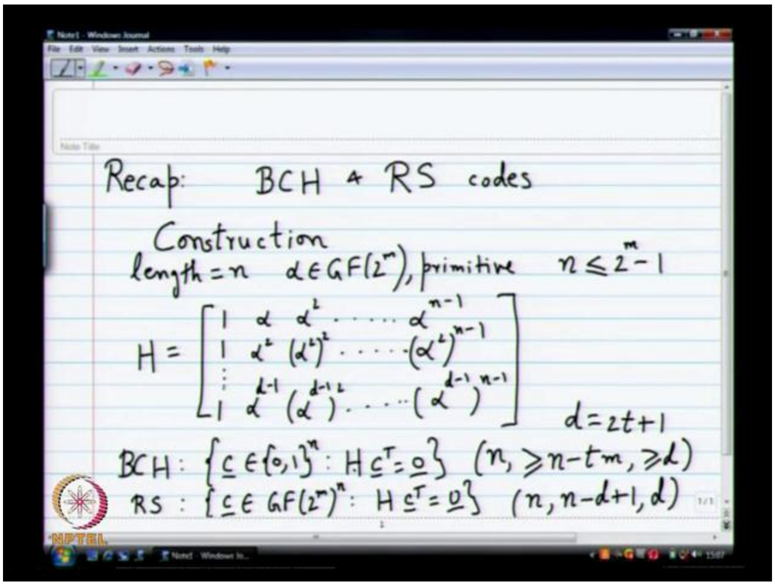
# Coding Theory
## Prof. Dr. Andrew Thangaraj
## Department of Electronics and Communication Engineering
## Indian Institute of Technology, Madras

## Lecture - 15
## Decoding BCH Codes

(Refer Slide Time: 00:14)



Recap: BCH + RS codes

Construction

length $= n$ $\quad \alpha \in GF(2^m)$, primitive $\quad n \leq 2^m - 1$

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ \vdots & & & & \\ 1 & \alpha^{d-1} & (\alpha^{d-1})^2 & \cdots & (\alpha^{d-1})^{n-1} \end{bmatrix} \qquad d = 2t+1$$

BCH : $\{\underline{c} \in \{0,1\}^n : H\underline{c}^T = \underline{0}\}$ $\quad (n, \geq n - tm, \geq d)$

RS : $\{\underline{c} \in GF(2^m)^n : H\underline{c}^T = \underline{0}\}$ $\quad (n, n-d+1, d)$

So, let us being with the quick recap, we are looking at BCH and Reed Solomon codes, so primarily we saw construction and properties. That is what we saw what is the construction is through this parity check matrix for the construction of length and if you want block length to be n you need you could take a primitive elements from G F primitive m and what is the property n should be smaller than or equal to 2 power m minus 1.

So, this is one choice, there are other choice we will just stick to this choice for simplicity k and then you construct parity check matrix has one alpha square. So, until alpha power n minus 1 alpha square alpha square, so until alpha squared power n minus 1 go all the way down to 1 alpha power d minus 1 alpha power d minus 1 square all the way to alpha power d minus 1 whole raise to the power n minus 1. This is the parity check matrix all the binary vectors which satisfy H c transpose equal 0 will give you the BCH code.

All the vectors over G F 2, the m will satisfy the same equation will give you it is Reed Solomon code. So, that is the idea, so for BCH set of all binary vectors k and Reed Solomon, if I write below, this c in k this is our definition. This is the construction and what are the properties of the codes, so for RS, it is very easy to come up with the parameters n what is k n minus d plus 1. Then minimum distance is exactly this, so those were the parameters for the Reed Solomon code. So, the BCH code it is a little bit more difficult, I mean to be very exact mostly you can you can come up with some numbers, but for instance if you take d to be d to t plus 1.

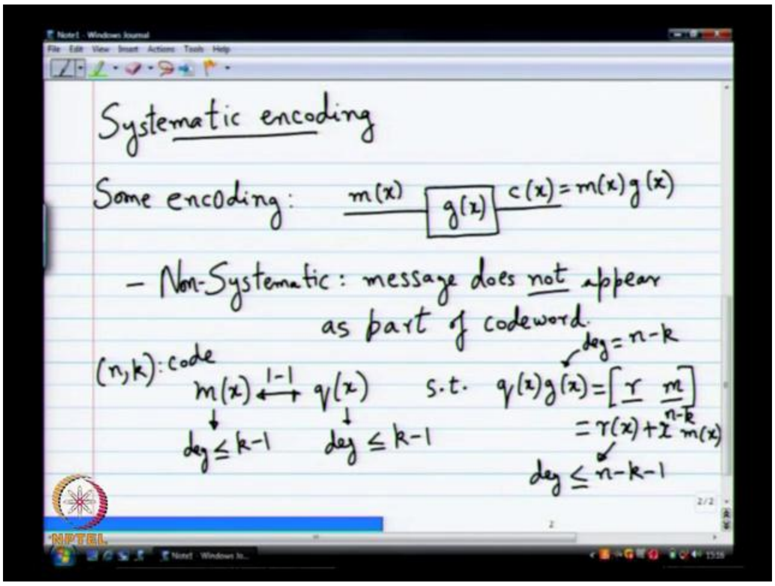
So, you have alpha power 1 all the way to alpha power 2 t k t is like a error correcting capability. Then, you know this is going to be n comma what n minus, so should be careful here just say greater than or equal to n minus t m can I say that think about it. So, see when I have d equals 2 t plus 1, my zeros are my roots are alpha square all the way to alpha power 2 t k. So, utmost t of m will give me a minimal polynomial and maximum degree for each of them is m.

So, the maximum possible degree for generate a polynomial is t m, so I know the degree to generate a polynomial is less than or equal o t m which means k is greater than i equal to n minus 2. So, in most this might be satisfied with equality, you might get an equality here in some cases in may restrict any quality. For instance, the case when we had n equal 15 and d equal 7 or 9, I do note that I forget what is one of the cases where the worsen any quality, k given the equal 5.

This this will not be satisfied you get something else, you go back and check that, but h this is the roughing equality which will work in many cases. So, this are the changes, first of all BCH is a binary code, this is binary and Reed Solomon is 2 to the m. So, hopefully this is construction and properties and I was pointing out how this is non trivial if you if you start with just a space of all binary vectors is very difficult to come up with this coeds. This is not so easy with huge space general you are able to do it, so that is the interesting point, so this is construction and parameters.

So, the next thing we have address of course is the decoding, so the decoding u is the most non trivial task in this process, but before that I want to address encoding little bit more in f careful detail. I did do that briefly here, there I think it was s, and you should do to for encoding, but I want to emphasis something in particular. I want to emphasis systematic encoding what first should you do for systematic encoding, we have not seen that.

(Refer Slide Time: 06:45)



Systematic encoding

Some encoding: $\xrightarrow{m(x)} \boxed{g(x)} \xrightarrow{} c(x) = m(x)g(x)$

– Non-Systematic: message does not appear as part of codeword.

(n,k): code

$m(x) \overset{1-1}{\longleftrightarrow} q(x)$ s.t. $q(x)g(x) = [\,\underline{r}\ \ \underline{m}\,]$

$m(x)$: deg ≤ k-1; $q(x)$: deg ≤ k-1; $g(x)$: deg = n-k

$= r(x) + x^{n-k} m(x)$

$r(x)$: deg ≤ n-k-1

So, let us see it, so let us first talk what is systematic encoding, so if you want to do some encoding, there is one possible encoding which is very easy to do. If you just if you just interested in some way of encoding this things what is the most obvious way of encoding this BCH and Reed Solomon codes what I been doing so far is simply come up with the message polynomial m of x. Then, multiply with the generate a polynomial G of x that has been like the simple way in which I been describing things.

So, for some reason, all the things are appearing here which I do not know, let us see hopefully we would not come up again. So, since you think that some mouse that is showing up to I have no idea why may be I am clicking this thing. This is in very bad location, so one way of doing encoding is to simply say you have message polynomial m of x multiplies with the generator polynomial g of x, you get a code word c of x equals m of x into G x.

So, this is this is very nice and it is immensely implementable, there is no problem there k the complexity is not very large. After all, it is multiplication by a polynomial is not is not so difficult, multiplication with polynomial remember is like conclusion. So, it is very easy to implement using some simple example can be very easily done, not problem, but one problem is with this encoding it is not systematic. What do you mean by systematic encoding what is systematic encoding k if it is systematic, so this is general non systematic, so assume message does appear has a part of the code word.

So, hopefully that is clear to you if you do m of x times 2 of x, there is no guarantee that message will appear by itself; any where it is just multiplication. It can go all over the place, so it will be good t p have a systematic encoder why would you like a systematic encoder, so decoder if only decode to an estimate of the code word, you correct the code word and then from the code word if you have systematic coding.

You can simply read of the message if you did not have a systemic encoder, then you will be worry about how to go back from the code word the message that is one problem. The other problem that also shows that often is the message usually will have a several distribution which you can control, so let us say that uniform distribution on k is something that you can control. If you do n times g, it does not same like some obvious control over the distribution of the message or some property of the bits into the message you want to maintain k at least for most of bits.

So, m times g in other hand if you have a systematic encoder the distribution of the code word also is controlled to a large extend k some constraints are satisfied to large extends, so these things are useful. So, systematic coding is useful for variety of proposes, so it is good to see if you have systematic encoder. So, I am going to describe now is another way of encoding sickly codes, so in this other way of encoding m of x will not map in m of x. This g of x will map to some other q of x into G of x, but it will still be a one to one map, so that way you way you come up with this.

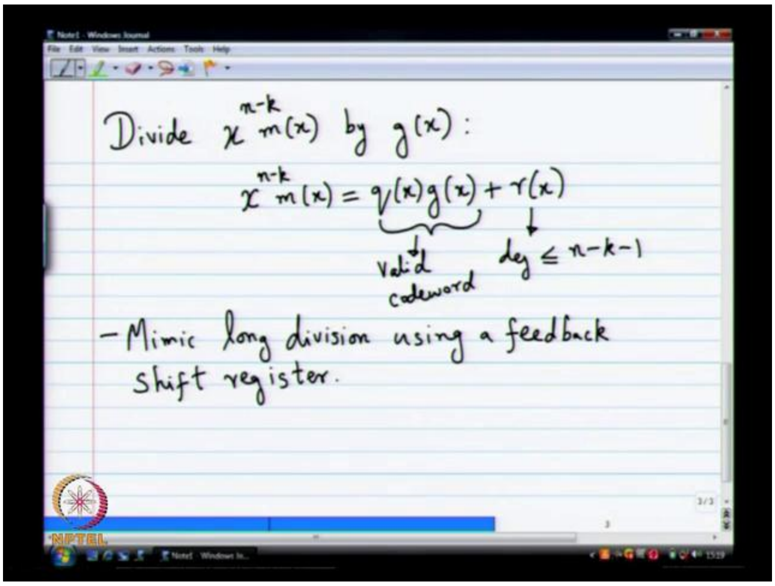
So, you go from m of x to some other q of x k, so remember this is what is m of x, so if you have m at key code, this is the degree less than or equal to k minus 1 polynomial q of x is also let say degree less than or equal to k minus 1. So, this is the way of one to one mapping, I will find a one to one mapping such that such that what q of x times G of x this will be a code word from the word. This will be of the form some r and then m k, so I am going to back and search between polynomial notation.

The vector notation, hopefully it is clear, I have written ugly looking expression here, but hopefully it is to you k q of x and G of x is some of c of x is a code word of the code, but it has the form r m k. So, in particular the last k bits of the code word of c of x equal m of x k, so that is the idea is that basically if you want in terms of polynomial. This will be r of x plus x to the power n minus k times m of x and the degree of r of x is less than or equal o n minus k minus 1 it, so I have to find the q of x which will give me this satisfies this equation.

So, why did I do x power n minus k times m of x that is what shift my m of x to that position, so I push it that position t and then add some r of x r of x should have some degree definitely less than or equal to n minus k minus 1. Only then, it will not disturb anything in my message, it will appear as it is, so r will come here at that equation for a while and tell me how I can find such a q of x and r of x the location, I have used is bit of again k find a q of x and r of x.

So, I have to divide something by something, dividing we had by what I want to find q of x and r of x q of x is going to be some coefficient r of x is going to be some remainder. So, what r should I divide by what x power n minus k times m of x have to divide by g of x, remember g of x has degree equal to equal to what n minus k. So, clearly when I divide the remainder will have degree less than or equal to n minus k minus some and I satisfy very single thing, I want k, so that is the formula for systematic encoding.

(Refer Slide Time: 14:25)



Divide $x^{n-k} m(x)$ by $g(x)$:

$$x^{n-k} m(x) = \underbrace{q(x)g(x)}_{\substack{\downarrow \\ \text{Valid codeword}}} + \underset{\substack{\downarrow \\ \text{deg} \le n-k-1}}{r(x)}$$

- Mimic long division using a feedback shift register.

I should divide x to power n minus k and m of x by g of x what will I get, I will take the coefficient q of x. The remainder r of x the remainder r of x will have degree strictly less than or equal to no strictly less than n minus k or less than or equal to m minus k minus 1.

It will satisfy this equation, so this is a valid code word and it has the form, it we can a just that minus anywhere you like. So, find the remainder and attach the minus sign and put to this, so arbitrary finite field the question was r of x is form this side. Now, when I divided, I pushed r of x to other side, so when it comes to this side it is going to minus, but like I said we are we are going to only dealing with characteristics.

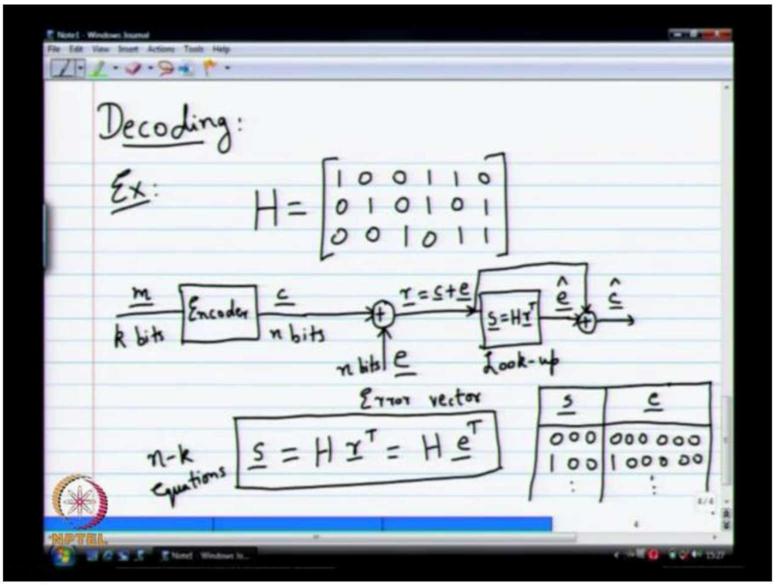
So, minus and plus are same there is no problem, but any way if you have characteristic that is not two, you add a minus sign here and then you then get this. In the remainder simply do a minus of x you will get the answer, it is very constructive method of course it works only for it things my hand. So, that is the problem with all this things, I think the more modern what are these things called you know that just we have they can differentiate your hand and the styles, so any way that is systematic encoding.

Hopefully, it is clear we have not given any examples, you get the general idea, so now suddenly look like you are dividing one polynomial by another polynomial, but this is division with remainder. So, you can basically limit your long division method remember a long division method it is very easy to mimic that and you can come with the simple shift resistance circuit which will implement division. So, you can mimic long division using a feedback shift register, it is a very common circuit that can be done quite easily with this.

So, then you do the division also quite easily so it is a easily do able operation through that and you get the answer alright that is sounds encoding more or less complexly k now we are able to do encoding with arbitrary error correcting capability any error correcting capability I want you can go to a large enough block length and chose your suitable valve of a.

Then, find suitable generate polynomial, then do this divisions and you get systematic encode, so both BCH and Reed Solomon you can do it either one can do depending on what you like. So, encoding part of the problem is solved, we have potentially we have good solution only if you can decode up to promised error correcting capability. If you can correct few errors d minus 1 by 2 error, then you are good to go, that is in the decoding part.

(Refer Slide Time: 18:26)



Decoding:

Ex:

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

n-k Equations $\boxed{s = H r^T = H e^T}$

| s | e |
|---|---|
| 000 | 000 000 |
| 100 | 100 000 |
| ⋮ | ⋮ |

So, I want quickly remind what we are doing seeing so far when comes of decoding the only we saw is this syndrome decoding which is good decoding. It is not like that decode, so let me briefly touch a point that and then we will go back to then we see about Reed Solomon and BCH code. I think many of you might have forgotten what is syndrome decoder is, it is hood remind you as well as what it is, let us do a simple example and remind what the syndrome decoder was. If you have let us say parity check matrix which might be like this my favorite example parity check matrix, how do you do syndrome encoding.

So, the idea is first of all what the pictures you need a picture for the decoder, so you have message m which gets encoded into code word c, let us say k bits here, n bits here. Today we model it, you should think of binary symmetric channel with some error probability p or equivalently you can think of n something being with the code word and that being the error vector.

So, e n bits again this is the error vector so that you get a received vector which is c plus e what do you do the received vector r what is first step, complete the syndrome s which is H times r transpose k, what I know in fact. So, what is now in fact that that this syndrome s which I compute H times r transpose is also is also H times each transpose, so I know that is what I use.

Then, I have a disturbing for e for my other vector e hat, I have seven distribution, it should be some any from the binary symmetric channel which mean the all 0 co error vector is the most probable. Then, what are the most error vectors, one way it error vector like two error vector like three vector, then you make table you call it syndrome table with error vectors and syndrome. So, you look up in the table and you get your e hat, so you do a look up, I want to write down the loop also in the same box so that I get that.

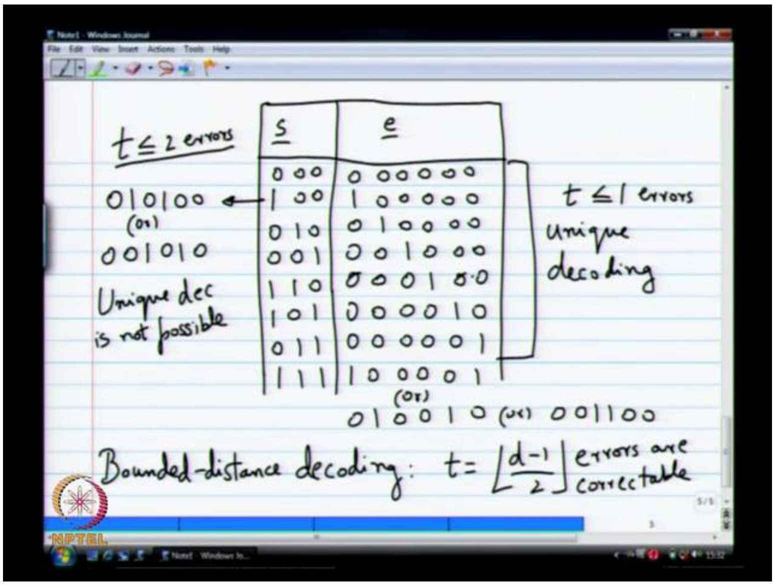
I get e hat, then what do you do with the e hat, add it to you get c hat from here if you want you can read proof based on systematic encoding or compute some computer seems to thinking for itself for too much. So, how do you make this table, let us let me write down few rows of the table with table, it is quite easy k once again you do it in reverse k this 0, you can always very easily right simple 0, 0, 0 and then 1,0, 0, 0, 0, 0, you write 1, 0, 0 etcetera. So, you make a table inclusion etcetera, so basically main thing I want to point out is the main equation you are solving is this by s equals H times a transpose you have n minus k equations and your unit vector e.

So, you have distribution for e you can do a maximum like we could kind of solution if you do not t the most likely error vector another way of doing it is to say that an error vector has only so many one that is called bounded distance of function. That is why you say I know my channel can introduce some error, but it can introduce almost some t error. It can never introduce more than t errors or another way of doing it is saying it might be decoder, I will only correct up to t error beyond t errors I will just give up.

This is let me transverse this, so hopeful beyond t error happen, so badly it does not really matter, so those are various ways of thinking about it and in this syndrome decoder the way we thought it was to say we will do maximum likelihood first elusion for this. So, we will find out which is error vector which is more slightly occurred which will solve s equals H times t transpose.

Otherwise, how many solution for this equation 2 power k if the n minus k solution n the variables, so that will be k 3 things that you can choose and you can chose them in 2 power k different ways. So, you have 2 power k solution, there is no way of taking any one, so you have 2 power solution you pick any one solution based on pick one solution based on the maximum likelihood method. You can say I will do bounded distance, I know I had off time only t errors are happened, and so that also is possible. So, let proceed since we will get different answers based on based on something like this for instance.

(Refer Slide Time: 25:18)



$t \leq 2$ errors

010100 (or) 001010

Unique dec is not possible

| $\underline{s}$ | $\underline{e}$ |
|---|---|
| 000 | 000000 |
| 100 | 100000 |
| 010 | 010000 |
| 001 | 001000 |
| 110 | 000100 |
| 101 | 000010 |
| 011 | 000001 |
| 111 | 100001 (or) 010010 (or) 001100 |

$t \leq 1$ errors Unique decoding

Bounded-distance decoding: $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors are correctable

Let me complete that table in case of that previous pervious things I will complete the table and then we will use both this rules and see what happens. So, table case if you make syndrome table like we discussed before go through and do this, so you will get a certain number of syndromes for this cases.

So, let me just write it down you can go back and check whether I am making a mistake or not, I think you get 1, 1, 0, 0, 1, 0, 1, 1, now for H t, next syndrome it is the only remaining syndrome 1, 1, 1, you have multiple answers. So, you have multiply answers even when you look at r minimum weight am I right even when you look at minimum weight you have multiple answers in cases. For instance you can do first and last one 0, 0, 0, 0, 0, 1 or 0, 1, 0, 0, 1, 0 or 0, 0, 1, 1, 0, 0.

So, if you use maximum likelihood while k fourth first seven there are no problem and for the last one you can pick any one without any problem. So, that is the maximum likelihood, on the other hand if you want to say some kind of bounded distance rule, there is some confusion here. You see there is problem, suppose if I say my decoder introduces less than or equal to 1, so if may say, I am sorry my channel introduces to less than or equal to one error. Then, only have this possibility and you can do unique decoder k, but if you say my decoder is introducing up to two error then you can do unique decoding in fact.

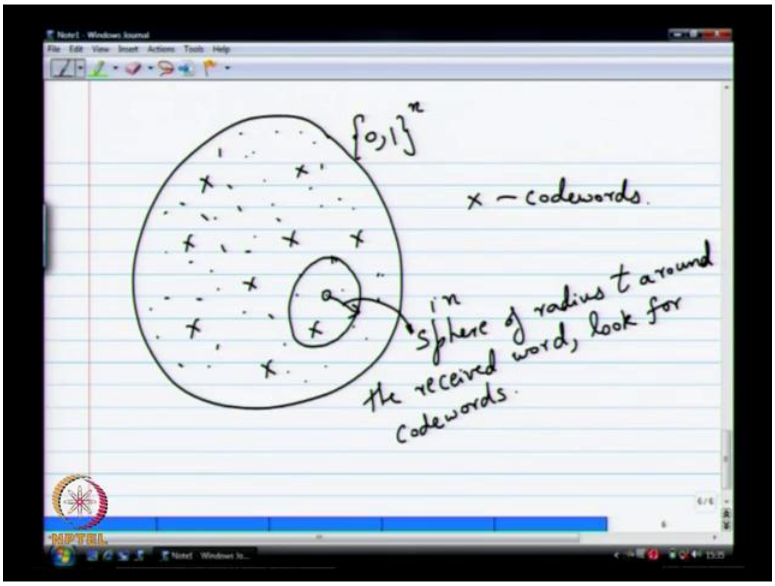
If you include other possibilities for two error what are the other possibilities that are six is 2, 15, these and two error patterns. So, it will end if you include other patterns, you will in fact keep getting the same syndrome, so every syndrome will correspond to multiples possibilities for each case. So, if you say two errors what happens if I say my syndrome is 1, 0, 0, what is the other possibly for two errors, what else can give me 1, 0, 0, if I also for two errors 1, 0, 0, 1, 0, 0, 2 and 4 1 2 and 4, 2 and 4, 3 and 5, 3 and 5. All this things will give 1, 0, 0, this is 0, 1, 0, 1, 0, 0 or 0, 0, 1, 0, 1, 0.

So, for t less than or equal to one errors you have unique decoding for t less then or equals to two errors. It is not possible unique decoding is not possible this how do I say that say unique decoding is not possible. If you say m l k if you say maximum likelihood decoding then t equal 2 and all you should not consider t does not even enter the picture. I am looking at the maximum probability which you have occurred, in fact when you have multiple choices with same probability, I can take any one and that will be still ml k.

So, that is the difference between m l decoding and bounded distance decoding if you say bounded distance decoding, you want in a way unique answers. So, I want one answer which is within t of my received word, so two problems are different, so you have, so hopefully you see the you appreciate the settle point here, so p equal to 2.

You should be correctable by bounded distance decode, it would not it will never work, but p equals 1, you can correct with bounded distance decoding in this code. So, in general if you do bounded distance decoding k t equals d minus 1 by 2 error are corrected, that is why the error correcting capability comes in. If you do ml decoding, you might correct some or two more other error, but it is not, you cannot be sure, you can uniquely. So, you know the probability of error, but then you can say I am correcting exactly, you can only approximately say, so m l more about probability and bounded distance is bit more algebraic.

(Refer Slide Time: 31:18)


$\{0,1\}^n$
x – codewords.
in sphere of radius t around the received word, look for codewords.

You want exact solutions k you can also view the picture, if you draw a picture and you put all star which are code words, you remember this picture long back. So, what are this dots k this entire space is let us say the binary vector space and that stars are the code words. You receive let us say a particular guy here, what do you do when you do bounded distance decoding being at starting at r with that receive point as center you draw a sphere of radius t. You draw a sphere of radius t starting with r k and if you find exactly one code word in that sphere, you declare that as your decoded code.

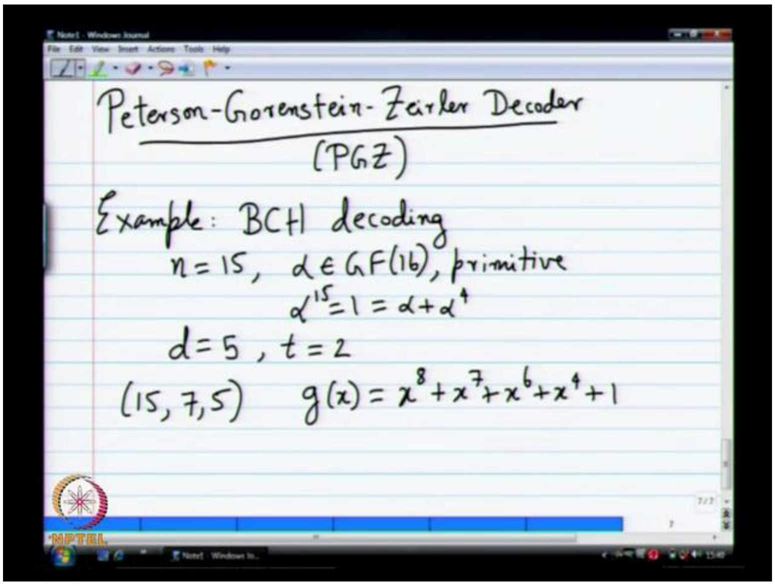
If you do not find anything, what do you do you give that is that is how you work with bounded distance equation. You do not find anything, you say I did not find anything; I fail if you find more than 1, then also you fail some problem, so sphere of radius in sphere of radius t around the received word.

Look for code words that is the idea in bounded distance decoder, if you find any code word, then you say I have decided to that nice anything else happen, you declare a failure. There are two type of failure, once again you may not find anything or you may find more than one both can happen with bounded distance equation is that. So, m l is little different what do you in m l, so what you do is from the received word you look at every possible code word. Then, compare the distance from every possible code word and then pick that code word which is closest that is little bit different.

You do not restrict your space your space to within sphere t around the receive word, you look at every possible code word find the distance from it and pick the one which is closest that is what you do in m l. So, that is different from this, so it turns out for BCH and Reed Solomon codes BCH and Reed Solomon codes, there is a very a efficiencies bounded distance decoder up to the error correcting capability G equals d minus 1 by two m l, decoding is hard. If I not wrong m l decoding is empty completely, it is very hard, but bounded distance decoding can be done.

That you can do even slightly better bounded distance decoding with very good complexity, there are some very advanced algorithms. You will not see all of them, you will see very basic bounded distance decoding algorithm which will work which is very efficient. It is not extensionally complex in it, what I mean by efficient, let us see that and particular decoder, we will discus I will just call Peterson it is due to three people.

(Refer Slide Time: 34:55)



Peterson-Gorenstein-Zierler Decoder

(PGZ)

Example: BCH decoding

$n = 15, \quad \alpha \in GF(16)$, primitive

$\alpha^{15} = 1 = \alpha + \alpha^4$

$d = 5 \ , \ t = 2$

$(15, 7, 5) \quad g(x) = x^8 + x^7 + x^6 + x^4 + 1$

I think amalgamation of all three Gorenstein Zierler decoder, so the first question in the final in is the answer is right if it is showing what is g. It is just giving you the names, so the original ideas are from the several people, but they also used these ideas of course this decoder have implemented and practiced. There are other decoders which are faster etcetera, but this gives you a flavor of the kind of ideas going to building a decoder for Reed Solomon and BCH codes. So, we will we will do two things, first thing we will see is a decoding of an example BCH code by example, we will see the decoding of BCH codes.

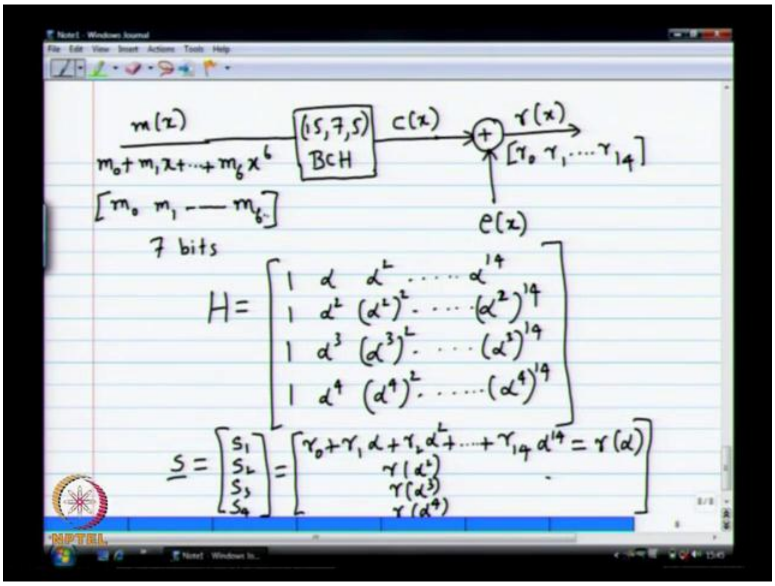
I will give a simple example, we will do the decoding, we will see what it involves, it basically involves solving some equations and then I will present the general decoding method for which is decoding method for Reed Solomon process BCH coding method is a obviously not a PGZ decoder. It is my interpretation or somebody else is interpretation of decoder, let us see the example I think the kind of ideas that go into decoding of this go back to favorite example.

So, example is BCH decoding and I will discuss the Reed Solomon decoder in some detail the example is also interesting general ideas which are important. First of all how do we set it up let us say and there is a example, I will pick n equals 15, I will have alpha G F 16 being primitive, so you have alpha 15 equals 1. Then, just as same as alpha plus you have this and we will look at the code with d equals 3 d equals 5, I am sorry d equals 5, t equals 2, 2 is a error decoding code. You saw this before for the three dimensions 15, 3 parameters 15, 11 and 5 not 1, sorry 11 was 3, 15, 9 and 5 no 7 and 5, 7, 5.

You remember this code what was the generator matrix for this code 8, 6, 8, 7, 6, 4, 1, so x power 8 x powers 6 x power 4, 15, 7, 5. This is a generator, so what we will do, so we remember in the syndrome decoder, we have we were dealing with linear codes and we were thinking of a message vector error vector. Once you can to the BCH codes we also we also view the vectors as polynomials, so we have a message polynomial which gives you polynomial.

Then, we have an error polynomial error if it is received polynomial which will give you a syndrome polynomial. We will see all the all of these things will become polynomial and you can see it is useful. I will also show you rate is rate is simple and straight forward to think of it in terms of polynomial vectors and what is basically because of the structure of the parity check matrix.

(Refer Slide Time: 39:29)



$m_0 + m_1 x + \cdots + m_6 x^6$

$[m_0 \; m_1 \; \text{---} \; m_6]$

7 bits

$[r_0 \; r_1 \cdots r_{14}]$

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{14} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{14} \\ 1 & \alpha^3 & (\alpha^3)^2 & \cdots & (\alpha^3)^{14} \\ 1 & \alpha^4 & (\alpha^4)^2 & \cdots & (\alpha^4)^{14} \end{bmatrix}$$

$$\underline{S} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} r_0 + r_1 \alpha + r_2 \alpha^2 + \cdots + r_{14} \alpha^{14} = r(\alpha) \\ r(\alpha^2) \\ r(\alpha^3) \\ r(\alpha^4) \end{bmatrix}$$

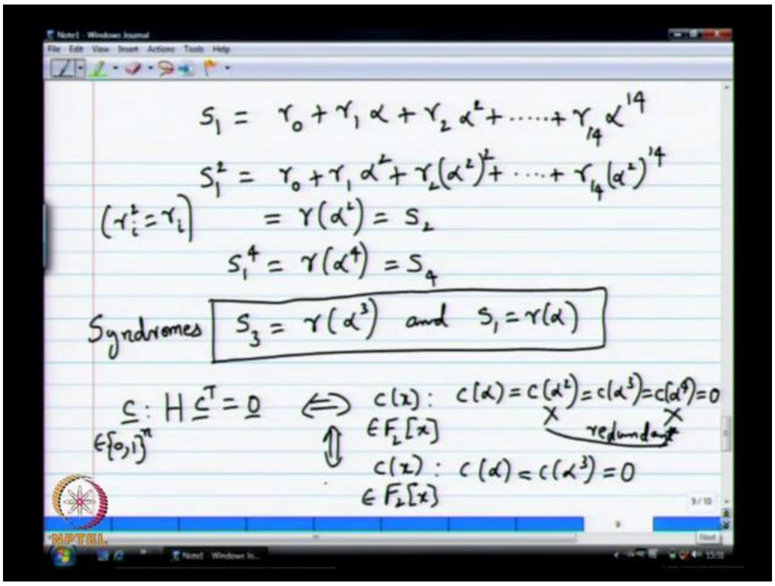
Let us that, so for this specific example, you have message polynomial which will be basically of the form m 0 plus m 1 x plus so on till m 6 x power 6, so k equals 7. So, you have this if you want to really think of it as a vector, you think of it as a vector m 0 m 1 so on till m 6. This is basically 7 bits, so I would not do the same for all the other polynomials; hopefully the relationship is clear to you.

So, you do the encoding 15, 7, 5 BCH maybe systematic decoding, we require a code word polynomial c x this is going through my channel and the channel adds a error polynomial e of x. So, I get a received polynomial r of s what is my parity check matrix the quality check matrix is basically one of a alpha square so on till alpha power 14, one of a square so on till square based to the power 14, 1 alpha power 3. This is my parity check matrix and suppose my received vector is n minus 1, I have a received polynomial r of x have to now compute the syndrome, so compute the syndrome vector is I can write it as s 1, s 2, s 3 and s 4.

So, if you do the multiplication with r transpose, you basically going to get r is 0 plus r 1 alpha, remember all these are bits dealing with the binary BCH code, everything is a bit r 1 alpha plus r 2 alpha square plus so on till r 14. I have put n minus 1, there is basically 14, so 14 r 14 alpha to the power 14, now let us start at a for a while how do you write it in the terms of r of x is basically r of alpha. That is why I said it is easy and nice to think of these things as polynomial as a for the vectors all the structure of the parity check matrix when I am actually doing H times r transpose.

I computing the syndrome the first syndrome is simply an evaluation of polynomial r of x at alpha of what would be the second one s 2 r of alpha square what will be the third one alpha power 3 fourth one will be r of alpha power. So, when you have this special structure for the parity check matrix for which we define the BCH Reed Solomon producing which we define the BCH code. Then, evaluating the syndrome doing H times r transpose is the same as evaluating the polynomial r of x at the roots what are the roots of alpha square alpha power 3 alpha power 4 simply evaluating it at the roots I get my syndromes. So, there is more than one nice thing about of thinking about at this way, I will show you all the notice things about it.

(Refer Slide Time: 44:13)



$$S_1 = r_0 + r_1\alpha + r_2\alpha^2 + \cdots\cdots + r_{14}\alpha^{14}$$

$$S_1^2 = r_0 + r_1\alpha^2 + r_2(\alpha^2)^2 + \cdots\cdots + r_{14}(\alpha^2)^{14}$$

$$(r_i^2 = r_i) \qquad = r(\alpha^2) = S_2$$

$$S_1^4 = r(\alpha^4) = S_4$$

Syndromes $\boxed{S_3 = r(\alpha^3) \text{ and } S_1 = r(\alpha)}$

$$\underset{\in\{0,1\}^n}{\underline{c}} : H\underline{c}^T = \underline{0} \iff \underset{\in F_2[x]}{c(x)} : c(\alpha) = \underbrace{\underset{\times}{c(\alpha^2)} = c(\alpha^3) = \underset{\times}{c(\alpha^4)}}_{\text{redundant}} = 0$$

$$\Updownarrow$$

$$\underset{\in F_2[x]}{c(x)} : c(\alpha) = c(\alpha^3) = 0$$

So, we have one s 1 equals r 0 plus r 1 alpha, I will write it down in the in the same format, so on till r 14 alpha to the power 14. So, what happens when I square this, suppose I square this and s 1 squared what will I get it is going to be r 0 square. Remember this is all characteristic, so I can simply square the individual term and add it up. So, I will get r 0 square plus r 1 square alpha square plus so on what is the r 0 square, it is a binary when you square it you get the same thing, so it becomes simply r 0 plus r 1 alpha square plus r 2 alpha square whole square plus so on till r 14 r.

The reason is r i square is equal to r i, so each of these guys are binary, so this is the same, so in fact what is this r of alpha square which is in fact s 2. So, in a way when I restrict myself to binary codes, the second row of paretic check matrix is redundant, did you see that second row of paretic check matrix is totally redundant. If the first row satisfied, then the second row is also satisfied because the second is the simply the square of the first row.

That is what I mean if you are if some vector is such that H times c transpose 0, some vector c, then it is enough I check the second row will also be 0 x square 0. You get zero exactly, so that is was the nice thing about looking at it in terms of valuation of routes. So, this is this is quite interesting, the same thing you can do for s 1 power 4 s 1 power 4 will be r of alpha power 4 which is which is s 4. So, the only other non trivial relationship that you have is s 3 which is equal r of alpha power 3, you cannot do anything else with it. So, s 3 and s 1 equals r of alpha r non trivial syndromes, these are the real syndromes the other syndromes just dependent on these things.

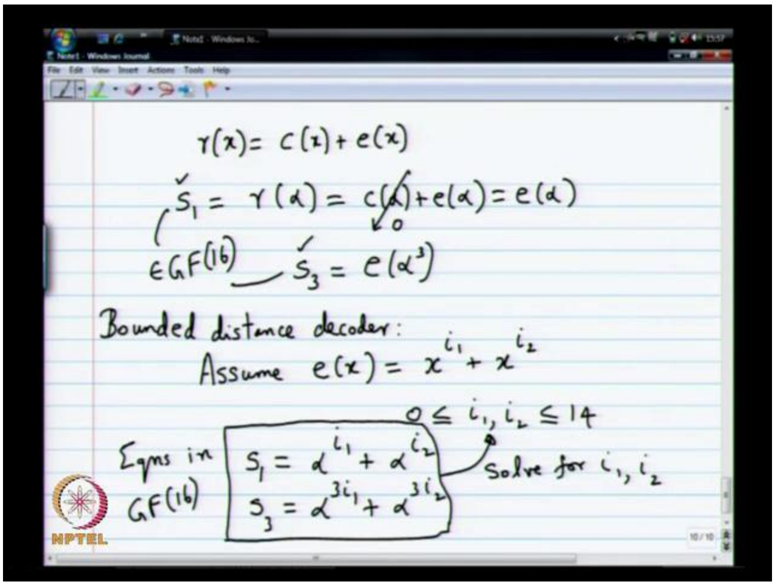
You do not have to use them in solving in anything; they would not tell you anything in your solution, so there are there is another way of thinking about it which will also tells some more light. If in case you are wondering where these all these coming from r of x, I am sorry before i go to r of x its not too difficult to see these things so instead of thinking of thinking of the code word as satisfying this equation.

You can equivalently think of it as c of x such that instead of saying c such that this is true c in binary such that this is true you can say c of x in f 2 x such that what c of alpha equals c of alpha square equals c of alpha power 3 c of alpha power 4 equals 0. So, all these things are routes of c of x that is my definition of the entire code, so clearly even here you see these two are redundant since c of x is a binary polynomial. If I have c of alpha being 0, then c of alpha square is also 0 c of alpha power 4 is also 0, those are conjugates is going to bring with it all the all it is conjugates as routes because c of x is a binary polynomial.

So, in fact this definition you can change and say this is equivalent to c of x in f 2 x of course there is a degree constrained I have not written it down c of alpha equals c of alpha power 3 equals 0. So, only the two syndromes are really meaningful c of r of alpha and r of alpha power 3, there are other really not anything new. So, that is the information about syndrome and then the next, so if you want to look at that bit longer. So, the next observation that is important is moving from the received vector to the error vector right in the in the syndrome d coder.

We had syndrome being equal to H times r transpose, but in fact we did not solve that equation. There is nothing to solve in that equation, we were able to write that as H times e transpose. Then, we solved for the error vector e either using the bounded distance method or the m l method or whatever you use find the unique solution in one of the two ways that is that is the various ways of doing it.

(Refer Slide Time: 49:44)



$$r(x) = c(x) + e(x)$$

$$\overset{\checkmark}{s_1} = r(\alpha) = \underbrace{\cancel{c(\alpha)}}_{0} + e(\alpha) = e(\alpha)$$

$$\in GF(16) \quad \overset{\checkmark}{s_3} = e(\alpha^3)$$

Bounded distance decoder:

Assume $e(x) = x^{i_1} + x^{i_2}$

$$0 \le i_1, i_2 \le 14$$

Eqns in GF(16) $\boxed{\begin{aligned} s_1 &= \alpha^{i_1} + \alpha^{i_2} \\ s_3 &= \alpha^{3i_1} + \alpha^{3i_2} \end{aligned}}$ Solve for $i_1, i_2$

So, let us see what happens here remember r of x is c of x plus e of x that is all module for a error in the polynomial world now what is s 1 r of alpha which is c of alpha plus e of alpha, but what do I know about c of alpha that 0, it is valid code word it is 0. So, s 1 simply becomes, so I do not know what happened, now sorry so s 1 simply becomes e of alpha because this k goes to 0.

So, similarly, s 3 also becomes what e of e of alpha power 3, so these are the two equations that we have to solve. Remember, the equation that we have to solve before was s equals H times e transpose, now we are solving these two equations s 1 equals g of alpha and s 3 equals e of alpha power 3, now I mean I am not going to go through the m ideas.

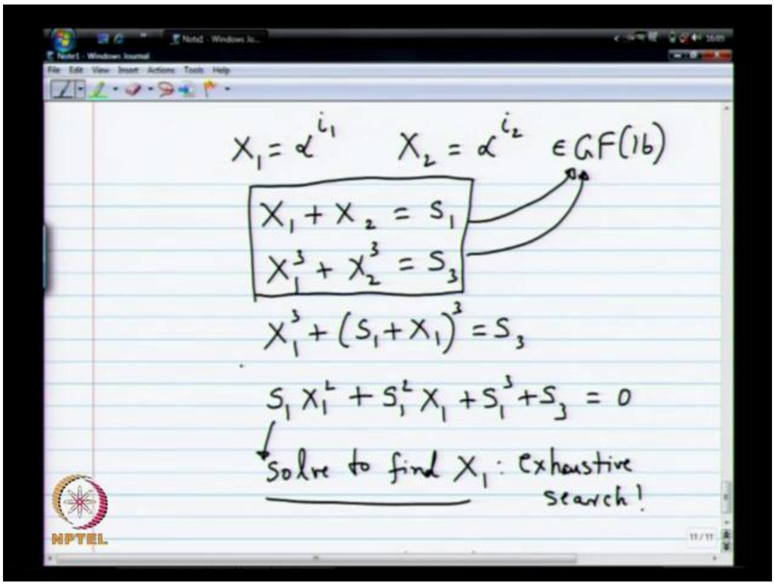
So, let us see what to do if you have to do a bounded distance d code, so let us say I am going to restrict myself bounded distance d code. So, if I am doing a bounded distance d coder what is an assumption I can make about e of x. Remember, I am starting an r of x and I am only looking around r of x with this fear of radius t t in this case is what it is 2. I am just looking at a two error correcting code, so only two. So, what is an assumption I can make about e of x, now it has only two terms, so e of x has only two terms, so e of x will be of the form, so if you do a bounded distance decoding you can assume e of x has the form e of x has that form.

Let us say x power i 1 plus x power i 2 what are these i 1 and i 2 0 to 14 is that, so this is an assumption I can about e of x, so once I assume that e of x is like this. Then, I am looking only at the sphere of radius 2 around my received, but I am not looking beyond that, so let us substitute back into this formula that we have remember s 1 and s 3. I know already s 1 and s 3, I know right these things are elements of what s 1 and s 3 belong to what are s 1 and s 3. It is r of alpha belongs to which field g of 16, so s 1 and s 3 some alpha power something, so it is belongs to g of 16, remember that.

So, I have this equation, now g of 16, it is not anything else, I think seriously, I mean you have invest a new touch screen thing. I think this is previously it was little bit different it was maybe I do not know what is happening. I think the the task bar went up, it is better one thing, but it is changing the page arbitrarily anyway, so let us see let me try and how can I get through of this bottom thing I should see that it goes away. Now, that problem would not be there, so let us see, so this is the problem you have, so if you plug in into this equation, I know s 1. I want to find i 1 and i 2 such that alpha power i 1 plus alpha power i 2 and then s 3 is what alpha power 3 i 1 plus alpha power 3 i 2.

So, these are the two equations that you have and you have to solve for i 1 and i 2 and these are equations in remember g of 16. So, these are not equations anything else, so one thing that is a little bit weird about this equations is equations are in g s 16, but the solutions are integers. It is not a good idea to have such things where equations are somewhere; you should have solutions in that same area you are looking in. Otherwise, it will just con complicated, so what is one way of moving from integers to g of 16. So, already there, but I want instead of saying solve for i 1 i 2, I do not want to solve for i 1 i 2, so what I will do is I will do a very a complicated change of variables.

(Refer Slide Time: 56:18)



$X_1 = \alpha^{i_1}$ $\quad X_2 = \alpha^{i_2}$ $\in GF(16)$

$$\boxed{\begin{aligned} X_1 + X_2 &= S_1 \\ X_1^3 + X_2^3 &= S_3 \end{aligned}}$$

$$X_1^3 + (S_1 + X_1)^3 = S_3$$

$$S_1 X_1^2 + S_1^2 X_1 + S_1^3 + S_3 = 0$$

Solve to find $X_1$ : exhaustive search!

So I will actually take a very change of variables, I am going to say x 1 equals alpha power i 1, then x 2 equals alpha power i 2, then you will see it is already begin to look a little bit better. So, what I will say instead of solving for i 1 and i 2, I will solve for x 1 and x 2 and now these case are in the field I want, so let us see, I have given a totally different. Either I did something or now I think it is these two guys are in g f 16. So, this is this is not bad, now I can write equations where everything is from the same field without any without any problems.

So, basically I have two equations x 1 plus x 2 equals s 1 and then x 1 power 3 plus x 2 power 3 equals s 3. These two guys once again are from g f 16 and I have to solve for these two things, so you have two variables two equations what is the standard way to proceed, I eliminate 1, so substitute to find out x one in terms x 2 and then substitute in the second equation. So, let us do that, so let us say we find x 2 in terms of x 1 what is x 2 in terms of x 1, s 1 plus x 1.

So, you get x 1 power 3 plus s 1 plus x 1 power 3 equals s 3, so if you simplify this you see that x 1 power 3 cancels, so this is nice, then what else will you have? So, s 1 x 1 square plus we you have 3 s 1 s 1 square, but 3 is same as 1 in my field and then you have s 1 square x 1 plus s 1 power 3 plus s 3 equals 0. So, what kind of n equation is this quadratic, I mean it is you are afraid it is a quadratic equation in x 1 with co efficients from g f 16. You can use your standard formula to solve it, there is a small problem with the standard formula, there is a 2 that shows up in the denominator which means at some point you have to divide it by 0.

It is not possible in g f 16 your standard formula does not work in case your standard formula does not work what you will do, complete the squares. You cannot complete the squares this standard formula comes from completing the squares if you do not complete squares what will you do, you have to do it in some other way. So, it turns out you can do it, but one very simple way is what I mean there is one advantage, you have with finite fields that is that they are finite, you try all the possibilities and then find the solution.

So, that you can do in finite fields which you cannot do in the real field or anything, but very imminently you can do that in the finite fields. There is no problem in fact even in the real fields you can do it, but any way some more little bit more complicated. In finite fields, you can very easily build the circuits which will check one systematically one after the other and find out the answer, in fact there is no general method beyond that for quadratic you have a method a bit more complicated. That is a method for solving quadratic equation beyond quadratic the finite fields, you just basically test everything.

I mean how bad can it get, you know I mean you can even if you go to the field of size 2048, you can build a parallel check for all the 2048 possibilities and it will be done in like a few nano seconds. You know it is very fast today, building this kind of circuits is trivial there is no big deal in fact even pipe line it and everything at the same block and it will work very fast. So, this is not a big problem, so it is a bit complex, but it is only as complex as the size of the field, it is not complex in any other ways.

So, you solved this solve to find x 1, there are some curious cases you have to consider before doing this. For instance s 1 could be 0 if s 1 is 0 what does it mean, why should s 3 be 0 if s 1 is 0 s 1 is just r of alpha why is r alpha power 3 it should be 0, see s 1 can be 0 only when s 1 and s 2 are equal which means there is no errors. So, if you restrict yourself to two errors, you would not get the case when s 1 is 0 and s 3 is non zero. Remember, you are restricting yourself to two errors that is important if you restrict yourself to two or lesser errors you can never have the case that s 1 is 0 and s 3 is non zero.

Actually, in practice if you get that it does not mean that went, I mean something inferable happen, it means more than two errors would have happen more than two errors would have happen to make s 1 0 and s 3 non zero. So, at that point you give up your bounded distance d coder says I have not found anything with in my radius. Then, you can work, so solve to find x 1 when I say that I am also saying determine with situation has happened determine whether two errors or lesser have happened determine whether more has happened and you are getting it.

So, all these situations are captured, so it is important, but one easy of solving is to borate 4 search for it exhaustive search that is one way of solving this equation. So, what I want to highlight is these kind of equations are critical, so you can see why ultimately for an arbitrary BCH code with an arbitrary error correction capability. You will ultimately get an equation of this kind, you will have an r of alpha which is the same as e of alpha which will be x 1 plus x 2 plus so on. Then, you will have r of alpha power 3 which will be e of alpha power 3 which will be x 1 power 3 plus x 2 power 3 plus x 3 power 3. If you have more than two errors, then you will have a r of alpha power 5 which will be x 1 power 5 plus x 2 power 5 plus x 3 power 5 so on.

Then, you will possibly have s seven which will be r of alpha power 7 and you will write all these equation and you have to solve these kinds of equations. So, these are non linear equations in multiple variables and in fact there exist very smart technique to solve them using a set of identity called Newton's identity. So, this relate x power sums to some other polynomial, so it can be solved, so it looks a bit a scary if you have not seen it before. You will think it is cannot be solved, but it can be solved. Also, there are methods that solve this, I am sorry this will give you both the answers, this is what you saying, you have to write final answers the comment was in this particular case.

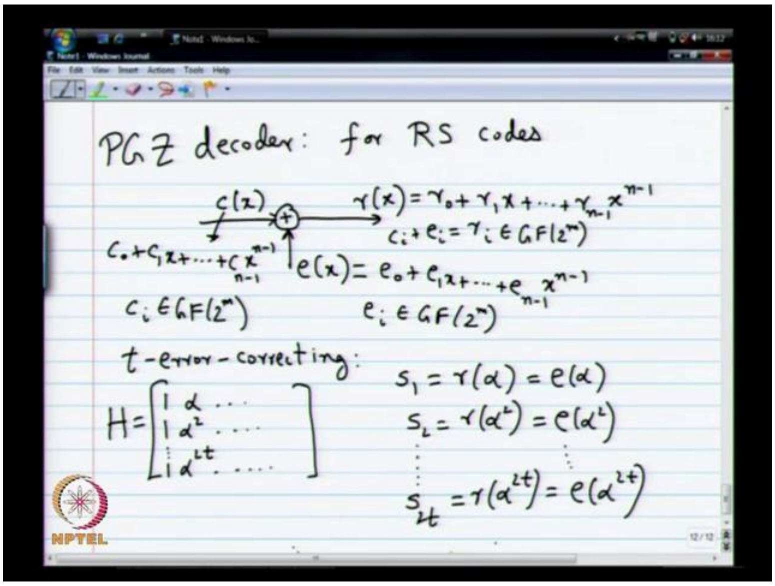
When I solved this equation, I will also get the value of x 2, two routes, so usually if there is no two distinct routes when you when you have a equation quadratic equations with real numbers you always expect two routes either with multiplicity. It can be either one route with multiplicity 2 or 2 distinct routes, so that is how we expect it, I mean finite fields all kinds of possibilities will occur you might have no solutions. You might have two distinct solutions or you might have same solution occurring twice only when you have two distinct solutions will there be two errors.

If you have same solutions occurring twice, then there would have been only one error that can also happen if there was only one error what will what will happen to this situation s 3 will be s 1 power 3. So, what happens, its constant term goes to 0, so what are the two solutions x 1 equals 0 and x 1 equals x 1 itself, but x 1 equals 0 makes no sense because x 1 is alpha power i. So, clearly it cannot be 0, so making it 0 is not possible, so from there you conclude that there is one error and that was cosmic error. So, these are all various ways of of figuring it out, so you have finally the moral of the story here.

So, you have a non linear equation in multiple variables which involves these power sums. You have to do some manipulation change of variables and get 2 1 equation or one type of equation and depending on how the solutions to that equations come about. You will have different kinds of result for your decoder if you have distinct results, then you have good results or if you have any solutions at all, you have good results. If you have no solution or some crazy situation happens, then your decoder has failed, so that is the way to think about this. So, I have some example, but I think if I do numerical examples it is not so interesting when you see you see what is actually happening.

There is nothing much beyond that I have to do so what we are going to do next is to look at the resole men decoder. So, maybe we will numerical example with the resole men d coder, it is a little bit more interesting. So, we will see it at that point, so let me do at least the set up of the d coder for resole men, it is a little bit more complicated this is the p and what I am going to do next is the PG set d coded was about to say that.

(Refer Slide Time: 01:07:09)



PGZ decoder: for RS codes

$$c(x) \rightarrow \oplus \rightarrow r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1}$$

$$c_i + e_i = r_i \in GF(2^m)$$

$$c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \qquad e(x) = e_0 + e_1 x + \cdots + e_{n-1} x^{n-1}$$

$$c_i \in GF(2^m) \qquad e_i \in GF(2^m)$$

t-error-correcting:

$$H = \begin{bmatrix} 1 & \alpha & \cdots \\ 1 & \alpha^2 & \cdots \\ \vdots & & \\ 1 & \alpha^{2t} & \cdots \end{bmatrix}$$

$$s_1 = r(\alpha) = e(\alpha)$$

$$s_2 = r(\alpha^2) = e(\alpha^2)$$

$$\vdots$$

$$s_{2t} = r(\alpha^{2t}) = e(\alpha^{2t})$$

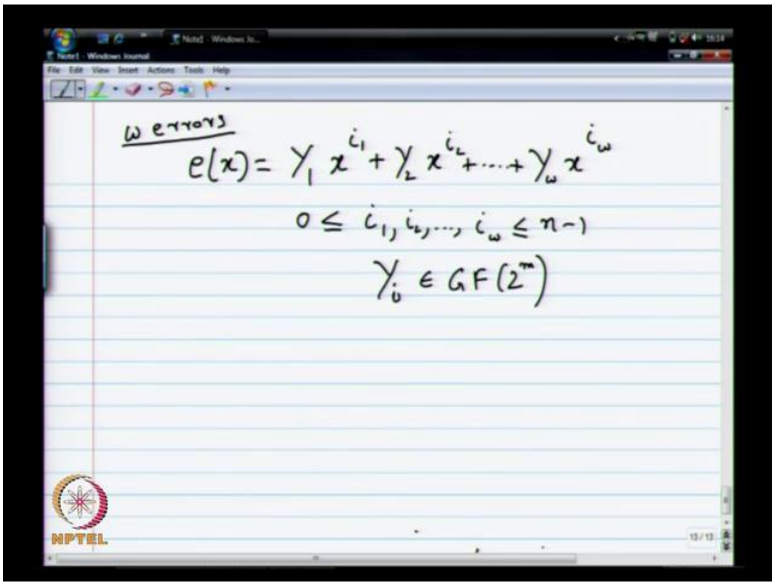
So, now you go to the PGZ decoder this is the way described it is for RS code, so remember what is a set up again, we have a code words c of x for which an error vector e of x the error polynomial e of x is being added. I get the same polynomial r of x, so if I have a resole men code, this guy is now what c 0 plus c 1 x plus so on till c n minus 1 x to the power m minus 1. Each of these c i's now belong to G F 2 power m, now the same thing will hold for the other guys also e of x will be e 0 plus e 1 x plus so on till e n minus 1 x rise to the power n minus 1 and each e i will belong to G F 2 power m.

So, same thing will happen to r of x also r 0 plus in fact r i will be c i plus e i, so this is our general set up and we want to find the syndrome corresponding to the same polynomial r of x. So, if I have let us say a t error correcting code, so how many syndromes I have to find go back to my parity check matrix 2 t. So, I will have my parity check matrix what will be my parity check matrix, remember this is not binary any more. So, I cannot simply throw away alpha square i have to do alpha square everything I could throw away alpha square because it is binary.

You cannot do that anymore, so I have one alpha so on 1 alpha square, so on all the way to 1 alpha power 2 t. So, the syndrome will evaluate as s 1 being r of alpha which is e of alpha likewise s 2 being r of alpha square e of alpha square all the way down to s 2 t which is r of alpha power 2 t which is e of alpha power 2 t. So, this is very similar, the set up we had for the BCH code, not very different, so we will now have to make a bounded distance assumption on the error vector.

We will have to say I have, I mean I have a t error correcting code, so I will only look at around my received word, I will only look at vectors that are at distance t or lesser away. So, what is that mean to e of x e of x has less than or equal to t terms where we have to take in general as t terms. So, how do I assume e of x, now so remember previously I could simply say x power i 1 plus x power i 2 because binary there was only one non zero thing. Now, I have some coefficient also, I have to put the coefficients, also we will take e of x to be of this form e of x.

(Refer Slide Time: 01:10:58)



w errors

$$e(x) = Y_1 x^{i_1} + Y_2 x^{i_2} + \cdots + Y_w x^{i_w}$$

$$0 \leq i_1, i_2, \ldots, i_w \leq n-1$$

$$Y_i \in GF(2^m)$$

So, we will say let us say what I should say should be careful here so let us say w errors is what we will assume. So, I could said w equal to t we will come to that, but let us start with w errors. It is good to do that, so we will take it as y 1 x power i 1 plus y 2 x power i 2 so on till y w x power i w and what are these i 1 to i w its between 0 and n minus 1 and what about these y i s, it belong to G F 2. So, this is r e of x and we have to solve for these equations, we will see how the equations become in the next lecture. So, we will stop here for now, there is about a minute left, you can stare at this picture see if that gives you some inspiration, so we will stop here.