

LDPC and Polar Codes in 5G Standard
n=3 Repetition Code
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology, Madras

So I promised in the previous class that we will take the basic script that we wrote and we will start adding an error control code into it so starting with this lecture we are going to see couple of simple error control codes, I will describe that in the next few minutes, we will first describe the error control codes, sort of informally give a very high level introduction on how these things work and then we will introduce these codes into the script that we wrote for BPSK simulation, see how the whole thing works with the error control code in place as opposed to the encoder situation and we will compare and proceed, etc. okay.

(Refer Slide Time: 00:56)

So we will start with one of the simplest codes it is the n equals 3 repetition code, it is very simple code, the description is out there for you to see we will start with 1 bit message okay so the message is 1 bit long so message could be 0 or 1 right. So message is just 0 or 1, my encoder is going to be a rate one third encoder, 1 by 3 so it means for every 1 bit that comes into the encoder, I am going to put out 3 bits of the code word, this is the codeword, this like I said is the message okay and it is 3 bits long so how am I going to do the encoding?

This is how the encoder works, if the message is 0 the code word is 000, if the message is 1 the code word is 111, so you can see why it is called the repetition code and why it is called

the n equals 3 repetition code, every bit that comes in is repeated 3 times if it is 0 you repeat 000, if it is 1 you repeat 111 okay so after that like I said once you have the encoder in place, what you do after that is similar to what you did before okay.

So if you remember the bit to symbol map, the BPSK bit to symbol map that we are using has this very simple rule it takes 0 to plus 1 then 1 to minus 1 okay, so that is what happens here so this 0 becomes plus 1 here, this 0 becomes plus 1 here and this 0 becomes plus 1 there so likewise for 1 also okay, 1 goes to minus 1 okay. So this the symbol vector here s takes, it is a vector of length 3, its vector could be plus 1 plus 1 plus 1 or minus 1 minus 1 minus 1.

And then of course you have noise okay so like I said it is normal with mean 0 and variance sigma square so it is a Gaussian distributor random variable, mean is 0 and variance is sigma square and you have your received vector as s plus n okay so this is the received vector and that is what you get, okay so there will be three received values here, r_1 , r_2 , r_3 okay so because of the noise, this r_1 , r_2 , r_3 could be any value okay it could be 1.1, minus 0.8, minus 1.1 like we saw before.

So you will have all sorts of values here, now the decoder's task is to look at r_1 , r_2 , r_3 and come up with an estimated codeword, estimated or decoded version of the codeword, I am going to put codeword in codes, the reason is some decoders do not necessarily put out a codeword okay so it is just an estimate, we do hope it is a code word but sometimes it might be slightly better to put out something which is not even a code word, sometimes that happens.

So once you have a code word or a close enough estimate of it you can go back to the message, I will comment on that little bit later but this is how the overall operation works okay, a few comments on the parameters that we had okay so first is E_b over N not okay, so if you remember E_b over N not is $1/2 R \sigma^2$ okay and R is $1/3$ okay so if you put R equals to 3 you will get $3/2 \sigma^2$ okay so this is E_b over N not okay.

Remember this is for the rate 1 by 3 repetition code right, N equals 3 repetition code, what happens if you have the uncoded situation okay so if you look at E_b over N not uncoded the rate is actually 1 so you have just $1/2 \sigma^2$ okay so notice what happens here, for the same value of E_b over N not in the uncoded case you will get $1 \sigma^2$ okay for the same value of E_b over N not in the coded case you will get another value of sigma okay so this is

something important to note okay when you do coding for the same E_b over N not the value of σ changes.


And if you look over carefully this is 3 by 2 σ^2 so E_b over N not is the same in these two, the σ here has to be larger okay so in the σ , in the coded case so I will call this σ coded and this I will call σ uncoded okay so for fixed, for the same E_b over N not okay σ coded will be greater than σ uncoded, okay so this is something that one needs to remember when you do simulations for error control code.

So you will fix your E_b over N not and you will find the σ corresponding to that okay now for the same E_b over N not in the coded case you will get a worst noise as in σ will be larger, remember σ is the standard deviation of the noise if it goes higher it means you are facing a higher level of noise, the level of noise will be higher okay so now since we are measuring the performance, bit error rate versus E_b over N not, your code has two different things to do here.

First it has to anyway overcome the noise and not only that that noise becomes higher in value or higher in standard deviation because of the loss in rate okay so this is something important and you can design codes which overcome that all okay something to remember. Let us come back to the repetition code hopefully this is clear to you how this is working so the encoder is presumably easy to implement we will maybe see some implementations later on and the channel is easy to simulate now what do you do at the decode okay.

So you have three values r_1 , r_2 , r_3 what do you do at the decoder, how do you come with the code word okay, so there are various strategies and this code is simple enough and I can describe from a high level one such strategy okay.

(Refer Slide Time: 07:41)

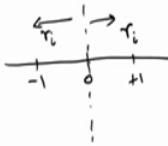


Decoder 1 for Repetition Code

$r = [r_1, r_2, r_3]$
received values

Hard Decision
Threshold at 0

Hard Decision:
If $r_i > 0$, $b_i = 0$.
If $r_i < 0$, $b_i = 1$.



PROF. ANDREW THANGARAJ
© 2016 NPTEL

$b = [b_1, b_2, b_3]$
received hard-decision values

Hard-decision Decoder

\hat{c}

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

n = 3 Repetition Code

So the first decoder that is considered is what is called a hard decision decoder, it is written here, hard decision decoder okay so we have real values here okay, what we do in a hard decision decoder is we directly do not use the real values, we first quantize them or make a hard decision to decide the bit corresponding to each received value, now this is not very optimal you can think about why it is not optimal because you are giving up a lot of information from the received value and only taking one bit of information.

But nevertheless this is one easy way to think of how the decoder works and it is quite intuitive so one can use a hard decision decoder, so in many coding situations people will want to use a hard decision decoder maybe the real values are coming too fast and maybe so many bits of information you cannot handle in your decoder, you don't want to for that reason you can make a hard decision choice also, but it is definitely sub-optimal to a significant level okay.

So how do we do hard decisions, remember our symbol vector is plus 1 and minus 1 and we have this thresholding at 0 as a sort of a remainder. So if your r_i is positive it lies on the side you want to say the bit is 0 okay and if your r_i is negative you want to say b_i is 1 okay so this is my, so received hard decision values okay. So now you do a hard decision decoder here okay so I wrote decoder here so maybe it is not a good idea to call this a decoder you make a hard decision here so this is threshold at 0 okay and then you run a hard decision decoder.

So now this because this b is just bits b_1, b_2, b_3 it is easy to list out what these bits should be okay remember even though at the transmitter end the code word was only 0 0 0 because of the noise and because errors can happen this bit vector b can be any one of the 8 possibilities

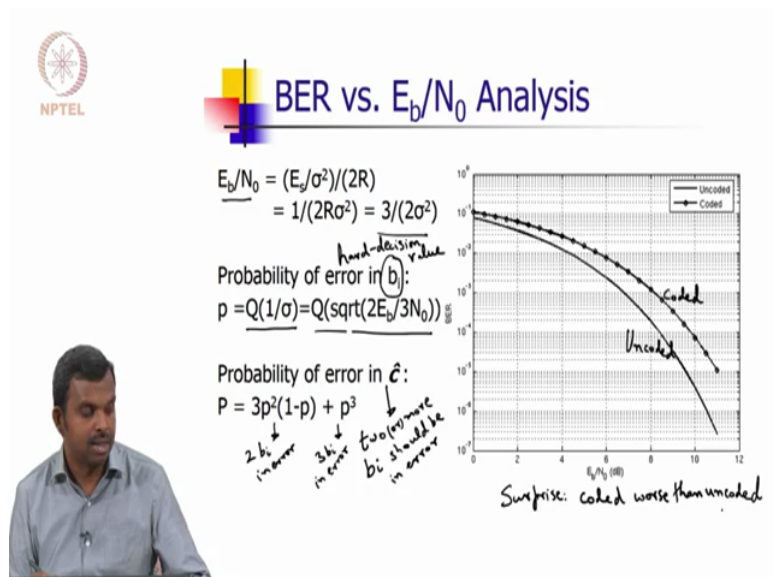
okay, what are the 8 possibilities, 000, 001 so on till 111 so you have 8 different possibilities and I have arranged them in a nice way here and if you look at the first 4 possibilities okay these are closer to 000 than 111 okay.

So if you look at 000, if 000 was transmitted this could be received without any flip on the channel but you know if even if 111 was transmitted you can actually receive 000 but then what should happen the noise should be so high 3 consecutive times and you should be able to flip every 0 to 1 okay so that may not happen with very high probability so you can say this is closer to 000 so you decide 000 okay so you see all these guys are mapped onto 000.

On the other hand here this is sort of closer to 111 okay so that is this notion of closeness is in a certain distance we will look at that a little but later but for now you can see intuitively how you do this okay so this is version of the hard decision decoder which is actually sometimes used in practice in many channels where you cannot use real values at the decoder but it is definitely sub optimal you are giving lot of information okay.

So hopefully this decoder was clear to you it is very simple you look at the received hard decision value and if it is closer to 000 as in if it is 000, 001 010 or 100 you decide 000 otherwise you decide 111 okay so it is one can implement this if needed in MATLAB okay.

(Refer Slide Time: 11:55)



So one can do an analysis of BER versus E_b over N not okay so I have shown that here you can see E_b over N not for the coded case is 3 by 2 sigma square and probability of error in b_i okay so b_i is the hard decision value okay we saw before is Q of 1 by sigma okay so now 1 by sigma you can rewrite in terms of E_b over N not and you get this formula right so you put

$1/\sigma$ equals square root of $2 E_b$ over N not by 3 which is what I have done here okay and you get Q of this.

So this is the formula. This is the probability of error in B_i okay now what is the probability of error in c cap, this is 2 or more b_i should be in error okay so if you put that in you will get this calculation okay so this is the case where 2 b_i in error this is the case where 3 b_i in error okay p power 3, they are all independent so it works out like this so this is the probability of error okay.


So you can express the probability of error in terms of E_b over N not okay so you evaluate this expression and here is the surprising and interesting case, interesting situation here. So this is the uncoded plot this is the coded plot okay so surprise, what is the surprise? Surprise is coded seems to be worse than uncoded okay.

So this is the warning one needs to be careful when you do decoding you cannot do very heavily sub optimal decoding because you are penalizing on the σ right so E_b over N not is normalized by rate so when you reduce rate, your σ goes worse and if you don't do good decoding you can be even worse okay so you spent so much energy to do coding and your worse often E_b over N not okay.

Now I would like to point out one more thing if you plot this BER versus SNR instead of E_b over N not, remember what is SNR, it is just $1/\sigma^2$, the R does not play a role okay if you do just SNR you will see a gain okay so that is why SNR is sometimes misleading, it is better to look at E_b over N not, to be very fair okay.

So this is the sort of negative thing, it was promising huge shift to the left, I am shifting to the right, well turns out repetition code is not that code for coding gain and in fact one can improve upon this and the next decoder that we see will improve upon this and at least recover as good a performance in the uncoded case so we will see that next okay.

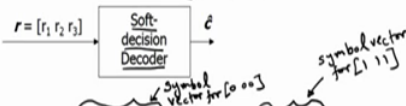
(Refer Slide Time: 14:53)



Decoder 2 for Repetition Code

$$r_1 + r_2 + r_3 > 0 \quad \hat{c} = 000$$

$$< 0 \quad \hat{c} = 111$$



If $|r - [+1 +1 +1]|^2 < |r - [-1 -1 -1]|^2$
 or $(r_1 - 1)^2 + (r_2 - 1)^2 + (r_3 - 1)^2 < (r_1 + 1)^2 + (r_2 + 1)^2 + (r_3 + 1)^2$
 $r_1 + r_2 + r_3 > -r_1 - r_2 - r_3$
 $r \cdot [+1 +1 +1] > r \cdot [-1 -1 -1]$,
 or $r_1 + r_2 + r_3 > 0$,
 then $\hat{c} = 000$
 else $\hat{c} = 111$

-Optimal decoder



So this is decoder 2 okay, so this a soft decision decoder okay so we would not give up the real values okay and notice this is how the decoder works okay so you look at the distance between the received vector and plus 1 plus 1 plus 1, remember plus 1 plus 1 plus 1 is the symbol vector for the code word 000, so likewise this minus 1 minus 1 minus 1 is the symbol vector for the code word 111 okay.

So you look at the received value and you find its distance from the two symbol vectors, what is the distance from 000, what is the distance from 111 okay but not in terms of bits but in terms of real values itself, in terms of the symbol vector plus 1 plus 1 plus 1 minus 1 minus 1 minus 1 okay whichever is closer, remember if it is closer to 111 or the code word 000 okay then you declare \hat{c} is 000.

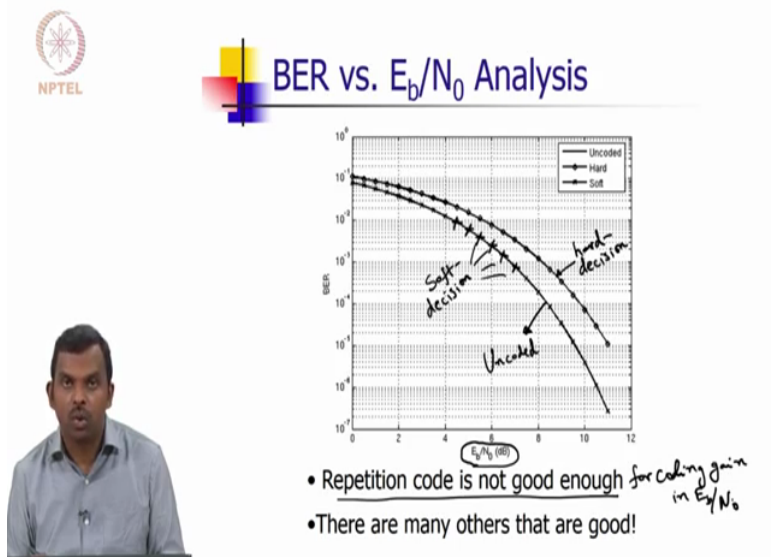
On the other hand if it is closer to minus 1 minus 1 minus 1 in this distance you declare \hat{c} is 111 okay, now this condition involves in finding the length and squaring and all that but given the structure of this you can simplify, so remember what is this on the left hand side, this is $r_1 - 1$ whole square plus $r_2 - 1$ whole square plus $r_3 - 1$ whole square it is less than $r_1 + 1$ whole square plus $r_2 + 1$ whole square plus $r_3 + 1$ whole square.

Okay now you can, let me write this a little bit more cleanly, okay so now you can expand this out it will be lot of cancellation and this will simply reduce to $r_1 + r_2 + r_3$ greater than minus $r_1 - r_2 - r_3$ okay and that is the same as this dot product you take a correlation with 111 or minus 1 minus 1 minus 1 whichever is greater now is the closer code word and inside that also simplifies to $r_1 + r_2 + r_3$ greater than 0 okay.

So the ((17:13)) decoder for the repetition code is quite easy to implement, you have r_1 , r_2 , r_3 you simply check for r_1 plus r_2 plus r_3 okay and if it is greater than 0 you let c cap to be 000, if it is less than 0 you let c cap be 111 so it is relatively easy decoder to implement okay now this turns out is an optimal decoder okay you are not giving up anything in this decoding okay it is an optimal decoder which is called by various names maximum likelihood decoding etc. so this is optimal for probability of block error okay as you may uniformly distributed code words okay.

So this is the optimal decoder so you cannot do better than this in decoding okay so remember the hard decision decoder I mentioned it suboptimal you are giving up a lot of information this one does not give up any information, gives us all the information in the best possible way okay so you cannot do better than this decoder okay.

(Refer Slide Time: 18:11)



So turns out one can analyze this decoder also, I am going to skip the analysis maybe I will talk about it in another lecture just to give you a brief idea of how the analysis works but if you do that this is the picture okay so remember this is uncoded, this curve is uncoded, this is hard decision decoder and it was worse and in fact the soft decision decoder lies on top of the uncoded curve okay so these are all soft decision decoder, the ML decoder maximum likelihood of the optimal decoder and that ends up lying on top of the uncoded curve okay.

Remember this plot is with E_b over N not in dB and that is very important so the rate normalization that you did the soft decision decoder simply just about recovers whatever loss you had in this okay so one can easily conclude repetition code is not good enough for coding

gain so you would not get coding gain in E_b over N not with repetition code okay but there are other codes which are good I will point out one more code immediately and so this also brings out a few important points so I will summarize a few things before I go on to the other code.

First thing is code tells you how to map messages to code words you should implement their encoder it is easy enough to do but once you do that you map the bits one by one into symbols and transmit, you have a received vector that will be a real valued vector and that you have to decode to get your estimated code word okay.

Now you can do the decoding in multiple ways if you do sub optimal things there is danger of doing even worse than the uncoded case that is used in simple situations that happens but if you do the optimal decoder things works okay. I have not mentioned too much about complexity, we will see soon enough how these decoders are going to be in practice if you want to implement them but one important point is you have to design your code well okay if you want coding gain you have to design your code well and you have to have your good decoder also okay, good decoders are important, codes are also important.

So that is sort of couple of simple takeaways from this whole picture and what I am going to do next is show you a small alteration of the script that we wrote before to implement repetition code, how to go about implementing it and then maybe we will try and recreate some of the points in this curve and see if they are getting it or not okay so we will do that next.