 \ldots 010 \ldots 0 \) with 1 in \( j \)-th position and 0 otherwise is the transpose of the \( j \)-th column of \( H \).

**Decoding algorithm** for the case the columns of \( H \) are arranged in the order of increasing binary numbers the columns represent.

- **Step 1** Given \( y \) compute syndrome \( S(y) = yH^\top \).
- **Step 2** If \( S(y) = 0 \), then \( y \) is assumed to be the codeword sent.
- **Step 3** If \( S(y) \neq 0 \), then assuming a single error, \( S(y) \) gives the binary position of the error.
EXAMPLE

For the Hamming code given by the parity-check matrix

\[
H = \begin{bmatrix}
0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

and the received word

\[ y = 1101011, \]

we get syndrome

\[ S(y) = 110 \]

and therefore the error is in the sixth position.

Hamming code was discovered by Hamming (1950), Golay (1950).

It was conjectured for some time that Hamming codes and two so-called Golay codes are the only non-trivial perfect codes.

Comment

Hamming codes were originally used to deal with errors in long-distance telephone calls.

ADVANTAGES of HAMMING CODES

Let a binary symmetric channel be used which with probability \( q \) correctly transfers a binary symbol.

If a 4-bit message is transmitted through such a channel, then correct transmission of the message occurs with probability \( q^4 \).

If Hamming \((7, 4, 3)\) code is used to transmit a 4-bit message, then probability of correct decoding is

\[ q^4 + 7(1-q)q^3. \]

In case \( q = 0.9 \) the probability of correct transmission is 0.6561 in the case no error correction is used and 0.8503 in the case Hamming code is used - an essential improvement.

IMPORTANT CODES

- Hamming \((7, 4, 3)\)-code. It has 16 codewords of length 7. It can be used to send \( 2^7 = 128 \) messages and can be used to correct 1 error.
- Golay \((23, 12, 7)\)-code. It has 4 096 codewords. It can be used to transmit 8 388 608 messages and can correct 3 errors.
- Quadratic residue \((47, 24, 11)\)-code. It has 16 777 216 codewords and can be used to transmit 140 737 488 355 238 messages and correct 5 errors.

Hamming and Golay codes are the only non-trivial perfect codes. They are also special cases of quadratic residue codes.

GOLAY CODES - DESCRIPTION

Golay codes \( G_{24} \) and \( G_{23} \) were used by Voyager I and Voyager II to transmit color pictures of Jupiter and Saturn. Generation matrix for \( G_{24} \) has the following simple form

\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\( G_{24} \) is \((24, 12, 8)\)-code and the weights of all codewords are multiples of 4. \( G_{23} \) is obtained from \( G_{24} \) by deleting last symbols of each codeword of \( G_{24} \). \( G_{23} \) is \((23, 12, 7)\)-code.
Matrix $G$ for Golay code $G_{24}$ has actually a simple and regular construction.

The first 12 columns are formed by a unitary matrix $I_{12}$, next column has all 1’s.

Rows of the last 11 columns are cyclic permutations of the first row which has 1 at those positions that are squares modulo 11, that is $0, 1, 3, 4, 5, 9$.

Maximum length code is $[2^m - 1, m, 2^{m-1}]$-code with the generator matrix whose columns are all binary representations of numbers from 1 to $2^m = n$.

Hadamard code $HC_{2n}$ is the code with generator matrices defined recursively as

$$M_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

and

$$M_{2n} = \begin{bmatrix} M_n & M_n \\ \bar{M}_n & M_n \end{bmatrix}$$

where $\bar{M}_n$ is the complementary matrix to $M_n$ (with 0 and 1 interchanged).

Hadamard code

$$M_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

This is an infinite, recursively defined, family of so called $RM_{r,m}$ binary linear $[2^m, k, 2^{m-r}]$-codes with

$$k = 1 + \binom{m}{1} + \ldots + \binom{m}{r}.$$}

The generator matrix $G_{r,m}$ for $RM_{r,m}$ code has the form

$$G_{r,m} = \begin{bmatrix} G_{r-1,m} \\ Q_r \end{bmatrix}$$

where $Q_r$ is a matrix with dimension $\binom{m}{r} \times 2^m$ where

- $G_{0,m}$ is a row vector of the length $2^m$ with all elements 1.
- $G_{1,m}$ is obtained from $G_{0,m}$ by adding columns that are binary representations of the column numbers.
- matrix $Q_r$ is obtained by considering all combinations of $r$ rows of $G_{1,m}$ and by obtaining products of these rows/vectors, component by component. The result of each of such a multiplication constitutes a row of $Q_r$. 

REED-MULLER CODES

RM_{r,m}$ is obtained from $G_{1,m}$ by adding columns that are binary representations of the column numbers.
**EXAMPLE**

**REED-MULLER CODES II**

Reed-Muller codes form a family of codes defined recursively with interesting properties and easy decoding.

If \( D_1 \) is a binary \([n, k_1, d_1]\)-code and \( D_2 \) is a binary \([n, k_2, d_2]\)-code, a binary code \( C \) of length \( 2n \) is defined as follows \( C = \{ u | u + v \text{, where } u \in D_1, v \in D_2 \} \).

Lemma \( C \) is \([2n, k_1 + k_2, \min\{2d_1, d_2\}]\)-code and if \( G_i \) is a generator matrix for \( D_i \), \( i = 1, 2 \), then \( \begin{bmatrix} G_1 & G_2 \\ 0 & 0 \end{bmatrix} \) is a generator matrix for \( C \).

Reed-Muller codes \( R(r, m) \), with \( 0 \leq r \leq m \) are binary codes of length \( n = 2^m \cdot R(m, m) \) is the whole set of words of length \( n \), \( R(0, m) \) is the repetition code.

If \( 0 < r < m \), then \( R(r + 1, m + 1) \) is obtained from codes \( R(r + 1, m) \) and \( R(r, m) \) by the above construction.

Theorem The dimension of \( R(r, m) \) equals \( 1 + \binom{m}{r} + \ldots + \binom{m}{r} \). The minimum weight of \( R(r, m) \) equals \( 2^{m - r} \). Codes \( R(m - r - 1, m) \) and \( R(r, m) \) are dual codes.

---

**SINGLETON and PLOTKIN BOUNDS**

To determine distance of a linear code can be computationally hard task. For that reason various bounds on distance can be much useful.

**Singleton bound:** If \( C \) is a \( q \)-ary \([n, M, d]\)-code.

\[ M \leq q^{n - d + 1}. \]

**Proof** Take some \( d - 1 \) coordinates and project all codewords to the resulting coordinates.

The resulting codewords are all different and therefore \( M \) cannot be larger than the number of \( q \)-ary words of length \( n - d + 1 \).

Codes for which \( M = q^{n - d + 1} \) are called **MDS-codes** (Maximum Distance Separable).

**Corollary:** If \( C \) is a binary linear \([n, k, d]\)-code, then

\[ d \leq n - k + 1. \]

So called **Plotkin bound** says

\[ d \leq \frac{n2^{k-1}}{2^k - 1}. \]

Plotkin bound implies that error-correcting codes with \( d \geq n(1 - 1/q) \) have only polynomially many codewords and hence are not very interesting.

---

**SHORTENING and PUNCTURING of LINEAR CODES**

If \( C \) is a \( q \)-ary linear \([n, k, d]\)-code, then

\( D = \{(x_1, \ldots, x_{n-1})|(x_1, \ldots, x_{n-1}, 0) \in C\} \) is a linear code - a shortening of the code \( C \).

If \( d > 1 \), then \( D \) is a linear \([n - 1, k, d']\)-code or \([n - 1, k, d - 1]\)-code a shortening of the code \( C \).

**Corollary:** If there is a \( q \)-ary \([n, k, d]\)-code, then shortening yields a \( q \)-ary \([n - 1, k - 1, d']\)-code.

If \( C \) is a \( q \)-ary \([n, k, d]\)-code and

\[ E = \{(x_1, \ldots, x_{n-1})|(x_1, \ldots, x_{n-1}, x) \in C, \text{ for some } x \leq q\}, \]

then \( E \) is a linear code - a puncturing of the code \( C \).

If \( d > 1 \), then \( E \) is an \([n - 1, k, d^*] \) code where \( d^* = d - 1 \) if \( C \) has a minimum weight codeword with wit non-zero last coordinate and \( d^* = d \) otherwise.

When \( d = 1 \), then \( E \) is an \([n - 1, k, 1]\) code, if \( C \) has no codeword of weight 1 whose nonzero entry is in last coordinate; otherwise, if \( k > 1 \), then \( E \) is an \([n - 1, k - 1, d^*]\) code with \( d^* > 1 \).
REED-SOLOMON CODES

An important example of MDS-codes are q-ary Reed-Solomon codes $\text{RSC}(k, q)$, for $k \leq q$.

They are codes generator matrix of which has rows labelled by polynomials $X^i$, $0 \leq i \leq k - 1$, columns by elements $0, 1, \ldots, q - 1$ and the element in a row labelled by a polynomial $p$ and in a column labelled by an element $u$ is $p(u)$.

$\text{RSC}(k, q)$ code is $[q, k, q - k + 1]$ code.

**Example** Generator matrix for $\text{RSC}(3, 5)$ code is

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 2 & 3 & 4 \\
0 & 1 & 4 & 4 & 1
\end{bmatrix}
\]

**Interesting property of Reed-Solomon codes:**

$$\text{RSC}(k, q) \perp = \text{RSC}(q - k, q).$$

Reed-Solomon codes are used in digital television, satellite communication, wireless communication, barcodes, compact discs, DVD, ... They are very good to correct burst errors - such as ones caused by solar energy.

SOCCER GAMES BETTING SYSTEM

Ternary Golay code with parameters $(11, 729, 5)$ can be used to bet for results of 11 soccer games with potential outcomes 1 (if home team wins), 2 (if guests win) and 3 (in case of a draw).

If 729 bets are made, then at least one bet has at least 9 results correctly guessed.

In case one has to bet for 13 games, then one can usually have two games with pretty sure outcomes and for the rest one can use the above ternary Golay code.

APPENDIX

**LDPC (Low-Density Parity Check) - CODES**

A LDPC code is a binary linear code whose parity check matrix is very sparse - it contains only very few 1’s.

A linear $[n, k]$ code is a regular $[n, k, r, c]$ LDPC code if $r << n, c << n - k$ and its parity-check matrix has exactly $r$ 1’s in each row and exactly $c$ 1’s in each column.

In the last years LDPC codes are replacing in many important applications other types of codes for the following reasons:

- LDPC codes are in principle also very good channel codes, so called Shannon capacity approaching codes, they allow the noise threshold to be set arbitrarily close to the theoretical maximum - to Shannon limit - for symmetric channel.
- Good LDPC codes can be decoded in time linear to their block length using special (for example "iterative belief propagation") approximation techniques.
- Some LDPC codes are well suited for implementations that make heavy use of parallelism.
- LDPC codes were first developed by Robert R. Gallager in 1963, but considered as impractical at that time. They were rediscovered in 1996.

Parity-check matrices for LDPC codes are often (pseudo)-randomly generated, subject to sparsity constrains. Such LDPC codes are proven to be good with a high probability.
LDPC codes were discovered in 1960 by R.C. Gallager in his PhD thesis, but ignored till 1996 when linear time decoding methods were discovered for some of them.

LDPC codes are used for: deep space communication; digital video broadcasting; 10GBase-T Ethernet, which sends data at 10 gigabits per second over Twisted-pair cables; Wi-Fi standard,....

An \([n, k]\) LDPC code can be represented by a bipartite graph between a set of \(n\) top "variable-nodes (v-nodes)" and a set of bottom \((n-k)\) "parity check nodes (c-nodes)".

The corresponding parity check matrix has \(n-k\) rows and \(n\) columns and \(i\)-th column has 1 in the \(j\)-th row exactly in case if \(i\)-th v-node is connected to \(j\)-th c-node.

\[
H = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0
\end{pmatrix}
\]

The LDPC-code with the Tanner graph

\[
\begin{array}{cccccc}
a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\
\oplus & \oplus & \oplus & \oplus & \oplus & \oplus
\end{array}
\]

has the parity check matrix

\[
H = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0
\end{pmatrix}
\]

and the following constrains have to be satisfied:

\[
\begin{align*}
a_1 + a_2 + a_3 + a_4 &= 0 \\
a_3 + a_4 + a_6 &= 0 \\
a_1 + a_4 + a_5 &= 0
\end{align*}
\]

In recent year has been interesting competition between LDPC codes and Turbo codes introduced in Chapter 3 for various applications.

In 2003, an LDPC code beat six turbo codes to become the error correcting code in the new DVB-S2 standard for satellite transmission for digital television.

LDPC is also used for 10Gbase-T Ethernet, which sends data at 10 gigabits per second over twisted-pair cables.

Since 2009 LDPC codes are also part of of the Wi-Fi 802.11 standard as an optional part of 802.11n, in the High Throughput PHY specification.