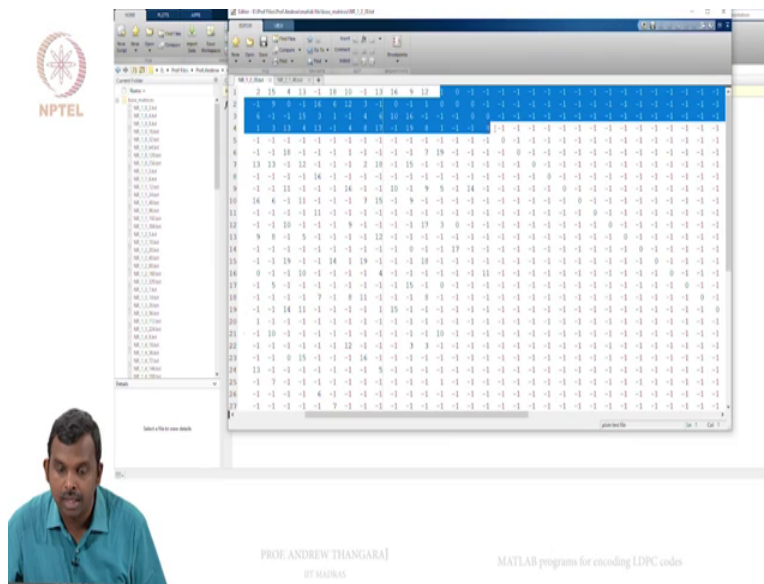


(Refer Slide Time: 01:0)

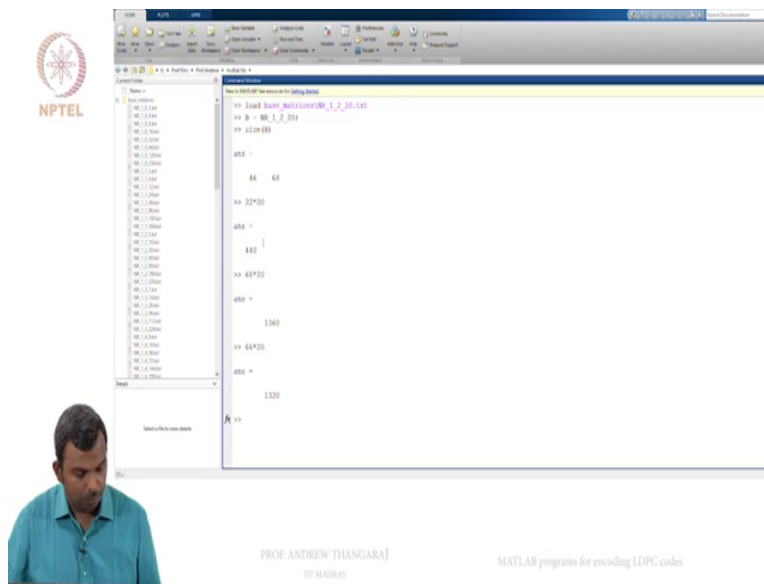


So it has all the base matrices as text files and I will show you maybe one text file opened here ok so this is again a MATLAB editor window in which I opened one of the text files infact two of them this is NR 1 to 20 so this just my notation the 1 means it is the first base graph, 2 is I think j equals to and then 20 is the expansion factor ok so this is base matrix.

The first base matrix if you remember the first base matrix is 46 rows and 68 columns ok it goes all the way upto 68 and if you go up and see in the first four rows right the first four rows you will have the double diagonal structure ok double diagonal structure sort of starts here ok so you can see that see that right there right so the double diagonal structure starts there and this part is the double diagonal ok.

So likewise you will have an every single code and infact this folder will be made available to you and you can download this folder with all the base matrices and in your working directory whichever computer you are using I would suggest that you keep this so that you can access the base matrices whenever we want ok. The base matrices will play a very important role and that is something you have to do ok.

(Refer Slide Time: 02:25)



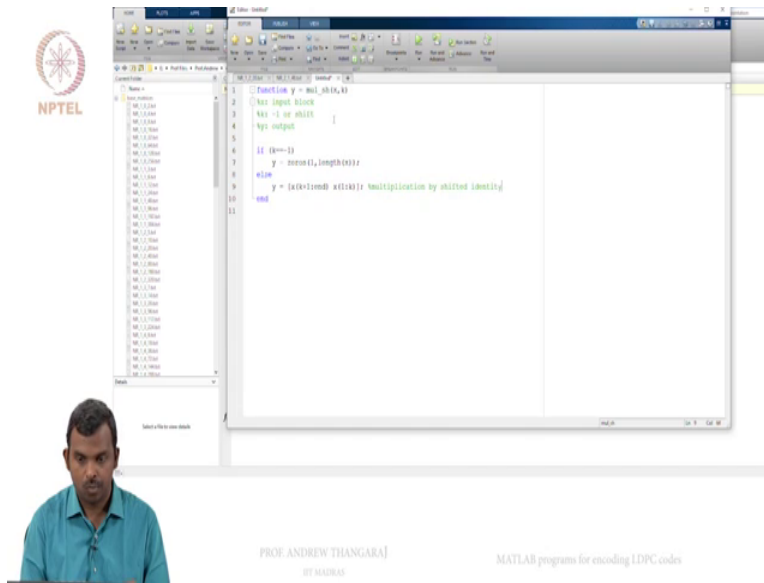
So let me show you how one can load extra, so I am in this directory the main directory and let's say I want to load one of these base matrices 1 to 20 that I showed you so I could do this I can do load base_matrices slash NR 1 2 or you can do a tab here you get this thing so it will load that matrix into your work space and the file that gives us NR underscore 1 underscore 2 underscore 20 that is a bit unwieldy to deal with so I usually like to do this so this brings that matrix N for me size of B is 46 by 68 ok, so 46 by 68 that is 22 block messages each message block is 20 that's long I told you this NR 1 to 20 gives you an expansion factor of 20.

So 22 into 20 that's 440 bits of messages ok and how many bits of code words will come out? 68 into 20, 1360 bits of code word and 22 into 20, 440 ok so a slight carry at here upto 1, 4 in the 40 message bits is ok that is the message bits that will come in but usually when you translate your code word in the LDPC standard the 5G standard the first two message blocks are not transmitted they are punctured ok so the maximum number of bits you will get is actually 66 into 20 that is 1320 ok the rate the lowest weight you can get is 1 by 3.

So in fact people also don't transmit all the parity bits they will transmit as many parity bits as they need ok so you don't transmit all the parity bits so can get different rates that way ok but for now we won't bother the those kind of things we will just work with the entire matrix and see what we can do. The puncturing of the first two blocks alone we may consider but generally this is the picture ok hope this is clear to you ok. So someone write a little piece of MATLAB code

for generating this bits at random and then doing the encoding both the double diagonal part as well as the main diagonal part which is slightly easier ok, so hopefully this is clear to you will see it is relatively easy program to write there is one little component one needs to take care of because other than that it's a easy enough program that one can write.

(Refer Slide Time: 04:56)

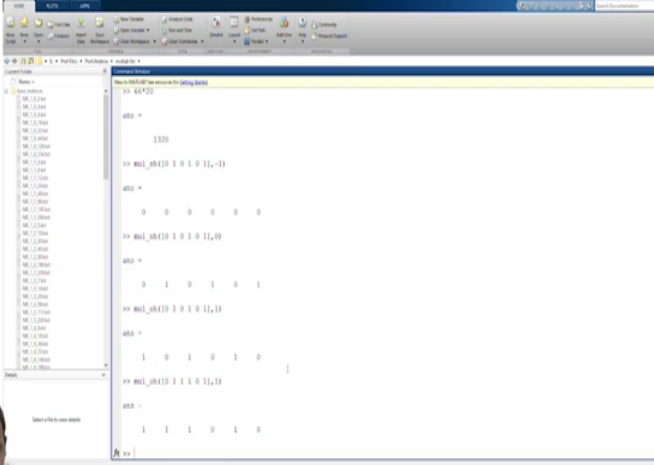


So lets get started ok, this is the editor window so both the files are here so I will take a new one ok so the first function I will write it is a little bit of utility function this does multiplication so I will say y equals so I will call it multiplication shift ok of vector x by k ok, so this is basically if you remember the every block of the message gets multiplied by a shifted identity matrix ok, so this does that so multiplication by the shifted identity matrix ok.

X is my input block ok and k is the shift ok so that is the way to think of it maybe I should write that down x is the input block, k is minus 1 or shift ok y is output ok so it is a little check you have to do here if k is minus 1 then y equals y is just zero, zeros of 1, length of x right so as many bits as over there else y is x of remember MATLAB has indeed a starting from 1 ok what while k starts from 0 ok, so if k where to be zero I have to put out the whole thing right so k plus 1 to end ok if k is 1 I have to start from 2 ok and then you go from 1 to k, is that ok?

So think about what this is doing this is doing the multiplication by shifted identity ok so let me save this, so this is my little utility function.

(Refer Slide Time: 7:09)



The MATLAB script demonstrates the encoding of a 6-bit message vector $[1, 1, 1, 1, 0, 1]$ into a 12-bit codeword. The process involves multiplying the message vector by a generator matrix G and adding the result to a parity-check matrix H to produce the final codeword.

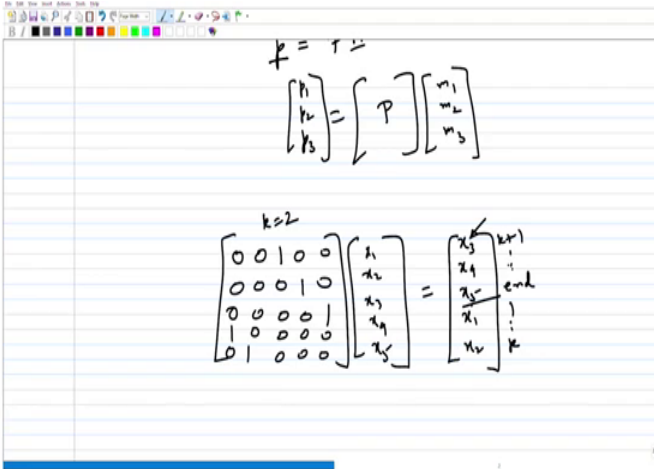
```

>> k=2;
>> H=[1 0 0 0 0 0;
      0 1 0 0 0 0;
      0 0 1 0 0 0;
      0 0 0 1 0 0;
      1 0 1 0 1 0;
      0 1 1 0 1 1];
>> H=mod(H,2);
>> G=[1 0 0 0 0 0;
      0 1 0 0 0 0;
      0 0 1 0 0 0;
      0 0 0 1 0 0;
      1 0 1 0 1 0;
      0 1 1 0 1 1];
>> G=mod(G,2);
>> m=[1 1 1 1 0 1];
>> m=mod(m,2);
>> r=m*G;
>> r=mod(r,2);
>> c=r+H;
>> c=mod(c,2);
  
```

PROF. ANDREW THANGARAJ
IIT MADRAS
MATLAB programs for encoding LDPC codes

Let me show you a little bit of an example so it is good to see how this function works. So let's take 01 01 01 01 length 6 vectors then maybe you do minus part ok so this is supposed to multiply by minus 1 I will get all zero and getting back so let's say we do zero I have to get the same thing again ok.

(Refer Slide Time: 7:41)



The handwritten notes show the matrix equation for LDPC encoding:

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} P \\ 0 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}$$

Below this, a specific example is shown with $k=2$:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \\ x_1 \\ x_2 \end{bmatrix}$$

PROF. ANDREW THANGARAJ
IIT MADRAS
MATLAB programs for encoding LDPC codes

Once again remember what needs to happen if you want you can write a little bit of thing here so what is this mulshift supposed to do ok.

So if I have matrix like this 00100, 00010, 00001, 10000 and then 010000 if you multiply by a vector x_1, x_2, x_3, x_4, x_5 what are you suppose to get out? You will get x_3, x_4, x_5, x_1, x_2 ok so if this corresponds to k equals to ok so it needs to start at 3 right k plus 1 till the end ok k plus 1 till the end and then 1 to k ok this is what I did here so that needs to happen here so you can see that is happening so when you put 1 the it got shifted up and the zero went to the last ok. So if you put 2 the same thing happens ok, so this is doing the multiplication correctly so lets do upto 4.

So the mulshift is working decently it should work even for larger ones it is small piece (()) (8:48) ok. So once I have this another useful piece of code to write is checking whether code word is valid or not ok so what should the checking whether code word is valid or not? Given a base matrix and you given the expansion factor and you have given the code word you have to multiply the expanded base matrix with the code word and see if you get all zeros ok.

(Refer Slide Time: 9:22)

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

```

1 function out = check_word(b,c)
2 % b: base matrix
3 % c: expansion factor
4 % candidate codeword, length = k+1 * z
5 % out = 1, if codeword is valid 0, else
6
7 % b: [1 0 0 0 0]
8
9 % c: [0 0 0 0 0]
10 for i = 1:z
11     for j = 1:5
12         out(i-1*j+1) = mod(b(i-1*j+1) + c(i-1*j+1)*b(i), 2);
13     end
14 end
15 if sum(out)
16     out = 1;
17 else
18     out = 0;
19 end

```

Below the code, there is a small video inset of Prof. Andrew Thangara, a man with a beard wearing a blue shirt. The text 'PROF. ANDREW THANGARA' and 'IIT MADRAS' is visible below the video. To the right, the text 'MATLAB programs for encoding LDPC codes' is visible.

So that is checking validity for the code word so lets call that a new function. So the answer is actually yes or no so I will just call it out ok equals check code word of base matrix ok the size is not so important but the expansion factor is important I call it z and then the code word itself ok so call it c ok so lets say c ok. So b is base matrix this is at this expansion factor and c is code word ok so that's is what it is maybe you should just save this ok and it will tell you 0 or 1 out, out will tell you whether you are right or wrong out is equal to 1 if code word is valid, zero otherwise, zero else ok, so this is what we need to do.

So we need to write a little bit of little piece of code to do this multiplication so I am going to keep it very simple so I am not in this lecture atleast going to try and write very efficient MATLAB code just write the simplest possible MATLAB code that one can get ok so I will call m, n as size of b this tell you the number of rows and number of columns in b and then for every row I have do something ok for every row this is the ith row in the ith row what do I do? I have to do my multiply shift right, so I will keep the ok so it is good to do multiple ways.

So one of the things to do is the out we can initially assign zeros of m into z 1, yeah so m into z, 1 so this is the output after the thing has been multiplied I am sorry this is not the output this is the H times so this is syndrome ok so syndrome is H times the code word ok or the received word ok to see if it is valid or not and syndrome is all zero then it is then the valid code word is valid so this is H times c transpose ok. So what do I do here? So it is all initially zero so I am going to go through every column 1 colon n so (12:20) so this the first block is controlled by either way so one can think of this as I times so I think it is I minus 1 times z plus 1 colon I times z right so this is the block of th syndrome and this equals just the same thing plus the mulshift of c of right j minus 1 times z plus 1 colon j times z ok.

So once again I am not trying to be very efficient here b of I, j ok so this is also a mudulo operation here, two ok. So this is reasonably alright it is a bit efficient I will agree with you if you look at it those of you who know MATLAB coding very well you will identify a lot things here which I could have done differently there is also this confusion between the row and the column I am thinking of the code word as a row vector and I think this command should work, should adjust the row and the columns suitably in case it complaints will see, will see if there is a problem or not and then I am using the mulshift to get the multiplication done and them technically adding row to a column but I think it should be ok that should work out quite ok I think.

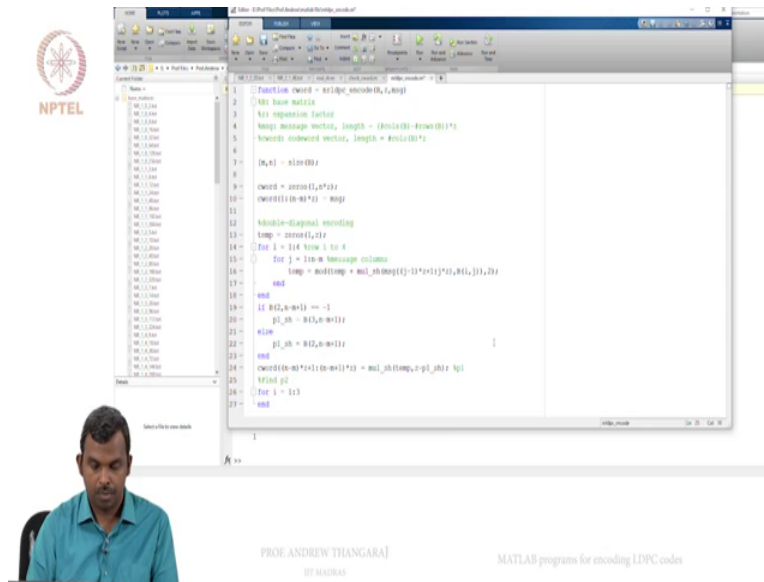
So lets see if there is a problem will make an adjustment (14:28) you know a transpose in one of those places ok so this is done so this tells you what it is so if any of (14:29) ok so if any of the things are non-zero in the syndrome you can set out equals zero else out equals one ok, so that is the final thing. So this is just computing checking whether a particular code word particular vector of length n into z a candidate code word I should say here ok so number of columns of b times z that is the length, b is vector ok that is it, so that's nice enough.

So we had a b here size of b was 46 by 68 and z was 20 if you remember that was the expansion factor so let me generate random code word or first thing is so lets generate a code word and then see if this works out so c is so my code word needs to be 68 into 20 right so I will do (random) I think lets do a()
(16:03) I think I now the command just being very sure yeah sure so you can do this rand I if you remember 0 1, 1, 10 this should give you random bits ok so lets take random vector 1, 68 into z ok then we can do check code word b, z, c ok it is complaining about matrix dimensions not agreeing so one needs to go there and then change what you get at the output here (so I think) will be I made a another mistake here (b), so this one we can conjugate.

So this if we do think it should work save this, there you go so its zero its is not a valid code word it is very-very unlikely that we find the valid code word ok. So one way to definitely find the valid code word is to set the set this as zeros of 1, 68 into z this case we should get 1 we do get 1, so it seems like we can't find the code word randomly that is obvious you won't be very easily define a code word, cant very easily find the code word at random
(17:47) this is working it seems to be working will actually generate a valid code word and see if it works over not ok.

So this is just check one can ofcourse make this, this all sets so all sorts of efficiencies improvements possible here if you were interested you can take it out ok, so it had two ingredients multiply in shift and this ok.

(Refer Slide Time: 18:10)



Now we are ready to write the actual encoder ok so lets write that. So 5G (18:25) ok, (18:29) base matrix needs to be given, the expansion factor needs to be given and the message needs to be given ok, so for now I will take my b and z as same as here little lazy (18:51) b is the base matrix, z is expansion factor, the message will be (the message) length equals number of columns of b minus number of rows of b multiplied by z ok so that's got to be the message and the code word.

So what I am going to actually put out will be code word vector the entire code word vector so length is going to be number of columns b multiplied by z ok so you can go ahead and later on if you gonna like it you can (did I make a mistake here?) oh it can't start with a 5 ok so I will just leave it as R LDPC ok so there you go so all that is done so from now on its sort of whichever show the code word and checking up sort of similar so will see will start the messenger and then size of b ok and you know the first four rows alone are little bit special as an I have to do them slightly carefully then compute the first parity.

Once I compute the first parity and the second parity, third parity and so on go on and then in sort of a very sequential fashion ok so the first four bits we will do that separately so this is the double diagonal encoding ok. So the messages given to me Msg so I need the code word, it is a code word will assume (20:48) equals zeros of 1, m times zero right so that is my code word so I can actually assign the first so 1 colon m, m minus n times z as equal to the message this is

the useful way of checking in case your message is not of the correct length this will complain so this is good to have right there ok.

So then we start with the double diagonal encoding so the double diagonal encoding is going to give you the first four parity blocks ok and how do I do this? This is not very difficult so (i) $(21:37)$ equals 1 colon if you remember I need to do and minus m ok, so I have to do the overall parity so I will its good to keep like temp here temp is zeros of $1, z$ so this is going to be my temporary think that I keep adding of j so i equals 1 to sorry here 4 ok so I made this mistake this is the rows ok so this are the rows, row 1 to 4 and j is 1 colon n minus m ok so this is column message columns right.

So what are they do for each i and j I am going to say temp equals mod of temp plus mulshift space is here now readable mulshift of the corresponding message (i) $(22:52)$ so it is gonna be (i) $(22:54)$ message or code use the message of you need j minus 1 times z plus 1 colon j into c ok comma b of $i, j, 2$ ok so this is gonna compute at the end of this loop temp will be all of the message in the first row, second row, third row, fourth row all of them added together. So when this happens if you remember the double diagonal structure the p_1, p_2, p_3, p_4, p_2 and p_3, p_4 will get cancelled p_1 alone will remain the first and the fourth row p_1 components will again cancel will be equal, only the second or third whatever it is will remain, is that right?

So I have to now look for what is in the second and third row in the parity part and then do shift accordingly ok, so I will do that little bit of calculation here so once again what is so I try to find the double diagonal for the parity what multiplies the parity ok that I need to find so that is either there are two possible values here, it is either b of $2, n$ minus m plus 1 or p of $3, n$ minus m plus 1 so what I will do is I will take p_1 shift to be equal to this ok so all maybe I will do slightly differently if this is equal to minus 1 p_1 shift is equal to b of $3, n$ minus m plus 1 else p_1 shift equals p of $2, n$ minus m plus 1 .

(Refer Slide Time: 25:07)

NPTEL

PROF. ANDREW THANGARAJ
BY MADRAS

MATLAB programs for encoding LDPC codes

So let me show you (why) where this is coming from, so you can go and look at this the to 20 for instance if you look at the parity part ok so you see this is a first row this is the double diagonal part the first row of the double diagonal you see there is 1 0 0 0, this is the second row of the double diagonal third row of the double diagonal and fourth row of the double diagonal. So you see this two are the same and this will cancel so p_1 will just be multiplied by this guy ok so this happens in the 1 to 20.

If you go to 21 now this what happens here, this is the double diagonal part right, so first row of the double diagonal part F0 here and this is actually minus 1 and this is 1 and this is zero ok, so this two are going to be equal to the first and the fourth will always be equal they will get cancelled but there will be a remaining shift which is either the second one or the third one ok, so which is sort of keep track of that so that remaining shift whether it is a second or the third I am taking care here, so this p_1 shifts will either be the second value or the third value depending on which is not minus 1 ok let me guess but I have to write it that is what I am writing ok.

So this is the p_1 shift so this p_1 shift is what multiplies the p_1 ok, so I have to undo this from temp to get p_1 ok so p_1 ok which is actually c word of so this is the n minus m plus 1 component so it is n minus m times c plus 1 colon n minus m plus 1 times z this is the p_1 first priority this is mulshift of temp, it is not p_1 shift, p_1 shift got multiplied by this I have to undo this so that would be z minus p_1 shift ok.

So one needs to be slightly careful here so the only confusion is what if p_1 shift is zero?

Ok if p_1 shift is zero I am putting a z into the mulshift if I put z into the mulshift what will happen? It will go z plus 1 to end and then 1 to z ok so it is consistent putting z into the mulshift is kind reasonable so I don't have to do anymore adjustments here everything else will give you the correct value ok. So this gives me the p_1 , is that ok? So think about what I did here, I did quite a few things I went through the first four rows I $(())(27:29)$ all of the first four rows and then looked up what was remaining is just p_1 and then find the multiplying factor matrix for p_1 and then I did the inverse of that to get p_1 itself ok, so that is what happen.

Think about this and then convince yourself that not made too many mistakes I think it is ok finally we will see if the code word is valid or not right, so that will tell us whether you are on the right track or not ok. So now from p_2 onwards ok so to find p_2 we use the first row so if you remember you can go back and look again to find p_2 now once the p_1 is known you find p_2 is in the first row, p_3 is second row, p_4 is in the third row then after that this will this is sort of redundant and then we will make a jump ok.

So that is the idea here, we make this little jump and then we go through ok, so there are various ways to write this so let me do that right away ok, so will do it two steps for I will do once again for I equals 1 to 3 this is the first three rows for computing p_2 to p_3 ok little bit long so ok we put some space becomes all ok.

(Refer Slide Time: 28:58)

```
18 = end
19 = if H(2,n-m) == -1
20 = p1_sh = H(2,n-m+1);
21 = else
22 = p1_sh = H(2,n-m);
23 = end
24 = cword((n-m+1):(n-m+1)*z) = mul_sh(temp,c-p1_sh);
25 = mod(p2,p1_sh);
26 = for j = 1:3
27 = temp = remod(c);
28 = for l = 1:n-m+1
29 = temp = mod(temp + mul_sh(cword((l-1)*z+1):(l-1)*z+1),H(1,3));
30 = end
31 = cword((n-m+1)*z+1):(n-m+1)*z+1) = temp;
32 = end
33 = remaining_parity;
34 = for i = 5:m
35 = temp = remod(c);
36 = for l = 1:n-m+1
37 = temp = mod(temp + mul_sh(cword((l-1)*z+1):(l-1)*z+1),H(1,3));
38 = end
39 = cword((n-m+1)*z+1):(n-m+1)*z+1) = temp;
40 = end
41 =
42 =
43 =
44 =
```

so first row and then once again I just need to do for j equals 1 colon I mean lot of people who watch this will closely will try that I am doing lot of computation repeatedly I think it is correct I agree with you but just like I said we are not trying to be very efficient here you can make the program more efficient if you believe you want to do it that this is just to get once encoder out ok.

So notice what I have done here, I have done from I equals 1 to 3 and then j equals 1 to n minus m plus I so I need to make a n minus m plus I column so if you look at the first row here I go upto n minus 1 and then I need to look up this also right. So this one also plays a role that is important to know ok and what are ())(29:46) here you can have directly you can have something like this c word so maybe will make it a little bit more efficient right so will keep it will keep temp as just to make it look a little better 1, c and then I will do temp equals mod of temp plus mulshift of c word of the this is just the same as before z plus 1 colon j times c , b of I, j, 2 ok and then once this computation is done you can assign c word of n minus m plus I time z plus 1 colon n minus m plus i plus 1 yeah times z to be equal to temp ok.

So this takes care of p2 to p4 I am sorry and p2,p3, p4 ok so this takes care of that and then for the remaining part I have to go from 4th b end for I equals 4 not 4, 5 colon m ok remaining parities so not there is no need to do any shifting here the shift gets so it is all identity right so I am sorry so if you notice it is all identity after this right, so it is all identity after this right so it is

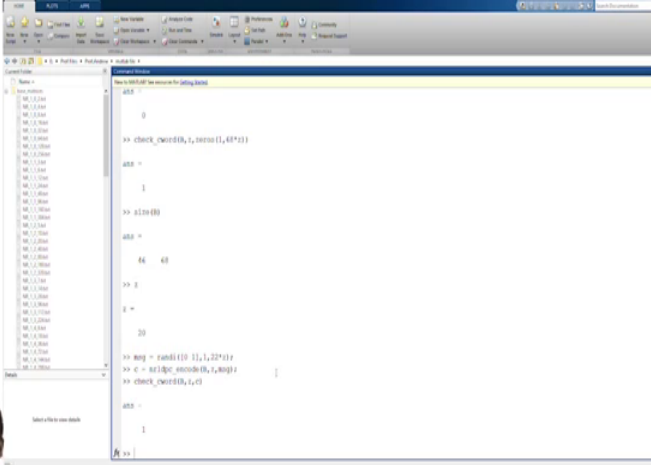
all identity is 00 even here it is all 0 0 0 0 0 so and also if you see this whole part is minus 1 so it is enough if you stop in the columns upto this point you don't have to go beyond this ok.

So for j so you can do the temp once again just a little bit here c and then you do for (something wrong here) for j equals 1 colon n minus m plus 3 or 4 ok don't need to go beyond that ok so once again notice what I am saying here so you have n minus m plus 1 here n minus m plus 2 here, n minus m plus 3 here, n minus m plus 4 here ok, so after that there is always zero, so no need to worry about all the other things so from p5 onwards you just have to stop you can stop here, there is some eleven etc here so most of it zero but nevertheless one can take care of this ok.

So I am going to go through the whole thing then write the same sort of commands so maybe I should just copy paste here (copy paste here mod of temp multiply I think you don't have to change anything here it is exact same thing, exactly so and then c word) (33:29) ok so I think that is odd of n's the whole thing it is good to check the last value so supposing I is m you have ok so I think there is something slightly wrong here let me just check this out, here you need a change I think this j I think is ok the I here the fifth is ok but the first I needs to do I think you need a minus 1 here, so this is correct ok.

So this is because I shifted down by one so you need this little bit of adjustments to make this work ok alright so I think that should take care of most of the things I think I have not made any mistake here so except for maybe should I worry about the column or problem I think it is ok everything looks fine here as far as I can see well the proof of the (34:43) is in the eating so lets try and run this and see what happens ok. So run it hopefully it should work I think not made anything significantly wrong here nothing is gone wrong here ok.

(Refer Slide Time: 35:03)



The screenshot shows a MATLAB script execution window with the following code and output:

```
%%
a = rand(100,1,48*1);
%% check_word(a,7,2000(1,48*1))
ans =
    1
%% 110000
ans =
    04 40
%% 1
z =
    20
%% msg = rand(100,1,1,22*1)
%% c = LDPC_encode(a,7,2000(1,48*1))
%% check_word(c,7)
ans =
    1
```

Below the MATLAB window, there is a small video inset of a man in a blue shirt, and text identifying him as Prof. Andrew Thangaraj from IIT Madras, with the title "MATLAB programs for encoding LDPC codes".

So let's try to run this if there is a problem we will deal with this. So my c is gonna be so I need to generate a message now. So my size of b is what? 46 by 68. My message expansion factor is I think 20. So my message is gonna be $\text{rand}(100, 1, 22)$ times. Right? n minus m 22 times z . So this is my message and I am gonna let $y = \text{NR_LDPC_encode}(p, z, \text{msg})$. Wow, there is no error. So it's no runtime error. So let's just check whether it is a code word or not. Right, so this is it. It is 1. So it says it's a valid code word. So the message was random.

(Refer Slide Time: 36:06)

NPTEL

Column 1247 through 1305
1 0 1 0 1 1 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0 1 0

Column 1306 through 1368
1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1 0

Column 1369 through 1431
1 1 1 1 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1 0 0 1 1

Column 1432 through 1494
0 1 1 1 1 0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 1

Column 1495 through 1557
1 1 0 0 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 1

Column 1558 through 1620
0 1 1

```
>> check_word(c,1,c)
ans =
     1
```

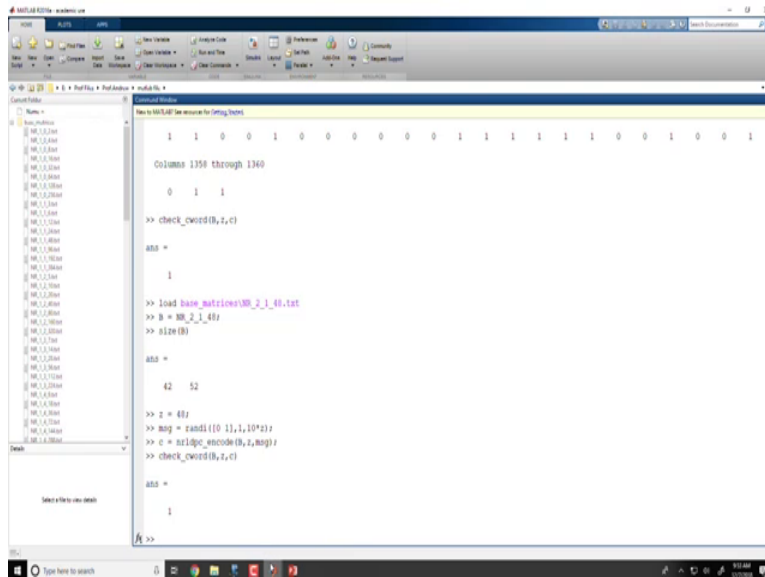
PROF. ANDREW THANGARAJ
@ IIT MADRAS

MATLAB programs for encoding LDPC codes

So if you look at the message it had a lot of zeros and ones and c word have a lot of sorry did I put c word here?

C I am sorry, code word we can a lot of zeros and ones and then when we did check code word I got one ok which means the encoding happened and you got the correct code word right, so remember once again what is (())(36:28) tell you? Out equals 1 code word is valid ok, so you can try various other so the encoders basically working so lets try one more will try loading the other one so the other example I gave you was the other one I pulled out, 1 to 20 right to 148.

(Refer Slide Time: 36:53)



```
1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1
Column 1359 through 1360
0 1 1
>> check_cword(b,z,c)
ans =
1
>> load base_matrices38_2_1_40.txt
>> B = B_2_1_40;
>> size(B)
ans =
42 52
>> z = 40;
>> msg = randi([0 1,1,10^4]);
>> c = xzldpc_encode(b,z,msg);
>> check_cword(b,z,c)
ans =
1
```

So let's try (36:52) and message is 10 times and the message I am getting c encode this (37:38) this code I will make it (37:45) this encoder for let me see (37:50) see how this work in MATLAB easy to write just a few lines of code you just need the base matrices (38:06) this is very easy to do hopefully so this is clear to you (38:14) next time decoder which you might imagine is significantly (38:27) decoder itself is easy to write down will write the similar program involving base matrices (38:34) decoders so that will see in the next lecture, thank you very much.