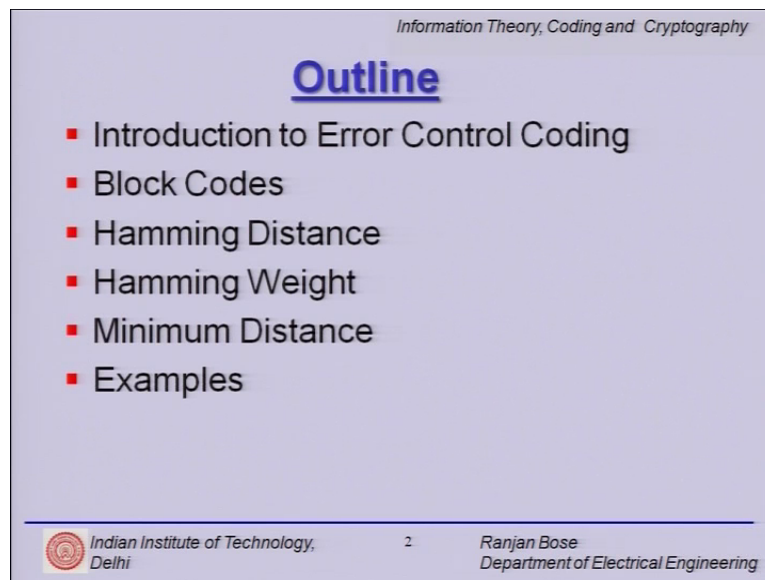


Information Theory, Coding and Cryptography
Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Module – 14
Linear Block Codes
Lecture – 14

Hello and welcome to our next module on Linear Block Codes.

(Refer Slide Time: 00:33)



The slide is titled "Information Theory, Coding and Cryptography" at the top right. The main heading is "Outline" in blue, underlined. Below it is a bulleted list of topics: Introduction to Error Control Coding, Block Codes, Hamming Distance, Hamming Weight, Minimum Distance, and Examples. At the bottom, there is a footer with the IIT Delhi logo, the text "Indian Institute of Technology, Delhi", the number "2", and the name "Ranjan Bose, Department of Electrical Engineering".

Information Theory, Coding and Cryptography

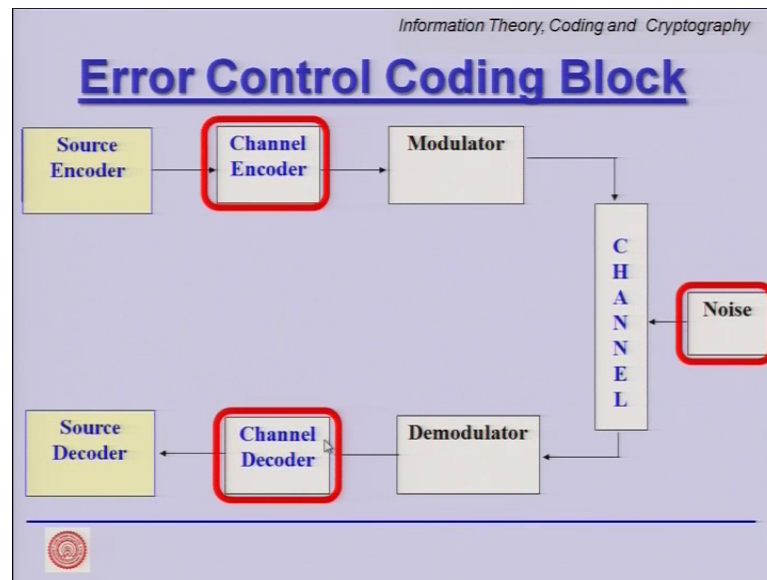
Outline

- Introduction to Error Control Coding
- Block Codes
- Hamming Distance
- Hamming Weight
- Minimum Distance
- Examples

Indian Institute of Technology, Delhi 2 Ranjan Bose
Department of Electrical Engineering

Let us start with a brief outline of the talk.

(Refer Slide Time: 00:36)



(Refer Slide Time: 00:37)

Information Theory, Coding and Cryptography

Recap

- Introduction to Error Control Coding
- Block Codes
- Hamming Distance
- Hamming Weight
- Minimum Distance
- Examples

Indian Institute of Technology, Delhi

3

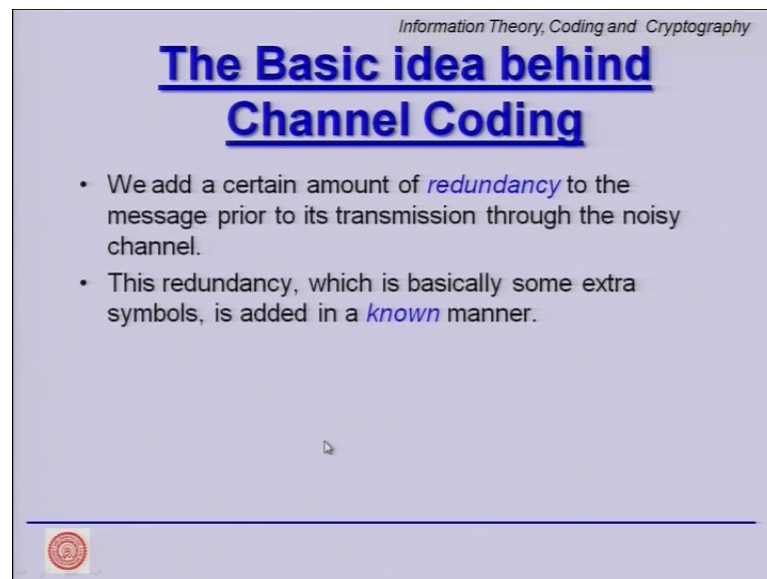
Ranjan Bose
Department of Electrical Engineering

What we will start with is some error control codes, we will look at block codes, we have started looking at a hamming codes, hamming distances, and then we will look at some examples. But before we start let us start with a quick recap of the things we have done already. We will revisit them today in terms of block codes, hamming distances, hamming weights, and finally we will look at some examples.

So, this is our error control coding block diagram. What we have is a source encoder in the communication channel followed by a channel encoder and a modulator. And then

we send our waveforms over the channel where they are corrupted by a noise. At the receiving end, we demodulate the channel decoder recovers from the errors introduced by the channel; and finally, the source decoder gives back the original bit stream. Now, in this module, we are focusing on the channel encoder, and channel decoder clearly they are in existence because of the presence of noise in the channel. The job of the noise is to flip a few bits here and there; and our job of the channel decoder is to recover from these errors.

(Refer Slide Time: 02:06)



Information Theory, Coding and Cryptography

The Basic idea behind Channel Coding

- We add a certain amount of *redundancy* to the message prior to its transmission through the noisy channel.
- This redundancy, which is basically some extra symbols, is added in a *known* manner.

We would established last time that the basic idea behind channel coding is to add redundancy in a known manner ok. So, this known manner is critical. This known manner is a mathematical method, it could be an algebraic structure, it could be a geometric structure that we will put in, so that we are the transmitter and our front at the receiver knows how we have added the redundancy.

But noise definitely does not know and it creates randomly it rugs the structure, but we would use that structure in built mathematical structure to bring recover from the erroneous bits. So, this is the general idea in layman's language mathematically how we do it, we will see shortly.

(Refer Slide Time: 02:56)

Information Theory, Coding and Cryptography

Hamming Distance

- The **Hamming distance** between two codewords is the number of places the codewords differ.
- The **Hamming distance** between two codewords c_1 and c_2 is denoted by $d(c_1, c_2)$.

Example: $d(10110, 11011) = 3$.

11011
10110

In order to do so we have defined some terms, some mathematical tools, so we talked about the hamming distance between two codewords as a number of places the codewords differ and we denote it by $d(c_1, c_2)$. So, we have done this earlier. And as a simple example, if we have these two vectors 1 0 1 1 0 and 1 1 0 1 1, we can clearly count the number of place where different. We will look at it again shortly. But if you clearly go for these two vectors first you see that they are of the same length and then you try to compare and you find out where they are different, and you find them different at these three places marked by blue and hence we say that the hamming distances is 3 ok.


So, this is how we calculate a hamming distance. It should be pointed out that it is not necessarily between binary bit streams that you can find the hamming distance. These could have been vectors of any symbols or characters and you can find the hamming distance. The only condition is that they get elements from the same set and they should be of equal length.

(Refer Slide Time: 04:17)

Information Theory, Coding and Cryptography

Hamming Weight

- The **Hamming weight** of a codeword (or any vector) is equal to the number of *non-zero elements* in the codeword.
- The Hamming weight of a codeword c is denoted by $w(c)$.
- It is easy to see that $d(c_1, c_2) = w(c_1 - c_2)$.

 Indian Institute of Technology,
Delhi

7

Ranjan Bose
Department of Electrical Engineering

We also defined what is a hamming weight, hamming weight of a codeword or for that matter any vector is equal to the number of non-zero elements in the codeword. We realize that hamming weight has no units, because it is just a number. So, hamming weight is typically denoted by w parentheses c , where c is the codeword or the vector in question.


We can see that the distance the hamming distance between c_1 and c_2 is nothing but the weight of $c_1 - c_2$ ok. This is pretty easy to observe, because $c_1 - c_2$ tells me where they are different, and so way if they are different it puts a 1; if they are not different it puts a 0, and therefore, weight is just the sum of the non-zero elements.

(Refer Slide Time: 05:08)

Information Theory, Coding and Cryptography

Block Code

- A **block code** consists of a set of fixed length codewords.
- The fixed length of these codewords is called the **block length** and is typically denoted by n .
- Thus, a code of blocklength n consists of a set of codewords having n **components**.
- A block code of size M defined over an alphabet with q symbols is a set of M **q -ary sequences**, each of length n .
- In the special case that $q = 2$, the symbols are called **bits** (short for **binary digits**) and the code is said to be a binary code.
- Usually, $M = q^k$ for some integer k , and we call such a code an **(n, k) code**.

 Indian Institute of Technology, Delhi

8

Ranjan Bose
Department of Electrical Engineering

We now talk about block codes. And block code consists of a set of fixed length codewords, and that is the block length. So, the block length is typically denoted by n and all the codewords in this block code are of equal length. And this vectors of length n each one has a n components. And a block code of size M defined over an alphabet q symbols is a set of M q -ary sequences, each of length n .

So, you do not have to necessarily have a binary block code, you can have a ternary, a quaternary or a hexadecimal or anything that you like. But if you have q equal to 2, we call them bits, so binary digits and it is a binary code that we have. But in general M is equal to q raise to the power k for some integer k . And if the block length is n then such a code is called an n comma k code. Please recall code is a set of codewords each having a length of n .


(Refer Slide Time: 06:24)

Information Theory, Coding and Cryptography

Example

▪ Uncoded bits	Codewords
00	00000
01	10100
10	11110
11	11001

- Here $M = 4$, $n = 5$, $k = 2$
- Suppose we have to transmit a sequence of 1's and 0's using the above coding scheme.
- **How do we do that ?**

 Indian Institute of Technology, Delhi

9

Ranjan Bose
Department of Electrical Engineering

Let us look at a quick example. So, what is the job of an encoding procedure, it takes uncoded bits and makes codewords out of them. So, let us look at this very simple example where we have 2 input bits. So, k is equal to two 00, 01, 10, 11 and we can have four corresponding codewords right here listed.

So, M is equal to 4, this is the size; n is equal to 5, if you see each codeword is 5 bit long and k clearly is 2, where 2 is the uncoded bits. So, it is a 2 comma, so n is 5, so 5 comma 2 block code. But question is this has to be used in a practical situation, so how do we transmit a sequence of bits after encoding that is the question we will now address.

(Refer Slide Time: 07:23)


Information Theory, Coding and Cryptography

Min. Distance and Min. Weight

- The **minimum distance** of a code is the minimum **Hamming distance** between any two codewords.
- If the code C consists of the set of codewords $\{c_i, i = 0, 1, \dots, M-1\}$ then the minimum distance of the code is given by

$$d^* = \min d(c_i, c_j), i \neq j.$$

- An (n, k) code with minimum distance d^* is sometimes denoted by (n, k, d^*) .
- The **minimum weight** of a code is the smallest weight of any non-zero codeword, and is denoted by W^* .

 Indian Institute of Technology, Delhi10Ranjan Bose
Department of Electrical Engineering

But before that let us define two other quantities, the minimum distance and minimum weight; so, these are the characteristics of a code, the minimum distance of a code is a minimum hamming distance between any two codewords. So, then lots of codewords present because code is clearly a set of codewords, we compare take two codewords at a time for all possible combinations. So, if you see this definition d^* is the minimum distance is minimum of all the distances between c_i codeword i and c_j codeword j where i is not equal to j . In the case, when i is equal to j clearly the distance will be 0 as a distance itself is 0. We do not consider those cases; we only consider different codewords and find it all the possible differences and choose the minimum of that distance.


So, what is this minimum distance, intuitively tells us how different are two vectors, if they are similar the distance will be less, if they are very different the distance will be more, so it is kind of a similarity measure. It is such an important parameter that an n comma k code with a minimum distance d^* is sometimes denoted as n comma k comma d^* that is how we displayed. Now what we next define is the minimum weight of a code is a smallest weight of a non-zero codeword and it is denoted by w^* . So, again this is a property of a code which is a set of codewords, I calculate the weight of each vector each codeword in my set. And whatever is the smallest barding the all-zero codeword, I calculate the weight of each codeword the smallest of them is w^* .

(Refer Slide Time: 09:24)

Information Theory, Coding and Cryptography

Linear Block Code (LBC)

- A **linear code** has the following properties:
 - The **sum** of two codewords belonging to the code is also a codeword belonging to the code.
 - The all-zero codeword is always a codeword.
 - The minimum Hamming distance between two codewords of a linear code is equal to the minimum weight of any non-zero codeword, i.e., $d^* = w^*$.
- The presence of an all-zero codeword is a necessary but not a sufficient condition for linearity.



We also briefly looked at what is a linear block code LBC. So, a linear code has a following properties. The sum of two codewords belonging to the code is also codeword belonging to the code. So, this is very interesting take any two codewords add them up and you get another valid codeword. This is interesting, and this is the first constraint we are putting in. Again this is kind of adding some kind of an algebraic structure. We have put a constraint. How will this help us, well, this makes that only a certain special set of codewords are the valid codewords. We know it the receiver knows it, but the noise does not know this. So, we will use this property to check whether some codewords have undergone flipping of bits are erroneous or not.

The second constraint for a linear code is that the all- zero codeword must always be a valid codeword, it must be contained in that set in that code. And finally, it can be shown for a linear block code that the minimum weight is also equal to the minimum distance of the code that is d^* is equal to w^* . So, if I can verify these three properties, then I can declare that a certain code is indeed a linear block code.

But please note that the presence of an all-zero codeword is a necessary, but not sufficient condition. Now, here in this slide the most important thing that comes out is this word called sum. If we are saying that the sum of two codewords is a valid codeword, then we must define the sum, because sum is an operation. But, we better have a table which tells us how two elements add up because there must be a rule that we need to follow. A later part of this lecture, we will focus on how to add two codewords, this is not a trivial job we will talk about it.

(Refer Slide Time: 11:45)

Information Theory, Coding and Cryptography


Example

- The code $C = \{0000, 1010, 0101, 1111\}$ is a linear block code of block length $n = 4$.
- Observe that all the ten possible **sums** of the codewords

↓

$0000 + 0000 = 0000, \quad 0000 + 1010 = 1010, \quad 0000 + 0101 = 0101, \quad 0000 + 1111 = 1111,$
 $1010 + 1010 = 0000, \quad 1010 + 0101 = 1111, \quad 1010 + 1111 = 0101, \quad 0101 + 0101 = 0000,$
 $0101 + 1111 = 1010 \text{ and } 1111 + 1111 = 0000$

are in C and the all-zero codeword is in C .



Indian Institute of Technology,
Delhi

12

Ranjan Bose
Department of Electrical Engineering

But, let us look at a simple example of a linear block code. These are the four codewords block length is clearly 4. Why is that look each one has 4 bits, so it is n is equal to 4, but in order to verify whether it is an indeed a linear code as well it is clearly a block code, because the block length is 4. But if it is a linear block code it must have the following two constraints satisfied.

One is the all-zero codeword must be present which is true; and sum of any two codewords is also valid codeword. So, we try out this there are only a finite ways of choosing 2 out of this 4. So, we once we get that we can try and add all the possible combinations. And if we do this exercise we indeed find that any two codewords if you add the resulting vector is again 1 of the 4 valid codewords. Thus we can declare that this C is indeed a linear block code ok.

(Refer Slide Time: 13:03)

Information Theory, Coding and Cryptography


Example

- The code $C = \{0000, 1010, 0101, 1111\}$ is a linear block code of block length $n = 4$.
- The minimum distance of this code is $d^* = 2$.
- In order to verify the minimum distance of this linear code we can determine the distance between **all pairs of codewords**

(which is $\binom{4}{2} = 6$ in number):

$d(0000, 1010) = 2, \quad d(0000, 0101) = 2, \quad d(0000, 1111) = 4$
 $d(1010, 0101) = 4, \quad d(1010, 1111) = 2, \quad d(0101, 1111) = 2$

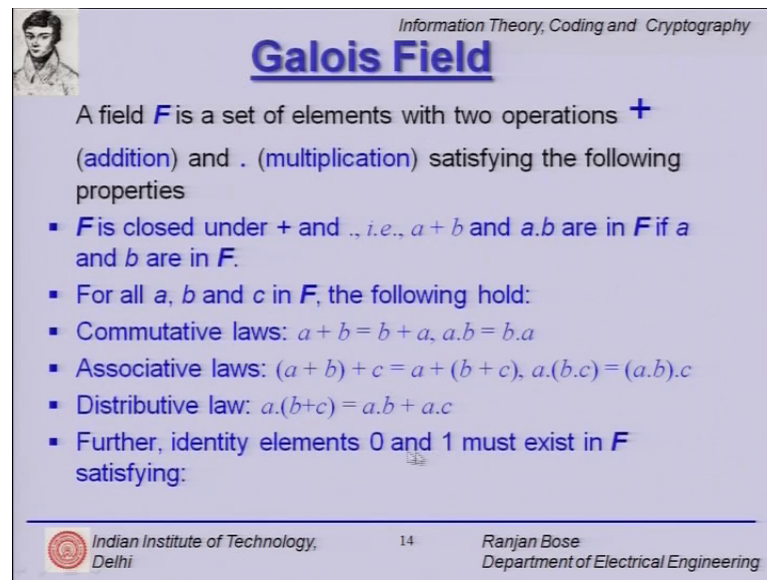
- **We observe that the minimum distance of this code is indeed 2.**

 Indian Institute of Technology, Delhi 13 Ranjan Bose
Department of Electrical Engineering

Now, what we can do is talk about the minimum distance of this code and we will shortly show that this is minimum distance is 2. How do we do that, we take two codewords and find the distance hamming distance. And once we do for all possible combinations, here it is 4 choose 2 is equal to 6 is the total number of pairs we can form from 4 valid codewords, we find that the distances are 2 2 4 4 2 2 clearly the minimum distance is 2.

But we have also shown that d^* is equal to w^* , so barring the all-zero codeword we find out the minimum weights. There are only 3 possible non-zero codewords. Weight for this is 2, where they are two non-zero elements, weight for this is 2, 2 nonzero elements, weight is 4. So, the minimum weight is 2 which is equal to the minimum distance. So, this linear block code satisfies all the properties.

(Refer Slide Time: 14:13)



Information Theory, Coding and Cryptography

Galois Field

A field F is a set of elements with two operations $+$ (addition) and \cdot (multiplication) satisfying the following properties

- F is closed under $+$ and \cdot , i.e., $a + b$ and $a \cdot b$ are in F if a and b are in F .
- For all a, b and c in F , the following hold:
 - Commutative laws: $a + b = b + a$, $a \cdot b = b \cdot a$
 - Associative laws: $(a + b) + c = a + (b + c)$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
 - Distributive law: $a \cdot (b + c) = a \cdot b + a \cdot c$
- Further, identity elements 0 and 1 must exist in F satisfying:

Indian Institute of Technology, Delhi 14 Ranjan Bose Department of Electrical Engineering

Now, we move ahead and we try to answer the basic question how do we add, and for that we need a brief introduction to something called as Galois fields. So, let us quickly define what is a field of field F is a set of elements with two operations addition and multiplication satisfying the following properties; So, what are we looking at we are looking at a set of elements ok, and two operations defined over them. First is the F is closed under addition and multiplication, that is a plus b and a into b are in F if a and b are in F .

Suppose a and b are two elements then the addition also is contained in F and that the product is also contained in F . This property number one, then the certain basic laws must hold like commutative a plus b is equal to b plus a similarly a into b is b into a . See these are so basic we take them for granted, but since we are defining the properties of a Galois field it is worthwhile looking at all of this. Then the associative law a plus b parentheses plus c is equal to a plus b plus c , similarly a into parentheses open b into c is equal to parentheses a into b into c and distributive a into b plus c is nothing but a into b plus a into c .


So, this is like our second nature we use them all the time, but they must be a valid under this definition. What else there are few more properties let us go we say that identity elements 0 and 1 must exist in F satisfying the 2 properties. So, what are we saying other

than all the elements in the set 2 elements must necessarily exist with some special properties, these two elements are 0 and 1.

(Refer Slide Time: 16:25)


Theory, Coding and Cryptography

Galois



1811 – 1832

- $a + 0 = a$
- $a \cdot 1 = a$
- For any a in F , there exists an additive inverse ($-a$) such that $a + (-a) = 0$.
- For any a in F , there exists a multiplicative inverse (a^{-1}) such that $a \cdot (a^{-1}) = 1$.
- The above properties are true for fields with both finite as well as infinite elements.
- A field with a finite number of elements (say, q) is called a **Galois Field** (pronounced Galva Field) and is denoted by $GF(q)$.
- If only the first seven properties are satisfied, then it is called a **ring**.

 Indian Institute of Technology,
Delhi

15

Ranjan Bose
Department of Electrical Engineering

And what are the two properties that we want them to satisfy any element a in the set plus 0 is a , and any element a in the set into 1 is a ok. Again they appear to be trivial, but they will hold value very shortly another thing is that for any element a in F , there exists an additive inverse minus a such that a plus minus a is 0 that is a minus a is also an element. If a plus b is equal to 0, then b is called the additive inverse of a ; this must exist. So, each and every element must have an additive inverse.

Similarly, for any element a in F there must exist a multiplicative inverse of course, except for 0. So, if a into b is 1 because clearly 1 is within the set 0 and 1 is defined. So, if a into b is 1 then b is the multiplicative inverse of a . Please note this set of properties that we have covered so far are true for fields with both finite as well as infinite elements. So, we are not saying that it must necessarily have a finite number of elements. What is interesting is that Galois fields may not exist for all any arbitrary number of elements. For example, Galois field for 2, 3, 4, 5 exist, but Galois field for 6 does not exist.

We will show later that if the number is a prime or a prime power, then the Galois field exists. So, a field with a finite number of elements say q is called a Galois Field, it is pronounced Galva based on this guy who was a French; and it is denoted by $GF(q)$. And please note if only the first 7 of the 8 properties you have discussed, that is the constraint

of multiplicative inverse is thrown out of the window then it is no longer a field, but it is only called a ring. So, we have defined together what is a Galois field and what is a ring. Sometimes the Galois field is simply called a field. But please note that this guy Galois in a very short duration of time has made indelible contribution to mathematics ok. And we will be using Galois fields over and over again for our course encoding theory.

(Refer Slide Time: 19:17)

Information Theory, Coding and Cryptography

Example

- Consider $GF(4)$ with 4 elements $\{0, 1, 2, 3\}$.
- The addition and multiplication tables for $GF(4)$ are

+	0	1	2	3	.	0	1	2	3
0	0	1	2	3	0	0	0	0	0
1	1	0	3	2	1	0	1	2	3
2	2	3	0	1	2	0	2	3	1
3	3	2	1	0	3	0	3	1	2

Addition Table Multiplication Table

- It should be noted here that the addition in $GF(4)$ is **not** modulo 4 addition.

Indian Institute of Technology, Delhi
16
Ranjan Bose
Department of Electrical Engineering

So, let us look at a simple example. Let us look at Galois field $GF(4)$ with 4 elements. Now, what are these elements, we have 0 and 1 they must be present and this is 2 and 3, where I could have labeled them a and b, it does not matter for a sake of convenience only we are saying 0, 1, 2 and 3 are the 4 elements. Now, this is a set, so there is nothing like this is holier than thou; 2 is not bigger than 1 this greater than or less than operation is not defined.

Only two operations make sense, the addition and multiplication. So, if we were to define this, I need to give you the addition table the rules for addition and rules for multiplication. Why did we start on all of this remember we wanted to look at linear block codes, where some of two codewords is a valid codewords.

Now, codewords are made out of elements taken from this set. So, some the addition must be defined. So, let us look at the addition and a multiplication tables for $GF(4)$. Now in order to define the addition and a multiplication table we write out the elements 0, 1, 2, 3 on the horizontal and the vertical axis. So, it is a matrix it is a kind of a table more

importantly I lookup table, where I need to know what happens when 0 adds with 0, then entry will be here 0 adds with 1, 2 adds with 3, 3 adds with 2 and so on and so forth.

Now, we only need to fill so one half of the element, because this table must be symmetric. We have established that $a + b$ is the same as $b + a$, $a \cdot b$ is $b \cdot a$. And therefore, we can start filling in the elements of this addition table first. So, if we were to fill out the numbers we can write them as follows. So, it tells you how to add. For example, if I want to know what is 2 plus 1, well it is 3. What is 3 plus 1, it is 2, what is 1 plus 1 it is 0. In fact, this diagonal is all 0's, so self addition is 0 ok.

So, for example, what is the rule for adding 3 with 3, I get a 0, who tells me that, well this table is skillfully constructed that it follows all the properties of the Galois field that we discussed in the previous slide. It is not easy just if I can interchange any two numbers it will seem to be a field, because this property addition property will violate some or the other rule. Let us look at the second operation the multiplication operation leading to this following multiplicative table.

Here please note that a product with 0 is 0. So, first column and first row is all 0's. And then multiplication with 1 is the self, so 1 2 3 and 1 2 3. So, essentially these 4 elements have really being thought and filled out. If you go to the addition table again you can look at an additive inverse, additive inverse means if $a + b = 1$ $a + b = 0$ then a is additive inverse of b .


So, if we have a 0 in each one of the rows and columns. So, all the elements do have an additive inverse. Similarly a multiplicative inverse other than the 0 element must exist right. So, 3 is the multiplicative inverse of 2, because $3 \cdot 2 = 1$, 2 is the multiplicative inverse of 3, 1 is the multiplicative inverse of 1. So, a multiplicative inverse also exists for all elements other than 0. Now, please note it is not trivial to construct it. For example, you just cannot say that GF 4 is just a modular for arithmetic. Because, if it were so, then $3 + 2$ right is 1, but $3 + 3$ you would not get a 0. So, this is not simply a modulo for arithmetic.

(Refer Slide Time: 24:07)

Information Theory, Coding and Cryptography

Linear Span

- Let \mathbf{S} be a set of vectors of length n whose components are defined over $GF(q)$.
- The set of all linear combinations of the vectors of \mathbf{S} is called the linear span of \mathbf{S} and is denoted by $\langle \mathbf{S} \rangle$.
- The linear span is thus a subspace of $GF(q^n)$, generated by \mathbf{S} .
- Given any subset \mathbf{S} of $GF(q^n)$, it is possible to obtain a linear code $\mathbf{C} = \langle \mathbf{S} \rangle$ generated by \mathbf{S} , consisting of precisely the following codewords:
 - all-zero word,
 - all words in \mathbf{S} ,
 - all linear combinations of two or more words in \mathbf{S} .

 Indian Institute of Technology, Delhi 17 Ranjan Bose
Department of Electrical Engineering

Let us quickly define something called as a linear span which we might use later. So, let \mathbf{S} be a set of vectors of length n whose components are defined over $GF(q)$. So, the terminology that we use is defined over $GF(q)$, Galois field with q elements. The set of all linear combinations of the vectors of \mathbf{S} is called the linear span of \mathbf{S} and is denoted by this angular bracket $\langle \mathbf{S} \rangle$.


The linear span is thus a subspace of $GF(q^n)$ generated by \mathbf{S} . And given any subset \mathbf{S} of $GF(q^n)$, it is possible to obtain a linear code \mathbf{C} as a span generated by \mathbf{S} , consisting of precisely the following codewords. The all-zero words, all-words in \mathbf{S} , all-linear combinations of 2 or more words in \mathbf{S} ; So, let us look at this definition of generating codewords \mathbf{C} with a simple example.

(Refer Slide Time: 25:15)

Information Theory, Coding and Cryptography

Example

- Let $S = \{1100, 0100, 0011\}$.
- All possible linear combinations of S are
 $1100 + 0100 = 1000$, $1100 + 0011 = 1111$,
 $0100 + 0011 = 0111$,
 $1100 + 0100 + 0011 = 1011$.
- Therefore, $C = \langle S \rangle =$
 $\{0000, 1100, 0100, 0011, 1000, 1111, 0111, 1011\}$.
- The minimum distance of this code is $w(0100) = 1$.

 Indian Institute of Technology, Delhi
18
Ranjan Bose
Department of Electrical Engineering

So, let S be the following three vectors. So, we write out all the possible linear combinations of S . So, I add them up, so $1\ 1\ 0$ and $0\ 1\ 0$, I get this, I can add this with this, I get this or I can add this with this and I get this. So, all these combinations are there and then we can all add all three of them and I get this. So, I get this 4 possible outcome. So, the code is defined as all the inherent elements C as follows. And this is kind of a trivial code with minimum distance equal to 1.

(Refer Slide Time: 26:08)


Information Theory, Coding and Cryptography

Example

- Let $S = \{12, 21\}$ defined over $GF(3)$.
- The addition and multiplication tables of field $GF(3) = \{0, 1, 2\}$ are given by:

+	0	1	2	·	0	1	2
0	0	1	2	0	0	0	0
1	1	2	0	1	0	1	2
2	2	0	1	2	0	2	1

- All possible linear combinations of 12 and 21 are:
- $12 + 21 = 00$, $12 + 2(21) = 21$, $2(12) + 21 = 12$.
- Therefore, $C = \langle S \rangle = \{00, 12, 21, 00, 21, 12\} = \{00, 12, 21\}$.

 Indian Institute of Technology, Delhi
19
Ranjan Bose
Department of Electrical Engineering

Now, let us look at a code which is defined over GF 3. So, Galva field with 3 elements. Let S is equal to 1 2 and 2 1. And these are the addition tables for the element 0, 1 and 2. Now, it can be noted that this time, the modulo three arithmetic holds. So, in general we will see that if 3 is a if GF q, q is a prime number in this case q is equal to 3, then modular arithmetic works.

But Galva fields are defined for prime numbers and prime powers, GF 4 was 2 raise the power 2. So, it was a prime power and hence we did not find a modular arithmetic to hold. But, here it is a prime number and so modular arithmetic does hold. So, again we find all possible linear combination and therefore, C is defined as 0 0, 1 2 and 2 1 are the possible codewords for this case. It is just one way, but this is clearly not a very methodical way to move forward.

(Refer Slide Time: 27:22)


Information Theory, Coding and Cryptography

Generator Matrix

- The generator matrix converts (encodes) a vector of length k to a vector of length n .
- Let the input vector (uncoded symbols) be represented by i .
- The coded symbols will be given by

$$c = iG$$

- where c is called the codeword and i is called the information word.

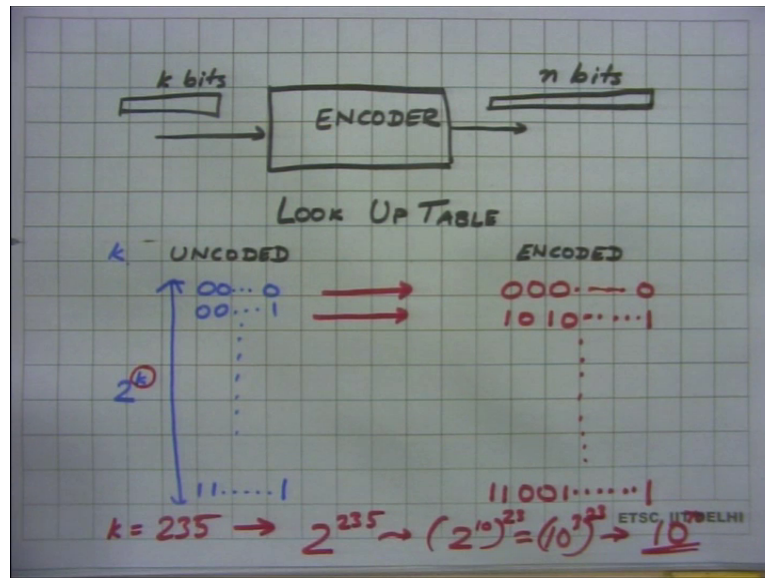


Indian Institute of Technology,
Delhi

Ranjan Bose
Department of Electrical Engineering

So, we now ask ourselves the most basic question is there a smart way to generate a linear block code all right. So, what is the motivation for this? Well the generator matrix converse that is encodes a vector of a length k to a vector of a length n. Let us look at an example, here.

(Refer Slide Time: 27:57)



Suppose I have my linear block code. So, this is an encoder, and the job of the encoder is to take k bits and convert them into a longer n bits. Now, one way is to do a lookup table, question is it is still efficient for large k . So, what is a lookup table well you have uncoded and you have encoded. Suppose, I have got a certain value of k so, here I will start with 0 0 0, 0 0 1 up to 1 1 1.

So, this is 2^k , for each 1 of them I will have a valid codeword. So, it could be 0 0 0 this 1 will have 1 0 1 0 0 something and so and so forth. Now, the problem is this k , if this k equal to say 235 then we are looking at 2^{235} . Now, this is a huge table it will take a lot of memory, I do not know whether my smart phone has a ability to carry this larger memory.

So, this lookup table is nice for small values of k , but the moment k becomes large, we are in trouble. We cannot store this big thing, this big lookup table. And even if you could searching it would be even more difficult. So, storage and search both are highly inefficient for large values of k . Because if you if you try to look at it, how big is 2^{235} you would have to do 2^{235} .

And then 23 roughly this is equal to 10^3 going to 23 and then this will go to 10^{23} close to 70 or something. This is a huge number, I mean this is not really practical to put them in the memory so, we must design our smarter way to do things. So, we go back to our slide and see, what can we do.

So, let us say that the input vector the uncoded symbols are represented by the information vector i . And the coded symbols are given by c , would not it be great to have c equal to i into all magical matrix G which we will called a generator matrix. So, i is my vector of length k . G should necessarily be a matrix k cross n , such that c becomes one cross m , where c is the codeword and i is the information word.

So, the generator matrix takes in the information word and converts into a codeword. In other words, it takes a vector of length k and converts it to a vector of a length n , but care must be taken that it indeed generates a linear block code, that is the all 0 codeword must be a valid codeword and all codewords any two codewords added together should be another valid codeword. Questions can it deliver, can it give you such a thing.

(Refer Slide Time: 32:36)

Information Theory, Coding and Cryptography

Generator Matrix

$$c = iG$$

- The generator matrix provides a concise and efficient way of representing a linear block code.
- The $n \times k$ matrix can generate q^k codewords.
- Thus, instead of having a large look-up table of q^k codewords, one can simply have a generator matrix.

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, let us look at the c equal to i into G . The generator matrix provides a concise and efficient way to represent a linear block code ok. In fact, is the only practical way for any large sized k , we do not have a choice. The n cross k matrix can generate q raise the power k codewords. So, in the previous example we only talked about binary, but please remember that my vectors code as well be a non-binary.

Thus, instead of having a large look-up table, large impractical look up table of q raise power k codewords one can simply have a generator matrix. So, just let us look at this a little bit more carefully, what we have done so far.

(Refer Slide Time: 33:42)

$$C = i \cdot G$$

$1 \times n$ $1 \times k$ $k \times n$ 235×255

$(255, 235)$
 (n, k)

$2^{235} \times 255$

ETSC, IIT DELHI

We have here c equal to i times G . Now, my c is 1 cross n . This i is 1 cross k ; and this G is k cross n . So, this 1 cross k multiplied with a matrix of k cross n gives me 1 cross n . The let us see, what is the problem or advantage with this kind of a representation. I do not have to worry about, this k being large here because input is coming in, it is none of my worries.

Now question is, what do we do with the generator matrix, how big is this. Well, whatever be is the size of k , k into n , n is greater than k . And so my generator matrix is nothing but of k into n , this size is highly manageable, because even if k is 235 and n is 255, suppose we are talking about this is my n comma k code, this is a practical number. So, if k is still my 235, the size of this G matrix is nothing but 235 into 255. Definitely I can store these number of bits, it is trivial.

So, suddenly a very large problem of storing 2 raise power 235 into n that is 255 bit long table lookup table has reduced to merely storing just a few 1000 bits. So now, we have converted a very impractical problem to a practical problem that touches every day of our lives. We come back to our slides. And we now show, that instead of having a large lookup table of q raise power k codewords, one can simply have a generator matrix.

(Refer Slide Time: 36:13)

Example

Consider a generator matrix $G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

$$c_1 = [0 \ 0] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [0 \ 0 \ 0] \quad c_2 = [0 \ 1] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [0 \ 1 \ 0]$$

$$c_3 = [1 \ 0] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1 \ 0 \ 1] \quad c_4 = [1 \ 1] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1 \ 1 \ 1]$$

Let us look at this example. Consider this G matrix, where G is equal to a 2 cross 3. And I would like to have my first codeword out. So, first codeword takes my i vector and multiplies it with G to yield a 3 bit long vector. So, a no surprise 0 0 was converted into 0 0 0. Now, similarly, we can take that mixed information vector, information void is 0 1 again multiplied with G matrix I get another 3 bit.

Similarly, I can have the third vector c 3 as the codeword number 3 multiplying with 1 0 and finally we can have the fourth one. K is equal to 2 so, there are only 4 possible input vectors the information word 0 0, 0 1, 1 0, 1 1. And corresponding to those, we have got four possible codewords right. Block length is 3, so this is a 3 comma 2 code; and the size of the generator matrix is simply a 2 cross 3.


(Refer Slide Time: 37:33)

Information Theory, Coding and Cryptography

Example

- Hence the generator matrix $G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
- Generates the code
 $C = \{000, 010, 101, 111\}.$

▶

 Indian Institute of Technology, Delhi Ranjan Bose
Department of Electrical Engineering

So, the code that is generated, by this generator matrix is 0 0 0, 0 1 0, 1 0 1, 1 1 1. What are these, these are the four encircled codewords here written out there.

(Refer Slide Time: 37:53)


Information Theory, Coding and Cryptography

Example

- Hence the generator matrix

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

- We observe that this is a (3, 2) code from the fact that the dimension of the generator matrix is 3×2 .

 Indian Institute of Technology, Delhi Ranjan Bose
Department of Electrical Engineering

Note that that this is a 3 comma 2 code from the fact that the dimension of the generator matrix is indeed 3 cross 2 ok. So, this is a simple example, which tells us that yes the generator matrix can give you a valid code.

(Refer Slide Time: 38:10)

The slide is titled "Information Theory, Coding and Cryptography" at the top right. The main heading is "Summary" in a large, bold, blue font. Below it, there is a bulleted list with three items: "Linear Block Codes", "Generator Matrix", and "Examples". At the bottom of the slide, there is a footer containing the Indian Institute of Technology Delhi logo, the text "Indian Institute of Technology, Delhi", the page number "24", and the name "Ranjan Bose" followed by "Department of Electrical Engineering".

So, we now come to the basic summary of today's lecture. We have had a very quick recap about linear block codes. Then, we moved on to a very interesting concept of the generator matrix, why generator matrix is an efficient way to represent linear block codes. And then, we looked at some examples, which tell you whether the codes are indeed a linear or they satisfying the properties of linear block codes and how you can use a simple generator matrix to generate a code.

With that, we come to the end of today's lecture.