LDPC and Polar codes in 5G Standard

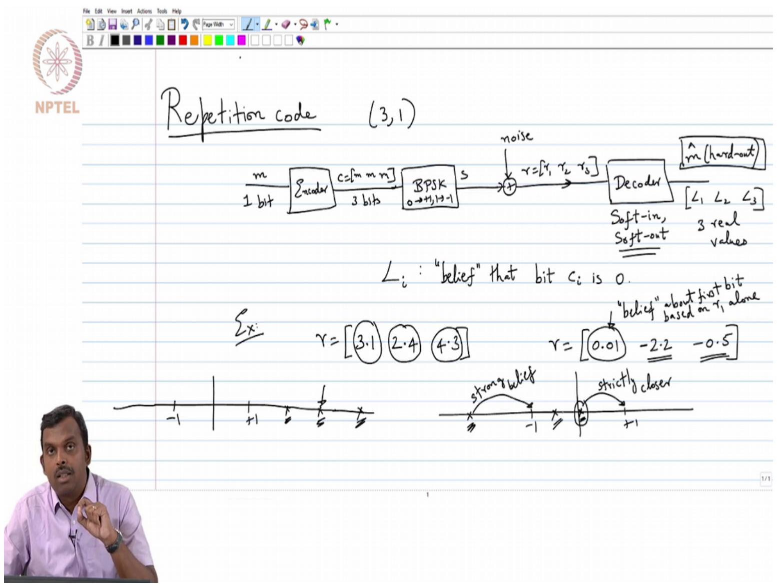
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology Madras

Log-Likelihood Ratio and Soft Input and Soft Output (SISO) Decoder for the Reception Code

Hello and welcome to this lecture on LDPC codes okay so from this week onwards we will start looking at low density parity check codes in particular how the low-density parity check codes are specified in the 5G standard, we will only use those codes and then also how to decode them. We will build up the decode slowly and the decoder for the LDPC codes has 2 important ingredients, the overall decoder is made up of 2 smaller decoders used repeatedly and combined in an interesting way, so the basic decoders that are used in the LDPC decoder is the decoder for a single parity check code and the decoder for the repetition code, okay.

So we have seen the repetition code in some detail, we have not talked about the single parity check code that much, I will talk about it in this 1st lecture and in particular there is a way to decode both these codes which are both simple and at the same time little bit different from what we have seen so far, so this lecture mainly I am going to focus on these 2 codes the repetition code and the single parity check code and how to build this simple decoder of both of them, okay.

(Refer Slide Time: 1:47)



So let me start with the repetition code okay so particular I will take the 3, 1 repetition code and we will look at it with BPSK modulation over AWGN as we have been doing so far and I will talk about slightly different decoder, it is sort of similar to them ML decoder it does not do very different from that but we will think about it a bit differently, that is all, okay. So let us get started, so how does this work once again you have a message bit m okay it goes through the encoder, produce a code word which is m, m, m okay.

So this is 3 bits, then you do BPSK we all know what it is by now 0 to plus 1, 1 to minus 1 and you get a symbol vector S okay and then this goes through noise and get a receive vector r which is r1, r2, r3 okay and the decoder has the task of producing m hat or c hat okay so what we will consider in this lecture is a soft in soft out decoder okay so this is the 1st modification in our decoder, so what do I mean by soft in? We already saw soft decoders before which taken the real value r1, r2, r3 and then produce the output but their output was only m hat right, so traditionally we have been thinking of the decoder as putting out m hat this is a hard output okay, so it tells you what the estimated bit was okay.

So we will not be interested in m hat in this soft output decoder we will be interested in some sort of probability as the output okay probability or belief or there are various words that people use for a soft output not only do I want to say what the bit is but I also want to say how much do I believe in that value for that bit okay so that is the idea behind this decoder. So this is hard output we will not be considering that in this. What we will want her is a soft output, so what I will put out here is l1, l2, l3 okay there are 3 real values that will come out of this decoder okay. What are each of these values Li is let us say believe I will put this in quotes here I will define it a bit more precisely later on, belief that bit Ci is let us say 0 okay.

So I want some soft output some real values to come out okay and that is this definition of Li and I want to say that for each of the rates l1, l2, l3 what is my belief that the 1st bit is 0, what is the belief 2nd bit is 0, what is the belief that the 3rd bit is 0. So actually for the repetition code it will turn out that all the 3 values at the same, so eventually you will get the same value for all 3 and it does make sense but you will see there are some subtleties in this okay. So let me motivate this with some examples so it is always good to look at examples, so examples I will focus on the decoder so let us say your R was… Let me take a simple little example here 3, 2 I will take it as a real numbers just to look at little bit more 2.4, 4.3 okay.

So I get 3 values all 3 are positive and all 3 are fairly large okay, so if you look at it for a while it is clear what these values are, so once again remember what is the line here? So we have this transmission which is either plus 1 or minus 1 and noise gets added and what I am receiving here is 3.1, 2.4, 4.3 are all way out here, right? So these are the 3 receive values okay so now what is this 3.1 represent okay, so the fact that it is 3.1 means I have a certain soft input about what that 1st bit could have been okay I have transmitted the 1st symbol which could have been plus 1 or minus 1 and I received the 3.1 okay so let us say it is here 3.1 okay, so if you look at 3.1 and if you receive 3.1 it is very likely that the transmitted symbol was plus 1, so you belief that the transmitted symbol was plus 1 is very high, okay.

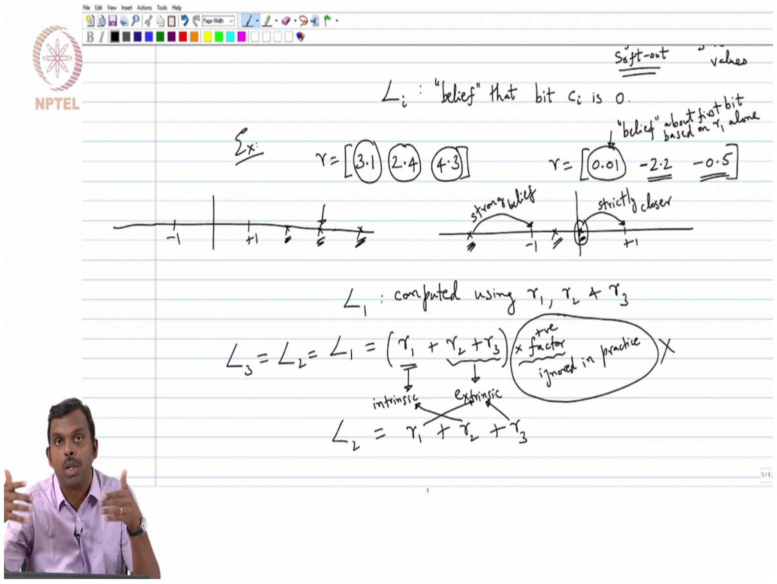
Same thing with 2.4, so you got 2.4 is somewhere here you really believe that the transmitted symbol is plus 1 and you also have 4.3 and that is way out here and that is also very strong indicator that the transmitted symbol was plus 1 okay so your quite confident about that. On the other hand let us look at another possibility here may be r is 0.01 right, minus 2.2 and then may be minus 0.5 okay, so this is another case here and let me draw that same line once again here and you will see what is going on here is slightly different, so you have plus 1 minus 1. You have got something really close to 0, so this is 0.01 so this is really close to 0 then you got something far away and then something intermediate okay, so now if you look at 0.01 okay it is really I mean of course if you do a hard threshold you going to say the transmitted symbol was plus 1 okay.

The hard threshold will go towards this, this is strictly closer okay but my confidence in saying that the transmitted symbol was plus 1 is not very high just looking at 0.01 it is not very high okay. So I not know whether it was plus 1 or minus 1 because it is just 0.01 of okay so in that sense 0.01 also represents a belief about 1st bit based on r1 alone right? Okay you have received r1 that is 0.01 so if I am not given any other information that is the best believe I have okay. On the other hand if you think about it very carefully the same bit was transmitted 3 times, right?

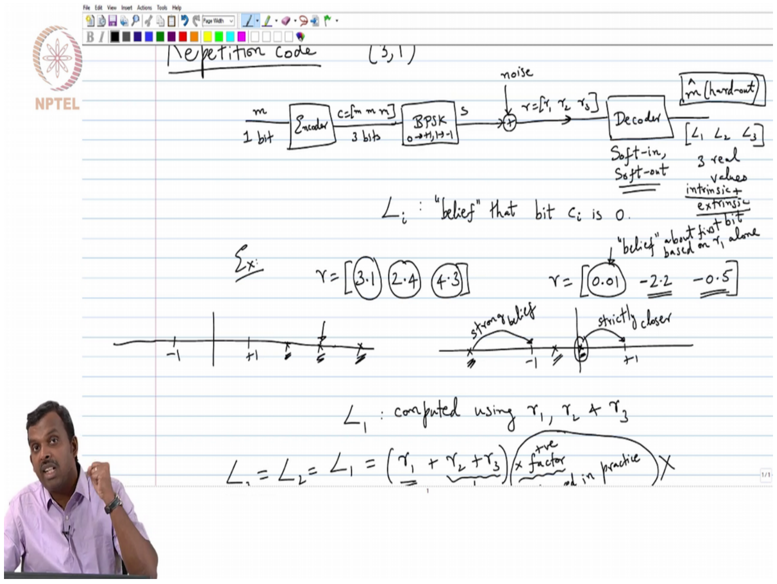
The same bit was transmitted once again and I got minus 2.2 which means if I have received it here my belief is quite strong about what the transmitted symbol could have been it has to go to this bit, so this is a strong belief okay. Really it could not have been plus 1 it would not have had such a huge noise, the noise is most likely to have been minus 1 okay the probability is worth that remember the Gaussian exponentially so as you go further away the probability is really dropped significantly, so minus 2.2 is a very strong indicator that it is minus 1 and remember from my code I know that I transmitted the same bit once again okay.

So to find a belief about the 1st bit not only should I use r1 I should also use r2 and even r3 okay so now r3 you might say minus 0.5 was not as strong as minus 2.2 is a little bit weaker than minus 2.2, it says it is minus 1 it is much better than 0.01 but still it could have been plus one also I mean it is not so far away from plus 1 but nevertheless it is still a stronger belief than 0.01, so when you want to decide about the 1st bit you want to take into account r1 also what r2 and r3 tell you about the 1st bit particularly because the same bit was transmitted again and again and again this is a repetition code, if the bit was 0 you transmitted plus 1, plus 1, plus 1, if the bit was 1 few transmitted minus 1, minus 1, minus 1 and all 3 were received for that okay, so you want to use r1 as well as r2 and r3 to compute the belief for the 1st bit okay.

(Refer Slide Time: 10:52)



Soft-out
values
L_i : "belief" that bit c_i is 0.
Ex:
$r = [3.1 \; 2.4 \; 4.3]$
"belief" about first bit based on r_1 alone
$r = [0.01 \; -2.2 \; -0.5]$
strong belief
strictly closer
-1
+1
L_1 : computed using $r_1, r_2 + r_3$
$L_3 = L_2 = L_1 = (r_1 + r_2 + r_3)$
× +ve factor ignored in practice
intrinsic
extrinsic
$L_2 = r_1 + r_2 + r_3$
NPTEL



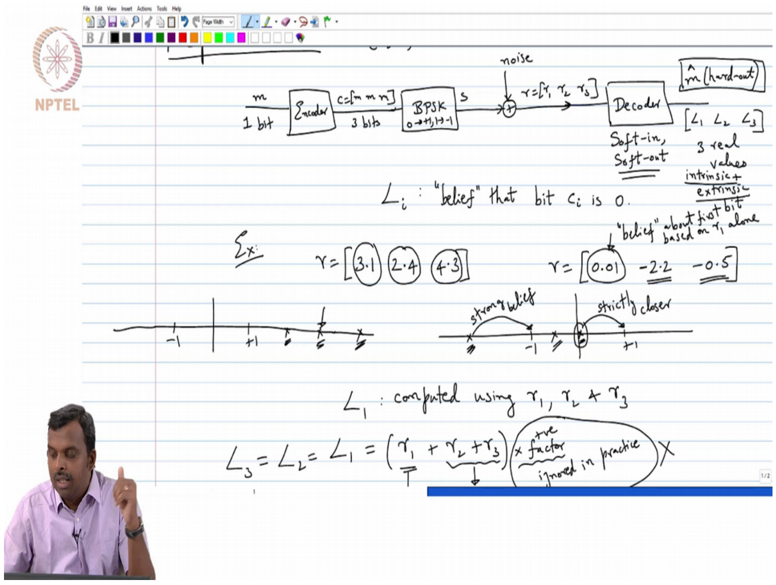
Repetition Code (3,1)
m
1 bit
Encoder
c=[m m m]
3 bits
BPSK
0→+1, 1→-1
s
noise
$r = [r_1 \; r_2 \; r_3]$
Decoder
Soft-in, Soft-out
$\hat{m}$ (hard-out)
$[L_1 \; L_2 \; L_3]$
3 real values
intrinsic + extrinsic
L_i : "belief" that bit c_i is 0.
Ex:
$r = [3.1 \; 2.4 \; 4.3]$
"belief" about first bit based on r_1 alone
$r = [0.01 \; -2.2 \; -0.5]$
strong belief
strictly closer
L_1 : computed using $r_1, r_2 + r_3$
$L_3 = L_2 = L_1 = (r_1 + r_2 + r_3)$
× +ve factor
NPTEL

So in other words if you just look at these examples it is clear that you compute l1 r1, r2 and r3 okay r1 tells you directly what it is for the 1st bit but then the 2nd bit and 3rd bit are also the same so you need to figure out how to combine r1 and r2 and r3 to tell you about l1 okay, so now how to do this combination? There is a lot of intuition that one could use and may be directly decide. It turns out… I was given the answer 1st and then I will tell you how to derive this, so it turns out the correct way to compute this is l1 equals r1 plus r2 plus r3 maybe I will put… You might want to scale it with something okay multiply by some factor okay so theoretically putting that factor there is good but in practice people just ignore this factor okay.

So ignored in this so you can just say r1 plus r2 plus r3 you cannot lose much by that factor it is a positive factor okay, so you do not lose much by ignoring that factor, okay. So this is the answer and this is the actual competition that you have to do to find out the total belief about the 1st bit based on r1 and r2 and r3 all 3 of them are being used, so if you look at it, it has 2 components one case what is called intrinsic, these 2 what are called extrinsic okay, so once again like a said we will completely ignore this factor, we will not use this at all will only use r1 plus r2 plus r3 okay and even that there are 2 components one is the intrinsic belief that you have based on what you received for that symbol r1 directly and then extrinsic what comes from r2 and r3 and why do I have to add these 3 one can do a small probabilistic derivation here I am going to do it very quickly but the moral of the story is you can see intrinsically you can see intuitively why this is r1 plus r2 plus r3, right? So the same bit was transmitted 3 times so you are allowed to add the 3 received values to get the total belief for every bit.

In fact this same for l2 also and l3 is also the same okay except that the intrinsic and extrinsic will differ, right? So if you look at l2 you have r1 plus r2 plus r3 okay, so this is the intrinsic part these 2 the extrinsic part okay likewise for l3 r3 will be the intrinsic part r1 plus r2 will be the extrinsic part, so what does intrinsic and what extrinsic changes in l2 and l3 but the overall expression is the same r1 plus r2 plus r3 okay, so this is an example of a soft in soft out decoder for the repetition code okay, so the soft in soft out decoder puts out 3 real values okay intrinsic plus extrinsic, this intrinsic extrinsic division is quite important okay, so the intrinsic one is what you receive from the Channel corresponding to that particular symbol extrinsic is what you gain or clean out of the other received values about this particle symbol.

(Refer Slide Time: 14:35)



$$\begin{tikzcd} m \text{ (1 bit)} \arrow[r] & \boxed{\text{Encoder}} \arrow[r, "c=[m\ m\ m]"', "\text{3 bits}"] & \boxed{\text{BPSK } 0\to+1,\ 1\to-1} \arrow[r, "s"] & \oplus \arrow[r, "r=[r_1\ r_2\ r_3]"] & \boxed{\text{Decoder}} \arrow[r] & \boxed{\hat{m}\text{ (hard-out)}} \\ & & & \text{noise} \arrow[u] & & \end{tikzcd}$$

Decoder: Soft-in, Soft-out

$[L_1\ L_2\ L_3]$ 3 real values intrinsic + extrinsic

L_i : "belief" that bit c_i is 0.

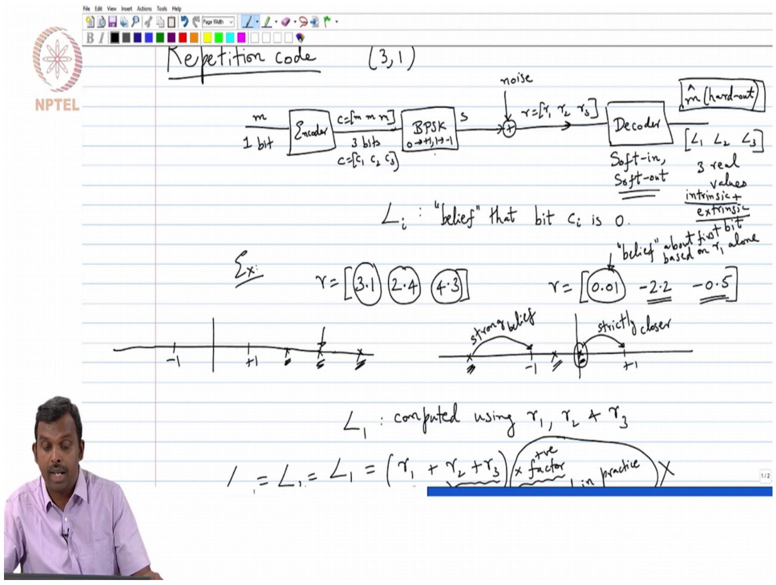
Ex: $r = [\,3.1\ \ 2.4\ \ 4.3\,]$ $\qquad$ $r = [\,0.01\ \ -2.2\ \ -0.5\,]$

"belief" about first bit based on r_1 alone

strong belief $\qquad$ strictly closer

L_1 : computed using r_1, r_2 & r_3

$L_3 = L_2 = L_1 = (r_1 + r_2 + r_3) \times$ +ve factor ignored in practice



Repetition code $(3,1)$

$$\begin{tikzcd} m \text{ (1 bit)} \arrow[r] & \boxed{\text{Encoder}} \arrow[r, "c=[m\ m\ m]"', "\text{3 bits}"] & \boxed{\text{BPSK } 0\to+1,\ 1\to-1} \arrow[r, "s"] & \oplus \arrow[r, "r=[r_1\ r_2\ r_3]"] & \boxed{\text{Decoder}} \arrow[r] & \boxed{\hat{m}\text{ (hard-out)}} \\ & & & \text{noise} \arrow[u] & & \end{tikzcd}$$

$c = [c_1\ c_2\ c_3]$

Decoder: Soft-in, Soft-out

$[L_1\ L_2\ L_3]$ 3 real values intrinsic + extrinsic

L_i : "belief" that bit c_i is 0.

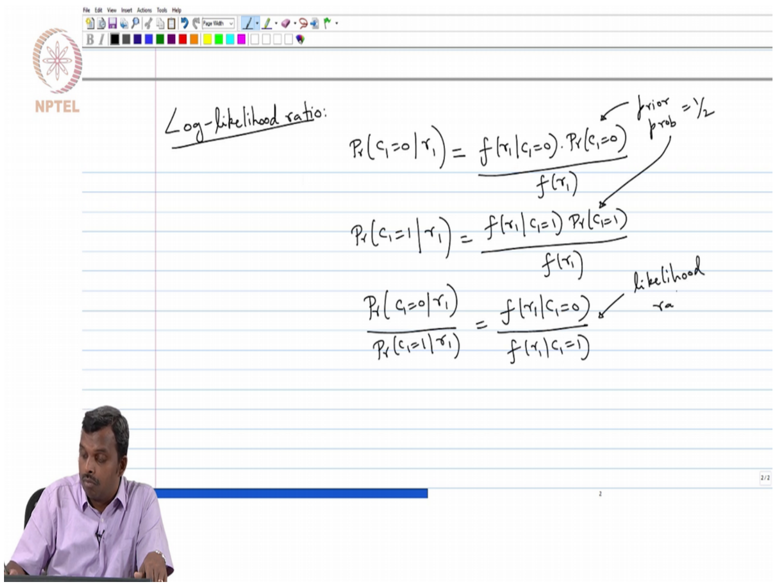
Ex: $r = [\,3.1\ \ 2.4\ \ 4.3\,]$ $\qquad$ $r = [\,0.01\ \ -2.2\ \ -0.5\,]$

"belief" about first bit based on r_1 alone

strong belief $\qquad$ strictly closer

L_1 : computed using r_1, r_2 & r_3

$L_3 = L_2 = L_1 = (r_1 + r_2 + r_3) \times$ +ve factor in practice



Log-likelihood ratio:

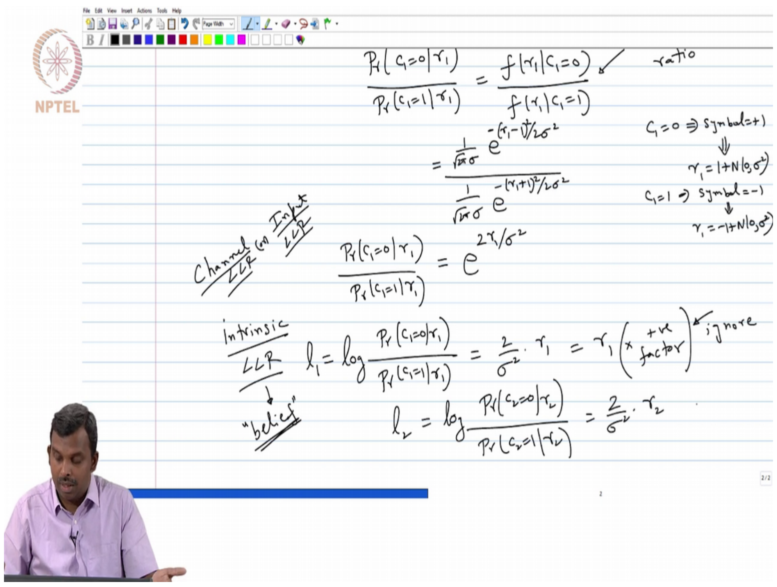
$$\Pr(c_1=0 \mid r_1) = \frac{f(r_1 \mid c_1=0)\cdot \Pr(c_1=0)}{f(r_1)}$$

prior prob $= 1/2$

$$\Pr(c_1=1 \mid r_1) = \frac{f(r_1 \mid c_1=1)\, \Pr(c_1=1)}{f(r_1)}$$

$$\frac{\Pr(c_1=0 \mid r_1)}{\Pr(c_1=1 \mid r_1)} = \frac{f(r_1 \mid c_1=0)}{f(r_1 \mid c_1=1)}$$

likelihood ra



$$\frac{Pr(c_1=0|r_1)}{Pr(c_1=1|r_1)} = \frac{f(r_1|c_1=0)}{f(r_1|c_1=1)} \quad \text{ratio}$$

$$= \frac{\frac{1}{\sqrt{2\pi}\sigma} e^{-(r_1-1)^2/2\sigma^2}}{\frac{1}{\sqrt{2\pi}\sigma} e^{-(r_1+1)^2/2\sigma^2}}$$

$c_1=0 \Rightarrow \text{symbol}=+1 \Downarrow r_1 = 1+N(0,\sigma^2)$

$c_1=1 \Rightarrow \text{symbol}=-1 \Downarrow r_1 = -1+N(0,\sigma^2)$

Channel LLR (or) Input LLR

$$\frac{Pr(c_1=0|r_1)}{Pr(c_1=1|r_1)} = e^{2r_1/\sigma^2}$$

intrinsic LLR → "belief"

$$l_1 = \log\frac{Pr(c_1=0|r_1)}{Pr(c_1=1|r_1)} = \frac{2}{\sigma^2}\cdot r_1 = r_1\left(\times \begin{matrix}+ve\\ \text{factor}\end{matrix}\right) \text{ ignore}$$

$$l_2 = \log\frac{Pr(c_2=0|r_2)}{Pr(c_2=1|r_2)} = \frac{2}{\sigma^2}\cdot r_2$$

So I am going to do a quick derivation here to show you how this comes from the main idea is this notion of a log likelihood ratio okay, so once again I want to briefly comment if you go back and look at the decoder those of you who have implemented and the ML decoder for repetition code and we have seen it we will quickly recognize this r1 plus r2 plus r3 also showed in the ML decoder and this is exactly like the ML decoder okay so what is it that we are doing which is different okay so what is different about the soft output like this point it is not very apparent for this code is not very obvious it seems like an unnecessary thing we are doing putting out soft output okay but when we combine this as part of the LDPC decoder you will see where this intuition for addition will come from.

So I will play on this little bit later on as well but for now let us see a quick definition involving log likelihood ratio and all that, so once again so this is how log likelihood ratio is defined, so supposing I want to ask this question so the probability that c1 equals 0 given r1 okay so remember once again what is the setting here? The setting is this c1, C2, C3, right? C is c1, C2, C3 the same 3 bits transmitted again and again. Now I want to look at r1 and ask the question what is the probability that c1 was 0 given r1 okay, so for this you can do this little competition using base rule you will see that this will be the PDF of r1 given c1 is 0 times the probability that c1 is 0 divided by the PDF of r1 okay the density of the received value evaluated at r1, so this is easy enough to write.

Now this is the prior probability and it is equal to half okay why is that c1 is the message itself it could be 0 or 1 with uniform probability okay. Now one can also… So this is the calculation you can do actually you can write down what the density will be given c1 is 0 and what is the density of the total density extra and find this out but usually people use something else here, they use this likelihood ratio so let me write down what that is? You can also write down what is the probability of c1 is 1 given r1 this will be sort of similar to this okay so once again this is also a prior probability and (())(17:21) okay and it is comfortable to take this ratio here.

It gets rid of some of the quantities that you do not really care too much about okay and you notice under the assumption of half probability I simply get the ratio of the likelihood, so this is the post area probabilities and this is the ratio of likelihood. So this is called likelihood ratio okay and one can compute this quite nicely okay so remember if c1 is 0 implies the symbol is plus 1 okay and r1 is 1 plus normal 0 Sigma square if c1 is 1 the symbol is minus 1 and r1 is minus 1 plus normal 0 Sigma square okay. So this ratio will work out as 1 by root 2 pie Sigma this is the normal density with mean on the numerator you will have a mean of plus 1 r1 minus 1 whole square by 2 Sigma square divided by the denominator will have e power r1 plus 1 whole square by 2 Sigma square okay, so one can simplify this.

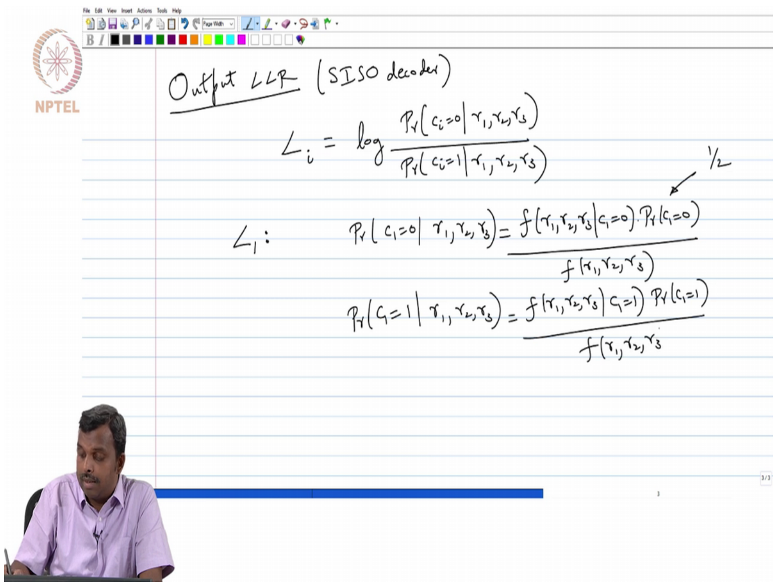
I am going to skip the simplification little bit will get overall e power 2 r1 by Sigma square okay so this is the likelihood ratio probability of c1 equals 0 given r1 divided by probability of c1 equals 1 given r1 okay. The log of this is called the log likelihood ratio and that you can see we will work out as 2 by Sigma square times r1 okay. So this is the log likelihood ratio and this is the... So this is the intrinsic LLR okay. So you can say it is proportional to r1 this is r1 into a positive factor okay r1 is the most critical component 2 by Sigma square actually depends on the Channel value, the Sigma channel noise variance if it is small 2 by Sigma square is going to become very large.

So at high SNR this factor will be very large at low SNR Sigma has 1 or 2 values like that then the factor is very small okay but nevertheless it is positive factor and in most cases we will usually ignore this factor okay we will use just r1 to represent this LLR okay, so this intrinsic LLR is sort of the belief okay this is the belief that we spoke about, okay so this is just intrinsic LLR it is also referred to as the Channel LLR or input LLR okay so this is something important to remember, so now whatever your modulation scheme may be whatever your Channel maybe okay.

When you use an LDPC code on that channel may be you are doing 64 QM instead of BPSK maybe some other channel like ISI channel or something else you are doing or FDM or whatever it is you are doing you may have some other factors in there. The 1st step before using the LDPC decoder or any other soft in soft out decoder is to compute the intrinsic log likelihood ratio or the Channel log likelihood ratio for a particular bit. So this will be the 1st bock that will come.

In our case we have just BPSK AWGN it is very simple model, input LLR is simply proportional to the received value K is proportional to r1 okay so same thing will happen for r2 as well, C2 probability of C2 equals 0 given r1 divided by probability of C2 equals 1 given r1 will also be… Probability of C2 equal 0 given r2 divided by probability of C2 equal 1 given r2 will be 2 by Sigma square times r2 okay. So sometimes I like to call this small l1 so likewise l2 which is let me write this down just for l2 alone probability of C2 equals 0 given r2 divided by probability of C2 equals 1 given r2 will also work out as 2 by Sigma square times r2 okay, so the same thing for l3 as well one can write down and expression for l3 okay so this is input log likelihood ratio.

(Refer Slide Time: 22:27)



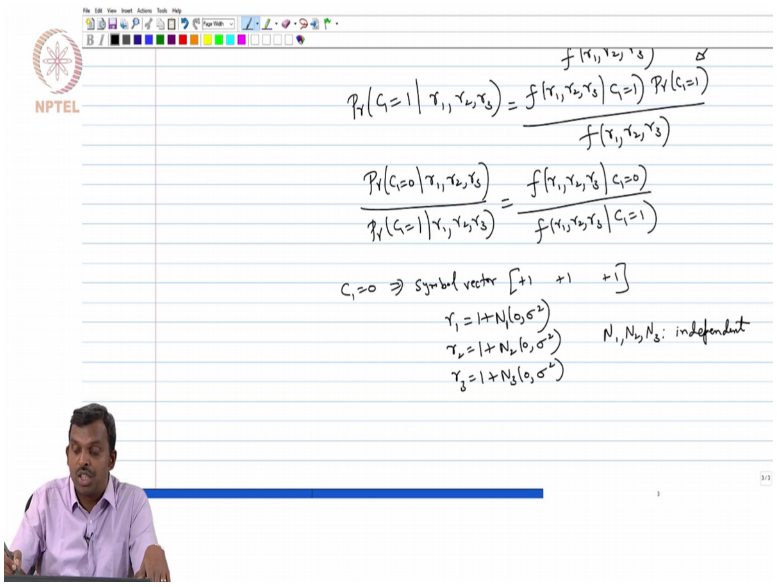
Output LLR (SISO decoder)

$$L_i = \log \frac{\Pr(c_i=0 \mid r_1, r_2, r_3)}{\Pr(c_i=1 \mid r_1, r_2, r_3)}$$

$1/2$

$$L_1: \quad \Pr(c_1=0 \mid r_1, r_2, r_3) = \frac{f(r_1, r_2, r_3 \mid c_1=0) \cdot \Pr(c_1=0)}{f(r_1, r_2, r_3)}$$

$$\Pr(c_1=1 \mid r_1, r_2, r_3) = \frac{f(r_1, r_2, r_3 \mid c_1=1) \Pr(c_1=1)}{f(r_1, r_2, r_3)}$$



$f(r_1, r_2, r_3)$

$$\Pr(c_1=1 \mid r_1, r_2, r_3) = \frac{f(r_1, r_2, r_3 \mid c_1=1)\Pr(c_1=1)}{f(r_1, r_2, r_3)}$$

$$\frac{\Pr(c_1=0 \mid r_1, r_2, r_3)}{\Pr(c_1=1 \mid r_1, r_2, r_3)} = \frac{f(r_1, r_2, r_3 \mid c_1=0)}{f(r_1, r_2, r_3 \mid c_1=1)}$$

$c_1=0 \Rightarrow$ symbol vector $[+1 \quad +1 \quad +1]$

$r_1 = 1 + N_1(0, \sigma^2)$

$r_2 = 1 + N_2(0, \sigma^2)$ $\quad N_1, N_2, N_3$: independent

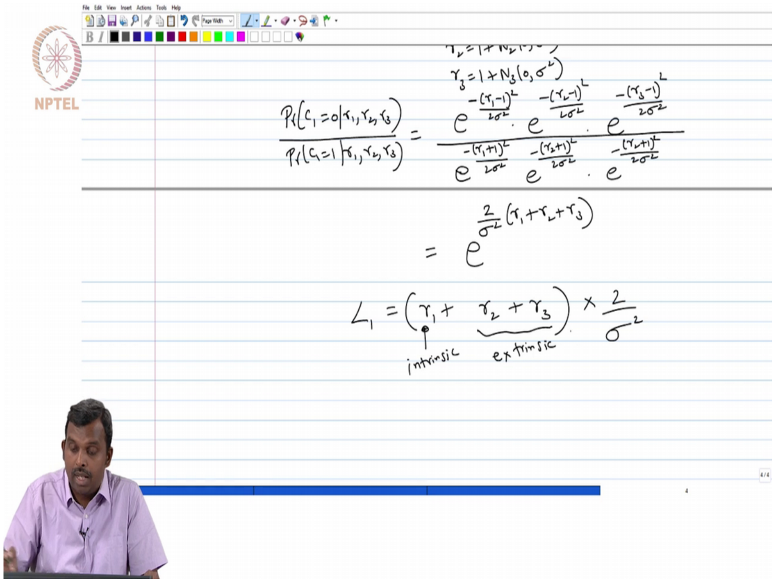
$r_3 = 1 + N_3(0, \sigma^2)$

Now like I said I am interested in a final belief which is a combination of all the 3 received values okay so for that what I need to do is the following okay so output LLR okay in my SISO decoder okay so this capital Li is actually the log of probability of Ci equals 0 given r1, r2, r3 all 3 received values divided by the probability of Ci equals 1 given r1, r2, r3. All the 3 received values have to be taken into account this is the output LLR this is what I really want at the output. What is my final belief after having seen all the 3 received values, so if you have look at the previous calculation it was the Channel LLR or the intrinsic LLR this is the output LLR which is the final belief after I have seen all the 3 received values okay.

So I am going to compute l1 1st you will see the computation of l1 is quite simple everything starts once again with probability of c1 equal 0 given r1, r2, r3 we will notice once again using base rule that this is the joint density given c1 is 0 times the probability of c1 is 0 divided by the overall joint density. So once again this is 1 half equal probability assumption likewise you can also write probability of c1 is 1 given r1, r2, r3 this will be f of r1, r2, r3 given c1 is one times probability of c1 is one divided by f of r1, r2, r3 okay so this is for the repetition code of fairly simple computation if you take the ratio you get probability of c1 equals 0 given r1, r2, r3 divided by probability of c1 this 1 given r1, r2, r3 is simply the likelihood ratio, right?

So it turns out for the repetition code is conditional joint density is easy to write down, so the repetition code makes that very easy. Let me show you how that is done c1 equal 0 implies symbol vector this plus 1 plus 1 plus 1, so that implies r1 equals 1 plus normal 0 Sigma square okay so I will put N1 here r2 is 1 plus N2 0 Sigma square r3 is 1 plus N3 0 Sigma square and N1, N2, N3 are independent okay something similar happens when c1 is plus 1 symbol vector is minus 1, minus 1, minus 1 and r1, r2 and r3 are minus 1 plus 1 normal distribution minus 1 plus another normal distribution minus 1 plus another normal distributed value all 3 are independent because different bits go through independent realisation to the (())(25:36) so once you know this once you know what happens when c1 is 0 and c1 is 1 you can plug in the normal distribution so this is independent conditional on c1 equals 0 this 3 values are independent so you can factor the things and multiply out and write down as a product.

(Refer Slide Time: 25:59)

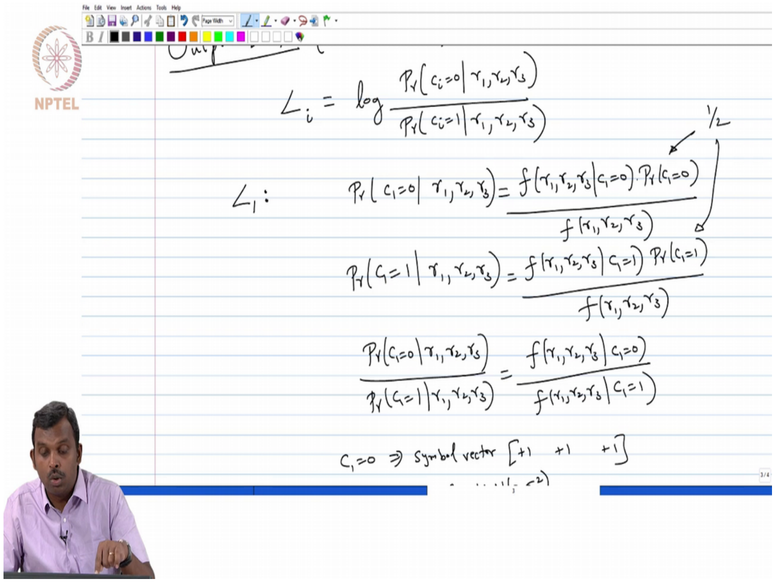


$$r_3 = 1 + N_3(0,\sigma^2)$$

$$\frac{P_r(c_1=0 \mid r_1,r_2,r_3)}{P_r(c_1=1 \mid r_1,r_2,r_3)} = \frac{e^{-\frac{(r_1-1)^2}{2\sigma^2}} \cdot e^{-\frac{(r_2-1)^2}{2\sigma^2}} \cdot e^{-\frac{(r_3-1)^2}{2\sigma^2}}}{e^{-\frac{(r_1+1)^2}{2\sigma^2}} \cdot e^{-\frac{(r_2+1)^2}{2\sigma^2}} \cdot e^{-\frac{(r_2+1)^2}{2\sigma^2}}}$$

$$= e^{\frac{2}{\sigma^2}(r_1+r_2+r_3)}$$

$$L_1 = (\underset{\text{intrinsic}}{r_1} + \underbrace{r_2 + r_3}_{\text{extrinsic}}) \times \frac{2}{\sigma^2}$$



$$L_i = \log \frac{P_r(c_i=0 \mid r_1,r_2,r_3)}{P_r(c_i=1 \mid r_1,r_2,r_3)}$$

L_1:

$$P_r(c_1=0 \mid r_1,r_2,r_3) = \frac{f(r_1,r_2,r_3 \mid c_1=0) \cdot P_r(c_1=0)}{f(r_1,r_2,r_3)}$$

$\frac{1}{2}$

$$P_r(c_1=1 \mid r_1,r_2,r_3) = \frac{f(r_1,r_2,r_3 \mid c_1=1)\, P_r(c_1=1)}{f(r_1,r_2,r_3)}$$

$$\frac{P_r(c_1=0 \mid r_1,r_2,r_3)}{P_r(c_1=1 \mid r_1,r_2,r_3)} = \frac{f(r_1,r_2,r_3 \mid c_1=0)}{f(r_1,r_2,r_3 \mid c_1=1)}$$

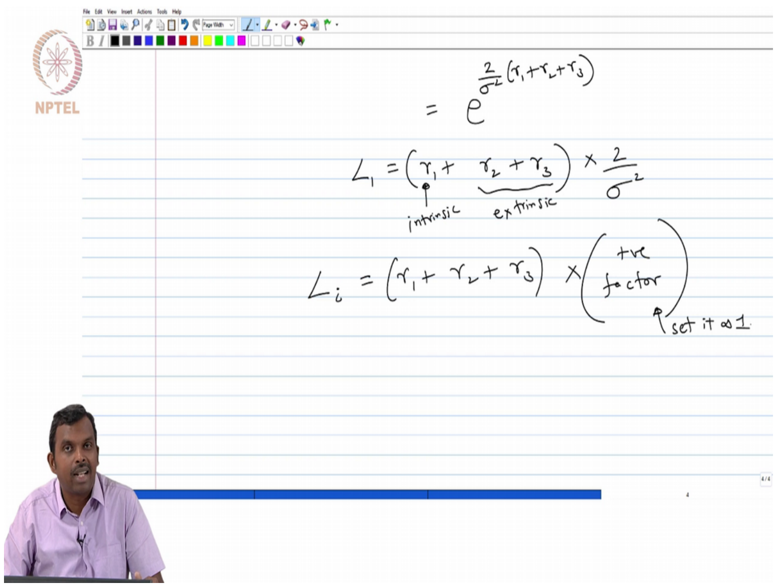
$c_1=0 \Rightarrow$ symbol vector $[+1 \quad +1 \quad +1]$

Let me do that for you here so my probability of c1 equals 0 given r1, r2, r3 divided by probability of c1 equals 1 given r1, r2, r3 is going to be there will be some 1 by root 2 pi Sigma which will cancel on both sides so I am not going to write down that part I will simply write down what happens in the e power down okay minus r1 minus 1 whole square pi 2 Sigma square e power minus r2 minus 1 whole square by 2 Sigma square e power minus r3 minus 1 whole square by 2 Sigma square divided by e power minus r1 plus 1 whole square by 2 Sigma square e power minus r2 plus 1 whole square by 2 Sigma square e power r2 plus 1 whole square by 2 Sigma square okay.

Alright so that was easy enough so notice what happened here the way in which the code words are setup is quite important here that is why this happened you can do a lot of simplification, cancelling, et cetera you will finally get 2 buy Sigma squared times r1 plus… sorry forgot the e power here you will get e power a lot of things will cancel you will get e power 2 by Sigma squared times r1 plus r2 plus r3 okay.

So if you take log on both sides I will get l1 to be 2 by Sigma square maybe I will write 2 by Sigma square on the other side because it is a positive factor does not affect too much you get r1 plus r2 plus r3 times 2 by Sigma square okay so clearly you can identify the extrinsic and intrinsic part now, this is the intrinsic these 2 are extrinsic okay. So go back and look at what I did and if you want to change instead of l1 if you want to put l2 you will get exactly the same calculation so you will see nothing will change eventually l2 will also become the same okay.

(Refer Slide Time: 28:19)



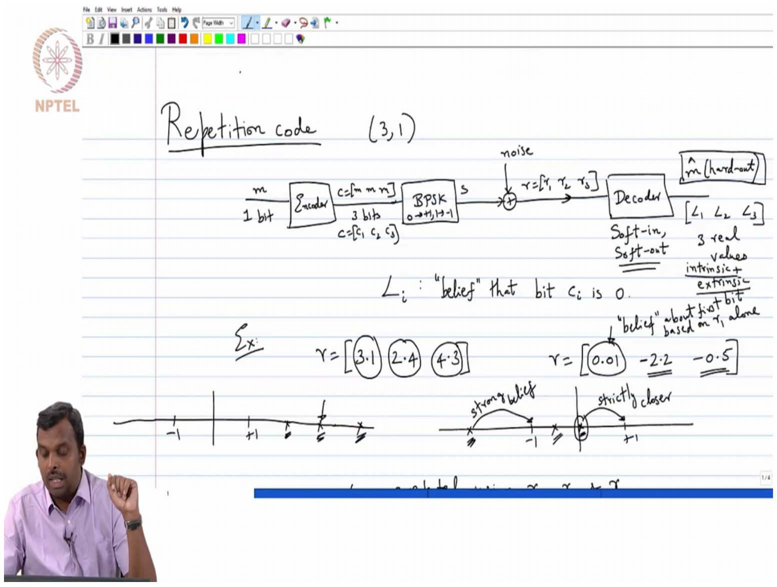
$$= e^{\frac{2}{\sigma^2}(r_1+r_2+r_3)}$$

$$L_1 = (\underset{\text{intrinsic}}{r_1} + \underbrace{r_2+r_3}_{\text{extrinsic}}) \times \frac{2}{\sigma^2}$$

$$L_i = (r_1+r_2+r_3) \times \underset{\text{set it as 1}}{\left(\begin{matrix}+\text{ve}\\ \text{factor}\end{matrix}\right)}$$

So finally one can write down that Li can be taken to be r1 plus r2 plus r3 times some positive factor which we will like I said we will set it as 1 okay so we do not bother about that positive factor. If you want exact log likelihood ratio you have to incorporate that which is usually not very there is not very relevant all you need to do is find out the capital Li. Okay so like I said Li is the same for the repetition code simply add the 3 guys it changes intrinsic and extrinsic changes but nevertheless it is the same okay.

(Refer Slide Time: 29:04)



So with did a quick derivation here and we show how to do soft in soft out decoding for the repetition code, how to put out 3 values and find out soft output for the repetition code hopefully this was easy enough. I am not going to show you the Matlab code for this at this point maybe I will not do it may be later on we will see that okay but what is more important is the single parity check code as well, okay so this is the repetition code we started with that hopefully this is clear.

In the next lecture we will move on to the single parity check code and we will see how to repeat the same thing for the single parity check code. That involves a slightly more complex calculation but we will do that and then after that we will write Matlab code for maybe both the single parity check code and the repetition code okay so like I said these 2 simple decoders are important ingredients in the decoding of healthy PC codes they will play a very important role we will see that later okay so will stop here for this lecture we will pick up with the single parity check code in the next one.