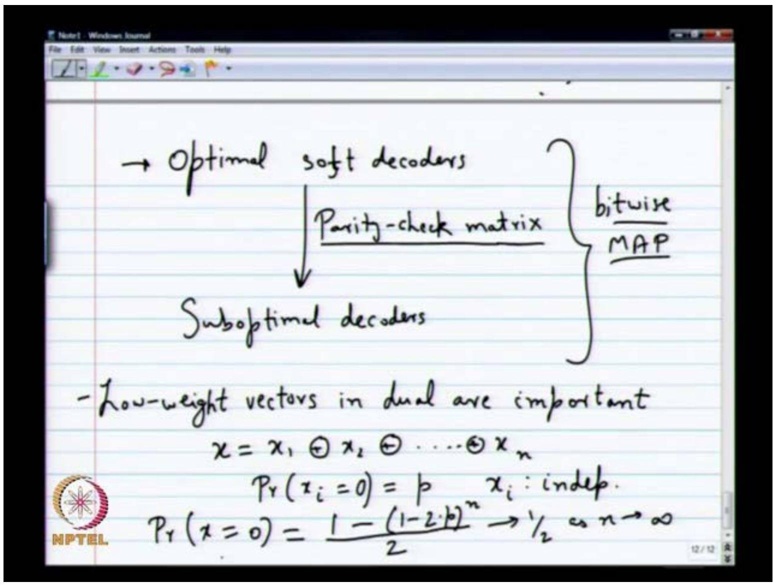
Coding Theory

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

Lecture - 21
Simulating Coded Modulation

(Refer Slide Time: 00:14)



→ Optimal soft decoders

$\downarrow$ Parity-check matrix

Suboptimal decoders

} bitwise MAP

- Low-weight vectors in dual are important

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

$$\Pr(x_i = 0) = p \qquad x_i : \text{indep.}$$

$$\Pr(x = 0) = \frac{1-(1-2p)^n}{2} \to 1/2 \text{ as } n \to \infty$$

So, let us continue. The one point I want emphasize about the, how the low weight vectors in the dual play a good role. So, I am not going to once again, it is a heuristic idea, but I want to emphasize that a little bit, because it is important, in what follows. We will use that later on. So, if you have an, if you have. So, basically low weight vectors in dual, are important. So, the reason, why this is true is, if you have, let us say, if it is just this, is again a heuristic justification. It is also valid very much in practice.

So, if you have x equals let us say x 1 or x 2 or so on, till x n. Probability that x i equals to 1, let us say x i equals to 0. Whatever these 2 things, x i equals to 0, is let us say some p. Some p and x i is an independent, if you have a situation like this, one can actually exactly compute the probability that x equals to 0. So, it will work out as something like 1 minus 1 minus 2 p to the power n by 2. So, this is another way of writing that expressions it is very, it is very easy to say.

As n becomes large, whatever the value of p is, what is going to happen to this probability? Yeah, this tends to half, as n tends to infinity. Of course, it is true, we going to take p to be less than a half, for instance when this works out. So, may be this 2 p minus 1, I do not know, one of those 2 expressions depending on which is greater, which is greater than half, less than half. So, anyway, so this is the, this is the kind of expression, you would get. So, what I am trying to say is, suppose the p, suppose p is very, very low, like 0.01, it means what? Means you know x 1 through x n very precisely.

I said very precisely, very good confidence, you know x 1 through x n. Probability that exercise 0 is 0.001 you think you know that. X1 is almost certain to be 1. You know that all this, but even if in that case, that extrinsic information, that all these exercises are giving about x can become very bad. As in what is, what is very bad? Now, if this extrinsic information evaluates to half, you might as well not use the extrinsic information. Even before that, I can tell you it is half, probability is half. So, extrinsic information being half, probability being half is very bad.

Likewise likely-hood ratio being 1 is very bad. L l r being 0 is very bad. All this things are not very good in formations. You do not have any good belief on that. So, if n becomes very, very large, if n becomes 10 and 20 all that. Not 10, 20 maybe it is not too bad, if n becomes 100, 200, 300 and all that, then this equation is definitely going to go to half. There is no point in using those large weight equations.

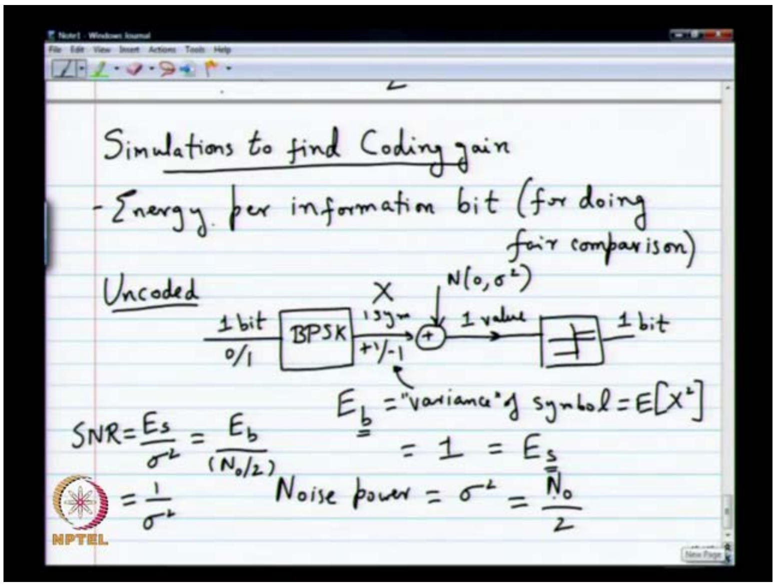
So, once it goes to half, what does it mean? The log likely-hood ratio from the extrinsic information is 0. So, whether you add it or not, what will happen? The final expression you calculate, whether you add it, you add this or not. Worry about it or not, absolutely no difference it makes, so you might as well not worry about this large length, large weight vectors in the dual. Those things you do not account for in your soft decoders. On the other hand, if the if there is very low weight vector in the dual, then it plays a very big role, because it will give you very good probabilities. And those you have to account for.

So, this principle is exploited, a lot in the soft decoders. In fact in the construction of course, you will exploit this principle. You will try to have codes, which in which this low weight vectors in the dual can be very easily accessed. So, you see that is the principle of very popular code called low density parity check. So, you use these kinds of ideas. So, this is, this is the idea while being heuristic, this plays a big role in practice. So, we should appreciate that.

So, hopefully this is clear. What happens, when you see too many things goes away. So, this is also crucial in your approximate decoder. So, you know which term to throw away, the term corresponding to the large weight vectors in the dual are not important. So, only the lower weight vectors play a big role. So, as you go along, what you do later on does not really matter. When initially you should take care of the low weight vectors very carefully, after that the larger weight vectors do whatever, add, subtract. Do not consider, may not make a big difference to you all final log likely-hood. All right, so this is clear.

So, one theme that have been following on all long is, this motion of sub optimal. And this motion of things, being very complex and complicated, what will happen when you look at sub optimal implementation with very complex computation is, analysis is pretty much impossible. So you cannot do precise analysis. So, you have to have some engineering way of doing the analysis. So, that engineering way of doing the analysis is, this idea of simulating and finding the coding gain.

(Refer Slide Time: 06:15)



Simulations to find Coding gain

- Energy per information bit (for doing fair comparison)

Uncoded

E_b = "variance" of symbol $= E[X^2]$

$= 1 = E_s$

$SNR = \frac{E_s}{\sigma^2} = \frac{E_b}{(N_0/2)}$

$= \frac{1}{\sigma^2}$

Noise power $= \sigma^2 = \frac{N_0}{2}$

Which is what I am going to describe now. Simulations and coding. To find, let us say to find coding gain. I have not described, what coding gain is. I will describe that soon enough and then will see how we can do simulations. From just a high level, how to do simulations, to fix this coding gain. It is a very simple idea, but nevertheless its important, because in practice today codes have become so complicated, the only way to find any, do any analysis, is to just stimulate. So, you should know how to go about doing that. That is why I think this is a good thing, to emphasize.

So, first of all, what is coding gain? So, to define that, one needs to be a little bit more careful. Little bit careful, you cannot just say something is the coding gain and let it go. So, the main idea here, in coding gain is the notion of a energy per bit. So, this is an important idea, per information bit. So, this helps in doing fair comparisons. So, this is for mainly doing fair comparisons. I think I pointed this out once before, when I was dealing with the repetition code. When you do not do any coding, all the energy you are spending is for information.

So, you are not spending any energy for anything, other than sending information, but when you do coding, you are going to send additional reddened parity bits, which are not information. You have to spend some energy there and you have to account for that energy in any comparison. You cannot just let it go. Then, it is not a fair comparison. I will show how to account for it in B P S K ((Refer Time: 07:57)) it is quite easy to do.

For other things also, but let us do that, because we are always looking at B P S K. So, in an un coded B P S K situation, what would you do? You would have 1 bit, B P S K. You have 1 bit, 0 or 1. It goes to 1 symbol, which is plus 1 or minus 1. Then noise added to it, normal with 0 mean and variance sigma square. You get 1 receive value. So this is just 1 value to make a decision. So, if you if you talk about energy per information bit which is usually called E b energy per information bit. Is basically, is defined as the variance of the symbol here.

So, why is it the variance of the symbol? Basically it is the it is the reason is you have to think of this as a representing, the symbol as a representing a voltage. Then the square of the voltage, becomes some kind of power, to normalize the resistance to one. So, that is the that is actually true. When you when you when you take wave forms and then you define the basis and convert them into constellations, the energy in the wave form correspond to the magnitude square of the vector right? So, you remember your digital communications well and the vector space representations of signals, the energy in the wave forms it is the integral x, mod x of t square is the same as norm of the x.

Now, that is in terms of probability if you think of this as some random variable x, that basically becomes expected value of x square. So, we will say x is usually uniformly distributed, plus 1 and minus 1 equally likely for a particular symbol. So, this simply becomes 1 for B P S K if it is not B P S K. You have to have to evaluate. This expected value of x square of course, I have assume mean is 0, it is usually always good to keep mean 0. Mean is not 0 I would account for that also. That is why this expected value of x square variance, but remember it is the expected value of you mean 0. So, this is 1 E b. In the un coded cases 1 all right?

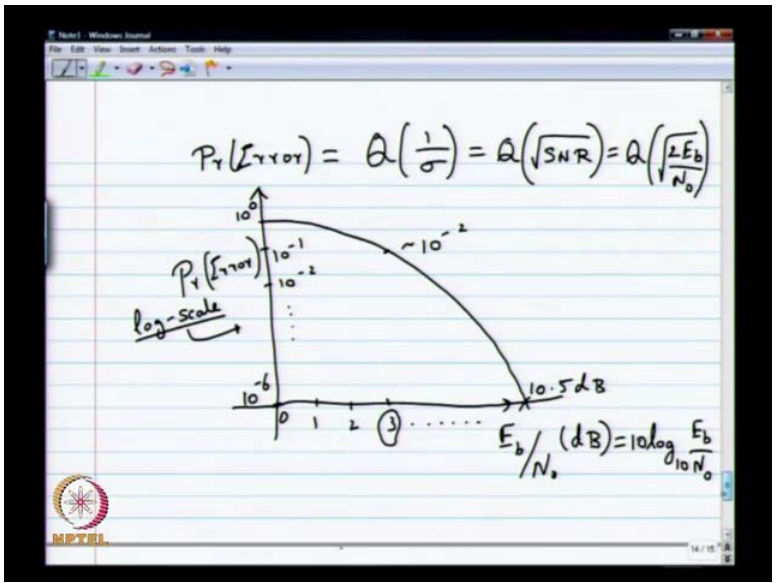
Now noise power, we simply take it as noise energy. So, noise energy will simply take it as sigma square, all right? So, the sigma square, if you convert from white Gaussian noise through the correlater etcetera. We will see this is the same as n naught by 2. So, this n naught is some number, that communications engineers will talk about it. Is the height of the noise spectrum, it is some number. And it depends on the temperature etcetera. So, you can put on some number for it. So, white noise that you expected that you see will be controlled by this n naught. If you do the receiver right, if you do orthogonal projections or match filtering carefully, you will get a noise of n naught by 2 that is for the variance. So, this is just from some kind of digital communications once again.

We do not have time to go into these details. So, assuming you doing either the course now or you have done it before. You know that this is true. So, if you want think of s n r for the un coded case, a good s n r to have is basically energy per symbol. This is the same as energy per symbol. Energy per information bit is the same as energy per symbol. So, s n r is usually defined as, E s by sigma square. So, if you want to convert to E b and n naught, this would work out to E b by n naught over 2. So, this is a this is a way to think about it. Usually s n r people will think of E s by sigma square.

So, how will you simulate such a system? It is very easy right, so if you are familiar with matlab if you want to generate thousand instances of the system, you going to generate thousand random bits do 1 minus 2 u. You get thousand symbols, then generate thousand numbers according to a pseudo random Gaussian distribution which matlab will give you. Then have a pseudo or add it, you get so many received values. What is the optimal decoder here? You have to just slice and you slice and get an estimate for the bit. So, if you do that, so here in this case, with the B P S K, we have assumed this simply becomes

1 by sigma square. So, let me just make sure I am not making any mistake here. So, alright so then the probability of error I am having to go to the next plage, but probability of errors may be keep a half page there and half a page.

(Refer Slide Time: 14:03)



$$P_r(\text{Error}) = Q\left(\frac{1}{\sigma}\right) = Q\left(\sqrt{SNR}\right) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

So, probability of error, what is probability of error for the un coded case? Is Q of 1 by sigma. Using the Q function, you get very simple answer. So, if you want to write in terms of S N R, it is going to be Q of what? Square root of S N R, if you want to write it in terms of E b over N naught it is going to be Q of square root of, square root of so it is very common. So, in fact you can even, you can even simulate the entire thing with the decoder also. So it is easy to simulate that you get you get your you get your answers.

Now, it is very common to present, so to speak performance of such a system using a curve like this. You going to put probability of error on the y axis. Then you going to put E b over N naught on the x axis and it is also very common to put E b over N naught n the d B scale. So, what is the conversion from regular scale to the d B scale? 10 times log base 10 E b over N naught. So, this is the conversion. So, you may be start with 0 d B, 1 d B, 2 d B, 3 d B so on. Alright and then, you compute probability of error for each point on the x axis you want. Then you can make a plot and the plot will look something like this.

So, it will go of like this and it will come down here and if you have something like 10 power minus 6 here, it is also very common to have a x axis in log scale, y axis in log scale. So, this would be like 10 power 0, then you will have 10 power minus 1, ten power minus 2 so on till may be the last line here is ten power minus 6 variable 0 be on the y axis, you cannot really all the way to 0 to keep on going. So, it is a problem with a log scale but, it works to an advantage in some case. So, it turns out if you do the computation you see with the B P S K 1 and an all that, this will, this will hit around 10.5 d B if I am not wrong.

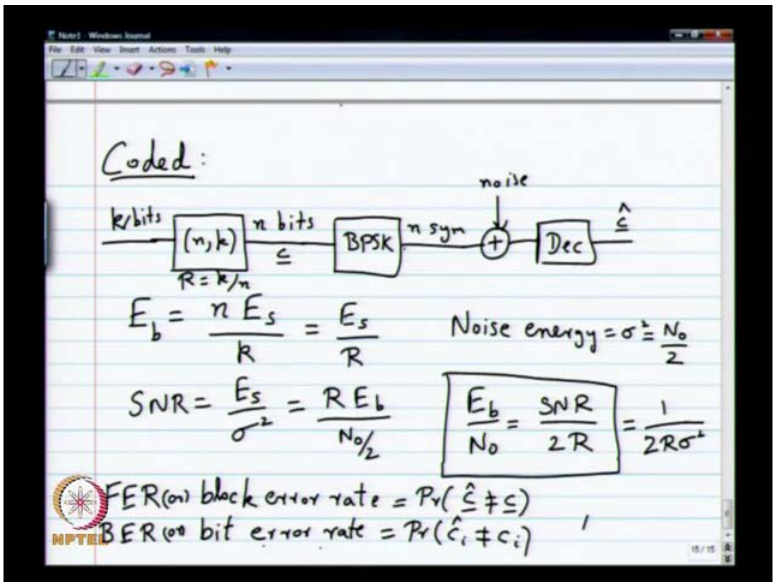
So, this is the good number to remember 10.5 d B E b over N naught for the un coded system gives you 10 power minus six probability of error. So, once again let me remind you to how simulate. Suppose I fix my E b over N naught s 3 d B. How do I simulate this whole system? So, you have to compute from 3 d B you have to compute sigma, Right? B P S K, E s s has been fixed plus 1 minus 1. You have to compute sigma for that. It is very simple formula. So, you go back to that formula and you compute sigma and you know how to simulate the whole system.

You run the, as many bits as necessary. Suppose for 3 I do not know the exact number. So, may be this is some roughly of the order 10 power minus 2. Then how many bits will you simulate? You need at least some 10 power 4 bits, why is that? A good statistic to remember is, if you are accounting for some event at least the given should happen at least 100 times before you can be sure.

Well when I say any one, either that even is compliment. Now, should happen at least 100 times. So, every possibility should happen 100 times could have another usually error is much less likely. So, you have to simulate till you get at least some 100 errors. Why hundred? Accepted hundred is a good number. So, if you if you have probability of 10 power minus 2, you want 100 errors, you have to simulate some 10 power 4 blocks. 10 power 10 power 4 bits you will get free.

So this is quite important, because many simulations you do this is involved. So, you get the answer. So but, you can also theoretically plot this curve, because you know the exact expression Q of 1 by sigma and that is related to the a r f c function. And it is theoretically most programs like matlab will have inbuilt functions for giving you a r f c. So, you can plot it also, without doing the simulation you can check. So, this is a good thing to do, if you have not done it before you should do it at least once during your stay here. I think it is a good idea. Now what happens on the quadrate system, that is what more interesting to us.

(Refer Slide Time: 19:11)



Coded:

$$\begin{tikzcd} \arrow[r, "k\ \text{bits}"] & \boxed{(n,k)} \arrow[r, "n\ \text{bits}"', "\underline{c}"] & \boxed{\text{BPSK}} \arrow[r, "n\ \text{sym}"] & \oplus \arrow[r] & \boxed{\text{Dec}} \arrow[r, "\hat{\underline{c}}"] & {} \\ & & & \text{noise} \arrow[u] & & \end{tikzcd}$$

$R = k/n$

$$E_b = \frac{n E_s}{k} = \frac{E_s}{R}$$

$$\text{Noise energy} = \sigma^2 = \frac{N_0}{2}$$

$$SNR = \frac{E_s}{\sigma^2} = \frac{R E_b}{N_0/2}$$

$$\boxed{\frac{E_b}{N_0} = \frac{SNR}{2R}} = \frac{1}{2R\sigma^2}$$

FER (or) block error rate $= Pr(\hat{\underline{c}} \neq \underline{c})$

BER (or) bit error rate $= Pr(\hat{c}_i \neq c_i)$

So, really interesting I think my this thing is different. So, if you do coded, then several thing change. First of all, you have k bits. So, this k bits are getting converted by the code n, k code to n bits and then these n bits are going through B P S K, you get n symbols. So, if you want to compute energy per bit? So, if you have E s to a energy per symbol, n times E s is the total energy bits, coming out. Here you divide that by k, this gives you the energy per bit. So, in other words, if the rate R is k over n, E b is E s divided by R. Now in the un coded case, what is R? R is equal to 1. So, E b is equals to s.

So, we are doing fine. So the coded case R will be less than 1. So, for the same E s, you will have a larger E b. So, that is fair and we are making it fair. Energy bit increases when you do coding for the same E s. Now I am not drawing the receiver side. Just its enough noise energy is the noise energy is simply sigma square, which is N naught by 2. Now, if you define s n r, s n r is simply E s by sigma square. Nothing changes, the other hand, if you convert it into E b over N naught. You going to get what? R times E b divided by N naught by 2.

So, the general relationship you get is E b by N naught equals s n r by 2 R. So, this is good relationship to keep in mind s n r by 2 R. So, if you do, if you do B P S K with E s equals 1. s n r is simply 1 by sigma square. So, B P S K becomes 1 by 2 R sigma square.

So, this is again a good formula to remember one by 2 r sigma square for B P S K .That is the E b over n.

Alright so, in most cases, when you do, when you do coded modulation you will not be able to get analytical expressions for a probability of error. First of all there will be 2 different probabilities of error. Let me also let me also talk about that. So, first it goes through a channel, noise is getting added, then you do a decoder. So, if your if your message is, if the code word is c, you get some kind of c cap here. So, you can talk about frame error rate called F E R, frame error rate or block error rate. Here, F E R is bit more common block error rate is also common.

This is basically, probability that c hat is not equal to c. Another thing you can talk about is what is known is B E R which is bit error rate. You can see why you people prefer frame error rate to block error rate. Why you get the same abbreviation, if you do bit error rate or block error rate on the other hand you say frame error rate you get different abbreviation and it is equally nice. Let us it will be a joke seriously not a serious answer.

So, bit error rate is probability that lets say c i hat is not equal to c i. And usually irrespective of i for most codes, this will be roughly the same. If incase it is different for each I, you have to take average of all of that. So, that is the understanding, but I will write as simply as c i hat not equal to c i, is very unlikely that u get much lower probe. Should may be happen, but usually it is the average bit probability. Now how do you simulate this? You can simulate this, it is not very hard. You generate k random bits do your encoder, get your n bits of the code word send it through B P S K, send it through noise, run your decoder, you have to know right code for the decoder.

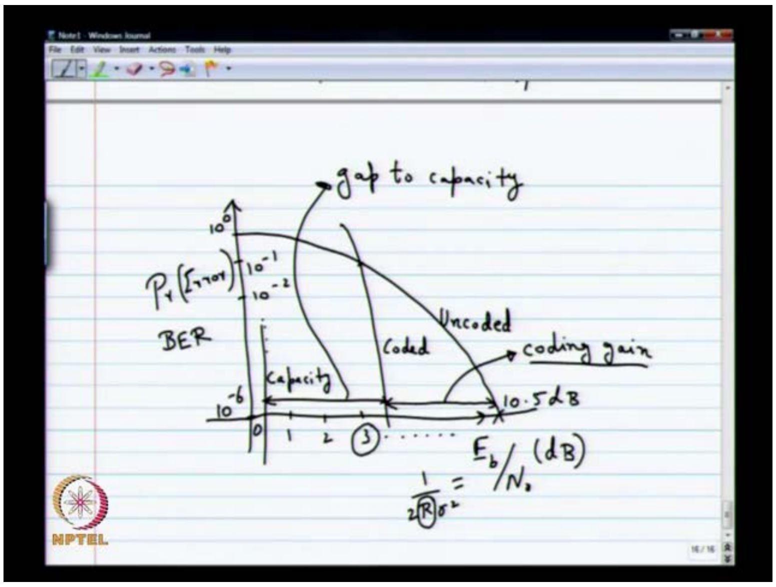
Its optimal decoder it is going to be very hard to write the code. You can imagine, at least theoretically it is possible. You can write the code, you get your c hat, then what you do in your simulation? You compare the c that was transmitted with the c hat that was, if c was not equal, you have made 1 frame error or block error. If it is not equal, you can also compare bit by bit and compute the total number of bit errors you made. So, at the end of say many runs of the simulations, you have the total number of frame errors.

Which you will divide by the number of runs, then you will have total number of bit errors, which you will divide by number of runs and then divide by the n. Also, that will give you the final bit error rate. So, usually what people do some times is important. It makes some difference in practice, instead of computing errors over coded bits, they compute errors over message bits. It is systematic encoding, they take the message bits. That is also done commonly, both frame error and bit error you only count for the message bits alone.

If you make some mistakes, in the does not matter, rest of the block can get right. So, that sum you simulate and it is like it is very important that you run simulations today, because any decoder you run, any modification you do or any intelligent algorithm code that you come up with it is very unlikely that given the complexities of soft decoding you might also have a very nice analysis for it. You will have to run decoder, run the algorithms to be able to do it.

Today if you want to work in coding, you should be good in programming also. The word coding now is, it can be used in multiple ways, in to mean the same thing or may be different. So, hopefully this is clear. What should you do then? Is now, then now, the idea is, what is coding gain right? So, let me just capture this, then I will take it and then will talk about coding gain. So, let us see.

(Refer Slide Time: 26:26)



So, I will remove all these other things, that I do not need. So, remember this is your un coded plot. So, usually go to mat lab, click on add text and add un coded, show the, show that is un coded or you put a legend for instance one of those things. Then what you do? You run your simulation for the coded case and hopefully you get some line which looks like this. So, this is a very typical kind of, common. So, this is coded. Remember what is different in the coded and un coded case? The definition of E b by N naught is different.

So, for the E b by N naught of 3 d B, you will get a certain sigma for the un coded case. The sigma you will get for the coded case will be different. In fact will be larger, weight is less than 1 means, the sigma you get for the coded case will be larger. That is the way, that is the way to do fair comparison. Why should the sigma for coded should be larger? Because you are spending more energy and transmitter to send all the extra symbols. So, you have to assume some kind of larger. So, you can think of the computation. So, remember, what is this? This is equal to, well not in d B, but let me write the other thing. This is equal to 1 by 2 R sigma square. For the un coded case R is equal to 1 for the coded case R is less than 1, some number that you have.

So, for the same E b over N naught, if you are plotting in the same access the sigma value will change, depending on whether or not you have code. So, that is something slightly subtle. Another way to think about it is, suppose I keep my, so this another digital communication way of thinking about it. So sometimes what you do is, you increase the clock rate. When you do coding, right? You know, what do you mean by clock rate? You put in more symbols for given time. So, what will happen, when you do that? That is the frequency that you are using increases. So, you let in more noise.

So, literally that is kind of captured by this increase in noise with same E b over N naught. That is that is the another way of thinking about any way does not matter now, comes in the definition of coding gain which is usually fixed at some particular probability of error. So, you have to, you have to whether its bit, usually it is in bit error rate. Let us say it is bit error rate. Usually 10 power minus 6 is the, is the conventional bit error rate at which people measure coding rate.

So, they will look at where this process, the access and where the un coded line process, the access and you draw some kind of a line there with a double arrows. And the length of the line is your coding gain and it is real fair coding gain. There is no unfairness in the comparison. You will see this gain in practice, if you are getting a certain 10 power minus 6 bit error rate for a certain power at the transmitter. If this coding gain happens to be let us say, 6 d B.

You can reduce that power by 6 d B. Do coding and still recover that exact same probability. That is what it means. Very real number, it is not some number that you, so this gives you a 6 d B power reduction, if this coding gain is. So, in practice this is how you evaluate coding gain. In these days it is mostly simulation based, it is very hard to do analysis only for the b c h kind of things. You can do analysis, very difficult to do analysis in practice for this new codes that are out there. I have not really described what these new codes are. This is how it will be justified. Hopefully this is clear. This is quite important, if at all you are doing any coding in your life after this, you will producing curves like this.

May be there are other types, you are to produce codes like this but, usually if you work on codes, you are you are looking at some curve like this. So, the other question to remember, another point to remember is, suppose you fix a particular rate and you say, I want to use more and more powerful codes, but for a particular rate let us say, rate is equals to half. You would, you would think, if you use more and more powerful codes, what will happen to the coded curve? It will go to the left, you want it to go the left, but it turns out, it cannot go more left than a certain point.

It will go up to a certain point, which you can call as a capacity achieving s n r with E b over N naught. Beyond that, it cannot go to left. So, that is the very standard resultant information theory. Once again we are not going into that and usually for rate equal to half and all that line will come somewhere here. Somewhere between 0 and 1. Let us say, for rate half it is about a 0.6 d B or something. So, this is the capacity line. So, what is this capacity? Means is fixed rate R that is the certain maximum coding gain that is feasible. You whatever you do, you cannot get better than that coding gain.

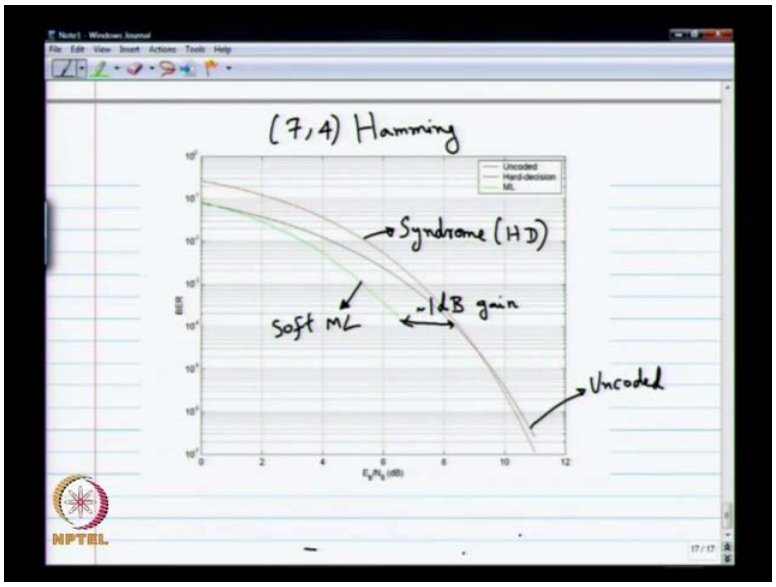
Then that is the very strongly analytical, really well established result. It is not some pyritic value. So, it is very mathematically clearly true result. So, if you take a course on information theory, you will learn more about it. In fact the course is being offered next semester with same information theory. You may not know about this. So it is good information to you. So it is being offered next semester, so you can take it up and you learn how to compute this capacity. So it is very interesting in it. Now how do you measure coding schemes? One way is to say, I get so much coding gain but, that is just a number right? How do I know that I am doing really well? How far are you from the left? And that is called gap to capacity.

That is also very nice absolute way of measuring, how good your code is. You might have a very fancy idea, by which you develop a fantastic code, but your gap to the capacity is 3 d B, then you are not doing that good. If somebody else has a code which has a gap capacity of 0.1 d B, they are doing way better. That does not matter, what you do.

It is an absolute measure, in terms of the best. That is possible these, both these things are equally important. In that, for a long time gap to capacity was, nobody worried about it. Today when they design codes, your gap to the capacity should go very very small. So, people have really come up with very smart codes, which have very very low gap to the capacity. So, those are the codes that we will be studying in the rest of the class.

So, at this point it is good to pause and look at our classical algebraic codes. The b c h and ((Refer time: 33:45)) codes which have fantastic structural properties. You have strong algebraic roots, they are very nice codes. How well do they, do here? That is, that is an important question. Where will they be in this curve, what is their coding gain? What is their gap to the capacity? So, it is an important question to ask. So, for that you going to pull out 2 different curves, that they already have. The first one is the 7,4 hamming core.

(Refer Slide Time: 34:13)



So here is that, that curve. You see that. So the 3 different curves here. So, hopefully all of them are visible to you. So one is the black curve, which is marked in the legend as what? Un coded. So, I think what I should do is, so I should probably call it from journal. So then I can write some comments on it. Where is my journal? Sorry I think I should insert from here, then I think it is to see and I can add some comments on it.

So, this curve so this black curve here is the un coded curve and you can see like a say 10 power minus 6 roughly around 10 and a half d B and then this is, remember this is 7,4 hamming code. The red curve is the hard decision, so this is syndrome decoder, hard decision. So, what do you do? When you do hard decision? You first slice and then you run a syndrome decoder and then the green curve is the soft M L. Soft M L decoder is that. So, how well is the 7, 4 hamming code is doing in terms of coding gain, the hard decoder is doing quite bad.

So even, so even at 6 d B it is doing worst than the un coded case. Only around 9 d B or 10 d B. Also you start saving some gains, because of the hard decoder but, the soft decoder is doing quite alright. So, even at let us say 10 power, so if you measure at let us say 10 power minus 4 bit error rate, roughly about how much is this distance? Let us say, so roughly around 1 d B, at 10 power minus 4. No I mean this, this going to cross some by here right? So, I think roughly about 1 d B, 1 d B of coding gain from the hamming code and the capacity is going to be somewhere at close to 0 and the gap is about 7 d B or so. You are far away from the capacity, 7 d B is lot, you might think.

What is 1 d B I do not know, you may not work with so much circuits. So, it turns out many communication systems at the very front and they have these power amplifier. So what called power amplifier? These are analog electronic devices and they are very very expensive, the reason why is the expensive is, you want linearity and face and everything in communication right? You are doing Q A M, you want face and all to be preserved you cannot have which is non-linear. So, all that you want and you want also high power. Just go over large distance and that happens to be a very expensive component.

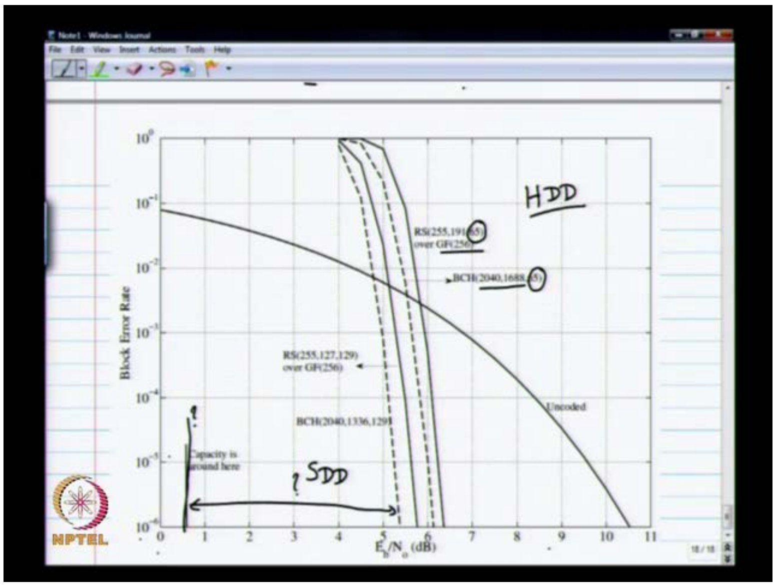
If you can reduce the power of that by 1 d B, the price will go down by several tens and thousands of dollars, imagine. So, it is a big reduction in price. So, coding gain really buys you a lot of things ,1 d B is not bad but, if you give a 7 d B you should think about it. Something more I can do to gain this, gain from this 7 d B right? So, think about various applications for instance, we think about satellite, that is way out, next to the mars, it is getting all its energy from sun. Does not get to much of energy from mars.

So you cannot keep spending all kinds of energy, you have 7 d B to gain, 7 d B is a big difference. May be you can go from mars all the way to Jupiter. It is still communicating. So, that is the way to think about this, the use of a codes. This used By NASA a lot. NASA is a big what you say, spender of money, good design codes. Anyway, so this is the hamming code. So, what about other codes? Let us say, may what would you say this is? This is good, bad or ugly or not bad?

So, if complexity is the major limitations, 7, 4 is really less, right? You what you can do, with real it is really less. So, if may be, if you go to longer block lines, you can do something better. 7, 4 is really low and may be you can even show that, for this block length. This rate you cannot do better than this. So it is really low block length, before you cannot do much. The capacity that I drew is valid only when block length becomes very large. It is in fact tender to infinity to true that result.

So, you can get to those values only for long block lengths. So, there is no point in comparing 7, 4 hamming code with something infinite and say have a gap of codes. Will have a gap you cannot do anything better than that. So let us take longer length code, let us say some rate Solomon codes.

(Refer Slide Time: 39:17)



So, I do not know, so this will be able to insert PDF files? See, it is not able to insert PDF files, I am sorry. So, I can open it then, I cut and paste it there, is it? This is what I do, then I copy it down to that. That is not bad so, this is what happens for block error rate. This is E b over N naught for the for some b c h and Reed Solomon codes. So, I believe the square, the solid line is, whether I have two lines solid line is the Reed Solomon code, the dotted line is the b c h code and the other curve is the un coded picture. What have I done here? I have done same minimum distance for b c h and Reed Solomon, this is over g f two fifty six. So I have taken g f 2 power 12 whatever b c h code a large b c h code. So, b c h will be slightly better than the Reed Solomon code.

We know that is true, the minimum distance is slightly better. So, it will be better and these are actually hard decision decoders. So, do hard decision decoding, hard decision decoders, no soft decision. So, you can see it crosses 10 power minus 6 around 6 d B right? 5 and a half to 6 and a half right? Look at the rate of the code. So, this is got a minimum distance of 65. So, this is got a minimum distance of 129. So, this is a rate half code, this is a rate 3 by 4 code roughly, those are the things I have spoken and still there is a gap to capacity. Gap to capacity is about 5 d B s for rate half and all that large gap it gives you, the coding gain.

The coding gain is also large. I have gone to a larger block length, it gives you some coding gain even with hard decision, right? So, that is, that is something valuable, right? Hard decisions are hard decision decoders are much simpler to implement. The p g say decoders are not too complicated, complexity is not exponential in k. Only depends on the error correcting capability. So, it can be implemented and you get decent coding gain. The only drawback of these codes, definitely is this gap.

You do anything about these gaps, that is what needs to be done for this gap. So, clearly soft decision decoding has to be used, to be otherwise you would not going to do, not going to do much better. It is long as you stick yourself to the hard decision decoding, this kind of gap will always be. So, you have to do some kind of soft decision de coding. So, it overcome the gap. So, it is something, so the question was should I say anything more about capacity will not be in this codes?

Just takes long, so the question is, if I write the implement ML soft, ML of the decode of these reed Solomon b c h codes. How far to the left, can it go actually? I do not know the answer to the question to be very frank. But, there are some results on ML of decoding of Reed Solomon codes but, not for this kind of block lengths. It is difficult to analyze, there are some soft decoders for each Solomon codes which give you about 2 d B or so additional coding may be maximum 2 d B, but not these kind. May be these kind of rates will get better. But, nobody has shown, it can get to 1 or some there is no such result.

But, Reed Solomon codes, you can run some decoder even ultra-theoretical, let say I mean decoder is very complex, but some more somebody has analyzed it and shown that it can, because it is so complex you cannot simulate. You have to analyze, so nobody has done that nobody has shown it is close to 1 or something. But, other codes people have shown simulation results very, very close to capacity.

In fact there is a paper which says there is an LDPC code low density parity check code which gets to 10 power minus 6, 0.01 d B away from the capacity. So 0.01 d B rate half 0.01 d B would almost there in picture. You cannot see it, so the question to ask, is there anything else left to do in coding? The capacity and you have seen the capacity, try to capacity. Yeah there are some limitations to that code also of codes, show it but.

So I do not know the answer to that question. If you have to do ML decoding for this b c h codes, can you push it to the left? Think maybe there are some theoretical answers, but do not know if there are any practical answers. But, definitely any implementation of the soft decoder is very hard for b c h codes, that much is true, is very, very hard and there is no sub optimal problem, is there is no easy way to iteratively improve a sub optimal answer. So, you saw in the bit was m a p the main motivation is you have one answer, immediately and then the next answer you get from next par digit and the next answer right? As long as keep it is independent you are okay.

But, in Reed Solomon codes, you see dual hairs, all kind of weight you know the dual also has a large weight a dual of m d s codes. So, m d s you can show that there are also large weight. So, you will have only large weight vectors in the dual. What do you do? You cannot do sub optimal vector, the way we do for the other codes, that is one problem. So, you have to overcome that in smart algebraic ways. It is possible, there are soft decoders, they give about 2 d B if not too much more than that, why does it show the fall? Why is it so steep? Yeah it is usually like that. Like that you can show the fall will be like that, you can do that computation in the expression.

So, these are actually theoretical curves, these are not simulated. I have just assumed Bounded distance decoding. If you do computations, for bounded distance decoding all errors up to a certain point can be corrected. That itself will give you a steep fall. What does less than 4, I am sorry. Is that 0.5 which one confused, what is the question?

Less than about 4 and a half d B, it is worse than un coded. Yeah that will always happen, for hard decision decoders. I am sorry as sigma is higher, because of the E b by n naught. Yeah, well hard decision decoders, will have this problem. There always be, they will cross the un coded curve somewhere. You do soft decisions, then you see it will be slightly better. You do something, it follow the un coded thing and then it will come down. This is also block error rate, now this is slightly different. Compute bit error rate something more will happen. Something different will happen.

Block error rate is also a misleading reason, why this we are showing. Bit error rate will be slightly. Compute better rate for a better distance decoding, because you can analytically compute. It was the question, was that is why I computed block error rate? Block error rate is very easy to compute, bit error rate is harder to compute in a bounded distance decoder. There are methods you can do approximate computations. That to look much better here, it come out to be better. Block error rate is a very easy expression. So I can evaluate very easily, at least the approximation of the block error rate. Any other question of these curves?

So, at this point, we have done some general background on soft decoding. Then we have seen, how to measure? How to do some sub optimal? Some general ideas and how to do sub optimal decoders and then how these classical codes do when you look at, look at them at a coded mode relation point of view. What does the coding gain that they provide? How far are they from capacity? What we are going to see from next classes? Codes which have very good sub optimal, soft decoders that can very it can get close to capacity. There are 2 types of codes which are very popular today, one is this low density parity check code. Which is what we will see immediately and after that will see turbo codes. Which will come up a little bit. So, we will stop here.