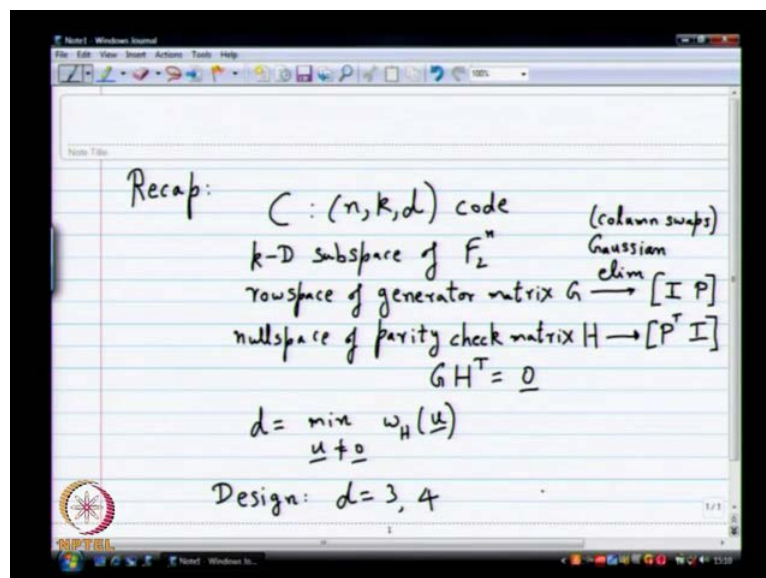


**Coding Theory**  
**Prof. Dr. Andrew Thangaraj**  
**Department of Electronics and Communication Engineering**  
**Indian Institute of Technology, Madras**

**Lecture - 7**  
**Optimal Decoders**

So, once again we will begin with the recap, this is kind of like a summary and an important summary, because we are going to move to some other part of the, part of the control coding in this class.

(Refer Slide Time: 00:28)



So, let me just quickly summarize this, so suppose you have  $C$  being  $n, k, d$  code, this we are thinking of this as a  $k$  dimensional subspace of, well we being saying  $F_2^n$  know. So, let me just write  $F_2^n$  and I pointed out this that vector space is a bit different has a inner product which is little bit different and has only a finite number of elements, so lot of restrictions come aboard because of that. So, usually we use row space description of  $C$ ,  $C$  is described as the row space of, of a generator matrix which is usually denoted as  $G$  and we think of  $G$  in systematic form, if it is not in the systematic form what can you do? You can do Gaussian elimination and let us say column swaps, if you do these two, you will end up in the form which looks like this  $[I P]$ .

So, now there is a also null space description of  $C$ , null space of a parity check matrix which is usually denoted  $H$  and we know systematic form  $H$  is going to be  $P^T I$ .

so the dimensions are crucial here dimension of  $G$  is  $k$  cross  $n$ , dimension of  $H$  is  $n$  minus  $k$  cross  $n$ ,  $k$  is a dimension that rank of  $G$  and all interesting and all that, so hopefully that a remember. So, interesting properties for instance  $G H$  transpose is a  $0$  matrix could be what dimension?  $k$  cross  $n$  by  $n$  minus  $k$ , so that will be the dimension of the  $0$  matrix.

So, case is all about the  $n$  and  $k$  do not think I want to add anything else and then  $d$ ,  $d$  is the minimum hamming weight of the non zero code word, that is the minimum distance why is this called as a minimum distance? If you imagine all your code words as living in the space  $F_2^n$ , any two code words are separated by at least  $d$  flips,  $d$  bit flips, you have flip at least the  $d$  flip bit to go from code word to the other, that is the idea. This is the hamming weight and this the hamming distance, also define between the two vectors case.

Now, is a good time to ask questions or some commons about, about any of this, so we also saw relationship between  $d$  and the parity check matrix, it was the minimum number of columns of  $H$  that add to  $0$ , if that was useful characterization for designing quality check matrixes with a given minimum distance. So, we saw that for  $d$  from the design point of the view,  $d$  equal  $3$  and  $d$  equals  $4$  as straight forward and simple. Then we saw some bounce, three types of bounce three bounce, hamming bounce, singleton bounce and Gilbert version of bounce and we saw that gives you the indication of how good your  $d$  is for a given  $n$   $n$   $k$  or you can ask any other questions, for a given  $n$   $n$   $d$  how good your  $k$  is? So, given  $k$   $n$   $d$  how small your  $n$  is?

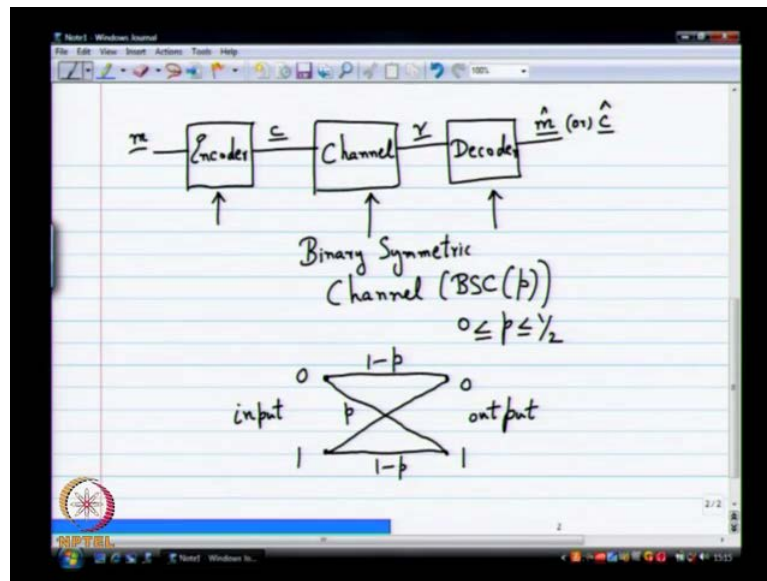
Those kinds of questions are answers by these bounce. So, we will see as we along the  $d$  is equals to five onward becomes a non tribune, so you have to use some interesting construction, the most popular construction uses finite fields and  $d$   $c$   $h$  codes of all that codes, etcetera, so that is a most popular construction, we will see that as we go along, so we need to learn about finite fields before we go there.

So, that is one path you will pick up from there soon enough, but before we go there we have to revisit, not revisit actually visit two other blocks in the block diagram that I drew in the very first class. If you remember the block diagram I drew on the very first class, we had an encoder, we had a channel, then we had a decoder, right? These are the three blocks that we had, so you have a message  $m$  which gets encoded into a code word  $c$  and

a channel produces some output and the decoder had to produce the estimate of the message, so you can call it  $\hat{m}$  or it can even be  $\hat{c}$ .

So, since we are thinking of encoder as one to one mapping you can either produce  $\hat{c}$  the decoder or  $\hat{m}$  of the decoder, you can go to the one to the other without any problem. For instance if your code is encoding a systematic, then even you produce a  $\hat{c}$ ,  $\hat{m}$  is simply the first  $k$  bits of  $\hat{c}$ , so it is not a problem.

(Refer Slide Time: 06:26)



So, this block we have seen a little bit, we are going to see this two blocks briefly in this lecture. So, we will see one model which is called the binary symmetric channel, so very popular and simple model abbreviated as B S C, it is parameterize by one parameter  $p$  which is usually between, which is always between 0 and half, so this is the channel that we will see, I will describe it very briefly, it is easy to describe and then we will also see that decoded. What is the best possible decoder that one can design for this channel, so we will see that, so let us see what is the B S C of  $p$  it is very easy?

So, it takes, so its binary, so it takes binary input two possible inputs 0 and 1 and there are two possible output 0 and 1, 1 once again, so its binary input, binary output. So, this is the input this is the output and it is a channel, so it will do something probabilistically, if you send a 0 as input in this channel, it is going to put out the 0 with probability  $1 - p$ , that is where the  $p$  comes in,  $1 - p$  and it will flip it or make a transition or an error to the probability  $p$ . So, that is the idea of the 0 and the channel is symmetric, so

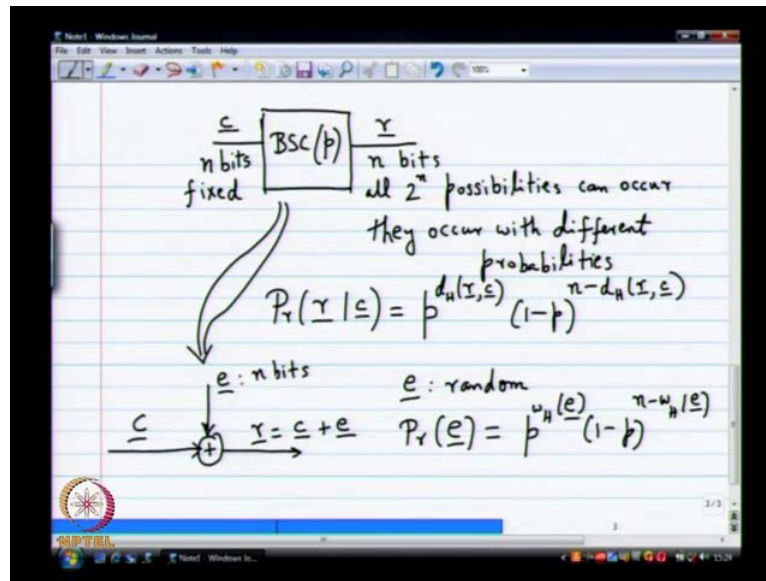
you can imagine  $y$  is said to be symmetric, so whether you send a 0 or a 1 a similar thing happens, symmetry is not a same, similar thing happens.

So, if you send a 1 you are going to receive a 1 with probability  $1 - p$ , if you send a 0 it is going to be flipped or error is going to be introduced or the transition is going to happen, probability  $p$  which is the same as what is there for 1. So, this is the binary symmetric channel it is very simple, it is very elegant and simple channel, it is elegant, because of the simplicity and also, because of its wide applicability. Almost any digital communication system that you might see at some level can be modeled as the binary symmetric channel roughly, if you did your design carefully enough it will become a binary symmetric channel. You putting a bit in and the bit comes out the other side of the channel, its either the same bit what was transmitted or it gets flipped with some probability, so it is very simple model can be used.

Another assumptions that is implicit in when I write down a BSC of  $p$  is that successive bits go through this channel independently, that also implied when I write it, I am not saying depend between the successive bits, but hopefully when I, it is good to make it explicit. So, if one bit goes through, second bit also goes through, another instances of the channel independent of the first one. So, this is the model we will use for most of the course, so more than maybe half of the course we will use this model then we will change to a little bit of more realistic model.

So, this model was very popular let say tell the eighties or early eighties, then the other more interesting models took over from there, so we will start with this model at least. So, under this model what can I say comes out of the channel? It is going to be binary, it is going to be binary vector, so I can denote that binary vector let us say  $r$ . So, now I have a nice complete block diagram where the output some are for everything, in case previously I did not know what to put at the output channel. Now, I am saying the output of the channel is  $r$  and then the decoder works on it we get a  $\hat{m}$  of a  $\hat{c}$ .

(Refer Slide Time: 10:49)



So, let us see how to describe this a little bit more precisely, so you have a code word  $c$  going into a binary symmetric channel with transition probability  $p$  and you get an  $r$ , in case this is the, this is the model for the channel. By the way the quick question why did I say  $p$  is restricted between 0 and half, why can be it greater than half? You can think about it if it is not clear to you and if it is greater than half you can re-label your 0 and 1 at the other side and you will still get back your  $p$  less than half, you simply call 0 as 1 and 1 as 0, it is just what you choose to call it, right?

It is not anything, there is no major ((Refer Time: 11:41)), so you flip that and you get back  $p$  is less than half, so it is enough to look at the  $p$  between 0 and half. Now, let us see how this picture looks, so it is important to realize that when, when you do encoding the code word that is going to go in is in a way random. So, you do not know ahead of time what code word you are going to put into the binary symmetric channel, so of course the  $c$  is also random.

So, let just not worry some reason this thing coming up, now it is gone hopefully it will stay gone, so my hand is doing something, it cannot take, let us see. So, anyway, so let us fix for now that  $c$  is some constant vector,  $c$  is some known vector that is transmitted. Let us say the all zero codeword, if you want to be very specific, some vector that is being transmitted, what are the possible  $r$ 's that they can receive on the other side? All to for  $n$  possibilities, so this is  $n$  bits and let say this is fixed, what happens here is, you definitely

get  $n$  bits, but all  $2^n$  possibilities can occur, yeah, so of course, they all occur with different probability, that is the important thing.

So, they all occur with some positive probability, but not all with equal, if all are in equal probability then cannot do much. So, may as well do not do anything, so that yeah unless  $p$  is half for instance anyway so does not matter. All  $2^n$  possibilities can occur, but they occur with different probability, so that is important, so they occur with different probability. So, it seems like a very simple thing right now, but I think you should try and understand exactly how it works. So, the next question I am going to ask is how will you characterize that probability?

So, you have to be able to write that down, so suppose I want to write probability of  $r$  given  $c$ , so this is what I am talking about here. So, notice here this is very bad notation in terms of probability, you cannot write the probability for some general vectors, wherever you need random variables you have to imagine there are random variables, wherever you need constants you have to imagine they must constants. So, it is a good notation it is very intuitive, but is not very precise probability notation, so we will do such notation throughout this course, if someone assumes you comfortable enough and probability to understand what it means when I write probability. So, given that the transmitted code word is fixed at  $c$ , it goes to a binary symmetric channel, what is the probability that you will get a vector  $r$  has the output of the channel?

So, it should depend on what? It should depend on  $p$ , it is the first question I want you to be convinced about, of course it depends on  $p$  and also depends on how many bits has to flipped to go from  $c$  to  $r$ . See  $r$  is a constant vector  $c$  is a constant vector,  $r$  and  $c$  will agree on the certain positions, they will disagree on the certain positions, wherever they agree the channel should not make an error, right? Channel should send a probability what  $1 - p$ , wherever they do not agree the channel has to make an error, only then you will get the  $r$  and that happens with the probability  $p$ .

So, you can argue and since successive bits go through independent version that is how we are imagining, then you can all multiply all these things together. So, if you do that calculation in your head you will finally get this answer. So, it will be  $p^{\text{hamming distance between } r \text{ and } c}$  and then  $(1 - p)^{n - \text{hamming distance of } r \text{ and } c}$ . So, this is a simple enough way of characterizing. So, there is

another very equivalent way of thinking of what this channel does, so that is by this model, so what people do is they look at this guide and write it equivalent form.

They say what is actually happening is  $c$  is going in some error vector gets added to  $c$  and what you are getting out is  $r$  which is the sum of  $c$  and  $e$ , once again when I mean sum that is always modular 2, so the  $x$   $r$  of  $c$   $n$ . Now, this is a very equivalent way of viewing of this picture, so here you are viewing bit by bit, here you are viewing the whole vector together as 1, so now what is  $e$ ?  $e$  is a error,  $e$  is a binary vector,  $n$  bit binary vector and so that is the thing, again  $e$  is very similar to this bits, so  $e$  is  $n$  bit, so what are the various possibilities?

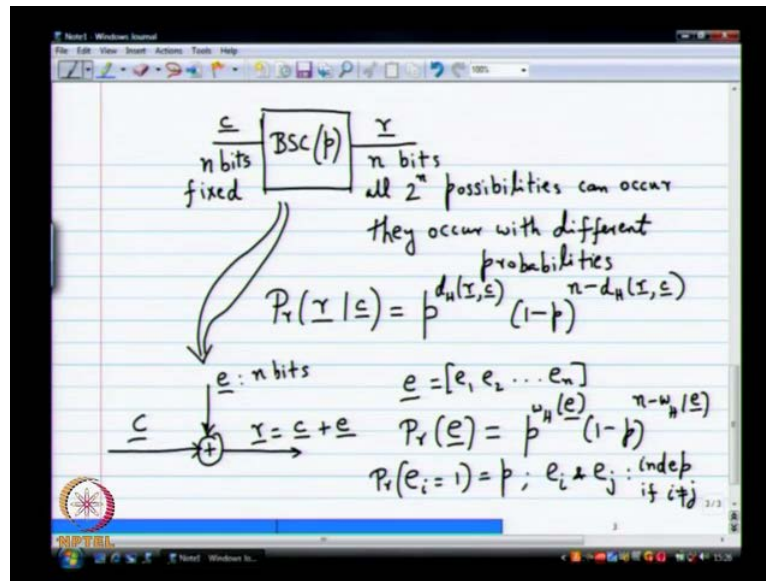
All  $2^n$  possibilities happen and they all happen with equal probability or not in general it depends on  $p$   $p$  is half and they all happen with equal probability or not, different probability in general, depends on  $p$ ,  $p$  is half, then they all occur with equal probability. But usually  $p$  won't be half, if  $p$  is half the situation is really bad, you cannot do anything, so usually  $p$  will be very small number, 0.01, point o 1 that kind of thing.

So, some vectors will be much more probable than others, so in this equivalent picture, so let me just go and fix this, because I do not know why is something is just altering all the time I have no idea, so anyway hopefully it would not come again. So, in this picture  $e$  becomes a random vector, right?  $e$  is random, so we have ask the question what is the probability of  $e$ ? If I give you particular vector  $e$ , what will be its probability?

Yes you have to write it as  $p^{\text{hamming weight of } e} (1-p)^{n - \text{hamming weight of } e}$ , is that right? Yeah it seems simple enough, hopefully you are convinced of this. So, I can either describe the binary symmetric channel as in the bit by bit format, which case I will write  $p$  of  $r$  given  $c$  or I can think of it as an error vector being introduced which gets exhort with  $r$ , both of them are perfectly equivalent and in one. I get very similar expression except that  $e$  itself becomes random and then I have weight of  $p$ .

So, let me ask one question 0 less than  $p$  less than half, what is the most probable error vector  $e$ ? All 0 must be probable, why is that true? Because when  $p$  is between 0 and half,  $1-p$  is greater than  $p$ , probability for getting a 0 in  $e$  is greater than probability of getting 1 in  $e$ .

(Refer Slide Time: 20:00)

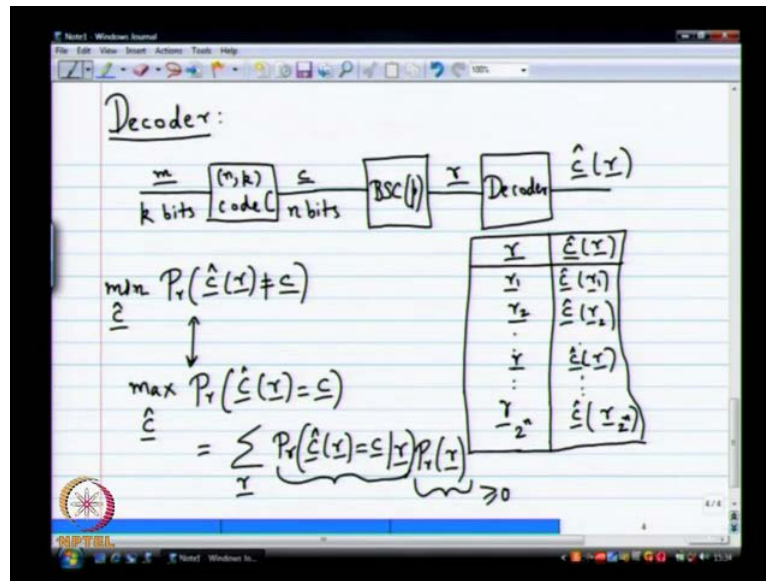


So, let me once again write this down little bit more in detail, so maybe instead of just saying  $e$  is random, I should say  $e$  is  $e_1, e_2, \dots, e_n$ , probability that  $e_i$  equal to 1 is  $p$ , what is probability that  $e_i$  is equal to 0?  $1 - p$  and  $e_i$  and  $e_j$  are independent if  $i$  is not  $j$ , so that is the idea. So, this is a the way to really see the relationship, you see everything works out very cleanly this way, so think of the error vector  $e$  as  $n$  bits each bit is 0 with probability  $1 - p$  and 1 with probability  $p$ . So, if the  $i$ th bit was 1 then it meant, then it means that  $i$ th bit of  $c$  got flipped on the channel, otherwise it does not, so it is very simple ideas here.

So, like I said the most probable error vector  $e$  is the all zero vector, after the all zero vector, what is the most probable? Vectors of weight one, error vectors of weight one are the next most probable, after that error vectors of weight two, etcetera, etcetera. That makes sense in the physical channel, if you design the channel well, most cases it would not make errors, if at all it makes error most likely it will make errors with weight, 1, etcetera, etcetera, so that is the way to think about it. So, that is channel will stop with the channel for now, then we will move on to the decoder.



(Refer Slide Time: 21:58)



So, let me draw a picture first, have  $m$  which as  $k$  bits, then it is encoded by  $n$   $k$  code and write  $n$   $k$  code here, let us say  $n$   $k$  code  $C$ , you get a code word  $c$  which is  $n$  bits and goes through a  $BSC$  parameterize by  $p$  and we get a received vector  $r$ , now we have a decoder here. I will imagine that the decoder is going to produce  $\hat{c}$ ,  $\hat{c}$  is an estimated code word  $c$  just equivalent to producing  $\hat{m}$ . So, we will produce a  $\hat{c}$  just because it is just easier had goes to a channel,  $c$  goes to a channel you get a  $r$ , it is easy enough to get  $\hat{c}$  first, that is the, that is the way to think forward. So, what is this decoder? This decoder right now will simply think of it as a big table, it is a huge table, what does the table contain?

First column is  $r$  and the next column is  $\hat{c}$  of  $r$ , so the output is clearly  $\hat{c}$  of  $r$ , so it looks like a receive vector and puts out an output of  $\hat{c}$ . It has a huge table, so table is basically has  $r_1, r_2, \dots$  so until what?  $r_{2^n}$ . It has all possible error like code words listed and so each of those things it would do a  $\hat{c}$  of  $r_1$ , there will be a vector here which will be a  $\hat{c}$  of  $r_1$  so on,  $\hat{c}$  of  $r_2$ , so on.

So, this is this big ugly table, just truly an ugly table, so that is the table. So, the first question is very easy to answer is on the left hand table I have  $2^n$  possibilities, all the  $2^n$  possibilities like that, what about the right hand side of the table? So, it will only be from the code, we should never output anything which is not from the code, does not make difference. So, you know the definitely a code word was transmitted, that

much is a guaranteed, so definitely the output should be the code word. But this is only  $2^k$  possibilities, but how many of them are there?  $2^n$ , so clearly several entries will occur several times, so that will happen cannot be, cannot be avoided it that is how the table will be.

So, the question first is what is the optimal way to build this table, for this channel? That is the first question we want to ask and answer, then we will go on and see maybe there are some optimal ways also and we will answer that question. First thing is what is the optimal way to do it? So, optimal way to do it with minimize probability of the error, minimize over  $\hat{c}$  will simply say  $\hat{c}$ . But remember  $\hat{c}$  is a function, so  $\hat{c}$  represents this entire table, so I have to minimize over a  $\hat{c}$  probability that  $\hat{c}(r)$  is not equal to  $c$ , this is what I want to minimize, am I right? Minimize the probability of the error, so what should be the optimal choose?

When I say optimal I am minimizing or maximizing something, I am optimizing something, the thing I want to optimize is probably of the error, seems like reasonable thing to optimize and what I am optimizing over? Various choices for this table, you might have seen this before and some form or the other, but I think is good to see again, because it is little bit of tricky issue here to convince yourself that is optimum. So, one thing that is this perfectly equivalent to is to say I want to maximize  $\hat{c}(r)$  probability that  $\hat{c}(r)$  equals  $c$ . Now, comes the crucial step, I know when I thought 3 5 6 to emphasize this crucial step, what is the crucial step next?

The point is you do not try to find the entire table in one step, you will always get confused, you will always get confused, you will never go anywhere, and so what should you do? You should attack the table one row at the time, you should given, it should split this up first, given  $r$  multiplied all probability of  $r$  and then you can make progress, you can attack one row at a time, so how do you attack one row at the time? So, I can write this probability expression as  $\sum_r \text{probability that } \hat{c}(r) \text{ equals } c \text{ given } r \text{ times what?}$

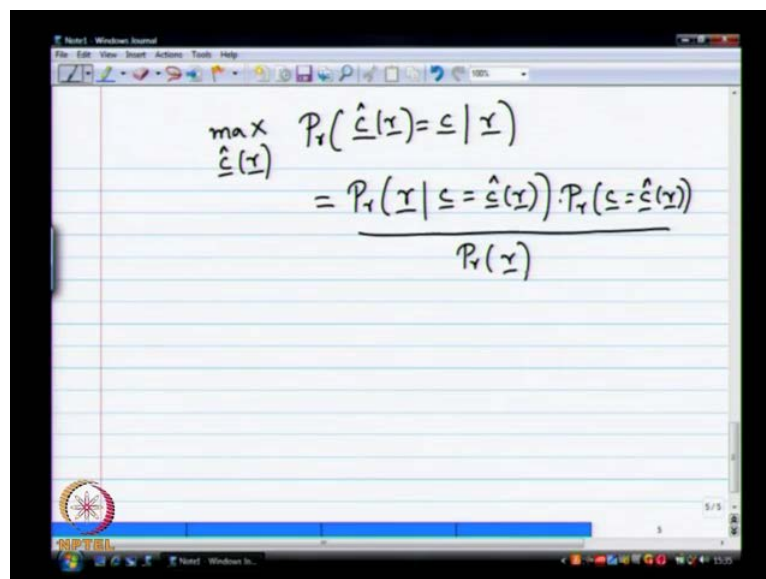
Probability of, so now I am in some  $r$  throw, so  $r$  throw which gives me the  $\hat{c}(r)$  particular row, so there are only part  $n$  terms in the submission, I am saying only  $2^n$ , but then can be a very large number, but it is got  $2^n$  in this. So, once you write it like this condition on  $r$  attack, trying to attack row by row you will notice something

interesting. So, what do you notice is probability of  $r$  whatever it is, is positive whatever your choice of  $\hat{c}$  is probability of  $r$  will never change, because  $\hat{c}$  happens after  $r$  has being received,  $r$  depends on the channel.

So, this term is independent of your optimization and it is non-negative, most cases it is going to be positive. So, if you are maximizing this sum, it as good as maximizing each of these terms that is the next step which makes going row by row possible. Because this term is positive, optimizing over, optimizing the decoder for maximizing the entire sum is exactly equivalent to maximizing this other term is that is inside each of these submissions.

So, the way to imagine argue this is you can do this in several ways, one way is to say, let me not go there, we will just leave it like this, it is easy to see from this point itself, we have not seen before thing about the for awhile you will convince yourself. Each of these terms has to be maximize the optimal  $\hat{c}$ , if it is not, then you go to the particular row change the decision and find the new  $\hat{c}$  which will have a strictly higher value for this object of function. This is a very simple argument to see that for each, for the optimal  $\hat{c}$  will also optimize each of these terms, so the easy way to see these, once you see this.

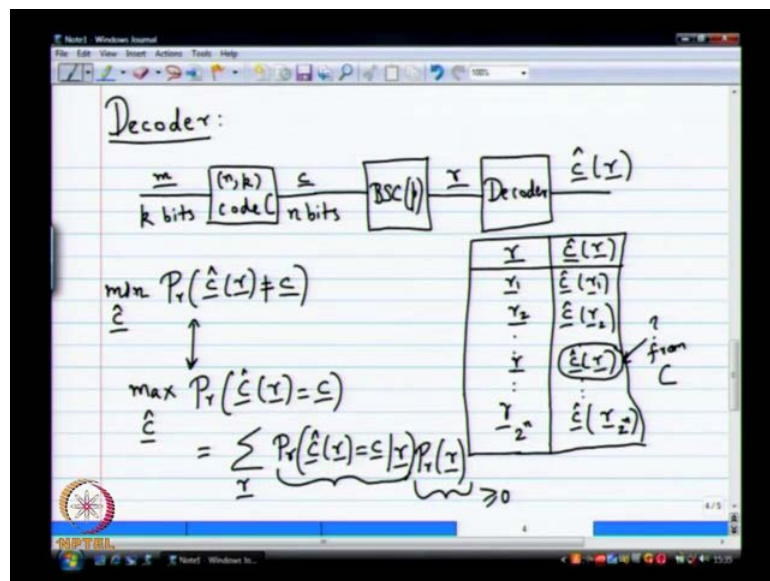
(Refer Slide Time: 29:10)


$$\begin{aligned} \max_{\hat{c}(r)} P_r(\hat{c}(r) = c | r) \\ = \frac{P_r(r | c = \hat{c}(r)) \cdot P_r(c = \hat{c}(r))}{P_r(r)} \end{aligned}$$

Now, I am simply looking at a particular row, so all I want to do is maximize probability that  $\hat{c}$  of  $r$  equal  $c$  given  $r$ . So, now I do not have to write  $\hat{c}$  here, I will simply

write  $\hat{c}$  of  $r$ , so this is only job they have to do, I am at the particular row I have to pick the best  $\hat{c}$  corresponding to that particular  $r$  that was received. Now, I use this formula to turn this formula around, here is how we are going to turn this formula around, probability of  $r$  given  $c$  equals  $\hat{c}$  of  $r$  or  $\hat{c}$  equals whatever write it any which way you want times probability that  $c$  equals  $\hat{c}$  of  $r$  divided by probability of  $r$ . So, once again if you want to go back to this table and figure out what I am trying to do?

(Refer Slide Time: 30:17)



I am trying to find out what this vector should be? What should this be? What are the possible values for that vector? They are from  $c$ , so this vector is from the code  $c$ , you cannot take from outside of that, you have only  $2^k$  possibilities of that, you are going to try all  $2^k$  possibilities and try and see which one of them maximizes this expression you have.

(Refer Slide Time: 30:48)

$$\begin{aligned} \max_{\hat{c}(r)} & \Pr(\hat{c}(r) = c | r) \\ & = \Pr(r | c = \hat{c}(r)) \cdot \Pr(c = \hat{c}(r)) \\ & \quad \text{indep. of } \hat{c} \end{aligned}$$

Uniform

$$\max_{\hat{c}(r)} \Pr(r | c = \hat{c}(r))$$

$$\hat{c}(r) = \arg \max_{u \in C} \Pr(r | c = u)$$

Now, we are moved from maximization over the entire decoder to maximizing over only choice of one code word for the particular row. So, now let us come back and see here, as you keep changing  $\hat{c}$  of  $r$  what will happen to this probability? What is the probability? So, probability that, at the transmitter  $c$  equals a particular code word, that is the same as the probated distribution on the message itself and usually we will assume that this is uniform or equally likely. We will say at the transmitter the codeword can be any one of the  $2^k$  possibilities with same probability, so that is an assumption we will make, so if you do not assume that only something changes because it's a bit of a pain, so we simply assume that.

The next question is what about the denominator here? It is independent of  $\hat{c}$ , comes much before  $\hat{c}$ , so what you can do under these two assumptions is, drop the second term in the numerator and drop the denominator in your maximization without losing any optimality. So, the same thing what you have to do is max over  $\hat{c}$  of  $r$  probability of  $r$  given  $c$  equals  $\hat{c}$  of  $r$ . So, this is what I have to do to find the code word corresponding to particular receive vector  $r$ , I have compute this expression for all possible code words and pick that code word which gave me the maximum possible value, is that all right? So, this is kind of ugly way of writing it, another nicer way of writing is to say  $\hat{c}$  of  $r$  equals argument of the maximization of  $u$  and  $c$  probability that of  $r$  given  $c$  is  $u$ , this is a kind clean notation for the same thing.

So, next little bit more elegant to look at, I vary  $u$  over all code words in the code and compute probability of the particular  $r$ , remember  $r$  is fixed now, I am at the particular row  $r$  given  $c$  equals  $u$ , then pick the argument that gave me the maximum probability, that will give me the  $\hat{c}$  of  $r$ , so this is, this is an optimal choice. Now, for the B S C what is this expression? We wrote that just now, just now we wrote that down, so let me just use that.

(Refer Slide Time: 33:45)

The image shows a whiteboard with handwritten mathematical derivations. On the left, a diagram shows a set of points  $\mathcal{C}$  in a space  $\mathbb{F}_2^n$ , with a vector  $r$  and its corresponding Hamming distance  $d_H(r, u)$  to a point  $u$ . The main derivation is as follows:

$$\hat{c}(r) = \arg \max_{u \in \mathcal{C}} p^{d_H(r, u)} (1-p)^{n-d_H(r, u)}$$

$$= \left(\frac{p}{1-p}\right)^{d_H(r, u)} (1-p)^n$$

$$= \arg \max_{u \in \mathcal{C}} \left(\frac{p}{1-p}\right)^{d_H(r, u)}$$

Since  $\frac{p}{1-p} < 1$ , the expression is equivalent to:

$$\hat{c}(r) = \arg \min_{u \in \mathcal{C}} d_H(r, u)$$

So, we have for  $\hat{c}$  of  $r$  argument maximization  $u$  and  $\mathcal{C}$  probability of  $r$  given  $u$  basically,  $r$  given you and that is nothing but  $p$  to the power hamming distance between  $r$  and  $u$  times  $1$  minus  $p$  to the power  $n$  minus having distance between  $r$  and  $u$ . So, now I can do little bit more of a minimization here, so I can say here  $p$  by  $1$  minus  $p$  raise to the power hamming distance  $r$  and  $u$  times, what?  $1$  minus  $p$  the power  $n$ . Now, this  $1$  minus  $p$  to the power  $n$  has no role to play in the maximization, so does not change if I change  $u$ , right? For a given  $r$  if I change  $u$  that will not change, so you might as well drop it from your maximization, so now we have a equal and maximization.

So, I have  $p$  by  $1$  minus  $p$  whole raise to the power  $d_H$  of  $r$  comma  $u$ . So, one can show without too much work the following result, I will write down the result and then I will say why it is true. So, it is not too difficult to do this, you can show this is same as argument, argument of the minimization over  $u$  and  $\mathcal{C}$ , the hamming distance  $r$  comma  $u$ . Ok yeah this is because  $p$  by  $1$  minus  $p$  is less than  $1$ ,  $p$  is less than half, so  $p$  is smaller

than  $p$  by  $1 - p$ , so  $p$  by  $1 - p$  will be less than 1, right?  $p$  is less than half, so  $1 - p$  is larger, so if you do  $p$  by  $1 - p$  you will get something which is less than 1, so if you want, this is less than 1, say something like 0.5, I have 0.5 to the power  $x$  over all possible  $x$  which will be the smallest whenever  $x$  is smallest, which will be largest I am sorry, it will be a largest whenever  $x$  smallest, because if we increase  $x$  what will happen?

It will only become smaller, so that getting multiplied more and more by 0.5, so it is as good as minimizing the exponent, you can take log and show that  $\log p$  by  $1 - p$  will be negative, so maximization becomes minimization, that is the another way of doing it. If you take a log what will happen here?  $d H$  of  $r$  comma  $u \log p$  by  $1 - p$ ,  $p$  by  $1 - p$  is less than 1, so log of that is negative, so maximization becomes minimization for  $d H$  of  $r$ , looks another way of writing it down if you like it, so this is  $\hat{c}$  of  $r$ , if the display driver recovered something very bad.

So, this is formula which is worth mugging up and I will do a pictorial representation here which will, which some of you appreciate more, so you have  $F^{2n}$  and you have your code words, the code words are the stars and you have  $F^{2n}$  and you have a particular vector which you have received, so this is your  $r$ , maybe I should put an arrow and draw it out, just to make it little bit more clear. Suppose, this is your received vector  $r$ , you are in this table we are building this decoder table, you have a received a particular vector  $r$ , how will you choose the code word corresponding to  $r$ ? You have to do this minimization, what is this minimization pictorially?

It is very simple, you look at the hamming distance from each of the code words, each of the stars that you have here, compute all of them and then find which is the smallest and simply say that is your  $\hat{c}$  of  $r$ . So, this is how you build the optimal decoder table, the decoder for the, the table for the best possible decoder, it will minimize probability of error, no other decoder can give you smaller probability of the error, how do you build the table? You go to each and every  $r$ , all the  $r$  is possible, right?

So, you go to each and every  $r$  and then look for the closest codeword in hamming distance, simply say that is my estimate of the transmitted code, so that how you build a table, very intuitive, right? I could have just drawn this picture and all of you would have accepted it without any problems, but it is good to know that no other decoder can give

you smaller probability of error. So, wondering maybe I can do something more here, so just does not work, this is only one which will you give the best possible probability of error.

(Refer Slide Time: 39:26)

Ex. ① (3, 1, 3) repetition code  
 $= \{000, 111\}$

$r$	$\hat{c}$
000	000
001	000
010	000
100	000
011	111
101	111
110	111
111	111

$n: \text{odd}$   
 $(n, 1, n)$

Let us see some quick examples assuming the display driver recovers. So, let us here are couple of examples, first example we will see is the 3, 1, 3 repetition code, so what is the 3, 1, 3 repetition code? Yeah it is very easy 0 0 0, 1 1 1, so of course the problem that I am asking you to do is to make the table for the optimal decoder, what is the optimal decoder basically? The best way to describe it is to make the table in this case you can easily make the table, how many possible received vectors are there? Eight possibilities and you have to see which of them are closed too, so let us just quickly do that, not very difficult.

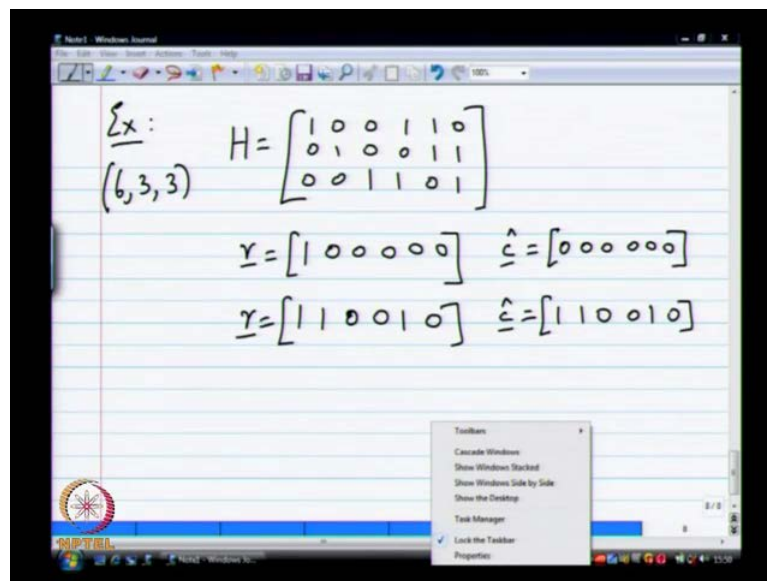
So, it is ok to draw the  $\hat{c}$  of  $r$ , I just wrote that down in the previous derivation for simplicity, for being explicit, you can see that that the  $\hat{c}$  is good enough, so that we can write it down 0 0 0, 0 0 1, 0 1 0, 1 0 0, I am writing it in a slightly different order, you can see why I am writing it in a different order. So, you will see that the closest code words for all these guys will be 0 0 0, closest code words for all these will be 1 1 1, all right? So, if you are to implement the decoder for the 3 1 3 repetition code over a binary symmetric channel the best possible decoder, how will you do it? You can implement the



stable at the receiver, so there might be much easier ways of implementing this table, not saying this is a only way to implement this table.

You do not have to have memory eight and all that, what would you do, what would be another way of doing it? Simply have 2 bit adder and see if the adder is 1 or 0 or 2 or 3 it is the first bit of the order goes to 0, then we just say, it is very easy, right? So, you know digital system how to implement this table, so it is very easy, you require very few operations you can do it. It is easily extendable to  $n$  comma 1 comma  $n$ , what will you do for the  $n$  comma 1 comma  $n$ ? Let say  $n$  is odd,  $n$  comma 1 comma  $n$ , what will your decoder look like? It simply count the number of one's, if it is less than  $n$  by 2, you decide it is 0 to all zero's, if it is greeter by  $n$  by 2 you decide it as all one's, so it is easy enough for this code, simple.

(Refer Slide Time: 42:45)



So, let me just try and do a slightly more complicated example, is to show it can become a little bit more painful. Suppose, I have a code  $H$  equals I think we are doing this code also quite a bit, so this is a, it is a 6 comma 6, 3, 3 code we agree, agree with all the three parameters, 6 is easy, 3 is easy because the first three columns will simply give you  $i$  and then what about  $d$ ? There is no reputation no all zero columns and then there are three columns that add to 0, so you get  $d$  equals 3.

So, now we have eight different code words how many different received vectors? 64 different code words, already the table has becoming a little bit unveiled, so you cannot

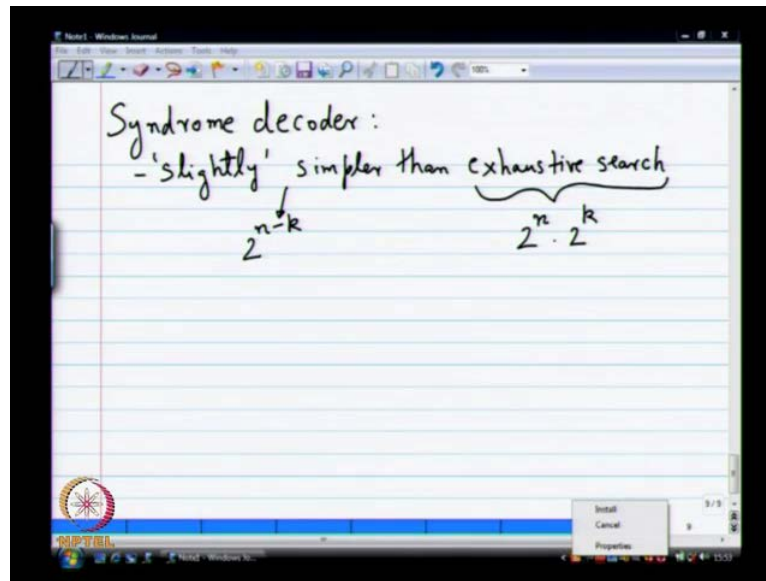
think simple logic by which you can implement it, so it seems like a little bit more complicated it does not seem that is easy. So, if I throughout some simple things you might be able to answer, but in general it is hard, so if I say, suppose I say  $r$  is 1 0 0 0 0, what will  $c$  have? Why should it be all zeros? Yeah that is the idea, so you know the minimum distance is 3 and all 0 is at distance 1 away from 1 0 0 0 0.

So, you cannot find anything which is closer than 1 away, if it is closer than 1 away what does it mean? Yeah itself it has to be a code word, so what about another  $r$ , let me give you another  $r$  which I think is easy. Yeah this is a code word, so it seems like its an interesting check to do is to compute whether it is a code word or not, so how do we compute, how do we check quickly that it is a code word? Do  $H$  times  $r$  transverse you get 0, so clearly it is having distance 0 from one code word and you can never get any other code word which is closer than 0.

So, you can easily conclude that, oh my God, why is it saying all kind of things, cancel I think something I am pressing or I am doing something, what I am not doing right click may be I am doing, so maybe I should disable the right clicking any way. So, quickly becomes a little bit more painful, so what we will see next, so what we will see next is for at least problems of this kind, so some 6 3 3 at least of this kind of a comparable size, how do you make this, this process a little bit simpler?

We will see eventually that it cannot be made simple, because people have shown that  $m$  decoding or best decoding for B S C is once again one of those  $n p$  complete problems. So, you cannot really do it very well at the end of the day, but we will see at least for small size problems that there is a better method than exhaustive search. So, table of size 64 is not needed, table of size 8 is good enough in this case, so it seems little bit surprising when you look at it, but it is very simple and intuitive idea which works out very well that is what we are going to see next and that is called a syndrome decoder.

(Refer Slide Time: 46:47)



So, its exact same decoder, but its implemented using something called the syndrome, so you call that syndrome decoder, what is the motivation? It is a slightly simpler, I am saying slightly, because it is not really all that simple, so it is (Refer Time: 47:04), but slightly simpler then exhaust to search can be in some cases. So, in what way see exhaustive search if you want to search through all the code words complexity is roughly you need  $2^k$  operations, of the order of  $2^k$ ,  $2^k$  times  $n$  if you want be very picky about it, but  $2^k$ , so that is the kind of operation you want.

So, it turns out  $2^{n-k}$  is usually, oh let me say I am sorry  $2^n$  sorry why am I,  $2^n$  is the is the just brute force way of building the table, right? You have to go through each col, each row and at each row is  $2^n$ ,  $2^n$  times  $2^k$  may be, because for each row you are going to compare with  $2^k$  different code words.

So, maybe you can say  $2^n$  into  $2^k$  if you do like that very rough count. It turns out overall you can get away with roughly about  $2^{n-k}$ , but once again  $2^{n-k}$  is also fairly large  $n$  is 1000 case 500, does not matter that is  $2^k$  or  $2^{n-k}$ , either way you cannot do this, so  $n-k$  also becomes very large. So, you cannot do much with this, but nevertheless it is an interesting idea which gives you several sub optimal decoders which can be implemented very easily.

So, to start with this it will seem like a very complicated idea, nevertheless it gives you a lot of sub optimal decoders which can be implemented. So, this idea of looking at the syndrome is suppose to looking at the entire received vector  $r$  is very powerful that gives you a lot of decoders. So, I will stop here I will little bit more time, but anyway the next lecture is 75 minutes. So, I will stop here.