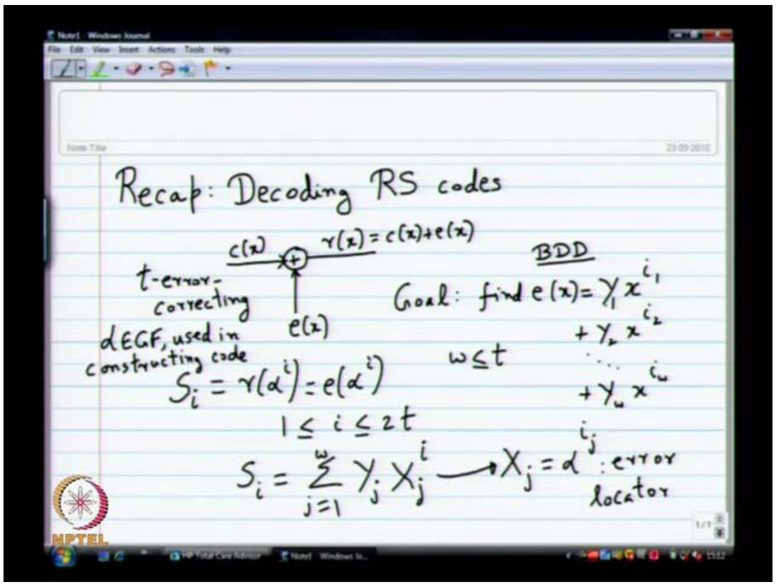
Coding Theory
Prof. Dr. Andrew Thangaraj
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras

Lecture - 17
Coded Modulation and Soft Decision Decoding

(Refer Slide Time: 00:16)



Recap: Decoding RS codes

$$\begin{tikzcd} \arrow[r, "c(x)"] & \oplus \arrow[r, "r(x)=c(x)+e(x)"] & {} \\ & e(x) \arrow[u] & \end{tikzcd}$$

t-error-correcting

$\alpha \in GF$, used in constructing code

$S_i = r(\alpha^i) = e(\alpha^i)$

$1 \leq i \leq 2t$

$S_i = \sum_{j=1}^{w} Y_j X_j^i \longrightarrow X_j = \alpha^{i_j}$: error locator

BDD

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

$w \leq t$

So, let us begin with as usual the recap, the last thing we were doing we were talking about the decoding Reed Solomon codes. So, let me quickly remind you what the what the entire idea was, so you have, so the first thing is we are going to view everything as polynomials if you remember correctly. So, we are going to say you have a code word polynomial, which gets transmitted through the channel and that is modeled as addition with a error polynomial to get a received polynomial.

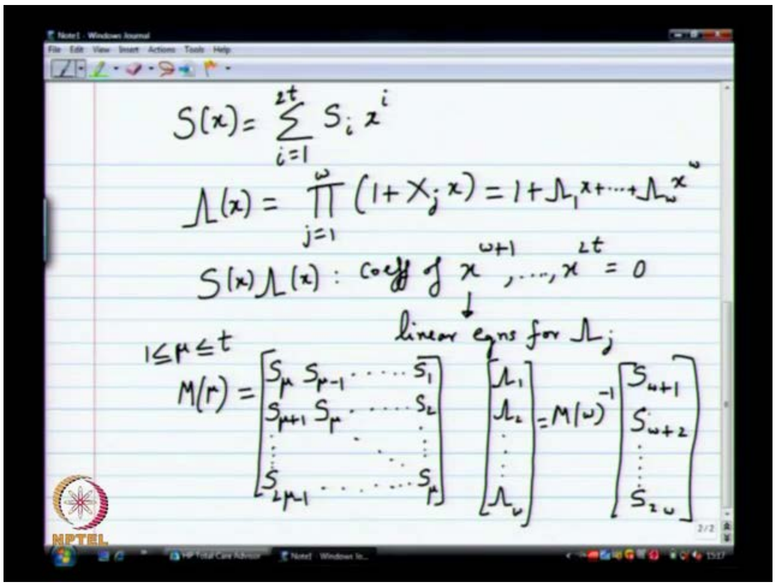
So, this is what you have given and each of this polynomials n minus 1 terms, so those are all understood. So, r of x becomes c of x plus e of x and the goal is to find e of x, that is the idea of this decoder. You have to find c of x of course, from r of x, but we try to find e of x. So, for that purpose, we do basically bounded distance assumption and we say that there are let us say for instance w errors in e of x only w non zero terms out of the possible n terms. Only w of them are non zero and we use a notation for it, basically you say its y 1 x power i 1 plus y 2 x power i 2 so on till y w x power i w.

So, we assume that the error polynomial is in this form, so that is the first simplification the bounded distance decoder assumption. When I say BDD it is bounded distance decoding, so you make that assumption and if your RS code is let us say t error correcting, you usually assume that this w is less than or equal to t. So, you have at most t errors, so the first task is to find syndromes the syndrome be i th syndrome its defined as r evaluated at alpha power i, what is alpha, alpha belongs to some Galva field used in constructing the code.

So, hope it is clear what I mean by RS alpha and we saw that the this is the same as e of alpha power i for i between 1 and 2 t. So, the first 2 t powers of alpha shows show up as rows in the parity check matrix, so alpha power i will c of alpha power will be 0. So, if you can evaluate r of alpha power i you get e of alpha power i. So, in effect you can write s i as summation, so I will just write this notation in short hand since I am repeating everything hope fully.

It is clear to you what i mean y j x j power i and what is this x j x j is alpha power i sub j where this is the error locator. So, this s i's can be evaluated and you have non linear equations, so these are basically power sum equations sums of powers of x. You want to determine the error locator, so the goal is to find obviously the error locator x j and the error magnitude y j you have to find out both.

(Refer Slide Time: 05:11)



$$S(x) = \sum_{i=1}^{2t} S_i x^i$$

$$\Lambda(x) = \prod_{j=1}^{\omega} (1 + X_j x) = 1 + \Lambda_1 x + \cdots + \Lambda_\omega x^\omega$$

$$S(x)\Lambda(x) : \text{coeff of } x^{\omega+1}, \ldots, x^{2t} = 0$$

$$\downarrow$$

linear eqns for Λ_j

$1 \le \mu \le t$

$$M(\mu) = \begin{bmatrix} S_\mu & S_{\mu-1} & \cdots & S_1 \\ S_{\mu+1} & S_\mu & \cdots & S_2 \\ \vdots & & \ddots & \vdots \\ S_{2\mu-1} & \cdots & \cdots & S_\mu \end{bmatrix} \qquad \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_\omega \end{bmatrix} = M(\omega)^{-1} \begin{bmatrix} S_{\omega+1} \\ S_{\omega+2} \\ \vdots \\ S_{2\omega} \end{bmatrix}$$

So, the trick the main trick is to once again think of polynomials and define a syndrome polynomial which is s i x power i and then also define another locator polynomial which is product one plus j equals one to w. So, these are the two polynomials we define a syndrome polynomial and an error locator polynomial. We in fact think of this polynomial in its expanded form, it will be it will be 1 plus lambda 1 x plus so on till lambda w x power w.

Then, it turns out you can show in the product s of x times lambda of x coefficients of x power w plus 1 through x power 2 t equals 0. So, this gives you linear equations for the lambda. So, from here you get linear equations for lambda j, so basically the way you when then you have to worry about when that linear equation is solvable etcetera.

So, for that we use this definition of a matrix which is which defines as m of mu for some mu between one, and t what is m of mu s, mu s mu minus 1 all the way to s 1 then s mu plus 1 s mu s 2 all the way down to s 2 mu minus 1 all the way to s mu. This is the matrix you start with mu equals t and find the determinant of this matrix, if this matrix has non zero determinant, then you conclude that w equals t if it has zero determinant. Then, what do you do try mu equals t minus 1, so that way you keep reducing it till the determinant becomes non zero at which point you would have found w.

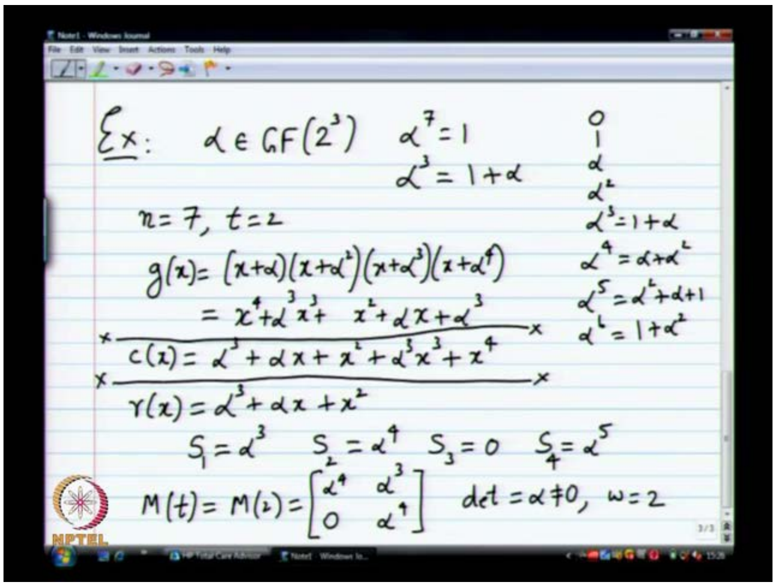
Once you found w, you can simply invert this matrix then multiply with what m of w inverse multiplied with s w plus 1 s w plus 2 all the way till s 2 w. This will give you lambda 1, lambda 2 is that, so you are not done at this point, you have only found the error locator polynomial from the error locator polynomial. You have to find the error locators, so basically you go through the Galva field and find all the roots of lambda of x. If you get exactly w distinct roots, then you have succeeded in you RS decoding, if anything else happens something has gone wrong.

So, you find the x once, you find the error locators, how do you find the magnitude go back and plug in into the syndrome expression. We get linear equation for the error magnitudes, solve them once again get your answer, so that is the that is the idea behind the decoding it is kind of a lengthy process and it will hardly ever show up in an exam. So, maybe you should not pay attention, but it is good to know that this is the method. One thing I should warn you is in practice there are lots of Reed Solomon decoders around Reed Solomon codes are the most common codes out there.

So, hard disks have them CD drives have them, DVD drives, I mean everything has Reed Solomon codes in them and nobody will use this decoder. Of course, there are other versions of this decoder, which are little bit more sophisticated, much simpler to implement etcetera. So, those are the versions that they use, but the basic trick will always remain the same you will define a error locator polynomial and use the fact that syndrome times error locator will give you some linear equations.

Ultimately, that is what you use and waste of solving those things, smartly there are in fact for in sense iterative way of finding lambda of x thing like that is used. So, to complete the picture and just show you how it really works, we are going to do a very simple example. So, it will be a contrived example that will come up with to give you a feel for how it works and then we will just move on.

(Refer Slide Time: 11:24)



Ex: $\alpha \in GF(2^3)$ $\alpha^7 = 1$

$\alpha^3 = 1 + \alpha$

$n = 7,\ t = 2$

$g(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)$

$= x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3$

$c(x) = \alpha^3 + \alpha x + x^2 + \alpha^3 x^3 + x^4$

$r(x) = \alpha^3 + \alpha x + x^2$

$S_1 = \alpha^3 \quad S_2 = \alpha^4 \quad S_3 = 0 \quad S_4 = \alpha^5$

$M(t) = M(2) = \begin{bmatrix} \alpha^4 & \alpha^3 \\ 0 & \alpha^4 \end{bmatrix}$ $\det = \alpha \neq 0,\ w = 2$

0
1
α
α^2
$\alpha^3 = 1 + \alpha$
$\alpha^4 = \alpha + \alpha^2$
$\alpha^5 = \alpha^2 + \alpha + 1$
$\alpha^6 = 1 + \alpha^2$

So, I will take a toy example, of course in reality in practice you would use much larger fields, but if I use large fields, now then we cannot do the computations that easy. So, use a very small field and we will try and correct two errors just to just to make the problem little bit more interesting. So, that is the example we are going to stick with alpha, let us say g f 2 power 3, so alpha power 7 is 1 and then alpha power 3 is 1 plus alpha. So, it is good to make the table. So, you have 0 1 alpha square alpha power 1, 1 plus alpha power 4 is alpha plus alpha square, alpha power 5 is what alpha square plus alpha plus 1, alpha

6 is what is the only other missing 1 plus alpha square keep this table with us help us in computation.

Let us say n equals 7 and t equals 2 n is 7 and t is 2 what are the four roots, the four roots of my code or four 0s of my code, sorry are alpha, alpha square, alpha power 3 and alpha power 4. So, those are the four 0s of my code, so if I have any code word of any received word, I will compute syndrome by plugging in alpha, alpha square, alpha power 3 and alpha power 4. For example, we need to come up with a received code word first, for that we need what you need to come up with the received code word approximate code.

For that, you have to compute the generator polynomial and play around, let us do that first generator is going to be x plus alpha times x plus alpha square and x plus alpha power 3 times x plus alpha power 4 for the first person to compute. This will get five bonus marks in the class, let me see whose going to do that interesting easiest term is the x power fourth term. The next easiest term is the constant term what is the next easiest term x power 3 is also not too bad, x power 3 you have to simply add alpha, alpha square, alpha power 3.

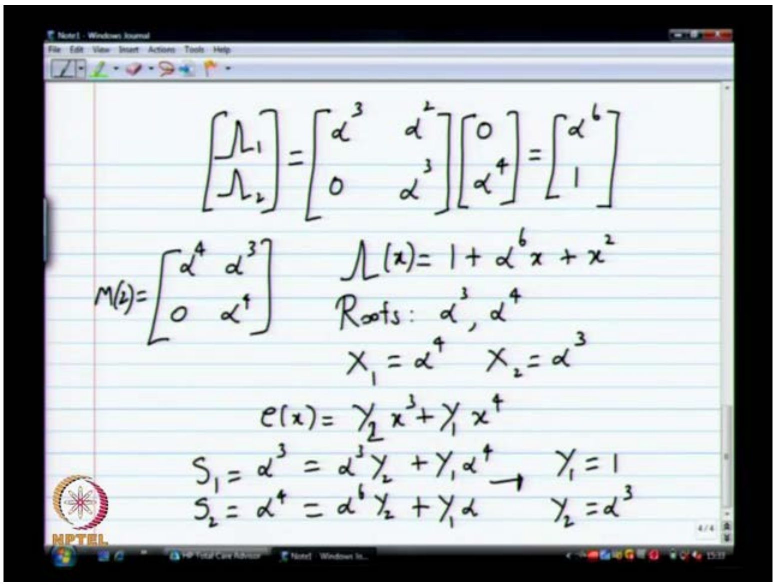
So, you get alpha power 5, x power 3 is that did I make a mistake, alpha power 3 what about x square x square is simply x square when x alpha. So, this is g of x, so let us say we transferred the code word c of x which equals the same thing. So, just to be consistent with the way we have being writing, so I will write c of x like this and to make our computation of syndrome easy what is the god r of x to assume. We want to make your computation of syndrome easy, so what should you get rid of last two terms, you can get rid.

So, it is may be a good idea to assume that the r of x is alpha power 3 plus alpha x plus x square, you people are staring at it like I am doing some crazy stuff. So, I am just saying e of is something I know the e of x already no point in doing the decoding to find out, but let us say you assume you dint know what c of x was this is your r of x. So, this part is unknown to you its suppose just for just for the sake of showing how the decoder works.

Of course, in real life you will have a much larger field and you cannot find out just by looking at the missing polynomial what the code word would have been its not possible, but just to make our job easy, we will do. So, there are two errors and I have clearly nowhere I have introduced them. So, I have to now go through my decoder and make sure that everything works out, so you have to compute the four syndromes, what are the four syndromes s 1 is alpha power 3 s 2 alpha power 4 s 3 s 3 is 0, s 4 is alpha power 5.

These are my quick answers used to check if its fine its fine, one conformation, one more conformation good, so these are the four syndromes. So, next step is what you have to make that matrix and find out if it is singular or not, I know for t equals to its going to be singular already , but any way let us pretend that we do not know it and we start with m of t which is m of 2. That would be what do I put here s 2, s 1, s 1 is alpha power 3. I am sorry and then s 3 s 2 is that, so clearly determinant is non zero is that, so we guess that determinant is non zero determinant is what its actually alpha which is not 0, so w equals 2.

(Refer Slide Time: 19:51)



$$\begin{bmatrix}\Lambda_1\\ \Lambda_2\end{bmatrix} = \begin{bmatrix}\alpha^3 & \alpha^2\\ 0 & \alpha^3\end{bmatrix}\begin{bmatrix}0\\ \alpha^4\end{bmatrix} = \begin{bmatrix}\alpha^6\\ 1\end{bmatrix}$$

$$M(2) = \begin{bmatrix}\alpha^4 & \alpha^3\\ 0 & \alpha^4\end{bmatrix}$$

$$\Lambda(x) = 1 + \alpha^6 x + x^2$$

Roots: α^3, α^4

$X_1 = \alpha^4 \quad X_2 = \alpha^3$

$$e(x) = Y_2 x^3 + Y_1 x^4$$

$$S_1 = \alpha^3 = \alpha^3 Y_2 + Y_1 \alpha^4 \rightarrow \quad Y_1 = 1$$
$$S_2 = \alpha^4 = \alpha^6 Y_2 + Y_1 \alpha \qquad Y_2 = \alpha^3$$

So, we say two errors have occurred and then we go through and find lambda 1 and lambda 2 as inverse of m of 2 what is inverse of m of two what this entry is. You will also had to divide by alpha, so the overall determinant is alpha, so alpha power 3, alpha power 2, let me write the original matrix here. It looks like this is alpha power 4 and alpha power 3 alpha power 4, this was my matrix alpha square 0.

It should be something else, 1 by alpha we have already done, but zero and alpha square have to be this. You are happy is this is easy to check that it is the inverse you just multiply and see if you get I you can check that you will get i. So, this is a valid inverse, then we have to multiply this with s 3 and s 4 s 3 is 0, s 4 is alpha power 5, s 4 is its not alpha power 5, two other people confirmed it to me one plus alpha power five is alpha power four. How many of you say s 4 is alpha power 5, nobody has actually checked, is it, so we do the whole thing you are telling me the final, alpha power 4 think I remember the two people who shook their heads 4.

We will go back and change this, so what will change if I change this nothing here will change, s 4 does not play a role in this only in the next part. So, instead of 5 I will have 4 here, so this is simply alpha power 6 and 1, so my lambda of x basically becomes 1 plus alpha power 6 x plus x square is that y. Are you wondering how I quickly multiplied the two matrices is easy multiplication it is 0 alpha power 4, simply scale the second column by alpha power 4 you get alpha power 6 and 1. So, your lambda of x becomes 1 plus alpha power 6 x plus x square is that yes or no, x square is the correct coefficient.

That much I am sure about because we introduced errors in locations 3 and 4, so alpha power 3 and alpha power 4 are the error locations the inverses are also the same multiply together. You should get 1, so this you have to find roots so what are the roots you can check that the roots will be alpha power 3 and alpha power 4. So, if you want you can check this they have to be the roots computation, no roots are this, then the error locations will be inverse of this which will also be the same what is the trick.

So, alpha power 3 and alpha power 4, so the error locations are x 1 equals if you want alpha power 4 and x 2 equals alpha power 3 is that. So, you have an error on the third location and the fourth location, so you know e of x was of the y 1 x power 3 no y 2 x power 3 plus y 1 x power 4. So, you can play around with this 1 and 2, it does not really matter it is just the way I am trying to be consistent here.

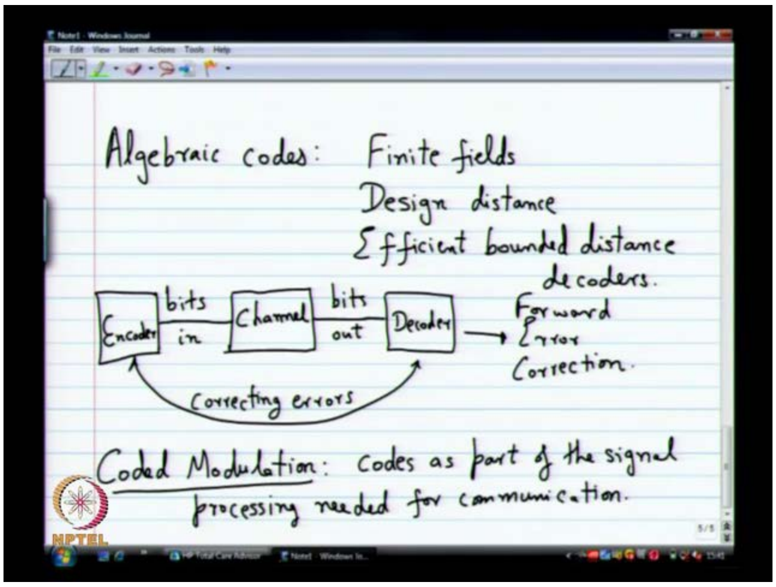
So, s 1 which was alpha power 4 was actually equal to instead of x i should put alpha here alpha power 3 y 2 plus y 1 alpha power 4 and s 2, which was again alpha power 4 equal to alpha power 6 y 2 plus alpha power 3, sorry about that is 2 and y 1 alpha. You solve these two, you are going to get y 1 equals 1 and y 2 equals alpha power 5 alpha power 3, you can check that.

So, it works out, so it works out quite consistently it is not a problem I mean this you will get them, we will get the right answer. So, if you want you can go back home and take this example and play around with it a little bit more. If you do not believe any of the stuff that we did you can add one error for instance and check that m of 2. In fact, it works out to zero determinant, so that you can check, then you will see m of 1 is m of one is very easy to correct, we will correct it. So, this is how a Reed Solomon decoder will function nay questions comments insights.

So, this is in a way significant stopping point in the course as in the we have come to a stage in the history of coding, so to speak where you have achieved something quite significant. So, Reed Solomon codes were considered crown jewel of algebraic codes for a long time and they were they are still use like I said it is the most popular code around any device you take is very likely to have a Reed Solomon code BCH code for that matter.

We see that encoding is very easy you have a shift resistor circuit decoding is also equally easy you only do computations with the small look up table which is very trivially implementable today. So, today Reed Solomon codes I mean I can list the number of places that they are ever where any place you imagine, you have Reed Solomon codes. So, very popular and they deserve their place also, so what we are going to see beginning from this lecture onwards means beginning at this moment onwards is something very different from what we have been seeing so far.

(Refer Slide Time: 28:51)



Algebraic codes: Finite fields
Design distance
Efficient bounded distance decoders.

Coded Modulation: Codes as part of the signal processing needed for communication.

So far we were looking at algebraic codes, basically what were there main characteristics, so to speak of algebraic codes, and you used finite fields for the construction. You get a certain design distance and you have good bounded distance decoders encoders are simple, but usually encoding is not so complicated. As you can imagine in the worst case, it is simply a multiplication by one big generator matrix and you can do it, it is not a problem. So, let us not worry about the encoder, they have good well not good very efficient bounded distance decoders.

So, for a long time the view of coding was that it is not this, what is this, so the way people viewed codes for a long time was that you have a certain communication channel on which you have done a lot of work. So, you have bits going in and bits coming out, so for a long time people were trying to do codes on encoder and a decoder will sit outside here. So, this was considered as the place for the encoding and decoding, so coding was seen as primarily a mechanism for correcting errors. So, that is why they are called error correction codes and this was this was how codes were used for a long time.

So, you have a channel in to which you are putting in some bits you are getting bits out, what happens inside the channel or how is that channel physically realized. What kind of electronics you do on the channel or syndrome processing you do on the channel is completely hidden from you. Some bits are going in, some bits are coming out, there is some probability that the bits might be in error. So, you either had a binary symmetric channel kind of model or in fact better model or a simpler model was the bounded distance model.

You always knew that with very high probability you would have a maximum of let us say some eight errors, eight errors eight byte errors or eight bit errors or whatever you had a bound on that number. Once you had that what did you do for designing you used an algebraic code I made a lot of sense to use an algebraic code because it has a design distance, you know its MDS. So, for the given distance, it is probably a very efficient code, then you have good bounded distance decoders that can be implemented everything was fine.

So, as electronics advanced people slowly started thinking of coding as part of the signal processing in the communication process itself instead of thinking of coding as coming in a separate outer box can we move coding inside closer to the modulation? Let us differentiate between coding and modulation, so far coding you are thinking of bits message bits that can be converted into code words what is modulation in a very loose way we might have defined it very rigorously in communications. I do not know digital communications, but the way to think of modulation is bits are getting converted into signals.

So, electrical signals is there anything to gain from pushing by pushing coding from the outside closer towards modulation and thinking of the whole thing as one big operation. So, people started in the late seventies early eighties or even may be early seventies talking about what is called coded modulation. So, this is a this is a classic error correction picture it was called forward error correction for a long time error correction or FEC and then people started talking about coded modulation.

So, here you think of codes as part of the signal processing need for communication, so instead of thinking of it as some box that sits outside of my entire digital communication device and fix this errors.

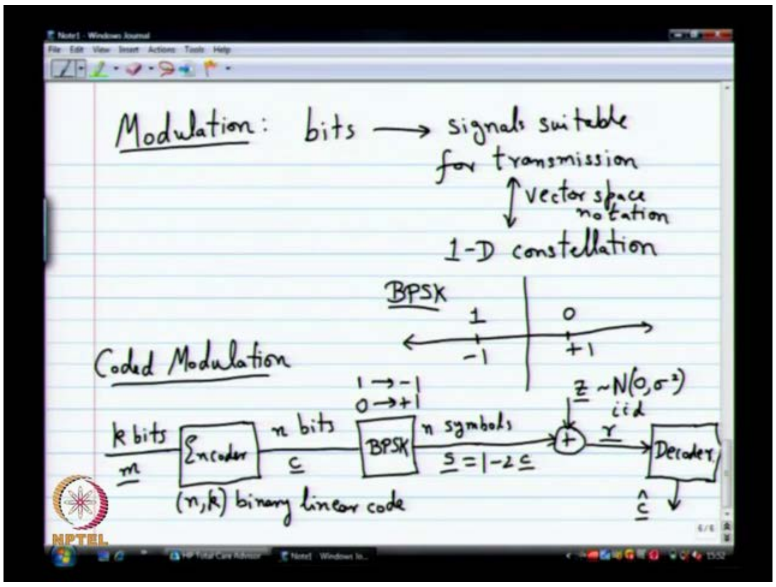
I occasionally make can I move it inside the modulation part of my design so that to give a better performance of course, there has to be a benefit. Now, it is no benefit decoded modulation nobody would do it what bother you just wait with the same pictures before encoder decoder and proceed first of all we have to be sure that there is some benefit in doing coded modulation. If you do that this is you gain something, so does your design become better all those things are questions you have to ask and of course, the answer was yes that is why people did that. The next thing you have to worry about at least in those days the VLSI part of things were not so advanced.

You still had to deal with almost hand optimization of circuit's kind of thing, so if you are doing those kind of things then doing coding at the signal processing level is probably more difficult. So, you will have to deal with maybe I will explain in more detail why it is a little bit more difficult because it requires a lot of memory. Your block length is going to be 2000, previously you were simply dealing with one bit at a time independently inside in the single processing part.

Now, if the block length is two thousand you cannot simply deal with it one bit at a time if you are doing coding inside right you have to wait for all the 1000 signals to come in which means you have to process a huge block of data. You have to do signal processing for that which might be difficult, so that was the significant road block for some time once that problem was solved, once VLSI became advanced enough, then it made obvious sense to do coded modulation.

So, what I am going to describe next is a very simple picture of modulation in a way that will be useful for us way that will be useful for coding it is fairly rigorous. If you do additional communication course, you will see that that is how most modulation schemes are. If you see it first time you might think I am making some basic mistake or making a huge approximation it is not a big approximation, so how do we think of modulation?

(Refer Slide Time: 36:22)



So, we will use this we will use thins two dimensional vector kind of representation for modulation which is quite in fact we will only use one dimension, but just for the sake of it I will draw pictures. So, you can think like I said modulation is the process by which bits are converted to signals suitable for transmission. So, I mean I am going to re do digital communications here, but the way we will model this is instead of thinking of signals we will use the vector space representation for the signals which will give the simple alphabet for the signals.

We will only deal with the alphabet and we know that you have, I mean you can think of the signals you can make a orthogonal basis ortho normal basis for them and then express your signal, in terms of those basis. So, you will get a point in the constellations of the speed for the signal instead of thinking of the old signal we will simply think of a point in a constellation. In fact we will mostly use only a one dimensional constellation, so we will assume these signals come from a one dimensional constellation pretty much for the entire course.

We will only deal with the one dimensional constellation, in fact we will mostly, so this is in the vector space notation vector notation or representation, we will only look at one dimensional constellations. In fact, we would not even look at generic one dimensional constellation, we will simply look at what is known as BPSK, what is BPSK. It is just one dimension and you have just two points in the constellation, which I will normalize to plus 1 and minus 1 and i will say 0 maps to plus 1 and 1 maps to minus 1. So, this is how we represent the constellation, it is one dimension you have simply a line and two points in that line this is binary PSK, it is a very standard constellation.

So, we will not go into any detail on the digital communication part of it, we will simply directly use this BPSK constellation for almost the entire course may be later on. If it is needed, I will mention how to do other constellations, it is not too difficult once you understand how BPSK constellation works, and so what does what does our encoder. Now, look like encoded modulation, so in fact we will not even look at really complicated ways of doing coded modulation there are ways of integrating the coding with the constellation etcetera.

We will not look at all of that we will simply use a coded modulation which is a very simple form of things basically what we will do is we will take k bits with the message which we will call m go through a code. This is the code part of it encoder let us say it is an n k binary linear code. Even if your code is a Reed Solomon code, you can always ultimately come back to become binary linear code, so you are going to represent all you are alphas as some number of bits only. So, this is a good picture to cover that case also, you get n bits which make a code word which call c and then I will have a BPSK block which represents this will output n symbols.

I will get a symbol vector which I will call s what is the BPSK block do if my bit is 0, it is going to make plus 1 if my bit is one its going to make a minus 1. So, I have n symbols there is a wonderful abuse of notation. You can do to write this s in terms of c, so it is very common in mat lab it will work very well. So, write it as 1 minus 2 c of course, it does not make any sense in binary field 2, 0 and then you would not get anything, so you have to imagine. Now, that this 1 and 0s are actually integers and then you do 1 minus 2 c, you get minus 1 and plus 1 it is just a simple short hand notation to represent BPSK.

It is not crucial, but this is good it simplifies some notation later on, now I am going to relay on your knowledge of digital communication to say what will happen if this signal goes through a additive white Gaussian noise channel. So, there is a very common model which is called the AWGN model which is additive white Gaussian model under which in the vector space notation something very simple happens to the signal. What happens to the signal a random variable gets added to each symbol either plus 1 or minus 1 and that random variable is independent from symbol to symbol. It has some variance which we will call as sigma square which might be some n not by 2 in your white noise assumption.

So, we can simply say sigma square, so that is that will be that will be my noise vector, so I will call it z this I will say is distributed normally with mean zero variance sigma square. So, this is the assumption on the noise, so that is a very rigorous justification for it you can start from a set of signals, let us say 2 signals expressive in terms of a basis like this. Then, send that signal out and assume some n of t gets added to it and then you design an optimal front and it is just called the match filter, you design that that. Then, you look at a sufficient statistic, which will be just this which will be some noise vector being added to the representation of the signal in the in the that space notation.

So, this is a very complete and accurate representation of what happens to a full fledged electrical signal and practice. So, this is good enough so you get a received vector r from this. So, now remember is a real vector it belongs to the n dimensional real space. So, it is a real number it can take any value the way a model z is simply a normal variable normally distributed variable zero mean and variance sigma square. So, it can take any value from minus infinity to infinity of course, the larger values are more and more unlikely, but still it might take those values.

So, r can theoretically be distributed over the entire real line. So, now comes the crucial step so far you might say I have not done anything very different I mean what is coded modulation here the crucial idea here is that if the decoder will work directly with r. So, that is the big difference the decoder will directly work with r and produce an estimate of c hat maybe I should re write that a little bit more clearly no decoder.

So, that why I am that is the way I have combined the coded and coding and modulation, so I am going to run a decoder directly on the received values without doing any bit by bit symbol by symbol de modulation will not do any. Of course, that is one choice I will talk more about that later is it so this is a big difference. So, hopefully you agree with me that this is very different from the decoders that we have been seeing so far, this decoders we been seeing so far work in some Galva field right for the Reed Solomon code. If you are using a Reed Solomon code here, they work in a Galva field not in the real field makes no sense to use them on the real field.

There is no direct way that you cannot compute a syndrome, it would not make any sense no roots no polynomials nothing it becomes a problem in what is known as detection or inference or learning whatever you want to call so many different areas these days. So, all of them are pretty much the same they can call it decoding it is not too bad, so it is important to understand what is going on here in more detail we will see in more detail, but this is the picture. In general that we will be looking at for pretty much the rest of the course so we will look at different coders and encoders and decoders which work in this picture and like there is a way in which you can study.

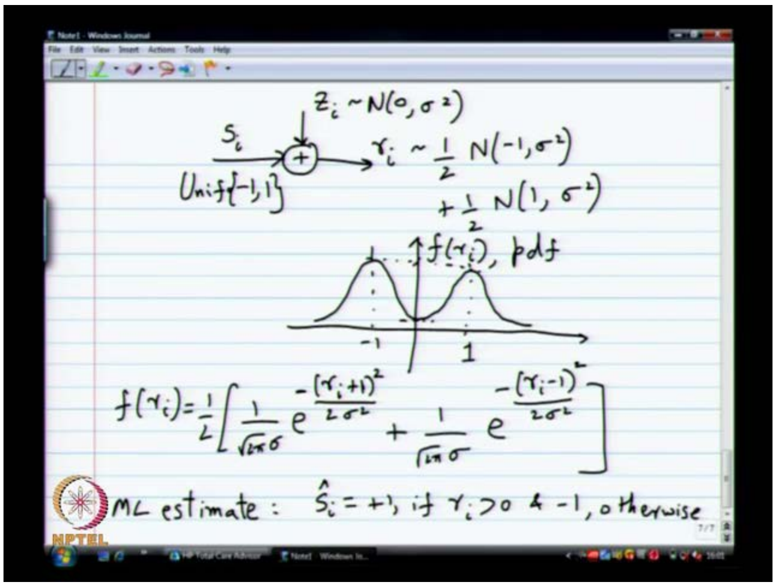
What is called capacity of the systems and show that a system like this if it works well will be about three d b better than a system like what we had before it is a rough number so maybe it will be more. So, basically there is lots of signal power that you can gain by doing coded modulation directly as opposed to only dealing with bits that is the idea. So, roughly we may not have time to see that also may be maybe we will see it, we will see depending on how much time we have any questions about this picture see c cap is supposed to be a code word.

So, if you are doing systematic enquire for instance I will let me not say it is a it is a estimate of coder, but you have to think of it in that way from that you are going to compute the message as let us say systematic. Let us assume systematic encoder so for the first k bits of c cap, you will say is m cap that is all you will not do any further decoding after that and supposed to be directly. So, I am going to ask more questions, so if you have any questions it is a good time to ask because typically people misunderstand something when I write like this it seems very clear when I write it , but when I ask the next few questions, it seems a little bit more confusing usually people get confused by that so stare at it for a while hopefully you understand these things is it.

So, one thing I have to clarify is the notation I will happily kind off do not know what to say walls let us say move here and there between random variables and values taken by them. So, I will not distinguish between those two too much in my notation, so for instance here the z vector. I have said is an random vector which is normally distributed in fact even the c or the s is the random vector message is supposed to be random everything is random, but we are not dealing with them in that fashion.

So, we are thinking of them as values taken also you just think of that also as the same thing, but remember these things have distributions, so that how they look, so the first question that I am going to ask you is the following it is a very simple question.

(Refer Slide Time: 48:43)


$z_i \sim N(0, \sigma^2)$

s_i, $\mathrm{Unif}\{-1, 1\}$

$r_i \sim \frac{1}{2} N(-1, \sigma^2) + \frac{1}{2} N(1, \sigma^2)$

$f(r_i)$, pdf

$$f(r_i) = \frac{1}{2}\left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r_i+1)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r_i-1)^2}{2\sigma^2}}\right]$$

ML estimate : $\hat{s}_i = +1$, if $r_i > 0$ & -1, otherwise

Let us suppose we look at one particular symbol, so let us say, s i particular symbol its going through the channel, so one z i which is normally distributed with 0 mean and variance sigma square goes in what will be the distribution of s i. If you assume equally likely messages etcetera, it is going to be uniform, it will it will be discreet distribution. So, it will take only 2 possible values minus 1 and plus 1 with probability half, so let’s have some notation for it we will say uniform with curvy linear this curve brackets minus one and one. So, it is curve brackets, so I get an r sub i, my first question is what is is the distribution of r sub i, nobody knows this that is what I like most about. So, r i is going to be, I mean you are essentially, it is going to be a two Gaussian s added up, so you can think of it as half normal minus 1 sigma square plus half normal plus 1 sigma square. So, this is a very, I mean I am writing a bad notation here, so hopefully you see what it is if you draw the PDF of r i how will it look?

This is the p d f, I will look like this depending on the value sigma which you will assume small enough compared to minus 1 and plus 1. It will look like this question what is this height minus 1 by sigma square all that would not come e power minus 1 by sigma, there will be no e power minus 1 by sigma square. Everybody agreed on that in that in that height good what will be what will that height be small factor e power minus 2 by sigma square will be there why you saying there will be no e power minus 2 by sigma square.

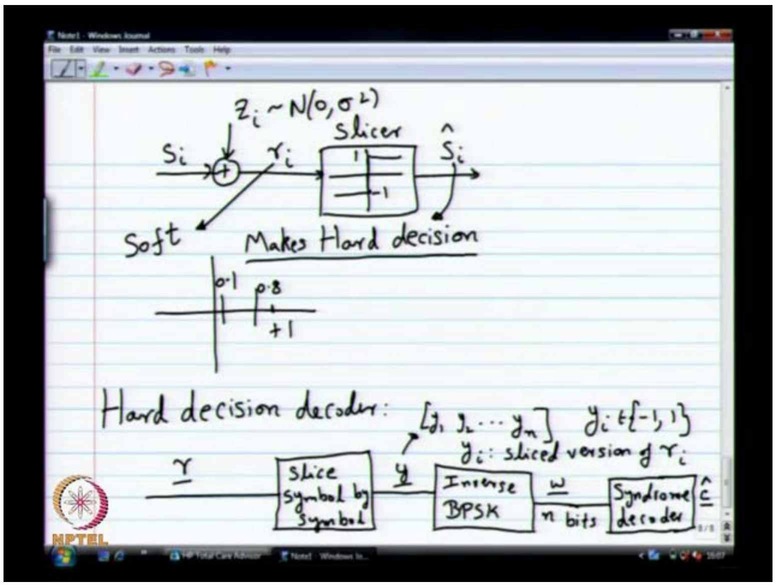
You cannot take it will be a small term, but if it will be there right it will also be a 1 by root 2 by sigma square before that, so you know what about this height. Hopefully, you know enough about normal distributions to compute these two points you have to write the expression. So, f of r i will basically be this expression half 1 by root 2 by sigma e power minus r i plus 1 by r i plus 1 square by 2 sigma square plus 1 by root 2 by sigma e power minus r i minus 1 by 2 sigma square.

So, this will be the actual full expression for f of r i the question i asked about that height was basically plugging in r i equals plus 1 or minus 1. So, this will be an even function even function or not it will be an even function, so you either put plus 1 or minus 1. You will get the answer and the question about this height was putting r i equals 0, so you get some other answer is that, so that is the PDF. So, if you are looking at if you are sitting at the receiver and looking at r i will take values around plus one and values around minus 1.

So, this is how the the picture will look for BPSK, so there is no, I am not talking about coding m. Now, suppose I ask the question what is the best estimate of s i given r i, so given r i. Suppose, I want to find s i cap nothing and ignore everything else it is very easy to do that, you can do an m l estimate any r i greater than 0. Then, you decided s i was s i cap is plus one r is less than 0 you decide s i cap is minus 1. That is the best thing you can do for a particular i is that that is important you ignore everything else for the particular i that is the best you can do.

So, let us do write that down also m l estimate or the maximum likely hood estimate you can also compute it correctly. I think most of you should have enough experience to know what I mean when I say this s i cap is plus 1 if r is greater than 0 and minus 1 other wise. So, basically you can do what is called slicing so you can slice r i it is a very crude term to use to express this basically you do a slicer and it will work like that, so let me write that down also.

(Refer Slide Time: 54:52)



If you have s i and z i normal 0 sigma square is getting added to it you get a r i you slice the picture for the slicer is this. So, normally the things are not marked, so basically its plus one and minus 1, so this is the slicer you would get a s i cap, so you can think of it as the best possible estimate. You can have for s i given that particular r i only, so you have not given anything else only given r i is the best possible estimate for s i what you can do so. So, the slicer is also called a hard decision hard decision device or the hard decider, so it say slicer make hard decisions.

So, this is called the hard decision and this r i is called soft, so the reason why you would say r i is soft and s i hat is hard is when you can see you can see why that makes sense. So, suppose I am given a value of r i suppose in this scale this is plus 1 suppose I am given a value of r i which is let us say something like point eight r i point eight. When i say r i is 0.8. So, it is close to plus 1 or another possible value for r i could be let us say 0.1. So, if I say r i 0.1 and in at one time. When I say r i is 0.8, the two mean something different, on the other hand if i say slice and look at what came out of the slicer for both.

I would get the same plus, so the slicer loses information in some way right it loses that softness of information. So, it loses the belief or how confident you are that s i capo is plus 1 so that is something that it loses, but remember given r i alone the slicer is the best you can do there is nothing else you can do it is the best you can do, but having r i itself might be more useful because it has more information in it value can mean something.

So, one choice for the decoder so remember those decoder I drew working on r one choice for you decoder is the hard decision decoder. So, let us say I make, I do a hard decision decoder, so what is a hard decision decoder, so when you get r from before this r is coming from before it so in the hard decision decoder the first thing you would do is slice symbol by symbol.

So, you would get something let me call that y the vector y what is this vector y basically it is what you get after slicing r i. So, y is y 1 y 2 y n y i is minus 1 plus 1 and y i is the sliced version of r i. So, y i is then what you can do is you can go from y to some other may be y was not a good choice because I have used z for something else well. Let us say from y you go to w how do you do that you do inverse BPSK, you can write an expression if you like for inverse BPSK which will be one minus y by 2. So, that is inverse BPSK, let us say u get a w now remember w now is n bits so at this point.

You can run your syndrome decode, so this is the hard decision decoder which will produce definitely a estimated code word would be output. So, remember when I do slicing symbol by symbol given r i alone y i that is the best possible thing I can get for y i because nothing seem to be anything suboptimal given that symbol , but this but overall there is some sub optimality here. So, I will come back and comment on this later on once I have come to this hard decision versions I can do inverse BPSK and then syndrome decoder is optimal right it is an m l decoder.

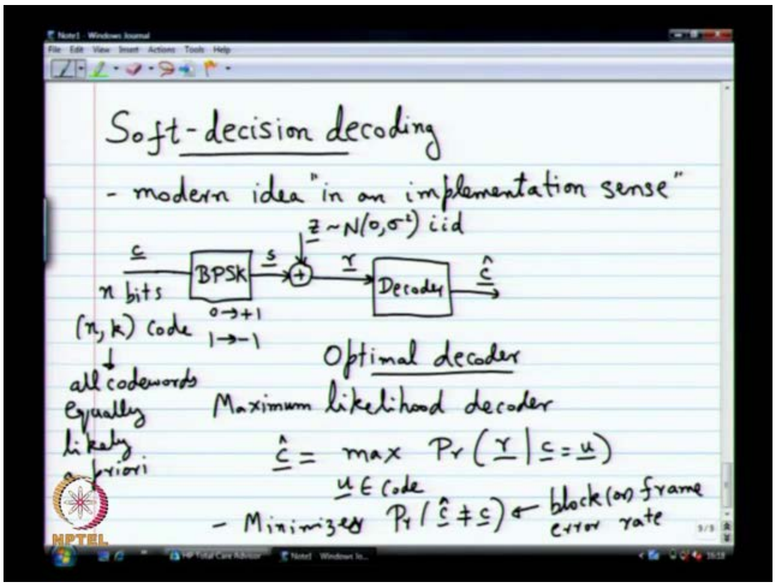
So, you do an m l decoding you get you get the c cap in fact from the code word c to w you can draw binary symmetric channel. So, you can do that its equivalently a binary symmetric channel remember how did i get r came from ultimately some code word c. You can do an equivalent channel from c to w and that will be a binary symmetric channel what will be the transition probability of the binary symmetric channel probability that the noise is strong enough. So, that your slicer makes a error, so what will be the probability q of 1 by q of 1 by sigma.

So, you remember hopefully you remember enough of digital communications to know this, so it is the probability that the slicer makes an error its q of 1 by sigma, so from the code word to w i have a binary symmetric channel. So, clearly the syndrome decoder is an optimal m l decoder you get it. So, this is the hard decision decoder and this is the same picture like we had before you are really not doing coded modulation here. It sounds like I am doing coded modulation, but the slicing symbol by symbol is basically a independent de modulation.

I am not taking the coding into account when I do slicing symbol by symbol, there is some information lost here the reason is r the bits the elements of r are not independent r 1 r 2 r 3. If you look at all of them together they are not independent really they are not independent, the code words form only the smaller subset of the all possible 2 power n vectors. Clearly, the vectors the elements of r are not independent, so you can treat them independently and slice happily symbol by symbol. You have to view it as one big block and decode to the possible code word that is the optimal thing to do.

On the other hand if you decide my complexity is too high, I cannot do all of that one choice is hard decision decoder, then you are back in the same old picture as before the syndrome decoding would very simply do come back to bits and you are happy. So, what we will see next is what to do if you do not want to do hard decision decoding, so anything that is not hard decision decoding is called soft decision decoding.

(Refer Slide Time: 01:03:37)



Soft-decision decoding

- modern idea "in an implementation sense"

$$\begin{tikzcd} \arrow[r, "\underline{c}"', "n \text{ bits}"] & \boxed{\text{BPSK}} \arrow[r, "\underline{s}"] & \oplus \arrow[r, "\underline{r}"] & \boxed{\text{Decoder}} \arrow[r, "\hat{\underline{c}}"] & {} \\ & & \underline{z} \sim N(0,\sigma^2) \text{ iid} \arrow[u] & & \end{tikzcd}$$

$0 \rightarrow +1$

$1 \rightarrow -1$

(n, k) code
$\downarrow$
all codewords equally likely a priori

Optimal decoder

Maximum likelihood decoder

$$\underline{\hat{c}} = \max_{\underline{u} \in \text{code}} \Pr(\underline{r} \mid \underline{c} = \underline{u})$$

- Minimizes $\Pr(\underline{\hat{c}} \neq \underline{c}) \leftarrow$ block (or) frame error rate

All of you know enough of coding already no need to read, this I should remind you it is kind of a modern idea its being around for all the time all over the place, but in a implementation sense it is definitely a modern idea. So, I will come later what I mean by this later on, so as we go along we will see what I mean, but soft decision decoders have been around since ages you know they have been around since seventies definitely what knows as a the b decoder is really a soft decision decoder.

They have been around for a long time, but they have been kind of resurrected recently and so much studied and so much more detail and several sub optimal soft decision decoders are out there today, which work really well in practice. These are the engineering solution soft decision decoding is pretty much taken over today, so many of the new devices new implementations.

You would have will be soft decision decoders and on the other hand I should also comment that soft decoding of algebraic codes is really hard its interesting not so easy to do soft decoding of algebraic codes there are algorithms. There are definitely very good algorithms that give you good soft decoders, but it is still kind of non trivial to come up with a very nice algorithm which works well in practice you remember in practice. We are going to look at field which are as large as 2048 and rates that are large usually 8.9.

In those cases, most of the soft decision decoders known available out there for algebraic codes do not work that way. So, one of the greatest openings open problems in coding is still to my best of my knowledge remains fairly open is coming up with good soft decision decoders. For large algebraic codes of high rate large feel algebraic codes with high rate that is kind of open problem. I see people are writing down the problem it is unlikely that it will be solved in future or may be people. Actually I should say today in coding people are not interested in their problem the reason is there are other codes which are not so in a way to speak algebraically structured as Reed Solomon codes and all that.

They have good soft decision decoders and they have taken over pretty much in an engineering sense. So, that is why today only the modern idea of is popular if you say you are working on a soft decision decoding of Reed Solomon codes you give an talk on a conference there might be two people sitting and listening to you. So, it is not a very popular area and people do not work on it , but I think it is an interesting problem any way so, let us see what to do if you cannot do hard decision decoding and that is soft decision decoding. So, let me write that down little bit once again so that you see what i mean by this.

So, remember I am going to stick to binary codes and we will start by example we will see very simple examples of how to do soft decision decoding we will get the idea and then we will see more generic structure for it. So, what is the overall picture again remember i have a code word. So, I am going to start with code words the mapping to bits message bits its king of trivial. In a way suppose you have NK code all codes are going to be binary from now on unless some huge exception all codes will be binary from now on.

So, let us just say n k code this is going to go through BPSK, BPSK will always be 0 going to plus 1, 1 going to minus 1 and then a WGM. So, the symbol vector s AWGZ which is always going to be normal which means 0 variance sigma square i i d. This gives you r, I want to have a decoder here which will produce c hat. So, this is kind of similar to the problem for binary symmetric channel, we also started like this for the binary symmetric channel and the we looked at some situations this is a very generic situation which comes under detection etcetera. So, people know what the best thing to do is it is not very surprising that people know.

It except that it may not be implementable in most cases , but set thing always known and the best thing is usually what is called maximum likely hood decoding. So, that is always the best possible decoder, so the optimal decoder is usually the maximum likely hood decoder. So, what does the maximum likely hood decoder do hopefully, if you remember this I derived it in sparely generally form early on in the class. I do not know, if you remember, so let me write down the c hat is you are going to maximize the probability of r given c equal to 2 over all you in the code.

So, this is what we are going to do I am writing probability, but clearly r is a continuous distribution so basically it will be some p d f i am simply writing probability to just to avoid all those clumsy notations. So, whenever you have a continuous distribution interpret p r as a p d f, so that is the that is the idea actually I should be care full here this is optimal only when all the code words are equally likely that will be something that we assume so all code words equally likely something we will assume a priori equally likely, but computing this expression is a daunting task.

So, let us try and compute it for simple cases, so I think the time is coming to an end, so I am going to stop in a little while, but we will try and compute this maximum likely hood decoder for some simple cases and then we will see it is a little bit non trivial to do it. So, clearly for large cases you can imagine why it is very difficult, how many computations do you have to do 2 power k 2 power k huge.

You have 2 power k code words and if k becomes like 5000 then you cannot do this. So, it is very difficult to do this computation you might think there is a way to simplify this. There are ways to simplify it for BPSK, I will show you some simplifications, but even then the number of computations you have to do is really large.

So, you cannot simplify it much beyond the difficult part, so there is also another way in which you can define optimality. So, this is optimal for what this minimizes what probability of what which error minimizes probability. That is very important minimizes the probability that c hat is not equal to c the entire code word. So, this optimal according to what is called as block error rate, this is called block error rate or frame error rate, so it is also possible to define optimality with respect to bit error rate. That turns out to be much more interesting in the modern sense, so to speak sop that s also this we will stop here for this lecture after a break we will pick up from here.