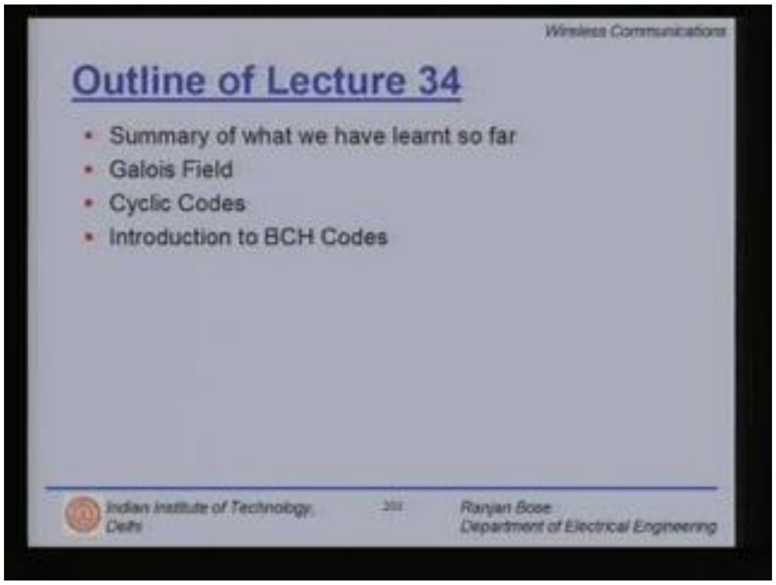
`Wireless Communications
Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi
Lecture No. # 34
Coding Techniques for Mobile Communications (Continued)

Welcome to the next lecture on wireless communications. Today we will talk about various coding techniques for mobile communication. First a brief outline for today's talk. We will first summarize what we have learnt so far followed by a study of the Galois field. We will then move over to an interesting domain of cyclic codes which happens to be a sub class of linear block codes and finally we will talk about the BCH codes.

(Refer Slide Time: 00:01:31 min)


Wireless Communications

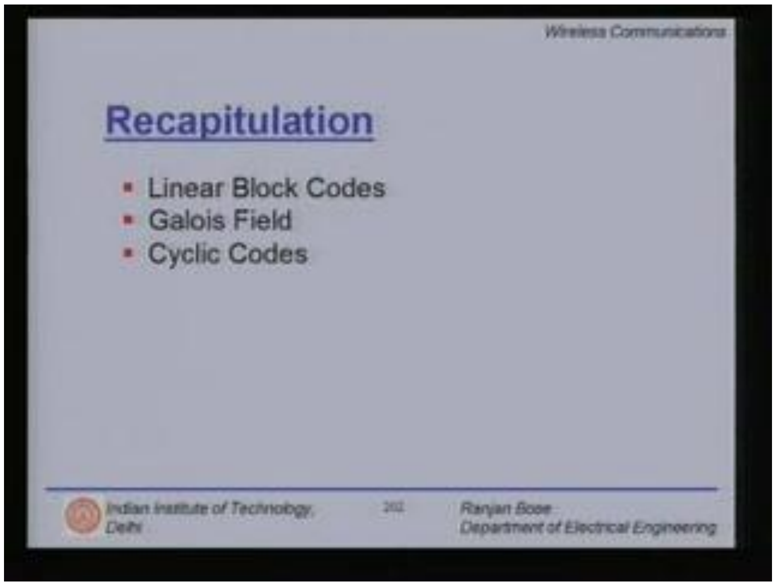
Outline of Lecture 34

- Summary of what we have learnt so far
- Galois Field
- Cyclic Codes
- Introduction to BCH Codes

Indian Institute of Technology, Delhi — 20I — Ranjan Bose, Department of Electrical Engineering

In the previous lecture we talked about linear block codes, we realized that linear block codes are LBC's have a fixed block length N and you can have a generator matrix to generate the entire set of code words. We then visited briefly Galois field and had a very short introduction to cyclic codes.

(Refer Slide Time: 00:01:52 min)


Wireless Communications

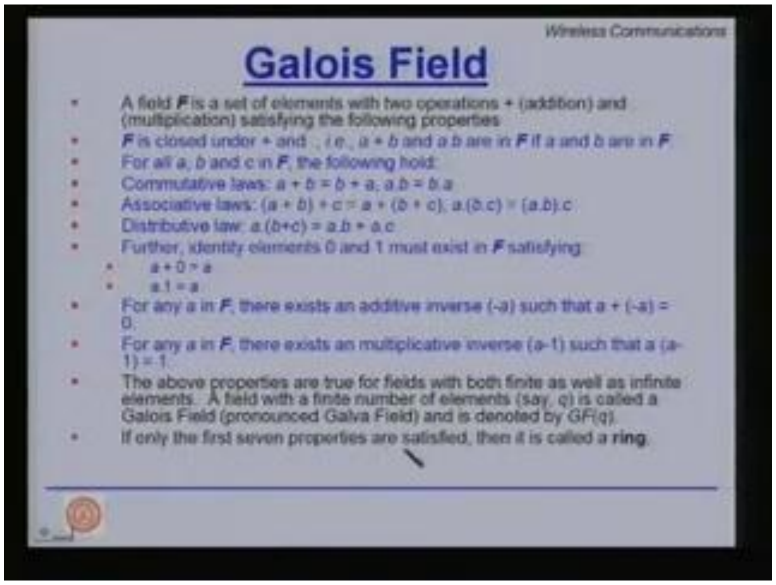
Recapitulation

- Linear Block Codes
- Galois Field
- Cyclic Codes

Indian Institute of Technology, Delhi   202   Ranjan Bose
Department of Electrical Engineering

Now let us refresh our memories about the Galois field theory, we know that a field F is a set of elements with two operations addition and multiplication and the elements satisfy the following 8 properties. First of all F is closed under summation and multiplication that is a + b and a into b are contained in the set. Also for all a, b and c within F the commutative law, the associative law and the distributive law holds good which is a + b is equal to b + a, (a + b) + c is equal to a + (b+c) are equal, under distributive law a times (b + c) is a times b + a times c these must hold. Further two interesting identity elements exists, one is a +0, the 0 should give back a and a times 1 should give back a; also there exists an additive inverse and the multiplicative inverse if we have to talk about a field.

(Refer Slide Time: 00:02:20 min)

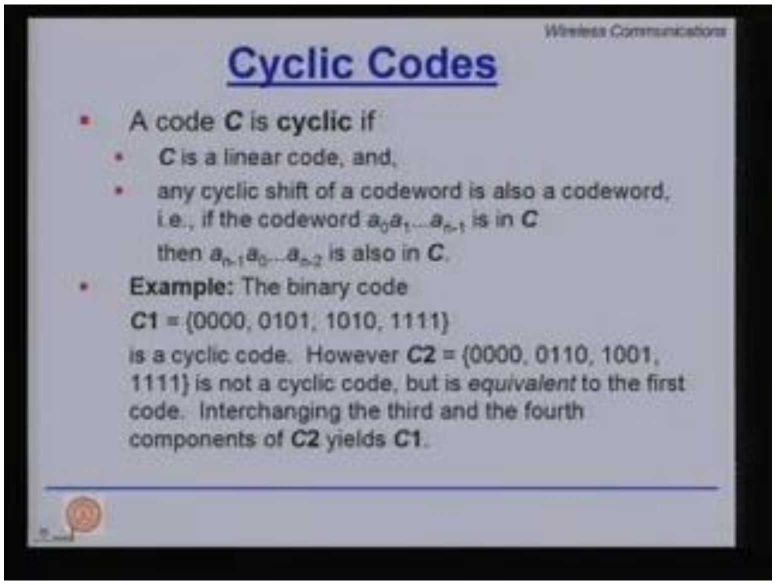

Wireless Communications

Galois Field

- A field ***F*** is a set of elements with two operations + (addition) and . (multiplication) satisfying the following properties
- ***F*** is closed under + and ., *i.e.*, *a* + *b* and *a*.*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.b = b.a
- Associative laws: (a + b) + c = a + (b + c); a.(b.c) = (a.b).c
- Distributive law: a.(b+c) = a.b + a.c
- Further, identity elements 0 and 1 must exist in ***F*** satisfying:
  - a + 0 = a
  - a.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 an multiplicative inverse (a-1) such that a.(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**.

If the last property doesn't hold for the elements that is there is not the presence of multiplicative inverse but only an additive inverse we call it a ring. If only the first 7 properties are satisfied it is called a ring otherwise it's a field. Now we learned last time that the objective of error control coding is to add redundancies in a known manner; in a known manner such that there is enough structure, algebraic structure within the code and whenever there is an error this structure breaks and if we are smart enough we should be able to detect an error and hopefully reconstruct it back. So that is the whole idea of adding structures.

(Refer Slide Time: 00:03:54 min)



Wireless Communications

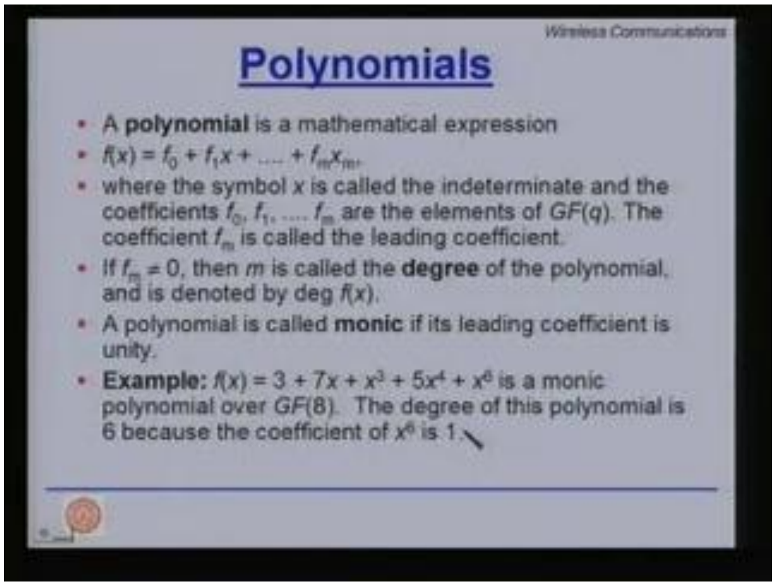
Cyclic Codes

- A code **C** is **cyclic** if
  - ***C*** is a linear code, and,
  - any cyclic shift of a codeword is also a codeword, i.e., if the codeword $a_0a_1...a_{n-1}$ is in ***C*** then $a_{n-1}a_0...a_{n-2}$ is also in ***C***.
- **Example:** The binary code **C1** = {0000, 0101, 1010, 1111} is a cyclic code. However ***C2*** = {0000, 0110, 1001, 1111} is not a cyclic code, but is *equivalent* to the first code. Interchanging the third and the fourth components of ***C2*** yields ***C1***.

Cyclic code is one step forward, it is a sub class of linear block code that it has one additional requirement an additional constraint, an additional structural requirement. What is that? A code C is cyclic if C is a linear code and on top of that any cyclic shift of a codeword is also a codeword. So if a_0 a_1 so and so forth till a_{n-1} forms a elements of codeword then any cyclic rotation if you put a_{n-1} before like this and then a_0 a_1 up to a_{n-2} this itself is also a valid codeword if it is a cyclic code.

Here is an example of binary cyclic code; here if you can see all of the elements if you rotate them to the left or to the right, you end up with another valid codeword. Please note this is another example of a linear block code but here it is not a cyclic code. The two properties of a linear block code are, the all zero codeword must exists as a valid codeword and some of any two codewords is also a valid codeword, on top of that any cyclic rotation of a codeword gives you another valid codeword that makes it is a cyclic code.

(Refer Slide Time: 00:05:57 min)



Wireless Communications

Polynomials

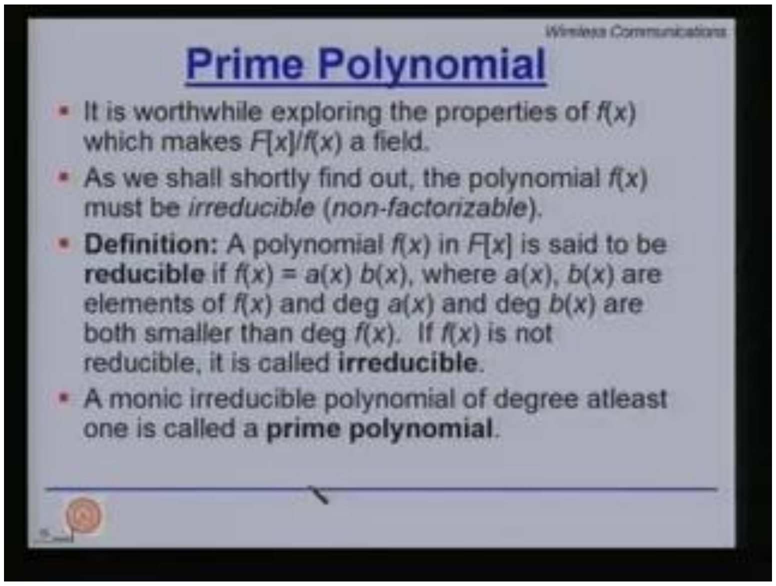
- A **polynomial** is a mathematical expression
- $f(x) = f_0 + f_1x + \dots + f_mx_m$,
- where the symbol x is called the indeterminate and the coefficients $f_0, f_1, \dots f_m$ are the elements of $GF(q)$. The coefficient f_m is called the leading coefficient.
- If $f_m \neq 0$, then m is called the **degree** of the polynomial, and is denoted by deg $f(x)$.
- A polynomial is called **monic** if its leading coefficient is unity.
- **Example:** $f(x) = 3 + 7x + x^3 + 5x^4 + x^6$ is a monic polynomial over $GF(8)$. The degree of this polynomial is 6 because the coefficient of x^6 is 1.

Now let us take a brief mathematical detour and get into the domain of polynomials because we will learn that it is very easy and logical to express a codeword which is a cyclic codeword as a polynomial. There is a one to one correspondence. So what is a polynomial? A polynomial is a mathematical expression, it can be denoted as follows and the symbol X is called indeterminate and the coefficients f_0, f_1 up to f_m are the elements of any GF (q), a Galois field. So we can define a polynomial over a field. It depends from what are you taking the coefficients from. If it is a binary polynomial then GF (2) is in question and then you have f_0 f_1 etc either a zero or a one, if it is GF (3) then your f_i can be 0, 1, 2.

If f_m the coefficient of the highest power is not zero then m is called the degree of the polynomial as denoted by degree f (x). A polynomial is called monic if its leading coefficient is unity. So here is a polynomial over GF (8) so you can see the coefficients can take values some 0, 1, 2 up to 7 but the highest power X^6 as a coefficient unity, hence this is the monic polynomial of degree 6.

Now let's have some definitions now let us explore the properties of f(x) which makes this F[X] divided by f(x) of field. So what is this capital F[X]? This is a set of polynomials and if you divide it by f(x) then you are doing a modulo operation that is whatever is the remainder after you divide any element, any symbol within this by f(x). You are left with a remainder and we are talking about that's it, what we are interested in finding out is the properties of f(x) which makes this thing of field that is it satisfies the 8 properties of a Galois field.

(Refer Slide Time: 00:07:36 min)



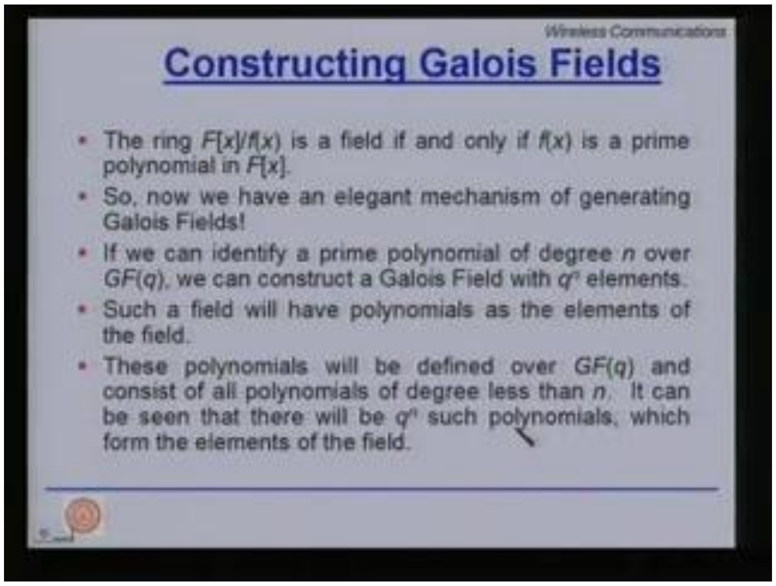
Wireless Communications

Prime Polynomial

- It is worthwhile exploring the properties of $f(x)$ which makes $F[x]/f(x)$ a field.
- As we shall shortly find out, the polynomial $f(x)$ must be *irreducible* (*non-factorizable*).
- **Definition:** A polynomial $f(x)$ in $F[x]$ is said to be **reducible** if $f(x) = a(x)\,b(x)$, where $a(x)$, $b(x)$ are elements of $f(x)$ and deg $a(x)$ and deg $b(x)$ are both smaller than deg $f(x)$. If $f(x)$ is not reducible, it is called **irreducible**.
- A monic irreducible polynomial of degree atleast one is called a **prime polynomial**.

We will shortly find out that the polynomial f(x) must be irreducible which is the polynomial version of non factorizable. That is you cannot represent it as a multiplication of two other polynomials. So polynomial f(x) in F[x] is said to be reducible if f(x) is a(x) times b(x) where both a(x) and b(x) are elements of f(x) and degree a(x) and degree b(x) are both smaller than degree f(x), it doesn't hold then it is called irreducible. A monic irreducible polynomial of degree at least one is called a prime polynomial, we will use this definition many times.

(Refer Slide Time: 00:09:26 min)



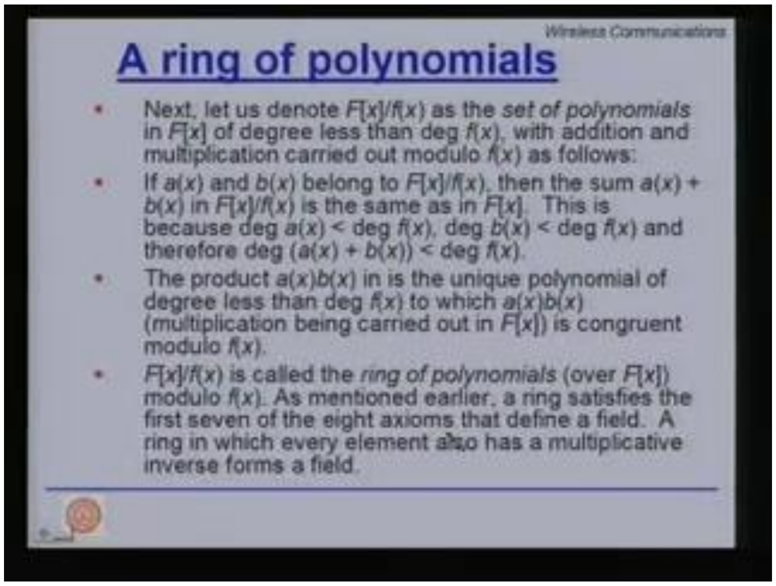
Wireless Communications

Constructing Galois Fields

- The ring $F[x]/f(x)$ is a field if and only if $f(x)$ is a prime polynomial in $F[x]$.
- So, now we have an elegant mechanism of generating Galois Fields!
- If we can identify a prime polynomial of degree n over $GF(q)$, we can construct a Galois Field with q^n elements.
- Such a field will have polynomials as the elements of the field.
- These polynomials will be defined over $GF(q)$ and consist of all polynomials of degree less than n. It can be seen that there will be q^n such polynomials, which form the elements of the field.

Now continuing with our mathematical detour, the ring F[x] over f(x) is a field if and only if f(x) is a prime polynomial in F[x]. So we are trying to find out when will this set be a field and the claim is this will only happen if and only if small f(x) is a prime polynomial. So if this is true we have an elegant mechanism of generating Galois fields. If we can identify a prime polynomial of degree n over GF (q), we can construct a Galois field with q^n elements. Such a field will have polynomials as the elements of the field. These polynomials will be defined over GF (q) and consist of all polynomials of degree less than n. We are talking about taking modulo f (x), small f (x) has highest power n so you can have the coefficients f_0, f_1, f_2 up to f_{n-1} and all the polynomials that will remain if you perform this operation, will have the degree less than n but there be n coefficients. Each of the coefficients can take value as one of the q elements of the Galois field. So there will be q^n such polynomials.

(Refer Slide Time: 00:11:18 min)


Wireless Communications

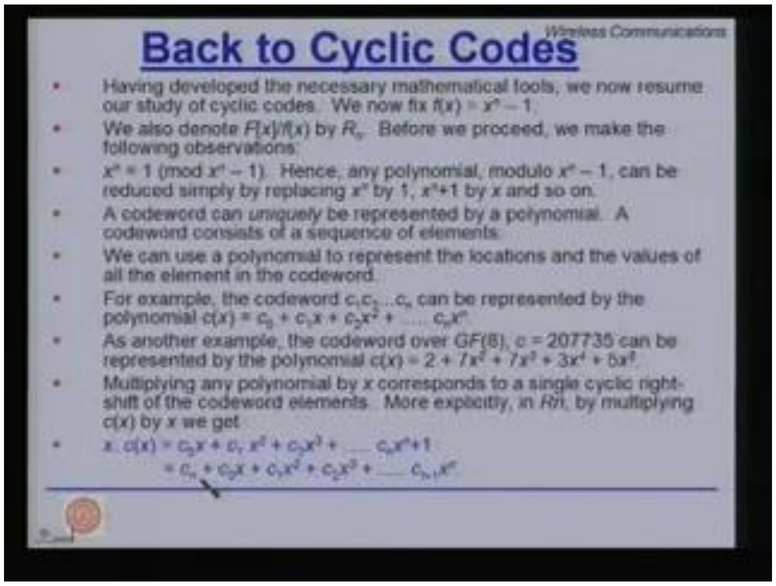
A ring of polynomials

- Next, let us denote *F[x]/f(x)* as the *set of polynomials* in *F[x]* of degree less than deg *f(x)*, with addition and multiplication carried out modulo *f(x)* as follows:
- If *a(x)* and *b(x)* belong to *F[x]/f(x)*, then the sum *a(x)* + *b(x)* in *F[x]/f(x)* is the same as in *F[x]*. This is because deg *a(x)* < deg *f(x)*, deg *b(x)* < deg *f(x)* and therefore deg (*a(x)* + *b(x)*) < deg *f(x)*.
- The product *a(x)b(x)* in is the unique polynomial of degree less than deg *f(x)* to which *a(x)b(x)* (multiplication being carried out in *F[x]*) is congruent modulo *f(x)*.
- *F[x]/f(x)* is called the *ring of polynomials* (over *F[x]*) modulo *f(x)*. As mentioned earlier, a ring satisfies the first seven of the eight axioms that define a field. A ring in which every element also has a multiplicative inverse forms a field.

So let us denote capital F[x] over f(x) as a set of polynomials in F[x] of degree less than the degree f(x) with addition and multiplication carried out modulo f(x). So you divided and whatever may be the remainder is the element. Now if a(x) and b(x) belong to capital F[x] by f(x) then the sum a(x) + b(x) in F[x] by f(x) is the same as in F[x], this is because we have already put the degree of a(x) is less than f(x) and degree of b(x) is also less than f(x). Carrying on this philosophy the product a(x) times b(x) is the unique polynomial of degree less than degree f(x) to which a(x) times b(x) multiplication is carried out in F[x] is congruent modulo f(x).

So capital F[x] over f(x) is called the ring of polynomials over F[x] modulo f(x). So what are we looking at? This capital F[x] over f(x) is a set of polynomials all of which have degree less than that of degree of f(x) because if it is a remainder after you carry out the division and we are talking about when will this set form a ring and when will it form a field. As mentioned earlier a ring satisfies the first seven of the 8 axioms that define a field. A ring in which every element also has a multiplicative inverse forms a field.

(Refer Slide Time: 00:13:00 min)



Back to Cyclic Codes

Wireless Communications

- Having developed the necessary mathematical tools, we now resume our study of cyclic codes. We now fix $f(x) = x^n - 1$.
- We also denote $F[x]/f(x)$ by R_n. Before we proceed, we make the following observations:
- $x^n = 1 \pmod{x^n - 1}$. Hence, any polynomial, modulo $x^n - 1$, can be reduced simply by replacing x^n by 1, x^{n+1} by x and so on.
- A codeword can uniquely be represented by a polynomial. A codeword consists of a sequence of elements.
- We can use a polynomial to represent the locations and the values of all the element in the codeword.
- For example, the codeword $c_1c_2...c_n$ can be represented by the polynomial $c(x) = c_0 + c_1x + c_2x^2 + ... c_nx^n$.
- As another example, the codeword over $GF(8)$, $c = 207735$ can be represented by the polynomial $c(x) = 2 + 7x^2 + 7x^3 + 3x^4 + 5x^5$.
- Multiplying any polynomial by x corresponds to a single cyclic right-shift of the codeword elements. More explicitly, in Rn, by multiplying $c(x)$ by x we get
- $x.\ c(x) = c_0x + c_1\ x^2 + c_2x^3 + ... c_nx^{n+1}$
  $= c_n + c_0x + c_1x^2 + c_2x^3 + ... c_{n-1}x^n$

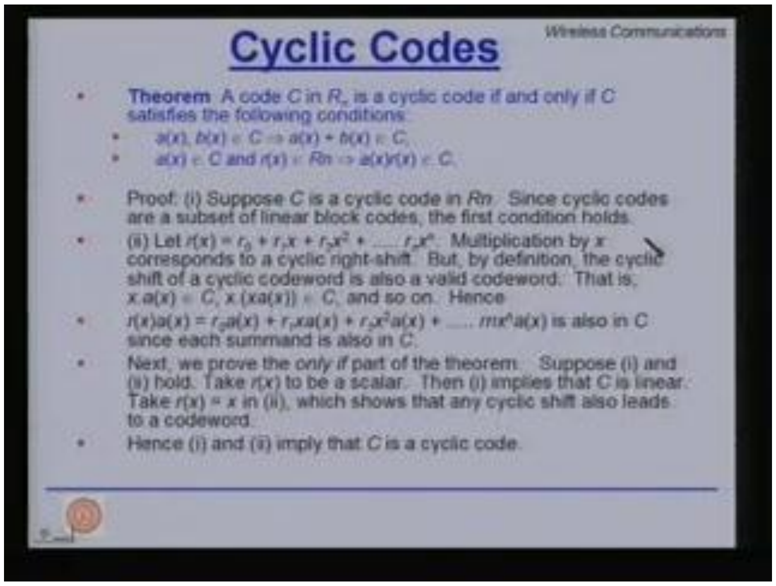
So now after this brief mathematical detour let’s go back to cyclic codes and see how this arithmetic with polynomials can help us construct cyclic codes. Now let’s fix for a change rf(x) the polynomial with which we divide as x^n-1. This is very interesting as we will soon see and we also denote if your small f(x) is x^n -1 then the set capital F[x] divided by f(x) by R_n. Let’s make the following observation; x^n is 1 provided you are taking it modulo x^n -1. So x^n is unity in R_n. So the modulo x^n -1 can be reduced simply by replacing x^n by 1 x^n +1 by x and so and so forth. A codeword can be uniquely represented by a polynomial; this is the most important observation on this slide.

A codeword consists of a sequence of elements and they can correspond one to one to the coefficients of the polynomial. So we can use a polynomial to represent the locations and the values of all the elements in the codeword. So if my codeword has a block length n then there are n elements in a codeword so we can use these elements of the codeword as coefficients of a polynomial. So polynomial has a unique codeword and a codeword can be uniquely represented by a polynomial. So for example the codeword c_1, c_2, up to c_n can be represented by the polynomial c_0 + c_1 x + c_2 x square up to c_n so there is a one to one correspondence. Codeword c_0 c_1 c_2 can be represented like this c_0 is missing.

Another example of the codeword over GF (8) let’s say 207735 then we have 27735 because the zero is missing because it is zero times x. You can uniquely represent any codeword as a polynomial. Now what is interesting is multiplying any polynomial by x raises the power of each of this x by 1 and hence it represents a single right cyclic shift. So shifting the elements, rotating them by one is very simple is just multiplying the polynomial representation of the codeword by x. If you multiply it again you shift it one more time and so and so forth. The only problem is that my block length of any codeword is fixed and is n.

If I keep on multiplying the power of x increases, if it is goes beyond x^n then I am actually elongating my codeword, I cannot. In fact I must have a way to put the maximum power back to here, I must do a modulo operation and this operation of putting the highest power which exceeds the block length back to the first place is done by modulo x^n-1. So as long as you keep on multiplying by x but finally take modulo x^n -1 you restrict the block length to n and at the same time you perform the proper cyclic shifts. So please note if you multiply c(x) by x, you raise the power by 1 but I need some way to put the c_n back before c_0 because this is clearly a representation of a codeword which is longer than block length n. We need to put this back here and this is done by modulo x^n -1hence the importance of modulo f(x).

(Refer Slide Time: 00:17:43 min)



Cyclic Codes

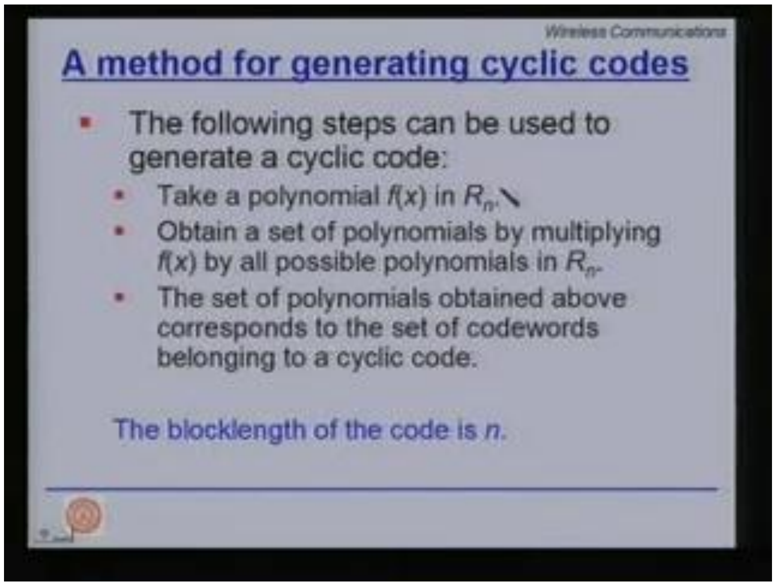
Wireless Communications

- **Theorem** A code C in R_n is a cyclic code if and only if C satisfies the following conditions:
  - $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$,
  - $a(x) \in C$ and $r(x) \in Rn \Rightarrow a(x)r(x) \in C$.
- Proof: (i) Suppose C is a cyclic code in Rn. Since cyclic codes are a subset of linear block codes, the first condition holds.
- (ii) Let $r(x) = r_0 + r_1x + r_2x^2 + \ldots\ldots r_nx^n$. Multiplication by x corresponds to a cyclic right-shift. But, by definition, the cyclic shift of a cyclic codeword is also a valid codeword. That is, $x.a(x) \in C$, $x.(xa(x)) \in C$, and so on. Hence
- $r(x)a(x) = r_0a(x) + r_1xa(x) + r_2x^2a(x) + \ldots\ldots rnx^na(x)$ is also in C since each summand is also in C.
- Next, we prove the *only if* part of the theorem. Suppose (i) and (ii) hold. Take $r(x)$ to be a scalar. Then (i) implies that C is linear. Take $r(x) = x$ in (ii), which shows that any cyclic shift also leads to a codeword.
- Hence (i) and (ii) imply that C is a cyclic code.

Now here is a theorem, a code C in R_n we know is when you take modulo x^n -1is a cyclic code if and only if C satisfies the following condition, a(x) and b(x) the elements in C then a(x) plus b(x) is an element in C, it's a linearity condition and a(x) element of C, r(x) element of R_n then a(x) r(x) is an element of C. So here is a brief proof for this one and the basic philosophy here is that if you have a cyclic code, it can be simply represented in R_n. So cyclic code will have polynomials contained in f(x) over small f(x) where small f(x) is x^n -1 and it will satisfy these following two properties; it's easy to prove these two here.

Now let us move over to a method for generating cyclic codes because all this mathematical tools are going to help us generate cyclic codes. First take a polynomial f(x) in R_n, obtain a set of polynomials simply by multiplying f(x) by all possible polynomials in R_n. So when you multiply f(x) by any other polynomial you do two things; you can multiply with something x raise to power something that will do a cyclic shift and then add it up which means that you are taking one codeword and adding with another valid codeword.

(Refer Slide Time: 00:18:54 min)



Wireless Communications

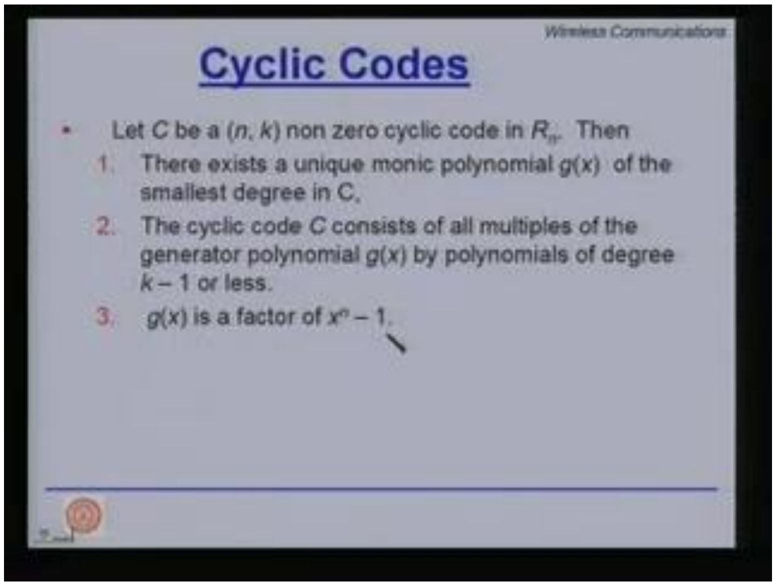
A method for generating cyclic codes

- The following steps can be used to generate a cyclic code:
  - Take a polynomial $f(x)$ in R_n.
  - Obtain a set of polynomials by multiplying $f(x)$ by all possible polynomials in R_n.
  - The set of polynomials obtained above corresponds to the set of codewords belonging to a cyclic code.

The blocklength of the code is n.

The set of polynomials obtained above correspond to the set of codewords belonging to a cyclic code. The block length of the code is n. Let us take a simple exercise, if f(x) is a polynomial in R_n then if you multiply this by x+1 which is a valid polynomial, it tantamount to first multiplying f(x) by x which is again a valid polynomial and then adding it to f(x) which is multiplying f(x) with one which also is a valid polynomial and the together the addition should also be a valid codeword in R_n. Hence we satisfy the definition of a cyclic code and this can be generalized for any complex polynomial within R_n.

(Refer Slide Time: 00:20:42 min)



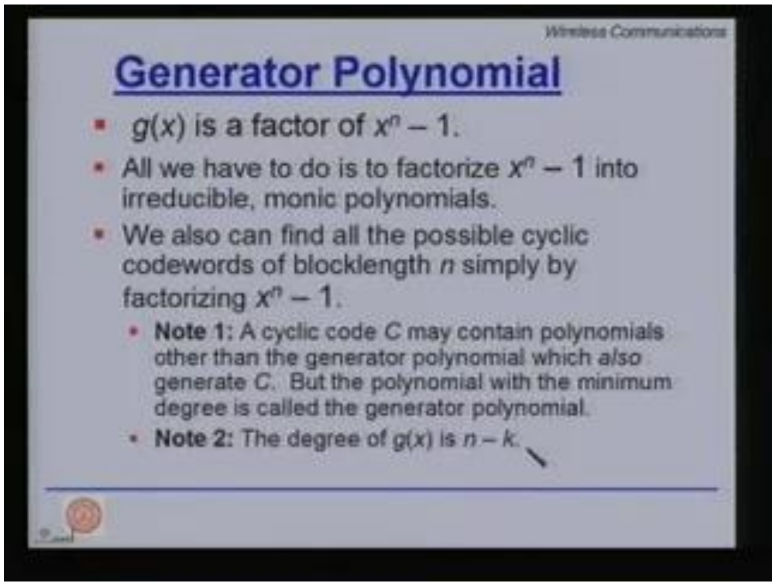
Wireless Communications

Cyclic Codes

- Let C be a (n, k) non zero cyclic code in R_n. Then
  1. There exists a unique monic polynomial $g(x)$ of the smallest degree in C.
  2. The cyclic code C consists of all multiples of the generator polynomial $g(x)$ by polynomials of degree $k - 1$ or less.
  3. $g(x)$ is a factor of $x^n - 1$.

So let C be a (n, k) non-zero cyclic code in R_n then there exists a unique monic polynomial g (x) of the smallest degree in C. This is the special polynomial, the cyclic code C consists of all multiples of the generator polynomial g(x) so we call this as the generator polynomial. It is the analog of the generator matrix, this has the capability to generate all the other valid cyclic code words. So the cyclic code C consists of all multiples of the generator polynomial g(x) by polynomials of degree k-1 or less and most importantly g(x) is a factor of x^n -1.

(Refer Slide Time: 00:21:38 min)



Wireless Communications

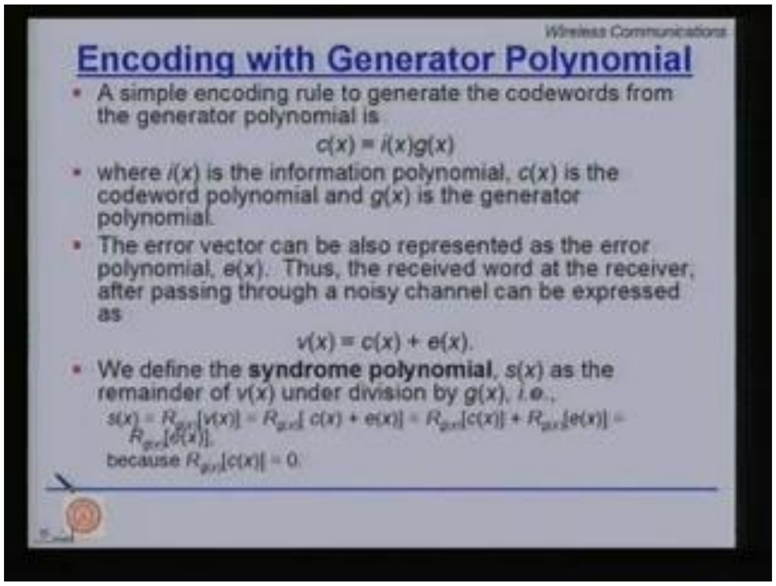
Generator Polynomial

- $g(x)$ is a factor of $x^n - 1$.
- All we have to do is to factorize $x^n - 1$ into irreducible, monic polynomials.
- We also can find all the possible cyclic codewords of blocklength n simply by factorizing $x^n - 1$.
  - **Note 1:** A cyclic code C may contain polynomials other than the generator polynomial which *also* generate C. But the polynomial with the minimum degree is called the generator polynomial.
  - **Note 2:** The degree of $g(x)$ is $n - k$.

This is the most important fact, if this is true then we have a very elegant and simple way to find out a generator polynomial simply by factorizing x^n -1. In fact this will be one of a preferred ways to find g(x) for cyclic codes, factorize x^n -1. All we have to do is to factorize this quantity into irreducible monic polynomials. These are the conditions required for g (x). We also can find all possible cyclic code words of block length n, simply by factorizing x^n -1. Note a cyclic code C may contain polynomials other than the generator polynomials which will also generate C. However the polynomial with the minimum degree is called the generator polynomial and the degree of g(x) is n-k.

So let us see how we can encode using generator polynomial, please note the generator polynomial itself is a valid codeword. If you multiplied with unity you get a polynomial g(x) which should be also valid codeword. A simple encoding rule to generate the code words from the generator polynomial is as follows; c(x) which is a codeword polynomial, we have established before a one to one correspondence between a polynomial and a word so c(x) be the codeword polynomial, i(x) be the information word and g(x) be the generator polynomial. So if you multiply i(x) with g(x) you would get a valid code word polynomial.

(Refer Slide Time: 00:22:46 min)



Wireless Communications

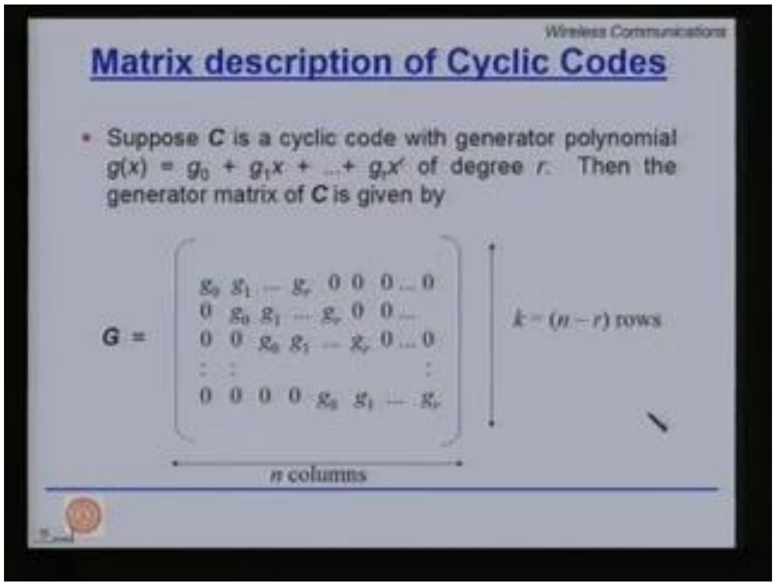
Encoding with Generator Polynomial

- A simple encoding rule to generate the codewords from the generator polynomial is
$$c(x) = i(x)g(x)$$
- where $i(x)$ is the information polynomial, $c(x)$ is the codeword polynomial and $g(x)$ is the generator polynomial.
- The error vector can be also represented as the error polynomial, $e(x)$. Thus, the received word at the receiver, after passing through a noisy channel can be expressed as
$$v(x) = c(x) + e(x).$$
- We define the **syndrome polynomial**, $s(x)$ as the remainder of $v(x)$ under division by $g(x)$, *i.e.*,
$s(x) = R_{g(x)}[v(x)] = R_{g(x)}[c(x) + e(x)] = R_{g(x)}[c(x)] + R_{g(x)}[e(x)] = R_{g(x)}[e(x)]$,
because $R_{g(x)}[c(x)] = 0$.

The error vector can also be represented as the error polynomial e (x). It should show at what location what is the magnitude of the error. Thus the received word at the receiver after passing through a noisy channel can simply be expressed as the received word v (x) is equal to c(x) the codeword polynomial plus the error polynomial e (x). Now let us maintain the analogy, so we had a generator matrix, generator polynomial we have a syndrome, we have a syndrome polynomial also. The syndrome polynomial s (x) is the remainder of v(x) under the division by g (x).

Clearly if g (x) completely divides c (x) that is the remainder is zero then you have a valid polynomial. Whatever be the residue that is the remainder after dividing by g (x) is the syndrome polynomial. Please note s (x) is the remainder if you divide by g (x), what v (x) which is nothing but remainder when you divide by g (x). What is v (x), nothing but c(x) plus e(x) and then by simple rule you can separate them out but c (x) when divided by g (x) should be zero by definition because g(x) times i(x) is c (x). So what you get is the remainder of e (x) when you divide it by g (x). Thankfully the syndrome depends only on the error polynomial as it should and has nothing to do with the code word polynomial. Just as a syndrome or a symptom of a disease tells about the disease, similarly the syndrome polynomial should tell us what kind of an error you have encountered. If you have a one to one mapping for every error pattern a unique syndrome then it is very easy to map back.

(Refer Slide Time: 00:25:53 min)



Wireless Communications

Matrix description of Cyclic Codes

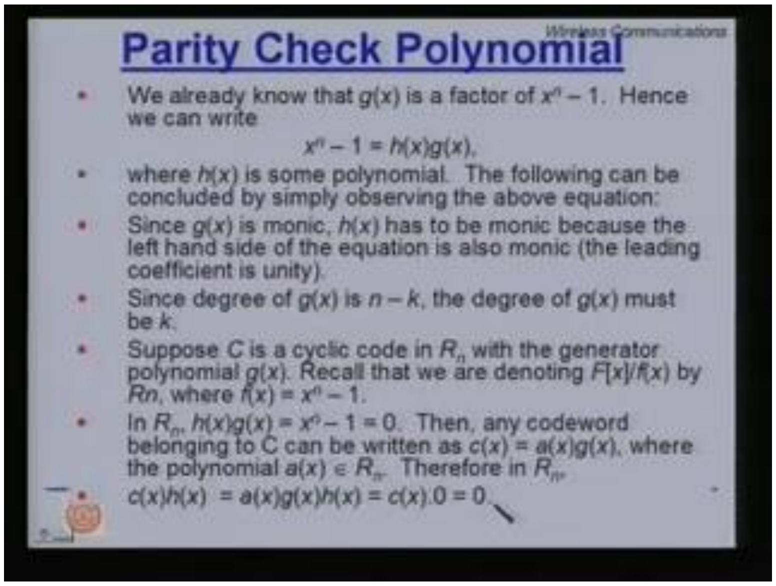
- Suppose **C** is a cyclic code with generator polynomial $g(x) = g_0 + g_1x + \ldots + g_rx^r$ of degree r. Then the generator matrix of **C** is given by

$$\mathbf{G} = \begin{pmatrix} g_0 & g_1 & \cdots & g_r & 0 & 0 & 0 \ldots 0 \\ 0 & g_0 & g_1 & \cdots & g_r & 0 & 0 \ldots \\ 0 & 0 & g_0 & g_1 & \cdots & g_r & 0 \ldots 0 \\ \vdots & \vdots & & & & & \vdots \\ 0 & 0 & 0 & 0 & g_0 & g_1 & \cdots\ g_r \end{pmatrix} \quad k = (n-r) \text{ rows}$$

n columns

It is also possible to have a matrix description of cyclic codes simply because it is a sub class of linear block code and must have a generator matrix but the generator matrices very easy to represent, once you have the generator polynomial and you have the generator polynomial which is very easy once you can factorize x^n -1, n is your block length. So if I give you a block length we can immediately take x^n -1 factorize it, get g (x) as a polynomial which can generate all the valid code words, once you have g (x) you can have generator matrix which is simply written as the shifted coefficients of g (x) in the various rows, is again a k by n matrix for (n, k) code. It is a one to one correspondence here if you have g (x) you can write the generator matrix.

(Refer Slide Time: 00:26:56 min)



Wireless Communications

Parity Check Polynomial

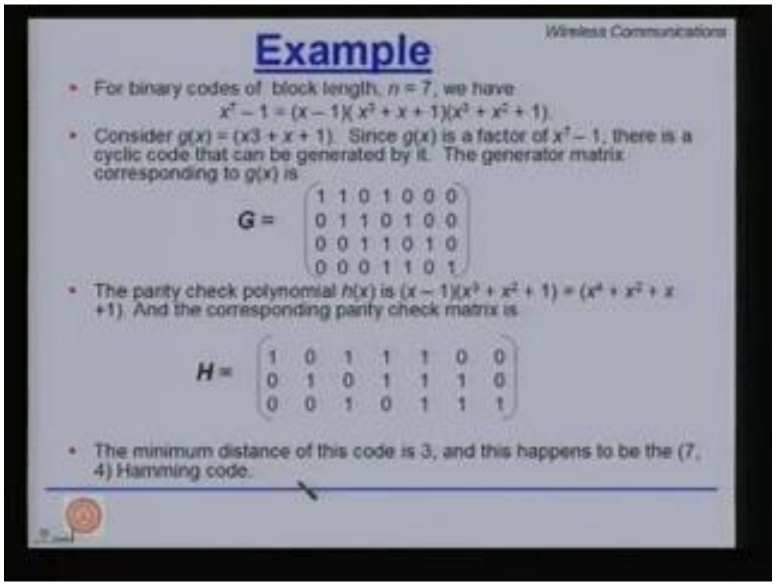
- We already know that $g(x)$ is a factor of $x^n - 1$. Hence we can write
$$x^n - 1 = h(x)g(x),$$
- where $h(x)$ is some polynomial. The following can be concluded by simply observing the above equation:
- Since $g(x)$ is monic, $h(x)$ has to be monic because the left hand side of the equation is also monic (the leading coefficient is unity).
- Since degree of $g(x)$ is $n - k$, the degree of $g(x)$ must be k.
- Suppose C is a cyclic code in R_n with the generator polynomial $g(x)$. Recall that we are denoting $F[x]/f(x)$ by Rn, where $f(x) = x^n - 1$.
- In R_n, $h(x)g(x) = x^n - 1 = 0$. Then, any codeword belonging to C can be written as $c(x) = a(x)g(x)$, where the polynomial $a(x) \in R_n$. Therefore in R_n,
- $c(x)h(x) = a(x)g(x)h(x) = c(x).0 = 0.$

Now just like we have a parity check matrix, we have a parity check polynomials for a valid cyclic code. What should it do? If you multiply it with the valid codeword you should get a zero and otherwise you should get a non-zero syndrome of sorts. So we already know that g (x) is a factor of x^n -1 hence you can write x^n -1 is equal to h (x) times g (x), so factor after all. Now h (x) is some polynomial, it's kind of a dual so the following can be concluded simply by observing the above equation. Since g(x) is monic, h(x) has to be monic because x^n is, the highest coefficient is one but please note here we are not talking only of binary cyclic codes, this is valid for all Galois fields.

So even if I am talking about cyclic codes over g (8), g (16) this will still hold good but g (x) must be monic. There is the highest coefficient should have coefficient equal to one, the highest power of x. So since g(x) is monic h(x) has to be monic because the left hand side of the equation is also monic. Since the degree of g (x) is n- k, the degree of g(x) must be the degree of h(x) must be k (Refer Slide Time: 28:47). Suppose C is a cyclic code in R_n with generator polynomial g (x) recall that we denoting F[x] by small f(x) by R_n where f(x) is x^n -1. Then any codeword belonging to C can be written as c(x) is equal to a(x) times g (x), we have noted this before where the polynomial e (x) is within R_n.

Therefore within R_n c(x) times h (x) is equal to a(x) g(x) h (x) but for every valid g(x) and h(x) it should be zero, so it is zero. That means if you multiply c(x) with h(x) you end up with a zero, just like the parity check matrix we multiplied with the codeword vector, you end up with a zero vector; if you multiply the parity check polynomial with a valid codeword polynomial you end up with the zero polynomial. Had there been an error you will get a non-zero polynomial and that will form your error polynomial.

(Refer Slide Time: 00:30:03 min)



Wireless Communications

Example

- For binary codes of block length, $n = 7$, we have
$$x^7 - 1 = (x-1)(x^3+x+1)(x^3+x^2+1).$$
- Consider $g(x) = (x3 + x + 1)$. Since $g(x)$ is a factor of $x^7 - 1$, there is a cyclic code that can be generated by it. The generator matrix corresponding to $g(x)$ is
$$G = \begin{pmatrix} 1&1&0&1&0&0&0 \\ 0&1&1&0&1&0&0 \\ 0&0&1&1&0&1&0 \\ 0&0&0&1&1&0&1 \end{pmatrix}$$
- The parity check polynomial $h(x)$ is $(x-1)(x^3+x^2+1) = (x^4+x^2+x+1)$. And the corresponding parity check matrix is
$$H = \begin{pmatrix} 1&0&1&1&1&0&0 \\ 0&1&0&1&1&1&0 \\ 0&0&1&0&1&1&1 \end{pmatrix}$$
- The minimum distance of this code is 3, and this happens to be the (7, 4) Hamming code.

So let's look at an example for binary codes of block length n =7, we have x^n -1is equal to x^7 -1 and the first step always in the search of cyclic codes is to factorize this. In a previous lecture we had looked at certain factorizing techniques.

So the first step is simple x^7 -1 is x -1 times x^6 +x^5 and so and so forth up to x^0 which can be further factorized into these two terms. So we done the factorization and all of them are irreducible, they cannot be factorized further which means the three of them and any product thereof can form a valid generator polynomial, it has to be a factor of x^7 -1. So just consider x^3 +x +1 since g(x) is a factor of x^7 -1 is a cyclic code that can be generated by it. Now having found out a generator polynomial which in our case here is x^3 + x+1 I can simply write the generator matrix by using the coefficients 1 1 0 1 are the first three coefficients in the decreasing order because the coefficients of x^2 is zero here which is reflected here. So x_0 x_1 x_2 and x_3 and then a cyclic shifted (Not understandable) (Refer Slide Time: 00:31:57 min) another one and another one.

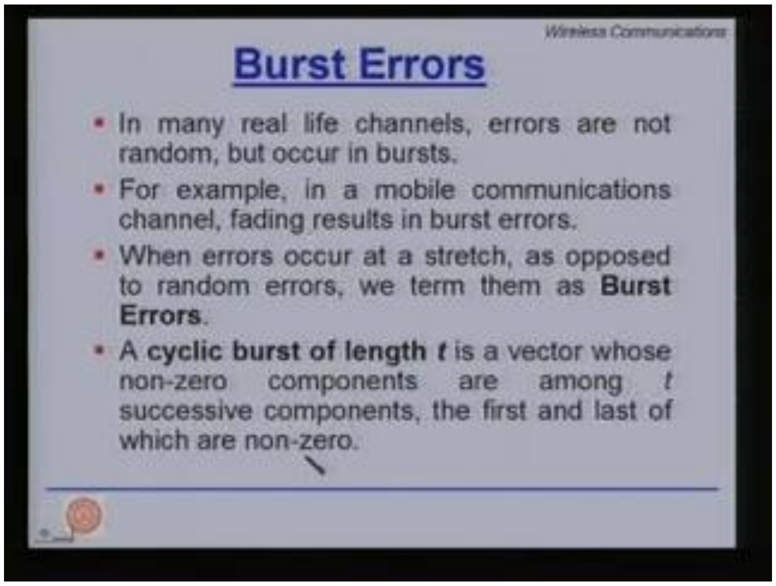
The number of rows is 4, the number of columns is 7, n =7 and k =4 and n- k is the highest power of x_0. If you know G you can also find out the parity check matrix so this is nothing but whatever is remaining. So if you multiply x -1 with x^3 + x^2 +1 you get this one and this corresponds to the coefficients of the parity check matrix. The number of rows is n – k is equal to 3, the number of columns is equal to n =7. So this is the G and the H matrix for your cyclic code. Please note here we have a recipe to make as many generator polynomials as possible. In fact 1, 2, 3 are clearly visible. Then product of these two 4, product of these two 5, product of these two 6 and then there are two more factors for x^7 -1, one is unity it's also a factor and itself x^7 -1 that's also a factor of x^7 -1.

So you have 8 possible generator polynomials out of x^7 -1. [Conversation between Student and Professor – Not audible ((00:33:30 min))] Question is being asked about h(x), h(x) as we know has a very interesting property. We have denoted h(x) times g(x) is equal to x^n -1. So if I choose my generator polynomial for an instance as x^3 + x+1 then whatever other two terms left their product is h (x). Had I chosen my generator polynomial is x^3 +x^2 +1 it's also valid generator polynomial because it's a fact is a monic, is a factor of x^7 -1. In that case the product of x -1 and x^3 + x+1 the first two terms would have been h (x), so g (x) times h (x) is x^n -1. So after the effort that we put in, to figure out factorization of polynomial it is now a piece of cake to figure out how to get to the various kinds of generator polynomials.

In fact once you factorize, you found out all possible generator polynomials for the block length 7 because I factorize x^7 -1 once. In fact I have not one, not two but the existence of 8 possible cyclic codes here, of course some of them are trivial. So the minimum distance of this code is 3 and if you observe it carefully this happens to be the (7, 4) hamming code. So the hamming code which is also perfect code also happens to be a cyclic code, a very versatile code. Now let us give a flavour to this error control coding from the perspective of mobile communications. In mobile communications we have already encountered that the wireless channel is a very hostile channel, it goes through deep fades and whenever we pass through a fade, we have a burst of errors.

It depends upon the fade duration, if the fade duration is large then we have several bits wiped out at a time and these are consecutive bits so far the philosophy of error control coding has been that we can handle random errors but if the errors start happening in bursts that is all the errors are situated next to each other then we are in problem.

(Refer Slide Time: 00:36:35 min)



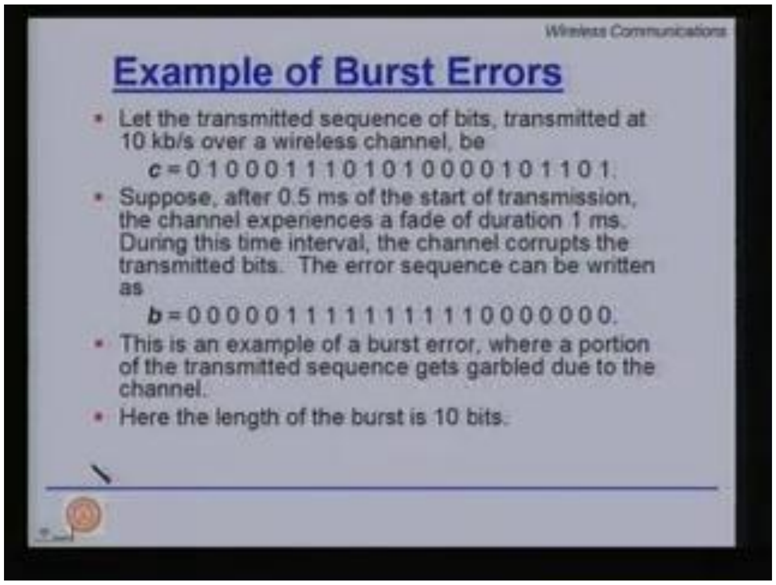
Wireless Communications

Burst Errors

- In many real life channels, errors are not random, but occur in bursts.
- For example, in a mobile communications channel, fading results in burst errors.
- When errors occur at a stretch, as opposed to random errors, we term them as **Burst Errors**.
- A **cyclic burst of length *t*** is a vector whose non-zero components are among *t* successive components, the first and last of which are non-zero.

Then we must device some smart techniques to overcome the burst errors as we will see cyclic codes are phenomenally good in overcoming burst errors. In many real life channels as we know in wireless channels errors are not random but occur in bursts. For example the mobile radio channel, when errors occur at a stretch as opposed to random errors we call them burst error. A cyclic burst of length t is a vector whose nonzero components are among the t successive components but note the first and the last of which are nonzero. So if you say it's a burst of length t then the first and the last elements in the error pattern should not be zero. Let's take a small example that the transmitted sequence of bits transmitted at 10 kilobits per second over a wireless channel be follow.

(Refer Slide Time: 00:37:15 min)



Wireless Communications

Example of Burst Errors

- Let the transmitted sequence of bits, transmitted at 10 kb/s over a wireless channel, be

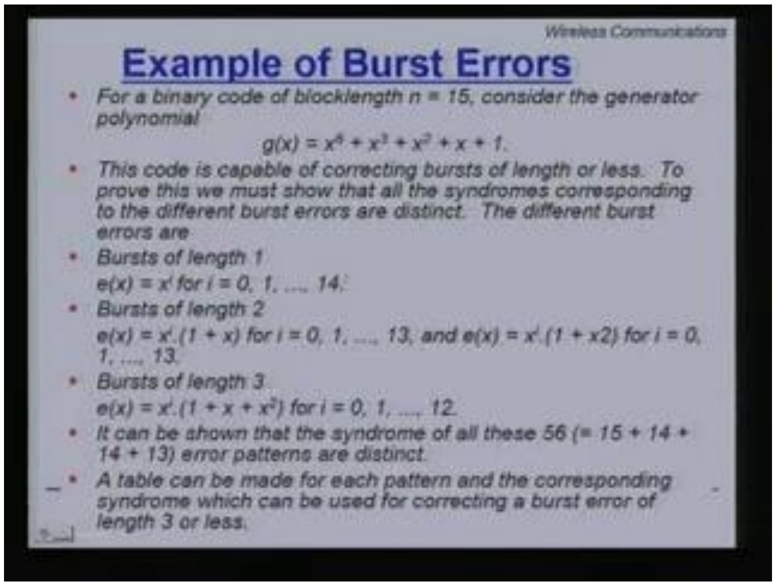
  $\boldsymbol{c} = 0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1.$
- Suppose, after 0.5 ms of the start of transmission, the channel experiences a fade of duration 1 ms. During this time interval, the channel corrupts the transmitted bits. The error sequence can be written as

  $\boldsymbol{b} = 0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0.$
- This is an example of a burst error, where a portion of the transmitted sequence gets garbled due to the channel.
- Here the length of the burst is 10 bits.

Suppose after 0.5 millisecond of the start of transmission, the channel experiences a fade of duration one millisecond. Now we have already devised mathematical tools to predict the average duration of fade. Let this be a measured quantity one millisecond. So during this time interval the channel corrupts all the transmitted bits so here is a start of fade and the fade lasts for so long and then the fade ends because we know that the data rate and we also know how long the fade last here this is kind of an error patterns. So all the bits within this fade have been flipped but clearly we do not know the fact that all of them have been flipped. If you knew it will just flip it back and a problem is solved. So fade doesn't mean that the all the bits are exactly flipped.

In fact we do not know whether the bit is flipped or not but there is a possibility have been flipped but most importantly for the burst of length t, the first and the last should not be zero, rest all can be zeros. It could have been 1 0 0 0 0 0 0 up to here and then one again it will still qualify as a burst of length t. This is an example of a burst error where a portion of the transmitted sequence gets garbled due to the channel; here the length of the burst is 10 bits.

(Refer Slide Time: 00:39:07 min)



Wireless Communications

Example of Burst Errors

- *For a binary code of blocklength n = 15, consider the generator polynomial*

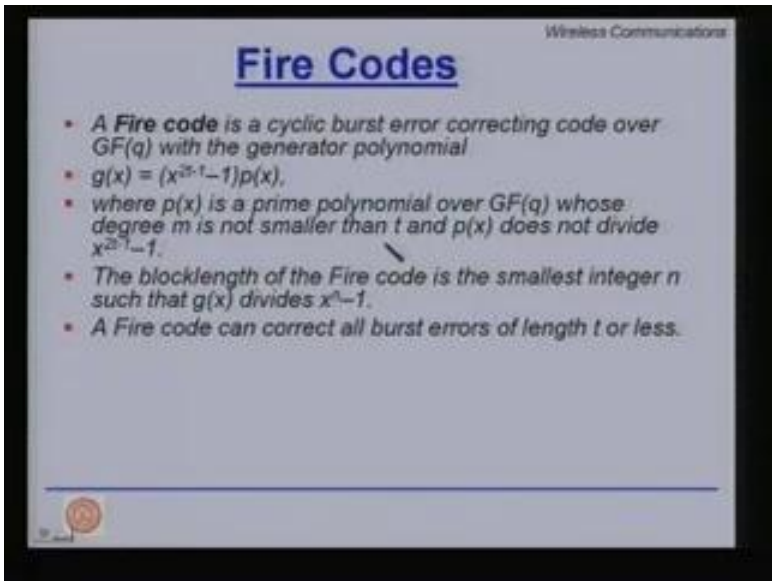
$$g(x) = x^6 + x^3 + x^2 + x + 1.$$

- *This code is capable of correcting bursts of length or less. To prove this we must show that all the syndromes corresponding to the different burst errors are distinct. The different burst errors are*
- *Bursts of length 1*
  $e(x) = x^i$ *for* $i = 0, 1, \ldots, 14.$
- *Bursts of length 2*
  $e(x) = x^i.(1 + x)$ *for* $i = 0, 1, \ldots, 13$, *and* $e(x) = x^i.(1 + x2)$ *for* $i = 0, 1, \ldots, 13.$
- *Bursts of length 3*
  $e(x) = x^i.(1 + x + x^2)$ *for* $i = 0, 1, \ldots, 12.$
- *It can be shown that the syndrome of all these 56 (= 15 + 14 + 14 + 13) error patterns are distinct.*
- *A table can be made for each pattern and the corresponding syndrome which can be used for correcting a burst error of length 3 or less.*

So let's look at an example for a binary code of block length n =15, consider the generator polynomial like this. So generator polynomial gives you a lot of information you only know n=15 but you know that the highest power 6 corresponds to n-k. So n-k 6 so you can find out k from here and then you can find out the code rate. This code is capable of correcting burst of length 3 or less, to prove this we must show that all syndromes corresponding to the different burst errors are distinct. The different bursts are burst of length one, it's just saying that one bit occurs at any one of the 15 locations. What are the 15 locations? 0 through 14 so e(x) which is our polynomial for burst error is x^i, bursts of length 2 can be denoted by e(x) is equal to x^i (1+ x) because they have to be two consecutive bits which are in error it's a bursts of length 2.

Bursts of length 3 can be denoted as follows, there have to be 3 consecutive errors, it can be shown that the syndrome of all this 56 possibilities. So they have only 56 possible bursts lengths of burst length 1, 2 or 3. So if I say this code can correct all burst errors of up to length 3 then all the possible error patterns, all the possible burst error patterns must have a unique syndrome polynomial. How do we find out the syndrome polynomial? Take this e (x) divided by g (x), whatever is the remainder is a syndrome polynomial. So it can be shown it for all these cases, if you take it modulo g (x) you come up with unique polynomials. If we have unique polynomials you can map it back and hence correct all errors of burst length 3 or less. A table can be made for each pattern and the corresponding syndrome which can be used for correcting a burst error of length 3 or less can be found out.

(Refer Slide Time: 00:41:54 min)

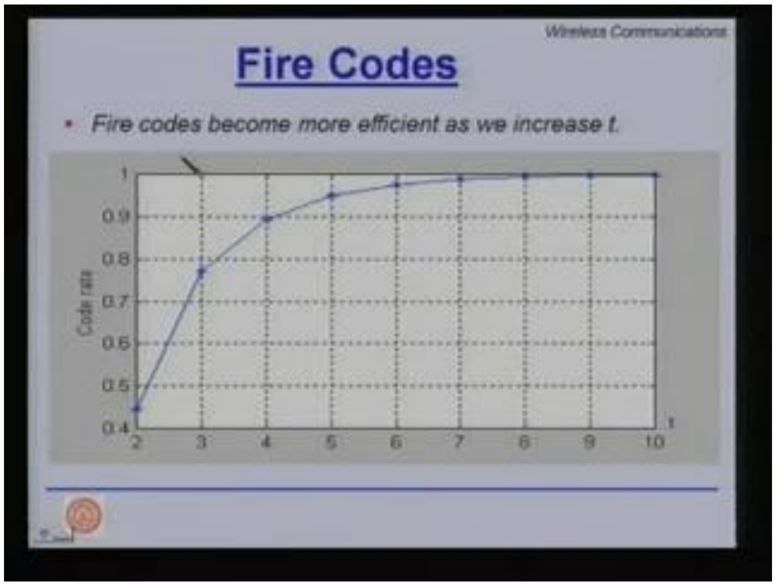

Wireless Communications

Fire Codes

- A **Fire code** is a cyclic burst error correcting code over GF(q) with the generator polynomial
- $g(x) = (x^{2t-1}-1)p(x)$.
- where p(x) is a prime polynomial over GF(q) whose degree m is not smaller than t and p(x) does not divide $x^{2t-1}-1$.
- The blocklength of the Fire code is the smallest integer n such that g(x) divides x^n-1.
- A Fire code can correct all burst errors of length t or less.

Let us look at another example, a code which is good at correcting burst errors it is called the fire codes. A fire code is a cyclic burst error correcting code over GF(q) with the generator polynomial given as g(x) is equal to x^{2t-1} -1 times p (x) where p(x) is a prime polynomial over GF(q) whose degree m is not smaller than t and p(x) does not divide x^{2t-1} -1. So under this constraint whatever code you form is called a fire code. It has some very interesting properties. The block length of the fire code is the smallest integer n such that g(x) divides x^n -1. A fire code can correct all burst errors of length t or less so within the definition of your generator polynomial I have put in how many burst errors can I correct. So I have designed rule here I fit in a value of t, if it is 3 you have x^5 -1times p (x) where p (x) must satisfy this property, p (x) does not divide x^5 -1.

(Refer Slide Time: 00:43:25 min)

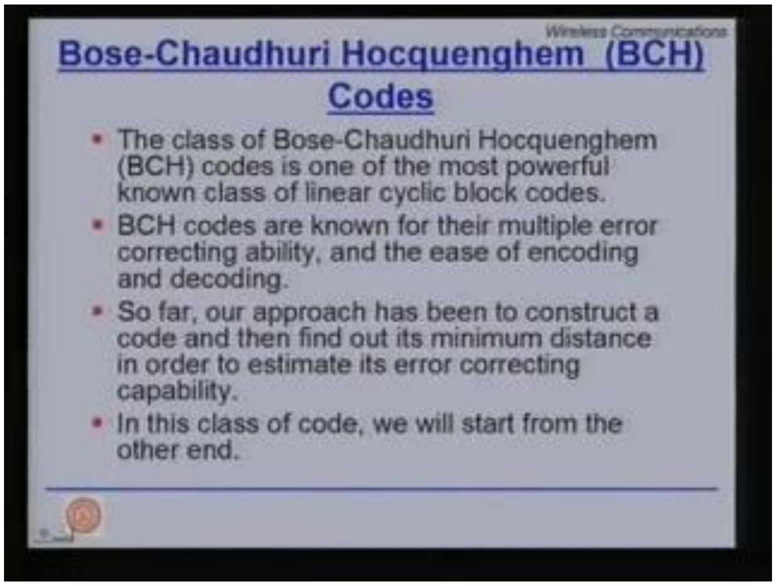


Fire Codes

- Fire codes become more efficient as we increase t.

What is interesting is fire codes become more efficient as we increase t. What is it mean? Efficiency in terms of the code rate. So if you increase this t, you have on the y axis the code rate and how do you get the code rate from here, n - k is the highest power of this whole thing. So as you increase your t on the x axis, the code rate tends to go up but your block length also tends to go up, so fire codes of large block lengths are very useful.

(Refer Slide Time: 00:44:13 min)


Wireless Communications

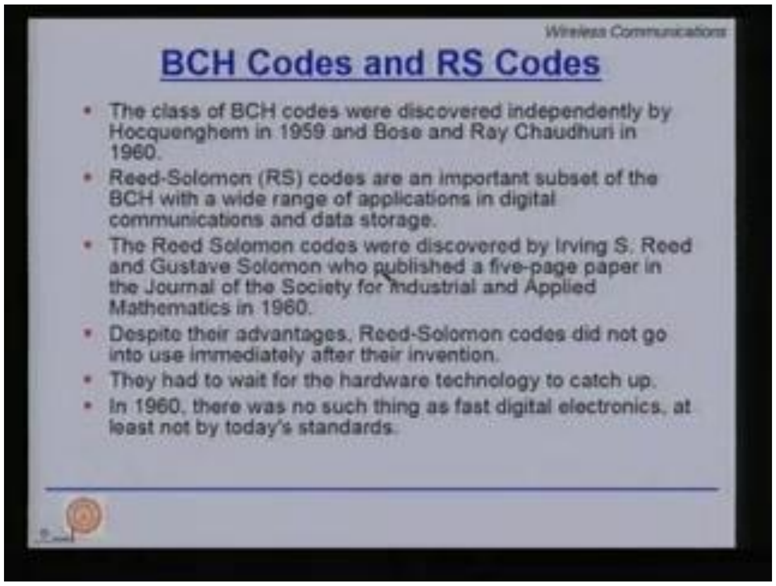
Bose-Chaudhuri Hocquenghem (BCH) Codes

- The class of Bose-Chaudhuri Hocquenghem (BCH) codes is one of the most powerful known class of linear cyclic block codes.
- BCH codes are known for their multiple error correcting ability, and the ease of encoding and decoding.
- So far, our approach has been to construct a code and then find out its minimum distance in order to estimate its error correcting capability.
- In this class of code, we will start from the other end.

Let us now speak briefly into a sub class of cyclic codes which is the Bose Chaudhuri Hocquenghem codes or more popularly known as the BCH codes.

So the BCH codes are a sub class of cyclic codes which are a sub class of linear block codes. The class of BCH codes is one of the most powerful known class of linear cyclic block codes. The BCH codes are known for their multiple error correcting capabilities and the ease of decoding and encoding. Now there is also paradigm shift, so far our approach has been to construct a code and then find out its minimum distance. The BCH code tells you please tell me how much is the desired minimum distance of the code and it will design a code accordingly. So it's an other way round so far we make a code we construct a code and then find the minimum distance and we talk about how many errors it can correct. If you go the other way round it tells you or it ask for the parameter how many errors is required to be corrected by the code and then constructs a code accordingly. In this case we start at the other end.

(Refer Slide Time: 00:45:47 min)



Wireless Communications

BCH Codes and RS Codes

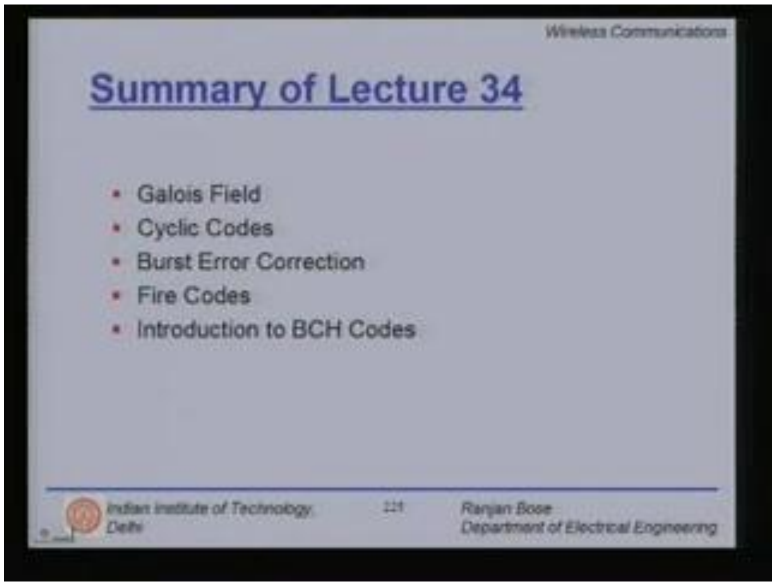
- The class of BCH codes were discovered independently by Hocquenghem in 1959 and Bose and Ray Chaudhuri in 1960.
- Reed-Solomon (RS) codes are an important subset of the BCH with a wide range of applications in digital communications and data storage.
- The Reed Solomon codes were discovered by Irving S. Reed and Gustave Solomon who published a five-page paper in the Journal of the Society for Industrial and Applied Mathematics in 1960.
- Despite their advantages, Reed-Solomon codes did not go into use immediately after their invention.
- They had to wait for the hardware technology to catch up.
- In 1960, there was no such thing as fast digital electronics, at least not by today's standards.

Now the class of BCH codes were discovered independently by Hocquenghem in 59 and Bose Ray Chaudhuri in 60 and hence the name BCH. The Reed Solomon codes are a sub class of BCH codes but an important sub class with a wide range of applications in digital signal processing is to communications and data storage. The Reed Solomon codes were discovered by Reed and Solomon who published a five page paper in the journal of society for industrial and applied mathematics in 60.

Despite their advantages RS codes, the Reed Solomon codes did not go into use immediately after their invention. Why? They had to wait for the hardware technology to catch by that time. So today the very popular RS codes which we find in our CD, the audio CDs, the video CDs even for deep space exploration and lot of wireless communication devices could not be put into immediate use because the required technology was not there. So in 60's there was no such thing as fast digital electronics not by today's standard.

So we will summarize today's talk here. We started off by looking at the Galois field followed by a detailed understanding of cyclic codes. We came up with an interesting polynomial representation of cyclic codes.

(Refer Slide Time: 00:47:05 min)



Wireless Communications

Summary of Lecture 34

- Galois Field
- Cyclic Codes
- Burst Error Correction
- Fire Codes
- Introduction to BCH Codes

Indian Institute of Technology Delhi 225 Ranjan Bose Department of Electrical Engineering

We realize that there is a one to one correspondence between polynomials and cyclic codes, what was interesting is that if you multiply a polynomial by x you right shift the codeword elements and what is important is for cyclic codes any shift, any cyclic shift of the codewords is also a valid codeword. Then we realize that there is a notion of a generator polynomial in cyclic codes and then the analogy doesn't stop there, you have a syndrome polynomial and is a parity check polynomial.

We also realize the importance that you have to take modulo x^n -1 so that when you multiply g(x) by any valid polynomial you do not over the block length. Then we also realize that g(x) the generator polynomial must be a factor of x^n -1 thereby we figured out a way to come up with all possible valid cyclic codewords of a certain block length n. All we have to do is take x^n -1 and factorize it and in front of our eyes will have all the possible valid generator polynomials. We then moved over to the domain of burst error correction. We realize that in mobile environments where we encounter deep fades, we tend to lose not random bits but also bits which are in burst error so we lose a sequence of bits. Our regular error control techniques are incapable of overcoming burst errors. We have to have special kind of codes for that. We looked at certain cyclic codes which are known to be good burst error correcting codes, one of the most important one is the fire codes.

We then briefly had a peak at the BCH codes which form an important sub class of cyclic codes not only because you can very easily encode and decode them but also that you can have a constructive strategy for BCH codes. At this moment in time please note that cyclic codes BCH codes and Reed Solomon code which is a sub class of BCH codes though non-binary, all form are from the linear block codes they all subsets of the linear block codes. So all the techniques that we have learned and developed for linear block codes are also valid for cyclic codes, your BCH codes and Reed Solomon codes. However in the subsequent lectures we will look at efficient encoding, decoding strategies for Reed Solomon and BCH codes as well. So this is a nice place to conclude our lecture and we will look at BCH codes starting from the next class.