Part I

Cyclic, stream and channel codes. Speccial decoding

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes.

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.
- They **have extremely concise specifications**.

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.
- They **have extremely concise specifications**.
- Their encodings **can be efficiently implemented** using simple machinery - shift registers.

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.
- They **have extremely concise specifications**.
- Their encodings **can be efficiently implemented** using simple machinery - shift registers.
- Many of the practically very important codes are cyclic.

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.
- They **have extremely concise specifications**.
- Their encodings **can be efficiently implemented** using simple machinery - shift registers.
- Many of the practically very important codes are cyclic.

2. Channel codes are used to **encode streams of data** (bits).

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.
- They **have extremely concise specifications**.
- Their encodings **can be efficiently implemented** using simple machinery - shift registers.
- Many of the practically very important codes are cyclic.

2. Channel codes are used to **encode streams of data** (bits). Some of them, as **Concatenated codes** and **Turbo codes**, **reach theoretical Shannon bound concerning efficiency**, and are currently used very often in practice.

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.
- They **have extremely concise specifications**.
- Their encodings **can be efficiently implemented** using simple machinery - shift registers.
- Many of the practically very important codes are cyclic.

2. Channel codes are used to **encode streams of data** (bits). Some of them, as **Concatenated codes** and **Turbo codes**, **reach theoretical Shannon bound concerning efficiency**, and are currently used very often in practice.

3. List decoding is a new decoding technique capable to deal, in an approximate way, with cases that many errors occur, and in many such cases this technique performs better than the classical **unique decoding** technique - the one we dealt with so far.

CHAPTER 3: CYCLIC, STREAM and CHANNEL CODES - SPECIAL DECODINGS

1. Cyclic codes are very special linear codes. They are of large interest and importance for several reasons:

- They **posses a rich algebraic structure** that can be utilized in a variety of ways.
- They **have extremely concise specifications**.
- Their encodings **can be efficiently implemented** using simple machinery - shift registers.
- Many of the practically very important codes are cyclic.

2. Channel codes are used to **encode streams of data** (bits). Some of them, as **Concatenated codes** and **Turbo codes**, **reach theoretical Shannon bound concerning efficiency**, and are currently used very often in practice.

3. List decoding is a new decoding technique capable to deal, in an approximate way, with cases that many errors occur, and in many such cases this technique performs better than the classical **unique decoding** technique - the one we dealt with so far.

4. Locally decodable codes can be seen as a theoretical extreme of coding theory with deep theoretical implications.

IMPORTANT NOTE

IMPORTANT NOTE

In order to specify a non-linear binary code with 2^k codewords of length n one may need to write down

$$2^k$$

codewords of length n.

IMPORTANT NOTE

In order to specify a non-linear binary code with 2^k codewords of length n one may need to write down

$$2^k$$

codewords of length n.

In order to specify a linear binary code of the dimension k with 2^k codewords of length n it is sufficient to write down

$$k$$

codewords of length n of a generator matrix (of a basis) of that code.

IMPORTANT NOTE

In order to specify a non-linear binary code with 2^k codewords of length n one may need to write down

$$2^k$$

codewords of length n.

In order to specify a linear binary code of the dimension k with 2^k codewords of length n it is sufficient to write down

$$k$$

codewords of length n of a generator matrix (of a basis) of that code.

In order to specify a binary cyclic code with 2^k codewords of length n it is sufficient to write down

$$1$$

codeword of length n - a generator codeword of the code C.

BASIC DEFINITION AND EXAMPLES

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

- (i) C is a linear code;
- (ii) any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \dots a_{n-1} \in C$, then also $a_{n-1}a_0 \dots a_{n-2} \in C$ and $a_1 a_2 \dots a_{n-1} a_0 \in C$.

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

- (i) **C is a linear code;**
- (ii) **any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \ldots a_{n-1} \in C$, then also $a_{n-1}a_0 \ldots a_{n-2} \in C$ and $a_1 a_2 \ldots a_{n-1}a_0 \in C$.**

Example

- (i) Code $C = \{000, 101, 011, 110\}$ is cyclic.

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

- (i) **C is a linear code;**
- (ii) **any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \ldots a_{n-1} \in C$, then also $a_{n-1}a_0 \ldots a_{n-2} \in C$ and $a_1 a_2 \ldots a_{n-1}a_0 \in C$.**

Example

- (i) Code $C = \{000, 101, 011, 110\}$ is cyclic.
- (ii) Hamming code $Ham(3, 2)$: with the generator matrix

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

- (i) **C is a linear code;**
- (ii) **any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \ldots a_{n-1} \in C$, then also $a_{n-1}a_0 \ldots a_{n-2} \in C$ and $a_1 a_2 \ldots a_{n-1}a_0 \in C$.**

Example

- (i) Code $C = \{000, 101, 011, 110\}$ is cyclic.
- (ii) Hamming code $Ham(3, 2)$: with the generator matrix

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

is equivalent to a cyclic code.

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

- (i) **C is a linear code;**
- (ii) **any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \dots a_{n-1} \in C$, then also $a_{n-1}a_0 \dots a_{n-2} \in C$ and $a_1 a_2 \dots a_{n-1}a_0 \in C$.**

Example

- (i) Code $C = \{000, 101, 011, 110\}$ is cyclic.
- (ii) Hamming code $Ham(3, 2)$: with the generator matrix

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

is equivalent to a cyclic code.

- (iii) The binary linear code $\{0000, 1001, 0110, 1111\}$ is not cyclic, but it is equivalent to a cyclic code.

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

- (i) **C is a linear code;**
- (ii) **any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \ldots a_{n-1} \in C$, then also $a_{n-1}a_0 \ldots a_{n-2} \in C$ and $a_1a_2 \ldots a_{n-1}a_0 \in C$.**

Example

- (i) Code $C = \{000, 101, 011, 110\}$ is cyclic.
- (ii) Hamming code $Ham(3, 2)$: with the generator matrix

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

is equivalent to a cyclic code.

- (iii) The binary linear code $\{0000, 1001, 0110, 1111\}$ is not cyclic, but it is equivalent to a cyclic code. – to get a cyclic code exchange first two symbols in all codewords.

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

- (i) **C is a linear code;**
- (ii) **any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \dots a_{n-1} \in C$, then also $a_{n-1}a_0 \dots a_{n-2} \in C$ and $a_1 a_2 \dots a_{n-1}a_0 \in C$.**

Example

- (i) Code $C = \{000, 101, 011, 110\}$ is cyclic.
- (ii) Hamming code $Ham(3,2)$: with the generator matrix

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

is equivalent to a cyclic code.

- (iii) The binary linear code $\{0000, 1001, 0110, 1111\}$ is not cyclic, but it is equivalent to a cyclic code. – to get a cyclic code exchange first two symbols in all codewords.
- (iv) Is Hamming code $Ham(2,3)$ with the generator matrix

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

- (a) cyclic?
- (b) or at least equivalent to a cyclic code?

FREQUENCY of CYCLIC CODES

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce.

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce. For example, there are 11 811 linear [7,3] binary codes, but only two of them are non-trivial cyclic codes.

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce. For example, there are 11 811 linear [7,3] binary codes, but only two of them are non-trivial cyclic codes.

Trivial cyclic codes. For any field F and any integer $n \geq 3$ there are always cyclic the following codes with codewords of the length n over F:

- **No-information code** - code consisting of just one all-zero codeword.

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce. For example, there are 11 811 linear [7,3] binary codes, but only two of them are non-trivial cyclic codes.

Trivial cyclic codes. For any field F and any integer $n \geq 3$ there are always cyclic the following codes with codewords of the length n over F:

- **No-information code** - code consisting of just one all-zero codeword.
- **Repetition code** - code consisting of all codewords (a, a, . . . ,a) for $a \in F$.

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce. For example, there are 11 811 linear [7,3] binary codes, but only two of them are non-trivial cyclic codes.

Trivial cyclic codes. For any field F and any integer $n \geq 3$ there are always cyclic the following codes with codewords of the length n over F:

- **No-information code** - code consisting of just one all-zero codeword.
- **Repetition code** - code consisting of all codewords (a, a, ...,a) for $a \in F$.
- **Single-parity-check code** - code consisting of all codewords with parity 0.

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce. For example, there are 11 811 linear [7,3] binary codes, but only two of them are non-trivial cyclic codes.

Trivial cyclic codes. For any field F and any integer $n \geq 3$ there are always cyclic the following codes with codewords of the length n over F:

- **No-information code** - code consisting of just one all-zero codeword.
- **Repetition code** - code consisting of all codewords (a, a, . . . ,a) for $a \in F$.
- **Single-parity-check code** - code consisting of all codewords with parity 0.
- **No-parity code** - code consisting of all codewords of length n

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce. For example, there are 11 811 linear [7,3] binary codes, but only two of them are non-trivial cyclic codes.

Trivial cyclic codes. For any field F and any integer $n \geq 3$ there are always cyclic the following codes with codewords of the length n over F:

- **No-information code** - code consisting of just one all-zero codeword.
- **Repetition code** - code consisting of all codewords (a, a, . . . ,a) for $a \in F$.
- **Single-parity-check code** - code consisting of all codewords with parity 0.
- **No-parity code** - code consisting of all codewords of length n

For some cases, for example for $n = 19$ and $F = GF(2)$, the above four trivial cyclic codes are the only cyclic codes.

AN EXAMPLE of a CYCLIC CODE

AN EXAMPLE of a CYCLIC CODE

Is the linear code with the following generator matrix cyclic?

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

AN EXAMPLE of a CYCLIC CODE

Is the linear code with the following generator matrix cyclic?

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

It is.

AN EXAMPLE of a CYCLIC CODE

Is the linear code with the following generator matrix cyclic?

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

It is. It has, in addition to the codeword 0000000, the following codewords

$$c_1 = 1011100 \qquad c_2 = 0101110 \qquad c_3 = 0010111$$

$$c_1 + c_2 = 1110010 \qquad c_1 + c_3 = 1001011 \qquad c_2 + c_3 = 0111001$$

$$c_1 + c_2 + c_3 = 1100101$$

AN EXAMPLE of a CYCLIC CODE

Is the linear code with the following generator matrix cyclic?

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

It is. It has, in addition to the codeword 0000000, the following codewords

$$c_1 = 1011100 \qquad c_2 = 0101110 \qquad c_3 = 0010111$$
$$c_1 + c_2 = 1110010 \qquad c_1 + c_3 = 1001011 \qquad c_2 + c_3 = 0111001$$
$$c_1 + c_2 + c_3 = 1100101$$

and it is cyclic because the right shifts have the following impacts

$$c_1 \rightarrow c_2, \qquad c_2 \rightarrow c_3, \qquad c_3 \rightarrow c_1 + c_3$$
$$c_1 + c_2 \rightarrow c_2 + c_3, \qquad c_1 + c_3 \rightarrow c_1 + c_2 + c_3, \qquad c_2 + c_3 \rightarrow c_1$$
$$c_1 + c_2 + c_3 \rightarrow c_1 + c_2$$

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted by

$$a_0 a_1 \ldots a_{n-1}$$

and to each such a codeword the **polynomial**

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

is usually associated

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted by

$$a_0 a_1 \dots a_{n-1}$$

and to each such a codeword the **polynomial**

$$a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

is usually associated – **an ingenious idea**!!.

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted by

$$a_0 a_1 \ldots a_{n-1}$$

and to each such a codeword the **polynomial**

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

is usually associated – **an ingenious idea**!!.

NOTATION: $F_q[x]$ will denote the set of all polynomials $f(x)$ over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted by

$$a_0 a_1 \ldots a_{n-1}$$

and to each such a codeword the **polynomial**

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

is usually associated – **an ingenious idea**!!.

NOTATION: $F_q[x]$ will denote the set of all polynomials $f(x)$ over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

Multiplication of polynomials If $f(x)$, $g(x) \in F_q[x]$, then

$$deg(f(x)g(x)) = deg(f(x)) + deg(g(x)).$$

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted by

$$a_0 a_1 \ldots a_{n-1}$$

and to each such a codeword the **polynomial**

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

is usually associated – **an ingenious idea**!!.

NOTATION: $F_q[x]$ will denote the set of all polynomials $f(x)$ over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

Multiplication of polynomials If $f(x)$, $g(x) \in F_q[x]$, then

$$deg(f(x)g(x)) = deg(f(x)) + deg(g(x)).$$

Division of polynomials For every pair of polynomials $a(x)$, $b(x) \neq 0$ in $F_q[x]$ there exists a unique pair of polynomials $q(x)$, $r(x)$ in $F_q[x]$ such that

$$a(x) = q(x)b(x) + r(x), deg(r(x)) < deg(b(x)).$$

Example Divide $x^3 + x + 1$ by $x^2 + x + 1$ in $F_2[x]$.

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted by

$$a_0 a_1 \dots a_{n-1}$$

and to each such a codeword the **polynomial**

$$a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

is usually associated – **an ingenious idea**!!.

NOTATION: $F_q[x]$ will denote the set of all polynomials $f(x)$ over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

Multiplication of polynomials If $f(x)$, $g(x) \in F_q[x]$, then

$$deg(f(x)g(x)) = deg(f(x)) + deg(g(x)).$$

Division of polynomials For every pair of polynomials $a(x)$, $b(x) \neq 0$ in $F_q[x]$ there exists a unique pair of polynomials $q(x)$, $r(x)$ in $F_q[x]$ such that

$$a(x) = q(x)b(x) + r(x), deg(r(x)) < deg(b(x)).$$

Example Divide $x^3 + x + 1$ by $x^2 + x + 1$ in $F_2[x]$.

Definition Let $f(x)$ be a fixed polynomial in $F_q[x]$. Two polynomials $g(x)$, $h(x)$ are said to be **congruent modulo** $f(x)$, notation

$$g(x) \equiv h(x) \pmod{f(x)},$$

if $g(x) - h(x)$ is divisible by $f(x)$.

EXAMPLES

If binary strings of length 7 are considered then

to the word 1010101 the following polynomial is associated

EXAMPLES

If binary strings of length 7 are considered then

to the word 1010101 the following polynomial is associated $1 + x^2 + x^4 + x^6$

EXAMPLES

If binary strings of length 7 are considered then

to the word 1010101 the following polynomial is associated $1 + x^2 + x^4 + x^6$

to the word 1000001 the following polynomial is associated:

EXAMPLES

If binary strings of length 7 are considered then

to the word 1010101 the following polynomial is associated $1 + x^2 + x^4 + x^6$

to the word 1000001 the following polynomial is associated: $1 + x^6$

EXAMPLES

If binary strings of length 7 are considered then

to the word 1010101 the following polynomial is associated $1 + x^2 + x^4 + x^6$

to the word 1000001 the following polynomial is associated: $1 + x^6$

The word starting with 2^{124} zeros and followed by one 1 has the polynomial representation:

$$x^{124}$$

In the alphabet $\{0, 1, 2\}$ $2x^2$ represents the string 002

EXAMPLE

EXAMPLE

If $x^3 + x + 1$ is divided by $x^2 + x + 1$, then

EXAMPLE

If $x^3 + x + 1$ is divided by $x^2 + x + 1$, then

$$x^3 + x + 1 = (x^2 + x + 1)(x - 1) + x$$

EXAMPLE

If $x^3 + x + 1$ is divided by $x^2 + x + 1$, then

$$x^3 + x + 1 = (x^2 + x + 1)(x - 1) + x$$

and therefore the result of the division is

EXAMPLE

If $x^3 + x + 1$ is divided by $x^2 + x + 1$, then

$$x^3 + x + 1 = (x^2 + x + 1)(x - 1) + x$$

and therefore the result of the division is

$$x - 1$$

and the remainder is

EXAMPLE

If $x^3 + x + 1$ is divided by $x^2 + x + 1$, then

$$x^3 + x + 1 = (x^2 + x + 1)(x - 1) + x$$

and therefore the result of the division is

$$x - 1$$

and the remainder is

$$x.$$

NOTICE

NOTICE

A **code** C of the words of length **n**

NOTICE

A **code** C of the words of length **n**
is a set of codewords of length n

$$a_0 a_1 a_2 \ldots a_{n-1}$$

NOTICE

A **code** C of the words of length **n** **is a set of codewords** of length n

$$a_0 a_1 a_2 \dots a_{n-1}$$

or C can be seen as

NOTICE

A **code** C of the words of length **n**
is a set of codewords of length n

$$a_0 a_1 a_2 \ldots a_{n-1}$$

or C can be seen as
a set of polynomials of the degree (at most) $n-1$

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

APPENDIX - III.

GROUPS

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

A group G is called an **Abelian group** if the operation $*$ is commutative, that is $a * b = b * a$ for any $a, b \in G$.

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

A group G is called an **Abelian group** if the operation $*$ is commutative, that is $a * b = b * a$ for any $a, b \in G$.

Example Which of the following sets is an (Abelian) group:

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

A group G is called an **Abelian group** if the operation $*$ is commutative, that is $a * b = b * a$ for any $a, b \in G$.

Example Which of the following sets is an (Abelian) group:

- The set of real numbers with operation $*$ being: (a) addition;

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

A group G is called an **Abelian group** if the operation $*$ is commutative, that is $a * b = b * a$ for any $a, b \in G$.

Example Which of the following sets is an (Abelian) group:

- The set of real numbers with operation $*$ being: (a) addition; (b) multiplication.

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

A group G is called an **Abelian group** if the operation $*$ is commutative, that is $a * b = b * a$ for any $a, b \in G$.

Example Which of the following sets is an (Abelian) group:

- The set of real numbers with operation $*$ being: (a) addition; (b) multiplication.
- The set of matrices of degree n and operation: (a) addition;

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

A group G is called an **Abelian group** if the operation $*$ is commutative, that is $a * b = b * a$ for any $a, b \in G$.

Example Which of the following sets is an (Abelian) group:

- The set of real numbers with operation $*$ being: (a) addition; (b) multiplication.
- The set of matrices of degree n and operation: (a) addition; (b) multiplication.

GROUPS

A **group** G is a set of elements and an operation, call it *, with the following properties:

- G is closed under *; that is if $a, b \in G$, so is $a * b$.
- The operation * is associative, that is $a * (b * c) = (a * b) * c$, for any $a, b, c \in G$.
- G has an identity e element such that $e * a = a * e = a$ for any $a \in G$.
- Every element $a \in G$ has an inverse $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.

A group G is called an **Abelian group** if the operation $*$ is commutative, that is $a * b = b * a$ for any $a, b \in G$.

Example Which of the following sets is an (Abelian) group:

- The set of real numbers with operation $*$ being: (a) addition; (b) multiplication.
- The set of matrices of degree n and operation: (a) addition; (b) multiplication.
- What happens if we consider only matrices with determinants not equal zero?

RINGS and FIELDS

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.
- R has an identity element 1 for multiplication

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.
- R has an identity element 1 for multiplication
- The distributive law holds: $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in R$.

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.
- R has an identity element 1 for multiplication
- The distributive law holds: $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in R$.

A ring is called a **commutative ring** if multiplication is commutative.

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.
- R has an identity element 1 for multiplication
- The distributive law holds: $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in R$.

A ring is called a **commutative ring** if multiplication is commutative.

A **field** F is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , with the following properties:

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.
- R has an identity element 1 for multiplication
- The distributive law holds: $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in R$.

A ring is called a **commutative ring** if multiplication is commutative.

A **field** F is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , with the following properties:

- F is a commutative ring.

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.
- R has an identity element 1 for multiplication
- The distributive law holds: $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in R$.

A ring is called a **commutative ring** if multiplication is commutative.

A **field** F is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , with the following properties:

- F is a commutative ring.
- Non-zero elements of F form an Abelian group under multiplication.

RINGS and FIELDS

A **ring** R is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , having the following properties:

- R is closed under both operations $+$ and $\cdot$.
- R is an Abelian group under $+$ (with a unity element for addition called **zero**).
- The associative law for multiplication holds.
- R has an identity element 1 for multiplication
- The distributive law holds: $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in R$.

A ring is called a **commutative ring** if multiplication is commutative.

A **field** F is a set with two operations $+$ (addition) and $\cdot$ (multiplication) , with the following properties:

- F is a commutative ring.
- Non-zero elements of F form an Abelian group under multiplication.

A non-zero element g is a **primitive element** of a field F if all non-zero elements of F are powers of g.

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate $(x+1)^2$ in $F_2[x]/(x^2+x+1)$.**

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate $(x+1)^2$ in $F_2[x]/(x^2+x+1)$.** It holds

$$(x+1)^2 =$$

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2 + 2x + 1 \equiv$$

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2 + 2x + 1 \equiv x^2 + 1 \equiv$$

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2 + 2x + 1 \equiv x^2 + 1 \equiv x \pmod{x^2+x+1}.$$

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2 + 2x + 1 \equiv x^2 + 1 \equiv x \pmod{x^2 + x + 1}.$$

How many elements has $F_q[x]/f(x)$?

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2 + 2x + 1 \equiv x^2 + 1 \equiv x \pmod{x^2 + x + 1}.$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2+2x+1 \equiv x^2+1 \equiv x \pmod{x^2+x+1}.$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

Example: Addition and multiplication tables for $F_2[x]/(x^2+x+1)$

+	0	1	x	1+x
0	0	1	x	1+x
1	1	0	1+x	x
x	x	1+x	0	1
1+x	1+x	x	1	0

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2 + 2x + 1 \equiv x^2 + 1 \equiv x \pmod{x^2 + x + 1}.$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

Example: Addition and multiplication tables for $F_2[x]/(x^2+x+1)$

+	0	1	x	1+x
0	0	1	x	1+x
1	1	0	1+x	x
x	x	1+x	0	1
1+x	1+x	x	1	0

•	0	1	x	1+x
0	0	0	0	0
1	0	1	x	1+x
x	0	x	1+x	1
1+x	0	1+x	1	x

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2+2x+1 \equiv x^2+1 \equiv x \pmod{x^2+x+1}.$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

Example: Addition and multiplication tables for $F_2[x]/(x^2+x+1)$

+	0	1	x	1+x
0	0	1	x	1+x
1	1	0	1+x	x
x	x	1+x	0	1
1+x	1+x	x	1	0

•	0	1	x	1+x
0	0	0	0	0
1	0	1	x	1+x
x	0	x	1+x	1
1+x	0	1+x	1	x

Definition: A polynomial $f(x)$ in $F_q[x]$ is said to be **reducible** if $f(x) = a(x)b(x)$, where $a(x),\ b(x) \in F_q[x]$ and

$$deg(a(x)) < deg(f(x)), \qquad deg(b(x)) < deg(f(x)).$$

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example: **Calculate** $(x+1)^2$ **in** $F_2[x]/(x^2+x+1)$**.** It holds

$$(x+1)^2 = x^2+2x+1 \equiv x^2+1 \equiv x \pmod{x^2+x+1}.$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

Example: Addition and multiplication tables for $F_2[x]/(x^2+x+1)$

+	0	1	x	1+x
0	0	1	x	1+x
1	1	0	1+x	x
x	x	1+x	0	1
1+x	1+x	x	1	0

•	0	1	x	1+x
0	0	0	0	0
1	0	1	x	1+x
x	0	x	1+x	1
1+x	0	1+x	1	x

Definition: A polynomial $f(x)$ in $F_q[x]$ is said to be **reducible** if $f(x) = a(x)b(x)$, where $a(x), b(x) \in F_q[x]$ and

$$deg(a(x)) < deg(f(x)), \qquad deg(b(x)) < deg(f(x)).$$

If $f(x)$ is not reducible, then it is said to be **irreducible** in $F_q[x]$.

Theorem The ring $F_q[x]/f(x)$ is a field if $f(x)$ is irreducible in $F_q[x]$.

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 \pmod{(x^n - 1)}$ we can compute, for any polynomial $f(x)$, $f(x) \bmod (x^n - 1)$ by replacing, in $f(x)$, x^n by

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 \pmod{(x^n - 1)}$ we can compute, for any polynomial $f(x)$, $f(x)$ mod $(x^n - 1)$ by replacing, in $f(x)$, x^n by 1, x^{n+1} by

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 \pmod{(x^n - 1)}$ we can compute, for any polynomial $f(x)$, $f(x) \bmod (x^n - 1)$ by replacing, in $f(x)$, x^n by 1, x^{n+1} by x, x^{n+2} by

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 \pmod{(x^n - 1)}$ we can compute, for any polynomial $f(x)$, $f(x) \bmod (x^n - 1)$ by replacing, in $f(x)$, x^n by 1, x^{n+1} by x, x^{n+2} by x^2, x^{n+3} by

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 \pmod{(x^n - 1)}$ we can compute, for any polynomial $f(x)$, $f(x)$ mod $(x^n - 1)$ by replacing, in $f(x)$, x^n by 1, x^{n+1} by x, x^{n+2} by x^2, x^{n+3} by x^3, ...

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 \pmod{(x^n - 1)}$ we can compute, for any polynomial $f(x)$, $f(x)$ mod $(x^n - 1)$ by replacing, in $f(x)$, x^n by 1, x^{n+1} by x, x^{n+2} by x^2, x^{n+3} by x^3, ...

Replacement of a word

$$w = a_0 a_1 \ldots a_{n-1}$$

by a polynomial

$$p(w) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

is of large importance because

RING (Factor ring) $R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the ring $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 \pmod{(x^n - 1)}$ we can compute, for any polynomial $f(x)$, $f(x)$ mod $(x^n - 1)$ by replacing, in $f(x)$, x^n by 1, x^{n+1} by x, x^{n+2} by x^2, x^{n+3} by x^3, ...

Replacement of a word

$$w = a_0 a_1 \ldots a_{n-1}$$

by a polynomial

$$p(w) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

is of large importance because

multiplication of $p(w)$ by x in R_n corresponds to a single cyclic shift of w. Indeed,

$$x(a_0 + a_1 x + \ldots a_{n-1} x^{n-1}) = a_{n-1} + a_0 x + a_1 x^2 + \ldots + a_{n-2} x^{n-1}$$

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1) Let C be a cyclic code.

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1 Let C be a cyclic code. C is linear $\Rightarrow$

(i) holds.

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1) Let C be a cyclic code. C is linear $\Rightarrow$

(i) holds.

(ii)

$$\textit{If } a(x) \in C, r(x) = r_0 + r_1 x + \ldots + r_{n-1}x^{n-1} \textit{then}$$
$$r(x)a(x) = r_0 a(x) + r_1 x a(x) + \ldots + r_{n-1}x^{n-1}a(x)$$

is in C by (i) because all summons above are cyclic shifts of $a(x)$.

(2) Let (i) and (ii) hold

- Taking $r(x)$ to be a scalar the conditions (i) and (ii) imply linearity of C.
- Taking $r(x) = x$ the conditions (i) and (ii) imply cyclicity of C.

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

- (i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$
- (ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

1. Let C be a cyclic code. C is linear $\Rightarrow$
   - (i) holds.
   - (ii)

$$\begin{aligned} \textit{If } \; a(x) \in C, r(x) = r_0 + r_1 x + \ldots + r_{n-1}x^{n-1} \textit{then} \\ r(x)a(x) = r_0 a(x) + r_1 x a(x) + \ldots + r_{n-1}x^{n-1}a(x) \end{aligned}$$

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1) Let C be a cyclic code. C is linear $\Rightarrow$

  (i) holds.

  (ii)

$$\begin{aligned} \mathit{If}\ \ a(x) \in C, r(x) = r_0 + r_1 x + \ldots + r_{n-1} x^{n-1} \mathit{then} \\ r(x)a(x) = r_0 a(x) + r_1 x a(x) + \ldots + r_{n-1} x^{n-1} a(x) \end{aligned}$$

is in C by (i)

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1) Let C be a cyclic code. C is linear $\Rightarrow$

(i) holds.

(ii)

$$\begin{aligned} &If\ \ a(x) \in C, r(x) = r_0 + r_1 x + \ldots + r_{n-1}x^{n-1} then \\ &\qquad r(x)a(x) = r_0 a(x) + r_1 x a(x) + \ldots + r_{n-1}x^{n-1}a(x) \end{aligned}$$

is in C by (i) because all summons above are cyclic shifts of $a(x)$.

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1) Let C be a cyclic code. C is linear $\Rightarrow$

 (i) holds.

 (ii)

$$\begin{aligned} \textit{If } \ a(x) \in C, r(x) = r_0 + r_1 x + \ldots + r_{n-1}x^{n-1} \textit{then} \\ r(x)a(x) = r_0 a(x) + r_1 x a(x) + \ldots + r_{n-1}x^{n-1}a(x) \end{aligned}$$

is in C by (i) because all summons above are cyclic shifts of $a(x)$.

(2) Let (i) and (ii) hold

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1) Let C be a cyclic code. C is linear $\Rightarrow$

  (i) holds.

  (ii)

$$\textit{If } a(x) \in C, r(x) = r_0 + r_1x + \ldots + r_{n-1}x^{n-1} \textit{then}$$
$$r(x)a(x) = r_0a(x) + r_1xa(x) + \ldots + r_{n-1}x^{n-1}a(x)$$

is in C by (i) because all summons above are cyclic shifts of $a(x)$.

(2) Let (i) and (ii) hold

- Taking $r(x)$ to be a scalar the conditions (i) and (ii) imply linearity of C.

An ALGEBRAIC SPECIFICATION of CYCLIC CODES

Theorem A binary code C of words of length n is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

1. Let C be a cyclic code. C is linear $\Rightarrow$
   - (i) holds.
   - (ii)

$$\begin{aligned} \mathit{If}\ \ a(x) \in C, r(x) = r_0 + r_1x + \ldots + r_{n-1}x^{n-1} \mathit{then} \\ r(x)a(x) = r_0a(x) + r_1xa(x) + \ldots + r_{n-1}x^{n-1}a(x) \end{aligned}$$

   is in C by (i) because all summons above are cyclic shifts of $a(x)$.
2. Let (i) and (ii) hold
   - Taking $r(x)$ to be a scalar the conditions (i) and (ii) imply linearity of C.
   - Taking $r(x) = x$ the conditions (i) and (ii) imply cyclicity of C.

OBSERVATION

OBSERVATION

- There are also non-linear codes that have cyclicity property.

OBSERVATION

- There are also non-linear codes that have cyclicity property.
- A code equivalent to a cyclic code need not be cyclic itself.

OBSERVATION

- There are also non-linear codes that have cyclicity property.
- A code equivalent to a cyclic code need not be cyclic itself.
- For instance, there are 30 distinct binary $[7, 4]$ Hamming codes, but only two of them are cyclic.

CONSTRUCTION of CYCLIC CODES

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{ r(x)f(x) \mid r(x) \in R_n \}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x)\rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x)\rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

- (i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

- (i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then

$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x)\rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x)\rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

(i) If $a(x)f(x) \in \langle f(x)\rangle$ and also $b(x)f(x) \in \langle f(x)\rangle$, then

$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x)\rangle$$

(ii) If $a(x)f(x) \in \langle f(x)\rangle$, $r(x) \in R_n$, then

$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x)\rangle$$

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{ r(x)f(x) \mid r(x) \in R_n \}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

(i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then

$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$

(ii) If $a(x)f(x) \in \langle f(x) \rangle$, $r(x) \in R_n$, then

$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x) \rangle$$

Example let $C = \langle 1 + x^2 \rangle$, $n = 3$, $q = 2$.

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{ r(x)f(x) \mid r(x) \in R_n \}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

- (i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then

$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$

- (ii) If $a(x)f(x) \in \langle f(x) \rangle$, $r(x) \in R_n$, then

$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x) \rangle$$

Example let $C = \langle 1 + x^2 \rangle$, $n = 3$, $q = 2$.
In order to determine C we have to compute $r(x)(1 + x^2)$ for all $r(x) \in R_3$.

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

- (i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then
$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$
- (ii) If $a(x)f(x) \in \langle f(x) \rangle$, $r(x) \in R_n$, then
$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x) \rangle$$

Example let $C = \langle 1 + x^2 \rangle$, $n = 3$, $q = 2$.
In order to determine C we have to compute $r(x)(1 + x^2)$ for all $r(x) \in R_3$.

$$R_3 = \{0, 1, x, 1 + x, x^2, 1 + x^2, x + x^2, 1 + x + x^2\}.$$

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{ r(x)f(x) \mid r(x) \in R_n \}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

- (i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then
$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$
- (ii) If $a(x)f(x) \in \langle f(x) \rangle$, $r(x) \in R_n$, then
$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x) \rangle$$

Example let $C = \langle 1 + x^2 \rangle$, $n = 3$, $q = 2$.
In order to determine C we have to compute $r(x)(1 + x^2)$ for all $r(x) \in R_3$.

$$R_3 = \{0, 1, x, 1 + x, x^2, 1 + x^2, x + x^2, 1 + x + x^2\}.$$

Result

$$C = \{0, 1 + x, 1 + x^2, x + x^2\}$$

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) to be a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

- (i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then
$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$
- (ii) If $a(x)f(x) \in \langle f(x) \rangle$, $r(x) \in R_n$, then
$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x) \rangle$$

Example let $C = \langle 1 + x^2 \rangle$, $n = 3$, $q = 2$.
In order to determine C we have to compute $r(x)(1 + x^2)$ for all $r(x) \in R_3$.

$$R_3 = \{0, 1, x, 1 + x, x^2, 1 + x^2, x + x^2, 1 + x + x^2\}.$$

Result

$$C = \{0, 1 + x, 1 + x^2, x + x^2\}$$
$$C = \{000, 110, 101, 011\}$$

CHARACTERIZATION THEOREM for CYCLIC CODES

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem **Let C be a non-zero cyclic code in R_n. Then**

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

- (i) Suppose $g(x)$ and $h(x)$ are two monic polynomials in C of the smallest degree, say **d**.

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

(i) Suppose $g(x)$ and $h(x)$ are two monic polynomials in C of the smallest degree, say **d**. Then the polynomial $w(x) = g(x) - h(x) \in C$ and it has a smaller degree than **d** and a multiplication by a scalar makes out of $w(x)$ a monic polynomial. Therefore the assumption that $g(x) \neq h(x)$ leads to a contradiction.

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

(i) Suppose $g(x)$ and $h(x)$ are two monic polynomials in C of the smallest degree, say **d**. Then the polynomial $w(x) = g(x) - h(x) \in C$ and it has a smaller degree than **d** and a multiplication by a scalar makes out of $w(x)$ a monic polynomial. Therefore the assumption that $g(x) \neq h(x)$ leads to a contradiction.

(ii) If $a(x) \in C$, then for some $q(x)$ and $r(x)$

$$a(x) = q(x)g(x) + r(x), \qquad \text{(where } deg\ r(x) < deg\ g(x)\text{)}.$$

and therefore

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

(i) Suppose $g(x)$ and $h(x)$ are two monic polynomials in C of the smallest degree, say **d**. Then the polynomial $w(x) = g(x) - h(x) \in C$ and it has a smaller degree than **d** and a multiplication by a scalar makes out of $w(x)$ a monic polynomial. Therefore the assumption that $g(x) \neq h(x)$ leads to a contradiction.

(ii) If $a(x) \in C$, then for some $q(x)$ and $r(x)$

$$a(x) = q(x)g(x) + r(x), \qquad \text{(where } deg\ r(x) < deg\ g(x)).$$

and therefore

$$r(x) = a(x) - q(x)g(x) \in C.$$

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$**.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

(i) Suppose $g(x)$ and $h(x)$ are two monic polynomials in C of the smallest degree, say **d**. Then the polynomial $w(x) = g(x) - h(x) \in C$ and it has a smaller degree than **d** and a multiplication by a scalar makes out of $w(x)$ a monic polynomial. Therefore the assumption that $g(x) \neq h(x)$ leads to a contradiction.

(ii) If $a(x) \in C$, then for some $q(x)$ and $r(x)$

$$a(x) = q(x)g(x) + r(x), \qquad \text{(where } deg\ r(x) < deg\ g(x)\text{)}.$$

and therefore

$$r(x) = a(x) - q(x)g(x) \in C.$$

By minimality condition

$$r(x) = 0$$

o

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that **all cyclic codes C have the form $C = \langle g(x) \rangle$ for some $g(x) \in R_n$.**

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

(i) Suppose $g(x)$ and $h(x)$ are two monic polynomials in C of the smallest degree, say **d**. Then the polynomial $w(x) = g(x) - h(x) \in C$ and it has a smaller degree than **d** and a multiplication by a scalar makes out of $w(x)$ a monic polynomial. Therefore the assumption that $g(x) \neq h(x)$ leads to a contradiction.

(ii) If $a(x) \in C$, then for some $q(x)$ and $r(x)$

$$a(x) = q(x)g(x) + r(x), \qquad \text{(where } deg\ r(x) < deg\ g(x)\text{)}.$$

and therefore

$$r(x) = a(x) - q(x)g(x) \in C.$$

By minimality condition

$$r(x) = 0$$

oand therefore $a(x) \in \langle g(x) \rangle$.

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(ii) It has to hold, for some $q(x)$ and $r(x)$

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(ii) It has to hold, for some $q(x)$ and $r(x)$

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) \pmod{x^n - 1}$$

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(ii) It has to hold, for some $q(x)$ and $r(x)$

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) \pmod{x^n - 1} \quad \text{and}$$

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(ii) It has to hold, for some $q(x)$ and $r(x)$

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) \pmod{x^n - 1} \quad \text{and}$$
$$r(x) \in C \Rightarrow r(x) = 0 \Rightarrow g(x) \quad \text{is therefore a factor of } x^n - 1.$$

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(ii) It has to hold, for some $q(x)$ and $r(x)$

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) \pmod{x^n - 1} \quad \text{and}$$
$$r(x) \in C \Rightarrow r(x) = 0 \Rightarrow g(x) \quad \text{is therefore a factor of } x^n - 1.$$

GENERATOR POLYNOMIALS - definition

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(ii) It has to hold, for some $q(x)$ and $r(x)$

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) \pmod{x^n - 1} \quad \text{and}$$
$$r(x) \in C \Rightarrow r(x) = 0 \Rightarrow g(x) \quad \text{is therefore a factor of } x^n - 1.$$

GENERATOR POLYNOMIALS - definition

Definition If

$$C = \langle g(x) \rangle,$$

for a cyclic code C,

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(ii) It has to hold, for some $q(x)$ and $r(x)$

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) \pmod{x^n - 1} \quad \text{and}$$
$$r(x) \in C \Rightarrow r(x) = 0 \Rightarrow g(x) \quad \text{is therefore a factor of } x^n - 1.$$

GENERATOR POLYNOMIALS - definition

Definition If

$$C = \langle g(x) \rangle,$$

for a cyclic code C, then g is called the generator polynomial for the code C.

HOW TO DESIGN CYCLIC CODES?

HOW TO DESIGN CYCLIC CODES?

The last claim of the previous theorem gives **a recipe to get all cyclic codes of the given length n in GF(q)**

HOW TO DESIGN CYCLIC CODES?

The last claim of the previous theorem gives **a recipe to get all cyclic codes of the given length n in GF(q)**

Indeed, all we need to do is to find all factors (**in GF(q)**) of

$$x^n - 1.$$

HOW TO DESIGN CYCLIC CODES?

The last claim of the previous theorem gives **a recipe to get all cyclic codes of the given length n in GF(q)**

Indeed, all we need to do is to find all factors (**in GF(q)**) of

$$x^n - 1.$$

Problem: Find all binary cyclic codes of length 3.

Solution: Make decomposition

$$x^3 - 1 = \underbrace{(x-1)(x^2+x+1)}_{\text{both factors are irreducible in GF(2)}}$$

Therefore, we have the following generator polynomials and cyclic codes of length 3.

Generator polynomials	Code in R_3	Code in $V(3,2)$
1	R_3	$V(3,2)$
$x+1$	$\{0, 1+x, x+x^2, 1+x^2\}$	$\{000, 110, 011, 101\}$
x^2+x+1	$\{0, 1+x+x^2\}$	$\{000, 111\}$
$x^3-1\ (=0)$	$\{0\}$	$\{000\}$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

- (i) All rows of G1 are linearly independent.

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1x + \ldots + g_rx^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

- (i) All rows of G1 are linearly independent.
- (ii) The $n - r$ rows of G represent codewords

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

- (i) All rows of G1 are linearly independent.
- (ii) The $n - r$ rows of G represent codewords

$$g(x), xg(x), x^2 g(x), \ldots, x^{n-r-1} g(x) \quad (*)$$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

- (i) All rows of G1 are linearly independent.
- (ii) The $n - r$ rows of G represent codewords

$$g(x), xg(x), x^2g(x), \ldots, x^{n-r-1}g(x) \quad (*)$$

- (iii) It remains to show that every codeword in C can be expressed as a linear combination of vectors from (*).

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

- (i) All rows of G1 are linearly independent.
- (ii) The $n - r$ rows of G represent codewords

$$g(x), xg(x), x^2 g(x), \ldots, x^{n-r-1} g(x) \quad (*)$$

- (iii) It remains to show that every codeword in C can be expressed as a linear combination of vectors from (*).

Indeed, if $a(x) \in C$, then

$$a(x) = q(x) g(x).$$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

- (i) All rows of G1 are linearly independent.
- (ii) The $n - r$ rows of G represent codewords

$$g(x), xg(x), x^2 g(x), \ldots, x^{n-r-1} g(x) \quad (*)$$

- (iii) It remains to show that every codeword in C can be expressed as a linear combination of vectors from (*).

Indeed, if $a(x) \in C$, then

$$a(x) = q(x)g(x).$$

Since $deg\ a(x) < n$ we have $deg\ q(x) < n - r$.

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

- (i) All rows of G1 are linearly independent.
- (ii) The $n - r$ rows of G represent codewords

$$g(x), xg(x), x^2 g(x), \ldots, x^{n-r-1} g(x) \quad (*)$$

- (iii) It remains to show that every codeword in C can be expressed as a linear combination of vectors from (*).

Indeed, if $a(x) \in C$, then

$$a(x) = q(x)g(x).$$

Since $deg\ a(x) < n$ we have $deg\ q(x) < n - r$.
Hence

$$\begin{aligned} q(x)g(x) &= (q_0 + q_1 x + \ldots + q_{n-r-1} x^{n-r-1}) g(x) \\ &= q_0 g(x) + q_1 x g(x) + \ldots + q_{n-r-1} x^{n-r-1} g(x). \end{aligned}$$

EXAMPLE

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore, there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore, there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4
$x - 1$	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore, there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4
$x - 1$	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$x + 1$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore, there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4
$x - 1$	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$x + 1$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$x^2 + 1$	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore, there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4
$x-1$	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$x+1$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
x^2+1	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
$(x-1)(x+1) = x^2 - 1$	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
$(x-1)(x^2+1) = x^3 - x^2 + x - 1$	$\begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}$

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore, there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4
$x-1$	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$x+1$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
x^2+1	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
$(x-1)(x+1) = x^2 - 1$	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
$(x-1)(x^2+1) = x^3 - x^2 + x - 1$	$\begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}$
$(x+1)(x^2+1)$	$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore, there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4
$x-1$	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$x+1$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
x^2+1	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
$(x-1)(x+1) = x^2 - 1$	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
$(x-1)(x^2+1) = x^3 - x^2 + x - 1$	$\begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}$
$(x+1)(x^2+1)$	$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$
$x^4 - 1 = 0$	$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$

EXAMPLE - II

In order to determine all binary cyclic codes of length 7, consider decomposition

$$x^7 - 1 = (x - 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

Since we want to determine binary codes, all computations should be modulo 2 and therefor all minus signs can be replaced by plus signs. Therefore

$$x^7 + 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

Therefore generators for 2^3 binary cyclic codes of length 7 are

$$1, \quad a(x) = x + 1, \quad b(x) = x^3 + x + 1, \quad c(x) = x^3 + x^2 + 1$$
$$a(x)b(x), \quad a(x)c(x), \quad b(x)c(x), \quad a(x)b(x)c(x) = x^7 + 1$$

ENCODING with CYCLIC CODES I

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

Let C be a cyclic $[n, k]$-code over a Galois field with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{r-1} x^{r-1} \text{ of degree } r - 1 = n - k - 1.$$

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

Let C be a cyclic $[n, k]$-code over a Galois field with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{r-1} x^{r-1} \text{ of degree } r - 1 = n - k - 1.$$

If a message vector m is represented by a polynomial $m(x)$ of the degree k, then message m is encoded,asy a polynomial $c(x)$, as follows:

$$m \Rightarrow c(x) = m(x)g(x),$$

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

Let C be a cyclic $[n, k]$-code over a Galois field with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{r-1} x^{r-1} \text{ of degree } r - 1 = n - k - 1.$$

If a message vector m is represented by a polynomial $m(x)$ of the degree k, then message m is encoded,asy a polynomial $c(x)$, as follows:

$$m \Rightarrow c(x) = m(x)g(x),$$

Such an encoding can be realized by the **shift register** shown in Figure below,

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

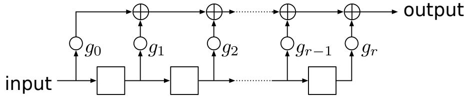
Let C be a cyclic $[n, k]$-code over a Galois field with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{r-1} x^{r-1} \text{ of degree } r - 1 = n - k - 1.$$

If a message vector m is represented by a polynomial $m(x)$ of the degree k, then message m is encoded,asy a polynomial $c(x)$, as follows:

$$m \Rightarrow c(x) = m(x)g(x),$$

Such an encoding can be realized by the **shift register** shown in Figure below, where input is the k-bit to-be-encoded message, followed by $n - k$ 0's, and the output will be the encoded message.



Shift-register for encoding a cyclic code.

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

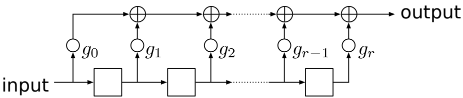
Let C be a cyclic $[n, k]$-code over a Galois field with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{r-1} x^{r-1} \text{ of degree } r - 1 = n - k - 1.$$

If a message vector m is represented by a polynomial $m(x)$ of the degree k, then message m is encoded,asy a polynomial $c(x)$, as follows:

$$m \Rightarrow c(x) = m(x)g(x),$$

Such an encoding can be realized by the **shift register** shown in Figure below, where input is the k-bit to-be-encoded message, followed by $n - k$ 0's, and the output will be the encoded message.



Shift-register for encoding a cyclic code. Small circles represent multiplication by the corresponding constants,

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

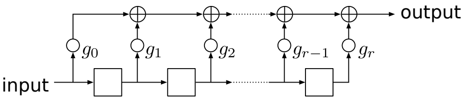
Let C be a cyclic $[n, k]$-code over a Galois field with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{r-1} x^{r-1} \text{ of degree } r - 1 = n - k - 1.$$

If a message vector m is represented by a polynomial $m(x)$ of the degree k, then message m is encoded,asy a polynomial $c(x)$, as follows:

$$m \Rightarrow c(x) = m(x)g(x),$$

Such an encoding can be realized by the **shift register** shown in Figure below, where input is the k-bit to-be-encoded message, followed by $n-k$ 0's, and the output will be the encoded message.



Shift-register for encoding a cyclic code. Small circles represent multiplication by the corresponding constants, $\bigoplus$ nodes represent modular additions,

ENCODING with CYCLIC CODES I

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message (codeword) polynomial and the generating polynomial for the code.

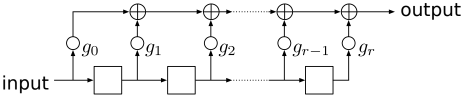
Let C be a cyclic $[n, k]$-code over a Galois field with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{r-1} x^{r-1} \text{ of degree } r - 1 = n - k - 1.$$

If a message vector m is represented by a polynomial $m(x)$ of the degree k, then message m is encoded,asy a polynomial $c(x)$, as follows:

$$m \Rightarrow c(x) = m(x)g(x),$$

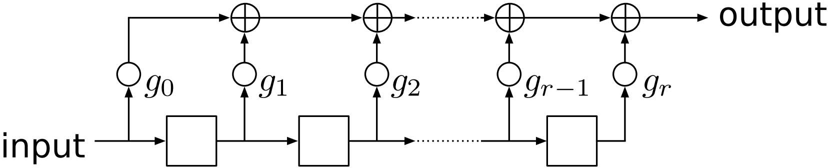
Such an encoding can be realized by the **shift register** shown in Figure below, where input is the k-bit to-be-encoded message, followed by $n - k$ 0's, and the output will be the encoded message.



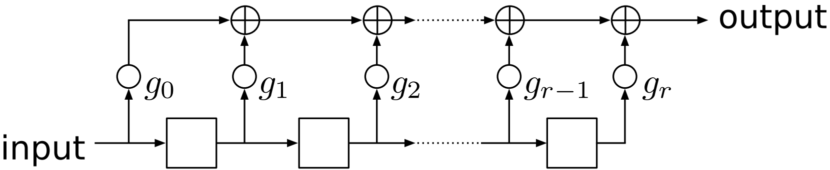
Shift-register for encoding a cyclic code. Small circles represent multiplication by the corresponding constants, $\bigoplus$ nodes represent modular additions, squares are shift cells.

EXAMPLE

EXAMPLE



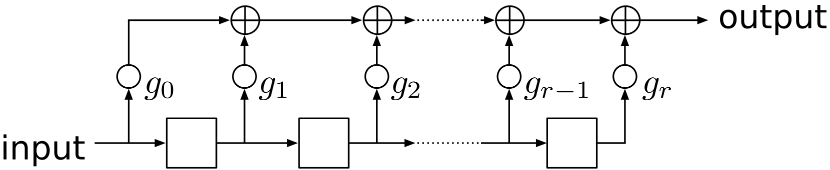
A shift-register encodings of cyclic codes. Small circles represent multiplications by the corresponding constants, $\bigoplus$ nodes represent modular additions, squares are delay elements.



A shift-register encodings of cyclic codes. Small circles represent multiplications by the corresponding constants, $\bigoplus$ nodes represent modular additions, squares are delay elements.

The input (message) is given by a polynomial
$m_{k-1}x^{k-1} + \ldots m_2x^2 + m_1x + m_0$

EXAMPLE



A shift-register encodings of cyclic codes. Small circles represent multiplications by the corresponding constants, $\bigoplus$ nodes represent modular additions, squares are delay elements.

The input (message) is given by a polynomial

$m_{k-1}x^{k-1} + \ldots m_2x^2 + m_1x + m_0$

and therefore the input to the shift register, step by step, is the word

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

Let us compute

$$(m_0 + m_1 x + \dots m_{k-1} x^{k-1}) \times (g_0 + g_1 x + g_2 x^2 \dots g_{r-1} x^{r-1})$$

$$=$$

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

Let us compute

$$(m_0 + m_1x + \dots m_{k-1}x^{k-1}) \times (g_0 + g_1x + g_2x^2 \dots g_{r-1}x^{r-1})$$

$$=$$

$$m_0g_0$$

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

Let us compute

$$(m_0 + m_1x + \dots m_{k-1}x^{k-1}) \times (g_0 + g_1x + g_2x^2 \dots g_{r-1}x^{r-1})$$

$$=$$

$$m_0g_0$$

$$+$$

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

Let us compute

$$(m_0 + m_1 x + \ldots m_{k-1} x^{k-1}) \times (g_0 + g_1 x + g_2 x^2 \ldots g_{r-1} x^{r-1})$$

$$=$$

$$m_0 g_0$$

$$+$$

$$(m_0 g_1 + m_1 g_0) x$$

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

Let us compute

$$(m_0 + m_1 x + \ldots m_{k-1} x^{k-1}) \times (g_0 + g_1 x + g_2 x^2 \ldots g_{r-1} x^{r-1})$$

$$=$$

$$m_0 g_0$$

$$+$$

$$(m_0 g_1 + m_1 g_0) x$$

$$+$$

$$(m_0 g_2 + m_1 g_1 + m_2 g_0) x^2$$

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

Let us compute

$$(m_0 + m_1x + \dots m_{k-1}x^{k-1}) \times (g_0 + g_1x + g_2x^2 \dots g_{r-1}x^{r-1})$$

$$=$$

$$m_0g_0$$

$$+$$

$$(m_0g_1 + m_1g_0)x$$

$$+$$

$$(m_0g_2 + m_1g_1 + m_2g_0)x^2$$

$$+$$

$$(m_0g_3 + m_1g_2 + m_2g_1 + m_3g_0)x^3$$

MULTIPLICATION of POLYNOMIALS by SHIFT-REGISTERS

Let us compute

$$(m_0 + m_1x + \dots m_{k-1}x^{k-1}) \times (g_0 + g_1x + g_2x^2 \dots g_{r-1}x^{r-1})$$

$$=$$

$$m_0g_0$$

$$+$$

$$(m_0g_1 + m_1g_0)x$$

$$+$$

$$(m_0g_2 + m_1g_1 + m_2g_0)x^2$$

$$+$$

$$(m_0g_3 + m_1g_2 + m_2g_1 + m_3g_0)x^3$$

$$+$$

$$\vdots$$

EXAMPLES of CYCLIC CODES

GOLAY CODES - DESCRIPTION

GOLAY CODES - DESCRIPTION

Golay codes G_{24} and G_{23} were used by spacecraft 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 spacecraft Voyager I and Voyager II to transmit color pictures of Jupiter and Saturn. Generator matrix for G_{24} has the 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 G_{24} and G_{23} were used by spacecraft Voyager I and Voyager II to transmit color pictures of Jupiter and Saturn. Generator matrix for G_{24} has the 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 symbol of each codeword of G_{24}. G_{23} is $(23, 12, 7)$-code. It is a perfect code.

GOLAY CODE II

GOLAY CODE II

Golay code G_{23} is a $(23, 12, 7)$-code and can be defined also as the cyclic code generated by the codeword

$$11000111010100000000000$$

This code can be constructed via factorization of $x^{23} - 1$.

GOLAY CODE II

Golay code G_{23} is a $(23, 12, 7)$-code and can be defined also as the cyclic code generated by the codeword

$$11000111010100000000000$$

This code can be constructed via factorization of $x^{23} - 1$.

GOLAY CODE II

Golay code G_{23} is a $(23, 12, 7)$-code and can be defined also as the cyclic code generated by the codeword

$$11000111010100000000000$$

This code can be constructed via factorization of $x^{23} - 1$.

Golay code G_{24} was used in NASA Deep Space Missions - in spacecraft Voyager 1 and Voyager 2. It was also used in the US-government standards for automatic link establishment in "High Frequency Radio Systems.

GOLAY CODE II

Golay code G_{23} is a $(23, 12, 7)$-code and can be defined also as the cyclic code generated by the codeword

$$11000111010100000000000$$

This code can be constructed via factorization of $x^{23} - 1$.

Golay code G_{24} was used in NASA Deep Space Missions - in spacecraft Voyager 1 and Voyager 2. It was also used in the US-government standards for automatic link establishment in "High Frequency Radio Systems.

Golay codes are named to honour Marcel J. E. Golay - from 1949.

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, 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, 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

$$v_0 = 11111111$$

$$v_1 = 10101010$$

$$v_2 = 11001100$$

Example 2: $RM(2,3)$ code is generated by the codewords

$$v_0, v_1, v_2, v_3, v_1 \cdot v_2, v_1 \cdot v_3, v_2 \cdot v_3$$

Example 2: $RM(2,3)$ code is generated by the codewords

$$v_0, v_1, v_2, v_3, v_1 \cdot v_2, v_1 \cdot v_3, v_2 \cdot v_3$$

where, for example $v_1 \cdot v_3 = 10100000$

Example 2: $RM(2,3)$ code is generated by the codewords

$$v_0, v_1, v_2, v_3, v_1 \cdot v_2, v_1 \cdot v_3, v_2 \cdot v_3$$

where, for example $v_1 \cdot v_3 = 10100000$

Special cases of Reed-Muller codes are Hadamard code and Reed-Solomon code.

Example 2: $RM(2,3)$ code is generated by the codewords

$$v_0, v_1, v_2, v_3, v_1 \cdot v_2, v_1 \cdot v_3, v_2 \cdot v_3$$

where, for example $v_1 \cdot v_3 = 10100000$

Special cases of Reed-Muller codes are Hadamard code and Reed-Solomon code.

Reed-Muller codes are closely related to Polar codes. David E. Muller discovered them in 1954 and Irving S. Reed was first to propose for them efficient decoding algorithm.

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** 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

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

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

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,...

[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.

[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).

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.

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;**
- **transmit at a higher data rate.**

CHANNEL CAPACITY

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.

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

$$R(p_b) = \frac{C}{1 - H_2(p_b)}$$

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.

CONVOLUTION CODES

CONVOLUTION CODES

Our first example of good, though simple, channel codes are convolution codes.

CONVOLUTION CODES

Our first example of good, though simple, channel codes are convolution codes.

Convolution codes have simple encoding and decoding, are quite a simple generalization of linear codes and have encodings as cyclic codes.

CONVOLUTION CODES

Our first example of good, though simple, channel codes are convolution codes.

Convolution codes have simple encoding and decoding, are quite a simple generalization of linear codes and have encodings as cyclic codes.

An (n, k) convolution code (CC) is defined by an $k \times n$ generator matrix, entries of which are polynomials over F_2.

CONVOLUTION CODES

Our first example of good, though simple, channel codes are convolution codes.

Convolution codes have simple encoding and decoding, are quite a simple generalization of linear codes and have encodings as cyclic codes.

An (n, k) convolution code (CC) is defined by an $k \times n$ generator matrix, entries of which are polynomials over F_2.

For example,

$$G_1 = [x^2 + 1, x^2 + x + 1]$$

is the generator matrix for a $(2, 1)$ convolution code, denoted CC_1, and

$$G_2 = \begin{pmatrix} 1 + x & 0 & x + 1 \\ 0 & 1 & x \end{pmatrix}$$

is the generator matrix for a $(3, 2)$ convolution code denoted CC_2

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

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

and

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

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

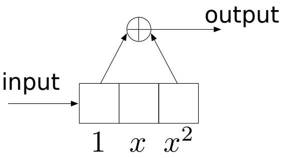
$$C_{0i} = I_i(x) + I_{i+2}(x), \quad C_{1i}(x) = I_i + I_{i-1} + I_{i-2}.$$

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

ENCODING

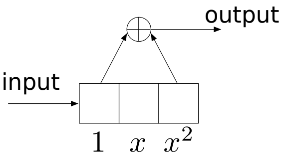
ENCODING

The **first shift register**



ENCODING

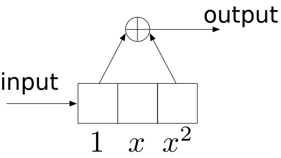
The **first shift register**



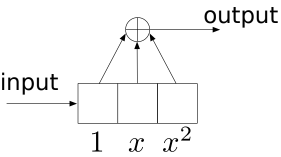
will multiply the input stream by $x^2 + 1$ and the second shift register

ENCODING

The **first shift register**

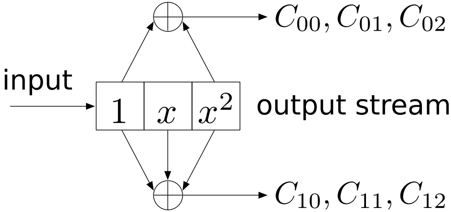


will multiply the input stream by $x^2 + 1$ and the second shift register



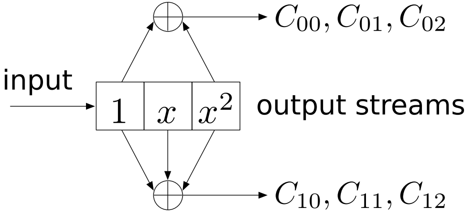
will multiply the input stream by $x^2 + x + 1$.

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.

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

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

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.

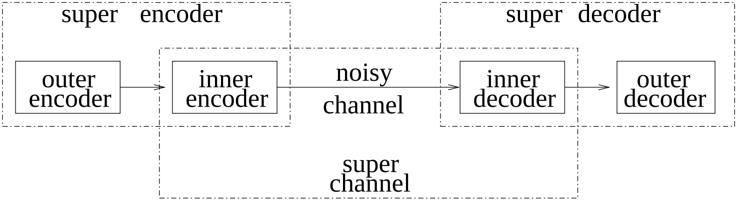
CONCATENATED CODES - I

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

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

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

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

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

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

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

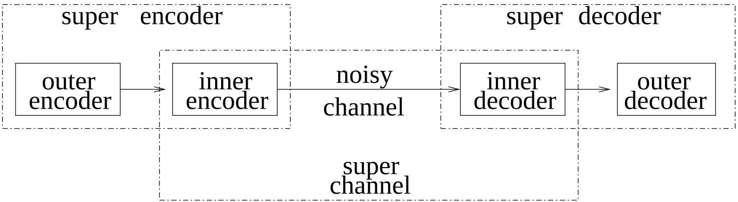
$$C_{out}(m_1, m_2, \ldots, m_k) = (m_1^{'}, m_2^{'}, \ldots, m_N^{'})$$

and then C_{in} encoding is applied to get

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

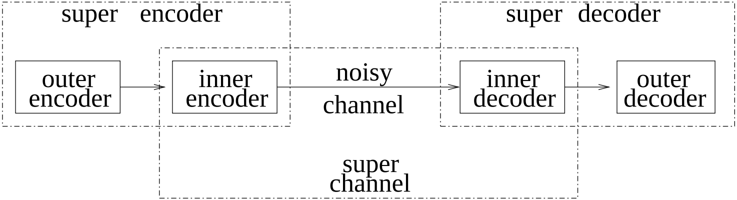
ANOTHER VIEW of CONCATENATED CODES

ANOTHER VIEW of CONCATENATED CODES



- **Outer code:** - (n_2, k_2) code
- **Inner code**: - (n_1, k_1) binary code
- **Inner decoder** - (n_1, k_1) code
- **Outer decoder** - (n_2, k_2) code

ANOTHER VIEW of CONCATENATED CODES



- **Outer code:** - (n_2, k_2) code
- **Inner code**: - (n_1, k_1) binary code
- **Inner decoder** - (n_1, k_1) code
- **Outer decoder** - (n_2, k_2) code

- **length** of such a concatenated code is $n_1 n_2$
- **dimension** of such a concatenated code is $k_1 k_2$
- if **minimal distances** of both codes are d_1 and d_2, then resulting concatenated code has minimal distance $\geq d_1 d_2$.

EFFICIENT DECODING of CONCATENATED CODES

EFFICIENT DECODING of CONCATENATED CODES

- A natural approach to decoding of concatenated codes is to decode first the inner code and then the outer code.

EFFICIENT DECODING of CONCATENATED CODES

- A natural approach to decoding of concatenated codes is to decode first the inner code and then the outer code.
- For a decoding algorithm to be practical it has to be polynomial time in the final block length.

EFFICIENT DECODING of CONCATENATED CODES

- A natural approach to decoding of concatenated codes is to decode first the inner code and then the outer code.
- For a decoding algorithm to be practical it has to be polynomial time in the final block length.
- Assume there is a polynomial unique decoding algorithm for the outer code.

EFFICIENT DECODING of CONCATENATED CODES

- A natural approach to decoding of concatenated codes is to decode first the inner code and then the outer code.
- For a decoding algorithm to be practical it has to be polynomial time in the final block length.
- Assume there is a polynomial unique decoding algorithm for the outer code.
- Next goal is to find polynomial time decoding algorithm for the inner code that is polynomial in the final block length.

EFFICIENT DECODING of CONCATENATED CODES

- A natural approach to decoding of concatenated codes is to decode first the inner code and then the outer code.
- For a decoding algorithm to be practical it has to be polynomial time in the final block length.
- Assume there is a polynomial unique decoding algorithm for the outer code.
- Next goal is to find polynomial time decoding algorithm for the inner code that is polynomial in the final block length.
- The main idea is that if the inner block length is logarithmic in the size of the outer code, then the decoding algorithm for the inner code may run in the exponential time of the inner block length.

EFFICIENT DECODING of CONCATENATED CODES

- A natural approach to decoding of concatenated codes is to decode first the inner code and then the outer code.
- For a decoding algorithm to be practical it has to be polynomial time in the final block length.
- Assume there is a polynomial unique decoding algorithm for the outer code.
- Next goal is to find polynomial time decoding algorithm for the inner code that is polynomial in the final block length.
- The main idea is that if the inner block length is logarithmic in the size of the outer code, then the decoding algorithm for the inner code may run in the exponential time of the inner block length.
- In such a case we can use an exponential time but optimal maximum likelihood decoder for the inner code.

APPLICATIONS

APPLICATIONS

- Concatenated codes started to be used for deep space communication starting with Voyager program in 1977 and stayed so until the invention of Turbo codes and LDPC codes.

APPLICATIONS

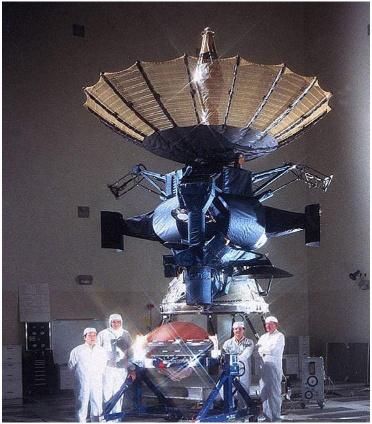
- Concatenated codes started to be used for deep space communication starting with Voyager program in 1977 and stayed so until the invention of Turbo codes and LDPC codes.
- Concatenated codes are used also on Compact Disc.

APPLICATIONS

- Concatenated codes started to be used for deep space communication starting with Voyager program in 1977 and stayed so until the invention of Turbo codes and LDPC codes.
- Concatenated codes are used also on Compact Disc.
- The best concatenated codes for many applications were based on outer Reed-Solomon codes and inner Viterbi-decoded short constant length convolution 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.

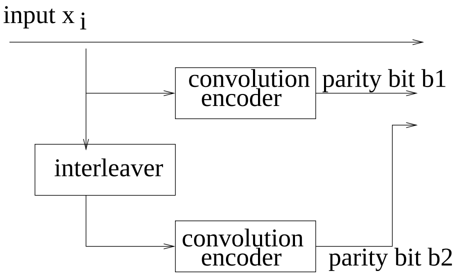
A **Turbo code** can be seen as formed from a parallel composition of two (convolution) codes separated by an **interleaver** (that permutes blocks of data in a fixed (pseudo)-random way).

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).

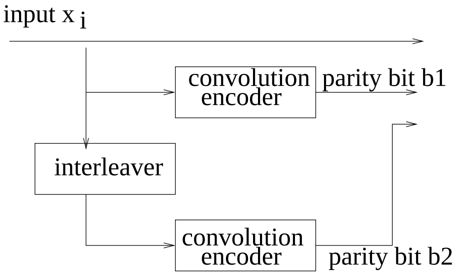
A Turbo encoder is formed from the parallel composition of two (convolution) encoders separated by an interleaver.



EXAMPLES of TURBO and CONVOLUTION ENCODERS

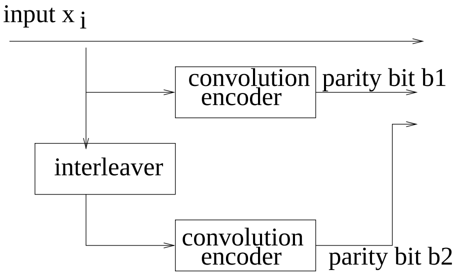
EXAMPLES of TURBO and CONVOLUTION ENCODERS

A Turbo encoder

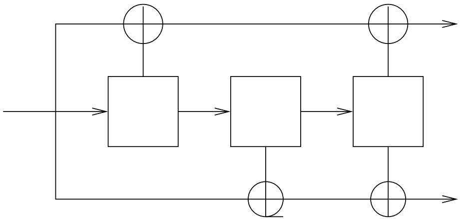


EXAMPLES of TURBO and CONVOLUTION ENCODERS

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.
- 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.
- 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.
- 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.
- 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

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 $3dB$.

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

  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 $3dB$.
- For code rate $\frac{1}{2}$ the relative increase in energy consumption is about 4.8 dB for convolution codes and 0.98 for Turbo 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.

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 $\{0, 1, \ldots, q-1\}$.

WHY ARE TURBO CODES SO GOOD?

- Turbo codes are linear codes.

WHY ARE TURBO CODES SO GOOD?

- Turbo codes are linear codes.
- A "good" linear code is one that has mostly high-weight codewords.

WHY ARE TURBO CODES SO GOOD?

- 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.

WHY ARE TURBO CODES SO GOOD?

- 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.

WHY ARE TURBO CODES SO GOOD?

- 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.

LIST DECODING

UNIQUE versus LIST DECODING

UNIQUE 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 codeword w to the message w_c that was received.

UNIQUE 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 codeword w to the message w_c that was received.

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

UNIQUE 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 codeword w to the message w_c that was received.

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

In the **list decoding** model the task to solve when a code C is used, is for a received message (a corrupted codeword) w_c and a given ϵ to output (list of) all codewords from C with the distance at most ε from w_c.

UNIQUE 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 codeword w to the message w_c that was received.

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

In the **list decoding** model the task to solve when a code C is used, is for a received message (a corrupted codeword) w_c and a given ϵ to output (list of) all codewords from C with the distance at most ε from w_c.

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

UNIQUE 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 codeword w to the message w_c that was received.

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

In the **list decoding** model the task to solve when a code C is used, is for a received message (a corrupted codeword) w_c and a given ϵ to output (list of) all codewords from C with the distance at most ε from w_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, including the Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors.

UNIQUE 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 codeword w to the message w_c that was received.

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

In the **list decoding** model the task to solve when a code C is used, is for a received message (a corrupted codeword) w_c and a given ϵ to output (list of) all codewords from C with the distance at most ε from w_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, including the Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors.

UNIQUE 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 codeword w to the message w_c that was received.

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

In the **list decoding** model the task to solve when a code C is used, is for a received message (a corrupted codeword) w_c and a given ϵ to output (list of) all codewords from C with the distance at most ε from w_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, including the Reed-Solomon codes, there are efficient algorithms for list decoding that allow to correct a large variety of errors.

List decoding seems to be a stronger error-correcting mode than unique decoding.

UNIQUE DECODING:

$$m - -- > e(m) - -- > NOISE - -- > n(e(m)) - -- > e(m)$$

UNIQUE DECODING:

$$m - -- > e(m) - -- > NOISE - -- > n(e(m)) - -- > e(m)$$

LIST DECODING:

$m --- > e(m) ---- > NOISE --- > n(e(m)) --- > S_m ---$such that $e(m) \in S_m$

LIST DECODING - INTUITION BEHIND

LIST DECODING - INTUITION BEHIND

For a polynomial-time list decoding algorithm to exist we need that any Hamming ball of a radius pn around a received word (where p is the fraction of errors in terms of the block length n) has a small number of codewords.

LIST DECODING - INTUITION BEHIND

For a polynomial-time list decoding algorithm to exist we need that any Hamming ball of a radius pn around a received word (where p is the fraction of errors in terms of the block length n) has a small number of codewords.

This is because the list size itself is a lower bound for the running time of the algorithm. Hence it is required that the list size has to be polynomial in the block length of the code.

LIST DECODING - INTUITION BEHIND

For a polynomial-time list decoding algorithm to exist we need that any Hamming ball of a radius pn around a received word (where p is the fraction of errors in terms of the block length n) has a small number of codewords.

This is because the list size itself is a lower bound for the running time of the algorithm. Hence it is required that the list size has to be polynomial in the block length of the code.

A combinatorial consequence of the above requirement is that it implies an upper bound on the rate of the code.

LIST DECODING - INTUITION BEHIND

For a polynomial-time list decoding algorithm to exist we need that any Hamming ball of a radius pn around a received word (where p is the fraction of errors in terms of the block length n) has a small number of codewords.

This is because the list size itself is a lower bound for the running time of the algorithm. Hence it is required that the list size has to be polynomial in the block length of the code.

A combinatorial consequence of the above requirement is that it implies an upper bound on the rate of the code. List decoding promises to meet this bound.

LIST DECODING - INTUITION BEHIND

For a polynomial-time list decoding algorithm to exist we need that any Hamming ball of a radius pn around a received word (where p is the fraction of errors in terms of the block length n) has a small number of codewords.

This is because the list size itself is a lower bound for the running time of the algorithm. Hence it is required that the list size has to be polynomial in the block length of the code.

A combinatorial consequence of the above requirement is that it implies an upper bound on the rate of the code. List decoding promises to meet this bound.

EFFICIENCY of LIST DECODING - SUMMARY

EFFICIENCY of LIST DECODING - SUMMARY

With list decoding the error-correction performance can double.

EFFICIENCY of LIST DECODING - SUMMARY

With list decoding the error-correction performance can double.

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

EFFICIENCY of LIST DECODING - SUMMARY

With list decoding the error-correction performance can double.

It has been shown, non-constructively, for any code rate R, that such 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 - SUMMARY

With list decoding the error-correction performance can double.

It has been shown, non-constructively, for any code rate R, that such 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 a list decoding up to $1 - \sqrt{2R}$ errors.

LIST DECODING - MATHEMATICAL FORMULATION

LIST DECODING - MATHEMATICAL FORMULATION

Let C be a q-nary linear $[n, k, d]$ error correcting code.

LIST DECODING - MATHEMATICAL FORMULATION

Let C be a q-nary linear $[n, k, d]$ error correcting code.

For a given q-nary input word w of length n and a given error bound ε **let the task be to output a list of codewords of C whose Hamming distance from w is at most ε**

LIST DECODING - MATHEMATICAL FORMULATION

Let C be a q-nary linear $[n, k, d]$ error correcting code.

For a given q-nary input word w of length n and a given error bound ε **let the task be to output a list of codewords of C whose Hamming distance from w is at most ε**

We are, naturally, interested only in polynomial, in n, algorithms able to do that.

LIST DECODING - MATHEMATICAL FORMULATION

Let C be a q-nary linear $[n, k, d]$ error correcting code.

For a given q-nary input word w of length n and a given error bound ε **let the task be to output a list of codewords of C whose Hamming distance from w is at most ε**

We are, naturally, interested only in polynomial, in n, algorithms able to do that.

(p, L)-list decodability: Let C be a q-nary code of codewords of length n; $0 \leq p \leq 1$ and let $L > 1$ be an integer.

LIST DECODING - MATHEMATICAL FORMULATION

Let C be a q-nary linear $[n, k, d]$ error correcting code.

For a given q-nary input word w of length n and a given error bound ε **let the task be to output a list of codewords of C whose Hamming distance from w is at most ε**

We are, naturally, interested only in polynomial, in n, algorithms able to do that.

(p, L)-list decodability: Let C be a q-nary code of codewords of length n; $0 \leq p \leq 1$ and let $L > 1$ be an integer.

If for every q-nary word w of length n the number of codewords of C withing Hamming distance pn from w is at most L, then the code C is said to be (p, L)-list-decodable.

LIST DECODING - MATHEMATICAL FORMULATION

Let C be a q-nary linear $[n, k, d]$ error correcting code.

For a given q-nary input word w of length n and a given error bound ε **let the task be to output a list of codewords of C whose Hamming distance from w is at most ε**

We are, naturally, interested only in polynomial, in n, algorithms able to do that.

(p, L)-list decodability: Let C be a q-nary code of codewords of length n; $0 \leq p \leq 1$ and let $L > 1$ be an integer.

If for every q-nary word w of length n the number of codewords of C withing Hamming distance pn from w is at most L, then the code C is said to be (p, L)-list-decodable.

Theorem let $q \geq 2$, $0 \leq p \leq 1 - 1/q$ and $\varepsilon \geq 0$ then for large enough block length n if the code rate $R \leq 1 - H_q(p) - \varepsilon$, then there exists a $(p, O(1/\varepsilon))$-list decodable code.
[$H_q(p) = p \log_q(q-1) - p \log_q p - (1-p) \log_q(1-p)$ is q-ary entropy function.]
Moreover, if $R > 1 - H_q(p) + \varepsilon$, then every (p, L)-list-decodable code has $L = q^{\Omega(n)}$

LIST DECODING POTENTIAL

LIST DECODING POTENTIAL

- The concept of list decoding was proposed by Peter Elias in 1950s.

LIST DECODING POTENTIAL

- The concept of list decoding was proposed by Peter Elias in 1950s.
- In 2006 Guruswami and Atri Rudra gave explicit codes that achieve list decoding capacity.

LIST DECODING POTENTIAL

- The concept of list decoding was proposed by Peter Elias in 1950s.
- In 2006 Guruswami and Atri Rudra gave explicit codes that achieve list decoding capacity.
- Their codes are called folded Reed-Solomon codes and they are actually nothing but plain Reed-Solomon codes but viewed as codes over a larger alphabet by a careful bundling codeword symbols.

- The concept of list decoding was proposed by Peter Elias in 1950s.
- In 2006 Guruswami and Atri Rudra gave explicit codes that achieve list decoding capacity.
- Their codes are called folded Reed-Solomon codes and they are actually nothing but plain Reed-Solomon codes but viewed as codes over a larger alphabet by a careful bundling codeword symbols.

APPLICATIONS in COMPLEXITY THEORY

Surprisingly, list-decoding found interesting applications in cryptography and in computational complexity theory. For example, in

APPLICATIONS in COMPLEXITY THEORY

Surprisingly, list-decoding found interesting applications in cryptography and in computational complexity theory. For example, in

- designing of hard core predicates from one-way permutations;

APPLICATIONS in COMPLEXITY THEORY

Surprisingly, list-decoding found interesting applications in cryptography and in computational complexity theory. For example, in

- designing of hard core predicates from one-way permutations;
- predicting witnesses for **NP**-problems;

APPLICATIONS in COMPLEXITY THEORY

Surprisingly, list-decoding found interesting applications in cryptography and in computational complexity theory. For example, in

- designing of hard core predicates from one-way permutations;
- predicting witnesses for **NP**-problems;
- designing randomness extractors and pseudorandom generators.

APPENDIX - I.

ANOTHER APPLICATIONS of REED-SOLOMON CODES

ANOTHER APPLICATIONS of REED-SOLOMON CODES

- Reed-Solomon codes have been widely used in mass storage systems to correct the burst errors caused by media defects.

ANOTHER APPLICATIONS of REED-SOLOMON CODES

- Reed-Solomon codes have been widely used in mass storage systems to correct the burst errors caused by media defects.
- Special types of Reed-Solomon codes have been used to overcome unreliable nature of data transmission over erasure channels.

ANOTHER APPLICATIONS of REED-SOLOMON CODES

- Reed-Solomon codes have been widely used in mass storage systems to correct the burst errors caused by media defects.
- Special types of Reed-Solomon codes have been used to overcome unreliable nature of data transmission over erasure channels.
- Several bar-code systems use Reed-Solomon codes to allow correct reading even if a portion of a bar code is damaged.

ANOTHER APPLICATIONS of REED-SOLOMON CODES

- Reed-Solomon codes have been widely used in mass storage systems to correct the burst errors caused by media defects.
- Special types of Reed-Solomon codes have been used to overcome unreliable nature of data transmission over erasure channels.
- Several bar-code systems use Reed-Solomon codes to allow correct reading even if a portion of a bar code is damaged.
- Reed-Solomon codes were used to encode pictures sent by the Voyager spacecraft.

ANOTHER APPLICATIONS of REED-SOLOMON CODES

- Reed-Solomon codes have been widely used in mass storage systems to correct the burst errors caused by media defects.
- Special types of Reed-Solomon codes have been used to overcome unreliable nature of data transmission over erasure channels.
- Several bar-code systems use Reed-Solomon codes to allow correct reading even if a portion of a bar code is damaged.
- Reed-Solomon codes were used to encode pictures sent by the Voyager spacecraft.
- Modern versions of concatenated Reed-Solomon/Viterbi decoder convolution coding were and are used on the Mars Pathfinder, Galileo, Mars exploration Rover and Cassini missions, where they performed within about 1-1.5dB of the ultimate limit imposed by the Shannon theorem.

FUTURE of CODING DEVELOPMENTS

The following reasons are behind increasing needs to develop new and new codes, new and new encoding and decoding methods:

FUTURE of CODING DEVELOPMENTS

The following reasons are behind increasing needs to develop new and new codes, new and new encoding and decoding methods:

- Needs for miniaturization, higher quality and better efficiency as well as energy savings of many important information storing and processing devices.

FUTURE of CODING DEVELOPMENTS

The following reasons are behind increasing needs to develop new and new codes, new and new encoding and decoding methods:

- Needs for miniaturization, higher quality and better efficiency as well as energy savings of many important information storing and processing devices.
- New channels are used, new types of errors start to be possible.

FUTURE of CODING DEVELOPMENTS

The following reasons are behind increasing needs to develop new and new codes, new and new encoding and decoding methods:

- Needs for miniaturization, higher quality and better efficiency as well as energy savings of many important information storing and processing devices.
- New channels are used, new types of errors start to be possible.
- New computation tools are developed - for example special types of parallelization,....

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 have another remarkable property:

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.

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.