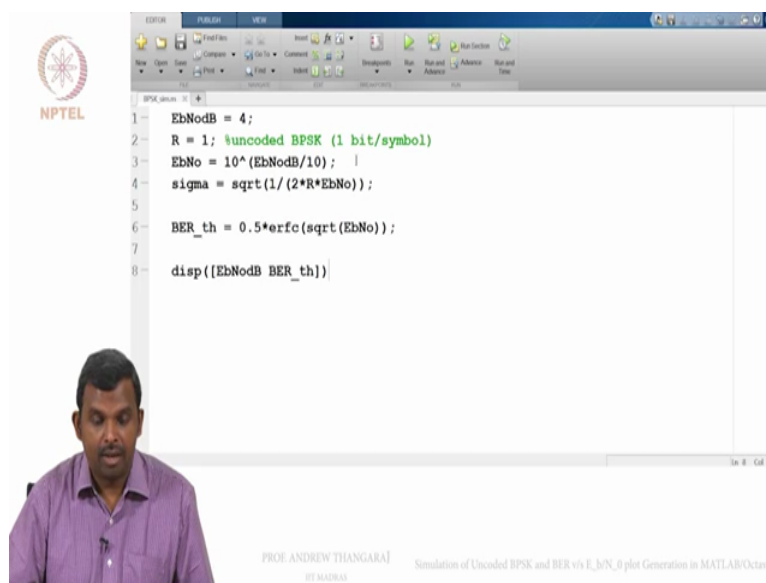


LDPC and Polar Codes in 5G Standard
Simulation of Uncoded BPSK and BER versus Eb/No plot Generation in
MATLAB/Octave
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology, Madras

So for this it is common to use the editor that comes in the interface so we will write a script in MATLAB okay, I am going to start typing hereso we are ready to start typing the little program so if you remember the communication system bits are going to get generated, they are going to go through this BPSK mappers, 0 to plus 1, 1 to minus 1 and then we are going to have noise getting added and the receiver does the thresholding at 0 right.

(Refer Slide Time: 01:04)



So that is the simple system but to make theoretical plot you just need to evaluate the Q function right so you do not need anything more beyond that, so let us say the first conversion is for rate and Eb over N not and all that so the way I usually like to write my code is to first define Eb over N not in dB okay so this could be let us say 4 dB okay and then the rate okay the rate is 1 okay.

So how did I get these two things Eb over N not could be anything else you just have to fix that you can change it to any other number okay the rate is 1 because I am doing uncoded BPSK okay so this is 1 bit per symbol okay, so tomorrow if you do coding we might change this parameter R but for now R is 1 okay so once you do this I know my formula for Eb over N not.

(Refer Slide Time: 02:06)

The slide contains the following handwritten formulas:

$$Q(x) = 0.5 \operatorname{erfc}\left(\frac{x}{\sqrt{2}}\right)$$
$$\frac{E_b}{N_0} = \frac{1}{2R\sigma^2}$$
$$\sigma = \sqrt{\frac{1}{2R\left(\frac{E_b}{N_0}\right)}}$$

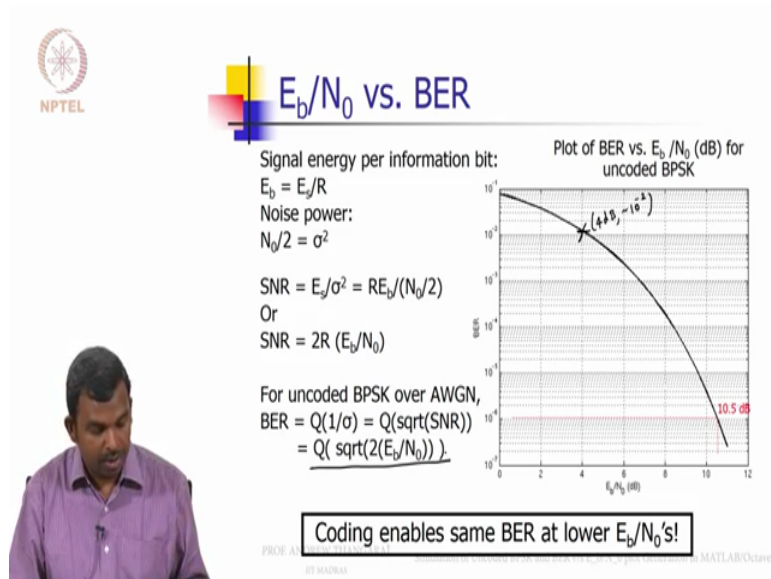
A note on the right side of the slide states: $\frac{E_b}{N_0} = 10$ (dB).

E_b over N not is 1 by $2 R$ sigma squared so from that formula I can find what sigma should be so for that it is useful to first convert E_b over N not to from dB to actual value okay so maybe I will write a quick little description here and then show you how that works okay, so E_b over N not is 1 by $2 R$ sigma squared so sigma is square root of 1 by $2 R$ times E_b over N not now remember we are going to have E_b over N not in dB, so E_b over N not is actually 10 power E_b over N not in dB divided by 10 right.

So if you have E_b over N not in dB you divide it by 10 raise to the power 10 you will get E_b over N not okay so you can plug that back in okay so if you do that you will get the value for sigma okay so from E_b over N not in dB right so which is 4 or 5 or something we can go to sigma, sigma is the noise variance of the noise that you add okay so this little formula one needs to do.

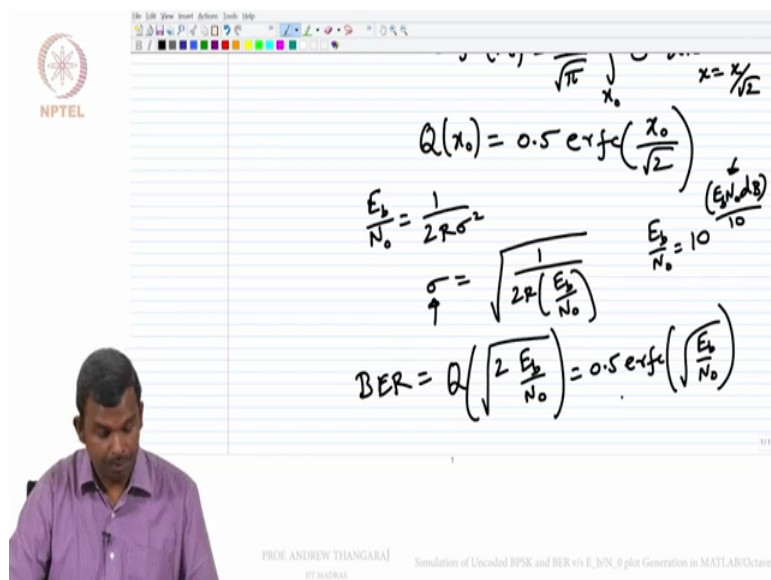
You can do it in multiple ways depending on your taste and how you like to program so maybe I will write like this, I will do an explicit conversion from E_b over N not to E_b , 10 power E_b N not dB by 10 and then you can do sigma equals sqrt 1 by 2 by R by or if you do not like this you can write 2 into R into E_b N not okay, so that gives you the value for sigma so this is most of the battle done okay so all these are useful conversions, we should do that in the beginning so some sort of a setup for either the simulation or the theoretical plot that you might have okay.

(Refer Slide Time: 04:05)



So now if you remember the value for bit error rate as a function of, value for bit error rate as a function of E_b over N not is given by this formula right it is just Q of square root of 2 times E_b over N not right so once I have E_b over N not, I just have to plug it into this and get my value for bit error rate but I know my Q is implemented as $0.5 \operatorname{erfc}$ of x by root 2 so if I combine those two, I will get my value okay.

(Refer Slide Time: 04:34)

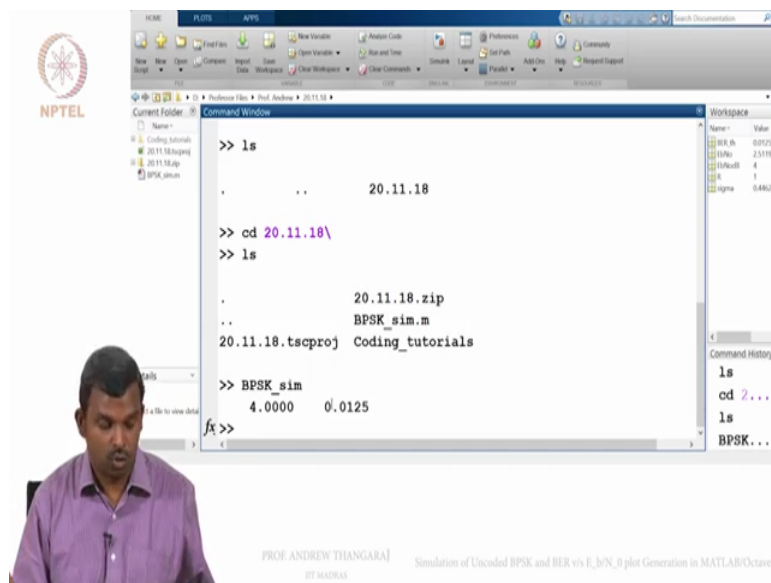


So let me just write that down once again okay so BER is Q of square root of 2 times E_b over N not okay and what is Q ? Q is $0.5 \operatorname{erfc}$ of whatever is inside the argument of Q you have to divide by root 2 so this root 2 will be cancelled so you simply get E_b over N not okay so this

is a little formula that one might want to remember and let us type into my MATLAB code okay.


So how do I find bit error rate, bit error rate theory okay, this is the theoretical formula is $0.5 \text{ erfc}(\sqrt{E_b/N})$ not that is it okay, so we can do a little display here, display E_b/N not dB and BER th so this will just show you what the two values are and let us try to run this so if you run it you might have to save it so let us save it I am going to save it into my folder that is available here and make a little sub folder here maybe that is not easy to so I will just leave it in this folder itself okay, so the title I will give as, I will just call it BPSK okay.

(Refer Slide Time: 06:46)



So that is just a file we have and now we are ready to run it and to run it maybe we should run it from the command line, you can also run it from the thing but let us run it from the command line, there it is and if you check what is in the directory or I need to go into this 20 dot directory okay so you can see my program BPSK sim.m we can try to run it, let us see if there are any errors, there are no errors and as expected if you remember 4 dB in the plot gave you around 10 to power minus 2 and you can see that that is panning out here, 4 dB is 10 power minus 2.

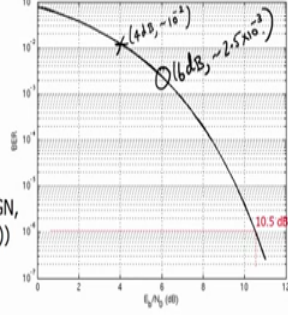
(Refer Slide Time: 07:48)



E_b/N_0 vs. BER


Signal energy per information bit:
 $E_b = E_s/R$
Noise power:
 $N_0/2 = \sigma^2$
 $SNR = E_s/\sigma^2 = RE_s/(N_0/2)$
Or
 $SNR = 2R (E_b/N_0)$
For uncoded BPSK over AWGN,
 $BER = Q(1/\sigma) = Q(\sqrt{SNR})$
 $= Q(\sqrt{2(E_b/N_0)})$

Plot of BER vs. E_b/N_0 (dB) for uncoded BPSK



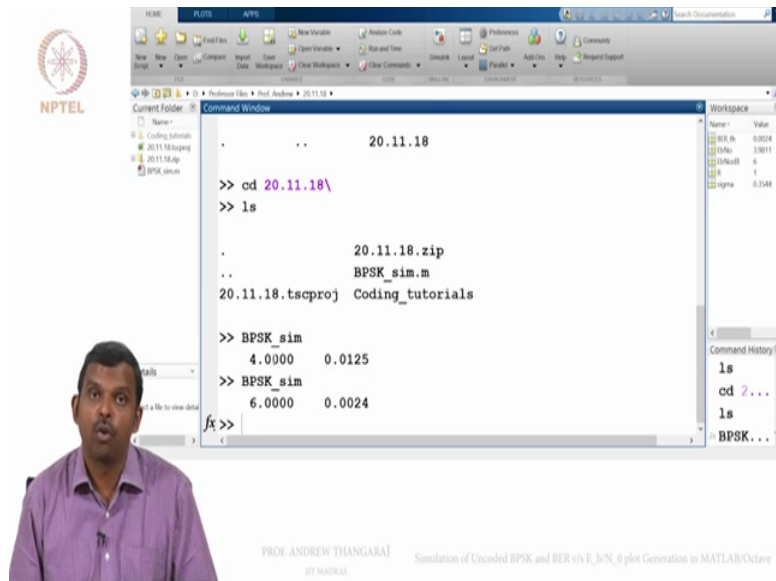
Prof. A. [unreadable] IIT MADRAS MATLAB/Octave

Coding enables same BER at lower E_b/N_0 's!



```
1 EbNodB = 6;  
2 R = 1; %uncoded BPSK (1 bit/symbol)  
3 EbNo = 10^(EbNodB/10);  
4 sigma = sqrt(1/(2*R*EbNo));  
5  
6 BER_th = 0.5*erfc(sqrt(EbNo));  
7  
8 disp([EbNodB BER_th])
```

Prof. ANDREW THANGARAJ IIT MADRAS Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave



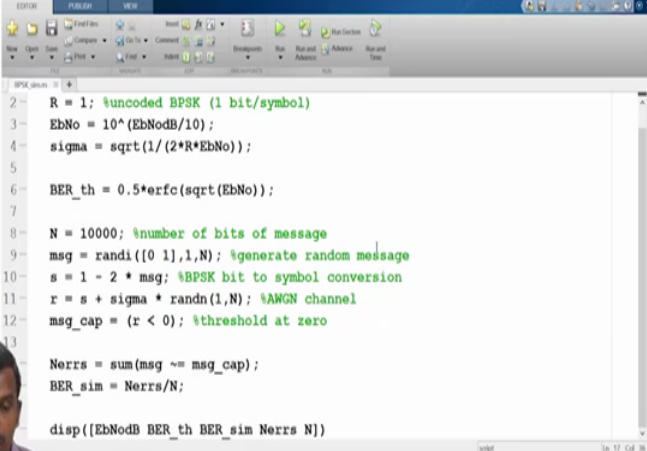

So maybe we should check on the theoretical plot for let us say 6 dB. 6 dB is around 2 to 3 so let us $2.5 \times 10^{\text{power} - 3}$ right so that is the plot so that is let us see if that comes about in our BPSK sim so if I want 6 dB I have to go back to my editor and change the E_b over N not to 6 dB and then save it you can come back to the command window and then run it again you will get $2.4 \times 10^{\text{power} - 3}$ so that is close enough and that is how I generated this plot.

So if you want to for instance get all these numbers together you can make some simple modifications to the editor, you can make the E_b/N not dB as a vector and maybe you might want to make some small changes here and there to make that work out but this is very basic piece of code to do that okay.

So now how do I verify the BER theoretical with the BER simulation okay so if you remember I have mentioned you can write some simulation to actually transmit the messages count the number of errors and divide by the total number of bits and to find the bit error rate let us do that next and then compare the two okay.

So for that I need to know how many bits I am going to send, let me fix it at 10 thousand okay so if some number you can play around with you can change it if you like and then first I will need the message vector okay so I will need to generate 10 thousand bits of message because it is rate 1 uncoded BPSK, to generate random bits the current code in MATLAB is to use this command called `randi` so I am going to do that for the message.

(Refer Slide Time: 09:54)



```
2 R = 1; %uncoded BPSK (1 bit/symbol)
3 EbNo = 10^(EbNodB/10);
4 sigma = sqrt(1/(2*R*EbNo));
5
6 BER_th = 0.5*erfc(sqrt(EbNo));
7
8 N = 10000; %number of bits of message
9 msg = randi([0 1],1,N); %generate random message
10 s = 1 - 2 * msg; %BPSK bit to symbol conversion
11 r = s + sigma * randn(1,N); %AWGN channel
12 msg_cap = (r < 0); %threshold at zero
13
14
15 Nerrs = sum(msg ~= msg_cap);
16 BER_sim = Nerrs/N;
17
18 disp([EbNodB BER_th BER_sim Nerrs N])
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave

rand i 0 okay so 0 1, 1, N so this will generate for me string of random binary 0 and 1 values of length N, vector of length N of this, so I will illustrate this a little bit before I run it but let me finish up my coding and then you can go back and see it so now I need to know so this is number of bits of message, this is generate random message and the next thing I do I have to do BPSK right.

So I have to take my message to plus 1 and minus 1 every time I have a 0 and the message it should become plus 1, 1 should become minus 1, there is usually a little device that are used so let is call it s, s is 1 minus 2 into msg okay so this is BPSK bit to symbol conversion okay, so you can check what it does, if msg is 0, I am going to get plus 1, if msg is 1 I am going to get minus 1 okay so this is quick and dirty way to convert this BPSK modulation into a simple MATLAB line of code okay.

So I have my symbol vector, now I need my received vector R okay so that is my transmission, the r is going to be s plus Gaussian noise, Gaussian noise with sigma as the standard deviation so MATLAB has this utility called rand n which will generate Gaussian distributed random variables and how many of them do I want? N of them okay, so this will generate N, rand n generates unit variance distribution values from that distribution so you multiply by sigma to get variance of sigma square okay so this is r and then now you can run your thresholding device, so this is AWGN channel.

And then now we have to do a thresholding right, so if r is greater than 0, I am going to say the transmitted symbol was plus 1 or the transmitted message bit was 0 okay and if r is less

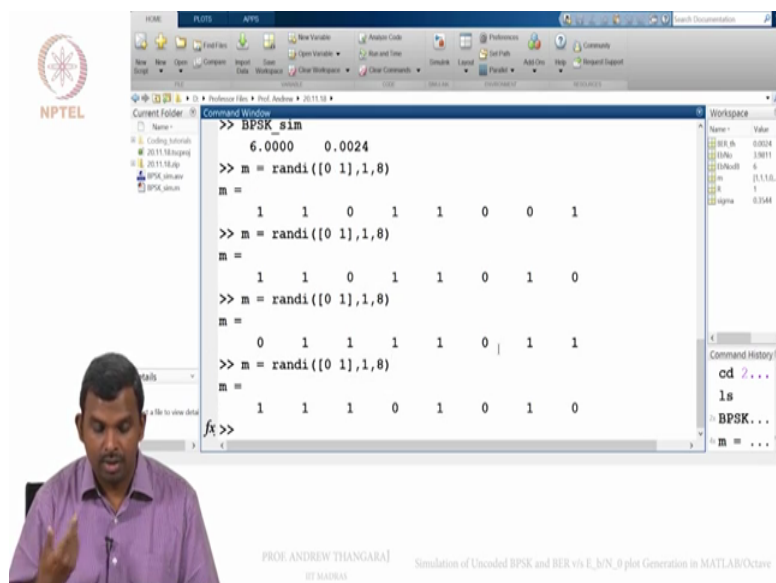
than 0, I am going to say the transmitted symbol was minus 1 and the transmitted message was 1 okay so the way to again do this very easily, the message cap which is m_{cap} which is what is decided on the other side is to simply say $r < 0$ okay so this is the threshold okay.

So what will happen here is whenever r is negative this condition is true okay and true means 1 in as a number if you want to compare it as a number true means 1 okay and if r is positive this condition is false and false is 0 okay, so this quickly does a thresholding at 0 and gives you the bit that you want okay so have a message and a message cap and now I can compare message and message cap, message is what I sent, message cap is what I received so I can compare and how do I compare? Comparing is not too bad, you can do it in multiple ways but one can get a DER sim from the comparison okay.

So I am going to see if msg is not equal to msg_{cap} okay, so this will generate wherever they are not equal it will generate a 1 and wherever it is equal it will generate a 0 so you can add it all up, this will give you the number of errors and you divide by N , this will give you the error rate okay.

So whenever you do Monte Carlo simulations like this it is good actually to have the number of errors separately stored so you might want to have that also, I will tell you why you need this, so let us store that in one variable and then divide to get the error rate and when you display it is good to display BER sim and the number of errors and the end okay, so in one shot you get a glimpse of what has happened, the E_b over N not in dB, the BER theoretical, the BER that you got from simulation and errors and N okay.

(Refer Slide Time: 15:12)



The image shows a MATLAB/Octave Command Window with the following code and output:

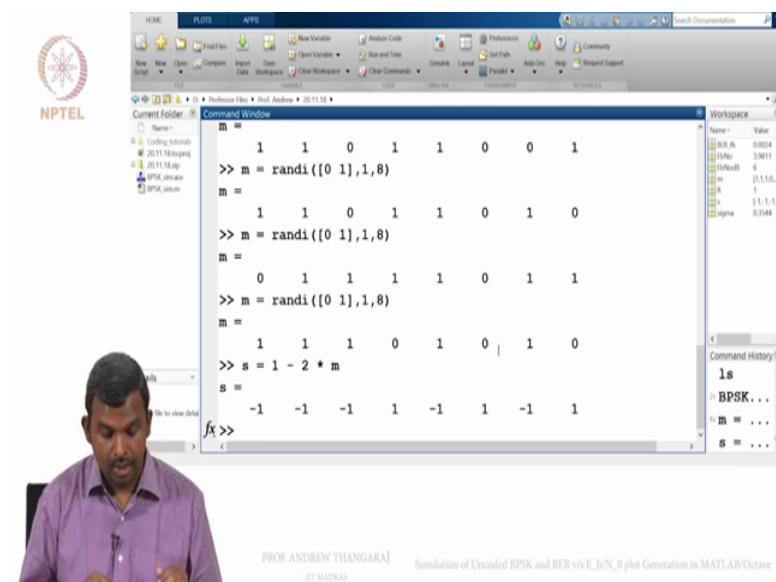
```
>> BPSK_sim
6.0000 0.0024
>> m = randi([0 1],1,8)
m =
     1     1     0     1     1     0     0     1
>> m = randi([0 1],1,8)
m =
     1     1     0     1     1     0     1     0
>> m = randi([0 1],1,8)
m =
     0     1     1     1     1     0     1     1
>> m = randi([0 1],1,8)
m =
     1     1     1     0     1     0     1     0
fx>>
```

The workspace on the right shows variables: BER_b (0.0024), BER_P (3.9811), EbN0 (6), m (1x8 double), s (1x8 double), and sigma (0.3344). The Command History shows: cd /... , ls , BPSK... , m = ...

PROF. ANDREW THANGARAJ
IIT MADRAS
Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave

So this is my little code, let me save this okay so this does simulation and theory, what I am also going to do is show you some of this commands just to give you a sense of how they work okay so let us go to the MATLAB command window, so I showed you this code `rand i` so `rand i`, I call it `m`, `rand i` of let us say 0 1 and then 1, just for illustration we will use some 8 okay.

(Refer Slide Time: 15:52)



The image shows a MATLAB/Octave Command Window with the following code and output:

```
     1     1     0     1     1     0     0     1
>> m = randi([0 1],1,8)
m =
     1     1     0     1     1     0     1     0
>> m = randi([0 1],1,8)
m =
     0     1     1     1     1     0     1     1
>> m = randi([0 1],1,8)
m =
     1     1     1     0     1     0     1     0
>> s = 1 - 2 * m
s =
    -1    -1    -1     1    -1     1    -1     1
fx>>
```

The workspace on the right shows variables: BER_b (0.0024), BER_P (3.9811), EbN0 (6), m (1x8 double), s (1x8 double), and sigma (0.3344). The Command History shows: ls , BPSK... , m = ... , s = ...

PROF. ANDREW THANGARAJ
IIT MADRAS
Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave

So you can see it generated a random bit, if you run it again it will generate another random bit, a 6 sequence of random bits you can keep changing this every iteration will be different etc. so this produces a random number generator okay so just has sequence like this okay so

now once I do that, I can transmit my BPSK conversion work like that right so you can see how this is working 1 goes to minus 1, 0 goes to plus 1 okay.

(Refer Slide Time: 16:04)

The screenshot shows a MATLAB Command Window with the following code and output:

```

s =
    -1    -1    -1     1    -1     1    -1     1
>> randn(1,8)
ans =
Columns 1 through 6
   -1.0689   -0.8095   -2.9443    1.4384    0.3252   -0.7549
Columns 7 through 8
    1.3703   -1.7115
>> randn(1,8)
ans =
Columns 1 through 6
   -0.1022   -0.2414    0.3192    0.3129   -0.8649   -0.0301
Columns 7 through 8
   -0.1649    0.6277
fx>> randn(1,8)
  
```

Below the Command Window, the speaker Prof. Andrew Thangaraj is visible. The slide title is "Simulation of Uncoded BPSK and BER v/s E_b/N₀ plot Generation in MATLAB/Octave".

The screenshot shows the next step in the MATLAB simulation:

```

r =
Columns 1 through 6
   -0.4970   -0.8967   -0.9299    1.5627   -1.2851    1.2469
Columns 7 through 8
   -0.7041    0.9136
>> s
s =
    -1    -1    -1     1    -1     1    -1     1
>> r = s + sigma * randn(1,8)
r =
Columns 1 through 6
   -0.9236   -1.4132   -1.4068    1.0372   -0.7440    1.9163
Columns 7 through 8
   -1.2363    1.0664
fx>>
  
```

The speaker Prof. Andrew Thangaraj is visible below the Command Window. The slide title remains "Simulation of Uncoded BPSK and BER v/s E_b/N₀ plot Generation in MATLAB/Octave".

And then we have the noise getting added, maybe I should show you the noise separately, the normal, so this is normally distributed random variables you can keep changing that, it will change, this is all variance one, so if you want to see sigma squared so you have to multiply by sigma okay so this is noise like that okay so I wanted to get r is s plus this okay and this is r okay and so if you remember I had this m cap which is so you can see what had happened okay just for comparison maybe you need to see s and then this okay.

(Refer Slide Time 17:04)

NPTEL

```

s =
    -1    -1    -1     1    -1     1    -1     1

>> r = s + sigma * randn(1,8)
r =
Columns 1 through 6
   -0.9236   -1.4132   -1.4068    1.0372   -0.7440    1.9163
Columns 7 through 8
   -1.2363    1.0664

>> mcap = (r < 0)
mcap =
     1     1     1     0     1     0     1     0

>> m
m =
     1     1     1     0     1     0     1     0

f

```

PROF. ANDREW THANGARAJ
IIT MADRAS

Simulation of Uncoded BPSK and BER vs E_b/N₀ plot Generation in MATLAB/Octave

So you see minus 1 was transmitted, minus 0.92 is received, minus 1 was transmitted minus 1.4 is received, minus 1 was transmitted minus 1.4 is received so on okay so that is the corresponding number that were received, so now we have this m cap which is r less than 0 and you can see it goes back to the bits okay so you can compare with m, you have the exact same m that was transmitted okay so this is r less than 0 is a convenient device for BPSK, is that okay?

(Refer Slide Time: 17:28)

NPTEL

```

>> BPSK_sum
Undefined function or variable 'BPSK_sum'.
Did you mean:
>> BPSK_sim
    1.0e+04 *
    0.0006    0.0000    0.0000    0.0018    1.0000

>> format short g
>> BPSK_sim
Columns 1 through 4
         6    0.0023883    0.0015         15
Column 5
    10000

>> BPSK_sim
     6    0.0023883    0.003         30    10000

f

```

PROF. ANDREW THANGARAJ
IIT MADRAS

Simulation of Uncoded BPSK and BER vs E_b/N₀ plot Generation in MATLAB/Octave

So we did this for a very small case and if you now I am going to run my BPSK sim if you remember that was for E_b over N not of 6 dB and then we are expecting 2.4 into 10 power minus 3 as the bit error rate, I was doing 10 thousand bit simulation, let us see what we get,

okay sorry, put sum okay so now the display is not that great so maybe we should do a format short g which will make it a bit better and maybe we should get all of these guys in the same line, so let us move this around so there you go.

So this is Eb over N not 6dB this is the theoretical bit error rate expected 2.3 into 10 power minus 3 okay it has so many accuracy and then 0.003 is what I got in simulation, there were 30 errors out of 10 thousand okay so you can see there is not an exact match but close enough but this kind of 30 is, this maybe not a very significant number so what you can do if you want maybe better accuracy is to modify this and you run for a larger number okay so you run it for 100 thousand let us say okay.

(Refer Slide Time: 19:05)

Simulation Length	BER	Theoretical BER	Errors	Power
15	0.0023883	0.0015	15	
10000	0.0023883	0.003	30	10000
100000	0.0023883	0.00208	208	1e+05
100000	0.0023883	0.00224	224	1e+05
100000	0.0023883	0.00256	256	1e+05

PROF. ANDREW THANGARAJ
IIT MADRAS

Simulation of Uncoded BPSK and BER v/s E_b/N_0 plot Generation in MATLAB/Octave

This computer is quite powerful and this program is very simple, it has enough memory so a 100 thousand is not a big deal but if you have a small computer not too much memory then you might have to think about how much you can run it for but you can usually run 100 thousand very easily and you can see the accuracy sort of improves and comes closer to 0.2 and then you can do more simulations if you like, some values will be closer around 0.002 okay.

(Refer Slide Time: 19:22)

Command	Block Length	BER	Number of Errors	Number of Trials	
>> BPSK_sim	6	0.0023883	0.00256	256	1e+05
>> BPSK_sim	6	0.0023883	0.01	1	100
>> BPSK_sim	6	0.0023883	0.01	1	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100

PROF. ANDREW THANGARAJ
Simulation of Uncoded BPSK and BER vs E_b/N₀ plot Generation in MATLAB/Octave

So now the reason why this number of times you repeat the thing is very important is supposing I run it only 100 times okay so what will happen is if you run it only 100 times this will keep giving you a number which you cannot trust okay so you see it gives you 0.01 sometimes it gives you 0 okay, the 0 or 1 error while the prediction is 0.0002 so that is why the number of errors you get in your simulation is very important.

(Refer Slide Time: 20:00)

Command	Block Length	BER	Number of Errors	Number of Trials	
>> BPSK_sim	6	0.0023883	0.01	1	100
>> BPSK_sim	6	0.0023883	0.01	1	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	6	0.0023883	0	0	100
>> BPSK_sim	10.5	1.0838e-06	0	0	1e+05

PROF. ANDREW THANGARAJ
Simulation of Uncoded BPSK and BER vs E_b/N₀ plot Generation in MATLAB/Octave

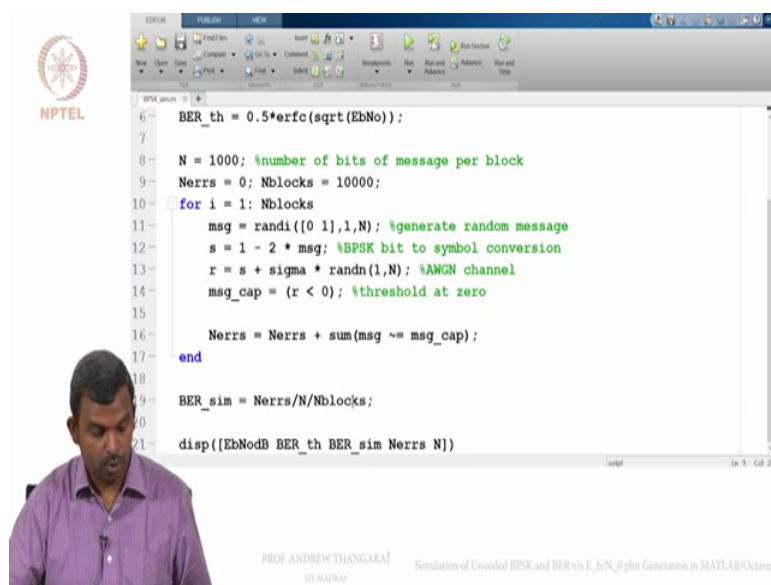
So typically when I run simulations I like to simulate for as many blocks as needed so that the number of errors is at least 100 okay so in your simulation that is always a good number to keep in mind, if you do that usually it is very good so that is why if you see, if you make it

100 thousand okay or 1 lakh as we call it here, it usually is pretty good for this Eb over N not okay for the Eb over N not of 6 okay.

Now if I change this Eb over N not to 10.5 for instance okay notice what happens and if I run this okay so 10.5 Eb over N not, I know my bit error rate is going to be around 10 power minus 6 and you can see the problem here, it goes to 0 right, 10 power minus 6 bit error rate you are simulating only 10 power 5 and you are getting 0 so what should you do, you should do something slightly smarter and for that it is usually good not to keep increasing N.

So of course you can keep increasing N to some 10 power something but that is not very nice because the reason is this is using storage of N okay so this works for small N, for large N this is not going to work very well so what you can do is you can make a loop okay you can make a loop and have a cumulative N, so that is also very nice thing to do, so maybe you want to keep your N as thousand okay, number of bits of message per block okay.

(Refer Slide Time: 21:24)



```
6 BER_th = 0.5*erfc(sqrt(EbNo));
7
8 N = 1000; %number of bits of message per block
9 Nerrs = 0; Nblocks = 10000;
10 for i = 1: Nblocks
11     msg = randi([0 1],1,N); %generate random message
12     s = 1 - 2 * msg; %BPSK bit to symbol conversion
13     r = s + sigma * randn(1,N); %AWGN channel
14     msg_cap = (r < 0); %threshold at zero
15
16     Nerrs = Nerrs + sum(msg ~= msg_cap);
17 end
18
19 BER_sim = Nerrs/Nblocks;
20
21 disp([EbNoB BER_th BER_sim Nerrs N])
```

NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave

So you repeat the block multiple times okay so how many time do you want to repeat for i equals 1 colon let us say 100 okay and then you have to end this somewhere here okay, it is still not ready to finish this so you repeat in blocks of thousand a hundred times okay so this way your memory is not very high okay your memory every time you create this vectors you are only creating thousand vectors but you are repeating it hundred times so you don't need too much memory and you can still overall simulate for a very large number.

But here it needs little bit of careful work here so what you should do is you should have this n errors which is equal to 0 in the beginning and you have to keep adding up here, is that

okay and finally when you divide so what I like to do usually is I also like to have a variable called N blocks which is hundred and then I put N blocks here okay and then I divide by N and I further divide by N blocks to get my overall BER sim.

So this way my simulation will be little slower because it is doing a loop and loops are not that fast in MATLAB but I am not using so much storage and they can increase my N blocks to as higher level as I like okay so for instance I might make my N block as 10 thousand okay so now overall I would be simulating 10^8 bits but I will be using only a memory of thousand at a time okay.

So this will not be very slow in at least in this computer, I believe this will run pretty fast and then if you look at it I am counting my total errors right, initially my number of errors is 0 and for every block I keep on adding the number of errors that I get, finally I divide by N, the number of bits in per block and number of blocks also so overall this is a valid way to do my simulation.

(Refer Slide Time: 23:26)

NPTEL

```
>> BPSK_sim
6 0.0023883 0 0 100
>> BPSK_sim
6 0.0023883 0 0 100
>> BPSK_sim
6 0.0023883 0 0 100
>> BPSK_sim
6 0.0023883 0 0 100
>> BPSK_sim
10.5 1.0838e-06 0 0 1e+05
>> BPSK_sim
10.5 1.0838e-06 2.2e-06 22 1000
>> BPSK_sim
10.5 1.0838e-06 1.2e-06 12 1e+07
fx>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave

NPTEL

```
>> BPSK_sim
6 0.0023883 0 0 100
>> BPSK_sim
6 0.0023883 0 0 100
>> BPSK_sim
6 0.0023883 0 0 100
>> BPSK_sim
10.5 1.0838e-06 0 0 1e+05
>> BPSK_sim
10.5 1.0838e-06 2.2e-06 22 1000
>> BPSK_sim
10.5 1.0838e-06 1.2e-06 12 1e+07
>> BPSK_sim
10.5 1.0838e-06 1.11e-06 111 1e+08
fx>>
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave

So let us go and run this now, so you see it took a little bit of time but it was printing it as N so maybe I should change what I am printing here to N into N blocks, was to show you how many bits are being simulated, let us go back and run it again it was pretty fast so it is not too bad these days so 1.08, still the number of errors is only 12 so maybe you want to increase the block further and was not too slow so maybe we put it one more 0 there so that way now we should get enough behavior.

So now we notice they delay, there is a little bit more of delay but still finally you get the answer comes pretty good, 10 power minus 6 is the bit error rate you expected 10.5 dB you got 111 errors out of 10 power 8 transmissions, so there was lot of transmissions were needed

to generate the errors but you did get 111 errors and you got a bit error rate simulation of 1.11×10^{-6} .

(Refer Slide Time: 24:41)

```
4 sigma = sqrt(1/(2*R*EbNo));
5
6 BER_th = 0.5*erfc(sqrt(EbNo));
7
8 N = 1000; %number of bits of message per block
9 Nerrs = 0; Nblocks = 100000;
10 for i = 1: Nblocks
11     msg = randi([0 1],1,N); %generate random message
12     %Encoding here
13     s = 1 - 2 * msg; %BPSK bit to symbol conversion
14     r = s + sigma * randn(1,N); %AWGN channel
15     %Decoding here
16     msg_cap = (r < 0); %threshold at zero
17
18     Nerrs = Nerrs + sum(msg ~= msg_cap);
19 end
```

NPTEL

PROF. ANDREW THANGARAJ
IIT MADRAS

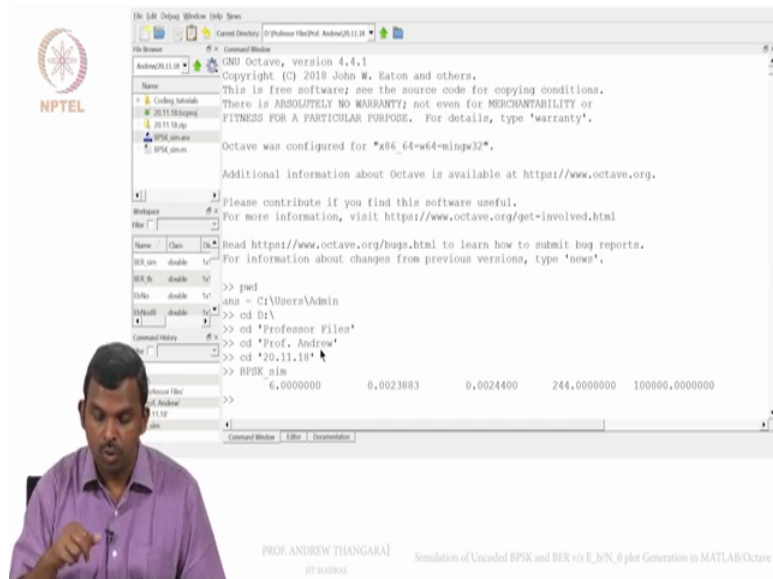
Simulation of Uncoded BPSK and BER vs E_b/N_0 plot Generation in MATLAB/Octave

So this is a basic setup in the BPSK simulation, okay so now like I mentioned we will keep this with us throughout so we will keep many elements in this things of N blocks and repeating over N blocks and rating a message, converting it into BPSK, sending it over but except before after the message is generated we will put in our coding block here okay where will the coding block come.

Encoding will come here and then before you to this threshold you would not do the threshold you will do a decoding here, okay we will introduce some code for encoding and decoding and you can simulate error control code as well and now you have to modify a lot of things right so you want to modify rate here for coding you have to modify all of that and then we can see how this works okay.

So we will do this as we go along but this is the simulation, so what I am going to try and do is show you maybe a comparison on how easily will this same program run on Octave so let us try that, it is usually pretty good but personally I use MATLAB because we do have a license of MATLAB we have not had a reason to, so I am going to open the file that we had.

(Refer Slide Time: 26:13)



The screenshot shows the Octave command window with the following text:

```
GNU Octave, version 4.4.1
Copyright (C) 2018 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

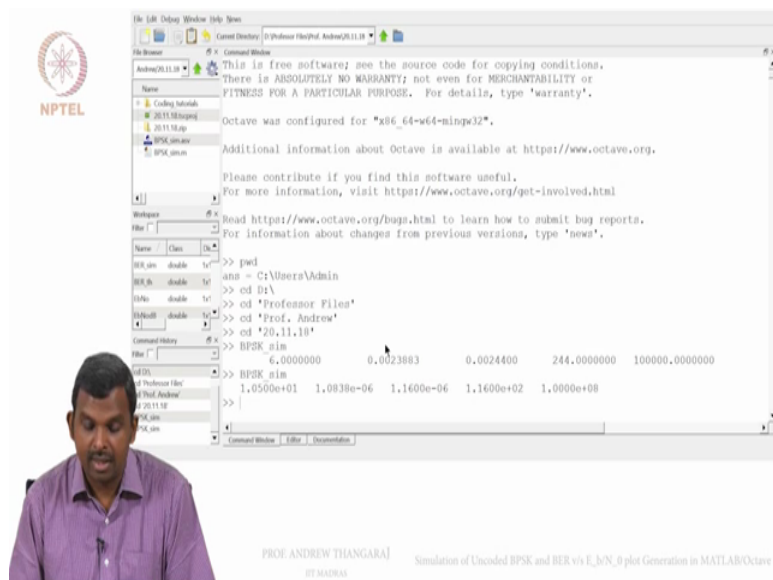
Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
```

Name	Class	Size
BPSK_sim	double	1x1
BPSK_B	double	1x1
BPSK_N	double	1x1
BPSK_S	double	1x1
BPSK_T	double	1x1
BPSK_U	double	1x1
BPSK_V	double	1x1
BPSK_W	double	1x1
BPSK_X	double	1x1
BPSK_Y	double	1x1
BPSK_Z	double	1x1
BPSK_1	double	1x1
BPSK_2	double	1x1
BPSK_3	double	1x1
BPSK_4	double	1x1
BPSK_5	double	1x1
BPSK_6	double	1x1
BPSK_7	double	1x1
BPSK_8	double	1x1
BPSK_9	double	1x1
BPSK_10	double	1x1
BPSK_11	double	1x1
BPSK_12	double	1x1
BPSK_13	double	1x1
BPSK_14	double	1x1
BPSK_15	double	1x1
BPSK_16	double	1x1
BPSK_17	double	1x1
BPSK_18	double	1x1
BPSK_19	double	1x1
BPSK_20	double	1x1
BPSK_21	double	1x1
BPSK_22	double	1x1
BPSK_23	double	1x1
BPSK_24	double	1x1
BPSK_25	double	1x1
BPSK_26	double	1x1
BPSK_27	double	1x1
BPSK_28	double	1x1
BPSK_29	double	1x1
BPSK_30	double	1x1
BPSK_31	double	1x1
BPSK_32	double	1x1
BPSK_33	double	1x1
BPSK_34	double	1x1
BPSK_35	double	1x1
BPSK_36	double	1x1
BPSK_37	double	1x1
BPSK_38	double	1x1
BPSK_39	double	1x1
BPSK_40	double	1x1
BPSK_41	double	1x1
BPSK_42	double	1x1
BPSK_43	double	1x1
BPSK_44	double	1x1
BPSK_45	double	1x1
BPSK_46	double	1x1
BPSK_47	double	1x1
BPSK_48	double	1x1
BPSK_49	double	1x1
BPSK_50	double	1x1
BPSK_51	double	1x1
BPSK_52	double	1x1
BPSK_53	double	1x1
BPSK_54	double	1x1
BPSK_55	double	1x1
BPSK_56	double	1x1
BPSK_57	double	1x1
BPSK_58	double	1x1
BPSK_59	double	1x1
BPSK_60	double	1x1
BPSK_61	double	1x1
BPSK_62	double	1x1
BPSK_63	double	1x1
BPSK_64	double	1x1
BPSK_65	double	1x1
BPSK_66	double	1x1
BPSK_67	double	1x1
BPSK_68	double	1x1
BPSK_69	double	1x1
BPSK_70	double	1x1
BPSK_71	double	1x1
BPSK_72	double	1x1
BPSK_73	double	1x1
BPSK_74	double	1x1
BPSK_75	double	1x1
BPSK_76	double	1x1
BPSK_77	double	1x1
BPSK_78	double	1x1
BPSK_79	double	1x1
BPSK_80	double	1x1
BPSK_81	double	1x1
BPSK_82	double	1x1
BPSK_83	double	1x1
BPSK_84	double	1x1
BPSK_85	double	1x1
BPSK_86	double	1x1
BPSK_87	double	1x1
BPSK_88	double	1x1
BPSK_89	double	1x1
BPSK_90	double	1x1
BPSK_91	double	1x1
BPSK_92	double	1x1
BPSK_93	double	1x1
BPSK_94	double	1x1
BPSK_95	double	1x1
BPSK_96	double	1x1
BPSK_97	double	1x1
BPSK_98	double	1x1
BPSK_99	double	1x1
BPSK_100	double	1x1

PROF. ANDREW THANGARAJ
Simulation of Uncoded BPSK and BER vs E_b/N₀ plot Generation in MATLAB/Octave



The screenshot shows the Octave command window with the following text:

```
GNU Octave, version 4.4.1
Copyright (C) 2018 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
```

Name	Class	Size
BPSK_sim	double	1x1
BPSK_B	double	1x1
BPSK_N	double	1x1
BPSK_S	double	1x1
BPSK_T	double	1x1
BPSK_U	double	1x1
BPSK_V	double	1x1
BPSK_W	double	1x1
BPSK_X	double	1x1
BPSK_Y	double	1x1
BPSK_Z	double	1x1
BPSK_1	double	1x1
BPSK_2	double	1x1
BPSK_3	double	1x1
BPSK_4	double	1x1
BPSK_5	double	1x1
BPSK_6	double	1x1
BPSK_7	double	1x1
BPSK_8	double	1x1
BPSK_9	double	1x1
BPSK_10	double	1x1
BPSK_11	double	1x1
BPSK_12	double	1x1
BPSK_13	double	1x1
BPSK_14	double	1x1
BPSK_15	double	1x1
BPSK_16	double	1x1
BPSK_17	double	1x1
BPSK_18	double	1x1
BPSK_19	double	1x1
BPSK_20	double	1x1
BPSK_21	double	1x1
BPSK_22	double	1x1
BPSK_23	double	1x1
BPSK_24	double	1x1
BPSK_25	double	1x1
BPSK_26	double	1x1
BPSK_27	double	1x1
BPSK_28	double	1x1
BPSK_29	double	1x1
BPSK_30	double	1x1
BPSK_31	double	1x1
BPSK_32	double	1x1
BPSK_33	double	1x1
BPSK_34	double	1x1
BPSK_35	double	1x1
BPSK_36	double	1x1
BPSK_37	double	1x1
BPSK_38	double	1x1
BPSK_39	double	1x1
BPSK_40	double	1x1
BPSK_41	double	1x1
BPSK_42	double	1x1
BPSK_43	double	1x1
BPSK_44	double	1x1
BPSK_45	double	1x1
BPSK_46	double	1x1
BPSK_47	double	1x1
BPSK_48	double	1x1
BPSK_49	double	1x1
BPSK_50	double	1x1
BPSK_51	double	1x1
BPSK_52	double	1x1
BPSK_53	double	1x1
BPSK_54	double	1x1
BPSK_55	double	1x1
BPSK_56	double	1x1
BPSK_57	double	1x1
BPSK_58	double	1x1
BPSK_59	double	1x1
BPSK_60	double	1x1
BPSK_61	double	1x1
BPSK_62	double	1x1
BPSK_63	double	1x1
BPSK_64	double	1x1
BPSK_65	double	1x1
BPSK_66	double	1x1
BPSK_67	double	1x1
BPSK_68	double	1x1
BPSK_69	double	1x1
BPSK_70	double	1x1
BPSK_71	double	1x1
BPSK_72	double	1x1
BPSK_73	double	1x1
BPSK_74	double	1x1
BPSK_75	double	1x1
BPSK_76	double	1x1
BPSK_77	double	1x1
BPSK_78	double	1x1
BPSK_79	double	1x1
BPSK_80	double	1x1
BPSK_81	double	1x1
BPSK_82	double	1x1
BPSK_83	double	1x1
BPSK_84	double	1x1
BPSK_85	double	1x1
BPSK_86	double	1x1
BPSK_87	double	1x1
BPSK_88	double	1x1
BPSK_89	double	1x1
BPSK_90	double	1x1
BPSK_91	double	1x1
BPSK_92	double	1x1
BPSK_93	double	1x1
BPSK_94	double	1x1
BPSK_95	double	1x1
BPSK_96	double	1x1
BPSK_97	double	1x1
BPSK_98	double	1x1
BPSK_99	double	1x1
BPSK_100	double	1x1

PROF. ANDREW THANGARAJ
Simulation of Uncoded BPSK and BER vs E_b/N₀ plot Generation in MATLAB/Octave

On Octave it seems to open, so let us not try the very ambitious version, we will try 6 dB and we will try a number of blocks is 100, let us see okay so let me save this and let me here also I should go to that okay it does not seem to like it, okay so now we are on, so let us run this BPSK sim here okay, so it ran pretty fast so you that it gave you 244.00024 so maybe we can also run 10.5 then we needed 10 power 8 right okay, so this will give you 10 power 8 so save it, go to the command window and then run again (())(27:40) of time, okay but it manage to finish us well.

You can see there were 116 errors this time and then the bit error rate, theoretical and bit error rate, the simulation one agree in Octave as well okay. So this is sort of true generally you can write in MATLAB and then the code more or less works with very little change in Octave as

well but remember I was using just basic commands, I was not using any advanced package based MATLAB commands, usually you do not have to, these kinds of commands are pretty good as well okay.

So that is the end of this lecture, hopefully you have learnt enough from this, you will be expected to at least, so this code will be made available to you, you can take it and then you will be expected to run it, produce some results and then answer some assignments based on that okay so we will expect that you can do this, so that is the end of this lecture so in the next lecture I will start talking a little bit more about error control codes, how to build encoders for them, how to build decoders for them and then we will have to write some code for doing encoding and decoding, we will do that after we learn about error control codes, thank you very much.