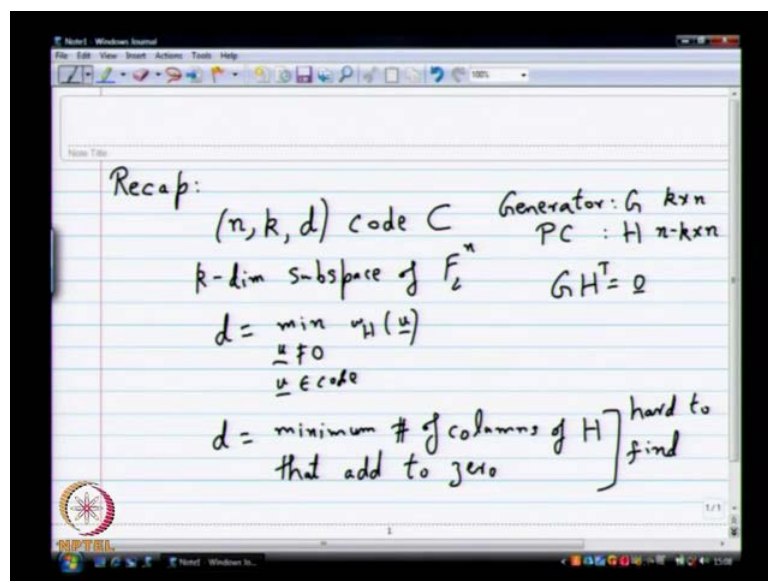


**Coding Theory**  
**Prof. Dr. Andrew Thangaraj**  
**Department of Electronics and Communication Engineering**  
**Indian Institute of Technology, Madras**

**Lecture - 5**  
**Operations on Codes**

So, let us begin the next lecture. The last thing we saw in the previous class was about minimum distance, but I want to recap what we saw in the previous lecture.

(Refer Slide Time: 00:27)



So, let us do a quick recap, so right now we are thinking of three parameters for any code  $n, k, d$  code case once again clearly this is going to be a dimensional sub space of  $F_2^n$  no change in that, but what is this new parameter  $d$ . Now,  $d$  is minimum over  $u$  such that  $u \in C$  the Hamming weight of  $u$ , so you look at all the non zero vectors in your code the one that has smallest weight among all those things you get minimum weight.

So, that is what we saw and then this minimum distance  $d$  like a point at out that is not a very linear algebraic parameter it is more of a combinatorial parameter. So, it is more difficult to determine directly the only relation it has with the parity check matrix is the following relationship you can also show  $d$  is the minimum number of columns of  $H$  that add to zero.

So, if the  $n$  code  $n$   $d$  codes  $c$  has generator matrix  $g$  and parity matrix  $h$  remember  $g$  is a cross  $n$  matrix full rank and  $h$  is a  $n$  minus cross  $n$  matrix you can think of them in systematic form this  $g$  becomes  $i$   $p$   $h$  will be  $p$  transpose  $i$ . Then the relationship  $g$   $h$  transpose equals zero is also satisfy, so these are all the things that we saw in the last class like a point it out this is hard to determine in general.

So, usually the strategy is to design a parity check matrix  $h$  and then show some  $d$  minus one when any  $d$  minus one or fewer columns are linearly independent. If you do that, then your minimum distance is at least  $d$  so that is the strategy that is used to show, so if you want to show it is exactly equal to  $d$  you also have to find the set of  $d$  columns which do definitely add up to 0. So, you do that you can find these things, so let us see couple of quick examples and work out the couple of parity check matrices at  $u$  and then ask you for the minimum distance we also started on the hamming code at  $s$ .

(Refer Slide Time: 03:48)

Ex:

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad n=6, \quad n-k=3, \quad d=3$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad d=?$$

So, we will come back to that soon enough but before that lecture quickly see two quick examples just to drive home this point of how easy it might be to find the minimum distance. So, the first example try that the next example may want to try let us see the first example. I have parity check matrices the easiest think to determine is what both the first easiest think to determine  $n$  is the easiest parameter and you can determine how for  $n$  minus 6. I am sorry, see even the easiest parameter can some time you have so and equals 6.

Then, next thing the you can determine as some parity check matrices the number of lineally independent rows and that would be  $n$  minus in this case what and this the form of the matrix is easy enough. You can quickly see that three rows a lineally independent in case you have a  $i$  sitting inside, so from there you can quickly see that  $n$  minus 3. From there, you can quickly conclude that has to be three years what about minimum distance, so it is very safe when you complete minimum distance to be patient.

Eliminate one after the other and then go to three in case do not quickly jump to case three and there is correct, but it is good to do the step by step the first step is first fall is not  $d$  equal. You know  $d$  is not see row that you know what about  $d$  equals one how do you eliminate  $d$  equals 1  $d$  equals one can happen only if there is the all zero column. So,  $d$  equals one is eliminate  $d$  cannot be one just be at least 2, how for 2 what should happen two column should be identical should be repeat and you can see the there is no representation.

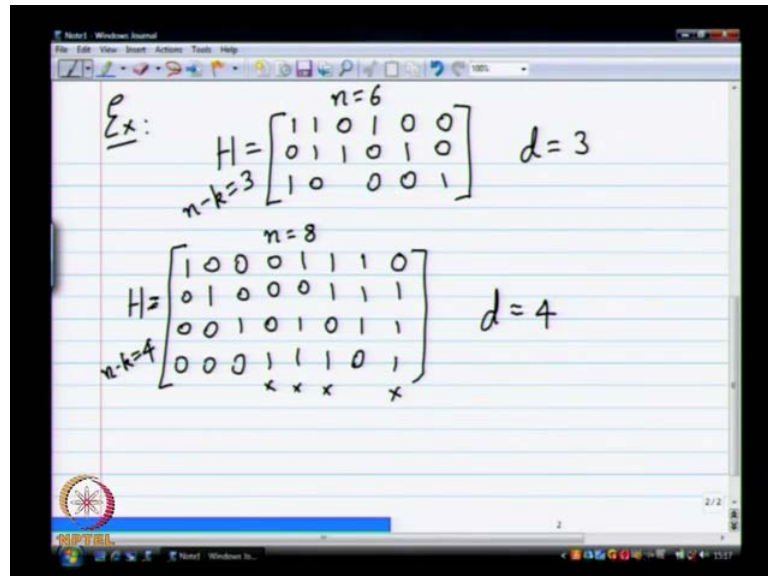
So, two is not also possible  $d$  as to be at least 3 and can you find three columns there add to 0 is request, so there are multiple choice is possible, so for instant effect pick this column this column and this column they add to 0. So,  $d$  clearly becomes 3, so know another question and that might be you of interest us how many minimum weight code words are there that might be an interesting question to think about. These are again difficult question, but you can try to answer at for instant in this code how many can you quickly see one as a putdown is clearly a minimum weight code word another one. You can have is this one this one and right another one you can have is this guy this guy and this guy anything else is anything possible.

So, notice there is like non trivial looking minimum weight code word those three things is issue add you get what 0. So, that is four can be anything more so in this cases really simple problem a roundly eight code words rights. You can eliminate everything in find the answer very easily, so I believe this a roundly four code words which around minimum weight and this also find the other once these are there I am right correct yes no body his happy, so there are four of them.

So, it can be a bit more weights it was its only think that I want point out the can be some hidden code words it is suddenly show up. So, let us know move on to the other case what about the other one case once again you do the follow the same strategy saved you

start the equals 1 and d equals 0, you can just through way just space strong what you know in case n minus its n is 8, then n minus is 4.

(Refer Slide Time: 08:56)



So, these two things are able to see quite easily n minus four this easily seen because once again there is i sitting there if you do not have an i you have to do Gaussian elimination generate the i and then you see for d you have to do little bit more work. Then the four let see d equals one is the fed d rolled doubt what about d equals two are also is rolled doubt what about d equals three if it is sharp likely sharp you have really check. Even within this what I do so within this guy see you have check the no 3 adds to 0 and then may be two of them will add.

Suddenly, they will give something which is weight 1, so that might can happen, so if make 1 here in this case it would not happen, but if I make one devise looking change can suddenly happens for instants. If I just remove this 0 and make it 1 what can happen then last two and you will see the last 3 add up to 0. It is a little bit more non trivial to find the minimum distance, but in this case you can no 3 will add to columns to 0. I am not saying it is 0 it is easy to check that check all possibilities how many possibilities, you have to check if you were to do an exhaustive search h 3.

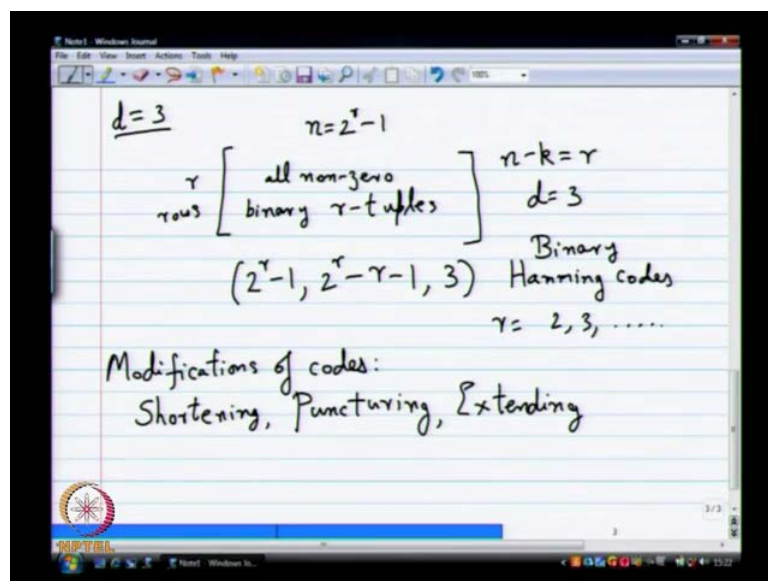
You do not have to do all h o 3, what can I avoid easily among the first four columns there is no way anything is going to add up to 0. So, there is no problem there you can completely remove that under that you cannot completely remove you can take you do

not have to take everything from. So, you can write you can reduce if you like you can be smart about it, but at the end of the day it is some kind of exhaustive it is nothing much you can.

So,  $d$  will not be three what about four one two three four not one two three so there are multiple possibilities multiple possibilities to get that for instance very interesting looking example. You can come up with if you like this, so if you take this guy and this guy what happens that would be a code word, this is not to give a non trivial example some strange things like this can happen.

So, you can get  $d$  equals four so this is the procedure think about it very carefully hard when I remember the first time this was thought to me and I thought about a lot of algebra type of algorithms. That should give me the minimum distance and after all I can put it in  $n$   $p$  there only and  $n$  and  $p$  and well smarter people than me though about it. So, anyway the final competition as you have to do will always end up being a exhaustive very difficult to do smart algorithm it is  $n$   $p$  complete. You do not know aware of very recent work that is  $p$  may be not empty nobody will have an algorithm. So, it is a difficult problem any questions go larger and larger than  $8$  choose  $3$   $n$  choose three can become better than loop, this case there are so many other ways of doing it anything else. So, let us move on let me come back to this hamming code, so the code that we saw.

(Refer Slide Time: 12:48)



So, I think idea was to construct a parity check matrixes with  $d$  equals three right that was the idea and we are going to pick  $r$  rows. So, this is a parity check matrixes we are going to construct and one of the choices for  $n$  is to pick  $n$  equals  $2$  to the power  $r$  minus  $1$  here and then you put all non zero binary  $r$  tuples binary  $r$  tuples as columns not as rows. This is going to columns and you get a parity check matrix and clearly  $d$  will be for this, what will be  $n$  minus will be equal to  $r$ , both these things need proof is not too difficult.

Why will  $n$  minus will be equal to  $r$  you can find a  $i$   $r$  inside this matrix, so you will have all those guys that will give you a linear degree of the linear independence you need. So, you will get  $n$  minus equal to  $r$  and then  $d$  equals three you can easily see you add two things you will another  $r$  tuples. This is also  $0$ , so have it somewhere occurring you get  $d$  equals  $3$ . So, what are the general parameters of this code  $n$  is  $2$  power  $r$  minus  $1$  and what is  $2$  power  $r$  minus  $1$ , what is  $b$ ?

So,  $b$  is  $r$  general binary hamming codes binary hamming codes an  $r$  can be you want you can start with one if you like, but most non trivial competition starts with  $2$ . Let me just put  $p$  equals  $1, 2, 3$  etcetera, I think it does not make any sense. Let us start with  $2$ , sorry  $r$  will make minus  $1$   $r$  equal to  $1$ , I think will make equals  $0$  and all this. So, this start with  $r$  equals  $2$  all right these are the binary hamming codes and any  $d$  equals  $3$  code.

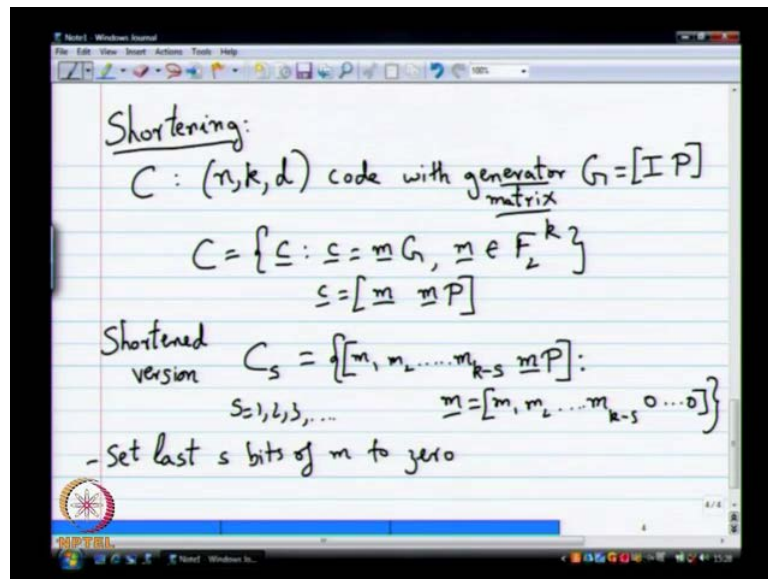
You might build with binary any binary  $d$  equals  $3$  cos you might build will look roughly like this and you have to make you want to make it if you want to make you want to make it very efficient, it will look like this. You know how  $d$  equals  $3$  can be satisfied, we know the reputations we should know all zero culms that it you get  $3$  seems like very simple recipe. It is not too hard many codes have  $d$  equals  $3$ , so that is the hamming code we have already seen some we already seen this code before, but think it is good see it once again, so  $d$  equals  $4$  onwards becomes a little bit more interesting,  $b$  equals  $4$  is not so hard.

It turns out you can get a  $d$  equals  $4$  code from a  $d$  equals  $3$  code and that is close to optimal you cannot do much better than that. So, we will see next what we going to see next is some simple modification of codes, we will see three ideas first idea, I will call shortening next idea is puncturing and third idea is extending, so I have multiple purposes in discussing these modification of codes. The first purpose is, it will give you a

feel for what this generator matrixes are parity check matrixes are how do you manipulate them how the minimum distance gets affected and all that.

When you discuss this, all that will become little bit more clear and it is a good reinforcement of those ideas. The other purpose is these things are extremely valuable in practice maybe not extending both shortening and puncturing are pretty much used in all practical applications. Nobody will use it, it will always be a shorter version or a punctured version, so it turns out in practice progress some parameters this is very, very crucial. So, both points of views is useful, so we will see that very simple ideas, but never the less important to see them in some detail.

(Refer Slide Time: 17:35)



So, let us start with shortening, so let us say we have a  $n$   $d$  code  $c$  is a  $n$   $d$  code with generator matrix  $G$ , I am dropping this generator matrix all the time maybe it is not a very good idea, but remember that I say generator, I mean generator matrix. Suppose, you have  $c$  code  $c$  given like this, so we know how to write this set of all  $c$  such that  $c$  equals  $mg$   $m$  comes from the set of all binary  $2$ . So, it is good to think of  $g$  as a systematic form at least for the shortening you always think of  $g$  as a systematic form  $i$   $p$ .

So, then the  $c$  becomes  $m$  and  $m$   $p$ , so what you do for a shortened version, so go to now define the shortened version you will call it  $c$  sub  $s$  and this  $s$  is integer so is basically one two three four so  $c$  sub  $s$  is basically. So, let me write it down carefully set of all  $c$ , so let me just write it down little bit differently. So, let me write the other thing first so

the basic idea is to restrict your messages to have only  $s$  non zero possibilities the last bits of the message you will always set to 0. That is the idea of the shortening you basically set what that  $s$  tells you set last  $s$  bits of  $m$  to 0.

Basically, you shorten your message more than shortening the code when you shorten the code you are shortening your message. So, you have first  $m-s$  bits such that any bits that you want and the last  $s$  bits you force them to 0. So, what will happen, I do that when I do  $m$  times  $g$  last rows of  $g$  pretty much drop out of consideration only have to worry about first  $m-s$  rows. When I form a code word what will happen the message part of the code word will have the first  $m-s$  bits, but then the last  $s$  bits of the message will always be 0 unless no point in transmitting that.

I do not have to transmit that, so I can just simply drop those  $s$  bit in code also the code word also, the way you write it, so you take always a little bit confusing to write it and in a very optimal way. So, let me just see maybe there is a efficient way so this seems like the best way you can write it, so you write as  $m_1 m_2$  so on till  $m-s$ . So, you take  $m-s$  bits and then what do you do leave out the 0, so the remaining things I am going to drop out it just does not mean much.

So, changing the generator matrix a little bit and then what you do, so little bit more confusing, so let me just write it let me just write. Then we will worry about what happens, so maybe I make it  $m \times p$  such that  $m$  is what  $m_1 m_2 m-s$  and then 0 just the laborious way of writing, but I think it is good to write it once see how it looks like stuck here. So, I can define another  $p$  prime if you want which is just the first  $m-s$  rows of  $p$  and simply write it as  $m \times p$  where  $m$  is simply the only the  $m-s$ .

You can do it that way also, it is just another way of writing it is that clear it should be clear. So, basically restrict yourself to on the  $m-s$  bits of the message and the remaining bits you set to 0 is there a question. I will come to that practical valuable let us come to that slowly there is a question about practical why it will be useful.

I mean see you do not have think of shortening as a trade idea it is never a less simple idea, so let us see the interesting question. Now, is what are the parameters for  $c$   $s$  what the generator matrix for the  $c$   $s$  what the parity check matrix those are the things that are interesting? So, I want to concentrate a little bit like I said  $i$  gives you a practice with thinking about what this matrixes are really.



(Refer Slide Time: 23:58)

$$C: G = \left[ \begin{array}{c|c|c} I_k & P & 0 \\ \hline 0 & X & X \end{array} \right] \quad (n, k, d)$$

$$C_s: G_s = \left[ \begin{array}{c|c} I_{k-s} & P'_{k-s} \\ \hline 0 & 0 \end{array} \right] \quad (n-s, k-s, \geq d)$$

Suppose  $c \in C_s, c \neq 0 + w_H(c) = d' < d$

$$\Rightarrow c = [c_1 \dots c_{k-s} \ 0 \dots 0 \ c_{k-s+1} \dots c_{n-s}] \in C$$

$$\Rightarrow \text{min. dist of } C < d \quad \times$$

So, let us start with the generator matrix for  $C$  you have, so let me write down the  $G$  a bit more laboriously  $I, I, I$ , so  $I, I$  you have  $0$ 's here, then you have  $P$  right so  $P$  would be here. So, this is a generator matrix for  $C$  what would be a generator matrix for  $C_s$  maybe it is  $G_s$  the last  $s$  row is  $0$  is that enough the last  $s$  row are  $0$ . So, it goes from  $1$  to minus  $s$  and from there below  $I$  will just remove that, then what else should I remove? The last  $s$  columns are  $0$  also I should remove know that is the idea this also should go away that is quite simple to see but anyway the last  $s$  columns are  $0$  should also withdraw.

So, you would get the  $G_s$  is that, so  $C_s$  would be  $k$  minus  $s$  and then you would have  $P$  from  $1$  to minus  $s$  is that, so what about parameters here you had  $n, d$  what will be the new parameters as you will show  $n$  is the easiest  $n$  will become  $n$  minus  $s$ . What will be minus  $s$  you have a  $k$  minus  $s$  what about  $d$  what about  $d$  it cannot go down, that is most important thing when you shorten  $d$  will not go down. So, it will get greater than or equal to  $d$  how do you quickly show that how do you show that  $d$  cannot go down you mean you used to intuit engineering answers.

You have to a precise proof you have to write it in terms of contradiction you have to assume. Now, suppose  $C_s$  has a code word of weight say some with  $d$  prime which is strictly less than  $d$  and what will happen suppose  $C_s$  has a  $c$  belongs to  $C_s$ , sorry  $C_s$  it is non zero  $c$  is non zero and weight of  $c$  is equal to some  $d$  prime which is strictly less than  $d$  and what can I do. Now, create a  $c$  prime which will be what  $c$   $1$  so on

till  $c$  minus  $s$ , then what will I do? I will add 0's then what will I put for the remaining  $s$   
 $ck$  minus  $s$  plus one all the way to  $c$  minus  $s$ .

Now, what should  $c$  prime be should  $c$  prime be anything special now should it belong to  
any code that you know of maybe any code that is in this page it should belong to  $c$ . So,  
this employee's  $c$  prime belongs to  $c$  why i got  $c$  by shortening a code from the code  
word from the original code. If I go from a code word in a short end version add zeros to  
one point I should get a code word in the original code. Now, what can happen to weight  
of  $c$  prime that going to be  $d$  prime which contradicts the assumption on the minimum  
distance of  $c$ .

So, that implies minimum distance of  $c$ , let me write it down carefully minimum distance  
of  $c$  is less than  $d$  which is a contradiction. So, our basic assumption cannot be true it  
cannot find the non zero code word in  $c$   $s$  it has weight less than  $d$  simple enough proof  
but if I have to write it down to be precise to the board. So, in most cases when you  
shorten it will be equal to  $d$ , so unless you are really lucky or you shorten with the lot of  
force side you cannot do the greeter the big just in the minimum distance.

Sometimes, it might happen you get a larger minimum distance so the question was what  
is the point in doing this it looks like a not gaining anything just designing another linear  
code with different parameters  $n$  minus  $n$  minus  $s$  minus. So, it turns out the way we  
design if you want a particular minimum distance  $d$  you can design it only for certain  
special values of  $n$  and not all possible  $n$ .

It is possible from the design point of view you may not be able to design  $n$   $d$  codes you  
will see some design strategy. Later on, in the course you will see that those strategies  
work only for certain specific values of  $n$ . Maybe, that is what that how it works and then  
you might want some other  $n$  and which is very close to  $n$  and maybe you want  $n$  minus  $s$   
minus, then how do you do it? You do it by shortening. That is that is how it is used in  
practice, so you design a larger code shorten it again to get what you want, so that is how  
it is all the time quality check matrix this is a more interesting question.

(Refer Slide Time: 30:21)

$$C: H = \begin{bmatrix} \overset{1 \dots k-s \ n}{P^T} & X \\ \underset{n-k}{\phantom{P^T}} & \underset{n-k}{I} \end{bmatrix}$$

$$C_s: H_s = \begin{bmatrix} \phantom{P^T} & \phantom{X} \\ \underset{n-k}{\phantom{P^T}} & \underset{n-k}{I} \end{bmatrix}$$

$$H = \begin{bmatrix} P^T & I \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_k \\ c_{k+1} \\ \vdots \\ c_n \end{bmatrix} = 0 \quad \begin{bmatrix} c_{k+1} \\ \vdots \\ c_n \end{bmatrix} = P^T \begin{bmatrix} c_1 \\ \vdots \\ c_k \end{bmatrix}$$

Suppose, I have a code  $c$  which has a quality check matrix which looks like this  $p$  transpose, then  $I$  and minus. So, this is for the original code what will happen to the quality check matrix of the shorten code first a fall let us take a let us take a small minus step. First, what will be the dimensions of the quality check matrix of the shorten code  $n$  minus cross  $n$  minus  $s$ , so clearly what does this suggest I think in the rows will go away.

So, you do not have keep removing anything from the row only the columns have to go which columns will have to go the last  $l \times s$  columns of  $b$  transpose have to go. So, that is the idea so you will just look at the last  $s$  columns of  $b$  transpose remove the last this columns, you will get is that if you not able to quickly I will suggest another way of thing about it about the parity check matrix.

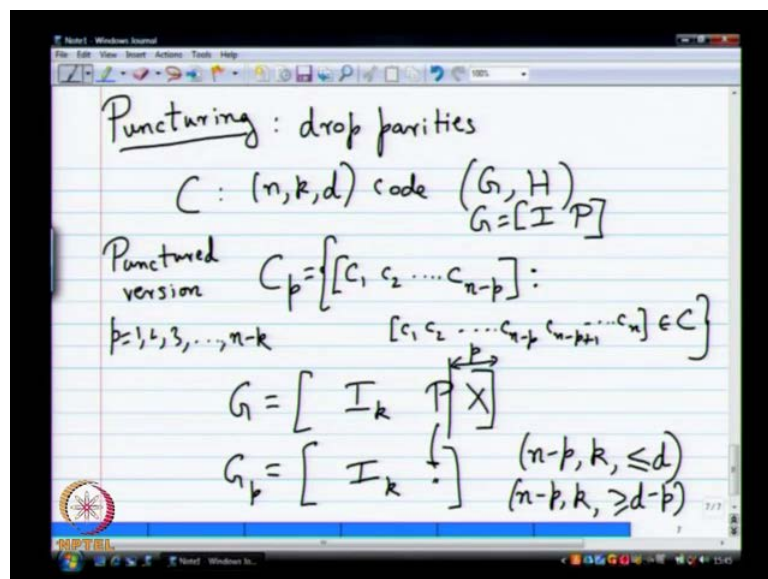
So, far for the encoder, we been using the generator matrix you can also use the parity check matrix for encoding nothing stops from doing that see you have a  $h$  equals let us say  $p$  transpose  $i$  columns of parity check matrix are associated with the bits of the code. Remember, what is the condition you want  $h$  times  $c$  transpose to be  $0$ , so if you have  $c$  here  $c$  plus  $1$  all the way to the  $c_n$ , this has to be equal to  $0$ . So, I can associate each of these bits with the columns of with the columns of the parity check matrix. If I have a valid code what should happen these columns will be multi plied by those bits on top and I have add them up, I should get  $0$ .

Remember, this is my message part, so once I send my message part, how will I compute my parity part the equation directly gives you the answer all you have to do is simply multiply the columns by those bits add them up. Whatever vector you end up with will simply become your parity right that is what the equation is right, so I can write this equation as  $c_1$  through  $c_n$  as simply being equal to that. So, you write that carefully  $p$  transpose times  $c_1$  through  $c_n$ , so that exactly what this equation means written in the different way.

So, I can load my message bits on the on top of  $p$  transpose multiply those columns add it all up I will get the vector at the end  $n$  minus a length vector that vector is exactly my parity vector. I have to rotate and send it out that what the equation means, now in shortening what I am doing my last test message bits are 0 which means the last  $s$  columns in  $p$  transpose simply drop out you do not have to worry about them at all.

So, this way of viewing the parity check matrix is also useful, now once again I think of parity check matrix even the minimum distance becomes a little bit easier to argue. Previously, I have to write down some contradiction proof and all that, but from parity check matrix it is that immediately cleared if you had  $d$  columns of  $h$  s adding to 0 what will happen same  $d$  columns in  $h$  will also add to 0. Clearly, there will be a problem that is another way of thinking about it same argument, but maybe when you start at it this is a easier thing to get to the answer, let us move on to the next part of it which is puncturing as we finished shortening.

(Refer Slide Time: 35:23)



We will see puncturing is a little bit easier to describe, so puncturing is basically idea is to draw parties this is very powerful idea it is used to a lot in practice, so you drop parities. So, let me just describe what happens so once again suppose  $c$  is an  $n$   $d$  code generator would be  $g$  etcetera parity check matrix  $x$ . So, the punctured version you once again need an integer  $p$  should be 1 2 3 and so on and  $cp$  basically  $p$ , so I am going to think of averting in systematic form  $g$  is going to be  $i$   $P$  and in my code the last  $n$  minus bits will be parity.

The first bits will be message, so when I draw parities all the it is going to be  $c_1 c_2$  so on till  $c_{n-p}$  such that  $c_1 c_2$  so on till  $c_n c_{n-p} c_{n-p+1}$  so on till  $c_n$  such that belongs to  $c$ . So, this is the punctured version, so in case you take all the code words of  $c$  simply drop the last  $p$  parity bits will remain will have a fill the same number of code words left. All do you know that the partition will happen I have to pick  $P$  carefully I cannot go beyond  $n$  minus.

So, I would not go beyond  $n$  minus I will make sure, so the first bits are not affected by the puncturing, so defiantly I would not lose code words if I do this because the first bits are messaged bits and not puncturing any of that. So, they will all be distinct in my two power codes so as long I put parties the number of code words will not go down. I will have the same number of code words. So, let quickly see what will happen to the generator matrix if you have a  $i$   $p$  what will happen to  $g$   $s$   $g$   $p$   $m$ , sorry so you will have the same  $i$ , nothing changes there what will happen to  $p$ .

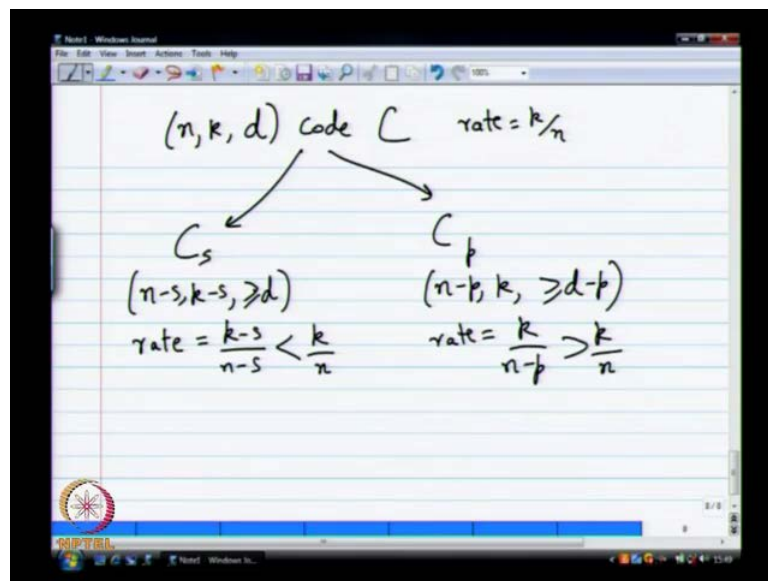
The last  $p$  columns will simply drop out puncturing is easy to describe what about  $n$   $b$  the three parameters for the punctured version this is going to be  $n$  minus  $p$  remains just show you why remains no problem. So, this is going to happen less than or equal to  $d$ , but when it is not good to just say less than or equal, so then it looks like I am suggesting it can go to 0. Then I can put greater than or equal to if it is greater than or equal to also you can also say  $n$  minus  $p$  greater than or equal to what can I say it will greater than or equal to what it is simple  $d$  has to be involved in the answer.

What will be greater than or equal to simply formula can I say  $d$  minus  $p$   $d$  minus  $p$  is what wrong with  $d$  minus  $p$  perfectly good enough expression why is  $d$  minus  $p$  so the intuit to argument in the original code. There will be lot of there will be some code word with minimum weight  $d$  when that gets punctured  $p$  bits are dropped worse case that can

happen to you is all  $p$  got punctured all  $p$  are in one last  $p$  bits in that case it will be  $d$  minus  $p$  many other case you have another advantage. So, you can only cannot be worse than this greater than equal to  $d$  minus.

So, I will leave it as a exercise to see what happens to the parity check matrix and puncturing it will be a little bit more complicated, it will be like what happens to the generator matrix. See when you in the generator matrix for shortening what happens in the generator matrix for shortening once a little bit more complicated, right? Nothing gets just right same thing will happen here and in puncturing in the parity check matrix should be a little more confusing, but I want to quickly point out one thing so what does shortening do to the rate of the code what do shortening and puncturing to the rate of the code.

(Refer Slide Time: 41:01)

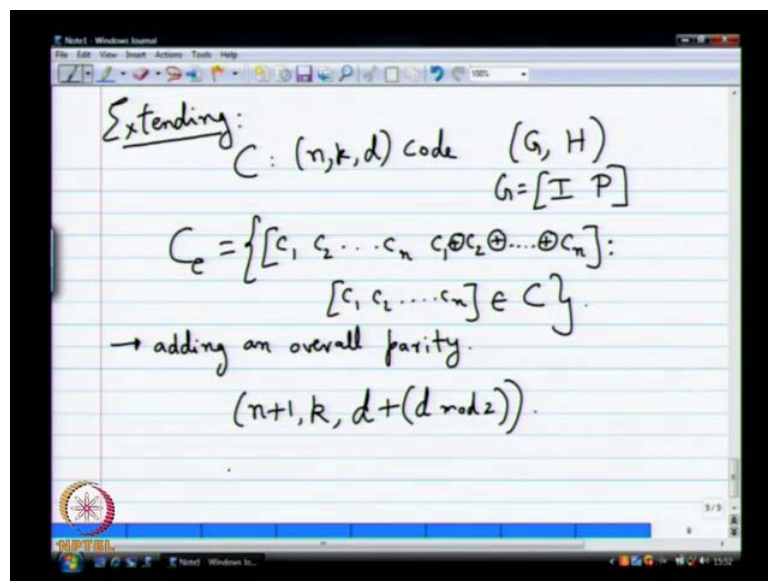


So, you have an  $n$   $d$  code suppose you do two things you go to  $c$   $s$  which is shorten to  $n$  bits or you go to  $c$   $p$  which is punctured by  $p$  bits. So, what you get here is  $n$  minus  $s$  greater than or equal to  $d$  what you get here is  $n$  minus  $p$  then let us say greater than or equal to  $d$  minus  $p$  I will say that  $\cos 2$ . So, what happens to the rate here the rate here is by  $n$  rate here becomes minus  $s$  by  $n$  minus  $s$  for a equals 1 2 3 and all what how will this compare with by  $n$  will be smaller this is smaller than by  $n$ . You can show that is very easy just cross multiply see say some  $n$  will cancel if  $n$  is greater than, so always be lesser.

So, that is the condition so when you shorten the rate goes down and the minimum distance can possibly increase when you puncture what happens to that rate goes up  $n$  minus  $p$  is greater than by  $n$ , but the minimum distance can possibly come down. So, this is what will happen, so puncturing is always used for any instance any standard communication standard that you pick up that will be puncturing they will always design a code with lower rate and puncture it to get higher.

What could be the advantage of doing that instead of designing two different codes one at some rate at the rate half another at a rate 1 by 3. Then so decoding might become easier cannot just decoder just run for all of these codes these things are advantageous from a design point of view. If it is not so important then you might want to design two different codes if you design two different codes you might be slightly better off, but you are not considering is soft by doing puncturing. Softening that is the idea so the last thing there are about six minutes, so let me do the last thing which is extending it is not too difficult, so easier thing.

(Refer Slide Time: 44:03)



So, once again you have a  $C$  which is an  $n$   $d$  code generator  $G$  parity check matrix  $H$ , once again I am thinking about of this as systematic form these are quite standard when you extend you do  $c_e$ ,  $e$  is not an integer or anything. It is just an extension denoting as a  $c_e$  what this has is  $c_1, c_2$  so on till  $c_n$  and then you have  $c_1$  x or  $c_2$  x or so on till in

writing  $x$  or because I want to drive home the point is just one bit for all  $c_1, c_2, c_3, \dots, c_n$  belonging to  $c$  this is the idea so extending.

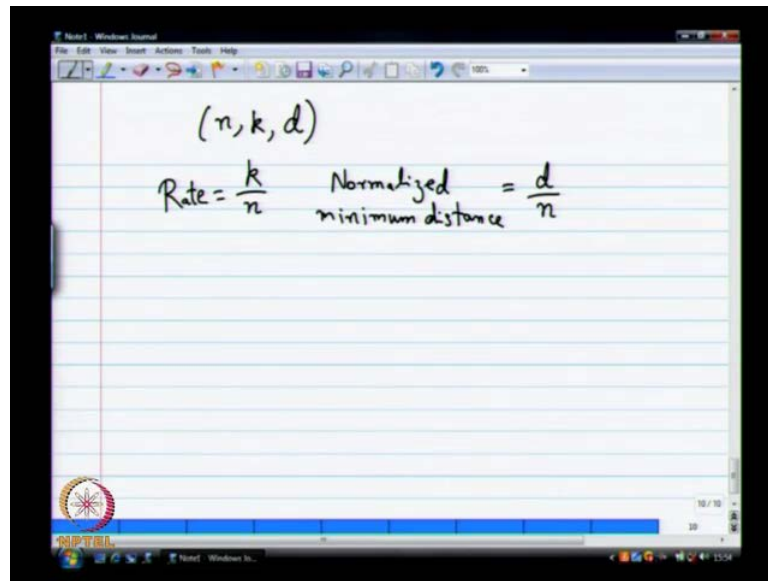
Basically, the act of adding an overall parity that is idea and extent, so the way to visualize this is you have the list of all code words listed down one below the other. What you will do you go to each column compute the overall parity and add an extra bit and put the disturbance. You do that you get this, so this is the idea, so the parameters let me quickly write down the parameters original code is  $n$   $d$  the parameters will be what  $n$  plus 1. That means they are adding only a parity, I am not adding any new message, it is just going to be parity added just, then what  $d$  or  $d$  plus 1 when will it be  $d$  when will be  $d$  plus 1 depends on  $d$  is even or odd  $d$  is even then what will happen.

Then, it will remain by itself so there is no point in extending a code with  $d$  even, so that seems like a bad thing to do. So, one way to write it will be to say  $d$  plus  $d$  percentage  $2$   $d \bmod 2$ , so  $d$  is even I get the  $d$  if  $d$  is odd minimum distance goes up by 1. The loss in rate is negligible is by  $n$  plus 1 and if  $n$  are fairly large just do not lose anything not losing much we have increased the minimum distance by 1 to from  $r$  to even. So, this is extending and very interesting exercise which I urged to be a strongly tied get the generator and a parity check matrix for the extended code.

So, a little bit non trivial but do some work it is not very not trivial but anyway it is a good thing to try you will get a good idea of what this parity check matrix and generated matrix look like. So, it is political think about that is an important exercise, I urge you to try it will get a clearer idea of what happens we saw a few techniques. Now, to see shortening puncturing and extending basic idea like I said was to get you familiar with what the parity check matrix is what means how do you manipulate etcetera hopefully.



(Refer Slide Time: 48:00)


$$(n, k, d)$$
$$\text{Rate} = \frac{k}{n} \quad \text{Normalized minimum distance} = \frac{d}{n}$$

Following which you can imagine which is not too difficult but still there are two conflicting things here. One is  $k/n$  which is by rate, another thing is basically minimum distance, but I will normalize it just to make sure I can compare it with rate normalized minimum distance which is  $d/n$ . So, these will kind of compete with each other. I want to increase both right in some sense like I said minimum distance is connected to their connecting capability. If I have more minimum distance, I can correct more errors, so I want to have more  $d/n$ . Why do I have more by  $n$ ? Yeah, it is giving very obvious answers.

Then, I send more information to a second some level, so you can get that so I want to get the by  $n$ . You can imagine there will be some bounds you cannot keep on increasing by  $n$  for a 6, 10 you cannot keep on increasing and  $d$  eventually something will stop of there will be some relationship. There are some bounds relating these quantities which is what we will see in the next lecture, we will see how to relate these two things that is very simple ideas. Our intuition will mostly be true by  $n$  goes up  $d/n$  will go down we will see here for the, now I will pick from here in the next lecture.