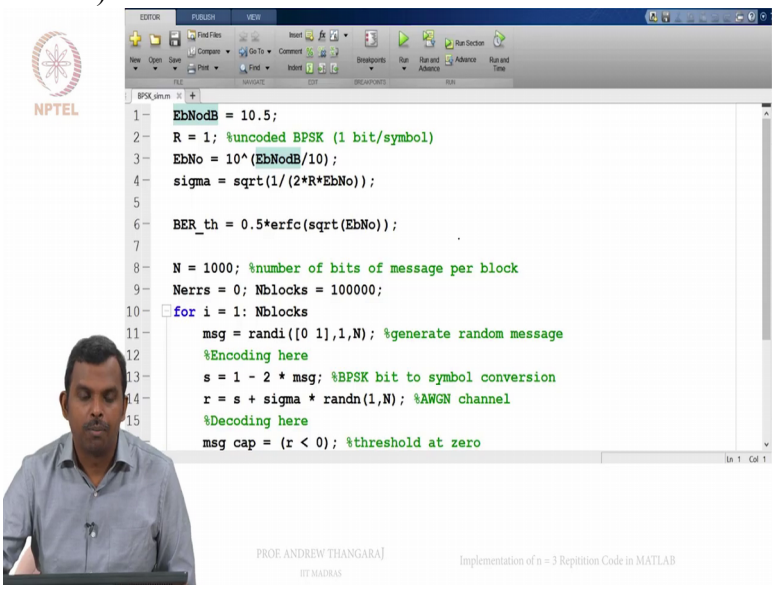
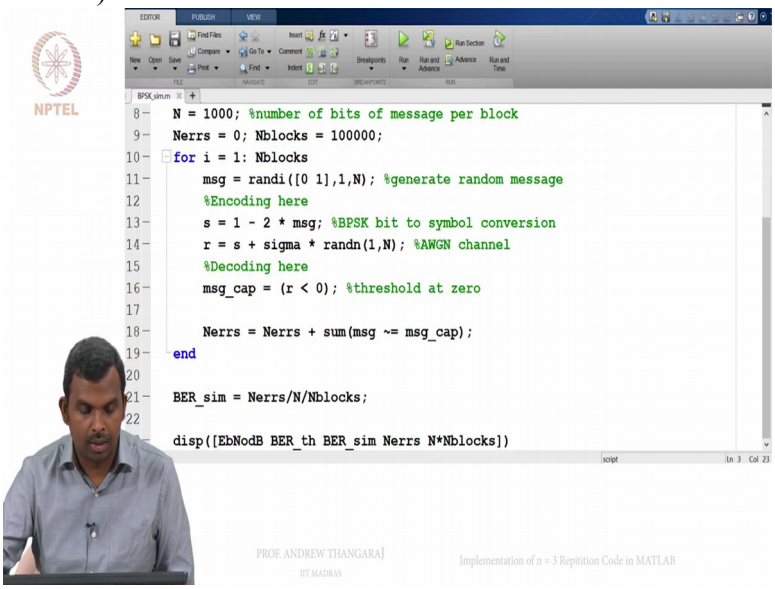
LDPC and Polar Codes in 5G Standard
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology Madras
Implementation of n = 3 Repetition Code in MATLAB

(Refer Slide Time 00:16)



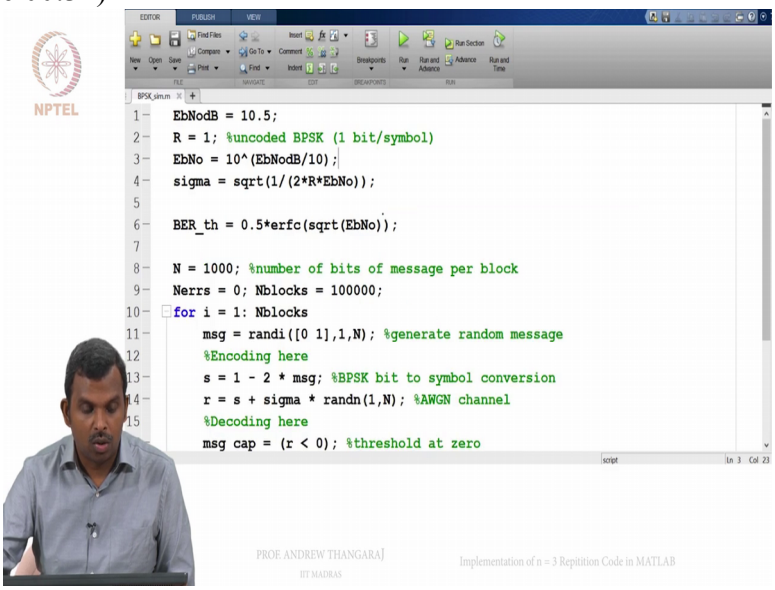
Ok. So this is the B P S K Simulation script that we had before. Hopefully you remember this how we were doing,

(Refer Slide Time 00:25)



running the, the uncoded thing, how we were using this randi command to generate messages etc etc, Ok.

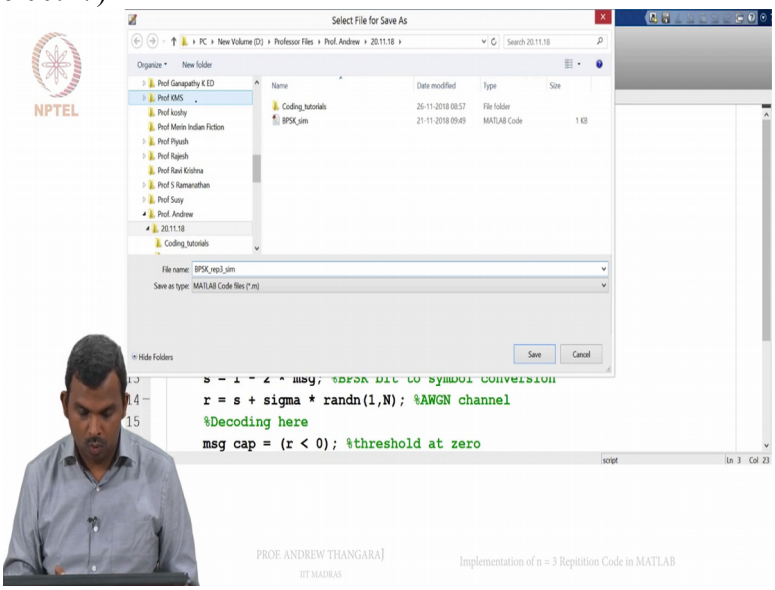
(Refer Slide Time 00:34)



So now let us see how to modify this for getting the coding implemented.

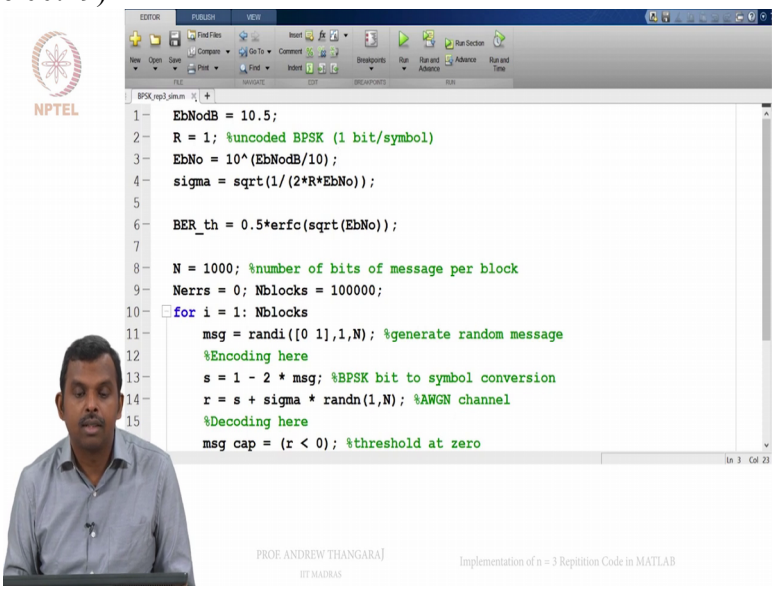
So I am going to save this as something else. So I will save B P S K rep 3 sim

(Refer Slide Time 00:47)



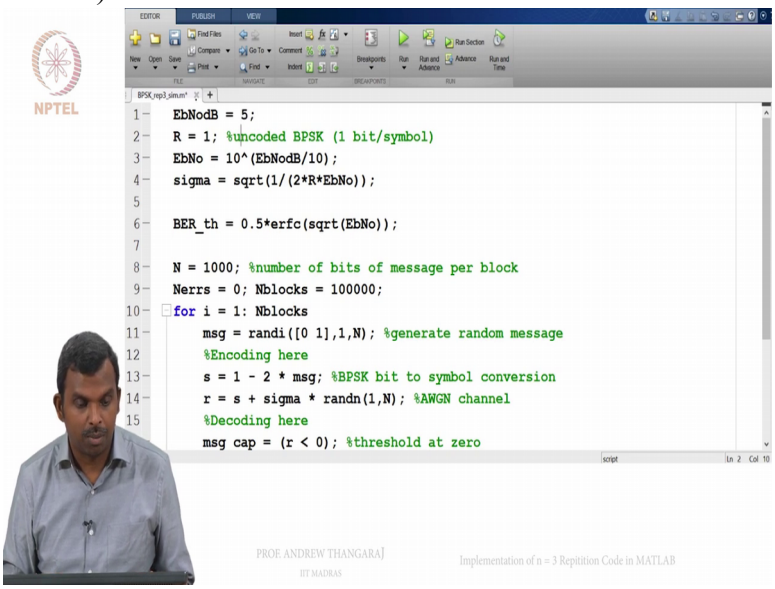
Ok. So this is the

(Refer Slide Time 00:49)



coded case with the repetition code. So the first thing is you can keep, may be E b over N naught at some reasonable value 5 or something

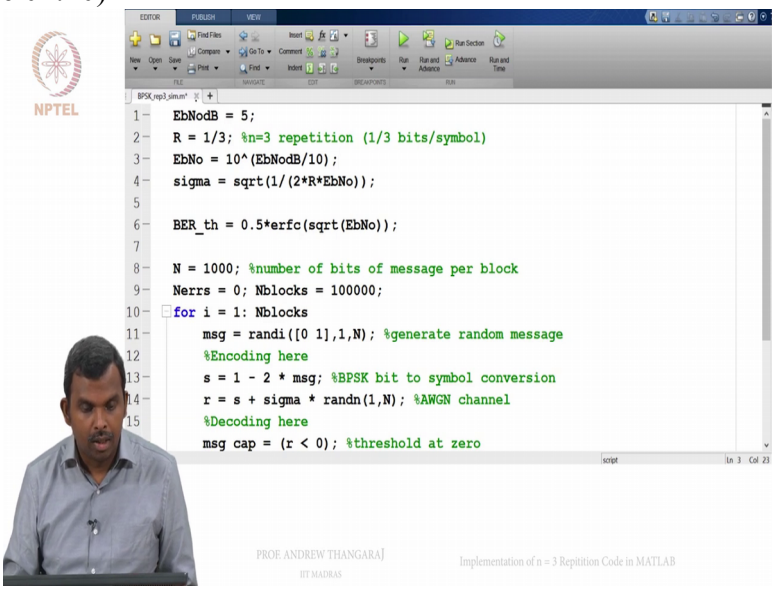
(Refer Slide Time 00:57)



that does not matter. First thing is the rate. Rate is going to 1 by 3.

So like I said n equals 3 repetition code. So this is 3 symbols for 1 bit, Ok so 1 by 3 bits per symbol,

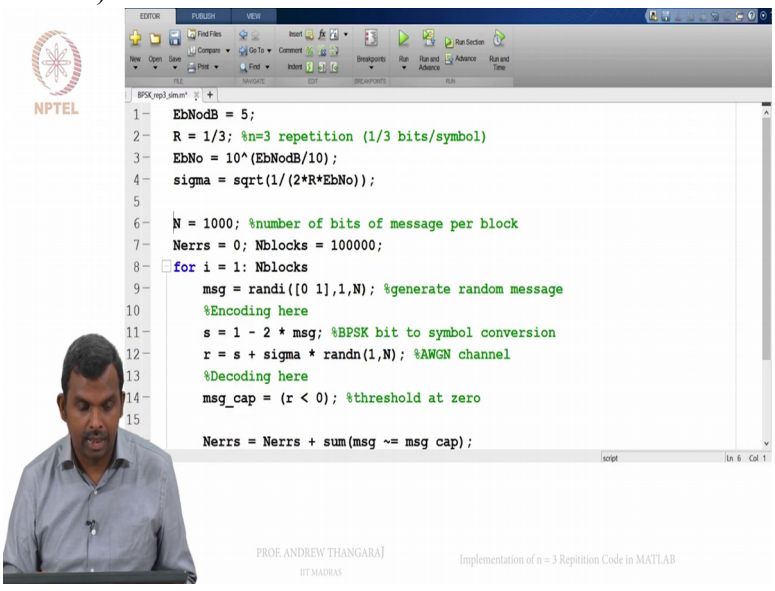
(Refer Slide Time 01:20)



Ok. So after you fix your rate, the rest of the calculation is not too bad. Your E b over N naught is 10 power d B to regular value conversion and then you find the sigma corresponding to E b over N naught Ok.

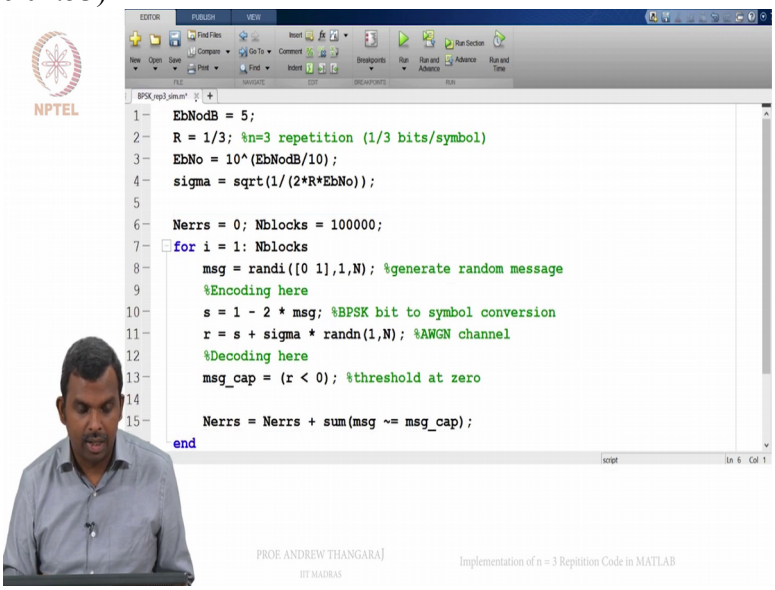
And then, so this is not quite correct so there is no need for theoretical thing. So we can

(Refer Slide Time 01:43)



delete that, Ok. So, so now here we have to change things a little bit, Ok. So you will, this n will not really enter the picture

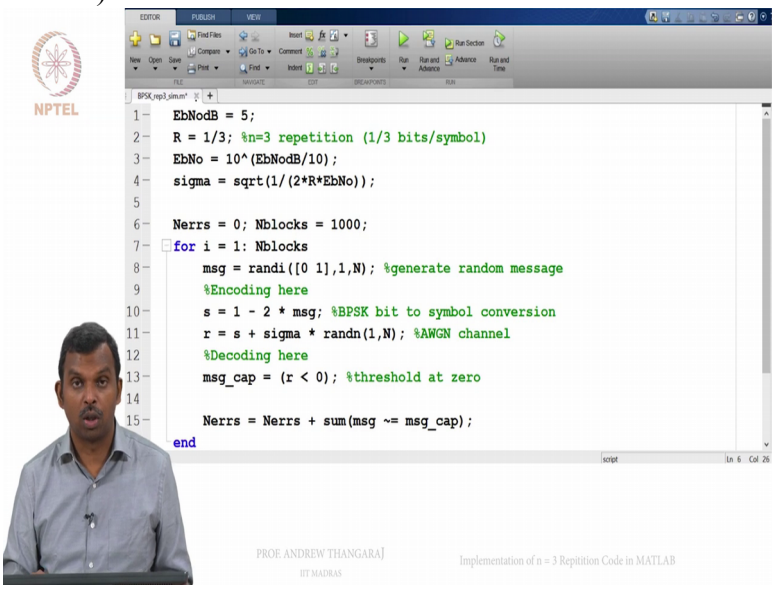
(Refer Slide Time 01:53)



so we can delete it off. So you have blocks, each block is a codeword block, Ok. Remember that, each block is a codeword block.

Let us say, we simulate 1000, 1000

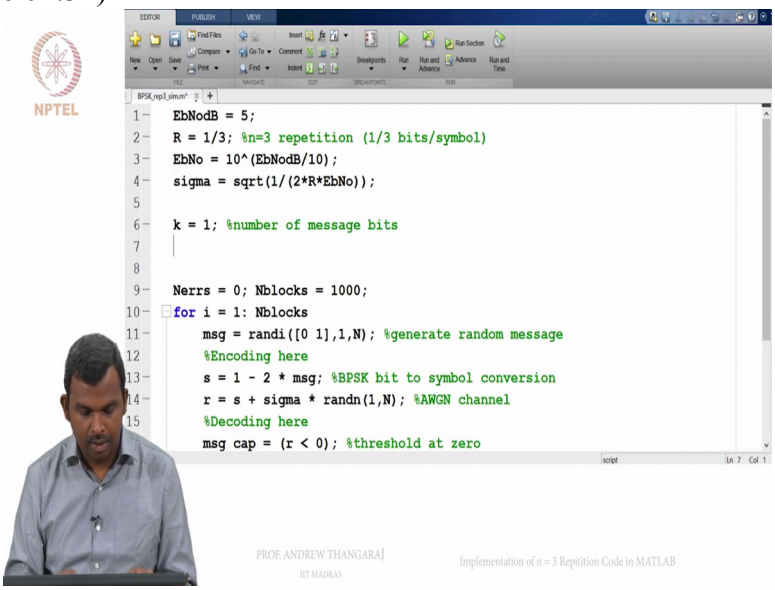
(Refer Slide Time 02:05)



codeword block. So what do we mean by codeword blocks? In one block there will be 1 message bit. It will get encoded into 3 codeword bits and that will be get transmitted, Ok. So now we will have to do block after block after block, Ok.

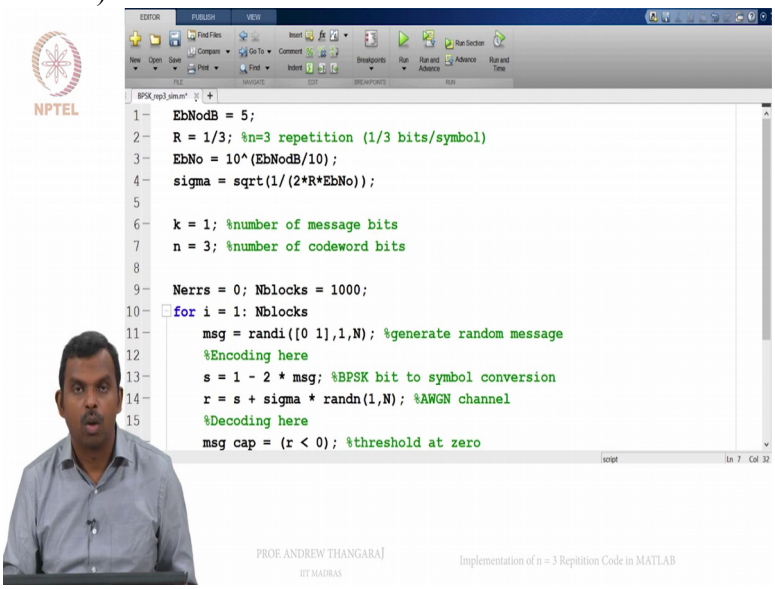
So it is useful to define a few things here. We can define k as the number of message bits,

(Refer Slide Time 02:31)



Ok and we can define small n as the number of codeword bits,

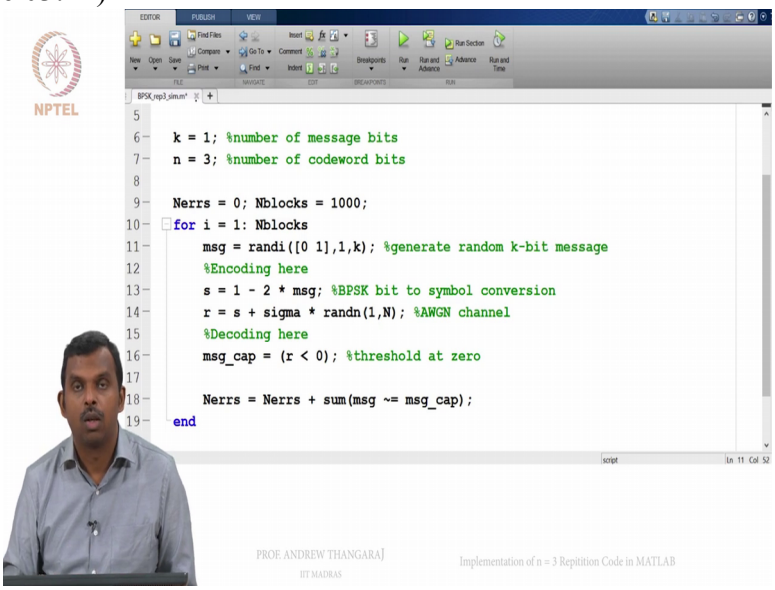
(Refer Slide Time 02:42)



Ok. So this is good to have. And once you have this, so let us go ahead here. The message itself for every block, this loop goes for every block for i equals 1 to n blocks, and then in every block I am going to generate a random bit, Ok.

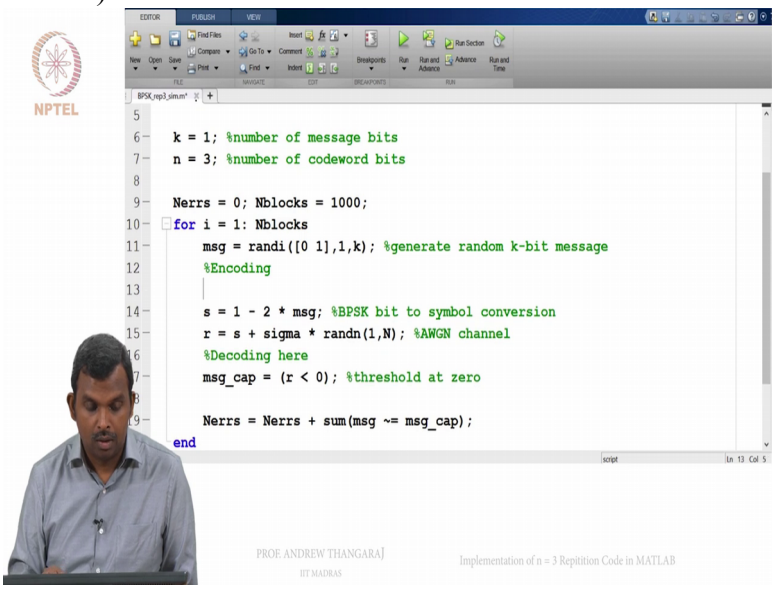
So this will just be 1. So since I need just k bits, I will put 1 comma k, Ok, generate random message, random k bit message, Ok,

(Refer Slide Time 03:12)



alright. So now I have to put in my encoding, Ok.

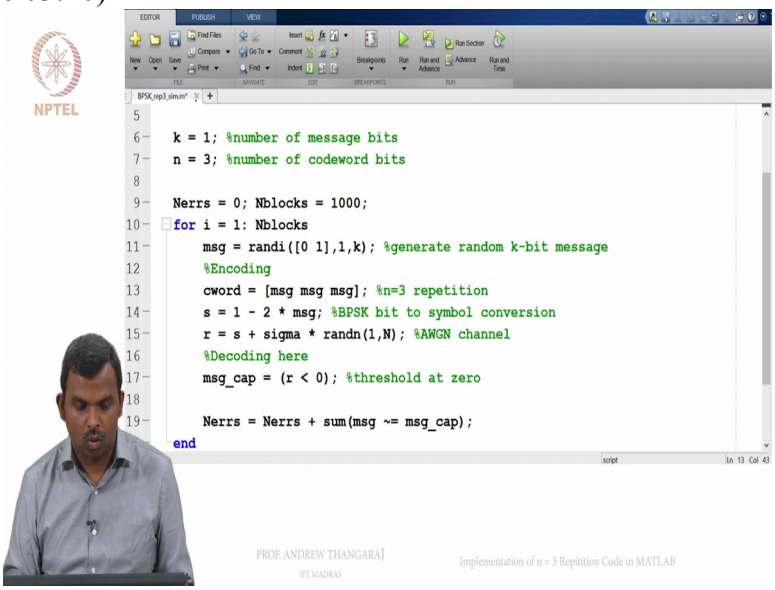
(Refer Slide Time 03:16)



So remember my encoding is just repetition code and I know my k is just 1.

So I just have 1-bit message here. One can do a lot of things here. So I will do a very, very simple encoding m s g m s g m s g, Ok. So this is n equals 3 repetition code,

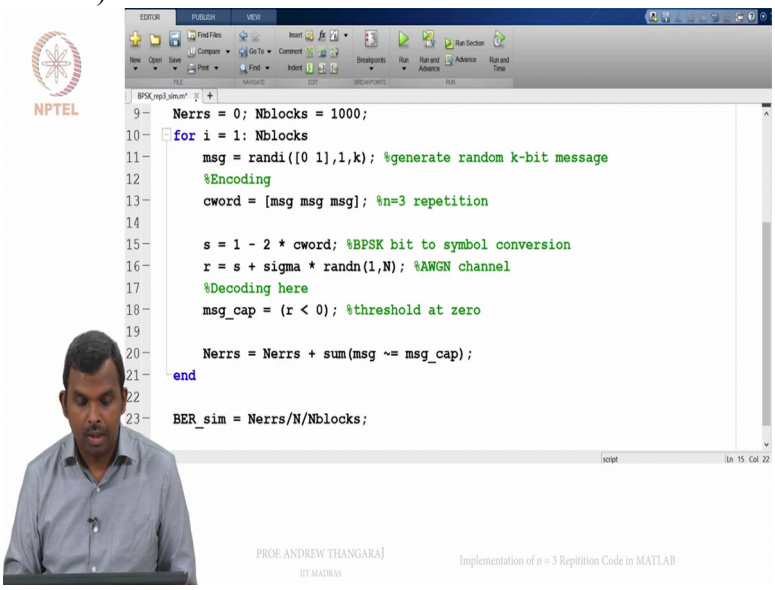
(Refer Slide Time 03:40)



Ok, alright? So now, so encoding is complete so maybe I will put down a little bit of a gap here, Ok.

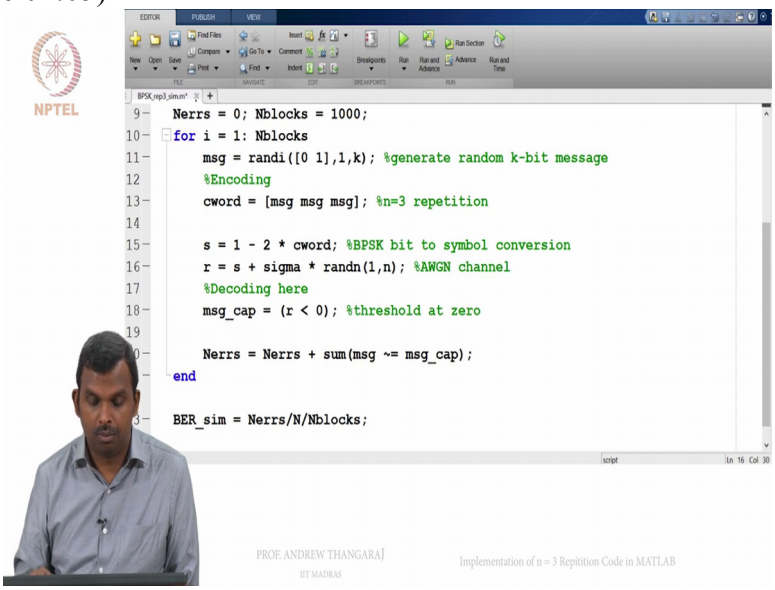
So then you have the symbol vector. It is 1 minus 2 into not message but codeword, right,

(Refer Slide Time 03:57)



B P S K bit to symbol conversion for the codeword and then after that this one come up, n,

(Refer Slide Time 04:05)



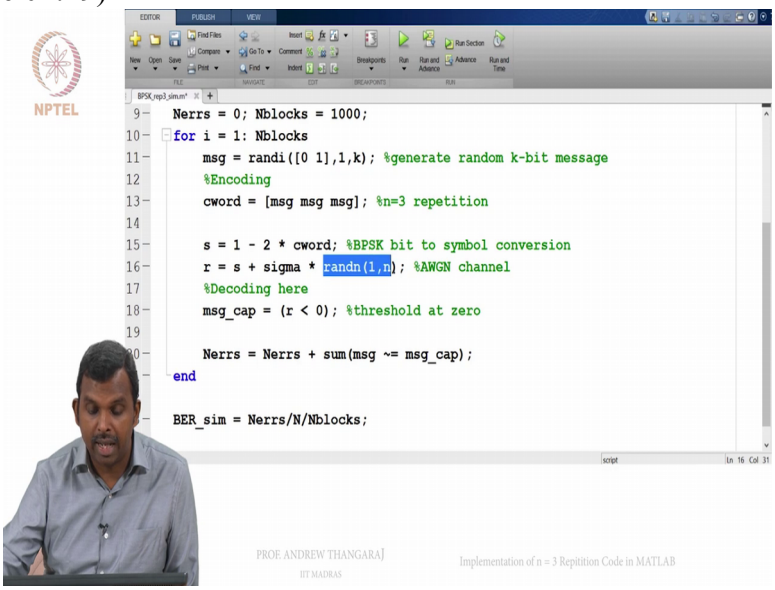
Ok. A W G N channel goes here. Is it Ok?

So hopefully you can see what I have done here. Nothing much has changed except that I have a k-bit message in every block and then I have a codeword. This encoding operation can be more complicated depending on what code you pick.

For the repetition code it is just message message message. No problem. Once you have the codeword everything is same as before, Ok. So you do 1 minus 2 into c word. You remember how I did 1 minus 2 into, if a codeword bit is zero I get plus 1, if the codeword bit is 1 I get minus 1, Ok.

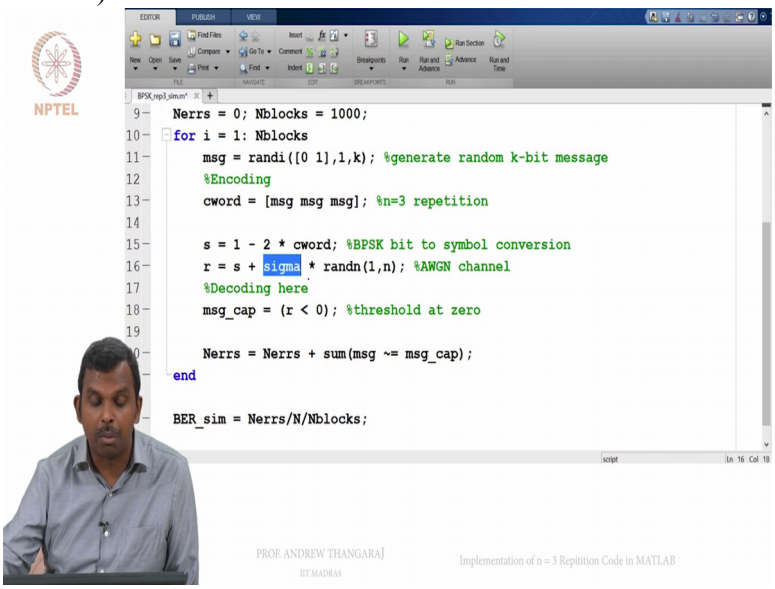
And then you have the value of sigma. You multiply this with n Gaussian random variables. Why do I do n here? Because that is my codeword right, so I have to generate n

(Refer Slide Time 04:49)



Gaussian values with the standard deviation of sigma,

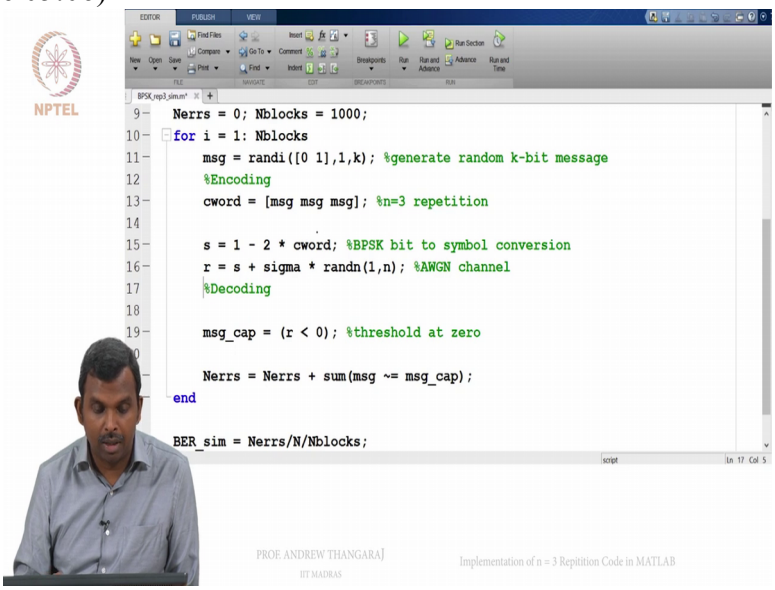
(Refer Slide Time 04:52)



Ok. So I do that, add it to the symbol vector s here.

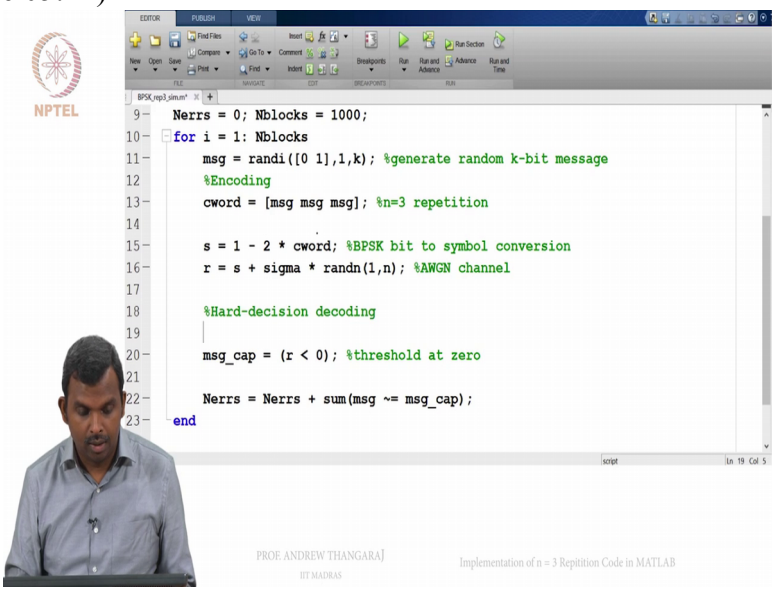
So all these things becomes vectors. Matlab is quite decent about handling these kind of things. We don't have to make much changes here, Ok. So after you get the received value you have to do decoding, Ok.

(Refer Slide Time 05:06)



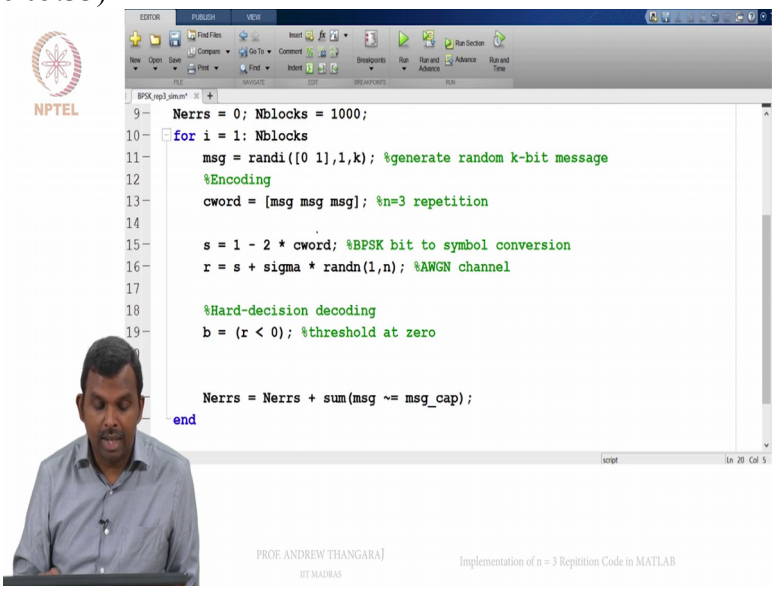
So you can do multiple types of decoding. So, so here I have done thresholding at zero, so may be one can do hard decision decoding here, Ok.

(Refer Slide Time 05:21)



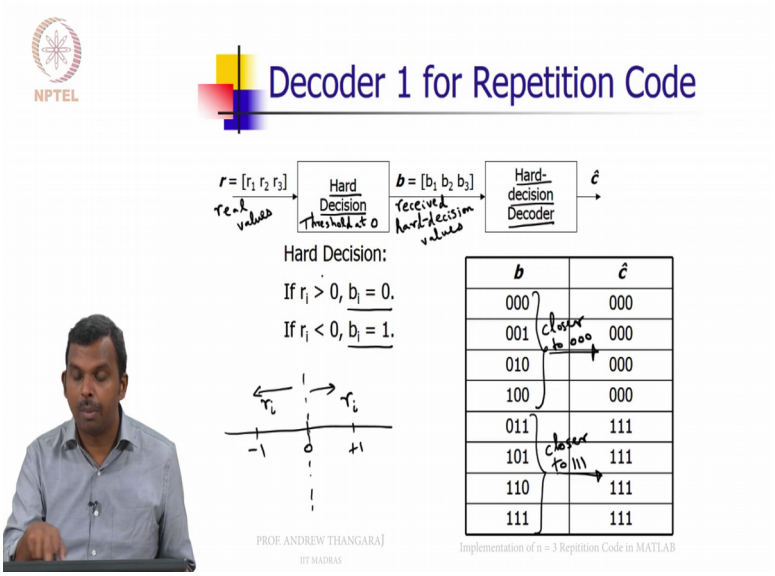
So remember hard decision decoding, you are still going to threshold at zero but we call that vector as b, right? b, b is the vector which should be (()) thresholding at zero.

(Refer Slide Time 05:33)



And then I am going to basically see if b is either 0 0 0, 0 0 1, 0 1 0 or 1 0 0 if it has a fewer number of 1s, so let me show you the entire picture. Here,

(Refer Slide Time 05:47)

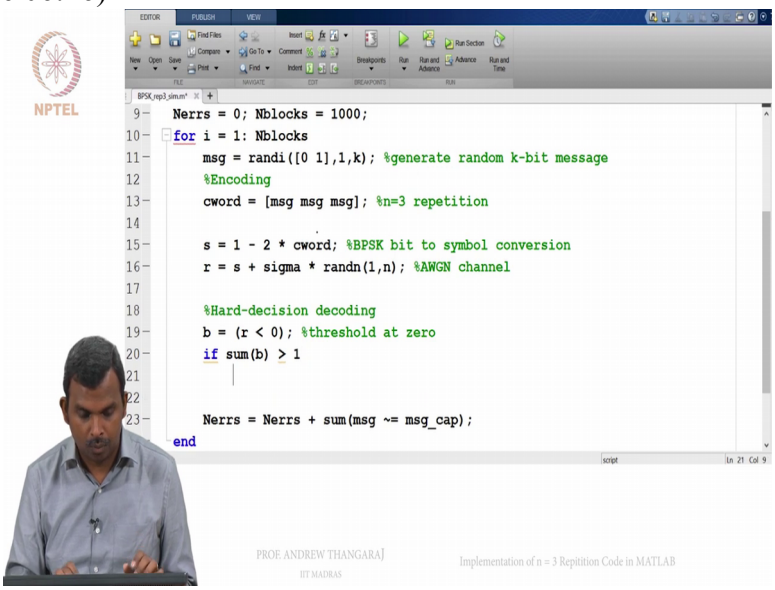


so this is the lookup table I have to implement, Ok.

So there are various ways to implement the lookup table. I am going to basically count the number of 1s in b. If it is 1 or 0, I will make the codeword as 0 0 0. If it is 2 or 3, I will make the codeword as 1 1 1. Ok, so that is my strategy, so let us do that, Ok.

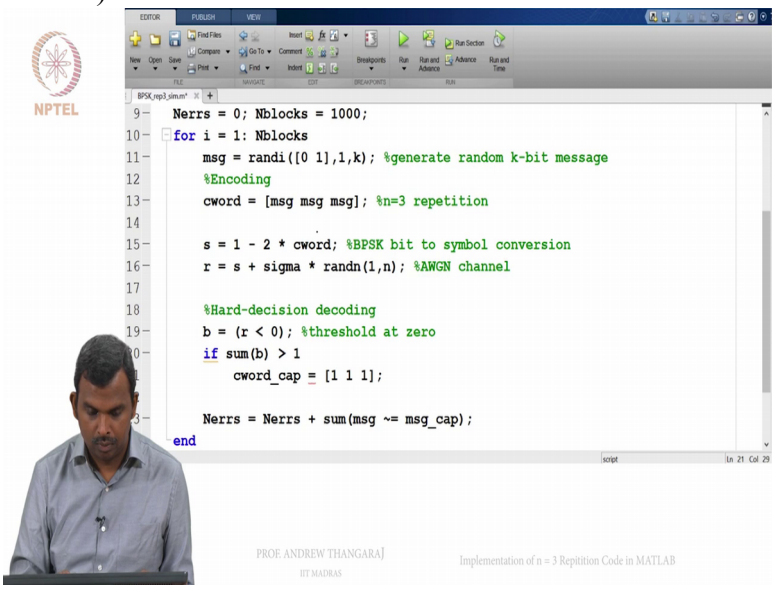
If sum of b which is going to count the number of 1s is greater than 1

(Refer Slide Time 06:16)



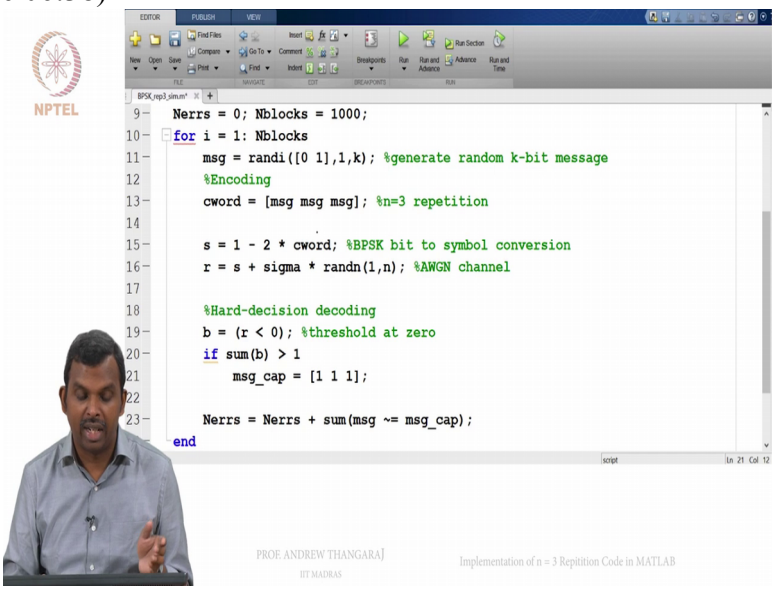
Ok, then c word cap equals 1 1 1,

(Refer Slide Time 06:31)



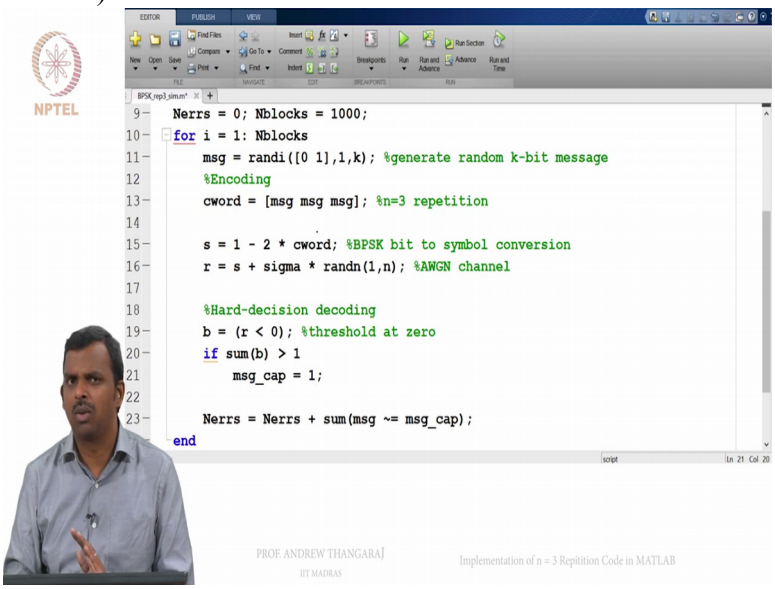
Ok. So actually I mean if you think about it you won't need the message, the codeword estimate,

(Refer Slide Time 06:38)



we can directly do the message estimate. We can simply say the message estimate is 1,

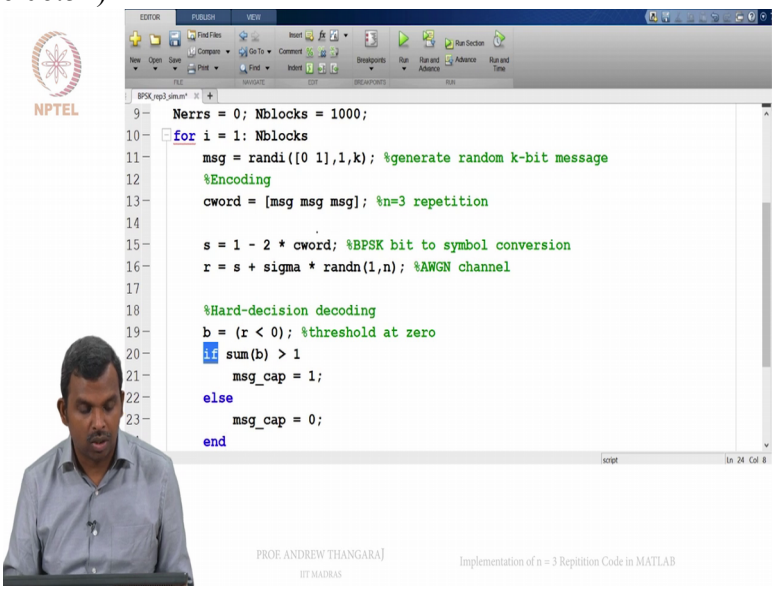
(Refer Slide Time 06:42)



Ok.

So this is better. You can try to find the codeword but really you are interested in the message, right? So that is, so you do that, else we say message cap equals zero. And then we can

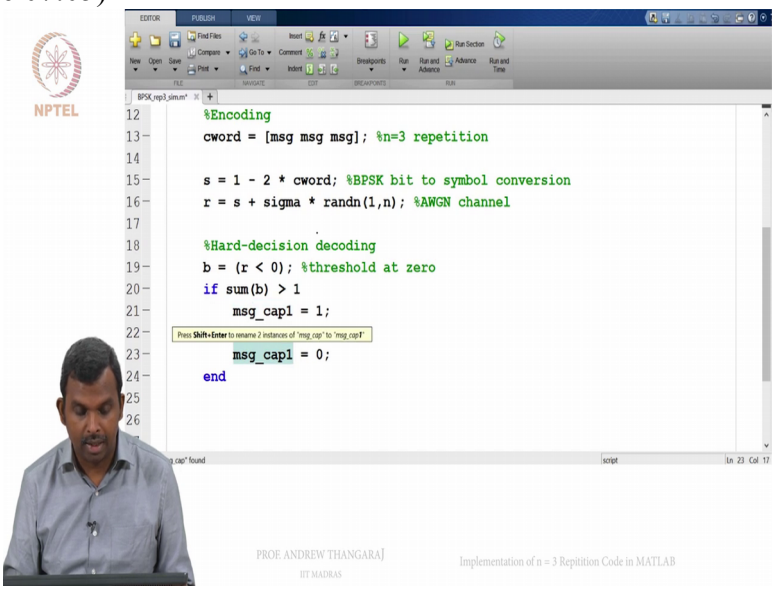
(Refer Slide Time 06:54)



end it. So that is it. Ok. So that is the simple hard decision decoder.

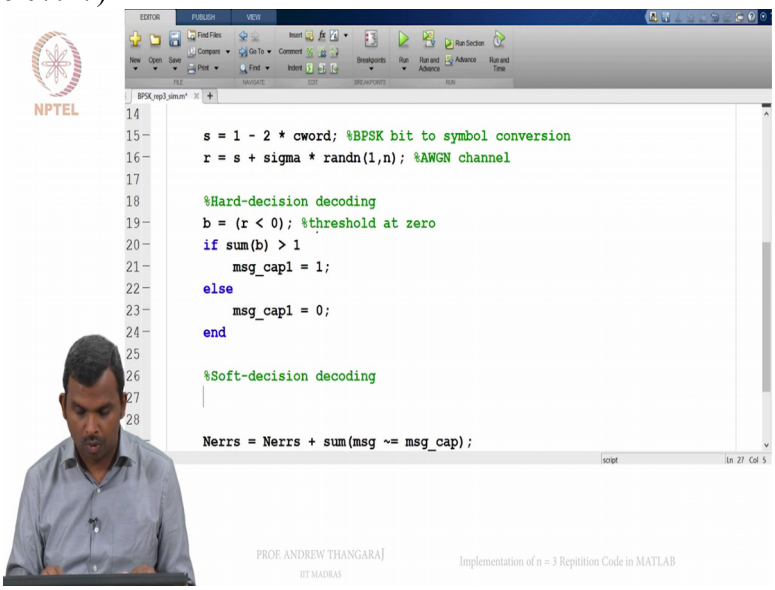
So how do you implement soft decision decoding? So maybe I will call this as message cap 1

(Refer Slide Time 07:05)



to indicate that this is hard decision decoding, Ok. The next one is soft decision decoding,

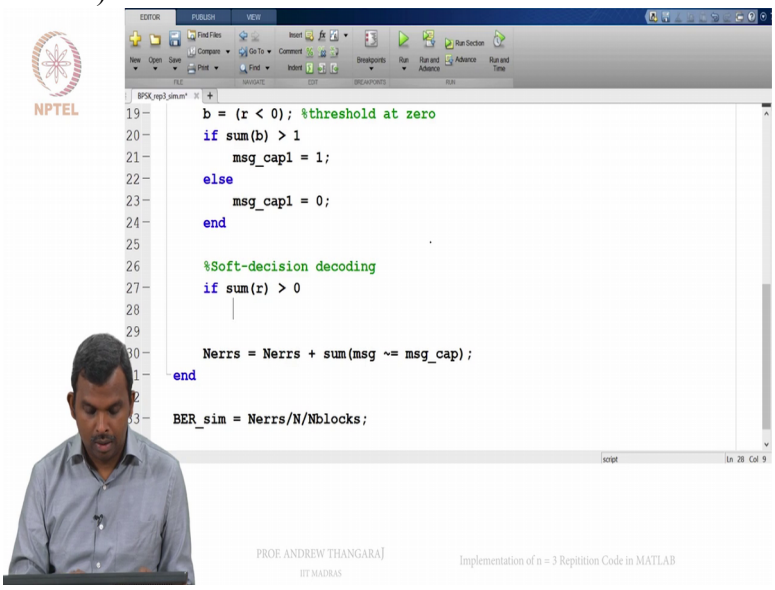
(Refer Slide Time 07:17)



Ok.

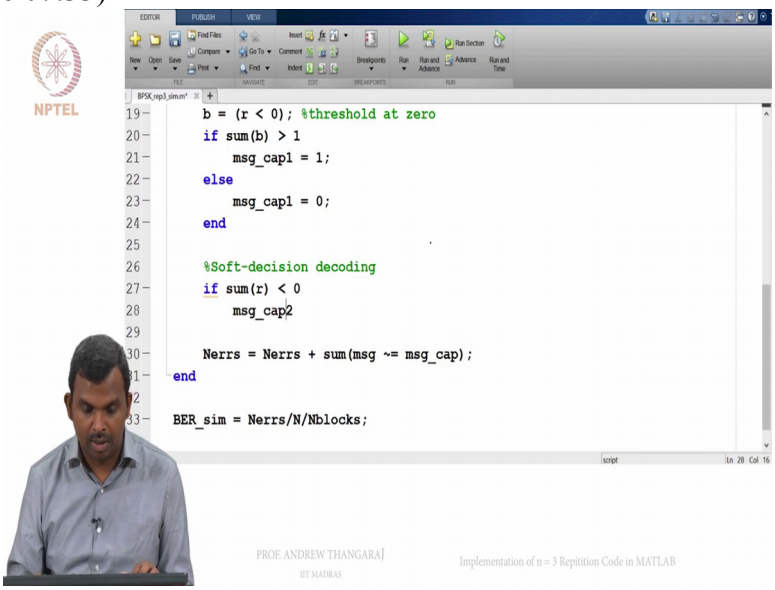
So in this case if you remember, I have to do r 1 plus r 2 plus r 3 and see if it is greater than 0. So it is simply sum of r and if it is greater than 0

(Refer Slide Time 07:27)



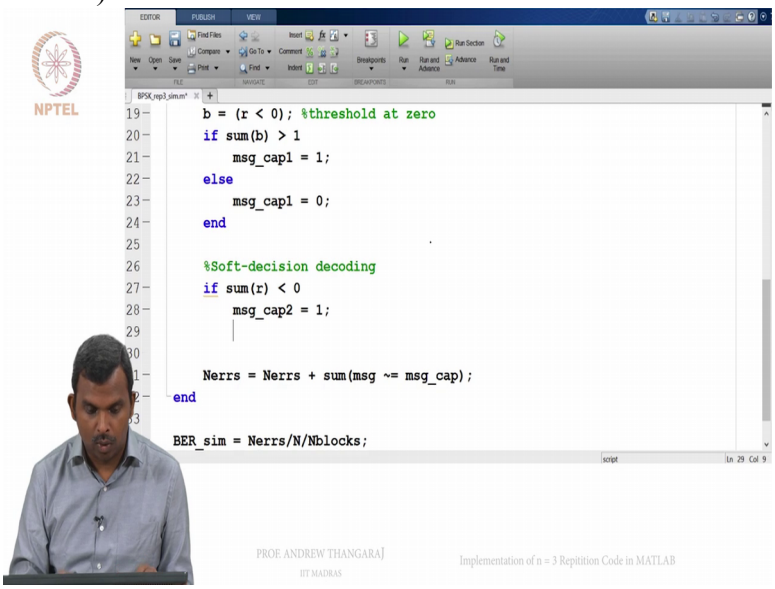
I will set message cap to or maybe I will put that as less than 0,

(Refer Slide Time 07:33)



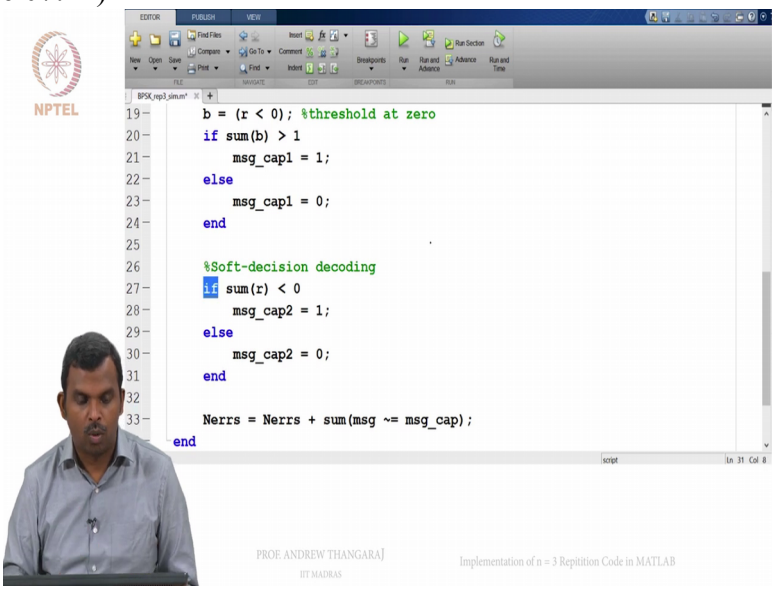
so have it similar to before equals 1

(Refer Slide Time 07:35)



else message cap equal to 0, that is it,

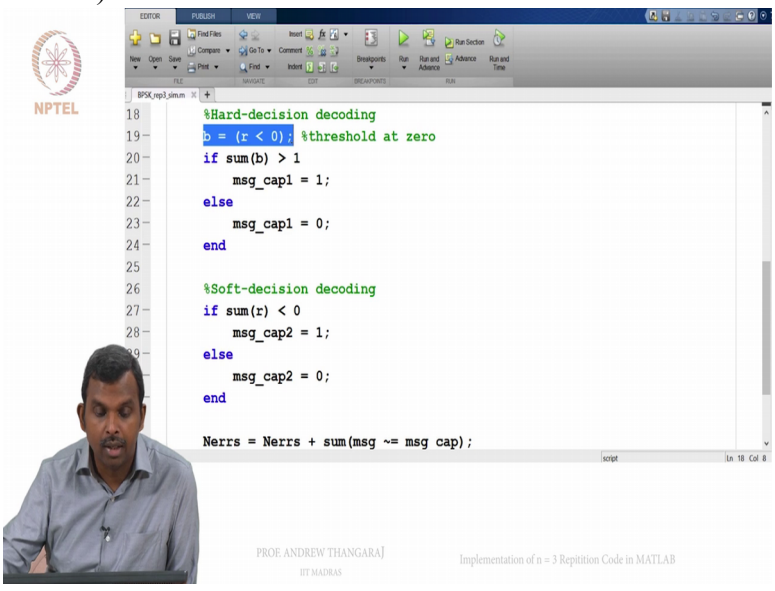
(Refer Slide Time 07:41)



Ok.

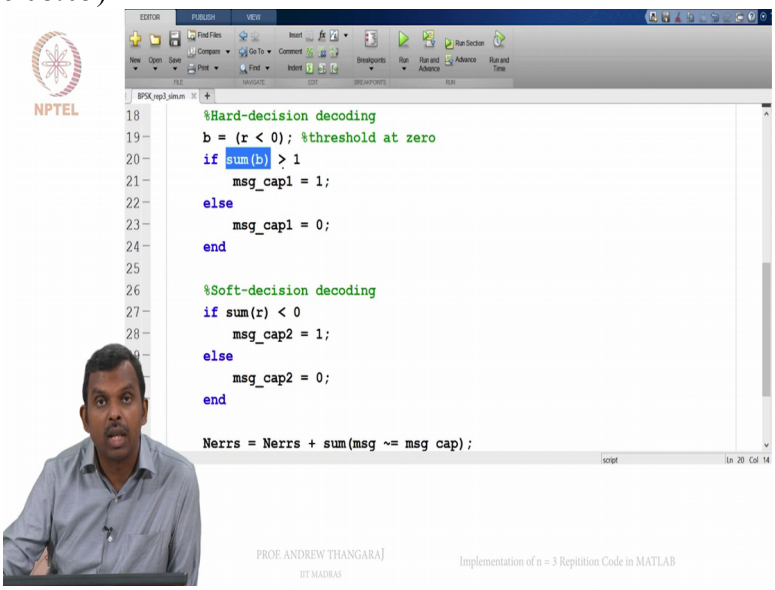
So these kind of, these kind of decoders for the repetition code are very, very easy to implement. Hopefully you agree with me that it is the same, so I have done the hard thresholding here. This is the threshold at 0 we spoke about how to do thresholding.

(Refer Slide Time 07:56)



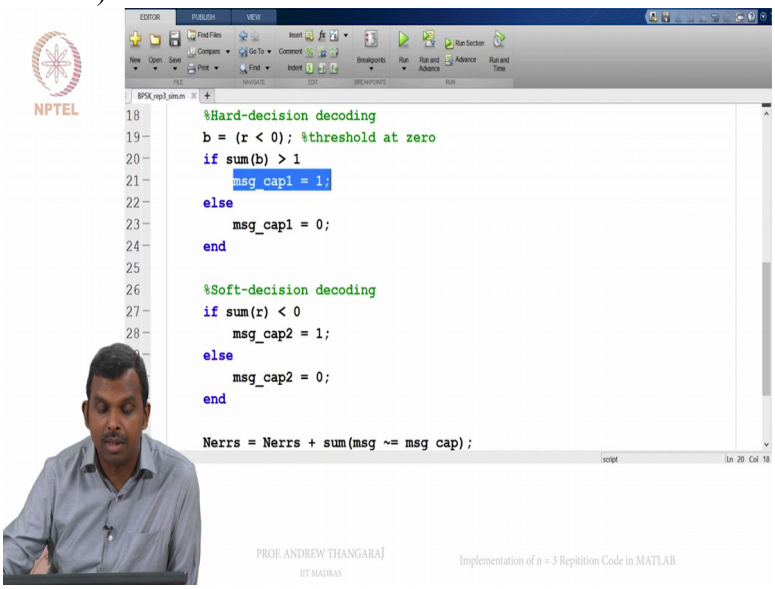
We can simply set r less than 0. If it is negative it will become 1, if it is positive it will be 0, right. So that is how it is. And I am counting the number of 1s

(Refer Slide Time 08:05)



here, Ok. If the number of 1s here is greater than 1, Ok then I set my message cap

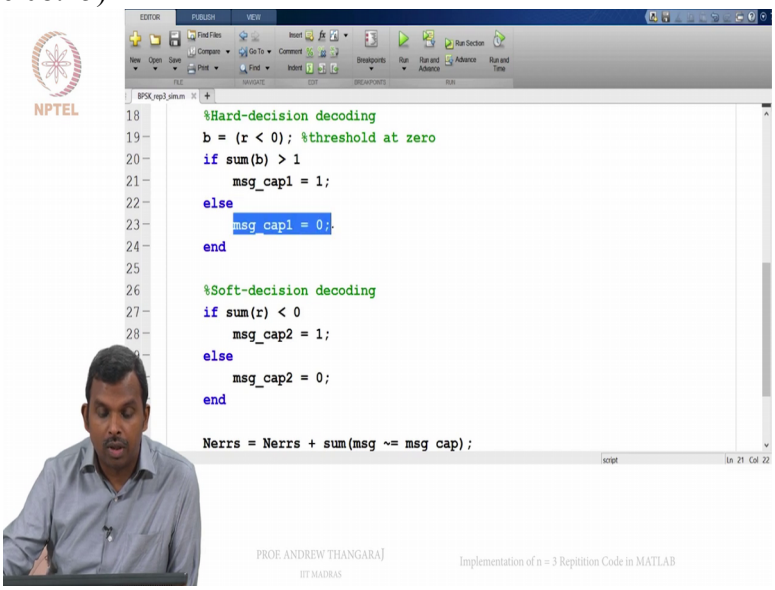
(Refer Slide Time 08:10)



as 1, 2 or 3 that is the case.

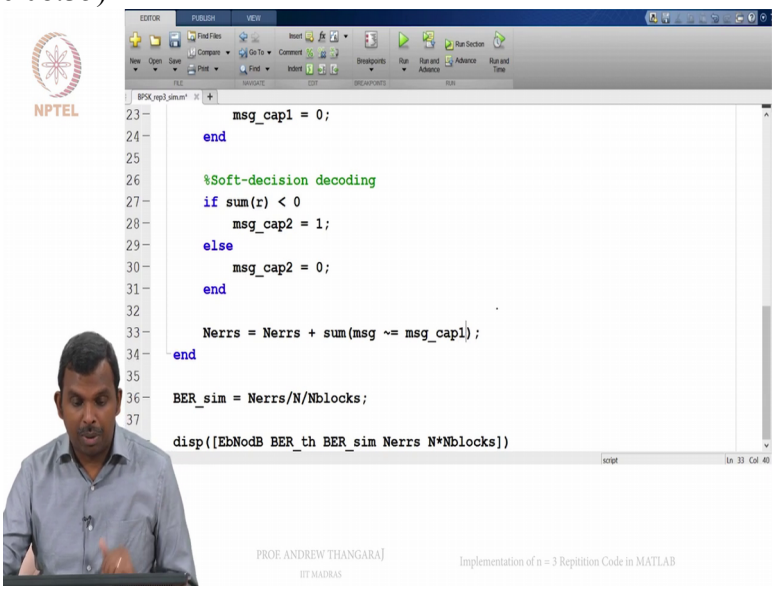
And here

(Refer Slide Time 08:15)



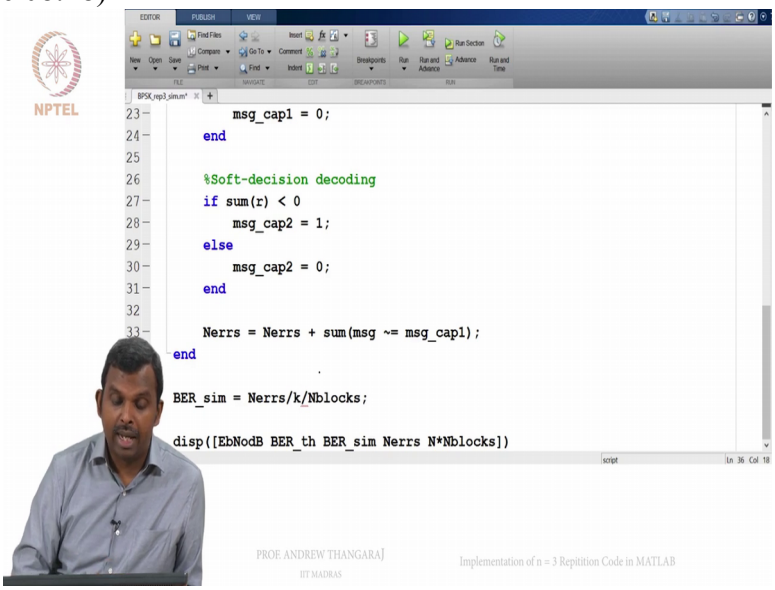
if, on the other hand if the number of 1s in b is 1 or 0, my message cap 1 is going to 0, Ok. Now you have soft decision decoding which is message cap 2 here, 1 or 0, is that Ok? So here, a number of errors you have to either pick 1 or 2, so

(Refer Slide Time 08:35)



maybe I will pick 1 here just to get the hard decision decoder working and after that I have to make a bit of an adjustment here. The B E R simulation will actually be N errors divided by k

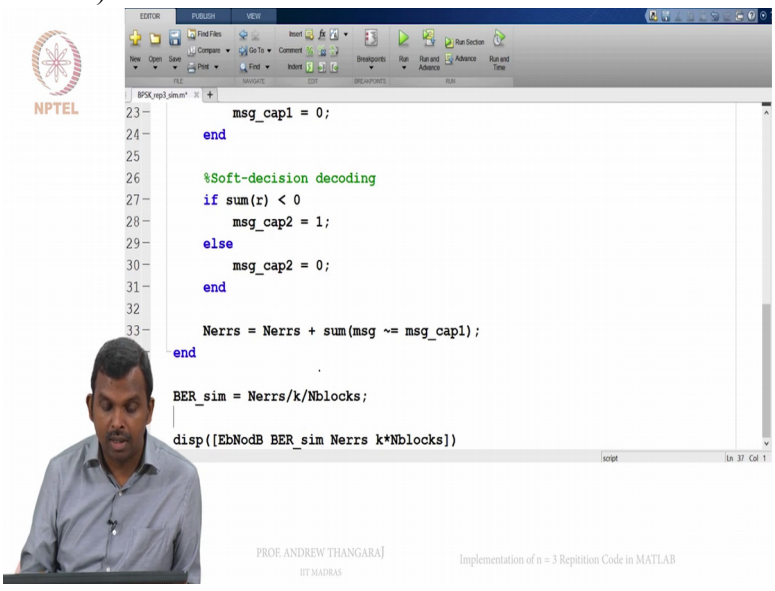
(Refer Slide Time 08:48)



by Nblocks.

So for every block you have one message bit here. There is no B E R t h and we can simply plot this, Ok. So this is how many bits, how many message bits were transmitted, how many errors happened and what B E R you got in

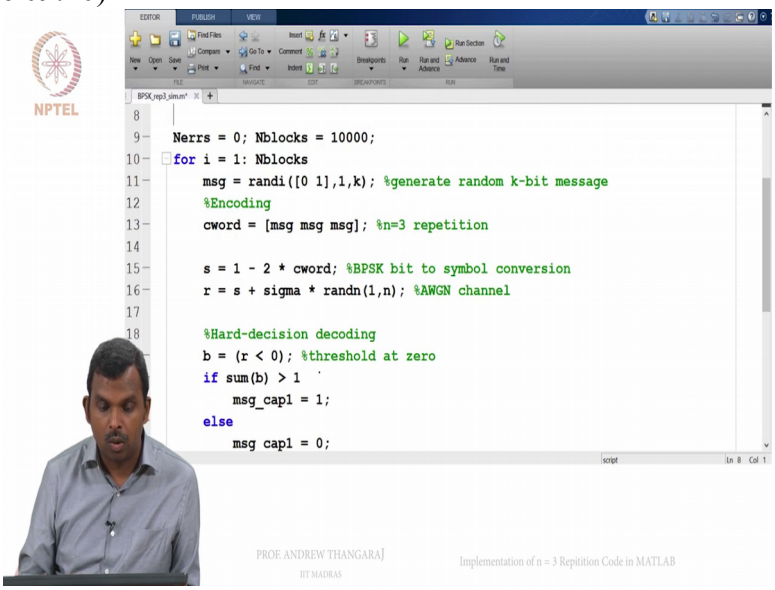
(Refer Slide Time 09:04)



simulation, Ok.

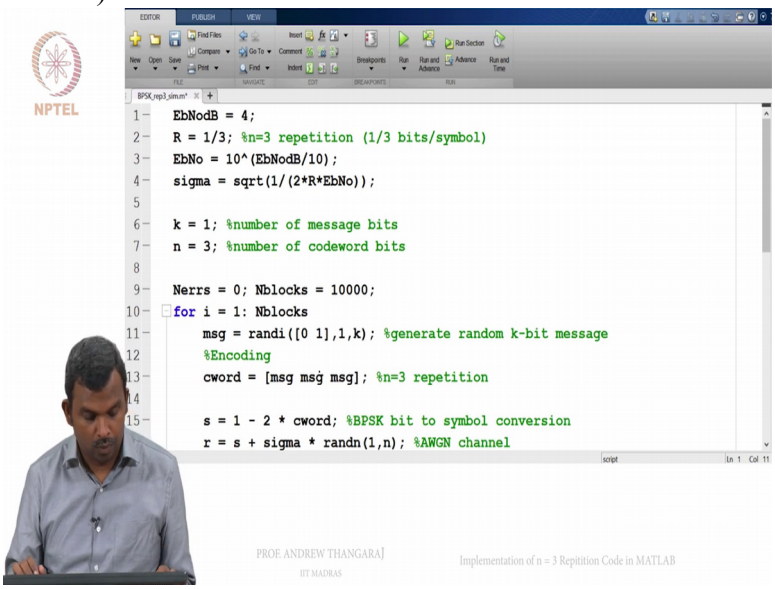
I am going to run this but let us, may be increase the number of blocks to 10000,

(Refer Slide Time 09:10)



Ok and my E b over N naught maybe I will keep it as 4,

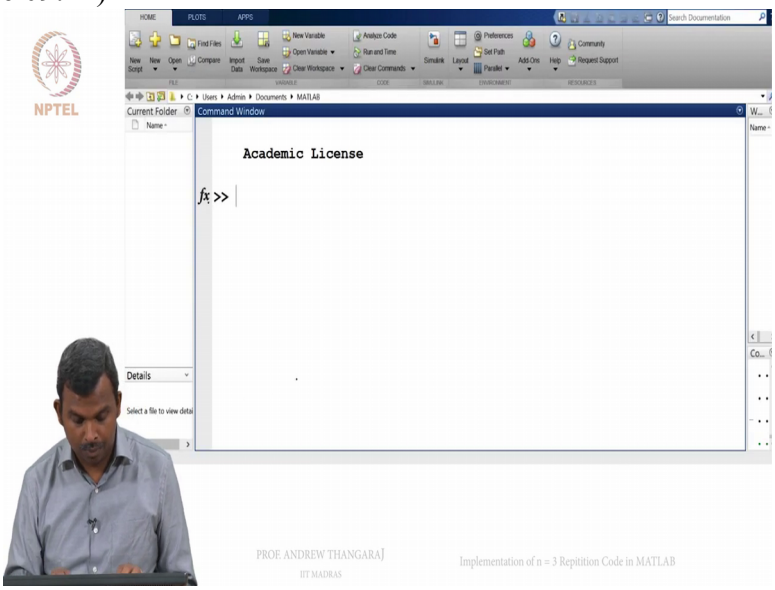
(Refer Slide Time 09:14)



Ok. So let us run this. So my, the name of this file is b p s k rep 3 sim.

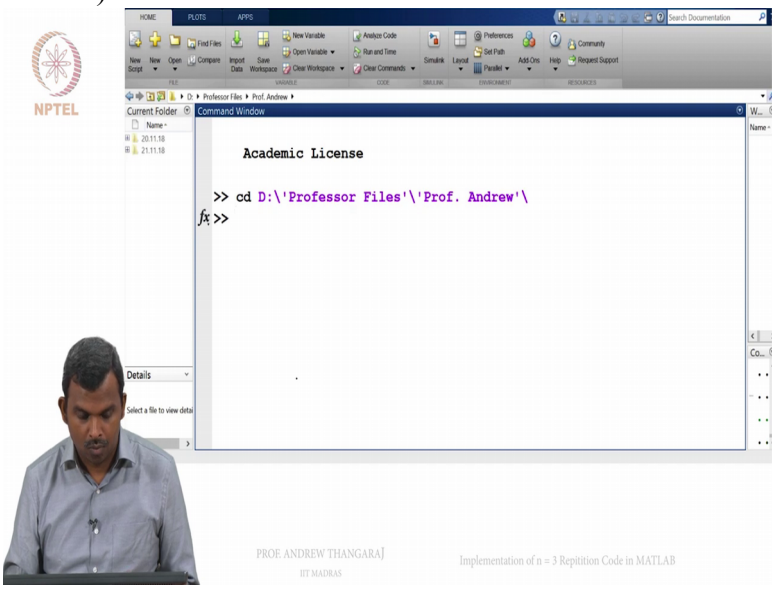
So let me go to the command window.

(Refer Slide Time 09:24)

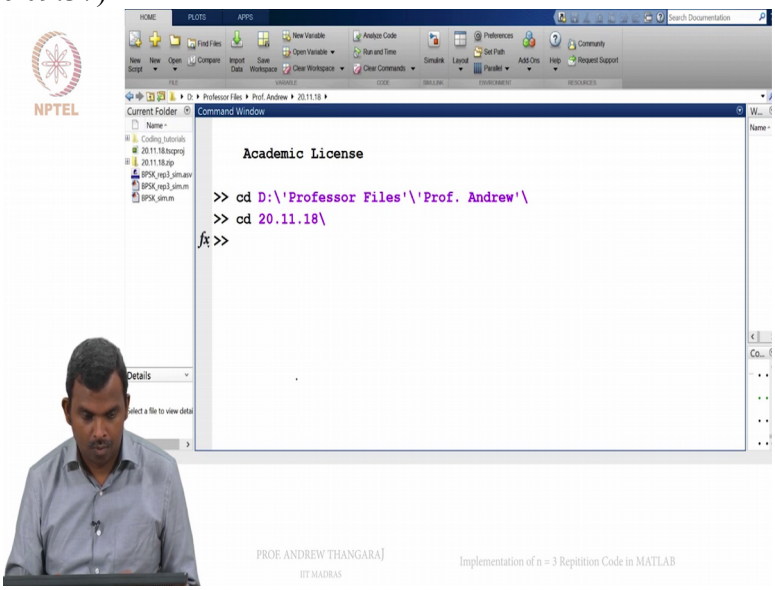


I do not think I am in the right directory.

(Refer Slide Time 09:33)

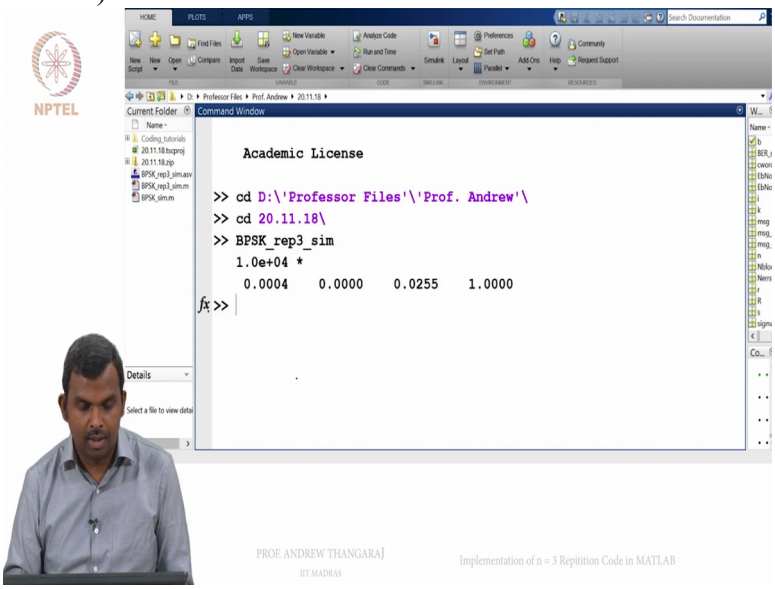


(Refer Slide Time 09:37)



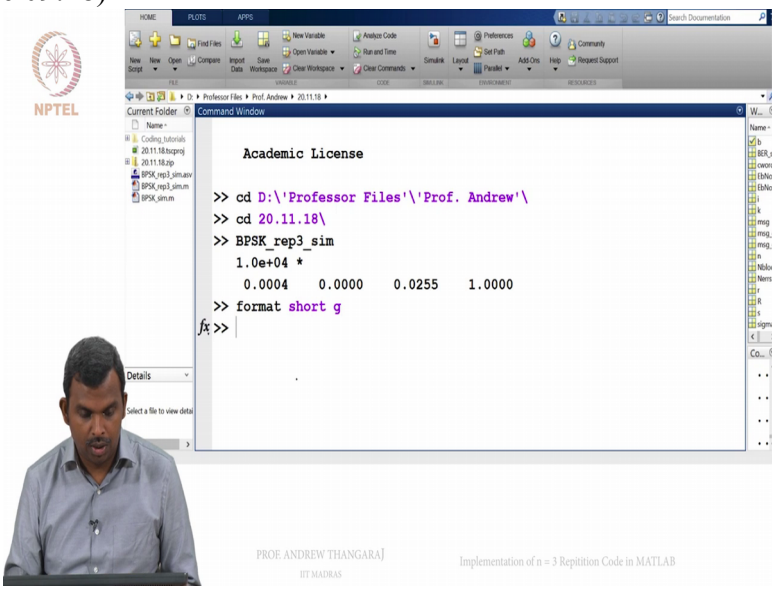
Ok. So Ok,

(Refer Slide Time 09:44)

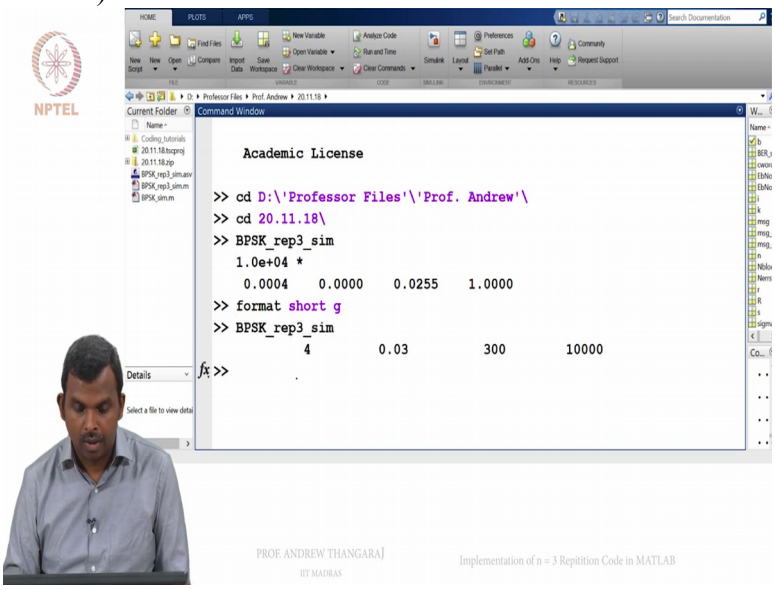


so let us run this. So I need to do format short g

(Refer Slide Time 09:48)



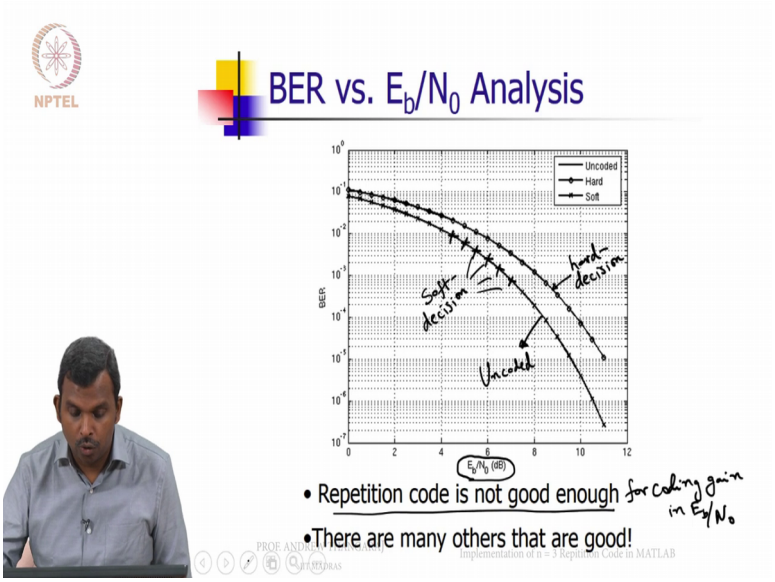
(Refer Slide Time 09:49)



so there you here. So it gives you an error of point zero 3. There were 300 errors if you remember, is a good statistic. 300 errors is not too bad and we ran 10000 blocks.

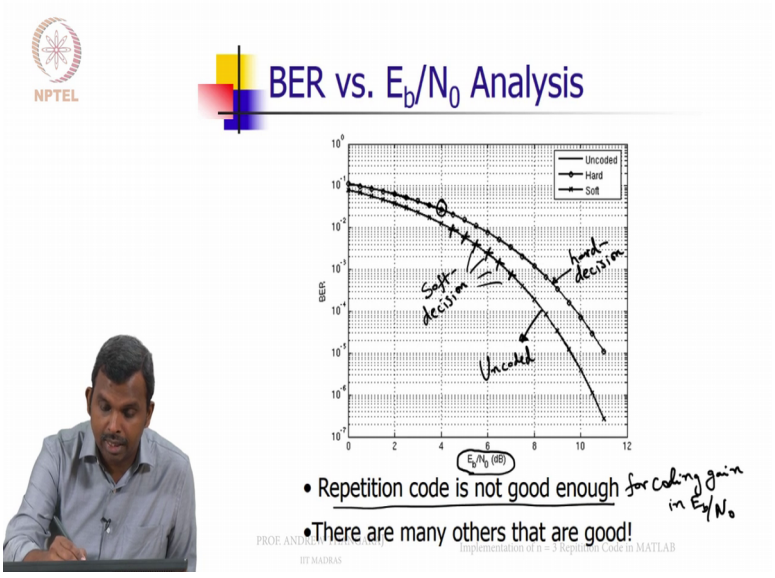
So let us compare and see if this is sort of expected in the plot, Ok.

(Refer Slide Time 10:08)



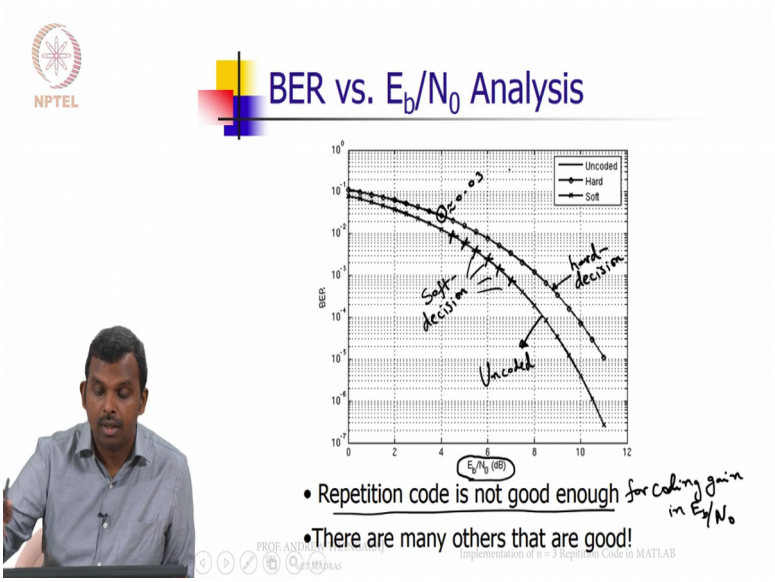
So 4 d B E b over N naught, 1,2,3 so this is point, point 3 so you can see 4 d B E b over N naught this is the number

(Refer Slide Time 10:19)



that I expect and that is point zero 3, Ok.

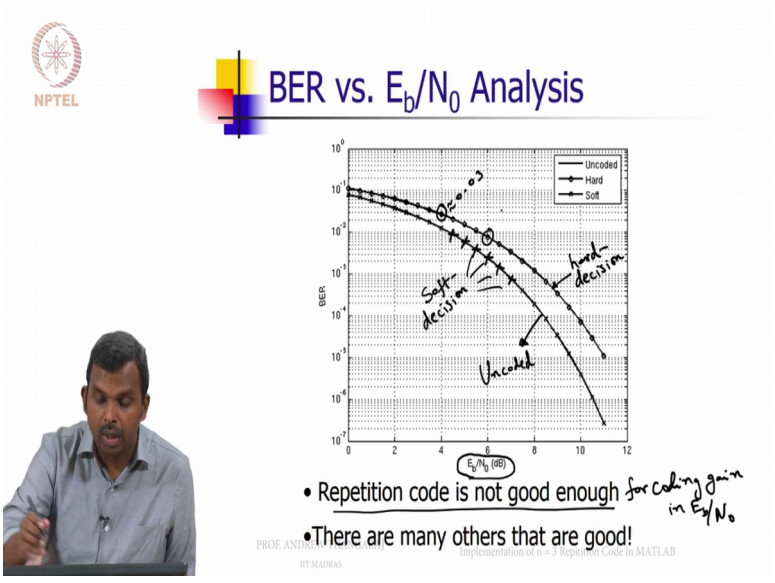
(Refer Slide Time 10:24)



So it is simulating correctly, may be one can do 6 d B as well.

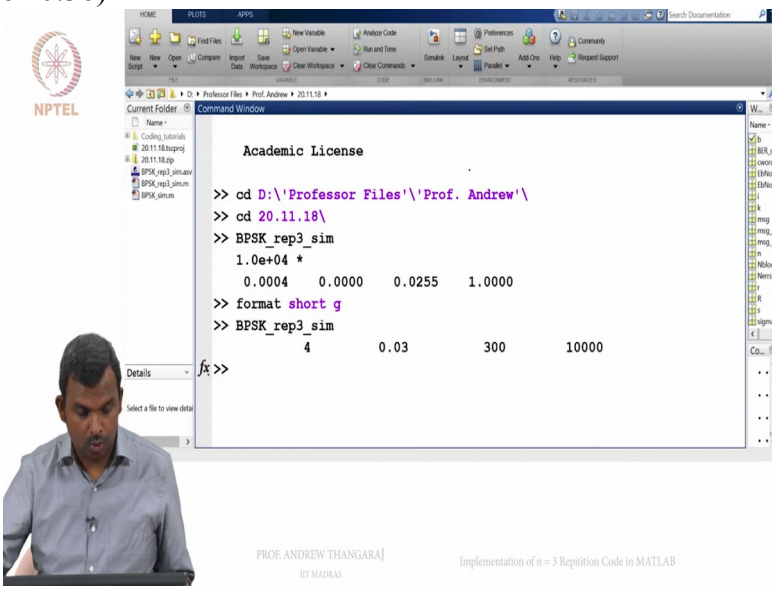
6 d B I expect around 7 or 8

(Refer Slide Time 10:31)



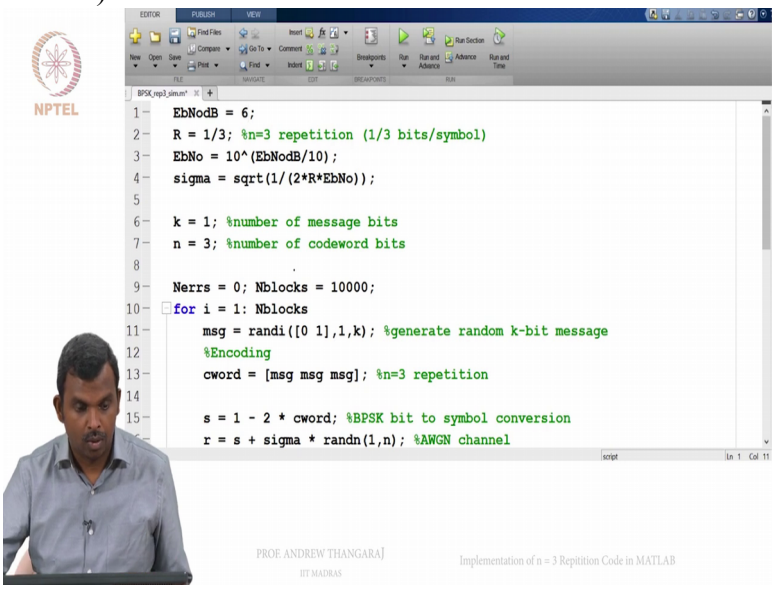
into 10 power minus 3 so let us, let us run that as well so

(Refer Slide Time 10:36)



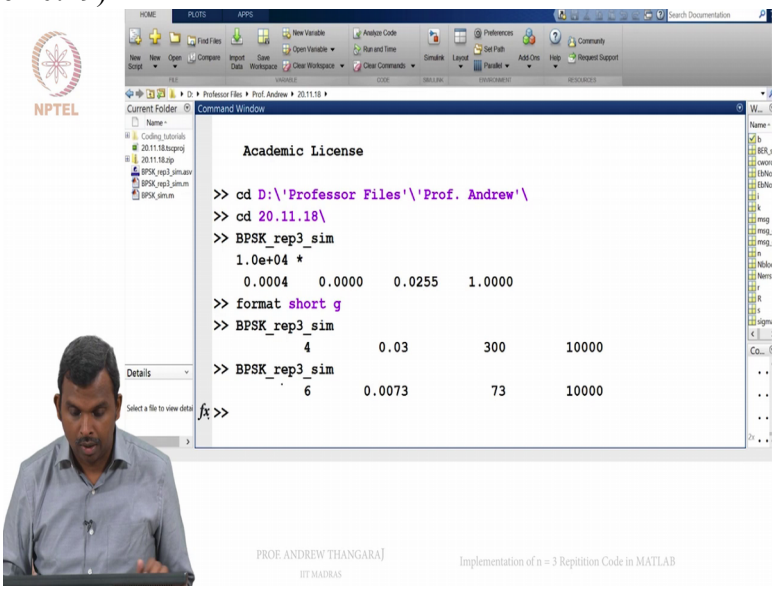
to get that I am going to change the E b over N naught to 6,

(Refer Slide Time 10:42)



Ok, save it and then come back to the command window then run it again,

(Refer Slide Time 10:49)

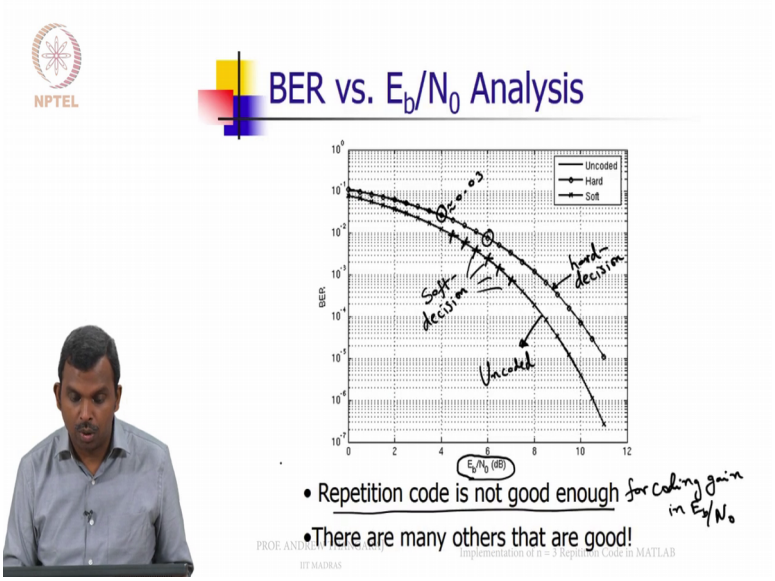


Ok.

So you had 73 errors, may be this is not very reliable, may be you can increase the number of blocks by 1 more but the answer we got was what we expected, around 7 or 8 into 10 power minus 3.

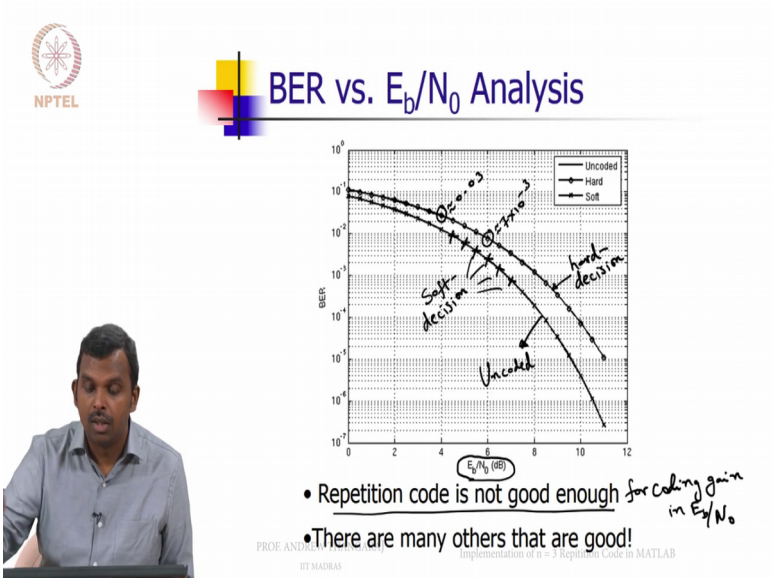
Let me show that

(Refer Slide Time 11:01)



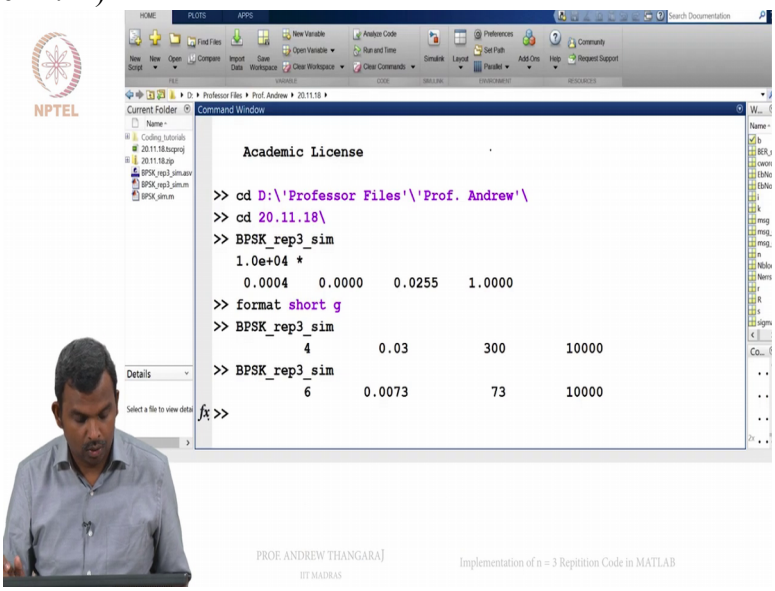
to you once again. This was around roughly 7 into 10 power minus 3, Ok.

(Refer Slide Time 11:09)



we are getting the right, right curve, so it is working out quite well. Let us do soft decision decoding for the same s n rs, same E b over N naught 4 and 6,

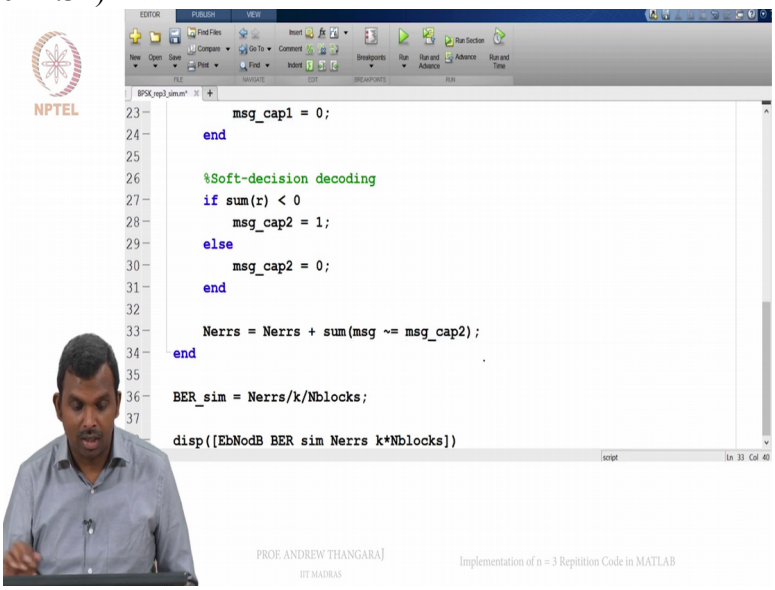
(Refer Slide Time 11:21)



Ok.

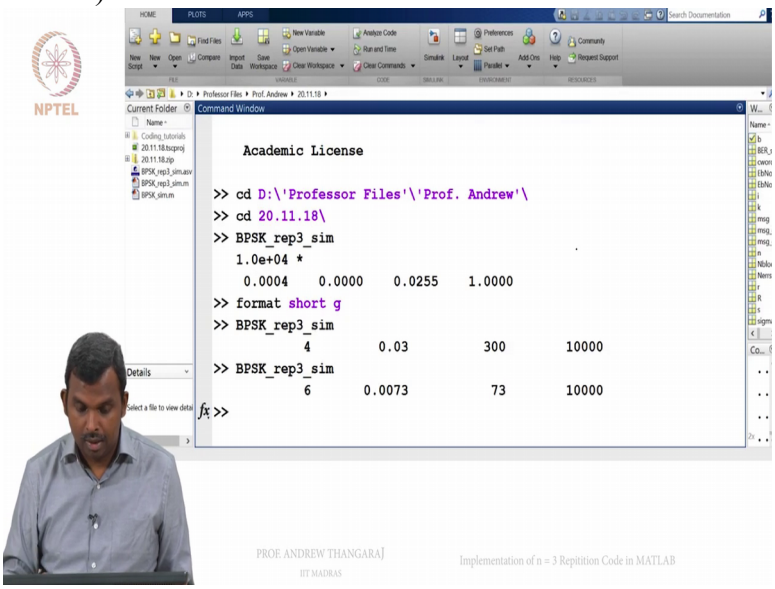
So I am going to do soft decision decoding, so for that all I have to do is to change my message cap 1 to message cap 2. I am doing actually both.

(Refer Slide Time 11:31)



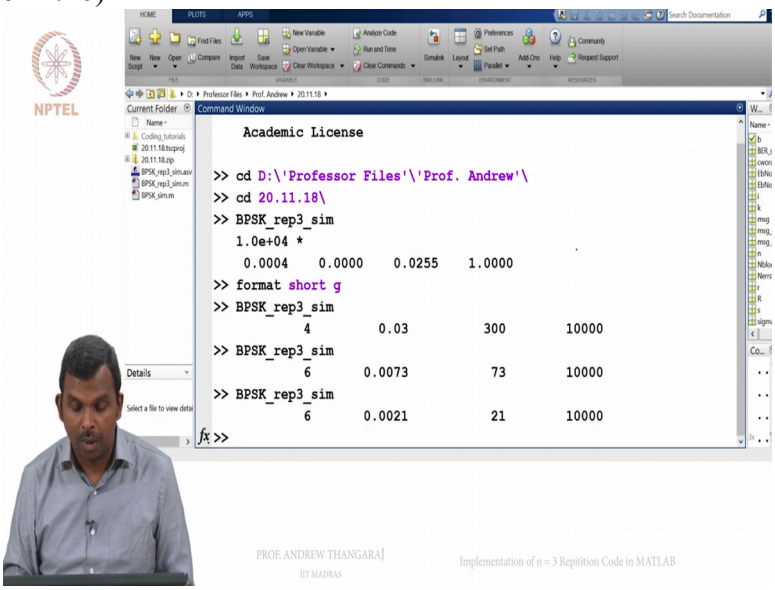
I am just not using one of them. So let us go back to the command window, Ok

(Refer Slide Time 11:40)



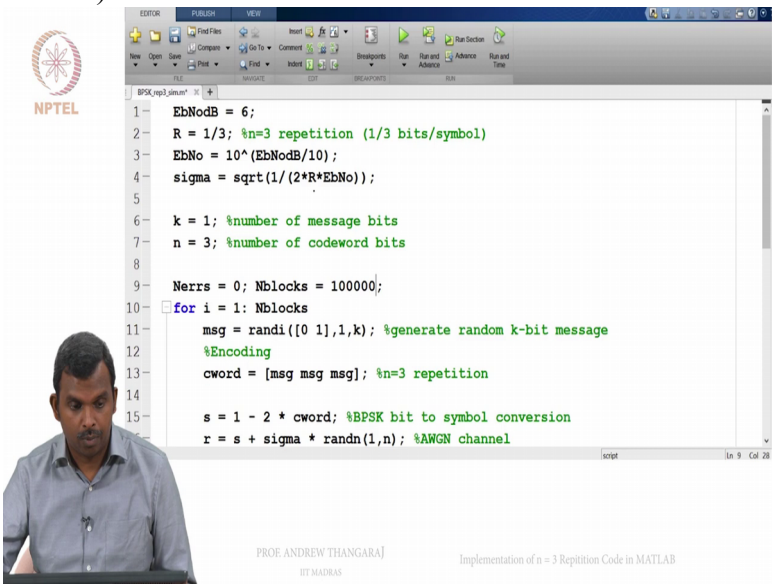
and now remember I am at 6 d B E b over N naught, 6 d B E b over N naught and soft decision decoding.

(Refer Slide Time 11:46)



Doing this and you can see the gains, right. So there were only 21 errors, may be this is not very good in terms of reliability. So may be one can increase the number of blocks to say 100000.

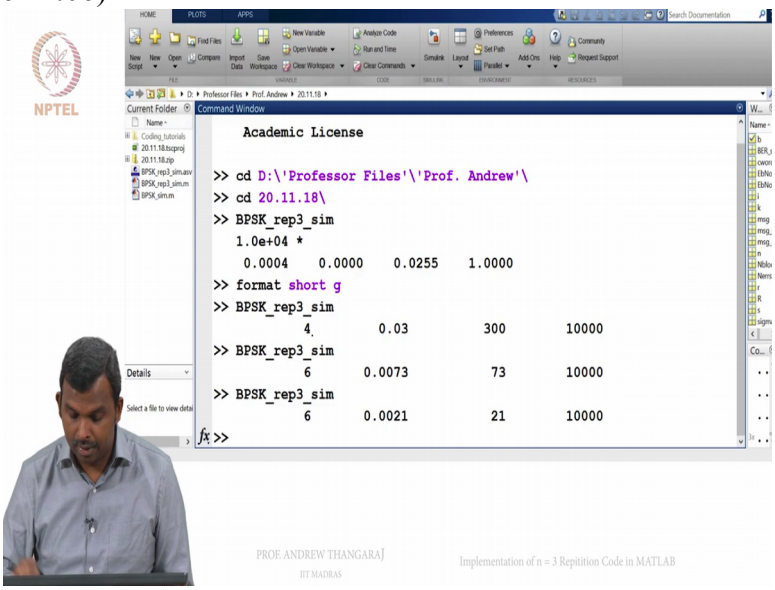
(Refer Slide Time 12:00)



Ok, so this will also give you reasonably good answers.

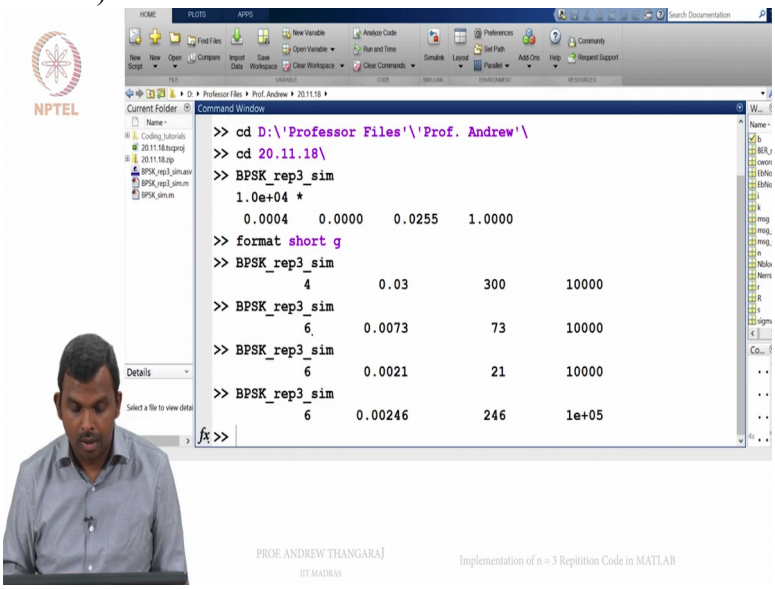
So

(Refer Slide Time 12:06)



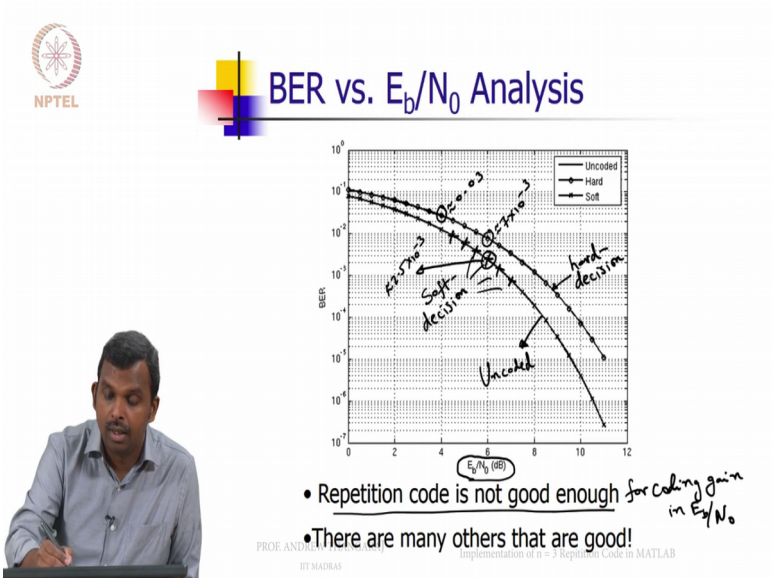
if you repeat this,

(Refer Slide Time 12:08)



so you are getting 100000 blocks simulated 2 point 4 6 into 10 power minus 3, so one can check that this is the correct value as well. So at 6 d B E b over N naught this is around 2 point 5. Right, this point is around, roughly around 2 point 5 into 10 power minus 3, Ok.

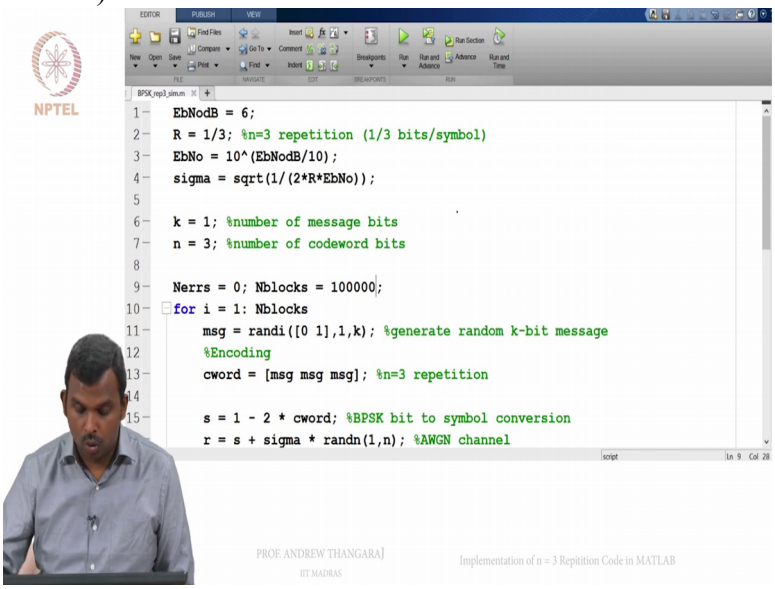
(Refer Slide Time 12:35)



So that is pretty good, so what we got in simulation agrees with that and you can see the gains at the same E b over N naught because you did better decoding, soft decision decoding. You had a gain of about factor of 1 half, a little bit more, Ok so 2 point 5 into 10 power minus 3 was the answer we got.

So hopefully the, this little coding exercise was good in terms of showing you how to, how to do

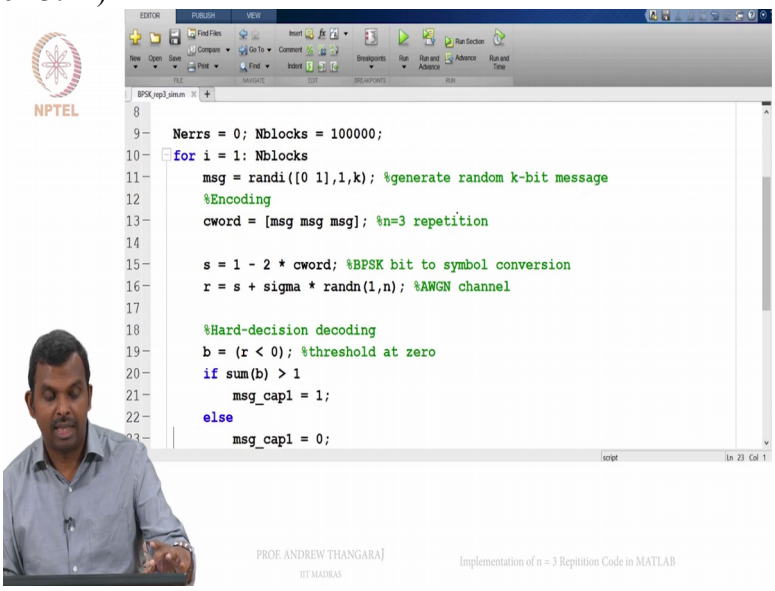
(Refer Slide Time 13:04)



error control codes. Let me quickly run through.

The first thing is the rate, Ok. You have to modify that and that changes your sigma calculation and you have number of bits in each, number of message bits in each block, number of codeword bits in each block, how many blocks you want to simulate and inside that you have

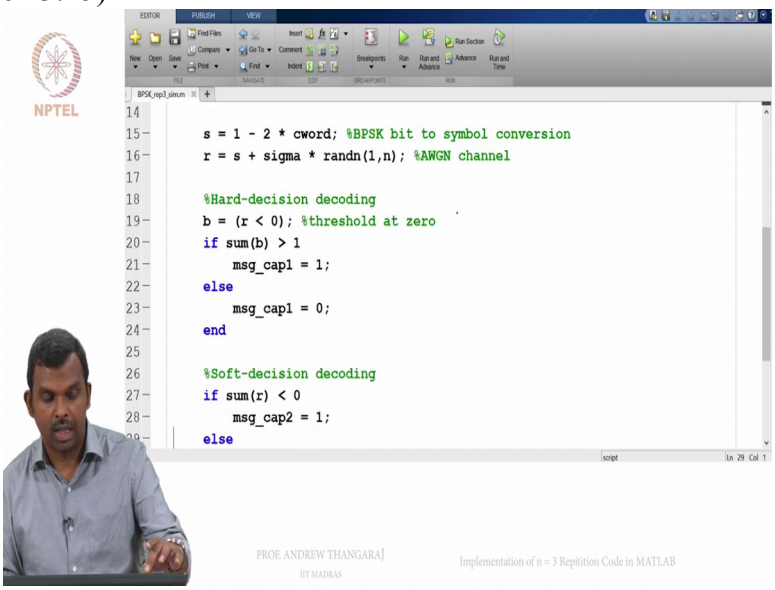
(Refer Slide Time 13:21)



the same picture.

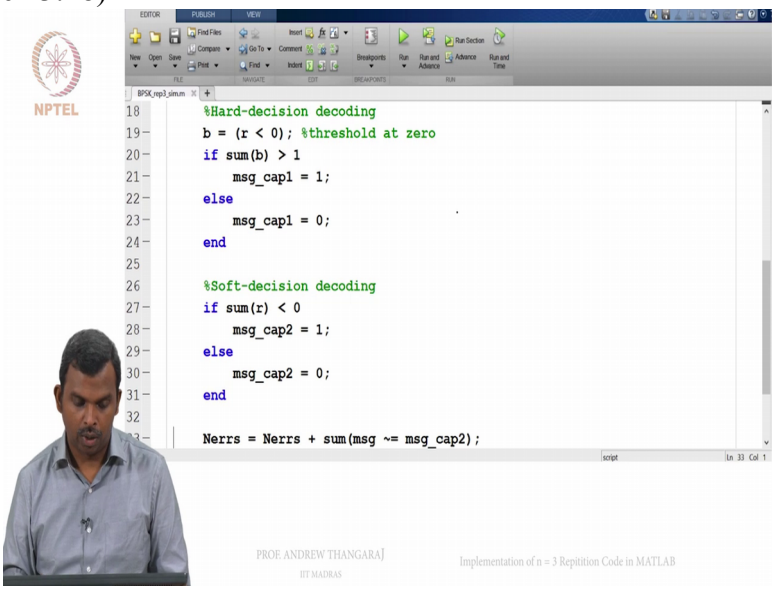
You are encoding, you do the transmission, you could choose to do either

(Refer Slide Time 13:25)



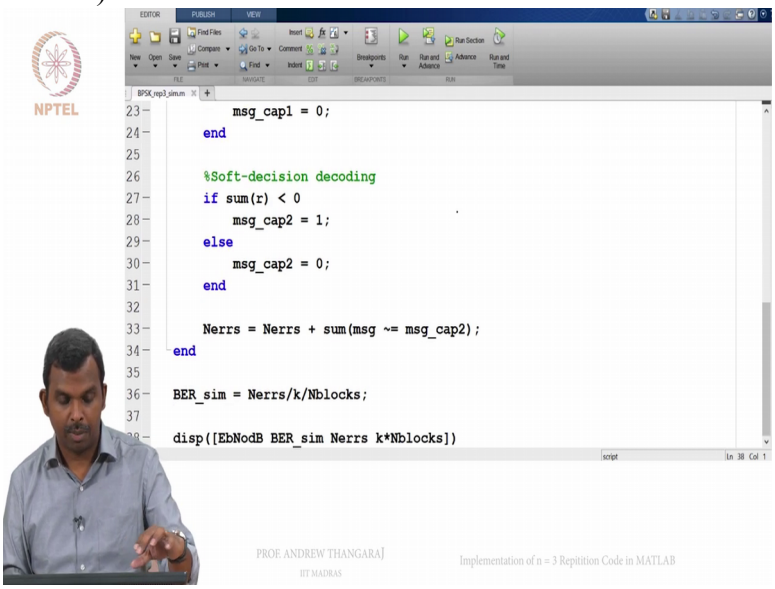
hard decision decoding or soft decision

(Refer Slide Time 13:28)



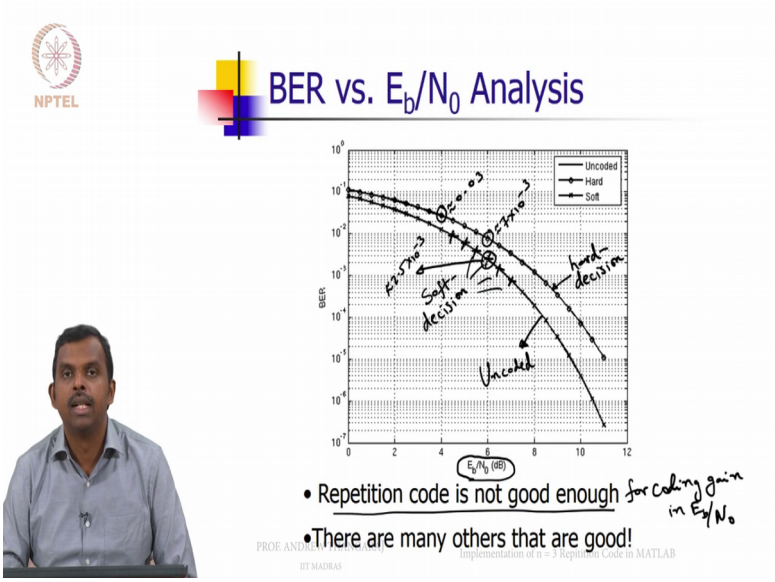
decoding, Ok. This is how it works. And then

(Refer Slide Time 13:32)



there are these minor adjustments in calculating the number of errors and all that. So that is repetition code for you,

(Refer Slide Time 13:38)



Ok.

So, so far we saw the repetition code. We saw how to implement it, may be little bit of analysis and then how to implement that in Matlab as well. And we generated these curves and we plotted it and we saw that at the end of the day, we did not get coding gain in E b over N naught that is because of the way the rate is normalized in the s n r, Ok?

So next code we are going to see a very famous code, it is called Hamming code and in that code we will get some coding gain, Ok. It is, it is the first real code that we are going to see. So let us go through that.