Part I

Linear, Cyclic, stream and channel codes. Speccial decodings

LINEAR CODES II. - continuation, decoding

APPENDIX I.

WISDOM

WISDOM

When you have eliminated impossible,

WISDOM

When you have eliminated impossible,

whatever remains,

WISDOM

When you have eliminated impossible,

whatever remains,
however impossible

When you have eliminated impossible,

whatever remains,
however impossible

must be

When you have eliminated impossible,

whatever remains,
however impossible

must be the TRUTH

WHY LINEAR CODES?

Most of the important codes are special types of so-called 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

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,

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,

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

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.II.

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.II.

Many practically important linear codes have also an efficient decoding.

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\bmod q}$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{mod } q}$ or $+_q$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{mod } q}$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\bmod q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\bmod q}$ or $\times_q$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\bmod q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\bmod q}$ or $\times_q$ or very simply $\times$ or $\cdot$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{ mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{ mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 =$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{ mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{ mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 =$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\bmod q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\bmod q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 =$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 = 3$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 = 3 \quad 5 \times_7 5$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\bmod q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\bmod q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 = 3 \quad 5 \times_7 5 = 4$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\bmod q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\bmod q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 = 3 \quad 5 \times_7 5 = 4$

Example — $GF(11)$

$7 +_{11} 8 =$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\bmod q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\bmod q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 = 3 \quad 5 \times_7 5 = 4$

Example — $GF(11)$

$7 +_{11} 8 = 4$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 = 3 \quad 5 \times_7 5 = 4$

Example — $GF(11)$

$7 +_{11} 8 = 4 \quad 7 \times_{11} 8 =$

MATHEMATICS BEHIND - GALOIS FIELDS $GF(q)$ – with q a prime.

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

addition modulo q — $+_{\text{ mod } q}$ or $+_q$ or very simply $+$
multiplication modulo q — $\times_{\text{ mod } q}$ or $\times_q$ or very simply $\times$ or $\cdot$

Example — $GF(3)$

$2 +_3 2 = 1 \quad 2 \times_3 2 = 1$

Example — $GF(7)$

$5 +_7 5 = 3 \quad 5 \times_7 5 = 4$

Example — $GF(11)$

$7 +_{11} 8 = 4 \quad 7 \times_{11} 8 = 1$

Comment. To design linear codes we will use Galois fields $GF(q)$ with q being a prime. One can also use Galois fields $GF(q^k)$, $k > 1$, but their structure and operations are defined in a more complex way, see the Appendix.

REPETITIONS - I.

Given an alphabet Σ, any set $C \subset \Sigma^*$ is called a **code** and its elements are called **codewords**.

Given an alphabet Σ, any set $C \subset \Sigma^*$ is called a **code** and its elements are called **codewords**.

By a **coding/encoding** of elements (messages) from a set M by codewords from a code C we understand any one-to-one mapping (encoder) e such that

$$e : M \to C$$

.

Given an alphabet Σ, any set $C \subset \Sigma^*$ is called a **code** and its elements are called **codewords**.

By a **coding/encoding** of elements (messages) from a set M by codewords from a code C we understand any one-to-one mapping (encoder) e such that

$$e : M \to C$$

.

Encoding (code) is called systematic if for any $m \in M \subset \Sigma^*$

$$e(m) = mc_m \quad \text{for some} \quad c_m \in \Sigma^*$$

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.

REPETITIONS - II.

1. A code C is said to be an **(n, M, d) code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.
   - Large M;

REPETITIONS - II.

1. A code C is said to be an **(n, M, d) code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.
   - Large M;
   - Large d;

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.
   - Large M;
   - Large d;
   - Encoding should be fast;

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.
   - Large M;
   - Large d;
   - Encoding should be fast; decoding reasonably efficient

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.
   - Large M;
   - Large d;
   - Encoding should be fast; decoding reasonably efficient
   - Encodings of similar messages should be very different.

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.
   - Large M;
   - Large d;
   - Encoding should be fast; decoding reasonably efficient
   - Encodings of similar messages should be very different.
   - Error corrections potential should be large.

REPETITIONS - II.

1. A code C is said to be an (n, M, d) **code**, if
   - n is the length of codewords in C
   - M is the number of codewords in C
   - d is the minimal distance of C
2. **A good code for encoding a set of messages should have**:
   - Small n.
   - Large M;
   - Large d;
   - Encoding should be fast; decoding reasonably efficient
   - Encodings of similar messages should be very different.
   - Error corrections potential should be large.

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.
Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.

Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2 \ldots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n))$

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.

Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2. \ldots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n))$
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.
Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2. \ldots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n))$
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$
   if $u = (u_1, u_2, \ldots, u_n),$, then $au = (au_1, au_2, \ldots, au_n))$

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.
Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2 \ldots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n))$
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$
   if $u = (u_1, u_2, \ldots, u_n),$, then $au = (au_1, au_2, \ldots, au_n))$

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.

Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \dots, u_n)$, $v = (v_1, v_2 \dots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \dots, u_n +_q v_n))$
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$
   if $u = (u_1, u_2, \dots, u_n)$,, then $au = (au_1, au_2, \dots, au_n))$

Lemma A subset $C \subseteq F_q^n$ is a linear code iff one of the following conditions is satisfied

1. C is a subspace of F_q^n.
2. Sum of any two codewords from C is in C (for the case $q = 2$)

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.

Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2. \ldots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n))$
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$
   if $u = (u_1, u_2, \ldots, u_n),$, then $au = (au_1, au_2, \ldots, au_n))$

Lemma A subset $C \subseteq F_q^n$ is a linear code iff one of the following conditions is satisfied

1. C is a subspace of F_q^n.
2. Sum of any two codewords from C is in C (for the case $q = 2$)

If C is a k-dimensional subspace of F_q^n, then C is called $[n, k]$-code.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.
Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2 \ldots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n)$)
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$
   if $u = (u_1, u_2, \ldots, u_n),$, then $au = (au_1, au_2, \ldots, au_n)$)

Lemma A subset $C \subseteq F_q^n$ is a linear code iff one of the following conditions is satisfied

1. C is a subspace of F_q^n.
2. Sum of any two codewords from C is in C (for the case $q = 2$)

If C is a k-dimensional subspace of F_q^n, then C is called $[n, k]$-code. It has q^k codewords.

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.

Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
   (if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2 \ldots, v_n)$ then
   $u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n)$)
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$
   if $u = (u_1, u_2, \ldots, u_n),$, then $au = (au_1, au_2, \ldots, au_n)$)

Lemma A subset $C \subseteq F_q^n$ is a linear code iff one of the following conditions is satisfied

1. C is a subspace of F_q^n.
2. Sum of any two codewords from C is in C (for the case $q = 2$)

If C is a k-dimensional subspace of F_q^n, then C is called $[n, k]$-code. It has q^k codewords. If the minimal distance of C is d, then it is said to be the $[n, k, d]$ code.

LINEAR CODES - continuation.

Linear codes are special sets of words of a fixed length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a (power of) prime.

Definition A subset $C \subseteq F_q^n$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
(if $u = (u_1, u_2, \ldots, u_n), v = (v_1, v_2 \ldots, v_n)$ then
$u + v = (u_1 +_q v_1, u_2 +_q v_2 \ldots, u_n +_q v_n))$
2. $au \in C$ for all $u \in C$, and all $a \in GF(q)$
if $u = (u_1, u_2, \ldots, u_n)$,, then $au = (au_1, au_2, \ldots, au_n))$

Lemma A subset $C \subseteq F_q^n$ is a linear code iff one of the following conditions is satisfied

1. C is a subspace of F_q^n.
2. Sum of any two codewords from C is in C (for the case $q = 2$)

If C is a k-dimensional subspace of F_q^n, then C is called $[n, k]$-code. It has q^k codewords. If the minimal distance of C is d, then it is said to be the $[n, k, d]$ code.

LINEAR CODES - continuation II.

If C is a linear $[n, k]$ code, then it has several bases.

If C is a linear $[n, k]$ code, then it has several bases.

A base **B** of C is such a sets of k codewods of C that each codeword of C is a linear combination of the codewords from the base **B**.

If C is a linear $[n, k]$ code, then it has several bases.

A base **B** of C is such a sets of k codewods of C that each codeword of C is a linear combination of the codewords from the base **B**.

Each base **B** of C is usually reperesented by a (k, n) matrix, $G_{\mathbf{B}}$, so called a **generator matrix of C**, the i-th row of which is the i-th codeword of **B**.

EXERCISE

EXERCISE

Which of the following binary codes are linear?

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES

$C_2 = \{000, 011, 101, 110\}$

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES

$C_2 = \{000, 011, 101, 110\}$ – YES

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES

$C_2 = \{000, 011, 101, 110\}$ – YES

$C_3 = \{00000, 01101, 10110, 11011\}$ – YES

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES

$C_2 = \{000, 011, 101, 110\}$ – YES

$C_3 = \{00000, 01101, 10110, 11011\}$ – YES

$C_5 = \{101, 111, 011\}$ – NO

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

How to create a linear code?

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

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.

EXCERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

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:

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

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,

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

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,

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

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.

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

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

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

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$ – YES
$C_2 = \{000, 011, 101, 110\}$ – YES
$C_3 = \{00000, 01101, 10110, 11011\}$ – YES
$C_5 = \{101, 111, 011\}$ – NO
$C_6 = \{000, 001, 010, 011\}$ – YES
$C_7 = \{0000, 1001, 0110, 1110\}$ – NO

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 I

BASIC PROPERTIES of LINEAR CODES I

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

BASIC PROPERTIES of LINEAR CODES I

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

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

BASIC PROPERTIES of LINEAR CODES I

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

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

Proof $x - y$ has non-zero entries in exactly those positions where x and y differ.

BASIC PROPERTIES of LINEAR CODES I

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

Lemma If $x, y \in F_q^n$, 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 $h(C) = w(C)$.

BASIC PROPERTIES of LINEAR CODES I

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

Lemma If $x, y \in F_q^n$, 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 $x \in C$

BASIC PROPERTIES of LINEAR CODES I

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

Lemma If $x, y \in F_q^n$, 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 $x \in C$

$$w(C) = w(x) = h(x,0) \geq h(C).$$

BASIC PROPERTIES of LINEAR CODES I

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

Lemma If $x, y \in F_q^n$, 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 non-linear code with m codewords, then in order to determine $h(C)$ one has to make in general $\binom{m}{2} = \Theta(m^2)$ comparisons in the worst case.

BASIC PROPERTIES of LINEAR CODES I

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

Lemma If $x, y \in F_q^n$, 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 non-linear code with m codewords, 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 linear code with m codewords, then in order to determine $h(C), m - 1$ comparisons are enough.**

BASIC PROPERTIES of LINEAR CODES I

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

Lemma If $x, y \in F_q^n$, 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 non-linear code with m codewords, 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 linear code with m codewords, then in order to determine $h(C), m-1$ comparisons are enough.**

EQUIVALENCE of LINEAR CODES I

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a permutation of the words or positions of the code;

(b multiplication of symbols appearing in a fixed position by a non-zero scalar.

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a permutation of the words or positions of the code;

(b multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over F_q^n if one matrix can be obtained from the other by a sequence of the following operations:

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over F_q^n if one matrix can be obtained from the other by a sequence of the following operations:

(a) permutation of the rows

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over F_q^n if one matrix can be obtained from the other by a sequence of the following operations:

(a) permutation of the rows

(b) multiplication of a row by a non-zero scalar

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over F_q^n if one matrix can be obtained from the other by a sequence of the following operations:

(a) permutation of the rows

(b) multiplication of a row by a non-zero scalar

(c) addition of one row to another

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

- (a) permutation of the words or positions of the code;
- (b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over F_q^n if one matrix can be obtained from the other by a sequence of the following operations:

- (a) permutation of the rows
- (b) multiplication of a row by a non-zero scalar
- (c) addition of one row to another
- (d) permutation of columns

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over F_q^n if one matrix can be obtained from the other by a sequence of the following operations:

(a) permutation of the rows

(b) multiplication of a row by a non-zero scalar

(c) addition of one row to another

(d) permutation of columns

(e) multiplication of a column by a non-zero scalar

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over F_q^n if one matrix can be obtained from the other by a sequence of the following operations:

(a) permutation of the rows

(b) multiplication of a row by a non-zero scalar

(c) addition of one row to another

(d) permutation of columns

(e) multiplication of a column by a non-zero scalar

Proof Operations (a) - (c) just replace one basis by another. Last two operations convert a generator matrix to one of an equivalent code.

EQUIVALENCE of LINEAR CODES II

Theorem Let G be a generator matrix of an $[n, k]$-code. Rows of G are then linearly independent .By operations (a) - (e) the matrix G can be transformed into the form: $[I_k|A]$ where I_k is the $k \times k$ identity matrix, and A is a $k \times (n-k)$ matrix.

EQUIVALENCE of LINEAR CODES II

Theorem Let G be a generator matrix of an $[n, k]$-code. Rows of G are then linearly independent .By operations (a) - (e) the matrix G can be transformed into the form: $[I_k|A]$ where I_k is the $k \times k$ identity matrix, and A is a $k \times (n-k)$ matrix.

Example

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

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

ENCODING with LINEAR CODES

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over F_q^n with a generator $k \times n$ matrix G.

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

Encoding of a message $u = (u_1, \ldots, u_k)$ using the generator matrix G:

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

Encoding of a message $u = (u_1, \dots, u_k)$ using the generator matrix G:

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \text{ where } r_1, \dots, r_k \text{ are rows of } G.$$

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n,k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

Encoding of a message $u = (u_1, \dots, u_k)$ using the generator matrix G:

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \text{ where } r_1, \dots, r_k \text{ are rows of } G.$$

Example Let C be a $[7,4]$-code with the generator matrix

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

A message (u_1, u_2, u_3, u_4) is encoded as:???

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n,k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

Encoding of a message $u = (u_1, \ldots, u_k)$ using the generator matrix G:

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \text{ where } r_1, \ldots, r_k \text{ are rows of } G.$$

Example Let C be a $[7,4]$-code with the generator matrix

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

A message (u_1, u_2, u_3, u_4) is encoded as:???
For example:
0 0 0 0 is encoded as? .....

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

Encoding of a message $u = (u_1, \ldots, u_k)$ using the generator matrix G:

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \text{ where } r_1, \ldots, r_k \text{ are rows of } G.$$

Example Let C be a $[7, 4]$-code with the generator matrix

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

A message (u_1, u_2, u_3, u_4) is encoded as:???
For example:
0 0 0 0 is encoded as? 0000000
1 0 0 0 is encoded as? .....

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n,k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

Encoding of a message $u = (u_1, \ldots, u_k)$ using the generator matrix G:

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \text{ where } r_1, \ldots, r_k \text{ are rows of } G.$$

Example Let C be a $[7,4]$-code with the generator matrix

$$\text{G}= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

A message (u_1, u_2, u_3, u_4) is encoded as:???
For example:
0 0 0 0 is encoded as? 0000000
1 0 0 0 is encoded as? 1000101
1 1 1 0 is encoded as? .....

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n,k]$-code over F_q^n with a generator $k \times n$ matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis codewords/vectors.

Corollary The code C can be used to encode uniquely q^k messages.
(Let us identify messages with elements of F_q^k.)

Encoding of a message $u = (u_1, \ldots, u_k)$ using the generator matrix G:

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \text{ where } r_1, \ldots, r_k \text{ are rows of } G.$$

Example Let C be a $[7,4]$-code with the generator matrix

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

A message (u_1, u_2, u_3, u_4) is encoded as:???
For example:
0 0 0 0 is encoded as? 0000000
1 0 0 0 is encoded as? 1000101
1 1 1 0 is encoded as? 1110100

SYNDROMES APPROACH to DECODING

DUAL CODES

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

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 *In* $F_2^4 : 1001 \cdot 1001 =$

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 =$

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 =$

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

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

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

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

$$C^\perp = \{v \in F_q^n \mid v \cdot u = 0 \text{ for all } u \in C\}.$$

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 *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

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

$$C^\perp = \{v \in F_q^n \mid v \cdot u = 0 \text{ for all } u \in C\}.$$

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

$$v \in C^\perp \Leftrightarrow vG^\top = 0,$$

where $G^\top$ denotes the transpose of the matrix G.

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

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

$$C^\perp = \{v \in F_q^n \mid v \cdot u = 0 \text{ for all } u \in C\}.$$

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

$$v \in C^\perp \Leftrightarrow vG^\top = 0,$$

where $G^\top$ denotes the transpose of the matrix G.

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

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

$$C^\perp = \{v \in F_q^n \mid v \cdot u = 0 \text{ for all } u \in C\}.$$

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

$$v \in C^\perp \Leftrightarrow vG^\top = 0,$$

where $G^\top$ denotes the transpose of the matrix G.

In general, the problem of finding the nearest neighbour in a linear code is NP-complete.

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

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

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $F_2^4 : 1001 \cdot 1001 = 0$
In $F_3^4 : 2001 \cdot 1210 = 2$
$1212 \cdot 2121 = 2$

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

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

$$C^\perp = \{v \in F_q^n \mid v \cdot u = 0 \text{ for all } u \in C\}.$$

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

$$v \in C^\perp \Leftrightarrow vG^\top = 0,$$

where $G^\top$ denotes the transpose of the matrix G.

In general, the problem of finding the nearest neighbour in a linear code is NP-complete. Fortunately, there are important linear codes with really efficient decoding.

HAMMING CODES

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

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 $Ham(r, 2)$.

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 $Ham(r, 2)$.

Example

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

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 $Ham(r, 2)$.

Example

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

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

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 $Ham(r, 2)$.

Example

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

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

Theorem Hamming code $Ham(r, 2)$

- is $[2^r - 1, 2^r - 1 - r]$-code,

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 $Ham(r, 2)$.

Example

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

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

Theorem Hamming code $Ham(r, 2)$

- is $[2^r - 1, 2^r - 1 - r]$-code,
- has minimum distance 3,

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 $Ham(r, 2)$.

Example

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

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

Theorem Hamming code $Ham(r, 2)$

- is $[2^r - 1, 2^r - 1 - r]$-code,
- has minimum distance 3,
- and is a perfect code.

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 $Ham(r, 2)$.

Example

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

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

Theorem Hamming code $Ham(r, 2)$

- is $[2^r - 1, 2^r - 1 - r]$-code,
- has minimum distance 3,
- and is a perfect code.

Properties of binary Hamming codes Coset leaders are precisely words of weight ≤ 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.

HAMMING CODES - DECODING

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

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 $S(y) = yH^{\top}$.

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.

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

EXAMPLE

For the Hamming code given by the parity-check matrix

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

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

$$y = 1101011,$$

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

$$S(y) = 110$$

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.

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).

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

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 telephon calls.

SOME BASIC IMPORTANT CODES

SOME BASIC 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.

SOME BASIC 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.

SOME BASIC 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.

SOME BASIC 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.

SOME BASIC 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 - DESCRIPTION

Golay codes G_{24} and G_{23} were used by Voyager I and Voyager II to transmit color pictures of Jupiter and Saturn.

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 very simple form:**

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 very simple form:**

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

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 very simple form:**

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

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.

GOLAY CODES - CONSTRUCTION

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.

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

$$0, 1, 3, 4, 5, 9.$$

REED-MULLER CODES

REED-MULLER CODES

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

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

REED-MULLER CODES

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

$$G_{r,m} = \left[G_{r-1,m} Q_r \right]$$

where Q_r is a matrix with dimension $\binom{m}{r} \times 2^m$ where ??????? are representations of the column numbers.

REED-MULLER CODES

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

$$G_{r,m} = \left[G_{r-1,m} Q_r \right]$$

where Q_r is a matrix with dimension $\binom{m}{r} \times 2^m$ where ??????? are 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 constitues a row of Q_r.

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 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, then

$$M \leq q^{n-d+1}$$

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, then

$$M \leq q^{n-d+1}$$

Proof Take some $d-1$ coordinates and project all codewords to the remaining coordinates.

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, then

$$M \leq q^{n-d+1}$$

Proof Take some $d-1$ coordinates and project all codewords to the remaining coordinates.

The resulting codewords have to be all different and therefore M cannot be larger than the number of q-ary words of the length $n-d-1$.

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, then

$$M \leq q^{n-d+1}$$

Proof Take some $d-1$ coordinates and project all codewords to the remaining coordinates.

The resulting codewords have to be all different and therefore M cannot be larger than the number of q-ary words of the length $n-d-1$.

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

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, then

$$M \leq q^{n-d+1}$$

Proof Take some $d-1$ coordinates and project all codewords to the remaining coordinates.

The resulting codewords have to be all different and therefore M cannot be larger than the number of q-ary words of the 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

$$d \leq n-k+1.$$

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, then

$$M \leq q^{n-d+1}$$

Proof Take some $d-1$ coordinates and project all codewords to the remaining coordinates.

The resulting codewords have to be all different and therefore M cannot be larger than the number of q-ary words of the 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

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

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, then

$$M \leq q^{n-d+1}$$

Proof Take some $d-1$ coordinates and project all codewords to the remaining coordinates.

The resulting codewords have to be all different and therefore M cannot be larger than the number of q-ary words of the 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 q-nary error-correcting codes with $d \geq n(1 - 1/q)$ have only polynomially many codewords and hence are not very interesting.

REED-SOLOMON CODES

REED-SOLOMON CODES

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

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

REED-SOLOMON CODES

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

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

RSC(k, q) code is $[q, k, q-k+1]$ code.

REED-SOLOMON CODES

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

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

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

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

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

REED-SOLOMON CODES

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

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

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

Example Generator matrix for $\mathrm{RSC}(3, 5)$ 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:

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

REED-SOLOMON CODES

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

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

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

Example Generator matrix for $\mathrm{RSC}(3, 5)$ 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:

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

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.

APPENDIX - I.

LDPC (Low-Density Parity Check) - CODES

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.

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 said to be 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.

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 said to be 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 recent years LDPC codes are replacing in many important applications other types of codes for the following reasons:

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 said to be 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 recent years LDPC codes are replacing in many important applications other types of codes for the following reasons:

1. 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.

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 said to be 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 recent years LDPC codes are replacing in many important applications other types of codes for the following reasons:

1. 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.
2. Good LDPC codes can be decoded in time linear to their block length using special (for example "iterative belief propagation") approximation techniques.

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 said to be 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 recent years LDPC codes are replacing in many important applications other types of codes for the following reasons:

1. 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.
2. Good LDPC codes can be decoded in time linear to their block length using special (for example "iterative belief propagation") approximation techniques.
3. Some LDPC codes are well suited for implementations that make heavy use of parallelism.

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 said to be 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 recent years LDPC codes are replacing in many important applications other types of codes for the following reasons:

1. 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.
2. Good LDPC codes can be decoded in time linear to their block length using special (for example "iterative belief propagation") approximation techniques.
3. Some LDPC codes are well suited for implementations that make heavy use of parallelism.

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 said to be 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 recent years LDPC codes are replacing in many important applications other types of codes for the following reasons:

1. 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.
2. Good LDPC codes can be decoded in time linear to their block length using special (for example "iterative belief propagation") approximation techniques.
3. Some LDPC codes are well suited for implementations that make heavy use of parallelism.

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,....

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 said to be 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 recent years LDPC codes are replacing in many important applications other types of codes for the following reasons:

1. 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.
2. Good LDPC codes can be decoded in time linear to their block length using special (for example "iterative belief propagation") approximation techniques.
3. Some LDPC codes are well suited for implementations that make heavy use of parallelism.

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,....

Parity-check matrices for LDPC codes are often (pseudo)-randomly generated, subject to

DISCOVERY and APPLICATION of LDPC CODES

LDPC codes were discovered in 1960 by R.C. Gallager in his PhD thesis,

DISCOVERY and APPLICATION of LDPC CODES

LDPC codes were discovered in 1960 by R.C. Gallager in his PhD thesis, but were ignored till 1996 when linear time decoding methods were discovered for some of them.

DISCOVERY and APPLICATION of LDPC CODES

LDPC codes were discovered in 1960 by R.C. Gallager in his PhD thesis, but were 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,....

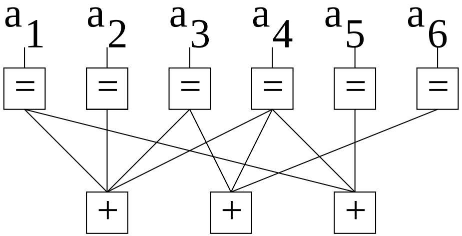
DISCOVERY and APPLICATION of LDPC CODES

LDPC codes were discovered in 1960 by R.C. Gallager in his PhD thesis, but were 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,....

BI-PARTITE (TANNER) GRAPHS REPRESENTATION of LDPC CODES

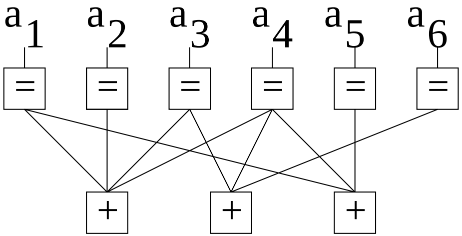
An $[n, k]$ LDPC code can be represented by a bipartite graph between a set of n top "variable-nodes (v-nodes)" and a set of bottom $(n - k)$ "parity check nodes (pc-nodes)".
Variable nodes:



Parity check nodes:

BI-PARTITE (TANNER) GRAPHS REPRESENTATION of LDPC CODES

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 (pc-nodes)".
Variable nodes:



Parity check nodes:

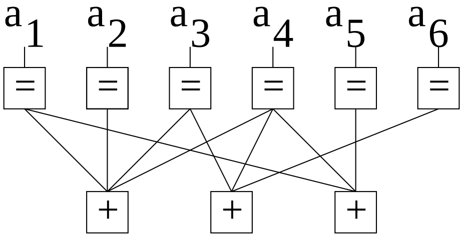
The corresponding parity check matrix has $n-k$ rows and n columns and i-th column has 1 in the j-th row exactly in case if i-th v-node is connected to j-th c-node.

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

TANNER GRAPHS - CONTINUATION

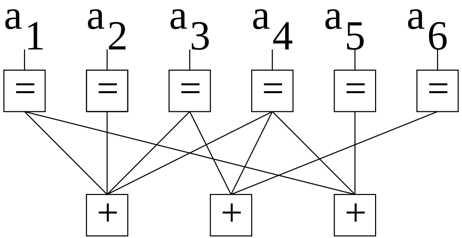
TANNER GRAPHS - CONTINUATION

The LDPC-code with the Tanner bipartite graph for $(6,3)$ LDPC-code.



TANNER GRAPHS - CONTINUATION

The LDPC-code with the Tanner bipartite graph for $(6,3)$ LDPC-code.

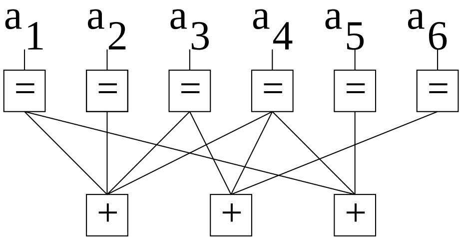


has the parity check matrix

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

TANNER GRAPHS - CONTINUATION

The LDPC-code with the Tanner bipartite graph for $(6,3)$ LDPC-code.



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 therefore the following constrains have to be satisfied:

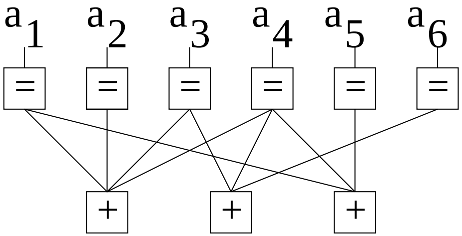
$$a_1 + a_2 + a_3 + a_4 = 0$$

$$a_3 + a_4 + a_6 = 0$$

$$a_1 + a_4 + a_5 = 0$$

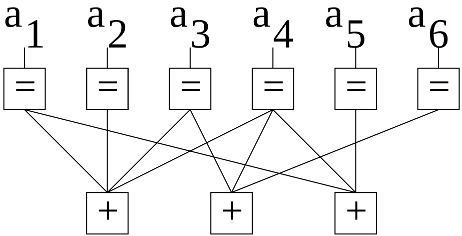
DECODING

Since for the LDPC-code with the Tanner bipartite graph for $(6, 3)$ LDPC-code.



DECODING

Since for the LDPC-code with the Tanner bipartite graph for $(6,3)$ LDPC-code.



the following constrains have to be satisfied:

$$a_1 + a_2 + a_3 + a_4 = 0$$

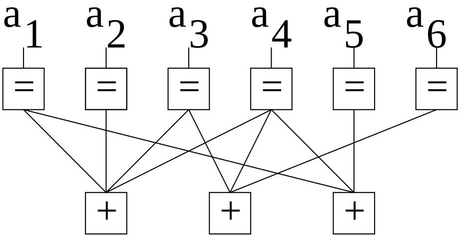
$$a_3 + a_4 + a_6 = 0$$

$$a_1 + a_4 + a_5 = 0$$

Let the word ?01?11 be received.

DECODING

Since for the LDPC-code with the Tanner bipartite graph for $(6,3)$ LDPC-code.



the following constrains have to be satisfied:

$$a_1 + a_2 + a_3 + a_4 = 0$$

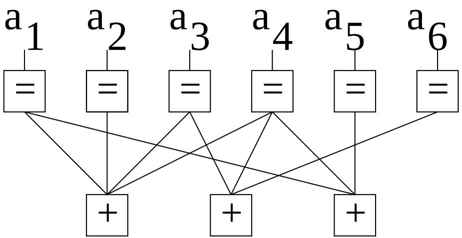
$$a_3 + a_4 + a_6 = 0$$

$$a_1 + a_4 + a_5 = 0$$

Let the word ?01?11 be received.From the second equation it follows that the second unknown symbol is

DECODING

Since for the LDPC-code with the Tanner bipartite graph for $(6, 3)$ LDPC-code.



the following constrains have to be satisfied:

$$a_1 + a_2 + a_3 + a_4 = 0$$

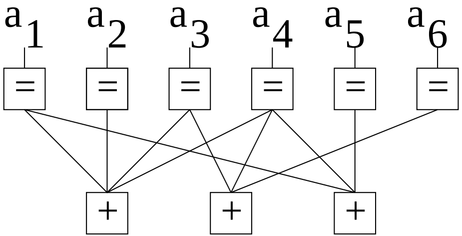
$$a_3 + a_4 + a_6 = 0$$

$$a_1 + a_4 + a_5 = 0$$

Let the word ?01?11 be received.From the second equation it follows that the second unknown symbol is 0. From the last equation it then follows that the first unknown symbol is

DECODING

Since for the LDPC-code with the Tanner bipartite graph for $(6,3)$ LDPC-code.



the following constrains have to be satisfied:

$$a_1 + a_2 + a_3 + a_4 = 0$$

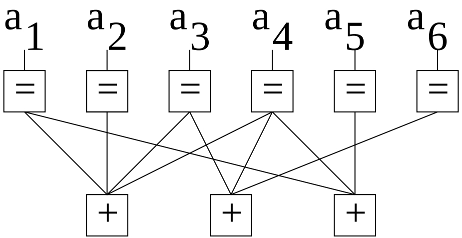
$$a_3 + a_4 + a_6 = 0$$

$$a_1 + a_4 + a_5 = 0$$

Let the word ?01?11 be received.From the second equation it follows that the second unknown symbol is 0. From the last equation it then follows that the first unknown symbol is 1.

DECODING

Since for the LDPC-code with the Tanner bipartite graph for $(6,3)$ LDPC-code.



the following constrains have to be satisfied:

$$a_1 + a_2 + a_3 + a_4 = 0$$

$$a_3 + a_4 + a_6 = 0$$

$$a_1 + a_4 + a_5 = 0$$

Let the word ?01?11 be received.From the second equation it follows that the second unknown symbol is 0. From the last equation it then follows that the first unknown symbol is 1.

Using so called **iterative belief propagation techniques**, LDPC codes can be decoded in time linear to their block length.

DESIGN of LDPC codes

DESIGN of LDPC codes

- Some good LDPC codes were designed through randomly chosen parity check matrices.
- Some LDPC codes are based on Reed-Solomon codes, such as the RS-LDPC code used in the 10-gigabit Ethernet standard.

LDPC CODES APPLICATIONS

LDPC CODES APPLICATIONS

- In the recent years have been several interesting competition between LDPC codes and Turbo codes introduced in Chapter 1 for various applications.

LDPC CODES APPLICATIONS

- In the recent years have been several interesting competition between LDPC codes and Turbo codes introduced in Chapter 1 for various applications.
- In 2003, an LDPC code was able to beat six turbo codes to become the error correcting code in the new DVB-S2 standard for satellite transmission for digital television.

LDPC CODES APPLICATIONS

- In the recent years have been several interesting competition between LDPC codes and Turbo codes introduced in Chapter 1 for various applications.
- In 2003, an LDPC code was able to 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.

POLYNOMIAL CODES

POLYNOMIAL CODES

A **Polynomial code**, with codewords of length n, **generated by a (generator) polynomial** $g(x)$ of degree $m < n$ over a GF(q) is the code whose codewords are represented exactly by those polynomials of degree less than n that are divisible by $g(x)$.

POLYNOMIAL CODES

A **Polynomial code**, with codewords of length n, **generated by a (generator) polynomial** $g(x)$ of degree $m < n$ over a GF(q) is the code whose codewords are represented exactly by those polynomials of degree less than n that are divisible by $g(x)$.

Example: For the binary polynomial code with $n = 5$ and $m = 2$ generated by the polynomial $g(x) = x^2 + x + 1$ all codewords are of the form:

$$a(x)g(x)$$

where

$$a(x) \in \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$$

POLYNOMIAL CODES

A **Polynomial code**, with codewords of length n, **generated by a (generator) polynomial** $g(x)$ of degree $m < n$ over a GF(q) is the code whose codewords are represented exactly by those polynomials of degree less than n that are divisible by $g(x)$.

Example: For the binary polynomial code with $n = 5$ and $m = 2$ generated by the polynomial $g(x) = x^2 + x + 1$ all codewords are of the form:

$$a(x)g(x)$$

where

$$a(x) \in \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$$

what results in the code with codewords

$$00000, 00111, 01110, 01001,$$

$$11100, 11011, 10010, 10101.$$

REED-MULLER CODES

REED-MULLER CODES

Reed-Muuller codes with parameters $0 \leq r \leq m$, notation $RM(r, m)$, are linear block codes, usually binary - mapping binary messages to binary codewords, used currently especially in wireless and deep-space communications.

REED-MULLER CODES

Reed-Muuller codes with parameters $0 \leq r \leq m$, notation $RM(r,m)$, are linear block codes, usually binary - mapping binary messages to binary codewords, used currently especially in wireless and deep-space communications.

Each Reed-Muller code $RM(r,m)$ is the code of k codewords of length $n = 2^m$, to encode k messages, and distance 2^{m-r}, where

$$k = \sum_{s=0}^{r} \binom{d}{s}.$$

There are several elegant, mathematically sophisticated ways, to describe Reed-Muller codes and they have nice and useful properties. As a congruence they are locally testable, locally decodable, list decodable and useful in probabilistically checkable proofs -see rest of this chapter.

REED-MULLER CODES

Reed-Muuller codes with parameters $0 \leq r \leq m$, notation $RM(r, m)$, are linear block codes, usually binary - mapping binary messages to binary codewords, used currently especially in wireless and deep-space communications.

Each Reed-Muller code $RM(r, m)$ is the code of k codewords of length $n = 2^m$, to encode k messages, and distance 2^{m-r}, where

$$k = \sum_{s=0}^{r} \binom{d}{s}.$$

There are several elegant, mathematically sophisticated ways, to describe Reed-Muller codes and they have nice and useful properties. As a congruence they are locally testable, locally decodable, list decodable and useful in probabilistically checkable proofs -see rest of this chapter.

$RM(r, m)$ code is generated by the set of all up to r inner products of the codewords v_i, $0 \leq i \leq d$, where $v_0 = 1^{2^d}$ and v_i are prefixes of the word $\{1^i 0^i\}^*$.

Example 1: $RM(1, 3)$ code is generated by the codewords

$$v_0 = 11111111$$

$$v_1 = 10101010$$

$$v_2 = 11001100$$

REED-MULLER CODES

Reed-Muuller codes with parameters $0 \leq r \leq m$, notation $RM(r, m)$, are linear block codes, usually binary - mapping binary messages to binary codewords, used currently especially in wireless and deep-space communications.

Each Reed-Muller code $RM(r, m)$ is the code of k codewords of length $n = 2^m$, to encode k messages, and distance 2^{m-r}, where

$$k = \sum_{s=0}^{r} \binom{d}{s}.$$

There are several elegant, mathematically sophisticated ways, to describe Reed-Muller codes and they have nice and useful properties. As a congruence they are locally testable, locally decodable, list decodable and useful in probabilistically checkable proofs -see rest of this chapter.

$RM(r, m)$ code is generated by the set of all up to r inner products of the codewords v_i, $0 \leq i \leq d$, where $v_0 = 1^{2^d}$ and v_i are prefixes of the word $\{1^i 0^i\}^*$.

Example 1: $RM(1, 3)$ code is generated by the codewords

$$v_0 = 11111111$$

$$v_1 = 10101010$$

$$v_2 = 11001100$$

REED-MULLER CODES

Reed-Muuller codes with parameters $0 \leq r \leq m$, notation $RM(r,m)$, are linear block codes, usually binary - mapping binary messages to binary codewords, used currently especially in wireless and deep-space communications.

Each Reed-Muller code $RM(r,m)$ is the code of k codewords of length $n = 2^m$, to encode k messages, and distance 2^{m-r}, where

$$k = \sum_{s=0}^{r} \binom{d}{s}.$$

There are several elegant, mathematically sophisticated ways, to describe Reed-Muller codes and they have nice and useful properties. As a congruence they are locally testable, locally decodable, list decodable and useful in probabilistically checkable proofs -see rest of this chapter.

$RM(r,m)$ code is generated by the set of all up to r inner products of the codewords v_i, $0 \leq i \leq d$, where $v_0 = 1^{2^d}$ and v_i are prefixes of the word $\{1^i 0^i\}^*$.

Example 1: $RM(1,3)$ code is generated by the codewords

$$v_0 = 11111111$$

$$v_1 = 10101010$$

$$v_2 = 11001100$$

REED-MULLER CODES

Reed-Muuller codes with parameters $0 \leq r \leq m$, notation $RM(r, m)$, are linear block codes, usually binary - mapping binary messages to binary codewords, used currently especially in wireless and deep-space communications.

Each Reed-Muller code $RM(r, m)$ is the code of k codewords of length $n = 2^m$, to encode k messages, and distance 2^{m-r}, where

$$k = \sum_{s=0}^{r} \binom{d}{s}.$$

There are several elegant, mathematically sophisticated ways, to describe Reed-Muller codes and they have nice and useful properties. As a congruence they are locally testable, locally decodable, list decodable and useful in probabilistically checkable proofs -see rest of this chapter.

$RM(r, m)$ code is generated by the set of all up to r inner products of the codewords v_i, $0 \leq i \leq d$, where $v_0 = 1^{2^d}$ and v_i are prefixes of the word $\{1^i 0^i\}^*$.

Example 1: $RM(1, 3)$ code is generated by the codewords

$$v_0 = 11111111$$

$$v_1 = 10101010$$

$$v_2 = 11001100$$

BCH CODES and REED-SOLOMON CODES

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[1] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[1] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[1] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \dots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

Applications of BCH codes: satellite communications, compact disc players,disk drives, two-dimensional bar codes,...

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[1] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \dots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

Applications of BCH codes: satellite communications, compact disc players,disk drives, two-dimensional bar codes,...

Comments: For BCH codes there exist efficient variations of syndrome decoding.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[1] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

Applications of BCH codes: satellite communications, compact disc players,disk drives, two-dimensional bar codes,...

Comments: For BCH codes there exist efficient variations of syndrome decoding. A Reed-Solomon code is a special primitive BCH code.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

REED-SOLOMON CODES - basic idea behind - I

REED-SOLOMON CODES - basic idea behind - I

A message of k symbols can be encoded by viewing these symbols as coefficients of a polynomial of degree $k-1$ over a finite field of order N, evaluating this polynomial at more than k distinct points and sending the outcomes to the receiver.

REED-SOLOMON CODES - basic idea behind - I

A message of k symbols can be encoded by viewing these symbols as coefficients of a polynomial of degree $k-1$ over a finite field of order N, evaluating this polynomial at more than k distinct points and sending the outcomes to the receiver.

Having more than k points of the polynomial allows to determine exactly, through the Lagrangian interpolation, the original polynomial (message).

REED-SOLOMON CODES - basic idea behind - I

A message of k symbols can be encoded by viewing these symbols as coefficients of a polynomial of degree $k - 1$ over a finite field of order N, evaluating this polynomial at more than k distinct points and sending the outcomes to the receiver.

Having more than k points of the polynomial allows to determine exactly, through the Lagrangian interpolation, the original polynomial (message).

REED-SOLOMON CODES - basic idea behind - I

A message of k symbols can be encoded by viewing these symbols as coefficients of a polynomial of degree $k-1$ over a finite field of order N, evaluating this polynomial at more than k distinct points and sending the outcomes to the receiver.

Having more than k points of the polynomial allows to determine exactly, through the Lagrangian interpolation, the original polynomial (message).

Variations of Reed-Solomon codes are obtained by specifying ways distinct points are generated and error-correction is performed.

REED-SOLOMON CODES - basic idea behind - I

A message of k symbols can be encoded by viewing these symbols as coefficients of a polynomial of degree $k-1$ over a finite field of order N, evaluating this polynomial at more than k distinct points and sending the outcomes to the receiver.

Having more than k points of the polynomial allows to determine exactly, through the Lagrangian interpolation, the original polynomial (message).

Variations of Reed-Solomon codes are obtained by specifying ways distinct points are generated and error-correction is performed.

Reed-Solomon codes found many important applications from deep-space travel to consumer electronics.

REED-SOLOMON CODES - basic idea behind - I

A message of k symbols can be encoded by viewing these symbols as coefficients of a polynomial of degree $k-1$ over a finite field of order N, evaluating this polynomial at more than k distinct points and sending the outcomes to the receiver.

Having more than k points of the polynomial allows to determine exactly, through the Lagrangian interpolation, the original polynomial (message).

Variations of Reed-Solomon codes are obtained by specifying ways distinct points are generated and error-correction is performed.

Reed-Solomon codes found many important applications from deep-space travel to consumer electronics.

They are very useful especially in those applications where one can expect that errors occur in bursts - such as ones caused by solar energy.

UNIQUE DECODING versus LIST DECODING

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

This list decoding error-correction task/model is not sufficiently good in case when the number of error can be large.
n the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

This list decoding error-correction task/model is not sufficiently good in case when the number of error can be large.
n the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

This list decoding error-correction task/model is not sufficiently good in case when the number of error can be large.
n the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

It has turned out that for a variety of important codes, say for Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors.

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

This list decoding error-correction task/model is not sufficiently good in case when the number of error can be large.
n the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

It has turned out that for a variety of important codes, say for Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors. The notion of list-decoding, as a relaxed error-correcting mode, was proposed by Elias in 1950s.

EFFICIENCY of LIST DECODING

EFFICIENCY of LIST DECODING

With list decoding the error-correction performance doubles.

EFFICIENCY of LIST DECODING

With list decoding the error-correction performance doubles.

It has been shown, non-constructively, that codes of the rate R exist that can be list decoded up to a fraction of errors approaching $1 - R$.

EFFICIENCY of LIST DECODING

With list decoding the error-correction performance doubles.

It has been shown, non-constructively, that codes of the rate R exist that can be list decoded up to a fraction of errors approaching $1 - R$.

The quantity $1 - R$ is referred to as the **list decoding capacity**.

EFFICIENCY of LIST DECODING

With list decoding the error-correction performance doubles.

It has been shown, non-constructively, that codes of the rate R exist that can be list decoded up to a fraction of errors approaching $1 - R$.

The quantity $1 - R$ is referred to as the **list decoding capacity**.

For Reed-Solomon codes there is list decoding up to $1 - \sqrt{2R}$ errors.

CHANNELS (STREAMS) CODING

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology.

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology. How well can channel coding be done?

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology. How well can channel coding be

done? So called **Shannon's channel coding theorem says** that over many common channels there exist data coding schemes that are able to transmit data reliably at all code rates smaller than a certain **threshold**, called nowadays the **Shannon channel capacity**, of the given channel.

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology. How well can channel coding be

done? So called **Shannon's channel coding theorem says** that over many common channels there exist data coding schemes that are able to transmit data reliably at all code rates smaller than a certain **threshold**, called nowadays the **Shannon channel capacity**, of the given channel.

Moreover, the theorem says that probability of a decoding error can be made to decrease exponentially as the block length N of the coding scheme goes to infinity. However, the

complexity of a "naive", or straightforward, optimum decoding schemes increased exponentially with N - therefore such an optimum decoder rapidly become unfeasible.

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology. How well can channel coding be

done? So called **Shannon's channel coding theorem says** that over many common channels there exist data coding schemes that are able to transmit data reliably at all code rates smaller than a certain **threshold**, called nowadays the **Shannon channel capacity**, of the given channel.

Moreover, the theorem says that probability of a decoding error can be made to decrease exponentially as the block length N of the coding scheme goes to infinity. However, the

complexity of a "naive", or straightforward, optimum decoding schemes increased exponentially with N - therefore such an optimum decoder rapidly become unfeasible.

A breakthrough came when D. Forney, in his PhD thesis in 1972, showed that so called concatenated codes could be used to achieve exponentially decreasing error probabilities at all data rates less than the Shannon channel capacity, with decoding complexity increasing only polynomially with the code length.

CHANNEL (STREAMS) CODING I.

Therefore, the task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

CHANNEL (STREAMS) CODING I.

Therefore, the task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

CHANNEL (STREAMS) CODING I.

Therefore, the task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

An important parameter of a channel code is **code rate**

$$r = \frac{k}{n}$$

in case k bits are encoded by n bits.

CHANNEL (STREAMS) CODING I.

Therefore, the task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

An important parameter of a channel code is **code rate**

$$r = \frac{k}{n}$$

in case k bits are encoded by n bits.

The code rate express the amount of redundancy in the code - the lower is the code rate, the more redundancy is in the codewords.

CHANNEL (STREAM) CODING II

CHANNEL (STREAM) CODING II

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**
- **tolerate more interference from the environment;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**
- **tolerate more interference from the environment;**
- **use smaller antennas;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**
- **tolerate more interference from the environment;**
- **use smaller antennas;**
- **use smaller antennas;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**
- **tolerate more interference from the environment;**
- **use smaller antennas;**
- **use smaller antennas;**
- **transmit at a higher data rate.**

CHANNEL CAPACITY

Channel capacity of a communication channel, is the tightest upper bound on the (code) rate of information that can be reliably transmitted over that channel.

CHANNEL CAPACITY

Channel capacity of a communication channel, is the tightest upper bound on the (code) rate of information that can be reliably transmitted over that channel.

By the **noisy-channel Shannon coding theorem**, the channel capacity of a given channel is the limiting code rate (in units of information per unit time) that can be achieved with arbitrary small error probability.

CHANNEL CAPACITY - FORMAL DEFINITION

Let X and Y be random variables representing the input and output of a channel.

CHANNEL CAPACITY - FORMAL DEFINITION

Let X and Y be random variables representing the input and output of a channel.

Let $P_{Y|X}(y|x)$ be the conditional probability distribution function of Y given X, which can be seen as an inherent fixed probability of the communication channel.

CHANNEL CAPACITY - FORMAL DEFINITION

Let X and Y be random variables representing the input and output of a channel.

Let $P_{Y|X}(y|x)$ be the conditional probability distribution function of Y given X, which can be seen as an inherent fixed probability of the communication channel.

The joint distribution $P_{X,Y}(x,y)$ is then defined by

$$P_{X,Y}(x,y) = P_{Y|X}(y|x)P_X(x),$$

where $P_X(x)$ is the marginal distribution.

CHANNEL CAPACITY - FORMAL DEFINITION

Let X and Y be random variables representing the input and output of a channel.

Let $P_{Y|X}(y|x)$ be the conditional probability distribution function of Y given X, which can be seen as an inherent fixed probability of the communication channel.

The joint distribution $P_{X,Y}(x,y)$ is then defined by

$$P_{X,Y}(x,y) = P_{Y|X}(y|x)P_X(x),$$

where $P_X(x)$ is the marginal distribution.

The channel capacity is then defined by

$$C = \sup_{P_X(x)} I(X,Y)$$

where

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} P_{X,Y}(x,y) \log \left(\frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)} \right)$$

is the mutual distribution - a measure of variables mutual distribution.

SHANNON NOISY CHANEL THEOREM

For every discrete memoryless channel, the channel capacity

$$C = \sup_{P_X} I(X, Y)$$

has the following properties:
1. For every $\varepsilon > 0$ and $R < C$, for large enough N there exists a code of length N and code rate R and a decoding algorithm, such that the maximal probability of the block error is $\leq \varepsilon$.
2. If a probability of the block error p_b is acceptable, code rates up to $R(p_b)$ are achievable, where and $H_2(p_b)$ is the binary entropy function. and $H_2(p_b)$ is the binary entropy function.
3. For any p_b code rates greater than $R(p_b)$ are not achievable.

ENCODING of FINITE POLYNOMIALS

ENCODING of FINITE POLYNOMIALS

An (n,k) convolution code with a k x n generator matrix G can be used to encode a k-tuple of **message-polynomials** (polynomial input information)

$$I = (I_0(x), I_1(x), \ldots, I_{k-1}(x))$$

ENCODING of FINITE POLYNOMIALS

An (n,k) convolution code with a k x n generator matrix G can be used to encode a k-tuple of **message-polynomials** (polynomial input information)

$$I = (I_0(x), I_1(x), \ldots, I_{k-1}(x))$$

to get an n-tuple of **encoded-polynomials**

$$C = (C_0(x), C_1(x), \ldots, C_{n-1}(x))$$

where

ENCODING of FINITE POLYNOMIALS

An (n,k) convolution code with a k x n generator matrix G can be used to encode a k-tuple of **message-polynomials** (polynomial input information)

$$I = (I_0(x), I_1(x), \ldots, I_{k-1}(x))$$

to get an n-tuple of **encoded-polynomials**

$$C = (C_0(x), C_1(x), \ldots, C_{n-1}(x))$$

where

$$C_j(x) = I_j(x) \cdot G$$

EXAMPLES

EXAMPLES

EXAMPLE 1 – when the code CC_1 is used:

$$\begin{aligned}(x^3+x+1)\cdot G_1 &= (x^3+x+1)\cdot(x^2+1, x^2+x+1)\\ &= (x^5+x^2+x+1, x^5+x^4+1)\end{aligned}$$

EXAMPLES

EXAMPLE 1 – when the code CC_1 is used:

$$\begin{aligned}(x^3 + x + 1) \cdot G_1 &= (x^3 + x + 1) \cdot (x^2 + 1, x^2 + x + 1)\\ &= (x^5 + x^2 + x + 1, x^5 + x^4 + 1)\end{aligned}$$

EXAMPLE 2 – when the code CC_2 is used:

$$(x^2 + x, x^3 + 1) \cdot G_2 = (x^2 + x, x^3 + 1) \cdot \begin{pmatrix} 1 + x & 0 & x + 1 \\ 0 & 1 & x \end{pmatrix}$$

ENCODING of INFINITE INPUT STREAMS

ENCODING of INFINITE INPUT STREAMS

One of the way infinite streams can be encoded using convolution codes will be Illustrated on the code CC_1.

ENCODING of INFINITE INPUT STREAMS

One of the way infinite streams can be encoded using convolution codes will be Illustrated on the code CC_1.

An input stream $I(x) = (I_0(x), I_1(x), I_2(x), \ldots)$ is mapped into the output stream $C = (C_{00}, C_{10}, C_{01}, C_{11} \ldots)$ defined by

ENCODING of INFINITE INPUT STREAMS

One of the way infinite streams can be encoded using convolution codes will be Illustrated on the code CC_1.

An input stream $I(x) = (I_0(x), I_1(x), I_2(x), \ldots)$ is mapped into the output stream $C = (C_{00}, C_{10}, C_{01}, C_{11} \ldots)$ defined by

$$C_0(x) = C_{00} + C_{01}x + \ldots = (x^2 + 1)I(x)$$

and

$$C_1(x) = C_{10} + C_{11}x + \ldots = (x^2 + x + 1)I(x).$$

ENCODING of INFINITE INPUT STREAMS

One of the way infinite streams can be encoded using convolution codes will be Illustrated on the code CC_1.

An input stream $I(x) = (I_0(x), I_1(x), I_2(x), \ldots)$ is mapped into the output stream $C = (C_{00}, C_{10}, C_{01}, C_{11} \ldots)$ defined by

$$C_0(x) = C_{00} + C_{01}x + \ldots = (x^2 + 1)I(x)$$

and

$$C_1(x) = C_{10} + C_{11}x + \ldots = (x^2 + x + 1)I(x).$$

The first multiplication can be done by the first shift register from the next figure; second multiplication can be performed by the second shift register on the next slide and it holds

$$C_{0i} = I_i(x) + I_{i+2}(x), \quad C_{1i}(x) = I_i + I_{i-1} + I_{i-2}.$$

ENCODING of INFINITE INPUT STREAMS

One of the way infinite streams can be encoded using convolution codes will be Illustrated on the code CC_1.

An input stream $I(x) = (I_0(x), I_1(x), I_2(x), \ldots)$ is mapped into the output stream $C = (C_{00}, C_{10}, C_{01}, C_{11} \ldots)$ defined by

$$C_0(x) = C_{00} + C_{01}x + \ldots = (x^2 + 1)I(x)$$

and

$$C_1(x) = C_{10} + C_{11}x + \ldots = (x^2 + x + 1)I(x).$$

The first multiplication can be done by the first shift register from the next figure; second multiplication can be performed by the second shift register on the next slide and it holds

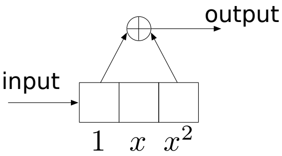
$$C_{0i} = I_i(x) + I_{i+2}(x), \quad C_{1i}(x) = I_i + I_{i-1} + I_{i-2}.$$

That is the output streams C_0 and C_1 are obtained by convoluting the input stream with polynomials of G_1.

ENCODING

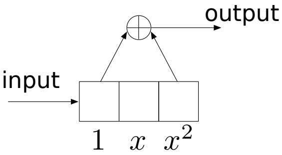
ENCODING

The **first shift register**



ENCODING

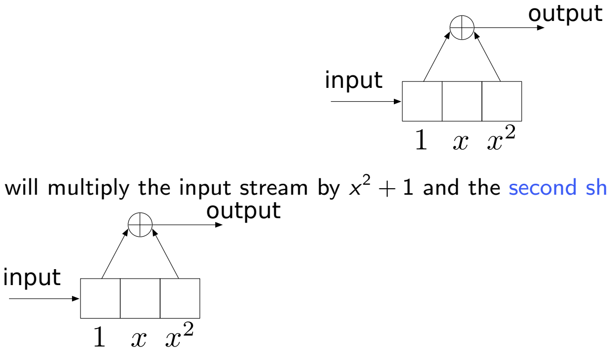
The **first shift register**



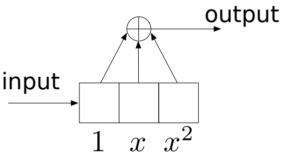
will multiply the input stream by $x^2 + 1$ and the second shift register

ENCODING

The **first shift register**

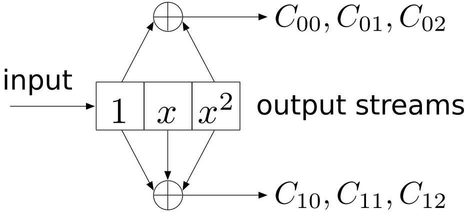


will multiply the input stream by $x^2 + 1$ and the second shift register



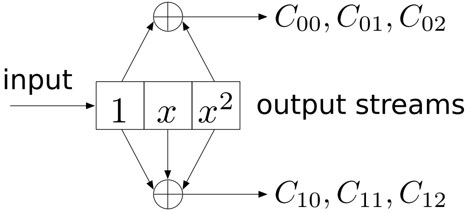
will multiply the input stream by $x^2 + x + 1$.

The following shift-register will therefore be an encoder for the code CC_1



ENCODING and DECODING

The following shift-register will therefore be an encoder for the code CC_1



For decoding of convolution codes so called

Viterbi algorithm

is used.

VITERBI ALGORITHM

VITERBI ALGORITHM

- In 1967 Andrew Vieterbi constructed his nowadays famous decoding algorithm for soft decoding.

VITERBI ALGORITHM

- In 1967 Andrew Vieterbi constructed his nowadays famous decoding algorithm for soft decoding.
- Vieterbi was very modest in evaluation of importance of his algorithm - considered it as impractical.

VITERBI ALGORITHM

- In 1967 Andrew Vieterbi constructed his nowadays famous decoding algorithm for soft decoding.
- Vieterbi was very modest in evaluation of importance of his algorithm - considered it as impractical.
- Although this algorithm was rendered as impractical due to the excessive storage requirements it started to be well known,

VITERBI ALGORITHM

- In 1967 Andrew Vieterbi constructed his nowadays famous decoding algorithm for soft decoding.
- Vieterbi was very modest in evaluation of importance of his algorithm - considered it as impractical.
- Although this algorithm was rendered as impractical due to the excessive storage requirements it started to be well known, because it contributes to a general understanding of convolution codes and sequential decoding through its simplicity of mechanization and analysis.
- Nowadays

VITERBI ALGORITHM

- In 1967 Andrew Vieterbi constructed his nowadays famous decoding algorithm for soft decoding.
- Vieterbi was very modest in evaluation of importance of his algorithm - considered it as impractical.
- Although this algorithm was rendered as impractical due to the excessive storage requirements it started to be well known, because it contributes to a general understanding of convolution codes and sequential decoding through its simplicity of mechanization and analysis.
- Nowadays (since 2006),

VITERBI ALGORITHM

- In 1967 Andrew Vieterbi constructed his nowadays famous decoding algorithm for soft decoding.
- Vieterbi was very modest in evaluation of importance of his algorithm - considered it as impractical.
- Although this algorithm was rendered as impractical due to the excessive storage requirements it started to be well known, because it contributes to a general understanding of convolution codes and sequential decoding through its simplicity of mechanization and analysis.
- Nowadays (since 2006), a Viterbi decoder in a cellphone takes up the area

VITERBI ALGORITHM

- In 1967 Andrew Vieterbi constructed his nowadays famous decoding algorithm for soft decoding.
- Vieterbi was very modest in evaluation of importance of his algorithm - considered it as impractical.
- Although this algorithm was rendered as impractical due to the excessive storage requirements it started to be well known, because it contributes to a general understanding of convolution codes and sequential decoding through its simplicity of mechanization and analysis.
- Nowadays (since 2006), a Viterbi decoder in a cellphone takes up the area **of a tenth of a square millimeter.**

BIAGWN CHANNELS

BIAGWN CHANNELS

Binary Input Additive Gaussian White Noise (BIAGWN) channel, is a continuous channel.

Binary Input Additive Gaussian White Noise (BIAGWN) channel, is a continuous channel.

A BIAGWN channel, with a standard deviation $\sigma \geq 0$, can be seen as a mapping

$$X_\sigma = \{-1, 1\} \rightarrow R,$$

where R is the set of reals.

BIAGWN CHANNELS

Binary Input Additive Gaussian White Noise (BIAGWN) channel, is a continuous channel.

A BIAGWN channel, with a standard deviation $\sigma \geq 0$, can be seen as a mapping

$$X_\sigma = \{-1, 1\} \rightarrow R,$$

where R is the set of reals.

The noise of BIAGWN is modeled by continuous Gaussian probability distribution function:

BIAGWN CHANNELS

Binary Input Additive Gaussian White Noise (BIAGWN) channel, is a continuous channel.

A BIAGWN channel, with a standard deviation $\sigma \geq 0$, can be seen as a mapping

$$X_\sigma = \{-1, 1\} \rightarrow R,$$

where R is the set of reals.

The noise of BIAGWN is modeled by continuous Gaussian probability distribution function:

Given $(x, y) \in \{-1, 1\} \times R$, the noise $y - x$ is distributed according to the Gaussian distribution of zero mean and standard derivation σ of the channel

$$Pr(y|x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-x)^2}{2\sigma^2}}$$

SHANNON CHANNEL CAPACITY

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

For BIAGWN channels, that well capture deep space channels, this limit is (by so-called Shannon-Hartley theorem):

$$R < W \log\left(1 + \frac{S}{N}\right) \quad \{\text{bits per second}\}$$

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

For BIAGWN channels, that well capture deep space channels, this limit is (by so-called Shannon-Hartley theorem):

$$R < W \log\left(1 + \frac{S}{N}\right) \quad \{\text{bits per second}\}$$

Shannon capacity sets a limit to the energy efficiency of the code.

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

For BIAGWN channels, that well capture deep space channels, this limit is (by so-called Shannon-Hartley theorem):

$$R < W \log\left(1 + \frac{S}{N}\right) \quad \{\text{bits per second}\}$$

Shannon capacity sets a limit to the energy efficiency of the code. **Till 1993 channel**

code designers were unable to develop codes with performance close to Shannon capacity limit, that is so called Shannon capacity approaching codes, and practical codes required about twice as much energy as theoretical minimum predicted.

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

For BIAGWN channels, that well capture deep space channels, this limit is (by so-called Shannon-Hartley theorem):

$$R < W \log\left(1 + \frac{S}{N}\right) \quad \{\text{bits per second}\}$$

Shannon capacity sets a limit to the energy efficiency of the code. **Till 1993 channel**

code designers were unable to develop codes with performance close to Shannon capacity limit, that is so called Shannon capacity approaching codes, and practical codes required about twice as much energy as theoretical minimum predicted.

Therefore, there was a big need for better codes with performance (arbitrarily) close to Shannon capacity limits.

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

For BIAGWN channels, that well capture deep space channels, this limit is (by so-called Shannon-Hartley theorem):

$$R < W \log \left(1 + \frac{S}{N}\right) \quad \{\text{bits per second}\}$$

Shannon capacity sets a limit to the energy efficiency of the code. **Till 1993 channel**

code designers were unable to develop codes with performance close to Shannon capacity limit, that is so called Shannon capacity approaching codes, and practical codes required about twice as much energy as theoretical minimum predicted.

Therefore, there was a big need for better codes with performance (arbitrarily) close to Shannon capacity limits.

Concatenated codes and Turbo codes, discussed later, have such a Shannon capacity approaching property.

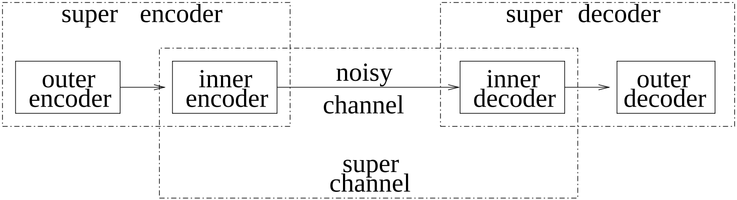
CONCATENATED CODES - I

CONCATENATED CODES - I

The basic idea of concatenated codes is extremely simple. A given message is first encoded by the first (outer) code C_1 (C_{out}) and C_1-output is then encoded by the second code C_2 (C_{in}). To decode, at first C_2 decoding and then C_1 decoding are used.

CONCATENATED CODES - I

The basic idea of concatenated codes is extremely simple. A given message is first encoded by the first (outer) code C_1 (C_{out}) and C_1-output is then encoded by the second code C_2 (C_{in}). To decode, at first C_2 decoding and then C_1 decoding are used.



CONCATENATED CODES II.

In 1962 Formey showed that concatenated codes could be used to achieve exponentially decreasing error probabilities at all data rates less than channel capacity in such a way that decoding complexity increases only polynomially with the code block length.

CONCATENATED CODES II.

In 1962 Formey showed that concatenated codes could be used to achieve exponentially decreasing error probabilities at all data rates less than channel capacity in such a way that decoding complexity increases only polynomially with the code block length.

In 1965 concatenated codes were considered as unfeasible. However, already in 1970s technology has advanced sufficiently and they became standardize by NASA for space applications.

CONCATENATED CODES BRIEFLY

CONCATENATED CODES BRIEFLY

A code concatenated codes C_{out} and C_{in} maps a message

$$m = (m_1, m_2, \ldots, m_K),$$

as follows: At first C_{out} encoding is applied to get

$$C_{out}(m_1, m_2, \ldots, m_k) = (m_1^{'}, m_2^{'}, \ldots, m_N^{'})$$

CONCATENATED CODES BRIEFLY

A code concatenated codes C_{out} and C_{in} maps a message

$$m = (m_1, m_2, \ldots, m_K),$$

as follows: At first C_{out} encoding is applied to get

$$C_{out}(m_1, m_2, \ldots, m_k) = (m_1^{'}, m_2^{'}, \ldots, m_N^{'})$$

and then C_{in} encoding is applied to get

CONCATENATED CODES BRIEFLY

A code concatenated codes C_{out} and C_{in} maps a message

$$m = (m_1, m_2, \dots, m_K),$$

as follows: At first C_{out} encoding is applied to get

$$C_{out}(m_1, m_2, \dots, m_k) = (m_1^{'}, m_2^{'}, \dots, m_N^{'})$$

and then C_{in} encoding is applied to get

$$C_{in}(m_1^{'}), C_{in}(m_2^{'}), \dots, C_{in}(m_N^{'})$$

BCH CODES and REED-SOLOMON CODES

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

---

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[2] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[2] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

$$\omega \text{ is the primitive } n\text{-th root of unity.}$$

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[2] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

Applications of BCH codes: satellite communications, compact disc players,disk drives, two-dimensional bar codes,...

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[2] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

$$\omega \text{ is the primitive } n\text{-th root of unity.}$$

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

Applications of BCH codes: satellite communications, compact disc players,disk drives, two-dimensional bar codes,...
Comments: For BCH codes there exist efficient variations of syndrome decoding.

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

BCH CODES and REED-SOLOMON CODES

BCH codes and **Reed-Solomon codes** belong to the most important codes for applications.

Definition A polynomial p is said to be **a minimal polynomial** for a complex number x in $GF(q)$ if $p(x) = 0$ and p is irreducible over $GF(q)$.

Definition A cyclic code of codewords of length n over $GF(q)$, where q is a power of a prime p, is called **BCH code**[2] of the distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \dots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

Applications of BCH codes: satellite communications, compact disc players,disk drives, two-dimensional bar codes,...

Comments: For BCH codes there exist efficient variations of syndrome decoding. A Reed-Solomon code is a special primitive BCH code.

[2] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes in 1959.

UNIQUE DECODING versus LIST DECODING

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

This list decoding error-correction task/model is not sufficiently good in case when the number of error can be large.

In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

This list decoding error-correction task/model is not sufficiently good in case when the number of error can be large.

In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

UNIQUE DECODING versus LIST DECODING

In the **unique decoding** model of error-correction, considered so far, the task is to find, for a received (corrupted) message w_c, the closest (and therefore unique) codeword w' to the one which was sent when the message w_c was received.

This list decoding error-correction task/model is not sufficiently good in case when the number of error can be large.

In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

It has turned out that for a variety of important codes, say for Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors.

LIST DECODING

The notion of list-decoding, as a relaxed error-correcting mode, was proposed by Elias in 1950s.

LIST DECODING

The notion of list-decoding, as a relaxed error-correcting mode, was proposed by Elias in 1950s.

In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

LIST DECODING

The notion of list-decoding, as a relaxed error-correcting mode, was proposed by Elias in 1950s.
In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

With list decoding the error-correction performance doubles.

LIST DECODING

The notion of list-decoding, as a relaxed error-correcting mode, was proposed by Elias in 1950s.

In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

With list decoding the error-correction performance doubles.

It has been shown, non-constructively, that codes of the rate R exist that can be list decoded up to a fraction of errors approaching $1 - R$.

LIST DECODING

The notion of list-decoding, as a relaxed error-correcting mode, was proposed by Elias in 1950s.

In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

With list decoding the error-correction performance doubles.

It has been shown, non-constructively, that codes of the rate R exist that can be list decoded up to a fraction of errors approaching $1 - R$.

The quantity $1 - R$ is referred to as the **list decoding capacity**.

LIST DECODING

The notion of list-decoding, as a relaxed error-correcting mode, was proposed by Elias in 1950s.
In the **list decoding** model the task is for a received (corrupted) message w_c and a given ϵ to output (list of) all codewords with the distance at most ε from the codeword that was sentw_c.

With list decoding the error-correction performance doubles.

It has been shown, non-constructively, that codes of the rate R exist that can be list decoded up to a fraction of errors approaching $1 - R$.

The quantity $1 - R$ is referred to as the **list decoding capacity**.

For Reed-Solomon codes there is list decoding up to $1 - \sqrt{2R}$ errors.

LISToT DECODING III

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

LISToT DECODING III

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

It has turned out that for a variety of important codes, say for Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors.

LISToT DECODING III

List decoding is considered to be successful in case the outputted list contains the codeword that was sent.

It has turned out that for a variety of important codes, say for Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors.

CHANNELS (STREAMS) CODING

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology.

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology.

How well can channel coding be done?

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology.

How well can channel coding be done? So called **Shannon's channel coding theorem says** that over many common channels there exist data coding schemes that are able to transmit data reliably at all code rates smaller than a certain **threshold**, called nowadays the **Shannon channel capacity**, of the given channel.

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology.

How well can channel coding be done? So called **Shannon's channel coding theorem says** that over many common channels there exist data coding schemes that are able to transmit data reliably at all code rates smaller than a certain **threshold**, called nowadays the **Shannon channel capacity**, of the given channel.

Moreover, the theorem says that probability of a decoding error can be made to decrease exponentially as the block length N of the coding scheme goes to infinity.

However, the complexity of a "naive", or straightforward, optimum decoding schemes increased exponentially with N - therefore such an optimum decoder rapidly become unfeasible.

CHANNEL CODING - BASICS

Channel coding is concerned with sending streams of data, at the highest possible rate, over a given communication channel and then obtaining the original data reliably, at the receiver side, by using encoding and decoding algorithms that are feasible to implement in available technology.

How well can channel coding be done? So called **Shannon's channel coding theorem says** that over many common channels there exist data coding schemes that are able to transmit data reliably at all code rates smaller than a certain **threshold**, called nowadays the **Shannon channel capacity**, of the given channel.

Moreover, the theorem says that probability of a decoding error can be made to decrease exponentially as the block length N of the coding scheme goes to infinity.

However, the complexity of a "naive", or straightforward, optimum decoding schemes increased exponentially with N - therefore such an optimum decoder rapidly become unfeasible.

A breakthrough came when D. Forney, in his PhD thesis in 1972, showed that so called concatenated codes could be used to achieve exponentially decreasing error probabilities at all data rates less than the Shannon channel capacity, with decoding complexity increasing only polynomially with the code length.

Therefore, the task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

Therefore, the task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

CHANNEL (STREAM) CODING II

CHANNEL (STREAM) CODING II

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**
- **tolerate more interference from the environment;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**
- **tolerate more interference from the environment;**
- **use smaller antennas;**

CHANNEL (STREAM) CODING II

Codes with lower code rate can usually correct more errors. Consequently, the communication system can:

- **operate with a lower transmit power;**
- **transmit over longer distances;**
- **tolerate more interference from the environment;**
- **use smaller antennas;**
- **transmit at a higher data rate.**

CHANNEL CAPACITY

Channel capacity of a communication channel, is the tightest upper bound on the (code) rate of information that can be reliably transmitted over that channel.

Channel capacity of a communication channel, is the tightest upper bound on the (code) rate of information that can be reliably transmitted over that channel.

CHANNEL CAPACITY

Channel capacity of a communication channel, is the tightest upper bound on the (code) rate of information that can be reliably transmitted over that channel.

Channel capacity of a communication channel, is the tightest upper bound on the (code) rate of information that can be reliably transmitted over that channel.

By the **noisy-channel Shannon coding theorem**, the channel capacity of a given channel is the limiting code rate (in units of information per unit time) that can be achieved with arbitrary small error probability.

CHANNEL CAPACITY - FORMAL DEFINITION

Let X and Y be random variables representing the input and output of a channel.

CHANNEL CAPACITY - FORMAL DEFINITION

Let X and Y be random variables representing the input and output of a channel.

Let $P_{Y|X}(y|x)$ be the conditional probability distribution function of Y given X, which can be seen as an inherent fixed probability of the communication channel.

CHANNEL CAPACITY - FORMAL DEFINITION

Let X and Y be random variables representing the input and output of a channel.

Let $P_{Y|X}(y|x)$ be the conditional probability distribution function of Y given X, which can be seen as an inherent fixed probability of the communication channel.

The joint distribution $P_{X,Y}(x,y)$ is then defined by

$$P_{X,Y}(x,y) = P_{Y|X}(y|x)P_X(x),$$

where $P_X(x)$ is the marginal distribution.

CANNEL ENCODING

CANNEL ENCODING On the other hand such codes require larger **bandwidth** and decoding is usually of higher complexity.

CANNEL ENCODING On the other hand such codes require larger bandwidth and decoding is usually of higher complexity.

The selection of the code rate involves a tradeoff between energy efficiency and bandwidth efficiency.

CANNEL ENCODING **On the other hand such codes require larger bandwidth and decoding is usually of higher complexity.**

The selection of the code rate involves a tradeoff between energy efficiency and bandwidth efficiency.

Central problem of channel encoding: encoding is usually easy, but decoding is usually hard.

APPENDIX II.

LOCALLY DECODABLE CODES -I

LOCALLY DECODABLE CODES -I

Classical error-correcting codes allow one to encode an n-bit message w into an N-bit codeword $C(w)$, in such a way that w can still be recovered even if $C(w)$ gets corrupted in a number of bits.

LOCALLY DECODABLE CODES -I

Classical error-correcting codes allow one to encode an n-bit message w into an N-bit codeword $C(w)$, in such a way that w can still be recovered even if $C(w)$ gets corrupted in a number of bits.

The disadvantage of the classical error-correcting codes is that one needs to consider all, or at least most of, the (corrupted) codeword to recover anything about w.

LOCALLY DECODABLE CODES -I

Classical error-correcting codes allow one to encode an n-bit message w into an N-bit codeword $C(w)$, in such a way that w can still be recovered even if $C(w)$ gets corrupted in a number of bits.

The disadvantage of the classical error-correcting codes is that one needs to consider all, or at least most of, the (corrupted) codeword to recover anything about w. On the other

hand so-called **locally decodable codes** allow reconstruction of any arbitrary bit w_i, from looking only at k randomly chosen bits of $C(w)$, where k is as small as 3.

LOCALLY DECODABLE CODES -I

Classical error-correcting codes allow one to encode an n-bit message w into an N-bit codeword $C(w)$, in such a way that w can still be recovered even if $C(w)$ gets corrupted in a number of bits.

The disadvantage of the classical error-correcting codes is that one needs to consider all, or at least most of, the (corrupted) codeword to recover anything about w. On the other

hand so-called **locally decodable codes** allow reconstruction of any arbitrary bit w_i, from looking only at k randomly chosen bits of $C(w)$, where k is as small as 3.

Locally decodable codes have a variety of applications in cryptography and theory of fault-tolerant computation.

LOCALLY DECODABLE CODES -II

LOCALLY DECODABLE CODES -II

Locally decodable codes have another remarkable property:

Locally decodable codes have another remarkable property:

A message can be encoded in such a way that should a small enough fraction of its symbols die in the transit, we could, with high probability, to recover the original bit anywhere in the message we choose.

LOCALLY DECODABLE CODES -II

Locally decodable codes have another remarkable property:

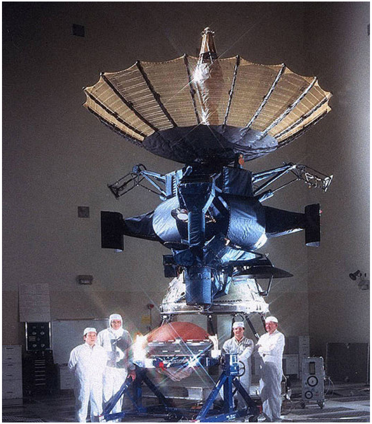
A message can be encoded in such a way that should a small enough fraction of its symbols die in the transit, we could, with high probability, to recover the original bit anywhere in the message we choose.

Moreover, this can be done by picking at random only three bits of the received message and combining them in a right way.

TURBO CODES

EXAMPLE from SPACE EXPLORATION

EXAMPLE from SPACE EXPLORATION



At the very beginning of the Galileo mission to explore Jupiter and its moons in 1989 it was discovered that primary antenna (deployed in the figure on the top) failed to deploy,

GALILEO MISSION - SOLUTION

GALILEO MISSION - SOLUTION

The primary antenna was designed to send 100, 000 b/s. Spacecraft had also another antenna, but that was capable to send only 10 b/s. The whole mission looked as being a disaster.

GALILEO MISSION - SOLUTION

The primary antenna was designed to send 100, 000 b/s. Spacecraft had also another antenna, but that was capable to send only 10 b/s. The whole mission looked as being a disaster.
A heroic engineering effort was immediately undertaken in the mission center to design the most powerful concatenated code conceived up to that time, and to program it into the spacecraft computer.

GALILEO MISSION - SOLUTION

The primary antenna was designed to send 100, 000 b/s. Spacecraft had also another antenna, but that was capable to send only 10 b/s. The whole mission looked as being a disaster.
A heroic engineering effort was immediately undertaken in the mission center to design the most powerful concatenated code conceived up to that time, and to program it into the spacecraft computer.

The inner code was a 2^{14} convolution code, decoded by the Viterbi algorithm.

GALILEO MISSION - SOLUTION

The primary antenna was designed to send 100, 000 b/s. Spacecraft had also another antenna, but that was capable to send only 10 b/s. The whole mission looked as being a disaster.
A heroic engineering effort was immediately undertaken in the mission center to design the most powerful concatenated code conceived up to that time, and to program it into the spacecraft computer.

The inner code was a 2^{14} convolution code, decoded by the Viterbi algorithm.

The outer code consisted of multiple Reed-Solomon codes of varying length.

GALILEO MISSION - SOLUTION

The primary antenna was designed to send 100, 000 b/s. Spacecraft had also another antenna, but that was capable to send only 10 b/s. The whole mission looked as being a disaster.
A heroic engineering effort was immediately undertaken in the mission center to design the most powerful concatenated code conceived up to that time, and to program it into the spacecraft computer.

The inner code was a 2^{14} convolution code, decoded by the Viterbi algorithm.

The outer code consisted of multiple Reed-Solomon codes of varying length. After all

reparations and new encodings it was possible to send up to 1000 b/s.

GALILEO MISSION - SOLUTION

The primary antenna was designed to send 100, 000 b/s. Spacecraft had also another antenna, but that was capable to send only 10 b/s. The whole mission looked as being a disaster.
A heroic engineering effort was immediately undertaken in the mission center to design the most powerful concatenated code conceived up to that time, and to program it into the spacecraft computer.

The inner code was a 2^{14} convolution code, decoded by the Viterbi algorithm.

The outer code consisted of multiple Reed-Solomon codes of varying length. After all

reparations and new encodings it was possible to send up to 1000 b/s. Mission was rescued.

GALILEO MISSION - SOLUTION

The primary antenna was designed to send 100, 000 b/s. Spacecraft had also another antenna, but that was capable to send only 10 b/s. The whole mission looked as being a disaster.
A heroic engineering effort was immediately undertaken in the mission center to design the most powerful concatenated code conceived up to that time, and to program it into the spacecraft computer.

The inner code was a 2^{14} convolution code, decoded by the Viterbi algorithm.

The outer code consisted of multiple Reed-Solomon codes of varying length. After all

reparations and new encodings it was possible to send up to 1000 b/s. Mission was rescued.

Nowadays when so called iterative decoding is used concatenation of even very simple codes can yield superb performance.

TURBO CODES

TURBO CODES

Channel coding was revolutionized by the invention of **Turbo codes**.

TURBO CODES

Channel coding was revolutionized by the invention of **Turbo codes**. Turbo codes were introduced by Berrou, Glavieux and Thitimajshima in 1993. Turbo codes are specified by special encodings.

TURBO CODES

Channel coding was revolutionized by the invention of **Turbo codes**. Turbo codes were introduced by Berrou, Glavieux and Thitimajshima in 1993. Turbo codes are specified by special encodings.

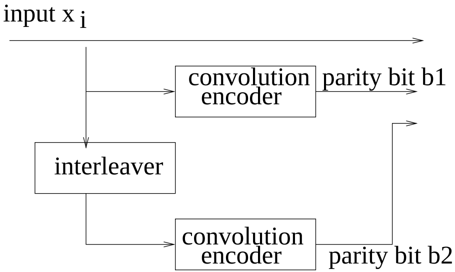
A **Turbo code** can be seen as formed from a parallel composition of two (convolution) codes separated by an **interleaver** (that permutes blocks of data in a fixed (pseudo)-random way).

TURBO CODES

Channel coding was revolutionized by the invention of **Turbo codes**. Turbo codes were introduced by Berrou, Glavieux and Thitimajshima in 1993. Turbo codes are specified by special encodings.

A **Turbo code** can be seen as formed from a parallel composition of two (convolution) codes separated by an **interleaver** (that permutes blocks of data in a fixed (pseudo)-random way).

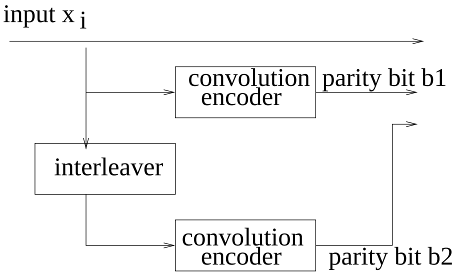
A Turbo encoder is formed from the parallel composition of two (convolution) encoders separated by an interleaver.



EXAMPLES of TURBO and CONVOLUTION ENCODERS

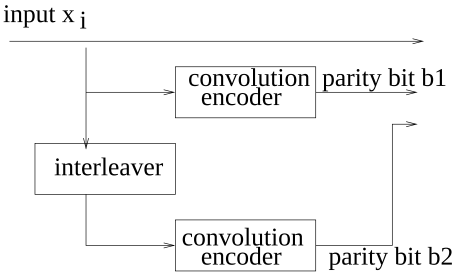
EXAMPLES of TURBO and CONVOLUTION ENCODERS

A Turbo encoder

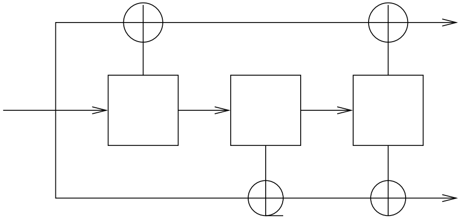


EXAMPLES of TURBO and CONVOLUTION ENCODERS

A Turbo encoder



and a convolution encoder



ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted

ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted and during the transmission symbols 7-10 are lost to get:

$$cenaje....c$$

ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted and during the transmission symbols 7-10 are lost to get:

$$cenaje....c$$

In such a case very important information was definitely lost.

ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted and during the transmission symbols 7-10 are lost to get:

$$cenaje....c$$

In such a case very important information was definitely lost.

However, if the input word is first permuted according to the permutation

$$3, 8, 7, 9, 10, 1, 2, 6, 4, 11, 5$$

ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted and during the transmission symbols 7-10 are lost to get:

$$cenaje....c$$

In such a case very important information was definitely lost.

However, if the input word is first permuted according to the permutation

$$3, 8, 7, 9, 10, 1, 2, 6, 4, 11, 5$$

then the input will be actually the word

ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted and during the transmission symbols 7-10 are lost to get:

$$cenaje....c$$

In such a case very important information was definitely lost.

However, if the input word is first permuted according to the permutation

$$3, 8, 7, 9, 10, 1, 2, 6, 4, 11, 5$$

then the input will be actually the word

$$n020kceeacj$$

ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted and during the transmission symbols 7-10 are lost to get:

$$cenaje....c$$

In such a case very important information was definitely lost.

However, if the input word is first permuted according to the permutation

$$3, 8, 7, 9, 10, 1, 2, 6, 4, 11, 5$$

then the input will be actually the word

$$n020kceeacj$$

and if the same four positions are lost the output will be

$$n020kc....j$$

ADVANTAGES of INTERLEAVING

let us assume that a word

$$cenaje200kc$$

is transmitted and during the transmission symbols 7-10 are lost to get:

$$cenaje....c$$

In such a case very important information was definitely lost.

However, if the input word is first permuted according to the permutation

$$3, 8, 7, 9, 10, 1, 2, 6, 4, 11, 5$$

then the input will be actually the word

$$n020kceeacj$$

and if the same four positions are lost the output will be

$$n020kc....j$$

However, after the inverse permutation the output actually will be

$$c.n.j.200k.$$

which is quite easy to decode correctly!!!!

DECODING and PERFORMANCE of TURBO CODES

DECODING and PERFORMANCE of TURBO CODES

- A soft-in-soft-out decoding is usually used - the decoder gets from the analog/digital demodulator a soft value of each bit - probability that it is 1 and produces only a soft-value for each bit.

DECODING and PERFORMANCE of TURBO CODES

- A soft-in-soft-out decoding is usually used - the decoder gets from the analog/digital demodulator a soft value of each bit - probability that it is 1 and produces only a soft-value for each bit.
- The overall decoder uses decoders for outputs of two encoders that also provide only soft values for bits and by exchanging information produced by two decoders and from the original input bit, the main decoder tries to increase, by an iterative process, likelihood for values of decoded bits and to produce finally hard outcome - a bit 1 or 0.

DECODING and PERFORMANCE of TURBO CODES

- A soft-in-soft-out decoding is usually used - the decoder gets from the analog/digital demodulator a soft value of each bit - probability that it is 1 and produces only a soft-value for each bit.
- The overall decoder uses decoders for outputs of two encoders that also provide only soft values for bits and by exchanging information produced by two decoders and from the original input bit, the main decoder tries to increase, by an iterative process, likelihood for values of decoded bits and to produce finally hard outcome - a bit 1 or 0. pause
- Turbo codes performance can be very close to theoretical Shannon limit.

DECODING and PERFORMANCE of TURBO CODES

- A soft-in-soft-out decoding is usually used - the decoder gets from the analog/digital demodulator a soft value of each bit - probability that it is 1 and produces only a soft-value for each bit.
- The overall decoder uses decoders for outputs of two encoders that also provide only soft values for bits and by exchanging information produced by two decoders and from the original input bit, the main decoder tries to increase, by an iterative process, likelihood for values of decoded bits and to produce finally hard outcome - a bit 1 or 0. pause
- Turbo codes performance can be very close to theoretical Shannon limit.
- This was, for example the case for UMTS (the third Generation Universal Mobile Telecommunication System) Turbo code having a less than 1.2-fold overhead. in this case the interleaver worked with blocks of 40 bits.

DECODING and PERFORMANCE of TURBO CODES

- A soft-in-soft-out decoding is usually used - the decoder gets from the analog/digital demodulator a soft value of each bit - probability that it is 1 and produces only a soft-value for each bit.
- The overall decoder uses decoders for outputs of two encoders that also provide only soft values for bits and by exchanging information produced by two decoders and from the original input bit, the main decoder tries to increase, by an iterative process, likelihood for values of decoded bits and to produce finally hard outcome - a bit 1 or 0. pause
- Turbo codes performance can be very close to theoretical Shannon limit.
- This was, for example the case for UMTS (the third Generation Universal Mobile Telecommunication System) Turbo code having a less than 1.2-fold overhead. in this case the interleaver worked with blocks of 40 bits.
- Turbo codes were incorporated into standards used by NASA for deep space communications, digital video broadcasting and both third generation cellular standards.

DECODING and PERFORMANCE of TURBO CODES

- A soft-in-soft-out decoding is usually used - the decoder gets from the analog/digital demodulator a soft value of each bit - probability that it is 1 and produces only a soft-value for each bit.
- The overall decoder uses decoders for outputs of two encoders that also provide only soft values for bits and by exchanging information produced by two decoders and from the original input bit, the main decoder tries to increase, by an iterative process, likelihood for values of decoded bits and to produce finally hard outcome - a bit 1 or 0. pause
- Turbo codes performance can be very close to theoretical Shannon limit.
- This was, for example the case for UMTS (the third Generation Universal Mobile Telecommunication System) Turbo code having a less than 1.2-fold overhead. in this case the interleaver worked with blocks of 40 bits.
- Turbo codes were incorporated into standards used by NASA for deep space communications, digital video broadcasting and both third generation cellular standards.
- Literature: M.C. Valenti and J.Sun: Turbo codes - tutorial, Handbook of RF and Wireless Technologies, 2004 - reachable by Google.

REACHING SHANNON LIMIT

- Though Shannon developed his capacity bound already in 1940, till recently code designers were unable to come with codes with performance close to theoretical limit.

REACHING SHANNON LIMIT

- Though Shannon developed his capacity bound already in 1940, till recently code designers were unable to come with codes with performance close to theoretical limit.
- In 1990 the gap between theoretical bound and practical implementations was still at best about 3dB

REACHING SHANNON LIMIT

- Though Shannon developed his capacity bound already in 1940, till recently code designers were unable to come with codes with performance close to theoretical limit.
- In 1990 the gap between theoretical bound and practical implementations was still at best about 3dB

Decibel

The decibel **dB** is a number that represents a logarithm of the ration of two values of a quantity (such as value $dB = 20\log(V_1/V2)$

A decibel is a relative measure. If E is the actual energy and E_{ref} is the theoretical lower bound, then the relative energy increase in decibels is

$$10\log_{10}\frac{E}{E_{ref}}$$

Since $\log_{10} 2 = 0.3$ a two-fold relative energy increase equals $3dB$.

- Turbo codes performance can be very close to theoretical Shannon limit.

- Turbo codes performance can be very close to theoretical Shannon limit.
- This was, for example the case for UMTS (the third Generation Universal Mobile Telecommunication System) Turbo code having a less than 1.2-fold overhead. in this case the interleaver worked with blocks of 40 bits.

- Turbo codes performance can be very close to theoretical Shannon limit.
- This was, for example the case for UMTS (the third Generation Universal Mobile Telecommunication System) Turbo code having a less than 1.2-fold overhead. in this case the interleaver worked with blocks of 40 bits.
- Turbo codes were incorporated into standards used by NASA for deep space communications, digital video broadcasting and both third generation cellular standards.

- Turbo codes performance can be very close to theoretical Shannon limit.
- This was, for example the case for UMTS (the third Generation Universal Mobile Telecommunication System) Turbo code having a less than 1.2-fold overhead. in this case the interleaver worked with blocks of 40 bits.
- Turbo codes were incorporated into standards used by NASA for deep space communications, digital video broadcasting and both third generation cellular standards.
- Literature: M.C. Valenti and J.Sun: Turbo codes - tutorial, Handbook of RF and Wireless Technologies, 2004 - reachable by Google.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.
- For decoding of Turbo codes so called soft decoding is used. **Soft decoding is an iterative process** in which each component decoder takes advantage of the work of other at the previous step, with the aid of the original concept of intrinsic information.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.
- For decoding of Turbo codes so called soft decoding is used. **Soft decoding is an iterative process** in which each component decoder takes advantage of the work of other at the previous step, with the aid of the original concept of intrinsic information.
- For sufficiently large size of interleavers, the correcting performance of turbo codes, as shown by simulations, appears to be close to the theoretical Shannon limit.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.
- For decoding of Turbo codes so called soft decoding is used. **Soft decoding is an iterative process** in which each component decoder takes advantage of the work of other at the previous step, with the aid of the original concept of intrinsic information.
- For sufficiently large size of interleavers, the correcting performance of turbo codes, as shown by simulations, appears to be close to the theoretical Shannon limit.
- Permutations performed by interleaver can often by specified by simple polynomials that make one-to-one mapping of some sets $\{0, 1, \ldots, q-1\}$.
- Turbo codes are linear codes.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.
- For decoding of Turbo codes so called soft decoding is used. **Soft decoding is an iterative process** in which each component decoder takes advantage of the work of other at the previous step, with the aid of the original concept of intrinsic information.
- For sufficiently large size of interleavers, the correcting performance of turbo codes, as shown by simulations, appears to be close to the theoretical Shannon limit.
- Permutations performed by interleaver can often by specified by simple polynomials that make one-to-one mapping of some sets $\{0, 1, \ldots, q-1\}$.
- Turbo codes are linear codes.
- A "good" linear code is one that has mostly high-weight codewords.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.
- For decoding of Turbo codes so called soft decoding is used. **Soft decoding is an iterative process** in which each component decoder takes advantage of the work of other at the previous step, with the aid of the original concept of intrinsic information.
- For sufficiently large size of interleavers, the correcting performance of turbo codes, as shown by simulations, appears to be close to the theoretical Shannon limit.
- Permutations performed by interleaver can often by specified by simple polynomials that make one-to-one mapping of some sets $\{0, 1, \ldots, q-1\}$.
- Turbo codes are linear codes.
- A "good" linear code is one that has mostly high-weight codewords.
- High-weight codewords are desirable because they are more distinct and the decoder can more easily distinguish among them.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.
- For decoding of Turbo codes so called soft decoding is used. **Soft decoding is an iterative process** in which each component decoder takes advantage of the work of other at the previous step, with the aid of the original concept of intrinsic information.
- For sufficiently large size of interleavers, the correcting performance of turbo codes, as shown by simulations, appears to be close to the theoretical Shannon limit.
- Permutations performed by interleaver can often by specified by simple polynomials that make one-to-one mapping of some sets $\{0, 1, \dots, q-1\}$.
- Turbo codes are linear codes.
- A "good" linear code is one that has mostly high-weight codewords.
- High-weight codewords are desirable because they are more distinct and the decoder can more easily distinguish among them.
- A big advantage of Turbo encoders is that they reduce the number of low-weight codewords because their output is the sum of the weights of the input and two parity output bits.

TURBO CODES - SUMMARY

- Turbo codes encoding devices are usually built from two (usually identical) recursive systematic convolution encoders, linked together by nonuniform interleaver (permutation) devices.
- For decoding of Turbo codes so called soft decoding is used. **Soft decoding is an iterative process** in which each component decoder takes advantage of the work of other at the previous step, with the aid of the original concept of intrinsic information.
- For sufficiently large size of interleavers, the correcting performance of turbo codes, as shown by simulations, appears to be close to the theoretical Shannon limit.
- Permutations performed by interleaver can often by specified by simple polynomials that make one-to-one mapping of some sets $\{0, 1, \ldots, q-1\}$.
- Turbo codes are linear codes.
- A "good" linear code is one that has mostly high-weight codewords.
- High-weight codewords are desirable because they are more distinct and the decoder can more easily distinguish among them.
- A big advantage of Turbo encoders is that they reduce the number of low-weight codewords because their output is the sum of the weights of the input and two parity output bits.
- A turbo code can be seen as a refinement of concatenated codes plus an iterative algorithm for decoding.