

LDPC and Polar codes in 5G standards
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology Madras
Fixed Point Quantisation for SC Decoder and LDPC Decoder

Hello and welcome to this lecture we are going to talk about couple of things in this lecture, the 1st thing is fixed point implementation for the decoder I think I mentioned it briefly before I want to say a few more things systematically about it and 2nd thing we will see yes rate matching for the NDPC code in the 5G standard, so how that is done is like I said precisely described in the standard for a point out simple implementation for the rate matching that I will do the, so it can be modified later on. These are 2 things we will do in this lecture, so let us get started.

(Refer Slide Time: 1:06)

NPTEL

BPSK: $0 \rightarrow +1$
 $1 \rightarrow -1$

Received value: $r = s + n$

r : real value (infinite precision in theory)
 (double precision) r_{max} in software
 hardware: 6 bits per received value
 $r_{max} = 3 \text{ (or } 4)$

Chip receiver

$$r' = \begin{cases} r, & \text{if } -r_{max} \leq r \leq r_{max} \\ -r_{max}, & \text{if } r < -r_{max} \\ r_{max}, & \text{if } r > r_{max} \end{cases}$$

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder

So if you remember the basic model that we have, we have BPSK which takes 0 to plus 1, 1 to minus 1 and then they have the received value after Additive White Gaussian Noise r is S plus some noise, right? So now this is plus 1 or minus 1 and this is Gaussian, Gaussian random available mean 0 and variance is going to be some Sigma Square that you will pick in the simulation based on the E_b over and not that you want, so this is the picture so if you want to draw it on an axis you can draw the value of r on an axis you have plus 1 here minus 1 here okay and the values are going to be around this minus 1 maybe depends on the Sigma Square, right? So you can have all sorts of values.

This is what you have okay, so generally as per this model this r is a real value, so what do I mean by real value you could get something like you know 1.002345 things like that, so the

precision can be very high something like that at least in the model the way we write Matlab code, Matlab uses double precision numbers, so they have quite a few decimal points okay so maybe 32 bits, 64 bits like that okay so how... Implementations when you want to put your decoder on a chip or on a processor, on a board which has low power typically people who do not you such high levels of quantisation.

So he cannot use 64 bit perceived value and all that is really hard. In fact the number of bits they will allow depending on the area that you can have on the chip et cetera maybe just 5 bits or 4 bits per value, so this is technically theoretically infinite precision in theory. So in practice double precision say in software we need simulation probably using 32 bits, 64 bits these days okay, so that is what is double precision. So in hardware you probably want 5 bits per receive values, so you have to quantise and I will mention briefly how I am going to do the quantising this is how it is done typically, okay.

So when you quantise usually 1st thing you do is set a maximum possible value, so you will clip the value that you considered to some regions, so for instance r equals s plus n , n is Gaussian, Gaussian can be plus infinity minus infinity can take a very large value sometimes occasionally does happen okay. So in practice what you do it is, 1st you clip into some r max, r max minus R max okay, so what I will do is r will come in and I will clip and I will get some r prime which is clip, so what will this do?

So r prime equals r if R is between plus R max and minus R max and this will be minus R max if R is less than minus R max and it will be plus R max if R is greater than R max, so this is the clipping operation, so we go above r max or below minus R max you set it as r max, so a typical value for r max could be 3 or 4 or something like that in this case for the BPSK case that seems fine enough you do not need much more than that.

So notice I am doing 3 or 4 I am assuming the receive symbol is plus 1 or minus 1 plus noise, so there will be some gain in your receiver chain to make sure that that happens correctly okay, so R max and minus R max is typically for the decoding purpose you could take it as 3 or 4 it is not wrong okay, so once you decide on an R max, the values in between have to be represented by just 5 bits okay, so now let us say we will take 6 bits this is having some convenience.

(Refer Slide Time: 5:57)

The slide content includes:

- Diagram:** A number line from $-r_{max}$ to r_{max} with tick marks. Above the line, it says "r: real value (infinite precision in theory)" and "(double precision) r_max in software". Below the line, it says "hardware: 6 bits per received value" and "r_max: 3 (or 4)".
- Clipping Function:** A box labeled "Clip noise" with input r and output r' . The function is defined as:

$$r' = \begin{cases} r, & \text{if } -r_{max} \leq r \leq r_{max} \\ -r_{max}, & \text{if } r < -r_{max} \\ r_{max}, & \text{if } r > r_{max} \end{cases}$$
- Quantization:** A box labeled "Quantized value" containing the formula:

$$\text{int} \left(\frac{r'}{r_{max}} \times \text{max_int} \right)$$
 Below the box, it says "integers from -31 to 31".
- Text:** "r': quantized to 6 bits", "1 bit: sign", "5 bits: magnitude", "max_int = 31".

At the bottom of the slide, there is a photo of Prof. Andrew Thangaraj and the text: "PROF. ANDREW THANGARAJ IIT MADRAS Fixed Point Quantization for SC Decoder and LDPC Decoder".

So if you look at 6 bits 1 bit is going to show you sign so then after this r' has to be quantised to say 6 bits okay 1 bit is sign the remaining 5 bits is magnitude okay, so that is one way to think about it so you can do a very quick and dirty operation to do this quantisation, it is pretty good in most cases, so you take the maximum integer that you want to allow it is just 31, right? So we have 5 bits you can go from minus 31 to 31 okay so those of you who know double, two's complement method of representation will tell me it is minus 32 to plus 31 but we just keep it as 31, so we will take minus 31 to plus 31 as the maximum integer and you have r_{max} and you have this r' as well and what I will do is for quantising simply r' divided by r_{max} okay and then I will multiply this with max_int and then take the integer part of it okay.

So this is very convenient and simple way to do quantisation, so this is the quantise value okay. So hopefully this is clear to you okay so you take r' the clipped value divided by r_{max} multiplied by max_int and take the integer value could take floor or ceil or something like that it does not matter, so you can take the integer value okay. So this gives you integers, so what will happen once you do this is in the receiver you will have integer value, so we can see automatically one bit becomes sign 5 bit magnitude when you do this if r' is negative this integers is going to be negative, so it will be minus 5 or plus 5 or things like that so I will show you when we write this in Matlab how the values look but this is how it works.

So you have your received value you clip it then you quantise you get integer values, so these are integers from minus 31 to plus 31 okay and you worked with this integer in your decoder okay. So you may be doing operations with this integer you may add them, if you add you

have to make sure that you do not exceed too larger number, so all that you have to take care of, so I will stop with this as far as this course is concerned.

So we will do quantisation and we will work with quantised value, so now quantisation has some effect on the decoders these decoders we implement a suboptimal, so you have to be slightly careful when you quantise and we watch out for things that happen because of quantisation okay. So one of the effects is quantisation makes receive values equal, so for instance you may never get 2 receive values which are the same if you do not quantise but if you quantise you will get the same values.

So because of that there can be some sub optimality of course there will be sub optimality and you have to take care of those things in the decoder. Some things to watch out for but generally this will work, what works with real values will typically works with these kind of integer values as well, okay. So this is how the quantisation is done. So let me show you in Matlab how I have done these modifications to the decoder to put in this and work with integers okay.

(Refer Slide Time: 9:28)

The screenshot shows a MATLAB script with the following key sections:

- Signal Generation:** A signal x is generated using $x = 1 - 2 * \text{cword}(\text{NBSK bit to symbol conversion})$.
- Channel Simulation:** The signal x is passed through a channel: $r = x + \text{sigma} * \text{randn}(1, N); \text{NBSK channel}$.
- Quantization:** The received signal r is quantized: $r_q = \text{floor}(\text{format}(\text{maxqr}/2))$.
- Decoding:** The quantized signal r_q is decoded using a successive cancellation decoder. The code includes:
 - Initialization: $l = \text{zeros}(n, 1, N); \text{N beliefs}$, $uosp = \text{zeros}(n, 1, N); \text{N decisions}$, $ns = \text{zeros}(1, 2^M - 1); \text{node state vector}$.
 - Function Definition: $\text{satx} = \text{R}(x) \text{ min}(\text{max}(x, -\text{maxqr}), \text{maxqr}); \text{N saturate FP value}$.
 - Decoding Loop: $\text{while done} == 0$ (iterate until all bits are decoded). Inside the loop, $\text{if depth} == n$ is checked.

So the Matlab code for the successive cancellation decoder. For the successive cancellation decoder for polar codes, so previously we did not have the integer the fix point version of this code so I have saved it as, in our polar SED code FP and then I have put in the integer think, so you can see what I have done, I have done the R max as 4, you will have the code with you, you can try with 3 et cetera if you think that works okay and I have put Maxqr as 31 okay so this is the maximum integer received value.

The other things do not change nothing much changes everything else is same you decode and all that and now when you transmit you will do the BPSK just like before you will do the receive value just like before okay in this simulation this is how it works and then I will do the quantisation, you can do the quantisation in multiple ways I have done it in a slightly different way here you can see I have divided r by R_{max} and then multiplied by max_{QR} and then it takes the floor of it to get RQ and then...so I am not clipping 1st so I am doing the clipping a bit differently here, so you can do r by r_{max} into max_{QR} and then you do the clipping okay so this is the.

The clippings come here okay, so this is also fine so you do r by r_{max} into max_{QR} and then you clip and here can say I have clipped up to max_{QR} on the positive side, on the negative side I have done $minus_{max_{QR}} + 1$ okay so I am doing $minus_{32}$ to $plus_{31}$ which is actual two's complement representation of integers okay, so hopefully that is clear so you will see that this is equivalent. You can also do the other way you can clip r to 1^{st} minus R_{max} to R_{max} and then this will not be needed you can just to the floor that is also fine.

(Refer Slide Time: 11:30)

The screenshot shows a MATLAB script with the following key sections:

- Channel Simulation:** Line 72: `s = 1 - 2 * cword; %BPSK bit to symbol conversion`; Line 73: `r = s + sigma * randn(1,N); %AWGN channel I`; Line 74: `%quantization`; Line 75: `rq = floor(r*(maxQR)/rmax);`; Line 76: `rq(rq>maxqr) = maxqr; %clipping`; Line 77: `rq(rq<-maxqr+1) = -maxqr+1;`
- LDPC Decoder:** Line 78: `%LDPC decoder`; Line 79: `l = zeros(1,N); %beliefs`; Line 80: `ucap = zeros(1,N); %decisions`; Line 81: `no = zeros(1,2*M-1); %node state vector`
- Quantization:** Line 84: `zatr = @(x) min(max(x,-(maxqr+1)),maxqr); %saturate FF value`; Line 85: `g = @(a,b,c) (1-2*c)*(zatr(a)-zatr(b)); %minimum`; Line 86: `g = @(a,b,c) zatr(b+(1-2*c)*a); %g function`
- Decoding Loop:** Line 88: `l(i,1) = rq; %belief of root`; Line 90: `node = 0; %depth = 0; %start at root`; Line 91: `done = 0; %decoder has finished or not`; Line 92: `while (done == 0) %traverse till all bits are decoded`; Line 93: `%leaf or not`; Line 94: `if depth == n`; Line 95: `if any(l==node+1) %is node frozen`; Line 96: `ucap(i+1,node+1) = 0;`; Line 97: `else`; Line 98: `if l(i+1,node+1) == 0`; Line 99: `ucap(i+1,node+1) = 0;`; Line 100: `else`

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder



```
70- cword = uz
71-
72- s = 1 - 2 * cword; 99998 bit to symbol conversion
73- r = s + sigma * randn(1,N); 16QAM channel I
74- quantization
75- rq = floor(r/max*maxq);
76- rj(rq*maxq) = maxq; clipping
77- rj(rq<-maxq+1) = -maxq+1;
78-
79- VSC decoder
80- l = zeros(1,N); 4beliefs
81- ucup = zeros(1,N); 4decisions
82- ns = zeros(1,2*M-1); node state vector
83-
84- satx = @(x) min(max(x,-maxq+1),maxq); saturate FF value
85- f = @(a,b) (1-2*(a<0)).*(1-2*(b<0)).*min(abs(a),abs(b)); minimum
86- g = @(a,b,c) satx(b+(1-2*c).*a); by function
87-
88- l(i,1) = rj; belief of root
89-
90- node = 0; depth = 0; %start at root
91- done = 0; %decoder has finished or not
92- while (done == 0) %traverse till all bits are decoded
93- %leaf or not
94- if depth == n
95- if any(f==(node+1)) %is node frozen
96- ucup(i+1,node+1) = 0;
97- else
98- if l(i+1,node+1) >= 0
99- ucup(i+1,node+1) = 0;
100- else
```



PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder



```
70- cword = uz
71-
72- s = 1 - 2 * cword; 99998 bit to symbol conversion
73- r = s + sigma * randn(1,N); 16QAM channel I
74- quantization
75- rq = floor(r/max*maxq);
76- rj(rq*maxq) = maxq; clipping
77- rj(rq<-maxq+1) = -maxq+1;
78-
79- VSC decoder
80- l = zeros(1,N); 4beliefs
81- ucup = zeros(1,N); 4decisions
82- ns = zeros(1,2*M-1); node state vector
83-
84- satx = @(x) min(max(x,-maxq+1),maxq); saturate FF value
85- f = @(a,b) (1-2*(a<0)).*(1-2*(b<0)).*min(abs(a),abs(b)); minimum
86- g = @(a,b,c) satx(b+(1-2*c).*a); by function
87-
88- l(i,1) = rj; belief of root
89-
90- node = 0; depth = 0; %start at root
91- done = 0; %decoder has finished or not
92- while (done == 0) %traverse till all bits are decoded
93- %leaf or not
94- if depth == n
95- if any(f==(node+1)) %is node frozen
96- ucup(i+1,node+1) = 0;
97- else
98- if l(i+1,node+1) >= 0
99- ucup(i+1,node+1) = 0;
100- else
```



PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder



```
70- cword = uz
71-
72- s = 1 - 2 * cword; 99998 bit to symbol conversion
73- r = s + sigma * randn(1,N); 16QAM channel I
74- quantization
75- rq = floor(r/max*maxq);
76- rj(rq*maxq) = maxq; clipping
77- rj(rq<-maxq+1) = -maxq+1;
78-
79- VSC decoder
80- l = zeros(1,N); 4beliefs
81- ucup = zeros(1,N); 4decisions
82- ns = zeros(1,2*M-1); node state vector
83-
84- satx = @(x) min(max(x,-maxq+1),maxq); saturate FF value
85- f = @(a,b) (1-2*(a<0)).*(1-2*(b<0)).*min(abs(a),abs(b)); minimum
86- g = @(a,b,c) satx(b+(1-2*c).*a); by function
87-
88- l(i,1) = rj; belief of root
89-
90- node = 0; depth = 0; %start at root
91- done = 0; %decoder has finished or not
92- while (done == 0) %traverse till all bits are decoded
93- %leaf or not
94- if depth == n
95- if any(f==(node+1)) %is node frozen
96- ucup(i+1,node+1) = 0;
97- else
98- if l(i+1,node+1) >= 0
99- ucup(i+1,node+1) = 0;
100- else
```



PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder

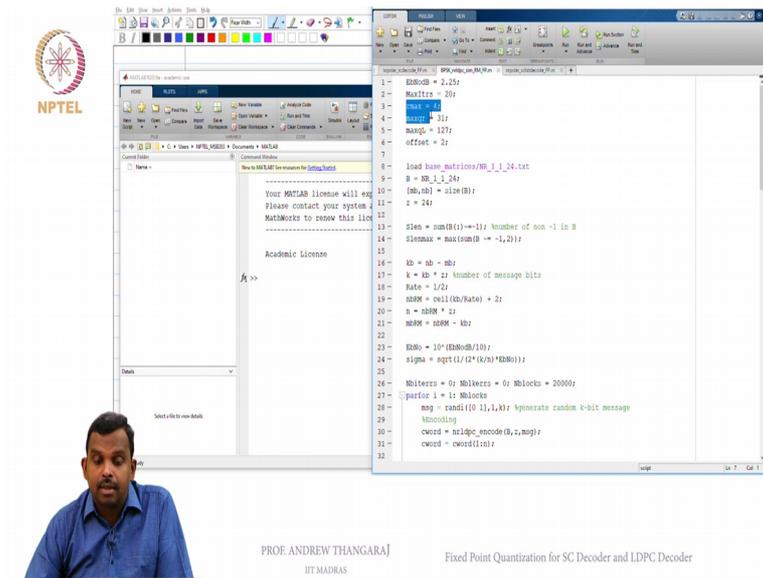
So after this nothing much changes, couple of things to watch out for, so if you look at f and g operation so if you remember the polar decoder and we only do f and g between do not do anything else in terms of operations on the received values. The f operation is okay because it does not do anything to increase the value of what is coming in, so it just takes minimum of the 2 values, so it does not need any special handling.

On the other hand g needs some saturation, the reason is so what g does is adds to values, so if you take for instance plus 31 and plus 31 you add you get plus 62 okay but you are allowed only to go between minus 31 to 31 you want to quantise it back, so that is why I have written the saturation, it saturates the FP value and you can see that we have written it I have done $\max(x, -31)$ okay so this will take care of the minus part, then after that it take the min of that max value, $\min(\max(x, -31), 31)$ okay, so this will ensure that whatever value x is it does the clipping, so the saturation does the clipping between minus 31 and plus 31 okay it will not allow the value to go above that. If it goes above that on either side it will clip it to either plus 31 on the positive side minus 31 on the negative side, so this is what I have done here. Previously we did not have the satx on the g function so now we put the satx also.

So this few changes are enough to make sure your code will work with the integer side nothing more is needed here everything else is exactly the same and you can work okay. So this is hopefully clear so you have to make sure the receive values becomes an integer, the saturation effect is something you have to watch out for and if you are doing any operations in your decoder which increases the value you have to saturate again okay.

So sometimes when you saturate like this you have to be paying some attention for instance in the LDPC code there are 2 types of values one is the LLR the messages, the other is the LLR for the total LLR these 2 are saturated at different levels. They have to be saturated at different levels otherwise your decoder will fail okay so I will comment that next but this is something important. When we do operations and when you are expecting the value to grow you have to saturate okay.

(Refer Slide Time: 13:45)

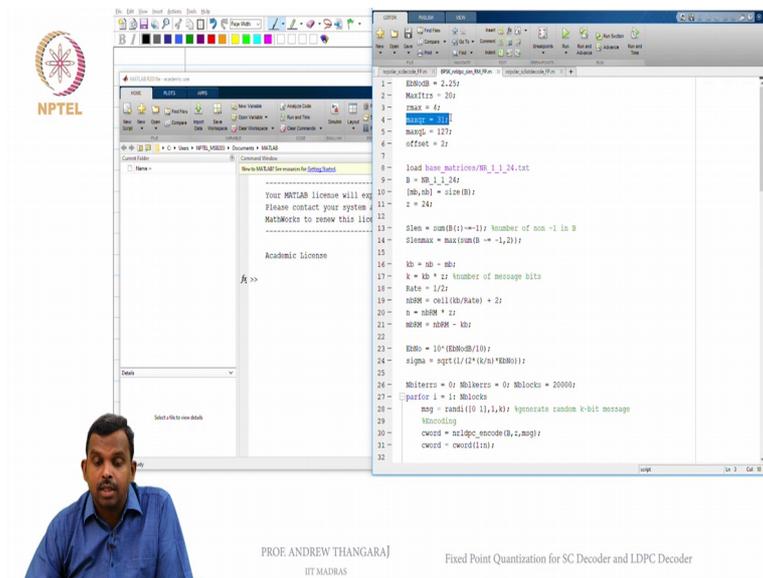


NPTEL

```
1- EbN0 = 2.25;
2- Nk = 20;
3- Nkmax = 31;
4- Nkmax = 31;
5- Nkmax = 127;
6- offset = 2;
7-
8- load base_matrices/NR_1_24.txt
9- B = NR_1_24;
10- [mb, nb] = size(B);
11- z = 24;
12-
13- Slen = sum(B(:) == -1); Number of non-1 in B
14- Slenmax = max(Slen == -1, 2);
15-
16- kb = nb - mb;
17- k = kb + z; Number of message bits
18- Rate = 1/2;
19- nBDM = ceil(kb/Rate) + 2;
20- n = nBDM * z;
21- nBDM = nBDM - kb;
22-
23- EbN0 = 10*(EbN0/10);
24- sigma = sqrt(1/(2*(k/N)*EbN0));
25-
26- Nkitters = 0; Nkitters = 0; Nblocks = 20000;
27- for i = 1: Nblocks
28-     msg = randi(0, 1, k); %generate random k-bit message
29-     %encoding
30-     cword = nldpc_encode(B, z, msg);
31-     cword = cword(i:n);
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder



NPTEL

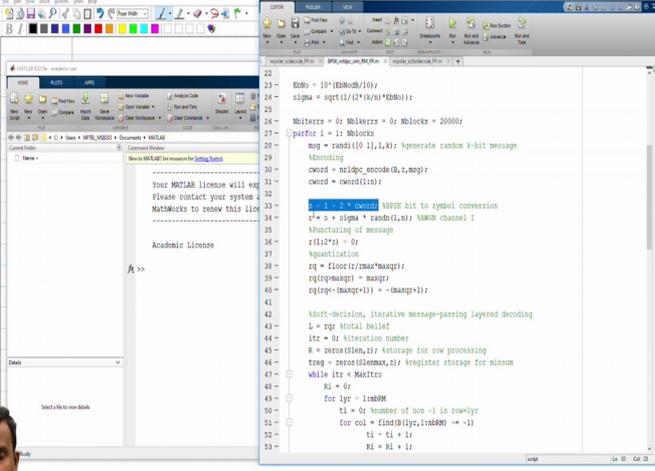
```
1- EbN0 = 2.25;
2- Nk = 20;
3- Nkmax = 4;
4- Nkmax = 31;
5- Nkmax = 127;
6- offset = 2;
7-
8- load base_matrices/NR_1_24.txt
9- B = NR_1_24;
10- [mb, nb] = size(B);
11- z = 24;
12-
13- Slen = sum(B(:) == -1); Number of non-1 in B
14- Slenmax = max(Slen == -1, 2);
15-
16- kb = nb - mb;
17- k = kb + z; Number of message bits
18- Rate = 1/2;
19- nBDM = ceil(kb/Rate) + 2;
20- n = nBDM * z;
21- nBDM = nBDM - kb;
22-
23- EbN0 = 10*(EbN0/10);
24- sigma = sqrt(1/(2*(k/N)*EbN0));
25-
26- Nkitters = 0; Nkitters = 0; Nblocks = 20000;
27- for i = 1: Nblocks
28-     msg = randi(0, 1, k); %generate random k-bit message
29-     %encoding
30-     cword = nldpc_encode(B, z, msg);
31-     cword = cword(i:n);
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder

So now let me move on to the LDPC code and show you once again how this was done on the LDPC code. Once again you see here I have max... r max is 4 and then I have max QR as 31, so this is the quantisation for the receive values. I also have a max Q for L okay. In this L value I am saying is 127 I am allowing it to go 2 bits higher in magnitude than r okay so this is important because that L represents the total LLR which can be larger and then you have to subtract it from r to compute the extrinsic and all that, so you have to allow it to be large okay so that is very important.

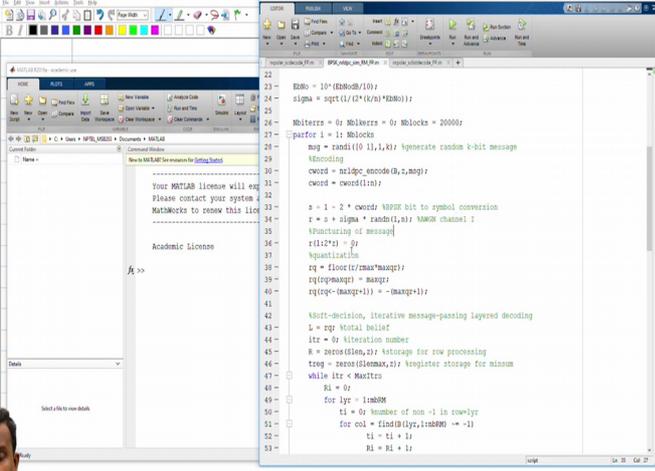
(Refer Slide Time: 14:22)



```
22 E280 = 10*(E280&8/10);
23 sigma = sqrt(1/(2*(R/A)*E280));
24
25
26 Mbitserr = 0; Mblkerrs = 0; Mbllocks = 20000;
27
28 for i = 1: Mbllocks
29     msg = randi(0 1,1,k); %generate random k-bit message
30     %Encoding
31     cword = nldpc_encode(B,t,msg);
32     cword = cword(:);
33     s = 1 - 2 * cword; %BPSK bit to symbol conversion
34     r = s + sigma * randn(1,n); %AWGN channel ;
35     %Puncturing of message
36     r(1:2*r) = 0;
37     %quantization
38     rq = floor(r/max*maxqr);
39     rj(rq>maxqr) = maxqr;
40     rj(rq<-maxqr+1) = -maxqr+1;
41
42 %Soft-decision, iterative message-passing layered decoding
43 L = exp(Mtotal belief);
44 itr = 0; % iteration number
45 R = zeros(1:nmax,2); %storage for row processing
46 treq = zeros(1:nmax,2); %register storage for minsum
47 while itr < Mmaxtrs
48     Rl = 0;
49     for lyr = 1:nBM
50         %l = 0; number of non-1 in rowlyr
51         for col = find@1(lyr,1:nBM) == -1)
52             % ti = ti + 1;
53             Rl = Rl + 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

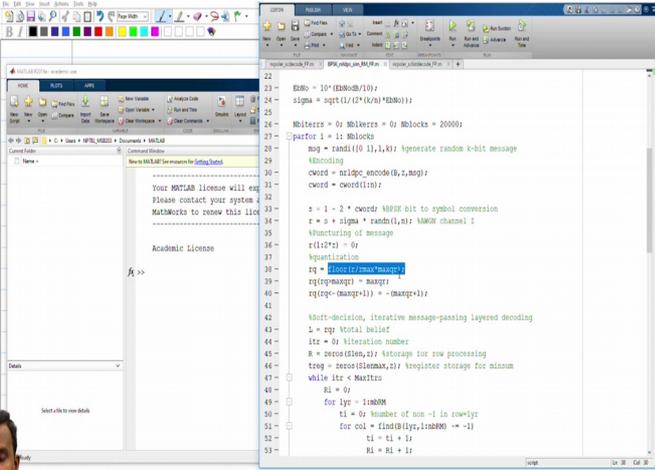
Fixed Point Quantization for SC Decoder and LDPC Decoder



```
22 E280 = 10*(E280&8/10);
23 sigma = sqrt(1/(2*(R/A)*E280));
24
25
26 Mbitserr = 0; Mblkerrs = 0; Mbllocks = 20000;
27
28 for i = 1: Mbllocks
29     msg = randi(0 1,1,k); %generate random k-bit message
30     %Encoding
31     cword = nldpc_encode(B,t,msg);
32     cword = cword(:);
33     s = 1 - 2 * cword; %BPSK bit to symbol conversion
34     r = s + sigma * randn(1,n); %AWGN channel ;
35     %Puncturing of message
36     r(1:2*r) = 0;
37     %quantization
38     rq = floor(r/max*maxqr);
39     rj(rq>maxqr) = maxqr;
40     rj(rq<-maxqr+1) = -maxqr+1;
41
42 %Soft-decision, iterative message-passing layered decoding
43 L = exp(Mtotal belief);
44 itr = 0; % iteration number
45 R = zeros(1:nmax,2); %storage for row processing
46 treq = zeros(1:nmax,2); %register storage for minsum
47 while itr < Mmaxtrs
48     Rl = 0;
49     for lyr = 1:nBM
50         %l = 0; number of non-1 in rowlyr
51         for col = find@1(lyr,1:nBM) == -1)
52             % ti = ti + 1;
53             Rl = Rl + 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

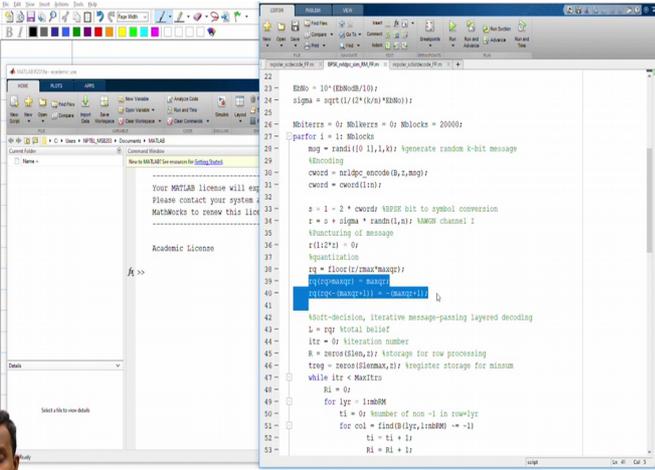
Fixed Point Quantization for SC Decoder and LDPC Decoder



```
22 E280 = 10*(E280&8/10);
23 sigma = sqrt(1/(2*(R/A)*E280));
24
25
26 Mbitserr = 0; Mblkerrs = 0; Mbllocks = 20000;
27
28 for i = 1: Mbllocks
29     msg = randi(0 1,1,k); %generate random k-bit message
30     %Encoding
31     cword = nldpc_encode(B,t,msg);
32     cword = cword(:);
33     s = 1 - 2 * cword; %BPSK bit to symbol conversion
34     r = s + sigma * randn(1,n); %AWGN channel ;
35     %Puncturing of message
36     r(1:2*r) = 0;
37     %quantization
38     rq = floor(r/max*maxqr);
39     rj(rq>maxqr) = maxqr;
40     rj(rq<-maxqr+1) = -maxqr+1;
41
42 %Soft-decision, iterative message-passing layered decoding
43 L = exp(Mtotal belief);
44 itr = 0; % iteration number
45 R = zeros(1:nmax,2); %storage for row processing
46 treq = zeros(1:nmax,2); %register storage for minsum
47 while itr < Mmaxtrs
48     Rl = 0;
49     for lyr = 1:nBM
50         %l = 0; number of non-1 in rowlyr
51         for col = find@1(lyr,1:nBM) == -1)
52             % ti = ti + 1;
53             Rl = Rl + 1;
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder

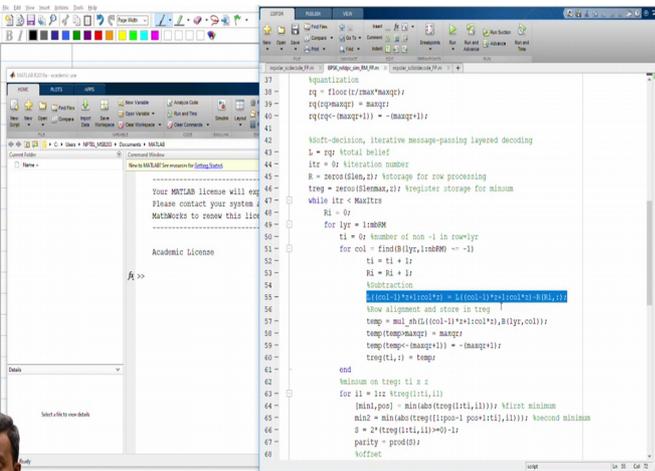



`22 E2B0 = 10*(E2B0B/10);
23 sigma = sqrt(1/2*(R/A)*E2B0B);
24
25 MbitsPerS = 0; MBlocks = 0; MLocks = 20000;
26
27 for i = 1: MBlocks
28 msg = randi(0,1,1,K); %generate random k-bit message
29 %encoding
30 cword = nldpc_encode(0,2,msg);
31 cword = cword(1:n);
32
33 s = 1 - 2 * cword; %BPSK bit to symbol conversion
34 r = s * sigma * randn(1,n); %AWGN channel
35 %Puncturing of message
36 r(1:2*n) = 0;
37 %quantization
38 rq = floor(r/cmax*maxqr);
39 rj(rq>maxqr) = maxqr;
40 rj(rq<-maxqr) = -maxqr;
41
42 %Soft-decision, iterative message-passing layered decoding
43 L = exp(Modal belief);
44 itr = 0; iteration number
45 R = zeros(1:n,2); %storage for row processing
46 treg = zeros(1:n,2); %register storage for minimum
47 while itr < MaxIters
48 R1 = 0;
49 for ijr = 1:nBM
50 ti = 0; %number of non-1 in rowijr
51 for col = find(0:1jy,1:nBM) == -1)
52 ti = ti + 1;
53 R1 = R1 + ti;`

PROF. ANDREW THANGARAJ
 IIT MADRAS
 Fixed Point Quantization for SC Decoder and LDPC Decoder

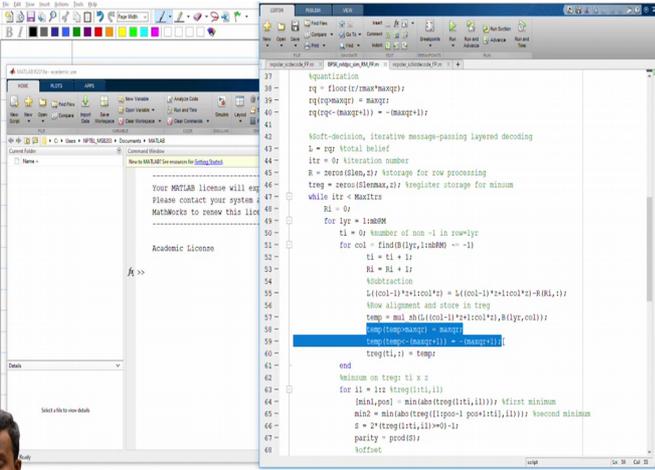
And then other than that what I do is largely the same I will compute the BPSK modulation then find the receive values of K and then of course this is puncturing and then after that I am doing exactly the same thing okay. So I do the computation of the floor, right? Convert it into integer and then you do the clipping okay so this clips okay so that is fine I have done the clipping okay.

(Refer Slide Time: 14:48)

`37 %quantization
38 rq = floor(r/cmax*maxqr);
39 rj(rq>maxqr) = maxqr;
40 rj(rq<-maxqr) = -maxqr;
41
42 %Soft-decision, iterative message-passing layered decoding
43 L = exp(Modal belief);
44 itr = 0; iteration number
45 R = zeros(1:n,2); %storage for row processing
46 treg = zeros(1:n,2); %register storage for minimum
47 while itr < MaxIters
48 R1 = 0;
49 for ijr = 1:nBM
50 ti = 0; %number of non-1 in rowijr
51 for col = find(0:1jy,1:nBM) == -1)
52 ti = ti + 1;
53 R1 = R1 + ti;
54 % (col-1)*2+1:col*2 = 1:(col-1)*2+1:col*2-0:(0:1)
55 % show alignment and store in treg
56 temp = mul_gh1((col-1)*2+1:col*2,0:1jy,col);
57 temp(temp>maxqr) = maxqr;
58 temp(temp<-maxqr) = -maxqr;
59 treg(ti,:) = temp;
60 end
61 %minimum on temp; ti x 2
62 for il = 1:2;treg(1:ti,il)
63 [min1,pos] = min(abs(treg(1:ti,il))); %first minimum
64 min2 = min(abs(treg(1:2*pos-1:2*pos+1:ti,il))); %second minimum
65 s = 2*(treg(1:ti,il)>0)-1;
66 parity = prod(s);
67 %offset
68`

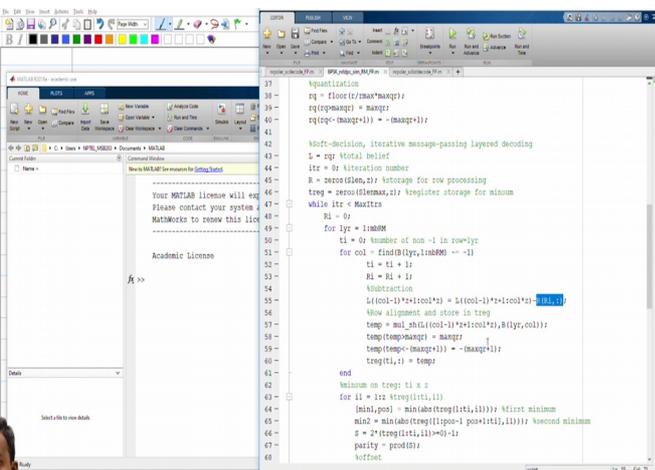
PROF. ANDREW THANGARAJ
 IIT MADRAS
 Fixed Point Quantization for SC Decoder and LDPC Decoder

`37 %quantization
38 rq = floor(r/max*maxqr);
39 rj(rq*maxqr) = maxqr;
40 rj(rq<-maxqr+1) = -maxqr+1;
41
42 %soft-decision, iterative message-passing layered decoding
43 L = exp(Mtotal*belief);
44 itr = 0; iteration number
45 R = zeros(Slen,2); %storage for row processing
46 treq = zeros(Slen,2); %register storage for minsum
47 while itr < MaxIters
48 for lyr = 1:nbDM
49 ti = 0; number of non -1 in row*lyr
50 for col = find(0|lyr,1:nbDM) == -1)
51 ti = ti + 1;
52 Rj = Rj + 1;
53 %subtraction
54 L((col-1)*+1:icol*2) = L((col-1)*+1:icol*2)-Rj(1,:);
55 %row alignment and store in treq
56 temp = mul_shd(L((col-1)*+1:icol*2),Rj(lyr,col));
57 temp(temp>maxqr) = maxqr;
58 temp(temp<-maxqr+1) = -maxqr+1;
59 treq(ti,1) = temp;
60 end
61 %minsum on treq; ti x z
62 for il = 1:itr %treq(ti,il)
63 [min1,pos] = min(abs(treq(ti,1:il))); %first minimum
64 min2 = min(abs(treq(ti,pos+1:il))); %second minimum
65 S = 2*(treq(ti,il)>=0)-1;
66 parity = prod(S);
67 %offset
68`

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder

`37 %quantization
38 rq = floor(r/max*maxqr);
39 rj(rq*maxqr) = maxqr;
40 rj(rq<-maxqr+1) = -maxqr+1;
41
42 %soft-decision, iterative message-passing layered decoding
43 L = exp(Mtotal*belief);
44 itr = 0; iteration number
45 R = zeros(Slen,2); %storage for row processing
46 treq = zeros(Slen,2); %register storage for minsum
47 while itr < MaxIters
48 for lyr = 1:nbDM
49 ti = 0; number of non -1 in row*lyr
50 for col = find(0|lyr,1:nbDM) == -1)
51 ti = ti + 1;
52 Rj = Rj + 1;
53 %subtraction
54 L((col-1)*+1:icol*2) = L((col-1)*+1:icol*2)-Rj(1,:);
55 %row alignment and store in treq
56 temp = mul_shd(L((col-1)*+1:icol*2),Rj(lyr,col));
57 temp(temp>maxqr) = maxqr;
58 temp(temp<-maxqr+1) = -maxqr+1;
59 treq(ti,1) = temp;
60 end
61 %minsum on treq; ti x z
62 for il = 1:itr %treq(ti,il)
63 [min1,pos] = min(abs(treq(ti,1:il))); %first minimum
64 min2 = min(abs(treq(ti,pos+1:il))); %second minimum
65 S = 2*(treq(ti,il)>=0)-1;
66 parity = prod(S);
67 %offset
68`

PROF. ANDREW THANGARAJ
IIT MADRAS

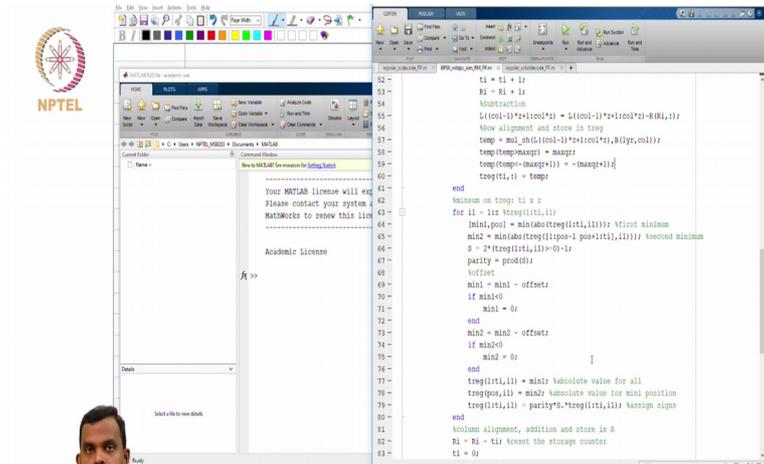
Fixed Point Quantization for SC Decoder and LDPC Decoder

So now in the decoder so this is inside the LDPC decoder you remember the LDPC decoder you do row operations, right? In the layer decoding row after row after row you work with it and then you subtract, you first subtract the total LLR minus what is already in the row then you do the min some on the row and then you add that value back to L total okay, so here is the subtraction, subtraction happens here the capital R is what is in the storage for the row processing okay.

So you do the subtraction and then you to some row alignment and after that notice here I do this okay so anytime you do an operation like this you have to also make sure that you quantise again okay so because this L has going from minus 127 to 127 this is only minus 31 to 31, so this can go above 31 so you quantise again make sure that it is quantise. So this

make sure that your bit with the... Number of bits you are allocating for the storage remains 5 okay so that is very important 5 or 6 in this case okay.

(Refer Slide Time: 15:48)



The screenshot shows a MATLAB script editor with the following code:

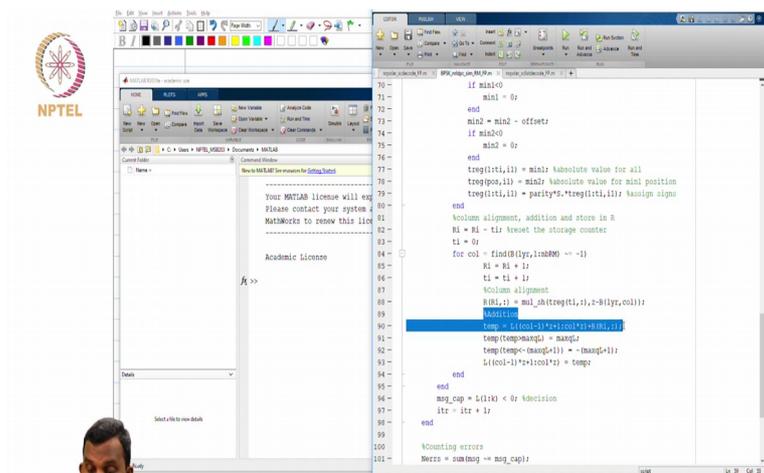
```

52 ti = ti + 1;
53 Ri = Ri + 1;
54 %Subtraction
55 L((col-1)*2+1:col*2) = L((col-1)*2+1:col*2)-R(Ri,i);
56 %Row alignment and store in temp
57 temp = mul_sh(L((col-1)*2+1:col*2),R(i),col);
58 temp(temp==magr) = magr;
59 temp(temp==(-magr+1)) = -(magr+1);
60 temp(ti,i) = temp;
61 end
62 %Column alignment
63 for ii = 1:2 %temp(ti,i,ii)
64 %min, pos) = min(abs(temp(ti,i,ii))); %first minimum
65 min2 = min(abs(temp((i+pos-1):pos+1,i,ii))); %second minimum
66 S = 2*(temp(ti,i,ii))-0-1;
67 parity = mod(S);
68 %offset
69 min1 = min1 - offset;
70 if min1 < 0;
71 min1 = 0;
72 end
73 min2 = min2 - offset;
74 if min2 < 0;
75 min2 = 0;
76 end
77 temp(ti,ii) = min1; %absolute value for all
78 temp(pos,ii) = min2; %absolute value for min1 position
79 temp(ti,ii) = parity*S.*temp(ti,ii); %assign signs
80 end
81 %Column alignment, addition and store in R
82 Ri = Ri + 1; %reset the storage counter
83 ti = 0;

```

NPTEL logo is visible in the top left corner of the MATLAB window. Below the MATLAB window, there is a small video feed of Prof. Andrew Thangaraj and his name and affiliation: PROF. ANDREW THANGARAJ, IIT MADRAS.

Fixed Point Quantization for SC Decoder and LDPC Decoder



The screenshot shows the continuation of the MATLAB script editor with the following code:

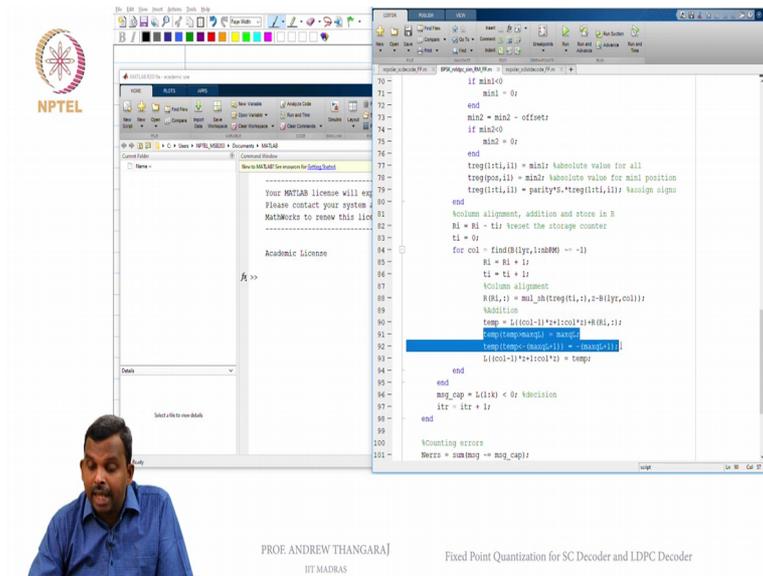
```

70 if min1 < 0;
71 min1 = 0;
72 end
73 min2 = min2 - offset;
74 if min2 < 0;
75 min2 = 0;
76 end
77 temp(ti,ii) = min1; %absolute value for all
78 temp(pos,ii) = min2; %absolute value for min1 position
79 temp(ti,ii) = parity*S.*temp(ti,ii); %assign signs
80 end
81 %Column alignment, addition and store in R
82 Ri = Ri + 1; %reset the storage counter
83 ti = 0;
84 for col = find(R(i),1:8000) == -1)
85 Ri = Ri + 1;
86 ti = ti + 1;
87 %Column alignment
88 R(Ri,i) = mul_sh(temp(ti,i),2-8(i),col);
89 %Subtraction
90 temp = L((col-1)*2+1:col*2)+R(Ri,i);
91 temp(temp==magr) = magr;
92 temp(temp==(-magr+1)) = -(magr+1);
93 L((col-1)*2+1:col*2) = temp;
94 end
95 end
96 msg_cap = L(1:k) < 0; %decision
97 itr = itr + 1;
98 end
99
100 %Counting errors
101 %errr = sum(msg == msg_cap);

```

NPTEL logo is visible in the top left corner of the MATLAB window. Below the MATLAB window, there is a small video feed of Prof. Andrew Thangaraj and his name and affiliation: PROF. ANDREW THANGARAJ, IIT MADRAS.

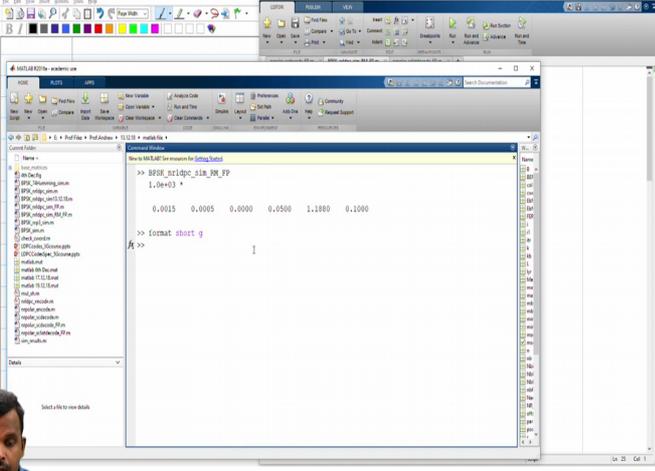
Fixed Point Quantization for SC Decoder and LDPC Decoder



And then you do the min some operation here, the min some does not require any checks for increase in value because you are not doing any addition, so that is okay and then you have addition here again okay. So notice what happens here, so once you have the min some value you have to add, so this is the addition part and when I add this temp ultimately becomes L, okay so to quantise again but the quantisation is to Max QL now okay so I am going from minus 128 to plus 127 okay, so I do that here to make sure I do not overflow here and that is fine, so these of only changes other than that there is nothing else to change everything else work as such okay.

So converting your code to fix point is little bit easy to start with but when you look at the operation inside making sure that the bit with are valid for the different types of number you have inside your decoder can be a little troubling. In the polar codes it is very easy and simple you only have to saturate the g operation. In the LDPC code you have 2 different values one is this r storage that you use for the row processing, the other is the L value which is the total LLR total belief and as you keep processing the L value has a larger quantisation than the are value, so you have to take care of that and keep saturating. Whenever you do addition or subtraction you have to keep saturating in a different way, so that is something important once you do that both of these work if you want I can run these for you.

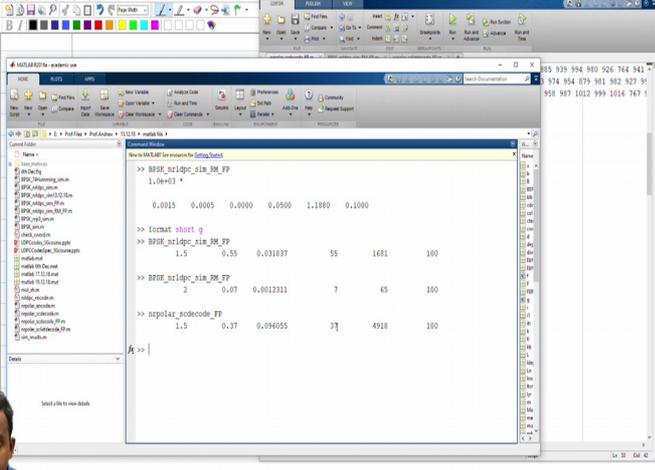
(Refer Slide Time: 17:15)



```
%% SC and LDPC Decoders for Fixed Point
>> BPSK_mldpc_mim_FP
1.0e+03 *
0.0015 0.0005 0.0000 0.0500 1.1880 0.1000
>> format short g
>>
1
```

PROF. ANDREW THANGARAJ
IIT MADRAS

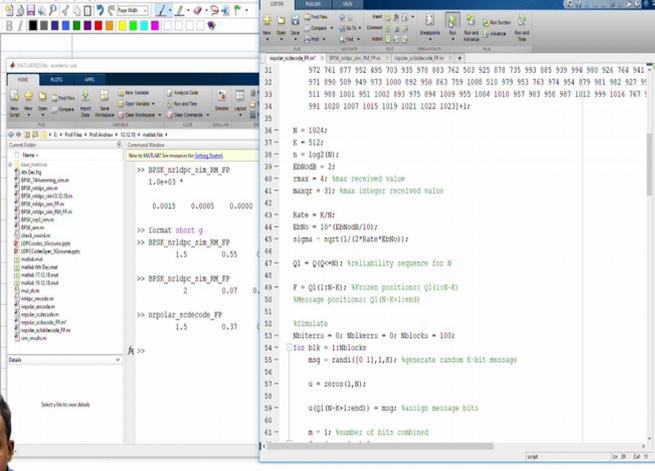
Fixed Point Quantization for SC Decoder and LDPC Decoder



```
%% SC and LDPC Decoders for Fixed Point
>> BPSK_mldpc_mim_FP
1.0e+03 *
0.0015 0.0005 0.0000 0.0500 1.1880 0.1000
>> format short g
>> BPSK_mldpc_mim_FP
1.5 0.55 0.031837 55 1681 100
>> BPSK_mldpc_mim_FP
2 0.07 0.0012311 7 65 100
>> nnpolar_decode_FP
1.5 0.37 0.096055 37 4918 100
>>
952 939 994 980 926 764 941
974 954 879 961 982 927 95
958 987 1012 999 1016 767 1
```

PROF. ANDREW THANGARAJ
IIT MADRAS

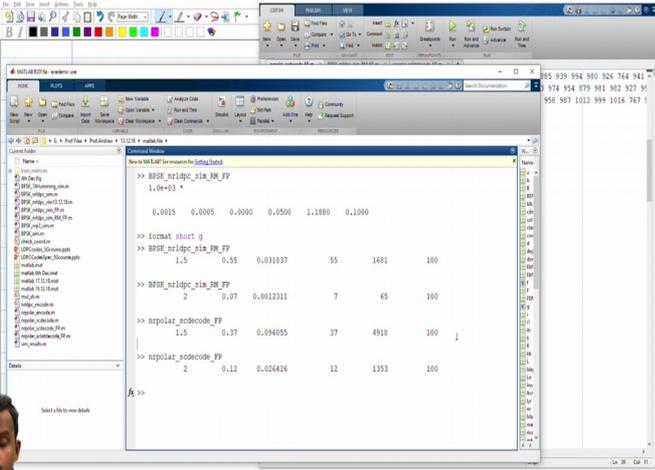
Fixed Point Quantization for SC Decoder and LDPC Decoder



```
%% SC and LDPC Decoders for Fixed Point
>> BPSK_mldpc_mim_FP
1.0e+03 *
0.0015 0.0005 0.0000
>> format short g
>> BPSK_mldpc_mim_FP
1.5 0.55
>> BPSK_mldpc_mim_FP
2 0.07
>> nnpolar_decode_FP
1.5 0.37
>>
31 972 761 877 952 495 703 935 978 883 742 503 925 878 735 953 885 939 994 980 926 764 941
32 971 850 509 949 973 1000 892 950 663 759 1000 510 979 953 763 974 954 879 961 982 927 95
33 511 988 1001 951 1002 893 975 894 1009 955 1004 1010 957 983 958 987 1012 999 1016 767 1
34 991 1020 1007 1015 1019 1021 1022 1023 1+
35
36 N = 1024;
37 K = 512;
38 N = log2(N);
39 EDMSB = 2;
40 rmax = 4; %max received value
41 mmax = 31; %max integer received value
42
43 Rate = K/N;
44 EDMS = 10*(EDMSB/10);
45 sigma = sqrt(1/(2*Rate*EDMS));
46
47 Q1 = Q1Q=0; %reliability sequence for N
48
49 F = Q1(1:N-K); %Frown positions: Q1(1:N-K)
50 %Message positions: Q1(N-K+1:end)
51
52 %simulate
53 Nblocks = 0; Mblocks = 0; %Nblocks = 100;
54 for blk = 1:Nblocks
55 msg = randi(0 1,1,N); %generate random N-bit message
56
57 u = zeros(1,N);
58
59 u(Q1(N-K+1:end)) = msg; %assign message bits
60
61 n = 1; %number of bits combined
```

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder

PROF. ANDREW THANGARAJ
IIT MADRAS

Fixed Point Quantization for SC Decoder and LDPC Decoder

Okay so let me show you how this decoder runs I have set it up so that EbNodB is 1.5, DB maximum iteration is 10 okay and then I am running it for 100 blocks okay. Let us run this okay so it is finished running and gave you some answers may be you cannot see that very clearly so maybe I should run it once again, I will run it once again just to show you how these numbers look it is not too bad we got 5 errors out of 100.

So it shows 55 errors out of 100, so it shows that it works and gave you some decoding, so if you want you can go to 2dB to increase that and we will get 20 hopefully. Yes so it had only 7 out of 100 in error when you went to 2dB okay. So that is the decoder, it works in simple way, so hopefully the fixed point part of it is clear you can also run the polar decoder this is, let us also run it at 1.5 I think again 100 blocks okay so let us run this once again to be sure that we get something reasonable okay it worked reasonably 7 out of 100 and if you go to 2 you will get you get some performance 12 out of 100 was the error so this is how the fix point is done okay.