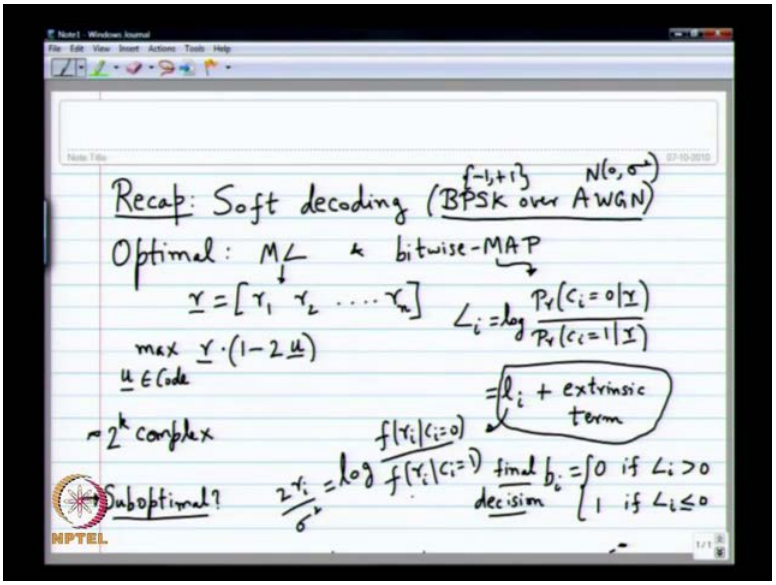
Coding Theory
Dr. Andrew Thangaraj
Department of Electronics & Communication Engineering
Indian Institute of Technology, Madras

Lecture - 20
Bitwise MAP Decoder from the Dual Code

(Refer Slide Time: 00:20)



So, let us begin, we going to begin once again with the recap. It is going to be a rather long recap I want to mention quite a few things in this recap. So, the general theme that we have been following in the last few lectures is soft decoding. So, in particular we saw two different optimal soft decoders, if you want to look at optimal soft decoders. There are two possibilities one is the ML decoder right, the other is the bitwise MAP decoder and the soft decoding.

We have been always will always assume BPSK over AWGN, so keep this in mind. So, BPSK will be minus 1 plus 1 and AWGN is noise zero mean variance sigma square. All these things will be underlying assumptions that I may not repeat every time, but that is the that is the idea here. So, for some reason the just come back, so the so now we have some good description for the ML decoder and the bitwise MAP decoder.

So, the ML decoder is quite easy to describe for BPSK and AWGN. You have to do something which is very simple, what do you have to do for the ML decoder suppose you have received vector r 1, r 2, r n; you have to compute correlations, so how many

correlations do you have to compute. So, the code is some reason it thinks I am doing something here with the, so I do not know how to avoid that sorry toolbar properties auto hide the taskbar is that. So, what do I do, so let us see auto hide and see what happens.

Oh my God apply it seems to be little bit better alright. So, what do you do with the received vector r if you have to do ML decoding. You have to compute correlation what are the correlations do you have to compute you have to do r dot 1 minus 2u right. So, what is this u it to maximizes over all u in the code. So, that particular code word which gives you the maximum correlation you pick as the ML decoded code word. And that is that will give good properties for the bitwise MAP there are various descriptions possible. We finally, settle down this likelihood ratio description which was the, which was at least there was a conceptual advantage in that. So, those no computational advantage even in then ML decoder, there is no computational advantage in changing anything.

Because the reason why mean you still have to do 2 power k correlations. So, that is a lot of correlations you cannot really do it if k becomes large. Same thing is true in the bitwise MAP. But, with the likelihood ratio at least there is a conceptual understanding of how the final expression splits into some convenient parts and those that is useful. So, I think if I remember right I called capital L i right. Did I use capital L i for the the final probability that the bit is 0. Given r divided by probability that the bit is 1 given r. So, there is a way to convert this into likelihoods because if you assume that your code is non trivial in every location. You will get a very simple conversion into likelihoods and if you do a lot of simplifications you can write it as l i what is l i this is basically probability that.

Probability that C i is 0, no I think I should write it in terms of f. So, that is better f of r i given C i equal to 0 divided by f of r I given c i equal to 1, which will simply be e power 2 r by sigma square right. So, I think is there also a log likelihood ratio did I I must have introduce that. So, did I say did I say log of this was capital L I, so nobody objected to that. So, small l i is also log of that right is it log of that, so I think that is that is a better thing to work with because you will see that there are some simple advantages there. So, log of this which works out to 2 r i by sigma square.

So, you can write this as small l i plus an extrinsic term the extrinsic term is a bit messy. So, I am not going to try and reproduce it the extrinsic term is messy and it involves a numerator and a denominator. The numerator is contributed by those code words which have 0 in the ith position. And the denominator is contributed by those code words which have a 1 in the ith position. You have this strict of dividing every term by something and then writing it up in some convenient form and finally if capital L i is. So, basically the decision on the ith bit which I think we called as b i maybe. So, the b i which is the decision on the ith bit this is basically 0 if L i is greater than 0, one if L i is let us say less than or equal to 0 does not matter.

So, this is the final decision remember that this is the final decision, in a bitwise MAP alright. So, this is a brief recap of what we did, but the main observation is both of them are roughly 2 power k complex the k is the dimension of the code. So, if n code is an n k code and if k is increasing k becomes 1000 and all that you have no hope of implementing either the optimal ML decoder or the optimal bitwise MAP decoder completely. But, this form this form for the this form for the bitwise MAP decoder is interesting it suggests that you might be able to either iteratively or approximately evaluate the extrinsic term.

Maybe even wrongly no say approximately I am wrongly maybe even wrongly, but there also there might be some advantage. So, sub optimal soft decoders are what those are the decoders that are really interesting. So, you are not really interested in optimal soft decoders because in most cases they are not implementable. So, sub optimal decoders are going to be crucial, so that is the first that is the main idea here the. So, the main reason you study optimal decoder is not to implement, so to realize that sub optimal decoders are readed. So, that is the reason why you study optimal decoders here, so what you do for sub optimal is an interesting question yes.

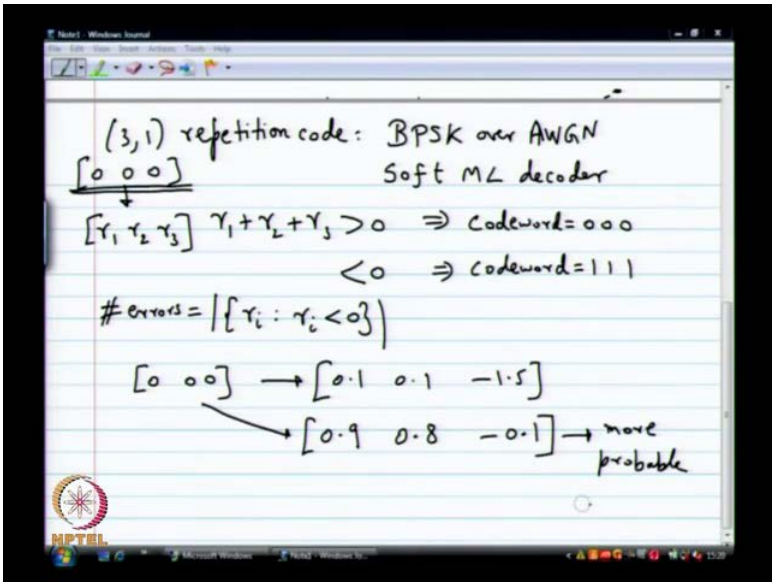
So, optimal for what so when whenever you say optimal you mean it has to minimize or maximize something right. So, the ML decoder will minimize the probability of block error the bitwise MAP decoder will minimize the probability of bit error and both of them are the same. In fact what happens is if you simulate either one of them you will see one will lie on top of the other both will be mean exactly the same in most cases. So, it is no need to worry of too much about it, but they are strictly not equal that is one point. So, another comment I want to make is just which was raised I think towards the

end of the maybe the last lecture about error correcting capability for these optimal decoders

So, if you remember on the B S C we saw syndrome decoder etcetera then we define the notion of an error correcting capability for a decoder. And then in fact we even had a bounded distance decoder which has which works only within a certain radius right. So, if there are lesser than, so many errors you can correct it, what about the soft ML decoder for BPSK AWGN.

What is it is error correcting capability in the sense that we defined before if there is one. So, many one error can it correct it if there is two errors can it correct it that is the kind of question I am asking. So, let us take let us say let us say we take the repetition code. And if I ask you the question what is it is what is the error correcting capability for the ML decoder what will the answer be. So, you think about it so what should happen if error correcting capability.

(Refer Slide Time: 10:22)



So, let us stay let us say let us say we fix the 3 comma 1 repetition code and then I use it on the BPSK AWGN just like before alright. I can like it this two does not like it AWGN we are using BPSK over AWGN and then we are the asking the question what is the error correcting capability. What are we using we using the soft ML decoder what does the soft ML decoder do. It evaluates r 1 plus r 2 plus 3 if this is greater than 0 it decides the transmitted bit was 0. If it is less than 0 it decides the transmitted bit is the code

word. Let me say code word error correcting capability is define with respect to the numbers of errors in the code word right.

So, the code word was 0 0 0 and in this case it decides the code word is 1 1 1. So, when you define error correcting capability you need this notion of number of errors it does not make too much sense. In BPSK and AWGN you will see why but number of errors is something that we used in BSC. So, when so you say what about r 1 r 2 r 3. What is the numbers of errors in r 1 r 2 r 3, suppose you are transmitting 0 0 0. What is the notion of number of errors in r 1 r 2 r 3, what is one way to define it. For instance if you want to define it like in the BSC sense how would you define it.

Yes, I take the signs, so you are doing some kind of hard decision and saying. Suppose I say if r i is positive I will say no error happened in the transmission of the ith bit. If r i is negative I will say some one error happened suppose you define the number of errors. In that sense what is the error correcting capability of the soft ML decoder. So, I have to define number of errors is this size of r I, such that r i is let us say less than 0. Remember I am I am transmitting the all zero code word that I fixed I fixed the all zero code word. And then I am saying the number of errors I am defining as number of errors I am defining as the number of positions where r i is negative.

So, if I do this in fact the soft ML decoder has no error correcting capability there are r 1 r 2 r 3 for which it will make an error right. Do you see do you see that or not suppose if r 1 is 0.1 r 2 is 0.1 r 2 r 3 is minus point minus 1.1. Technically there is only one error, but then your soft ML decoder is going to wrongly decode to 1 1 1 right. So, remember why did we ask that question of error correcting capability in the BSC why was it so important. Why did it make sense it made sense because lower weight errors were more probable always only the number of errors matter in the probability number of bit errors matter in the probability of the entire vector. Now, when you go from 0 0 0 to r 1 r 2 r 3 number of errors does not make much sense for instance.

You might say I transmitted 0 0 0 and I received let us say 0.1 0.1 I do not know minus 1.5 or you might say I transmitted 0 0 0 and I received 0.9. Let us say 0.8 and then minus 0.1, now both these guys from an error correcting capability point of view are thought of as being the same right. You do not you do not see difference between these two guys, but then this one is much more probable then this guy. So, this is more probable then the

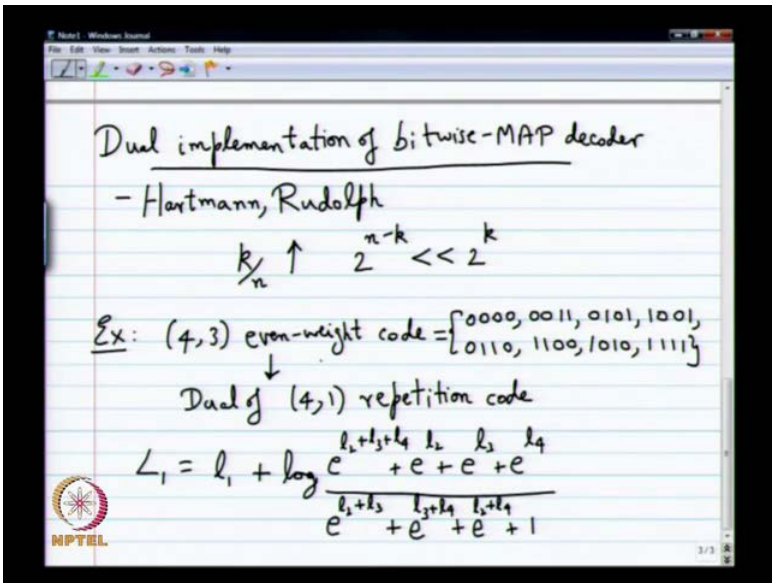
previous one, so asking questions in terms of error correcting capabilities for soft decoders is not entirely correct. You should not you should not worry about it the fundamental principle there the reason why we asked for error correcting capability in the BCH in the in the BSC was lower weight errors are more likely always and.

So, we asked that question here the only question you should ask is, what is the probability of error, what are the more likely errors. The more likely errors are corrected by the soft ML decoder that is all whatever the ML decoder corrects is the best possible thing you can do. So, in general for soft decoders questions like error correcting capabilities are seldom asked you it is tough to answer that question it is not very easy entirely you cannot answer that question. So, people do not worry about error correcting capability they worry only about probability of error. Probability of error is lower you are happy well then even if you whatever SNR you pick.

Something like this the vector below will be more probable than vector. So, that is what I am trying to say it is not that dependent on its well that is true I mean. When you go to with in the opposite is true, when you go to high SNRs. Actually where it really hurts is when you go really high SNR you would like to have a guarantee on error correcting capability was at high SNRs. You can say that the number of errors are going to be low that you can say, but you cannot say the opposite. That is what I am trying particularly at a lower SNR comparing two vectors much more difficult alright. So, is just a point of departure I want to point that out was somebody asked me that question.

I think towards the end of the last class I want to rewrite that. So, in general you should not ask this question for soft decoders, do not worry about number of corrected errors. Only at high SNRs you have worry about it, but that is something will come back to maybe later if we have time slightly more difficult notion to get your head out alright. So, that is the that is the that is that, so what we are going to next is this is just this one more thing about the bitwise MAP decoder after that we will move on to something little bit little bit different.

(Refer Slide Time: 17:16)



So, I will call this as the dual implementation of bitwise MAP decoder. So, this is a crucial idea, so this place fundamental rule and the simplification of these of at least one kind of sub optimal decoder. So, that is the that is the idea here, so I am going to do this once again by example. There are general expressions and this idea is basically due to two people called Hartmann and Rudolph.

I encourage you to read their original paper it is not too difficult to read. So, they are the ones who introduce this idea of looking at the dual code to implement the bitwise MAP decoder for the original code. So, far the bitwise MAP decoder was implemented using the code words of the original codes itself. Now, when you go to higher and higher data rates what will happen when k by n increases what happens your dual code. Will have much fewer code words then the then the original code. So, if you have a bitwise MAP decoder described in terms of the code words of the dual you have a computational advantage. It may not be a huge advantage, but at least there will something.

So, this is kind of similar to the syndrome decoder. So, if you remember when we motivated the syndrome decoder there also something very similar happens, complexity is really 2 power n minus k it is not 2 power k even though it is sub optimal ML. So, this is kind of reminiscent of that idea. So, instead of using the generator matrix for the decoding you are going to use the parity check matrix. So, I will do this once again by example we will do a couple of examples and see cases where it works cases where it

does not work and then will rest it leave it there. So, without going much deeper into the technical details.

So, for some reason this things whenever I put my hand down it thinks I am I am writing something there that is a problem. I think the touch pad technology is distinguish between my hand and the pants maybe there is some setting that you can do to make that happen I do not know anyway. So, let us take the first example the first example I am going to take is the let us make it a little bit little bit non trivial we will take the 4. Yes 4 is fine 4 comma should I take 4 comma. So, we will take 4 comma 3 even weighed code. So, let me remind you, what is the even weighed code what is the even weighed code.

So, it is basically the dual of the repetition code that is another way of defining it. So, this is the dual of 4 comma 1 repetition code. So, another way of describing the even weighed code is to say that it contains all the length 4 binary vectors of even weight. There will be exactly 8 in number, so it is a 4 comma 3 code and you can also check that this is a valid code etcetera.

All that you can do, so let us see what, what is the what are the various code words of this code. So, there will be eight of them 1101 0110 01 then what 0110 1100 then what 1010 and then 1 1 1 1. So, those are the eight code words of this code let us try and write down the expression for what for the for capital L 1.

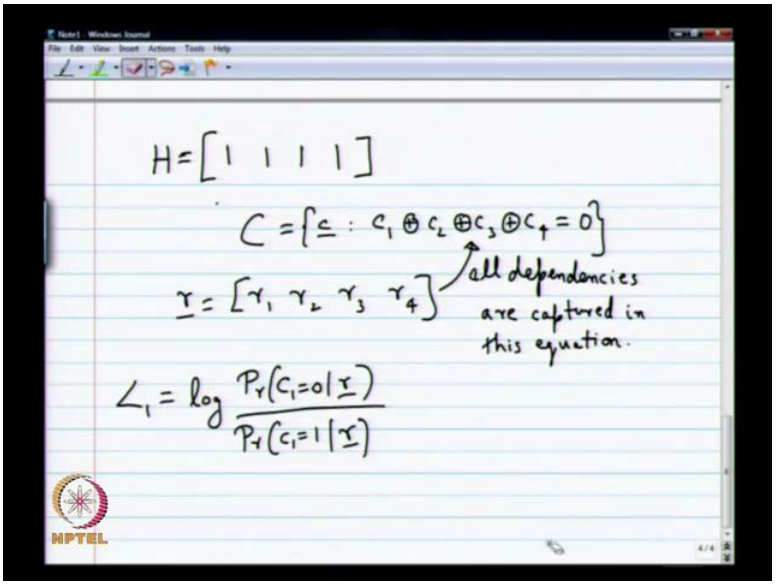
Let us try to write this down if you have to write the expression for capital L 1. What would happen capital L 1 remember is the a posterior likelihood log likelihood ratio for the first bit. So, you will have a form similar to before, so you will have four terms in the numerator and then four terms in the denominator yes all the this. So, BPSK over AWGN minus 1 plus 1 everything sigma squared those things I am not going to repeat for every single situation that is assumed.

So, it is basically going to be l 1plus what will be the extrinsic term yes you have to take logarithm of this e power products right. So, it is going to be e power, so it is going to be e power l 2 plus l 3 plus l 4 plus e power l 2 plus e power l 3 plus e power l 4 is that everyone agree, so the numerator. And then the denominator you would have e power l 2 plus l 3 ok l 3 plus l 4 l 2 plus l 4 plus 1 is that there should be a plus 1 much. I can see of course, if you have not been keeping up with these lectures you suddenly happily

relaxing for 2 weeks. And showing up this expression will be surprising to you what I cannot help that you know you have to be you have to keep up with the lectures.

So, this is the expression any insight anyone willing to I mean take a take a wild swing at see. If you can notice something here which you think is interesting no not really right I mean. I do not want a brilliant answer I just want a simple answer to say not really I mean it is not does not seem very obvious. At least actually there might be, but it does not seem very obvious that it should be. It is it is some very laminating answer, but on the other hand I, I will describe the same code in a slightly different way and then I will ask you the question about what is this capital L 1. And how are we trying to find it and you will see there should be a simpler explanation and that is where this idea of using the dual comes. So, let us see so if you look at the parity check matrix for this code.

(Refer Slide Time: 25:31)



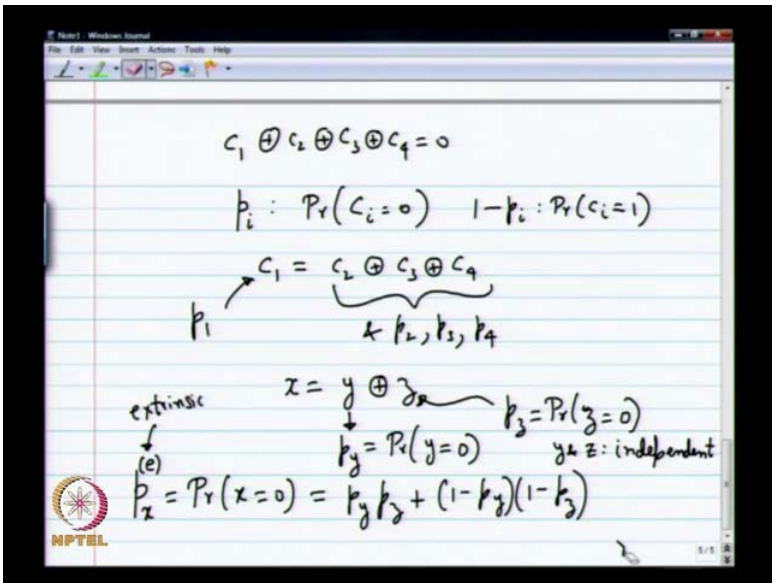
What is the parity check matrix four 1 s in a row right. So, that is the parity check matrix for this code it is the dual of the repetition code. So, the generator matrix for the repetition code becomes the parity matrix for this code. So, you have 1 1 1 1 essentially this code is described by set of all code word C such that C 1 plus C 2 plus I will put a circle around plus because we are also dealing with real numbers here. So, I want to distinguish between plus and xor C 1 xor C 2 xor C 3 xor C 4 xor equals 0. So, that is the that is the complete description for the code is the set of all code words C which xors to 0.

So, any dependencies between the four bits of the code word is captured by this equation. In fact any dependency between the four received values is captured by just this condition. So, this you better convince yourself of this because I am going to change this in a tricky fashion later and rather ask you the same question it will seem like it is true. But, then it will not be true, so it is good to good to convince yourself that this is true all dependencies are captured in this equation. So, that is the only condition that relates C 1 C 2 C 3 and C 4, this no other condition that relates it yes or no right for all code words this is the only condition.

Now, if I ask you the question given r 1 r 2 r 3 r 4 what is r 1 r 2 r 3 r 4 you know how it is describe right. You encode the code word I mean you modulate the code word to plus 1 minus 1 and send it through a four independent realization of the AWGN channel you get four received values. So, given these four received values the questioning when I am asking when I compute capital L i 1 is what is logarithm of probability that C 1 is 0 given r. So, this is the vector r by the way divided by probability let us C 1 is 1 given r is that alright. So, what so anything occurs do you know, so if you can use this equation in smart way to simplify this expression right.

The answer you get clearly should be the same as the answer you had before, so that was also me expression for the same probability if you can. Now, use this equation in a smart way to somehow simplify this probability in what saying how you do it. But, if you can do that then this the answer you get should also be the same or it should not suddenly you should not get anything that is drastically different or entirely different. So, this equation clearly captures everything and that is what that is what we are going to use is that ok. So, let me let me describe that that approach is not is not entirely different from what we have been seeing before. But, you will see that the way we think about it will change, so let me let me look at this again.

(Refer Slide Time: 29:16)



So, I want to look at this question I have given this equation C 1 xor C 2 xor C 3 xor C 4 is 0, I am given a received vector r which means probability that. So, let us say I use I try and compute this probability that C 1 is 0. Given r 1 this is for me easy to compute, so let me not let me not do this computation let me just. So, let us let us let us look at this equation first and then ask the question, what does how we can use this equation to simplify the previous expression. So, that is what I want to find out.

So, I am going to approach this in a slightly different way suppose somebody tells me p i is the probability that C i is 0. Suppose somebody tells me this do not worry about how I how I got this answer somehow somebody tells me p i is a probability that C i is 0. So, of course 1 minus p i will be the probability that C i is 1. So, that is and now from this equation I have, so suppose I rewrite this equation as C i equals C 2 xor C 3 xor C 4 I can do that.

Now, I have one value p 1 which tells me something about C 1. So, that is something that is given to me. And then this equation and the values p 2 p 3 and p 4 will tell me something about C 1 again right yes or no. So, if you try and compute what that expression is it looks like you will get something little bit complicated. But, to compute that I will do that in two steps, so instead of computing everything totally. The first step I am going to ask is suppose I have an equation of this form suppose I say x equals y xor z.

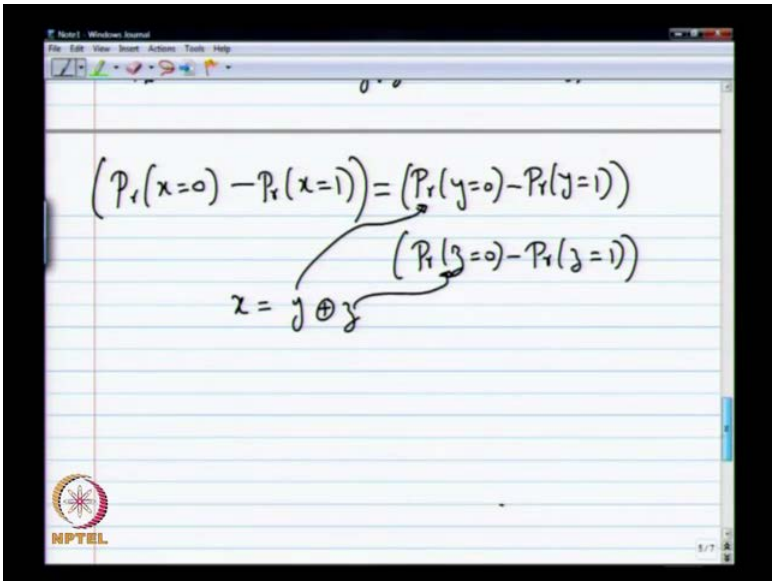
So, we will try and use this information in a slightly different way suppose I give you x equals y xor z and p y is the probability that y is 1.

Let us y is 0 does not matter likewise here p z is the probability that z is 0. So, if you now try to compute probability that x is equal to 0. So, you are going to get yeah, so y and z both have to be either 0 or both have to be 1. So, you would get something like this p y p z plus 1 minus p y times 1 minus p z yes assuming independence. So, let us say this are independent, so y and z independent is that now this equation looks.

So, maybe this is I should call it I should call it something, so let us say this is some p e x, the reason why I put e is some kind of extrinsic information about x. So, notice this does not use any intrinsic information I might have a bored x if somebody told me ahead of time that p x was something that I had already. This information is new it kind of it has to be in a way combined with the previous information I had.

So, will see how to put it together in the decoding context later, but anyways so this question is interesting, so this expression seems simple enough. But, now if you try and use this here, so you C 2 xor C 3 xor C 4 you will see you have many more terms right. You have four terms and it does not seem like there is any simple way of simplifying it. The crucial trick is to look at to look at this quantity you have to look at is basically.

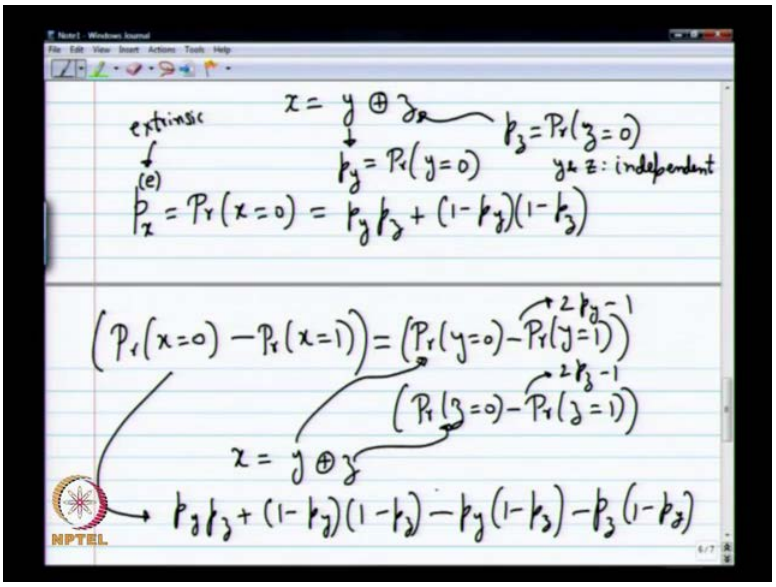
(Refer Slide Time: 34:00)



Probability that x equals to 0 minus probability that x equal to 1 if you look at this it turns out this will become it will become something like this. Let me write this term more clearly probability that y is 0 minus probability that y is 1 times probability that z is 0 minus probability that z is 1. So, it turns out in this equation x equal to y plus z this is true you have given only this guy and this guy with only these two information if you have to find about find anything about probability of x you go ahead. And find the probability this equation will be true, so you can simplify this it does not seem very obvious from.

From this equation probability of x equal to 0 minus probability of x equal to 1 does not seem very obvious from this. But, you can simplify it and you can write it this way if you want I will quickly show you how that works out let me quickly show, how that works out some reason not able to go there, so this is extremely.

(Refer Slide Time: 35:28)

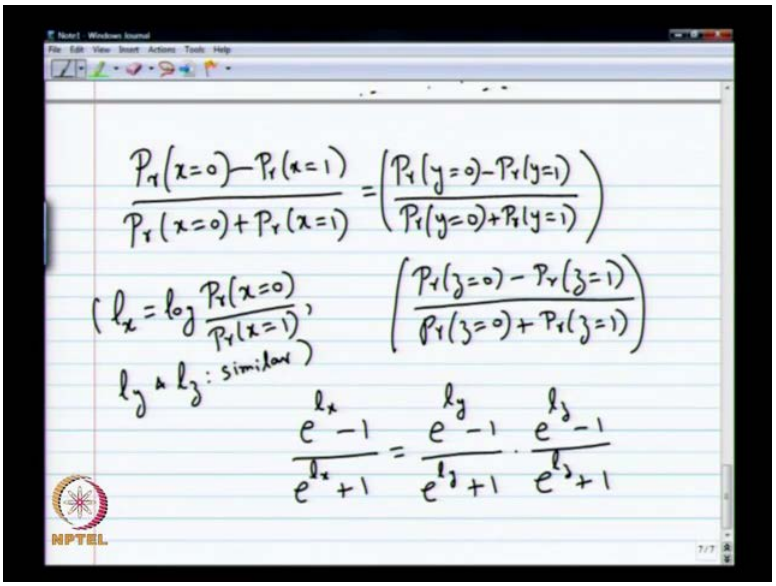


So, if you look at this expression we try and evaluate it is going to be p y p z plus 1 minus p y 1 minus p z minus what p y into 1 minus p z minus p z into 1 minus p y. If you simplify this you will get these difference remember what is this difference it is 2 p y. So, this guy is what p y minus 1 minus p y right. So, it is 2 p y minus 1 and this is also 2 p z minus 1. So, if you simplify this you will get you will get that, so you are getting 4 times p y p z. You can see these two are the same, so it is a it is an interesting simplification that happens. So, if you have x equals y xor z and you are given only

probabilities for y and z and only using that if you want to compute probabilities for x it turns out this equation will follow this.

This rule probability that x equal to 0 minus probability that x equal to 1 will be the product of these two probabilities is that ok. So, by itself it may not be may not be so interesting, but when you convert it into log likelihood ratio by itself also. It is interesting by the way, but anyway if you convert it into likelihood ratios it becomes even more interesting. So, how do we convert this into log likelihood ratios. So, the first important trick is to realize that you can divide anything you want by probability of x equal 0 plus probability of x equal to 1 what is why can you do that it is always 1. So, you can happily divide anything you want by that ok.

(Refer Slide Time: 37:24)



So, that will help you in this in this computational probability. So, let me do that so if I do that you will see probability of x equal to 0 minus probability of x equal to 1. So, this seem like a I know very basic elementary computations and probability. But, to be patient with me we will get somewhere at the end of this end of this work plus x equal to 1. This is probability of y equal to 0 minus probability of y equal to 1 divided by probability of here I will do y equal to 0 plus probability of y equal to 1. So, I should do that this times.

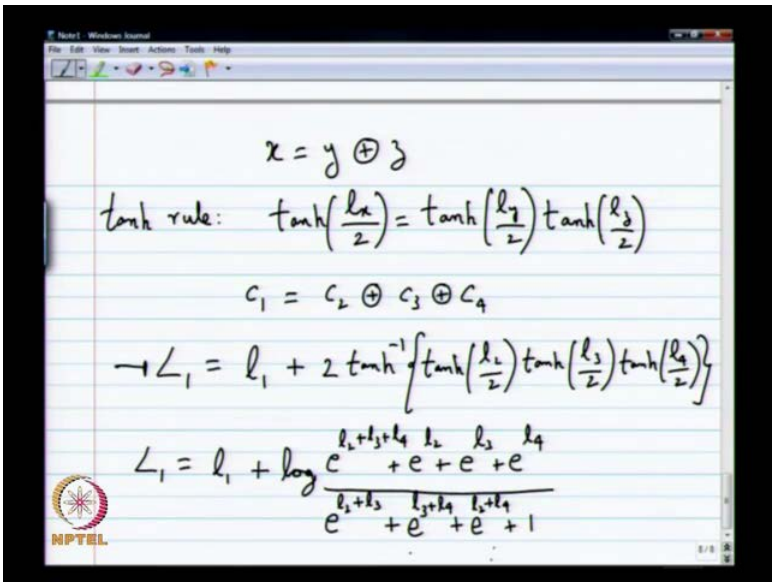
Remember this times probability of z equal to 0 minus probability of z equal to 1 divided by probability of z equal to 0 plus probability of z equal to 1. So, what is the next trick to

get to likelihood ratios, so you have to divide numerator and denominator by probability of x equal to 1. Then you get some get towards likelihood, so if you call let us say l x as log of probability that x equal to 0 divided by probability that x equal to 1 and similarly for l y and l z. If you do this then it turns out you can write this same expression as e power l x minus 1 divided by e power l x plus 1 equals e power l y minus 1 divided by e power l y plus 1 times e power l z minus 1 divided by e power l z plus 1 is that alright.

So, we can now further do some simplifications for instance might notice that there is some kind of a tan hyperbolic hidden in this right. Can you I do not know if you can see that what is tan hyperbolic of x, it is, yes. So, you can write it as see remember sine hyperbolic x is e power x minus e power minus x by 2 and cos hyperbolic x is e power x plus e power minus x by 2.

So, no if I divide that 2 I get tan hyperbolic x, now that that if you cross multiply by multiplied by e power x or something. You will get e power 2x minus 1 by e power 2x plus 1. So, this becomes tan hyperbolic of l x by 2 this becomes tan hyperbolic of l y by 2 this becomes tan hyperbolic of l z by 2, so essentially what this says is.

(Refer Slide Time: 40:30)



If you have an equation of the form x equal y xor z and then your given probabilities for only y and z from that if you compute a probability for x. Those three probabilities p x p y and p z remember this is some kind of an extrinsic probability for x for y and z. Somehow somebody tells you probability using that you compute the probability for x, if

you do that those three probabilities follow the tan hyperbolic rule it is called the tan hyperbolic rule.

Tan h rule basically tan hyperbolic of the log likelihood ratio of x by 2 is equal to the tan hyperbolic of l y by 2 times tan hyperbolic of l z by 2. Now, why is this form better any better and any of the previous forms that we have had remember. Now, if I go back to my equation what is my original equation that I started with C 1 equals C 2 xor C 3 xor C 4 I have a received vector r 1 r 2 r 3 r 4 r 1 is telling me something about C that I C 1 that I can easily compute like log likelihood ratios to r 1 by sigma square. There is no problem the trick was in the computing the extrinsic part.

What are r 2 r 3 and r 4 telling me about C 1, now anything they tell me should be through this equation. Now, r 2 tells me something about C 2 I know that r 3 tells me something about C 3 and r 4 tells me something about C 4. So, somebody is giving me these three probabilities for C 2 C 3 and C 4. Now, I want to figure out the extrinsic probability that I get from these three probabilities about C 1. Once i get that that is going to be independent of anything I compute with r 1 can simply either add the l l r s or multiply the probabilities. And get the final answer is that make sense, so that is the high level idea here. So, why is the tan h rule important in all these, so it turns out you can extended very easily. So, remember if I have x equals y xor z I was able to do it and if it is y xor z xor w.

Still I will have to do is add another tan hyperbolic of l w by 2. So, that is the nice thing about the tan hyperbolic rule it is just extends very easily after all y xor z is another random variable which is binary. So, simply keep extending this, so it turns out the extrinsic information the extrinsic information about the bit 1. So, remember when I write capital L 1, so how will I write capital L 1 and write it as small l 1 plus the extrinsic information. And what is the only extrinsic information that is possible that comes through this equation and what is the extrinsic information that comes through this equation. The solution of this, so how will I write that to write it is 2 times tan hyperbolic inverse of what

Tan hyperbolic l 2 by 2 times tan hyperbolic l 3 by 2 times tan hyperbolic l 4 by 2 is that seems good. So, who is going to now verify that this expression is exactly equivalent to

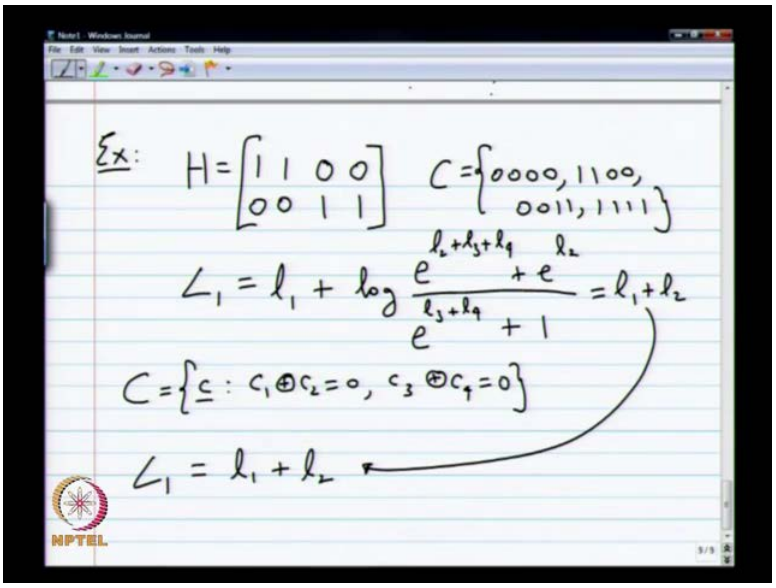
the previous expression we had. Remember which is what is the previous expression, we had this, this was our previous expression right.

You first of all how many of you think that these two are equivalent. Yes maybe it can be is in the log disturbing you a little bit no there is an inverse expression which can be written in terms of log. So, you have to think about it with tan hyperbolic inverse you can write it in terms of log tan. So, there is a way to do that, so it is it is very similar e power see it is a very symmetric kind of expression.

No, if you do the inverse you will get a log there, so the log will come from there and by 2 will also go away very easily. And these two expressions will work out to be exactly the same right yes nobody is. So, but somehow I think this expression is little bit more exciting than the than this expression you do not see anything here I mean where is the where is the code here. So, to speak mean you literally used every code word of the code that is what we have used here on the other hand in this expression you using a parity check condition. The only parity check condition that there is in this code you are using and you get a very nice equation is that. So, the next thing we going to do is a slightly more complicated example.

To just to show that such an idea can mean it is you have to pay attention when you use this idea in general. So, this case for the repetition code it is very easy works out very nicely, but for the other code other codes for even a slightly more complex code. This e this equivalence is very difficult to see, so let us not equivalence this kind of simplification is difficult to. So, let me show you one more example let us see.

(Refer Slide Time: 47:32)



Lets see how of you see anything interesting here, so the next example we will do. So, it is it is I am going to do a slightly tricky example. So, try and stay with me little bit, so it is it will be a kind of like it is it will be a simple example only. But, it will be something which will not be true at the end, so do not be surprise by it.

So, finally we will do something and will show that two things are not the same. So, basically will you have to pay attention is what I will show, so think about that we go on. So, let me take a code define by a certain parity check matrix and I want this parity check matrix to have a very simple structure. So, maybe I will take it to be my God this is just no it just trying me nuts this constant dividing that I have to do to. So, let us say we have a code, so I trying to see if I can get a slightly more non trivial code which is. So, yes maybe I can try this 0 and let us say no let me check it keep this zero then maybe I will keep this as 0 0 1 1.

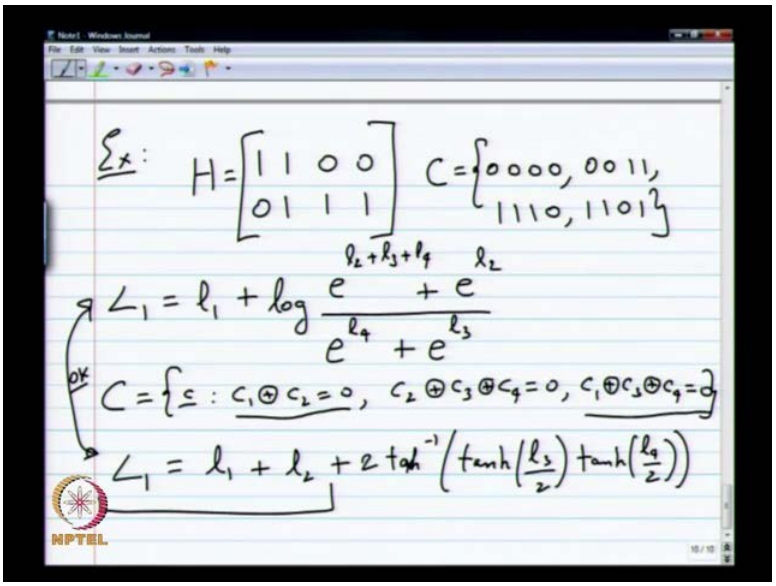
Let me see let me think about this for a while should be, so we will keep this code. So, this is a this is a good enough code the code from this is quite easy right. We have seen this before what is the code this is a parity check matrix which is the code, so you have this is one of those self dual codes right. So, maybe this is this is not so interesting no maybe I should change this a little bit. So, how can I so let us just keep it like this it is we will keep it like this.

So, if you compute l 1 directly what you get capital l 1 you get small l 1 plus log e power l 2 plus l 3 plus l 4 plus e power l 2 divided by e power l 3 plus l 4 plus 1 is that. Now, you might want to argue that the condition imposed by the parity check matrix are what are the conditions imposed by the parity check matrix C 1 xor C 2 equals 0. And then C 3 xor C 4 yes you can do for the thing for this is this is kind of two codes put together no. So, that is why it is like that another reason why that happens anyway, let us not worry about that, so let us let us keep this.

So, suppose I want to repeat this maybe this will work let me see this is simple enough example maybe it will work. Let us see suppose I want to repeat the previous argument what it should what that I should do capital L 1 l 1 plus, so let us let us go back. So, I want to look at capital L 1, so it is going to be small l 1 plus 2 times inverse of what it is just C 2 right.

So, it will be just l 2 and then what will you do, so if you say that is it then that is wrong clearly right. I mean you would not you would not get the expression that you have here what is missing in this. So, this is a this is a good example, so anyway that is fine we got it, so this work out to l 1 plus l 2, which is the same as this, so in this example maybe it is good. So, let me change this example slightly, so I should change this example.

(Refer Slide Time: 52:16)



So, if I if I do the following say for instance if I do if I do something very similar to that. So, that is I think I was not very confident of the previous code it was not it was not a

good example. So, suppose I change this I put a 1 here. So, let us do this it is a little bit more tricky, so the code in this case becomes what all zeros and then what 1 1 0 0 is not a code word 0 0 1 1.

What else triple 1 0 yes that is there and then 1 1 0 1 that is also there. So, those are the four code words in this code, so let us the same trick as before. Let us see if it works capital L 1 is going to be small l 1 plus log e power the first term is always going to be there l 2 plus l 3 plus l 4. Then the next term is going to be e power l 2 it is no change here. But, the denominator changes denominator becomes e power e power l 4 plus e power l 3, so it would not work I think we will see.

So, you will see there is a way to fix it I will also show you the way to fix it. So, you will see that there is you have to pay attention with this independence stuff you have to be very careful with what the dual code means. So, once let us see if we get carried away and we say C is the set of all code word C is not getting carried away it is very true C 1 xor C 2 is 0. And then C xor C 3 xor C 4 is 0 and if you want to do capital L one you are going to say small l 1 plus small l 2. I am kind of stop there and clearly this is not true what did we miss, so there is some implied dependence between through C 2 right. See l 1 see C 1 depends on C 2 and then you are saying C 2 C 3 C 4 depends, but actually C 1 also through C 2 depends on C 3 and C 4

So, it turns out this linear independence which is kind of sufficient to define the entire code does not fully capture this statistical independence between the code word bits. So, just because you have a set of linearly independent vectors you cannot just stop that stop there. You have to take all possible linear combinations of the bases vectors and take every single parity check that comes from every single check. Because linear independence what does it statistical independence you might still have dependence through that. For instance there is one more code word here in the dual what is what are the four code words in the dual 0 0 0 0 1 1 0 0 0 1 1 1.

What is the other code word in the dual 1 0 1 1 right 1 0 1 1 is another code word in the dual. So, in the definition of my code I really do not have to write that I do not have to really write 1 xor C 3 xor C 4 equal to 0. Because anywhere linear independence implies that is true it would not give me any new condition. But, when I write statistical independent assumption like this you have to account for the for this guy also you cannot

just generally ignore that one. So, C 1 plus C 2 is 0 and you also have C 1 xor C 3 xor C 4 equals 0. So, clearly r 2 will tell me something about C 1 and r 3 and r 4 also tells me something about C 1. I cannot just stop that based on linear independence I have to look at every single dual code word and collect all the information that coming through from these things.

But, luckily here the way these things has worked out the extrinsic information I get from this equation is statistically independent of the extrinsic information I get from these why is that because r 2 is involved here and r 2 is not involved there. So, that is only r 3 and r 4, so I can clearly add there is no problem add the log likelihood ratios or multiply the probabilities I can do that. So, if I add the third one also I should get I should get this expression original expression will I get that yes I will get that right. So, you can 2 tan inverse tan hyperbolic inverse yes.

So, you can do that and you will see that will agree with the expression that we had you will see these two guys agree they are is that. But, never the less I want to point out one thing to you if you want to simplify complexity here. It is not really any simplification or complexity if four guys here you can probably stop here I know I have only so many parity checks at hand I will only compute whatever it is. That I know about my first bit using only these parity checks if I want to define it for the what will I do I will add more parity checks. And see if that gives me something more independent, so this is a crucial idea even though it comes like burden to you right.

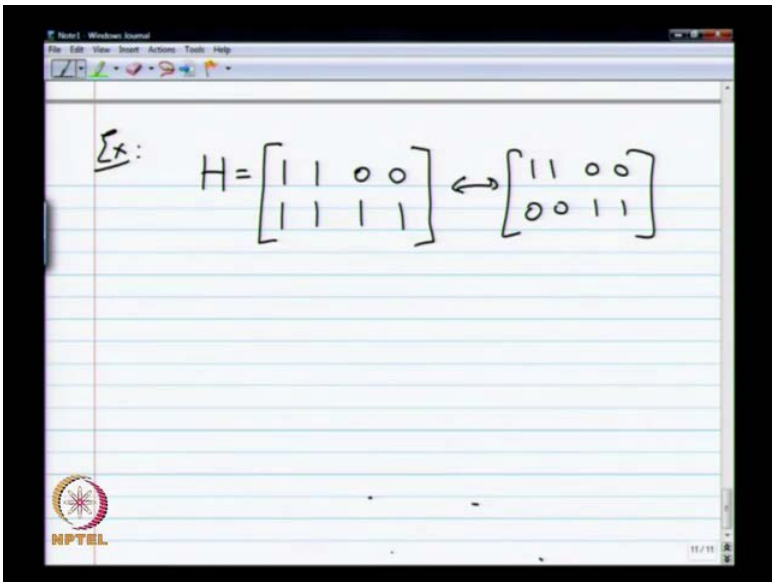
One more term which you have to account for it is a crucial idea and sub optimal soft decoders you look at as many parity checks that is you can handle. So, then you can compute it then try and inclemently add additional parity checks. But, the only crucial thing is if there is this independence that holds then it is very easy to add the new information that is coming in. However if this independence does not hold you do not know how to combine really, so guess what people will do in approximate soft decoders. They will assume independent even if it is dependent and keep adding. So, hopefully this silly exercise we are doing with these computations is telling you something at high level.

The important idea that I am trying to convey here is you can look at the bitwise MAP expression from the dual code point of view every parity check. That you have not

necessarily linear linearly independent ones even the linearly dependent parity checks can give you additional extrinsic information about the particular bit. The only condition is that bit should be involved in that parity check if of course, you have C 2 C 3 C 4 is 0 it is not going to tell you anything about C 1 the… But, the bit itself should be involved in the parity check every time the bit is involved in the parity check, you get some additional information if it is independent then it is very easy to account for you simply keep adding everything. You get if it is not independent then it is difficult in general, but one approximate thing to do is to simply assume independence.

And add and how do you check for independence between two different parity checks they should not have any other received value in common. They can have of course C 1 in common when they will have C 1 in common. But, nothing else should be common if it is not common then necessary definitely it is independent and if it is common. Then it is not independent, but in a sub optimal decoder you might want to assume independence and keep adding and see what happens it is the idea. So, I want to show one example where it not be independent, but when you add maybe you will get some expression which looks reasonable close. So, who knows so let me try and cook up that example it is a little bit more difficult to come up with that example.

(Refer Slide Time: 01:00:42)

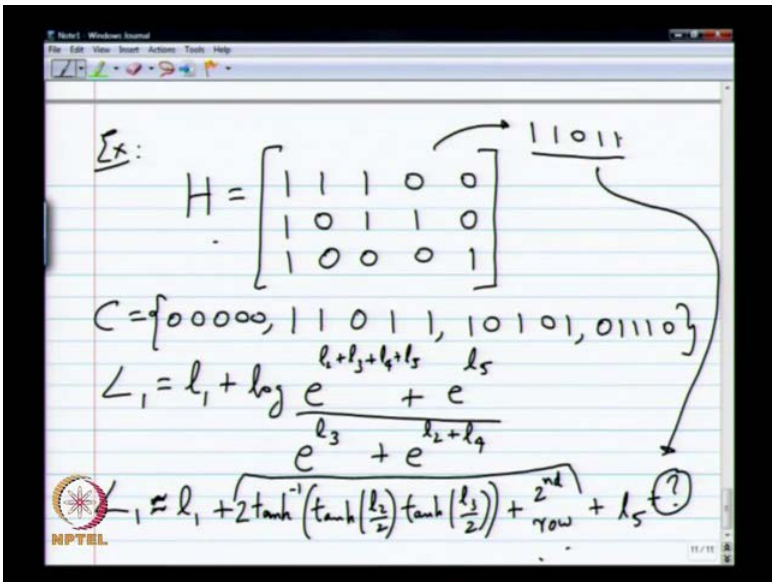


So, let us see H what can we do for this example. Let us try this will this work is this work no actually it do not work you know. Let us keep it like this let us keep it like this

let us keep it like this maybe this points this gives us something interesting to about. So, this actually is very equivalent to 1 1 0 0 0 0 1 1 and you work with this parity check and it everything seems to work fine, but if you directly work with this one also that might actually work no. So, maybe so maybe I this is not a good example, so maybe yes maybe you need more rows. And that is not a that is too painful, so maybe I will erase one column what will happen if I do this.

Yes, third column is always 0, so maybe it is not a good example. But, I want to give you an example where you have dependence, so maybe we will take up larger example. And then we will do approximate only will let us not the optimal thing we will just do the approximate thing and then see what happens ok.

(Refer Slide Time: 01:02:27)



So, let us take a slightly larger example, so it is a bit painful. So, let us try this 1 1 1 let me try 0 0. So, we will try five columns and then let us try this let us try this I mean let see where it goes. So, maybe let us just try this it is a painful code it has just four code words again right. So, it is not so difficult to write this code down, so what will be this code the all zero code word code. And then 1 1 this one is a code word 1 0 1 0 1 and what about 0 1 1 1 0, so that is correct.

Those are the four code words, so if you want to write capital L 1, actually this is a bad example because the code itself has only four code words. And you are trying to go to the dual code which has eight code words. So, maybe it is a good example but anyway so

let us let us let us see this just to make sure that you are to be careful when there is dependence. So, the expression might be totally different between the actual answer you will get and the wrong answer you get by assuming independence. So, I want to show that l 1 is quite simple here small l 1 plus log e power l 2 plus l 3 plus 4 plus l 5 plus you will have only one other guy which will be e power l 5 am I right.

The denominator you are going to have e power l 3 plus e power l 2 plus l 4. So, by now I think enough people must be getting enough experience with reading this expressions down it is it is quite easy right. It is not very difficult just look at it and you can write it down, now if you were to try our approach the initial cut of our approach. The first approach would be the capital L 1 equals small l 1 plus look at the first parity check that first parity check is going to give you something involving l 2 and l 3 am I right. So, it is going to be the 2 tan hyperbolic inverse, so if you were to do the first parity check it is going to be tan hyperbolic l 2 by 2 tan hyperbolic l 3 by 2.

So, this is one approximation, so let me put approximate. So, clearly it is not equal so it is an open approximation, but you might you might want to say start with the second row instead of starting with the first row. Then instead of l 2 and l 3 you will have nose l 3 and l 4 or you might want to start with the last row. So, I am going to argue staring with the last row is smart thing to do why is it why is starting with the last row. A smart thing to do well actually a lower weight is better you know. So, you think about it if you have if you have x equals y plus z and if you have y plus z plus w when somebody gives you probability for y z and w you have to compute probability for x.

You have three different things you have add it all up it the value will go down. So, it is all it is only heuristic, but think about how I it works. But, if you have only two things then you get a better probability, so anyway. So, that is a that is a heuristic anyway it is good to think about that of course, it depends on the actual received value also. But, the heuristic is to go for the lower weight vectors first because that gives you really good information which you would not get from the larger weight vectors.

See as you keep on xoring bits they will tend to equally likely bits right. You take you take for instance n bits each likely to be 0 or 1 with probability let say 0.1. But, when you xor all of them together the probability will become very close to half or there will be 1 minus 1 minus p raise to the power something divided by 2. You can check that out,

so as you xor more and more bits the you will come closer and closer to 0.5 which is very bad. You know 0.5 probability is not as good, so you have to go for lower weight code words in the dual. Now, the row is very difficult to account for and if you assume independence there you will get something involving l 3 and l 4 and the fourth row third row is easy to account for. Because that is going to be independent and you get l 5 out and you see even here e power l 5 comes out when you get l 5 plus something involving l 2 l 3and l 4.

But, this summation is clearly different from that function of l 2 l 3and l 4 and it is not very easy to approximate. It that way also it may not even be a good approximation you may not be able to do it. And the only way to do that is to actually look at the original code, I mean you have to really undo all the dependencies without undoing the dependencies. It is hard tough to approximate a good exercise is to actually write this down, simplify it. And see how different it is from the other thing we will see there a lot of things, in fact there is one more. There is one more code word in the dual which also has a 1 in the first position which is what the xor of all three 1 1 0 1 1 that is also in the dual. So, this is also a valid dual code word right, it is the xor of all the three or this also you will get one more contribution and how do account for that that is also tough.

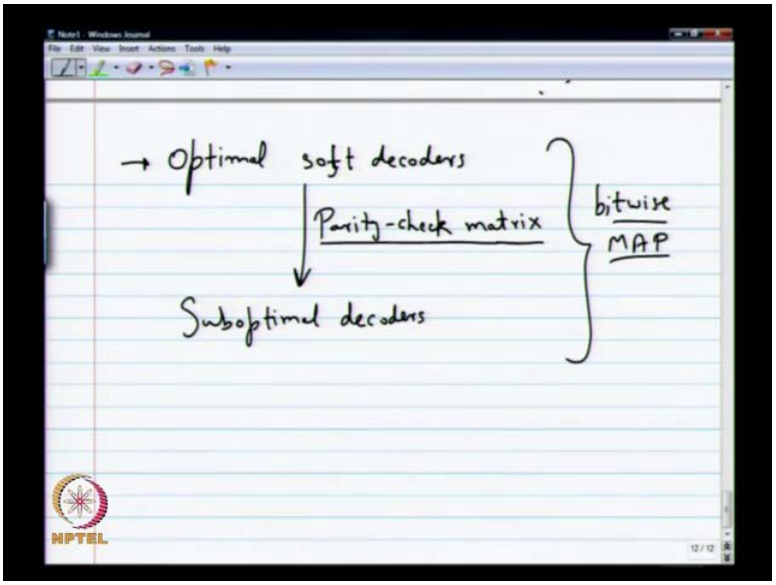
You cannot just add adding is tough was it involves both l 2 and l 4 and l 5 and it is totally dependent on the previous things. You have to undo the dependency there you cannot do this very easily is that. So, this is my example bad example that things do not work out the previous examples things worked out yes. What do you I mean I do not know what to do, so the question was is there any connection between these things and other areas in general. So, there are huge connections of course when, so single processing and these kind of approximate soft ideas lots of connection. And in a in a starting course you mean it is no absolutely no time to look at all those things, so anyway what I want to convey with this.

With these examples mean I have done mostly by example because it is difficult to do any theory here. So, it is all approximate and mostly heuristic the basic idea is instead of thinking about the bitwise MAP expression in terms of the original code words, you think about them using the dual code words. Every parity check you have or every code word in the dual which has a 1 in the in the position you want for instance in C 1 that will be involved in the in the bitwise MAP expression.

It has to be involved somehow you have to make it make it involve as long as the parity checks are independent as they do not have any common received values you can happily add the log likelihood ratios. Whenever they start having some bits in common you run into trouble you really cannot do anything. One approximate thing is to do is to simply assume independence and add that is the standard method that is used in almost all sub optimal soft decoders.

And it really works quite well without too much without too much of a problem and practice question is about how this I do not know it there is well it is. So, the question is basically is there a way to analyze this things that is really no good way to analyze these things. Once you bring in the dependence it is difficult to analyze, so we will see some analyzes assuming independence not assuming with the dependence in the picture ok.

(Refer Slide Time: 01:11:51)



So, to summarize in the couple of minutes that we have we want to move from optimal decoders optimal soft decoders to sub optimal soft decoders and the crucial thing here is the parity check matrix. So, parity check matrix is used in one specific type of sub optimal decoders. But, that is the kind that I will be talking about mostly, so parity check matrix plays a big role not just the parity check matrix. But, also linear combinations of the rows of the parity check matrix also matter.

So, how they interact with how they overlap with each other kind of plays a role in this. So, all these it turns out is nice and easy for the bitwise MAP decoder and I should

confess that I really do not know if such a method exists for the ML decoder. So, I do not know I mean may maybe there is when I am I have not educated myself on it. But, I think that there should exists, but maybe the language is little bit more difficult maybe some and word and some decoding etcetera and all. But, it is not as simple as the bitwise MA decoder, that is I said the bitwise MAP decoder is seeing being is much more popular in the in the soft decoding idea than ML decoders. So, people use bitwise MAP and it works out works out alright, so we will stop here with this lecture and will take a break and continue next.