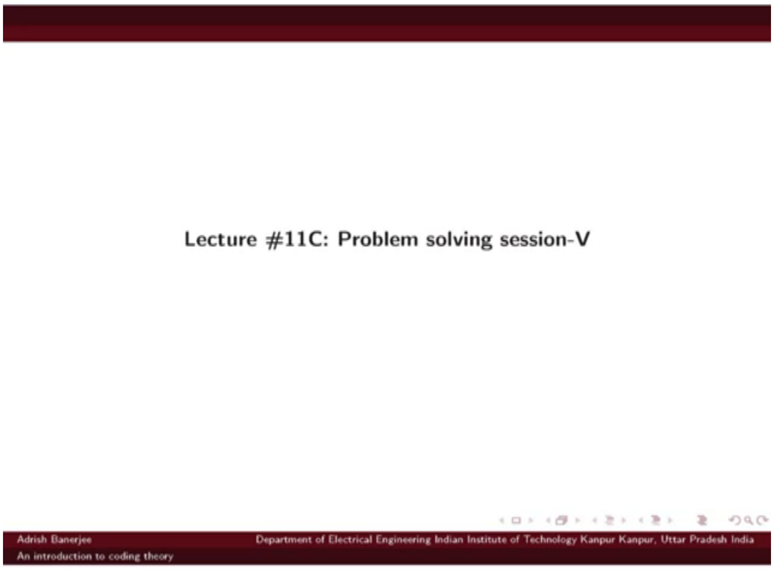
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 05
Lecture Number 22
Problem Solving Session-V

(Refer Slide Time 00:13)

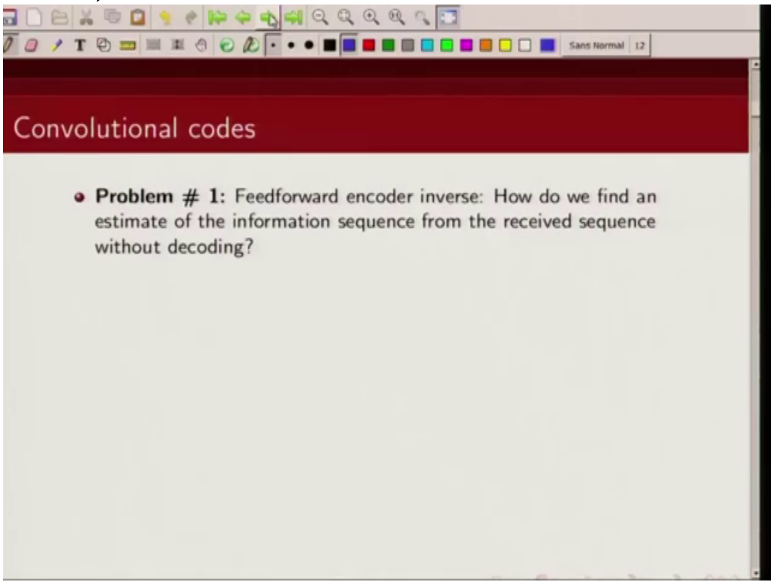


So today we are going to continue with some more

(Refer Slide Time 00:18)

problems related to convolutional code. So let us solve some codes then we will move to our other topic. So first

(Refer Slide Time 00:26)

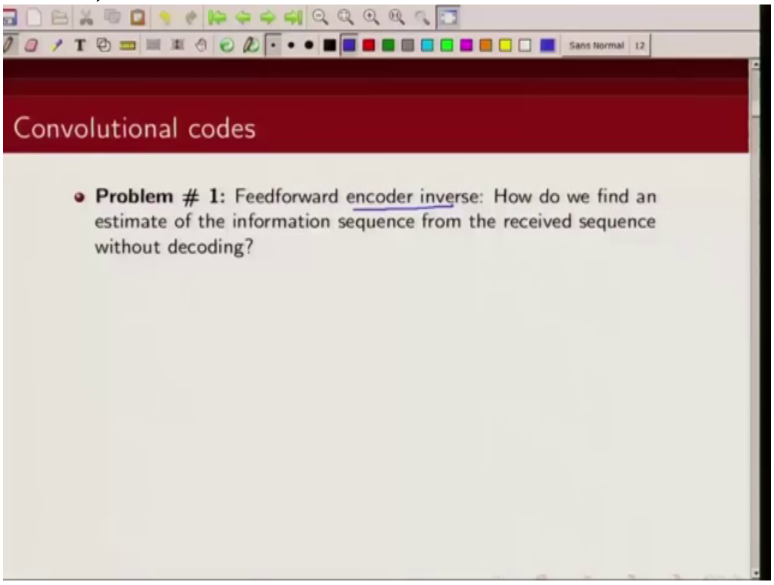


question is on feed forward encoder inverse. So what is encoder inverse, we will talk in a minute. So many a times, we are interested in estimating the information sequence directly from the received sequence without decoding it. So for example if you are encoding a sequence using systematic encoder, then you can directly from the received bits, you can get back your information bits. However if you are using a non-systematic encoder then you cannot directly get the information bits. So we are talking about encoder inverse which will allow us to

(Refer Slide Time 01:13)

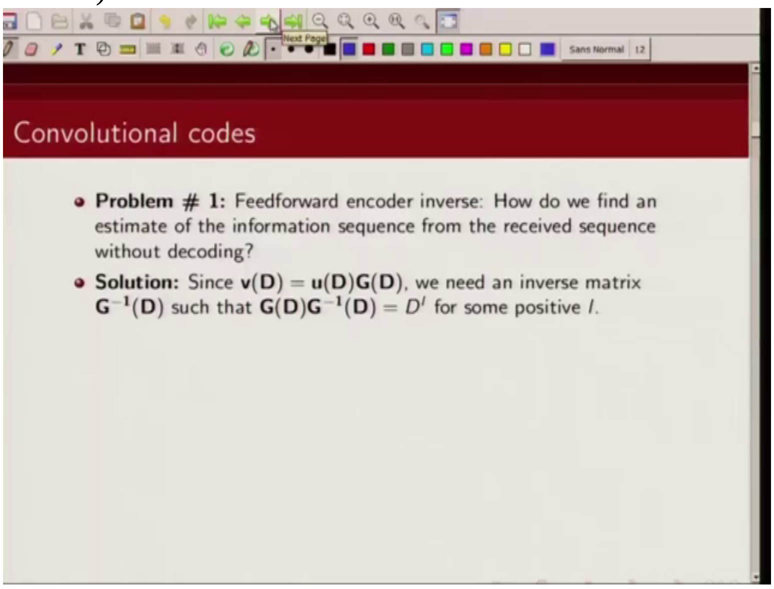
recover back the information bits directly without decoding.

(Refer Slide Time 01:19)



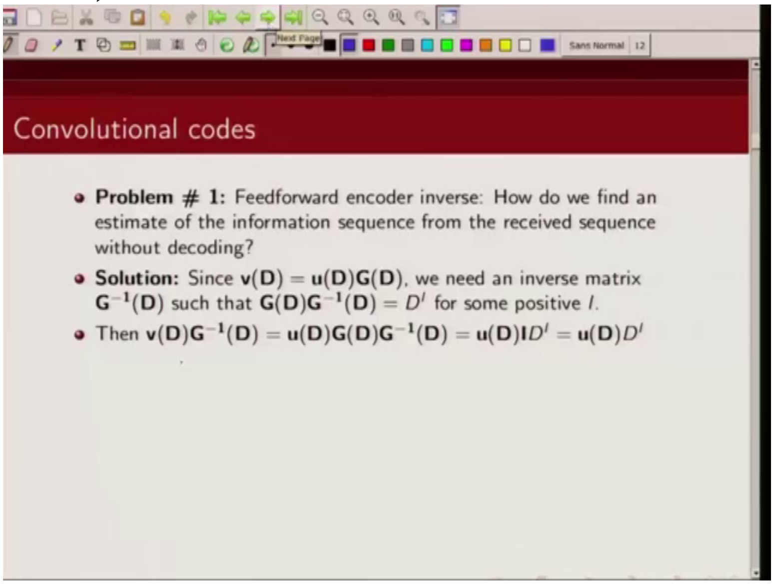
So in this problem we will look into what is an encoder inverse and under what condition the encoder inverse exists. So

(Refer Slide Time 01:32)



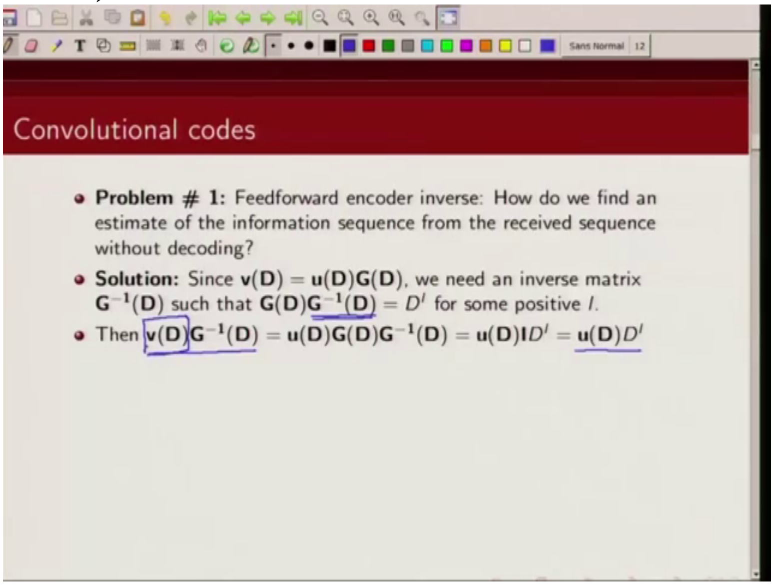
as we know that our coded bits can be written as our information bits times this generator matrix, encoding matrix and the problem that we are looking at is finding out the encoder inverse and we will talk about whether a feed forward inverse for this encoding matrix exists or not and under what condition it exists. So if there exists a feed forward inverse, then if we, from the received sequence

(Refer Slide Time 02:10)



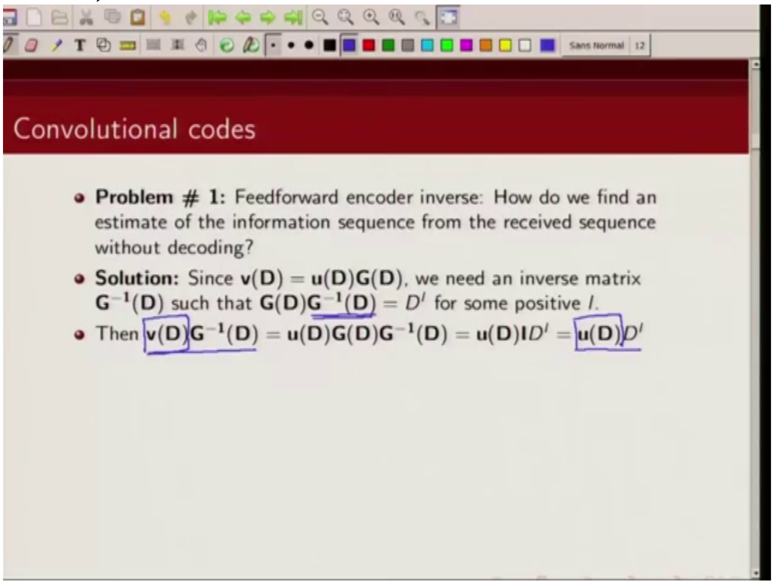
if we just multiply by the encoder inverse we can get back our original information bits without decoding after some delay. So this D l is some delay, D l. So what we are saying is we are interested in finding this encoder inverse. Does this encoder inverse exists? Feed forward encoder inverse, does that exist such that G D, G inverse is some delay element and what's the use of this? So if you have your information sequence v D,

(Refer Slide Time 02:50)



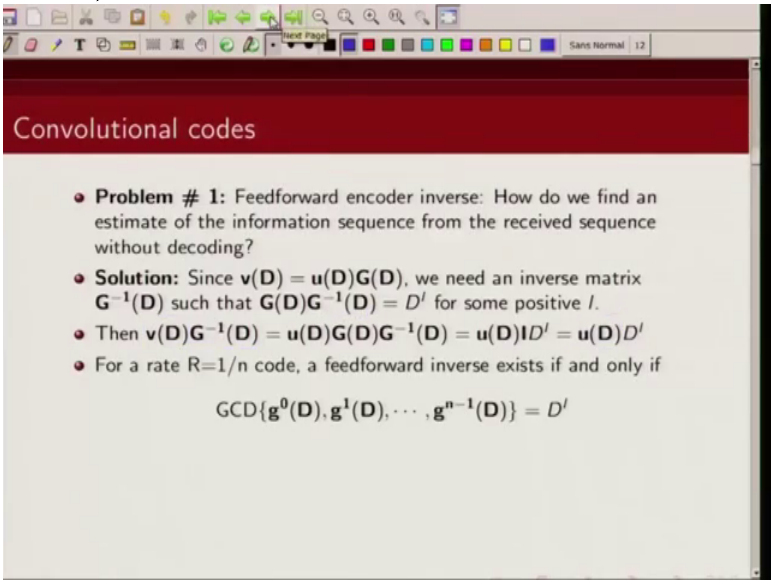
if it passes through this encoder inverse circuit, we can directly get back our information sequence.

(Refer Slide Time 02:59)



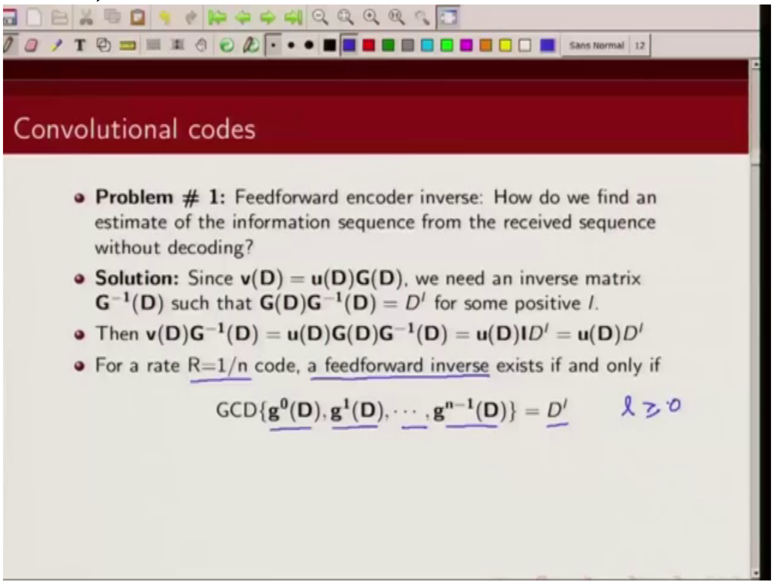
And in many cases, for example if the channel conditions are good you may directly want to first guess or check whether the information bits are directly, estimate information bits so you may want to pass it through this encoder inverse circuit.

(Refer Slide Time 03:18)



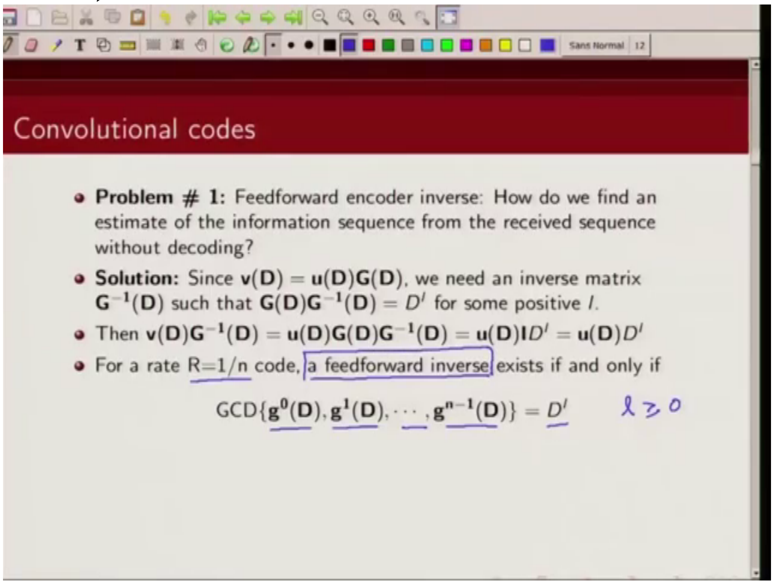
So I am now stating without proof the conditions under which these encoder inverse exist, a feed forward encoder inverse exist. So for a rate 1 by n code, a feed forward inverse will exist if the greatest common divisor between these n generator sequences of this rate 1 by n code, if the greatest common divisor among these generators is some delay element, this l is something which is greater than equal to 0. So

(Refer Slide Time 04:00)



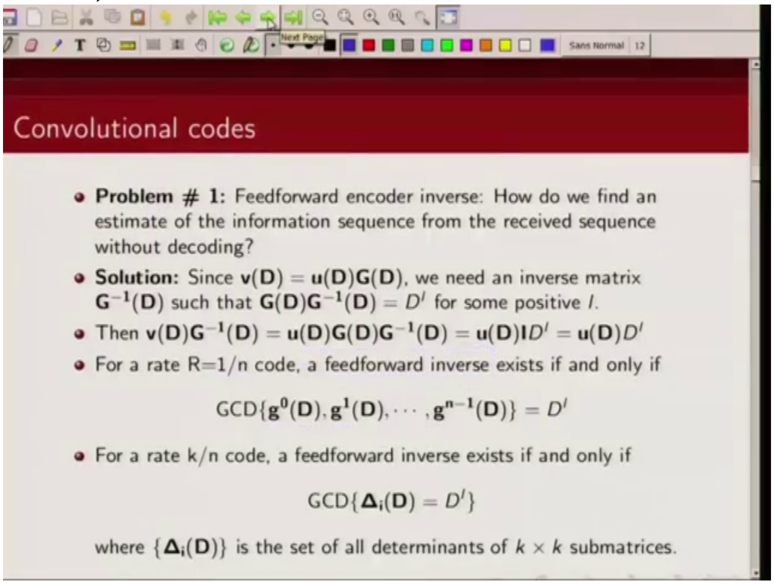
they don't have any term common in them, just some D times, basically some delay element. So we don't want these generator sequences to have any term common between them. If they have any term common between them, then a feed forward inverse would not exist.

(Refer Slide Time 04:22)



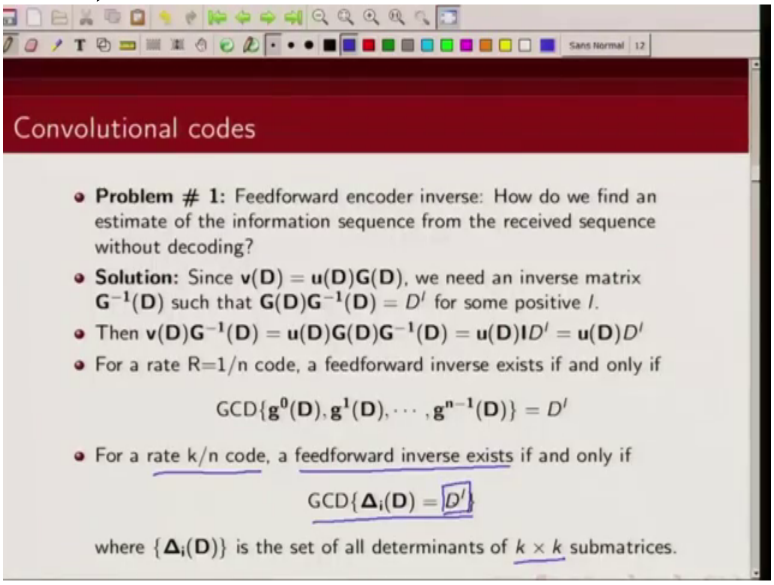
Then there would be a feedback inverse. Similarly

(Refer Slide Time 04:28)



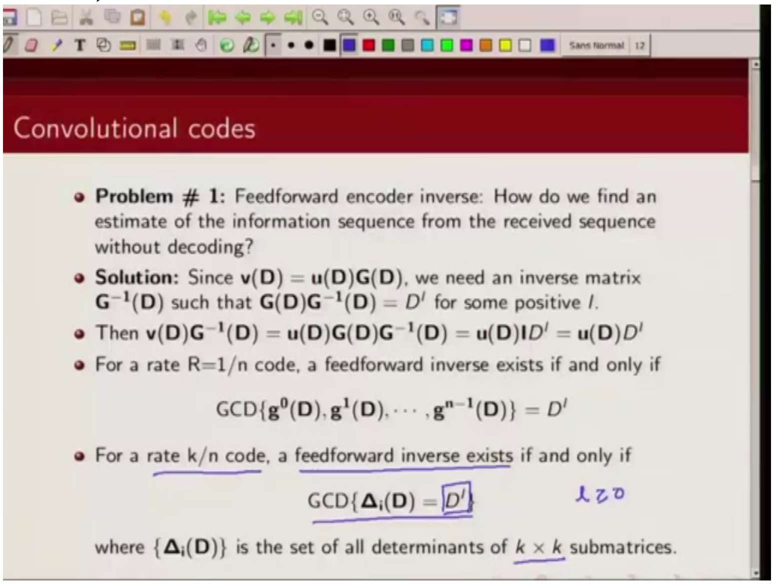
for a rate k by n code a feed forward inverse will exist if and only if the greatest common divisor, if we look at set of all determinants of k cross k sub matrices of this generator matrix, then the g c d of this set of determinants should be again from D to power l

(Refer Slide Time 04:58)



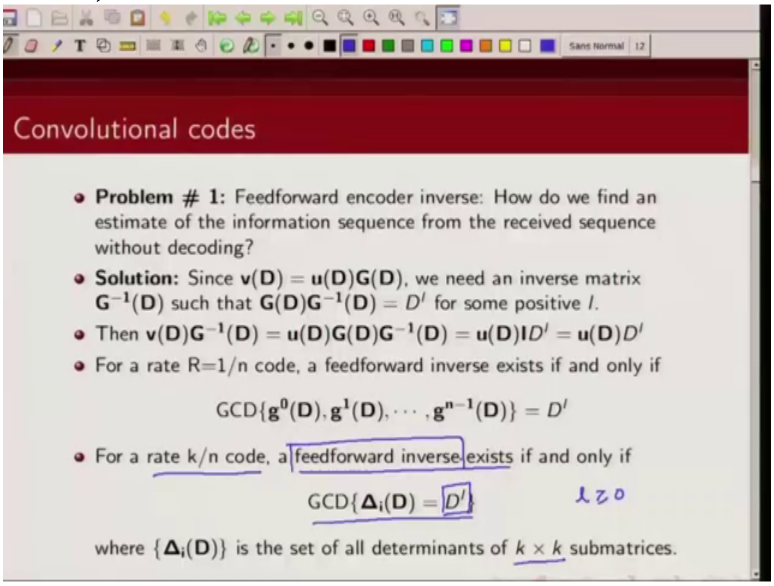
where l is a positive number. So we don't want

(Refer Slide Time 05:03)



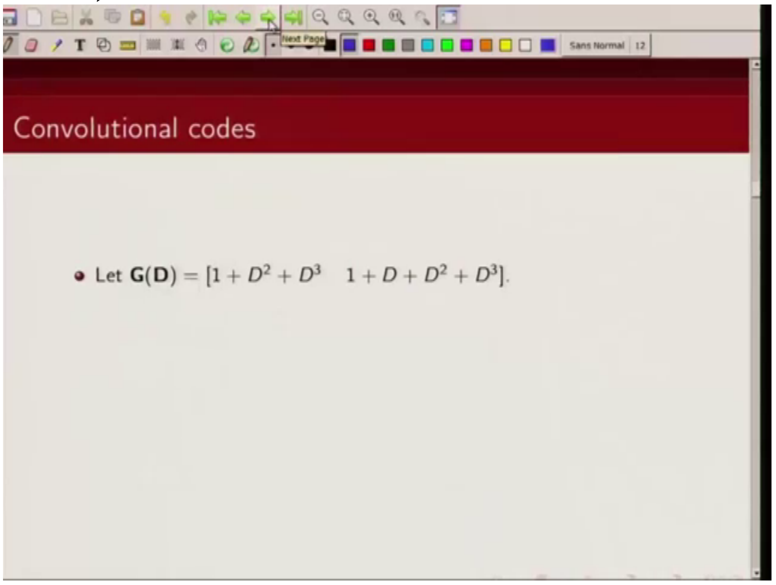
the determinants of this k cross l, all possible k cross k sub matrices to have any common term among them. If this condition is satisfied a feed forward inverse exists.

(Refer Slide Time 05:20)



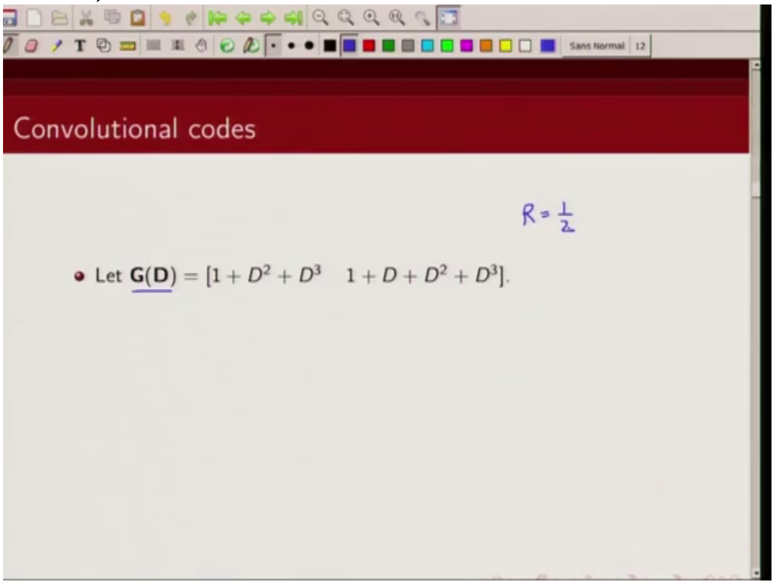
. So let us

(Refer Slide Time 05:22)



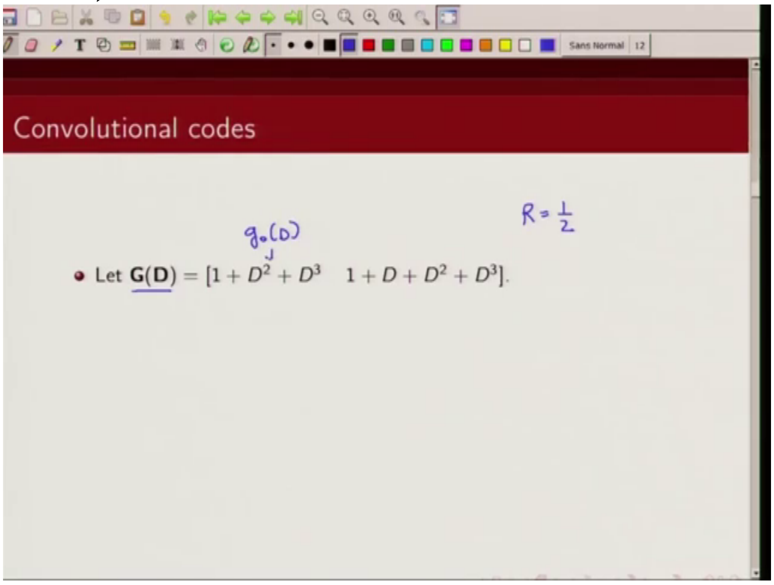
take an example where feed forward inverse exists. So we are considering a feed forward rate 1 by 2. This is a rate 1 by 2 encoder.

(Refer Slide Time 05:37)



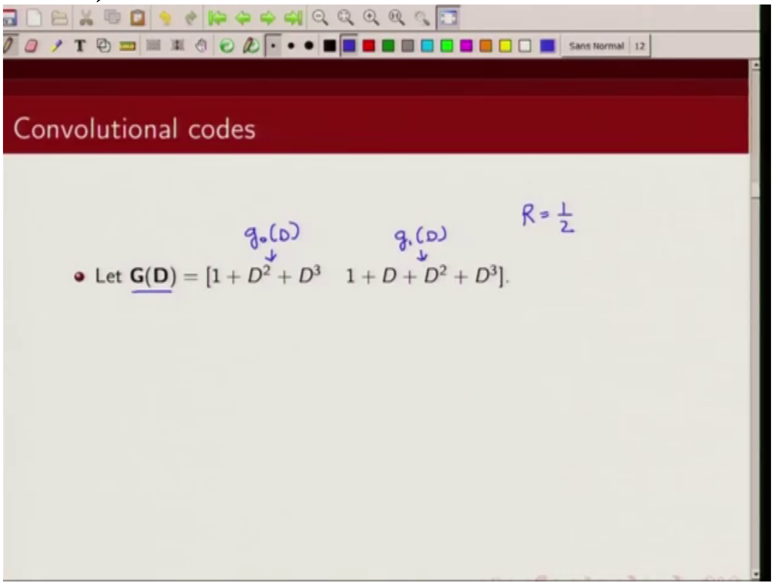
So g 0 D is this one

(Refer Slide Time 05:42)



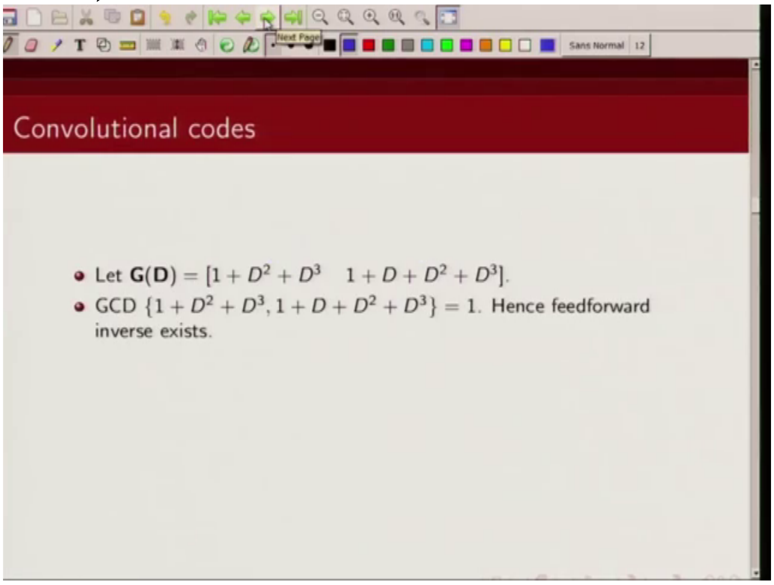
and g 1 D is this

(Refer Slide Time 05:46)



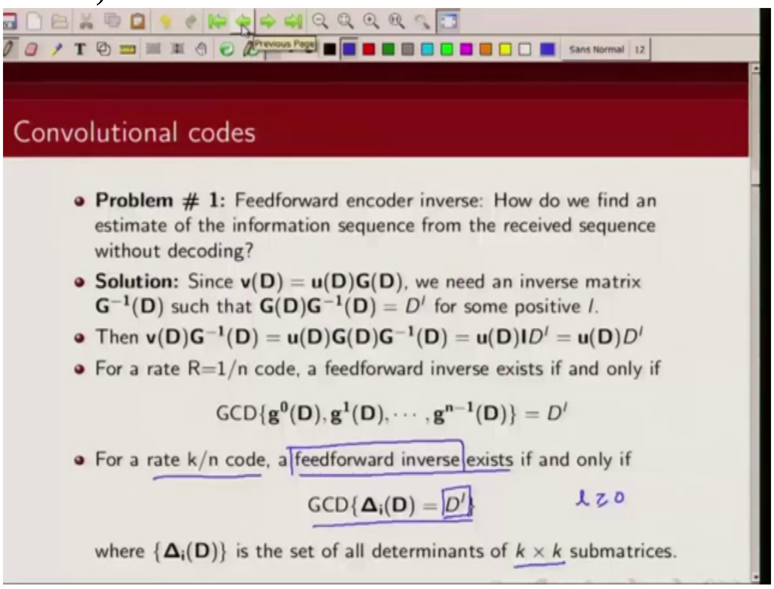
and what is the common

(Refer Slide Time 05:49)



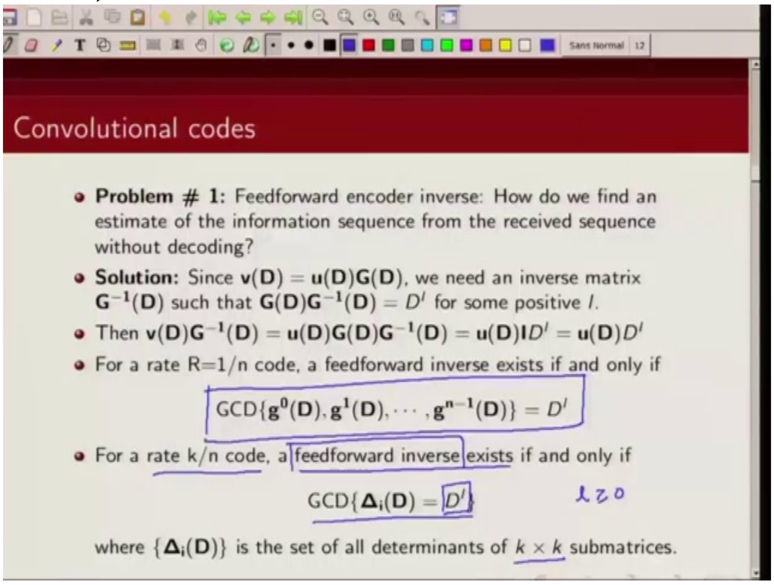
divisor between them? We can check basically they don't have any common terms. So the greatest common divisor is 1. So if we go back

(Refer Slide Time 06:01)



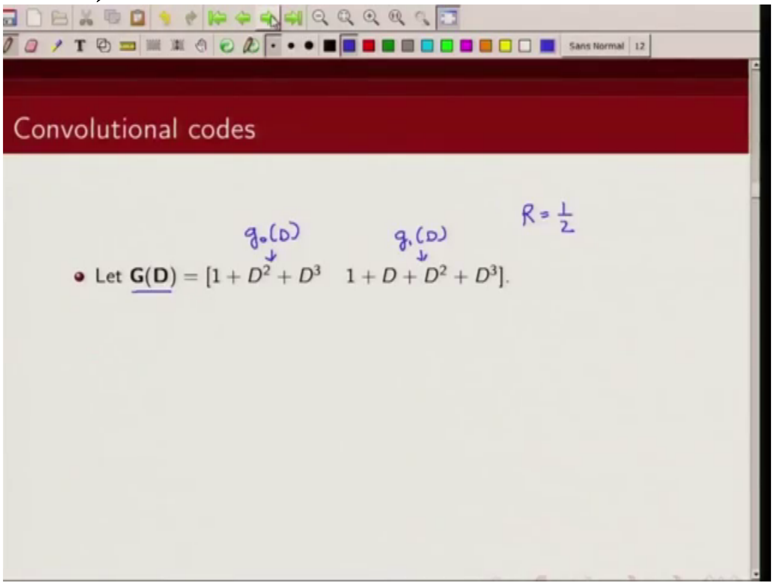
and look at our condition for

(Refer Slide Time 06:08)



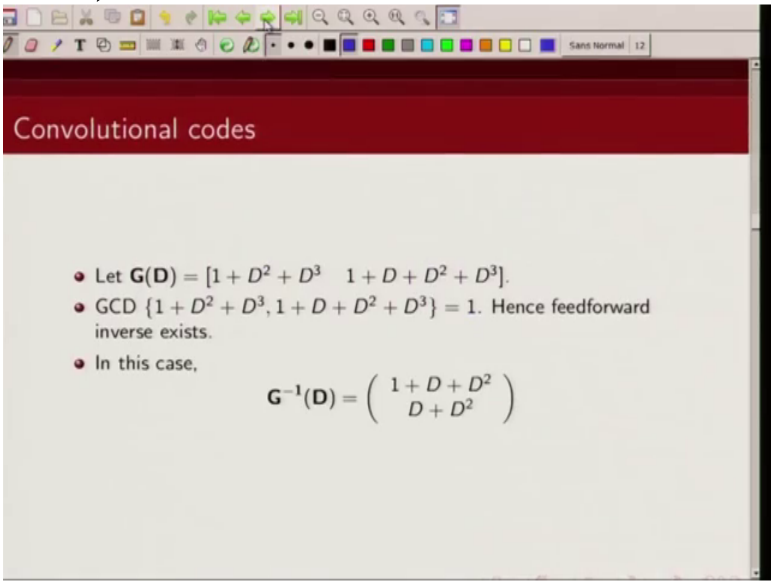
encoder inverse to exist, this condition is satisfied. So for this particular

(Refer Slide Time 06:14)



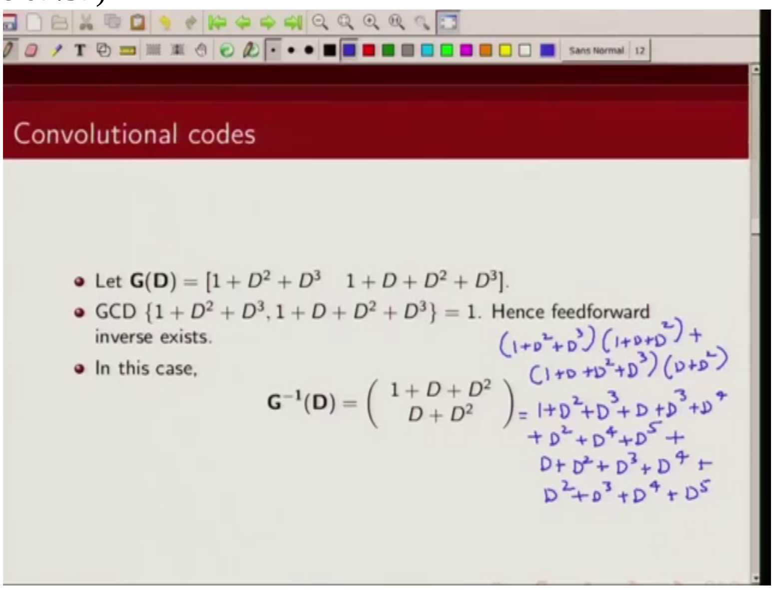
code with G D given by this will have a feed forward

(Refer Slide Time 06:19)



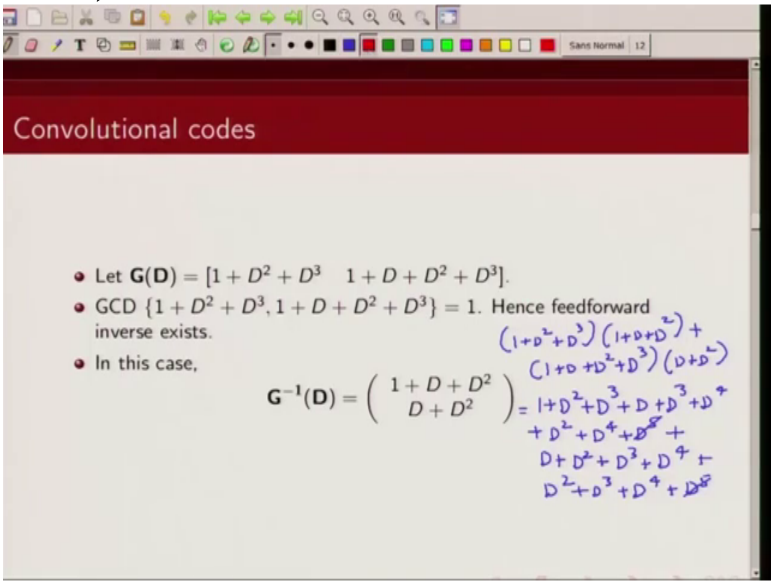
encoder inverse and in this particular case the feed forward inverse is given by this, Ok. So you can check G D, G D inverse will be 1. So you can just do a simple check 1 plus D square plus D cube into 1 plus D plus D square plus 1 plus D plus D square plus D cube into D plus D square. This is, so this is 1 plus D square plus D cube plus D times D cube plus D four plus D square times D four plus D five then multiply this with this, we get plus D times D square plus D cube plus D four plus D square plus D cube plus D four plus D five, Ok and let's see.

(Refer Slide Time 07:37)



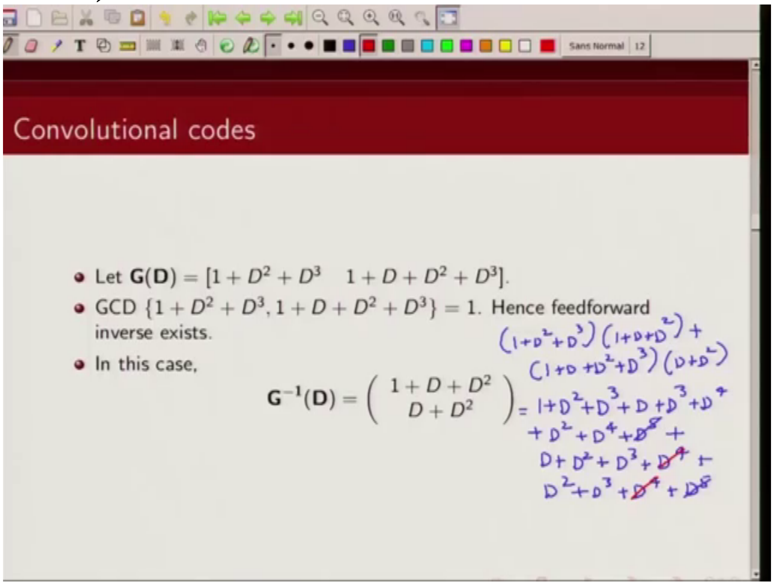
So D five, D five cancels out; D four,

(Refer Slide Time 07:42)



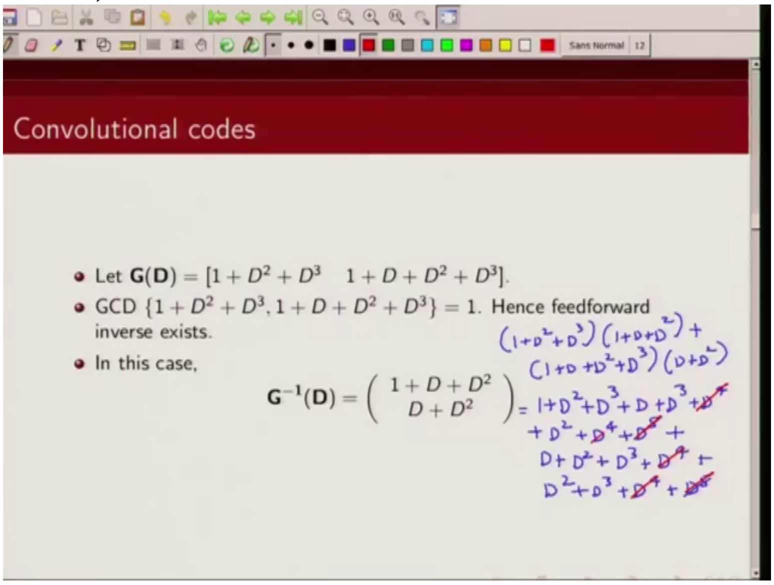
D four cancels

(Refer Slide Time 07:43)



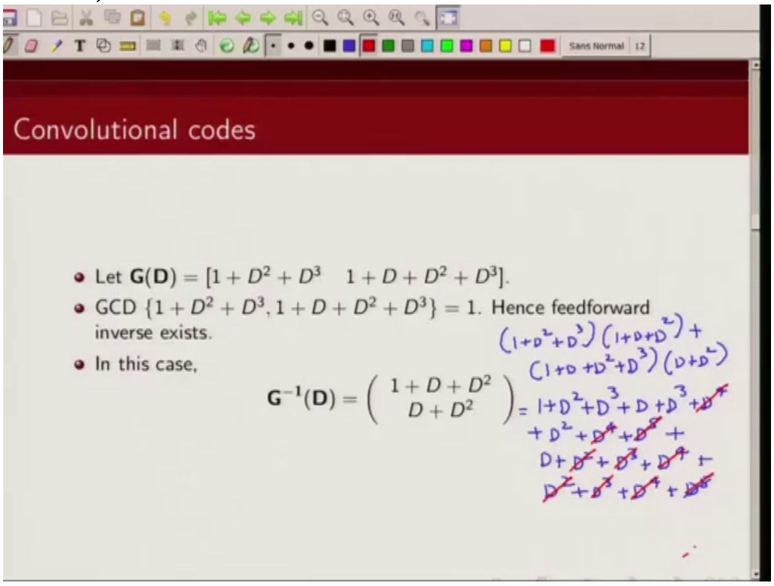
out then this D four, D four cancels

(Refer Slide Time 07:49)



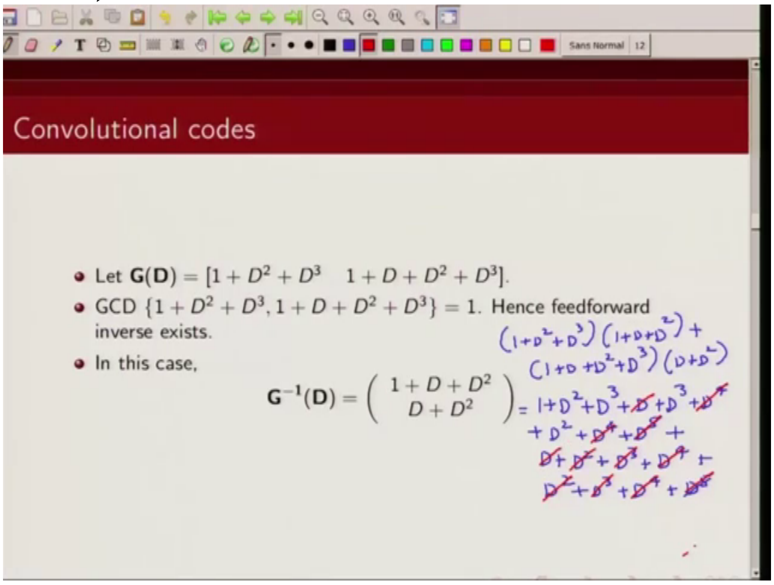
out; D three, D three cancels out; D two, D two cancels out;

(Refer Slide Time 07:55)



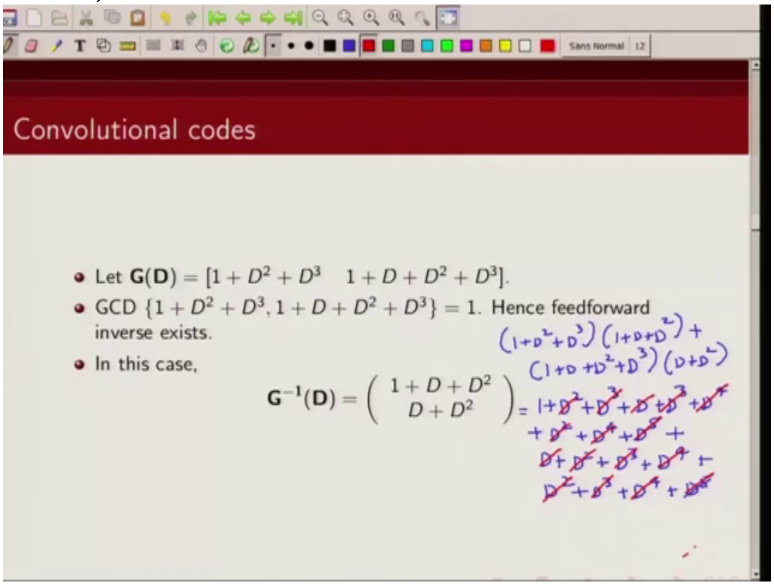
D, D cancels out;

(Refer Slide Time 07:58)



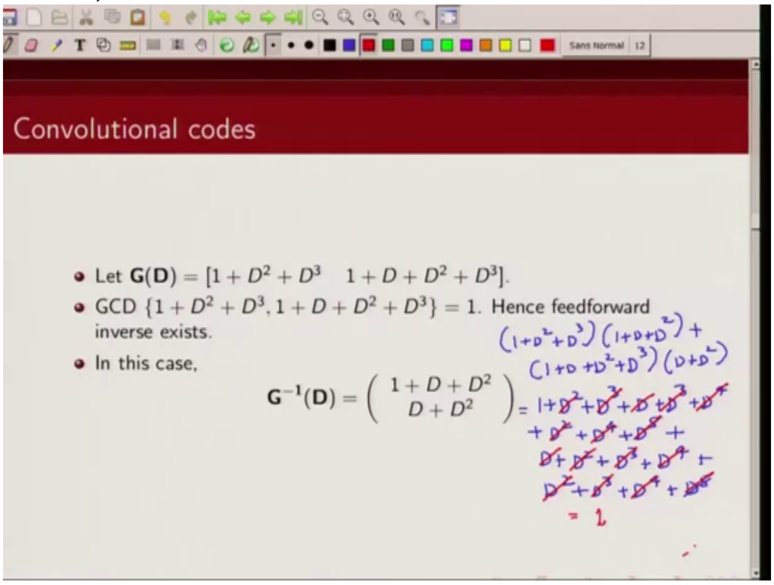
D three, D three cancels out; D two, D two cancels out;

(Refer Slide Time 08:02)



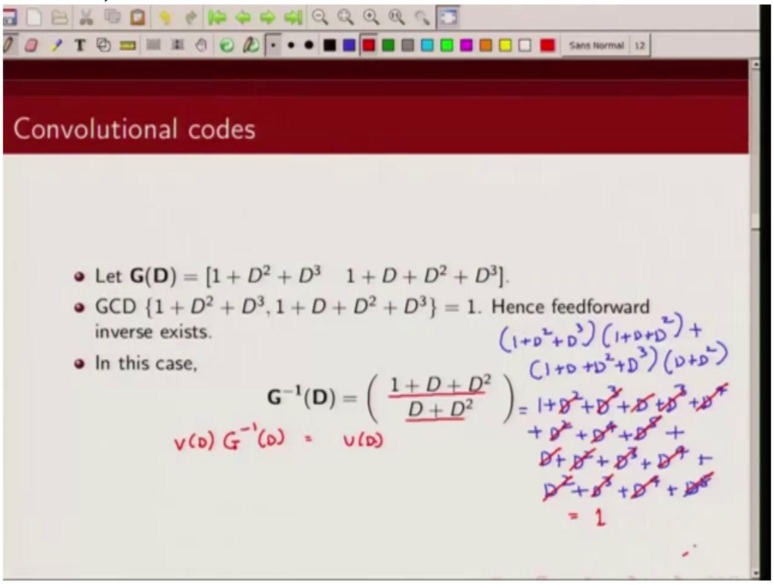
so what we are left with is basically 1,

(Refer Slide Time 08:04)

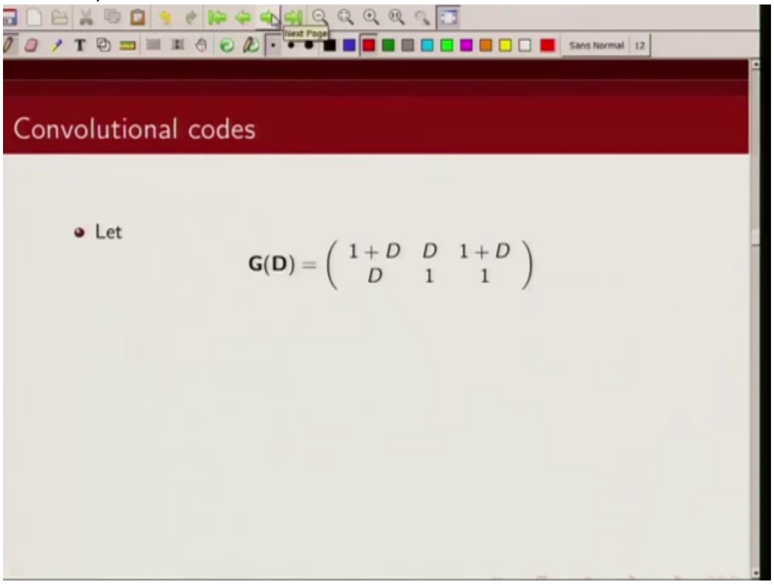


Ok so and you can see this is a feed forward inverse. So if you have your v D and you have passed through, this, this thing what you will get is get back your information sequence, Ok, get back information sequence.

(Refer Slide Time 08:27)

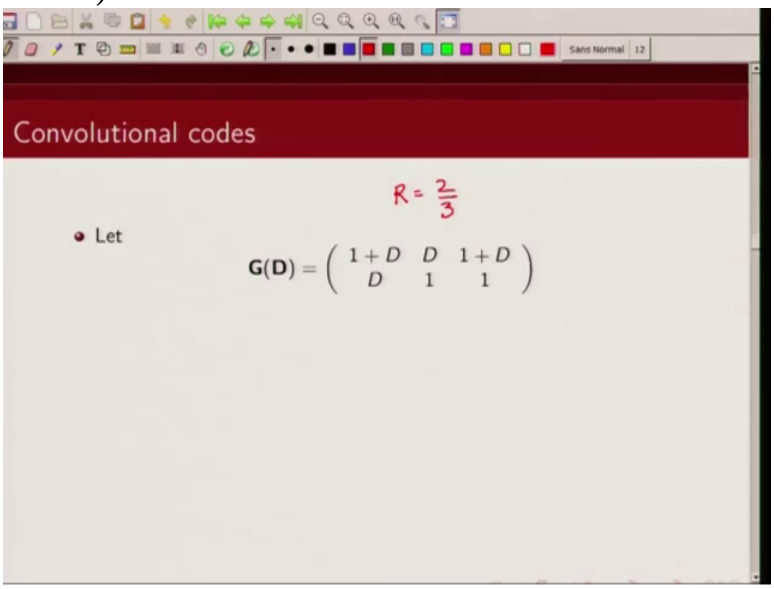


(Refer Slide Time 08:28)



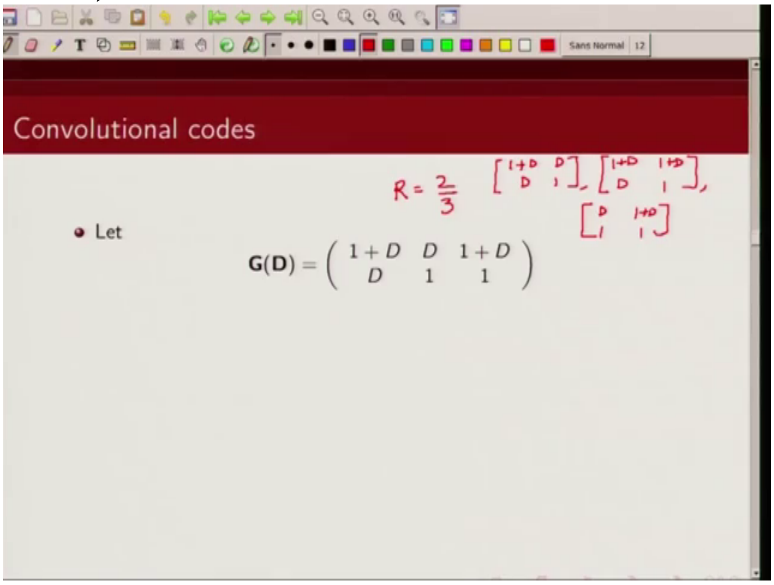
Now let us look for example for rate r equal to 2 by 3.

(Refer Slide Time 08:36)



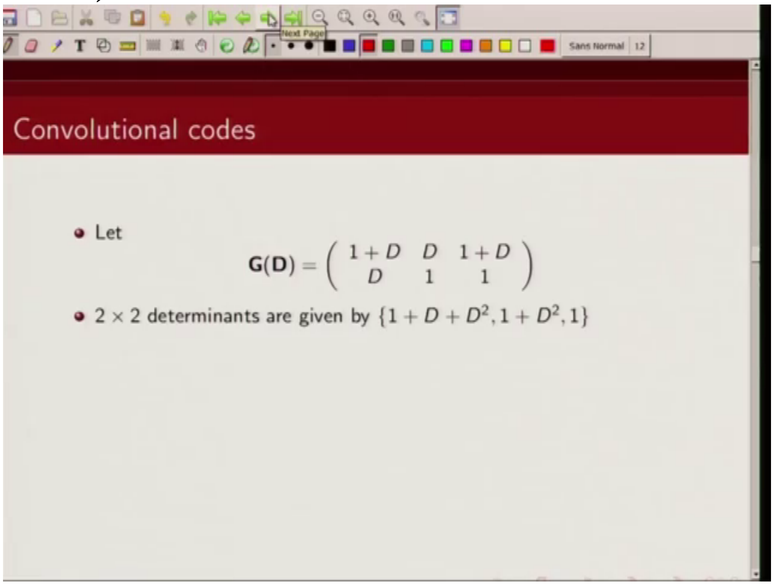
So in this case we first have to find the determinant of all 2 cross 2 sub matrices. So what are those 2 cross 2 sub matrices? One of them is this, 1 plus D D D1 next one is 1 plus D D, l plus D 1, and the third one is D 1, 1 plus D, 1.

(Refer Slide Time 09:04)



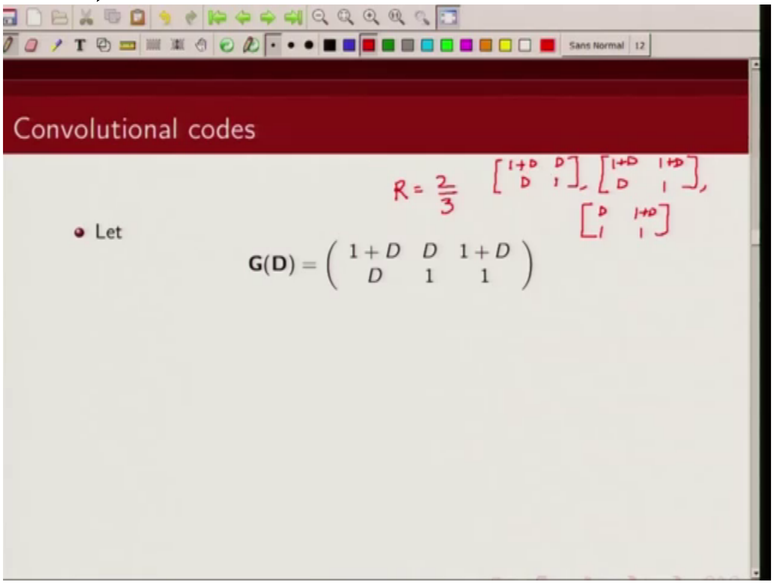
So these are the three 2 cross 2 sub matrices

(Refer Slide Time 09:09)



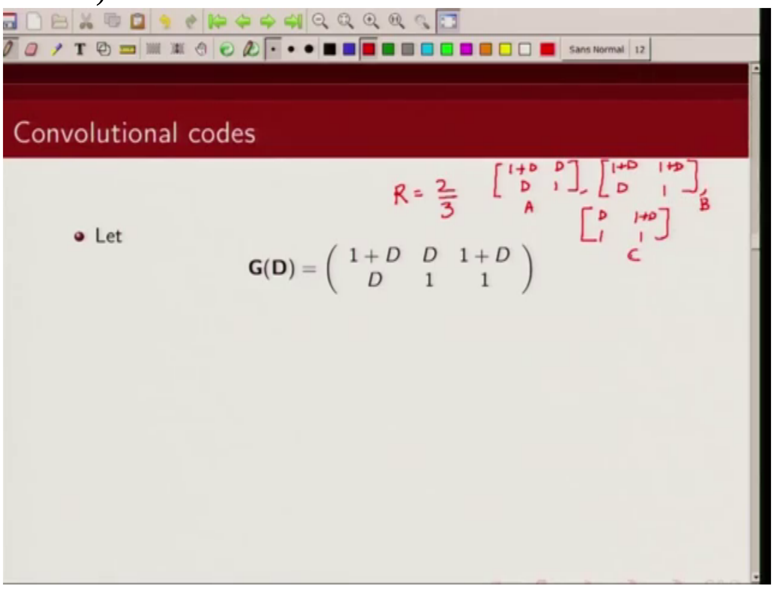
and we can find out the determinant

(Refer Slide Time 09:13)



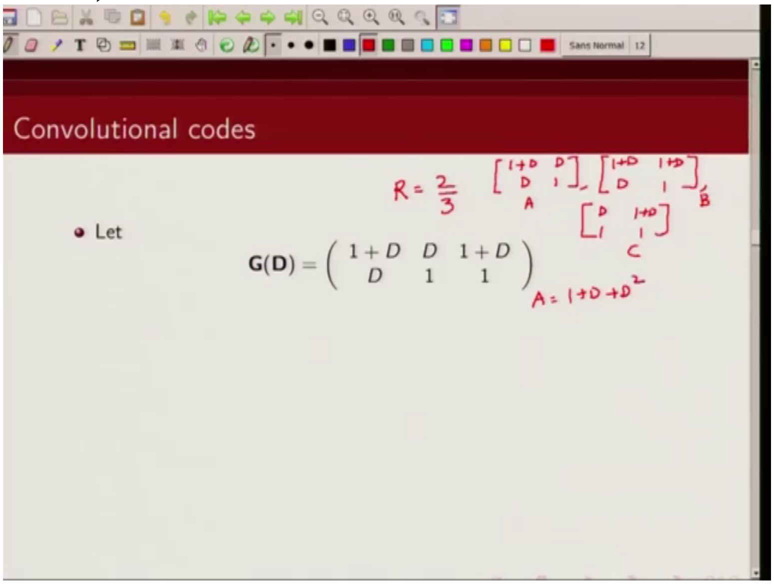
in this case. In this case let's call it A, B and C.

(Refer Slide Time 09:19)



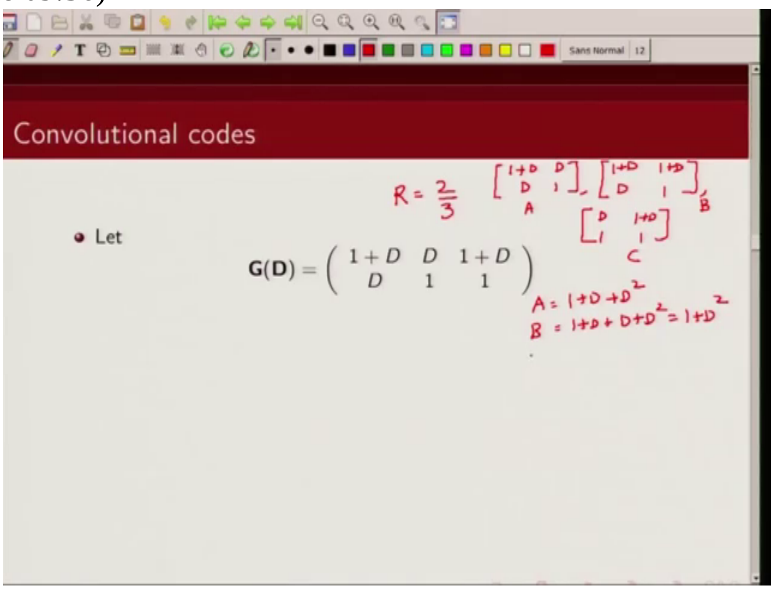
In case of A, the determinant is 1 plus D plus D square in

(Refer Slide Time 09:26)



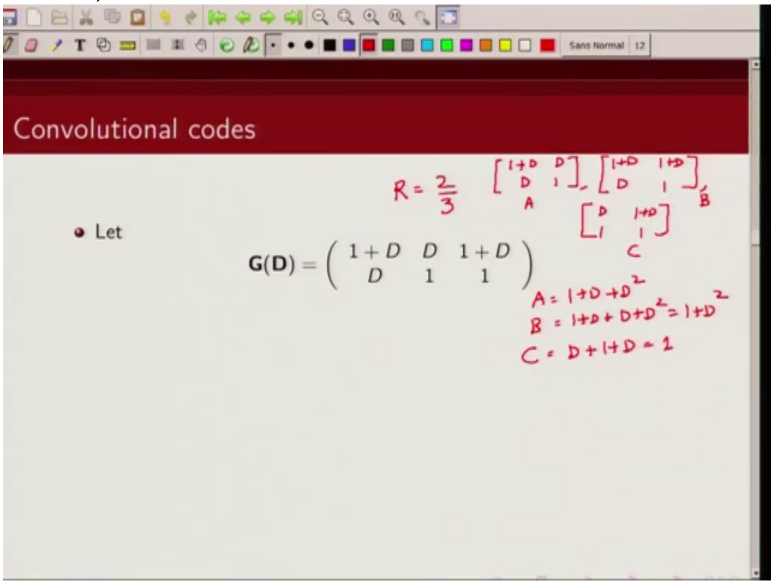
case of B the determinant is 1 plus D plus D plus D square so that's 1 plus d square. And

(Refer Slide Time 09:36)



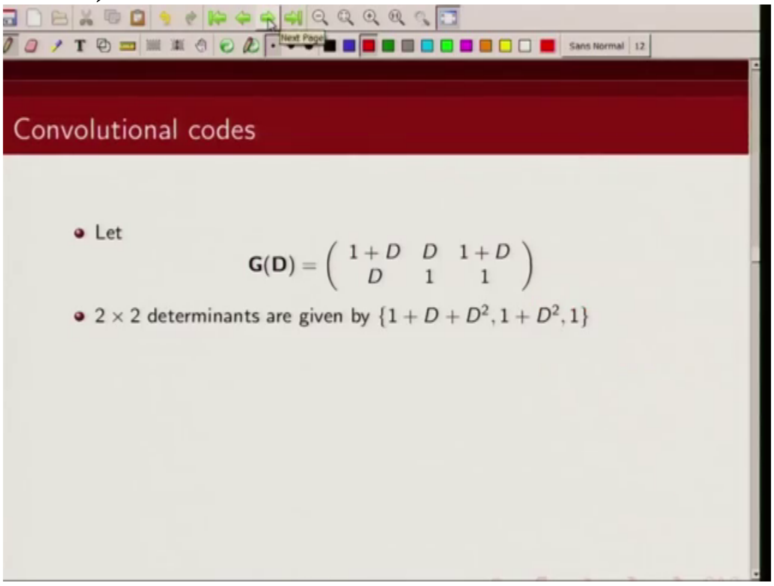
C is D plus 1 plus D, so that's 1.

(Refer Slide Time 09:42)



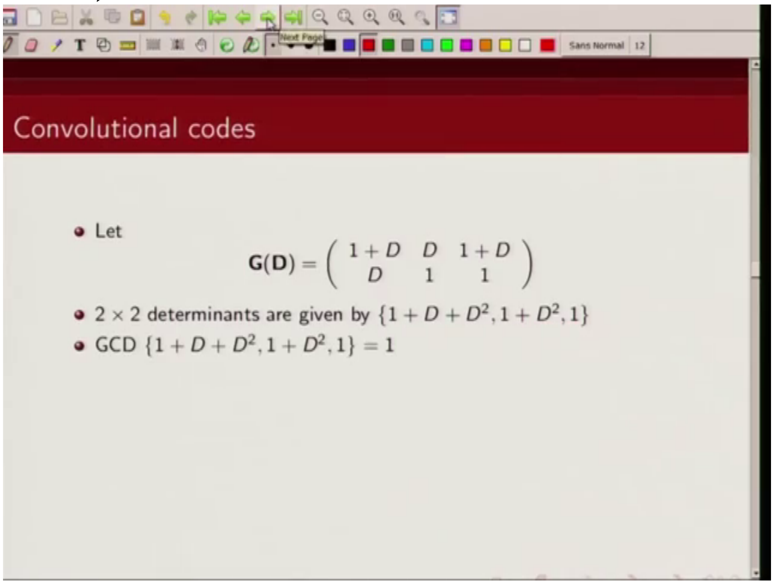
So these are the determinants of these 2 cross 2 sub matrices. And that's what

(Refer Slide Time 09:52)



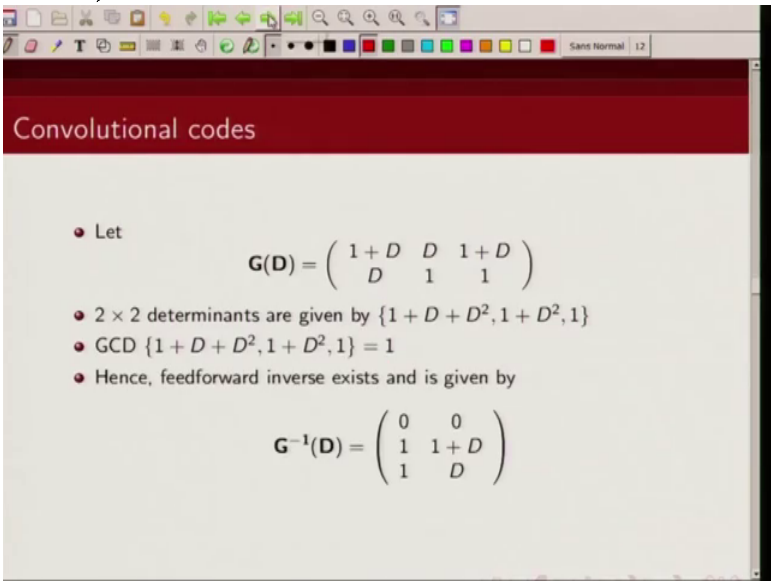
I have listed here, 1 plus D plus D square, 1 plus D square and 1. Now we need to check what is the greatest common divisor among them. And in this case

(Refer Slide Time 10:07)



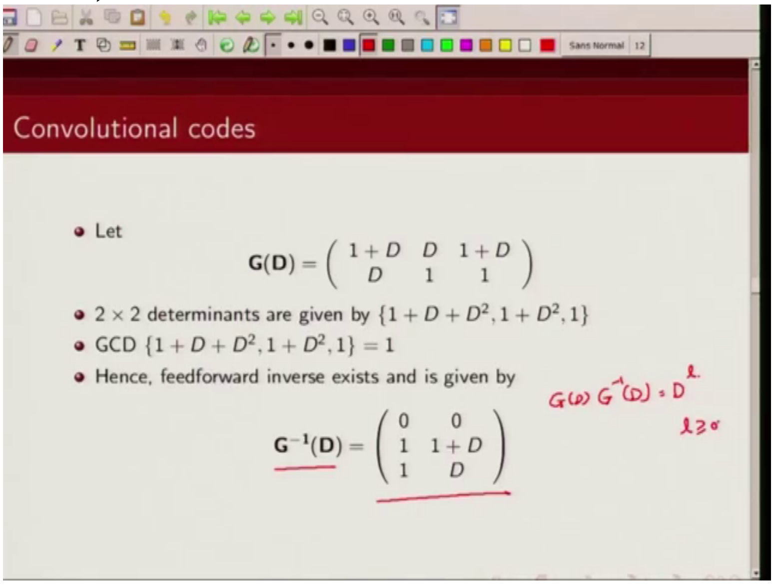
the greatest common divisor is again 1. So they don't have determinants of these 2 cross 2 sub matrices do not have any term common among them. So in this case also

(Refer Slide Time 10:25)



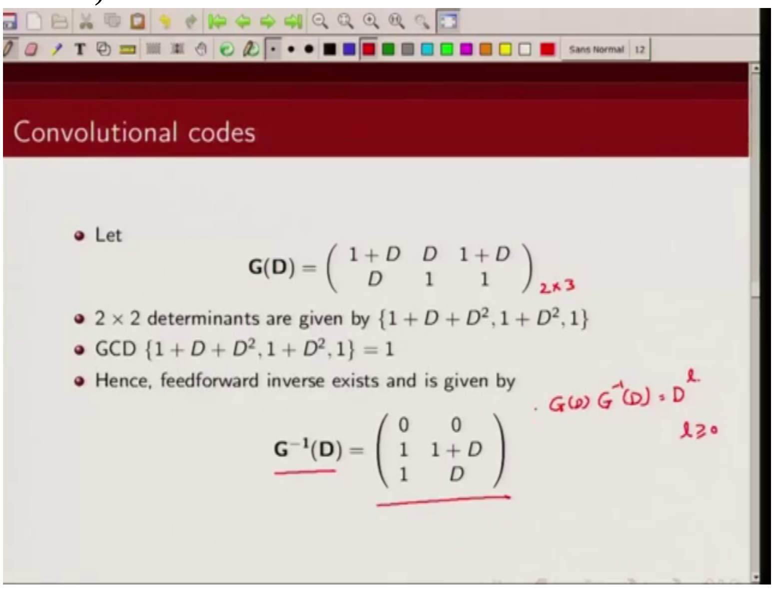
a feed forward inverse exists and this is given by this, Ok. And again we can check that G D G D inverse, this basically will be some delay elements where l is greater than equal to zero, it will be something like this.

(Refer Slide Time 10:52)



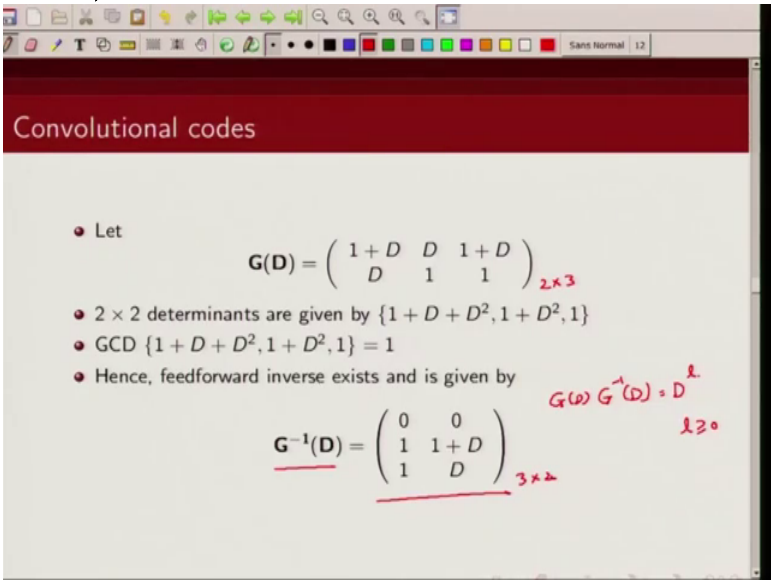
We can verify this quickly. Let's see this will be 1 plus D times 0 and then this will be D 1 plus D, this is 2 cross 3 and this will be

(Refer Slide Time 11:14)



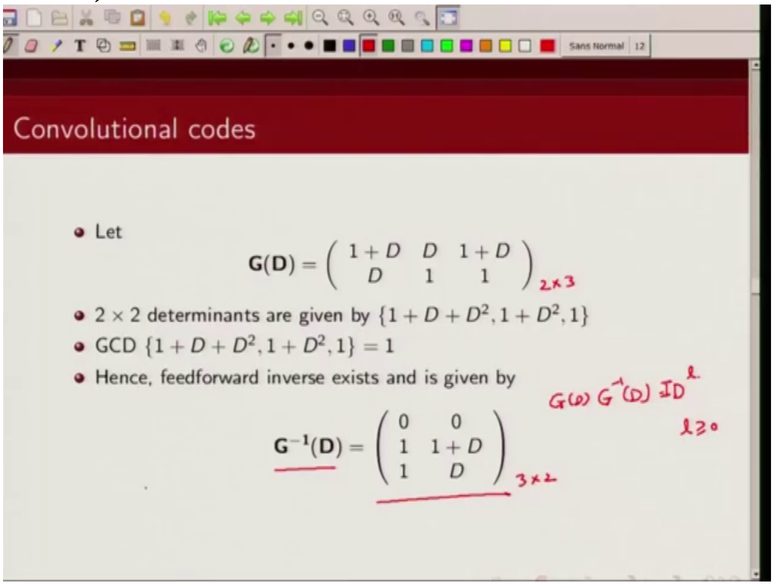
3 cross 2 matrix

(Refer Slide Time 11:17)



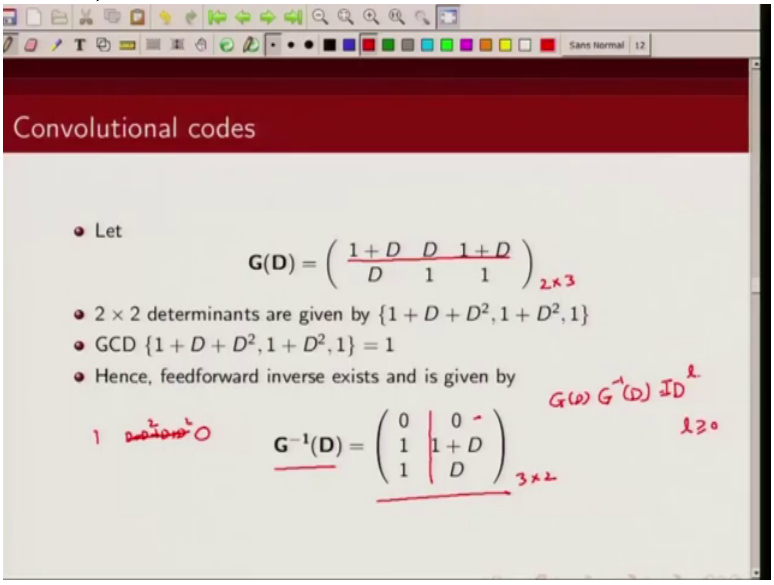
so what we will get is a 2 cross 2 matrix and so this will be some i times 2 cross 2 matrix. So

(Refer Slide Time 11:27)



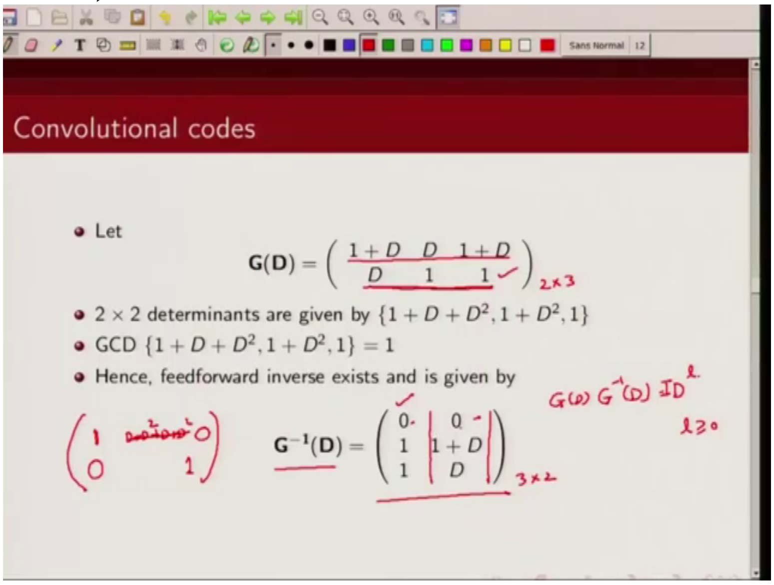
let's just work out. So this will be 1 plus D times 0, that is zero and then you have D times 1 and this is 1 plus D. So that's 1. First term will be 1. And this will be, multiply this by this, so that's 1 plus D into 0 that is zero, D into 1 plus D so that would be D plus D square and then 1 plus D into D so that's again be D plus D square so this will be zero.

(Refer Slide Time 12:10)



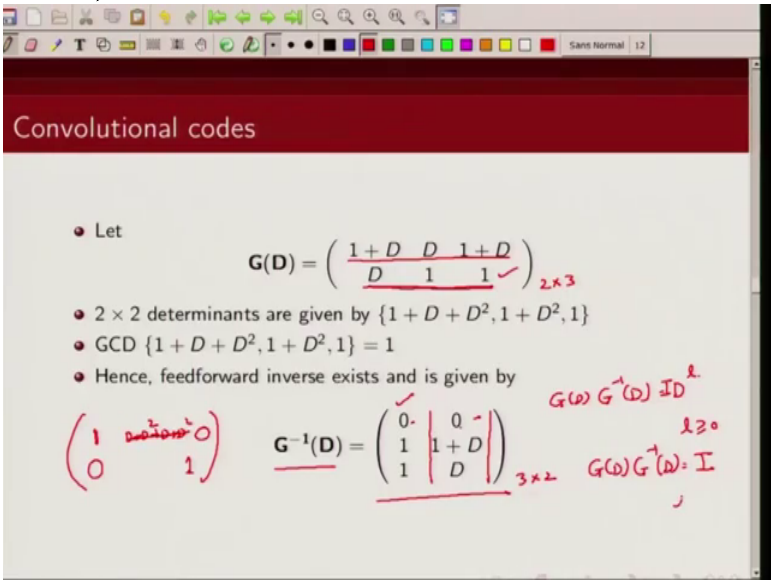
Next multiply this row by this column, so what we get D times 0, one times 1, so that's 1 plus 1 is 0 and if you multiply this by this, the second row by second column, what you get is zero times D, one times 1 plus D and one time d , so that is 1 plus d plus d so that is 1. So again

(Refer Slide Time 12:43)



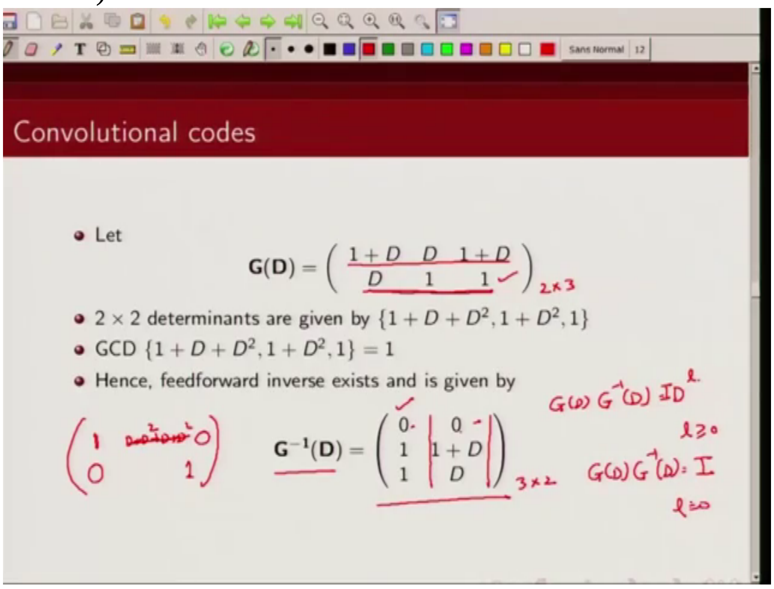
what we are getting for this case is G D, G D inverse is identity matrix. So l is zero

(Refer Slide Time 12:53)



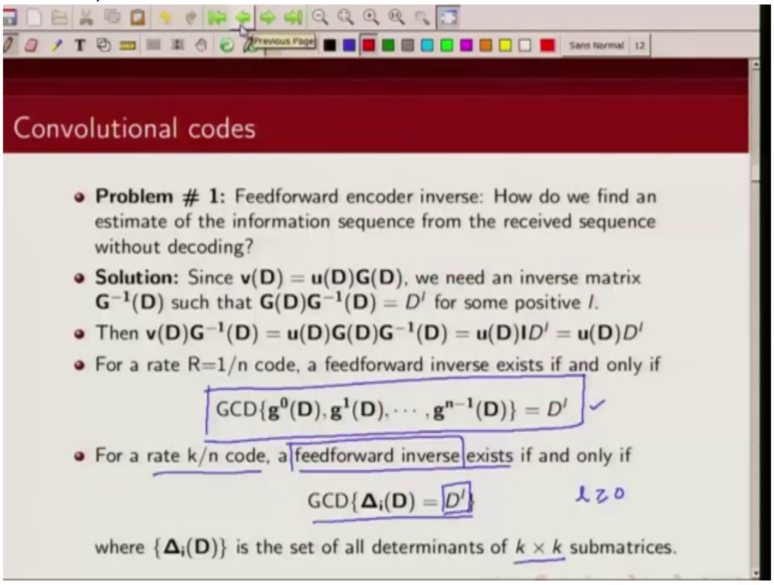
especially here. Ok

(Refer Slide Time 12:55)



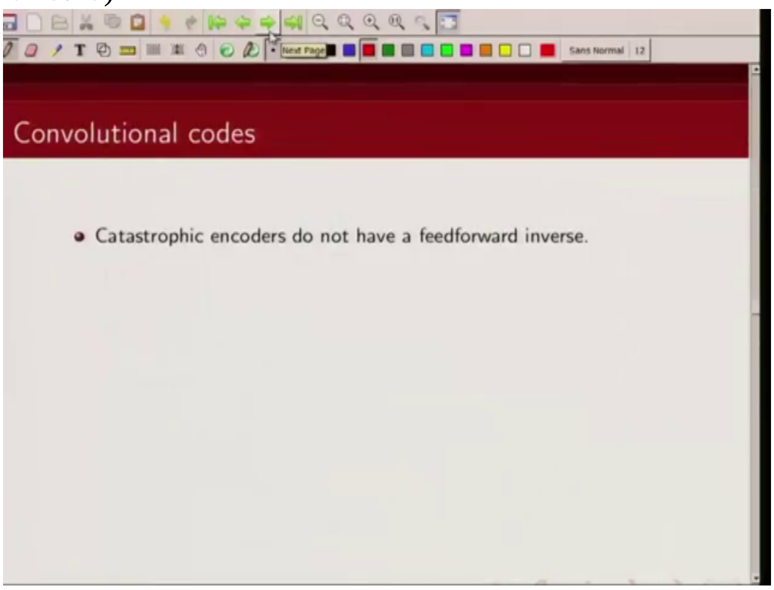
So this is a inverse for this generator matrix and we can see that this, all the terms are feed forward terms, 1, 1 by d and 1 1 so this is a feed forward inverse for convolutional code with this generator matrix. Ok. So now we, to recap basically, so the condition under which the feed forward

(Refer Slide Time 13:29)



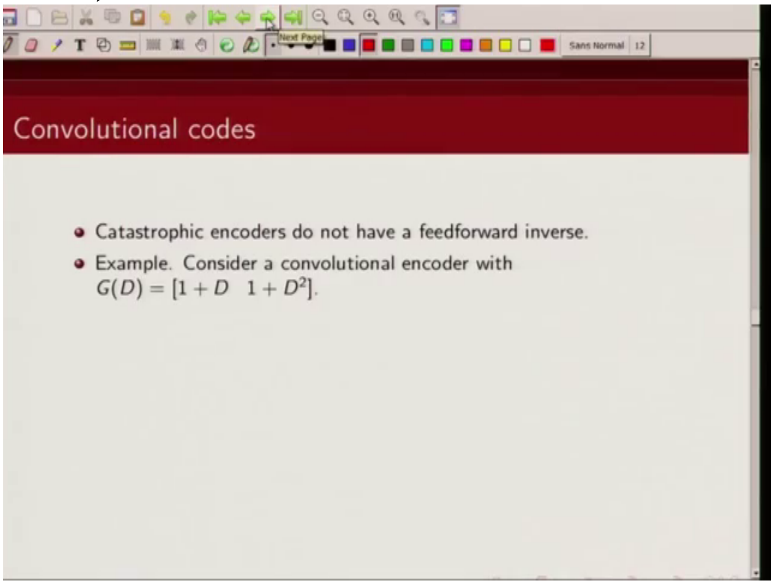
inverse for a convolutional code whose generator matrix is given by G D is given this condition for rate 1 by n code and for the k cross n code it is given by this condition.

(Refer Slide Time 13:45)



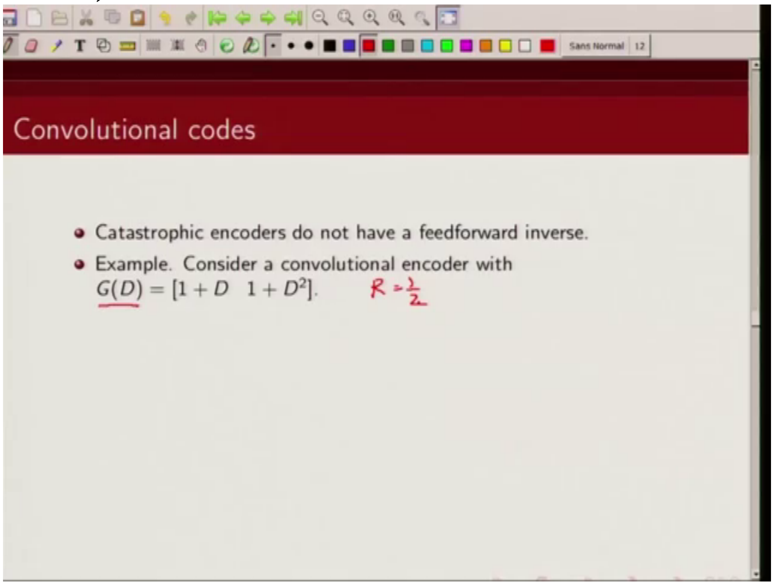
Now catastrophic encoders do not have a feed forward inverse. So for a catastrophic encoder we will just show you that their inverse has feedback terms. So let’s look at one example.

(Refer Slide Time 14:04)



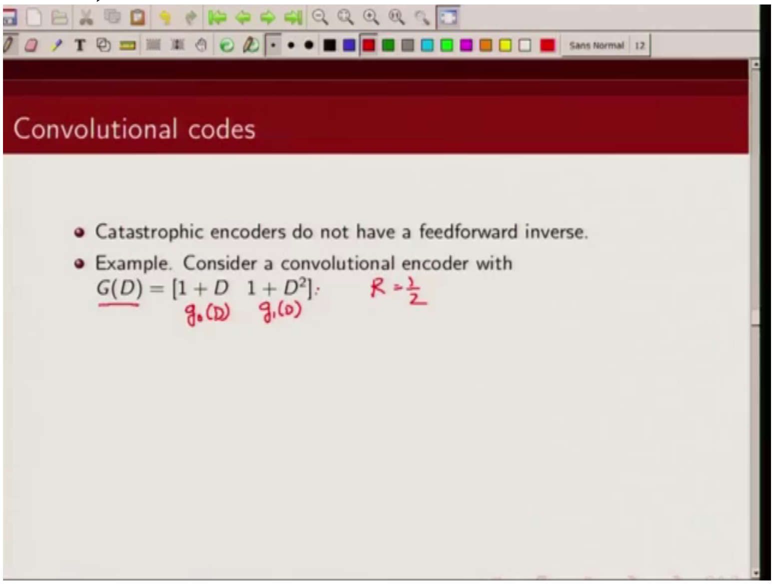
So let us consider a convolutional code whose generator matrix is given by this. So this is a rate 1 by 2 convolutional code.

(Refer Slide Time 14:17)



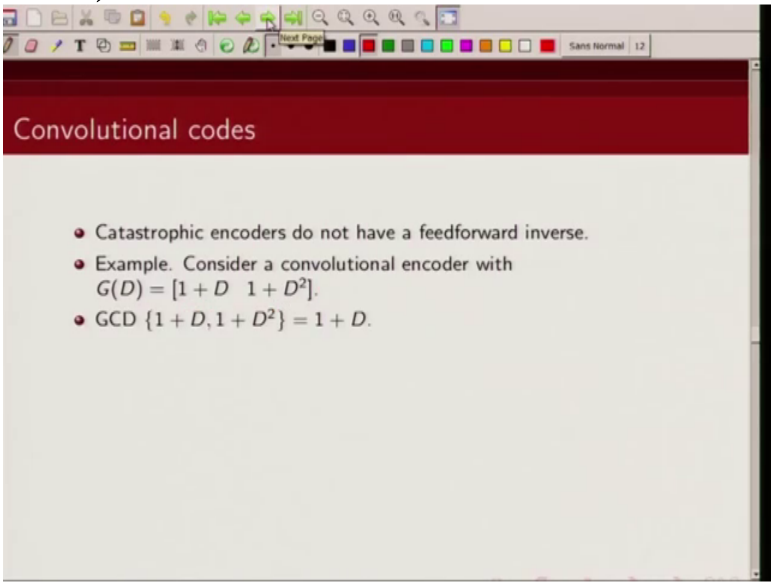
And it has 4 states because the maximum degree of D is 2. So the generator sequence is G 0 D is given by 1 plus D and G 1 D is given by 1 plus D square.

(Refer Slide Time 14:34)



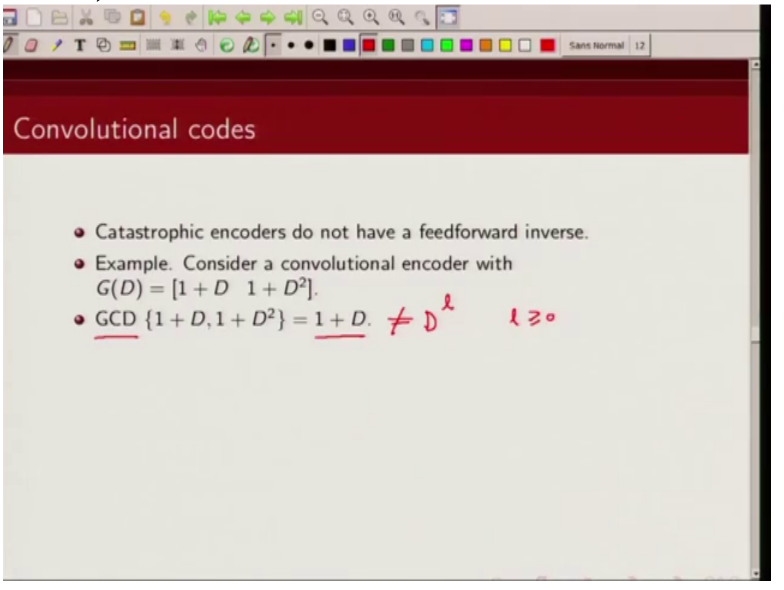
Now first

(Refer Slide Time 14:36)



thing that we check is what is the greatest common divisor among G 0 and G 1. As it turned out in this case, the greatest common divisor is 1 plus D. So that's not same as D to power l for some l greater than 0.

(Refer Slide Time 14:57)

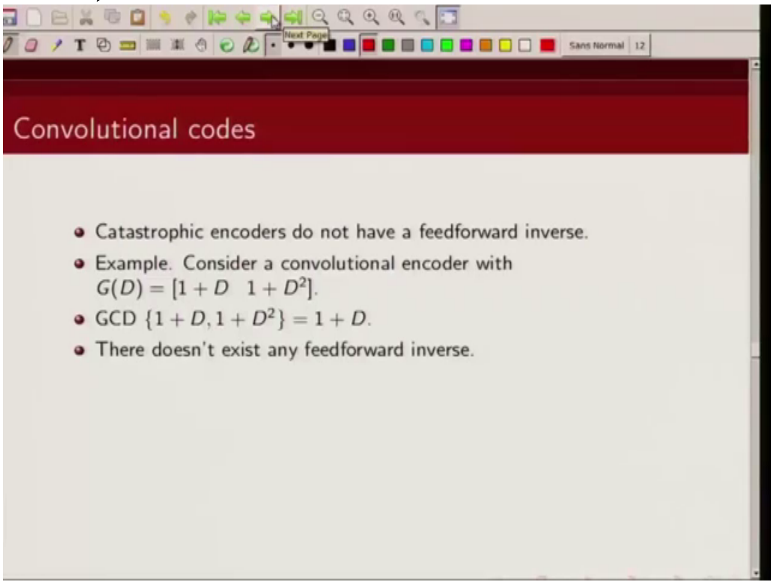


So that means for this generator matrix we do not have any

(Refer Slide Time 15:03)

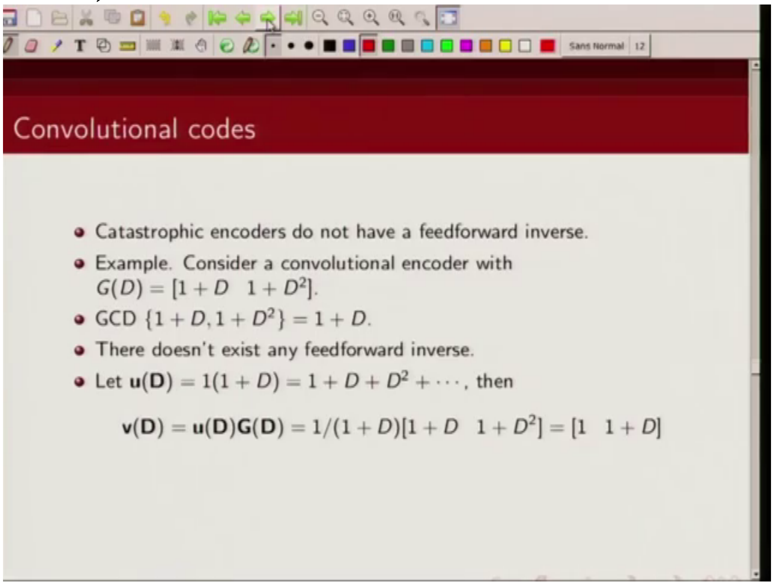
feed forward encoder inverse. So there

(Refer Slide Time 15:09)



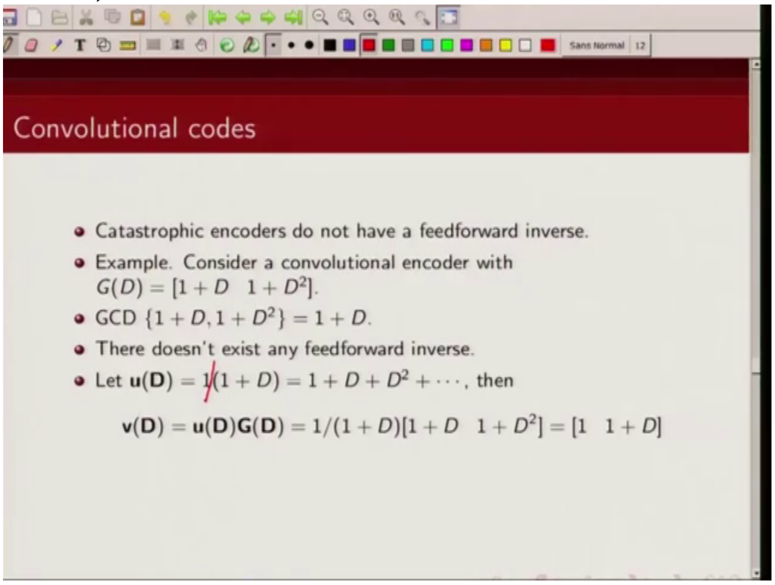
doesn't exist any feed forward inverse for this particular convolution code with generator matrix given by this.

(Refer Slide Time 15:20)



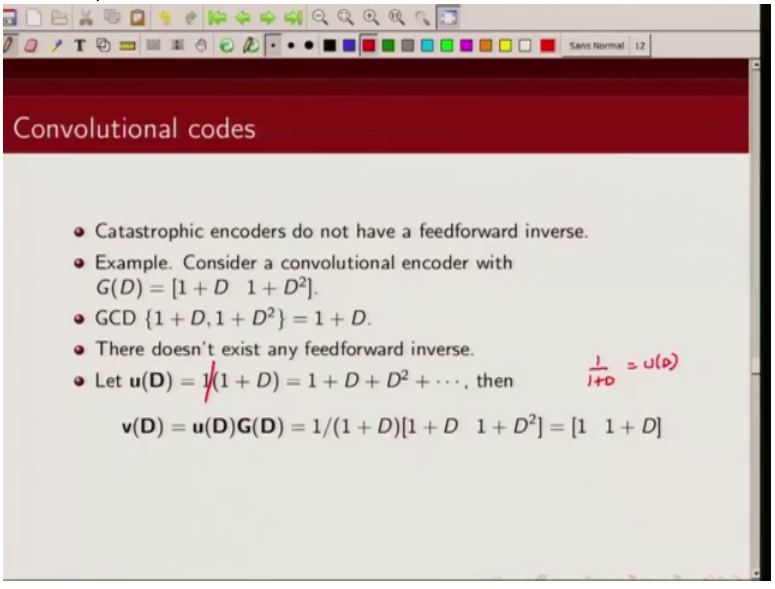
So let us take an example of u

(Refer Slide Time 15:24)



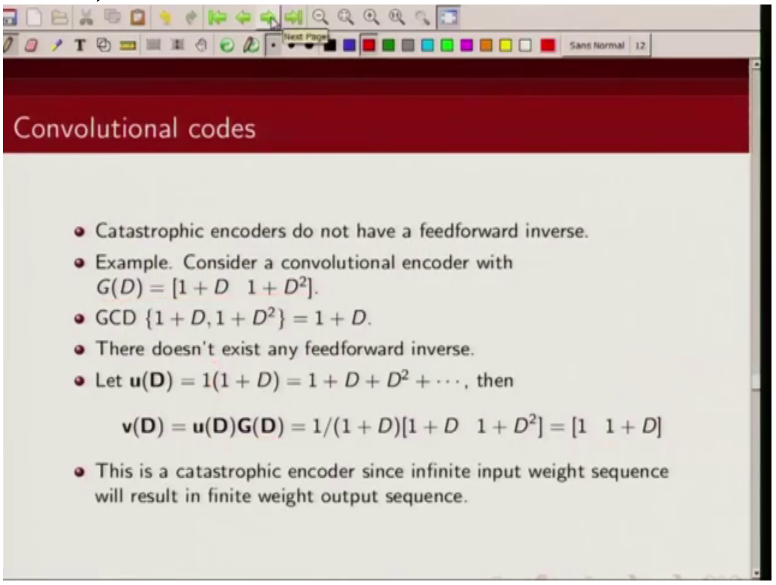
of D given by 1 by 1 plus D. This is typo, this is 1 1 by D. Now 1 1 by D can be written as 1 plus D plus D square plus D cube so this is an all 1 sequence. So our input is all 1 sequence which can be written as, in this D notation it can be written like this. This is my u of D. Now if I

(Refer Slide Time 15:47)



give this input to my convolutional code whose G D is given by this, what's my output? My output is u D times G D so this will be given by this. So note I just, my input has infinite weight but the output only has weight 3. And this is precisely an example of catastrophic encoder.

(Refer Slide Time 16:19)

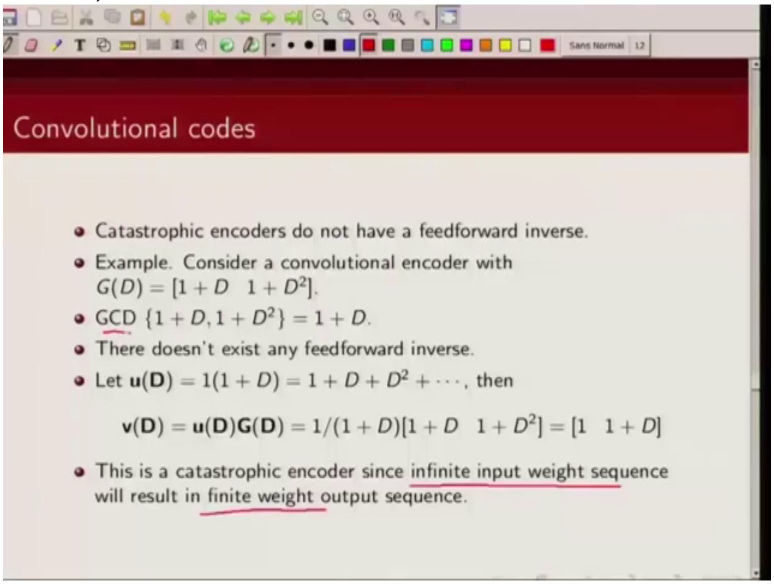


So you can see my input has infinite weight but my output has finite weight. So catastrophic encoder would not have

(Refer Slide Time 16:29)

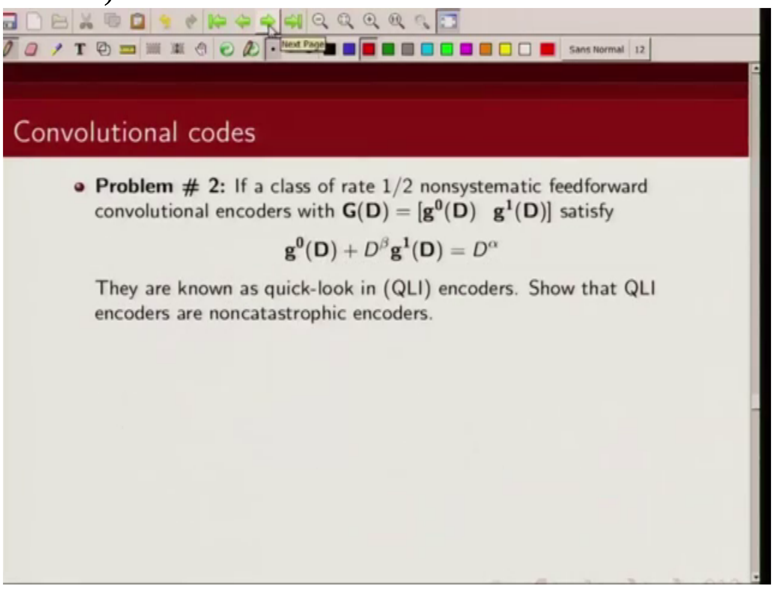
a feed forward inverse as this

(Refer Slide Time 16:34)



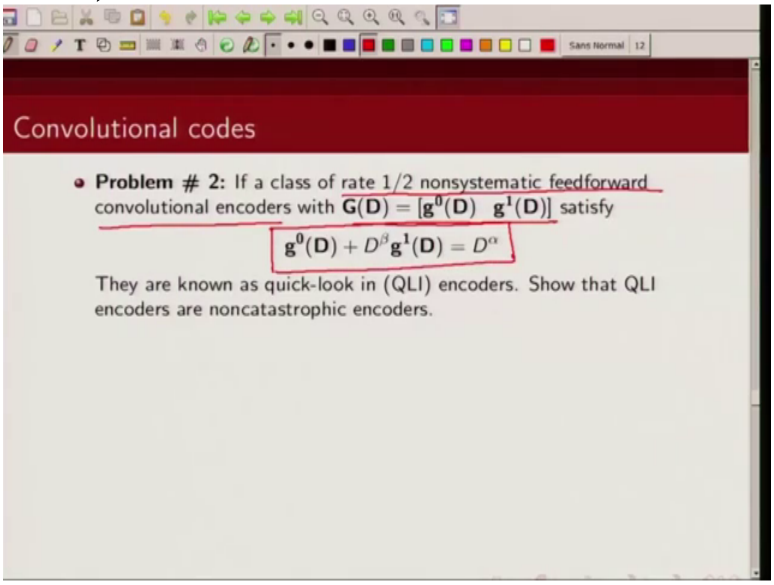
condition is violated.

(Refer Slide Time 16:39)



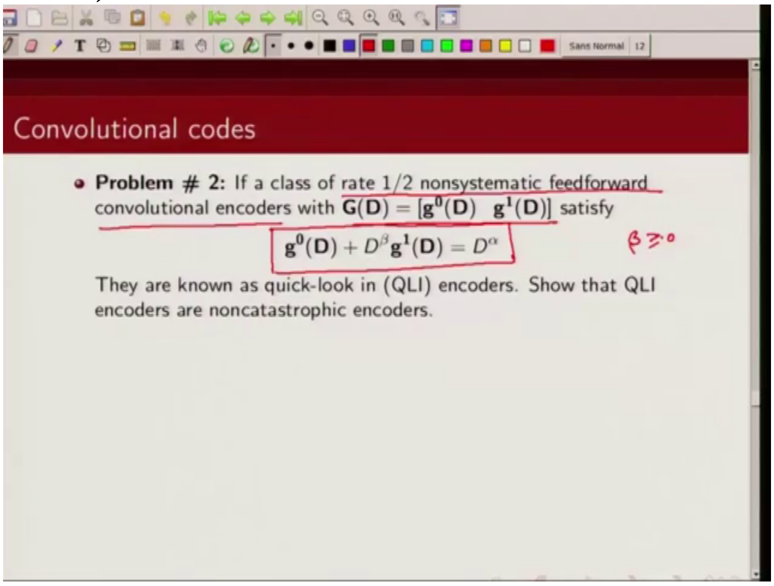
Next we look at a class of rate one half non systematic encoders. So we are looking at the rate one half non systematic feedback, feed forward encoders whose generator matrix is given by this and these generator G 0 D and G 1 D satisfy this property.

(Refer Slide Time 17:10)



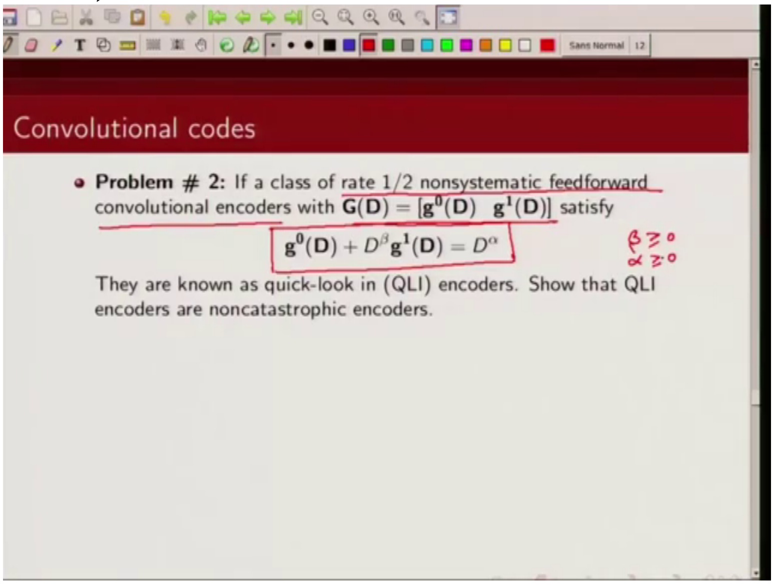
So what is this property? It is G 0 D plus some delay, beta is greater than equal to zero so

(Refer Slide Time 17:22)



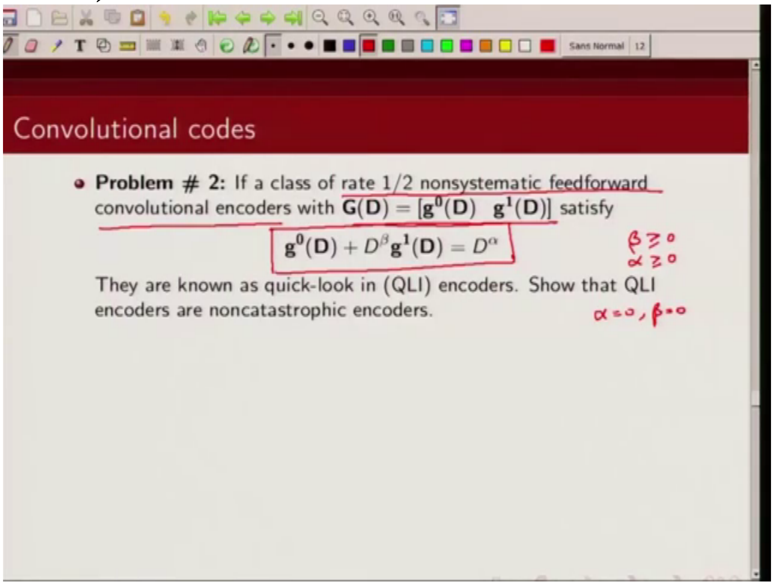
some delay of G 1 D is given by D alpha where alpha is also something greater than 0,

(Refer Slide Time 17:30)



Ok. Now let's take a simplified case and let's say alpha is 0, beta is 0. So what does it say?

(Refer Slide Time 17:41)

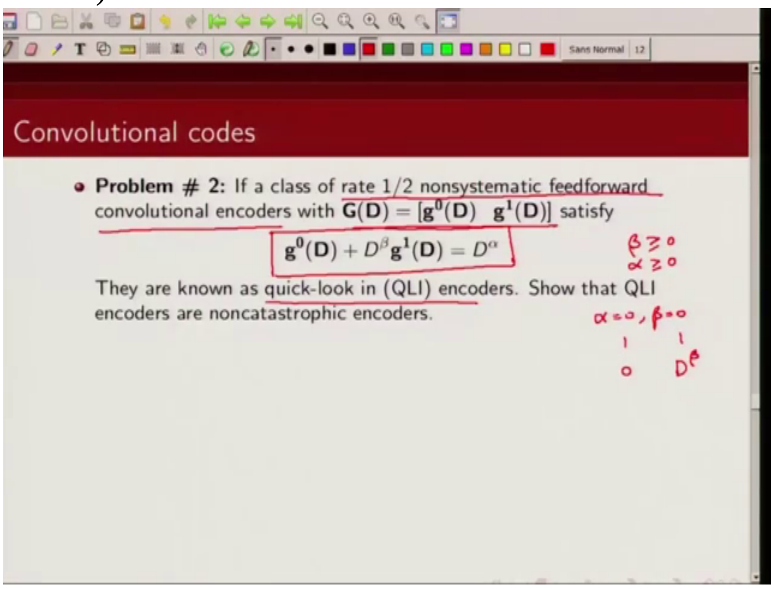


It says G 0 D plus G 1 D is 1. So then I can essentially, looking at these generators I can essentially find out that this encoder has a very simple encoder inverse, so which is just 1 and 1. If alpha is this. So these are known as quick look-in encoders. Why they are called quick look-in encoders because quickly looking at these encoders you can actually easily find the encoder inverse and essentially encoder inverse just consists of 2 tabs. So in some sense for a systematic encoder, the inverse

(Refer Slide Time 18:32)

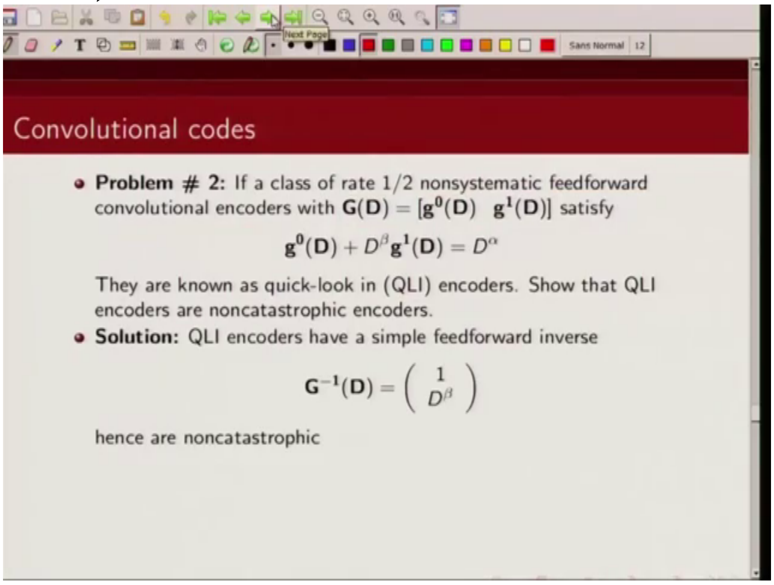
is of form 1 and 0 for a rate 1 by 2 code. And here they are of the form 1 and d beta. So

(Refer Slide Time 18:42)



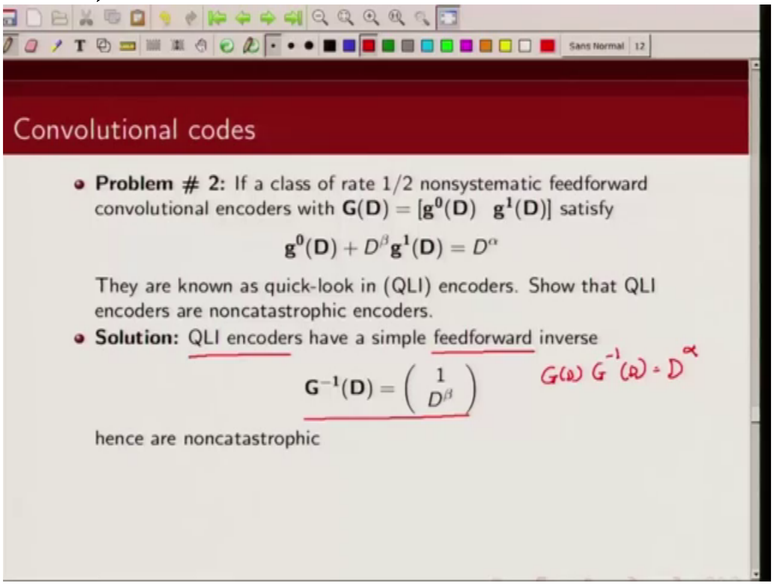
they are in some sense closest to systematic code if you like to call them. So this

(Refer Slide Time 18:49)



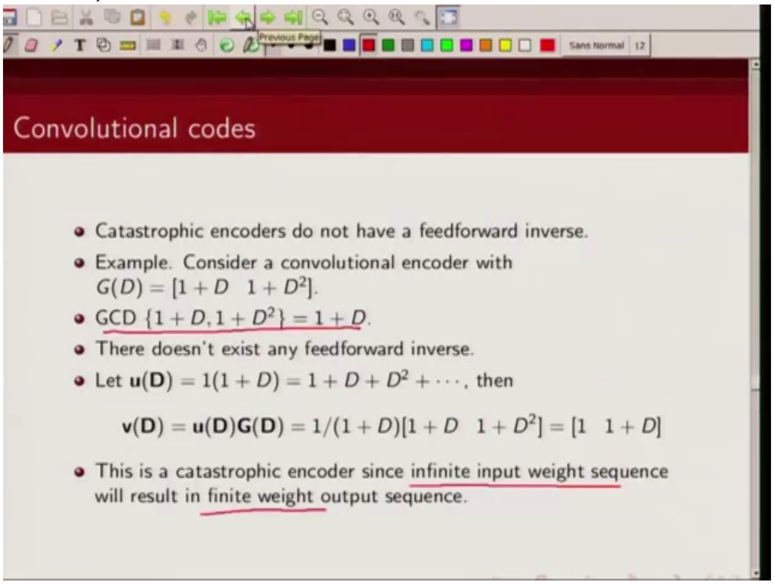
quick look-in encoders have a very simple encoder inverse, a feed forward encoder inverse and that is given by this. You can verify that G D G D inverse is your d times alpha,

(Refer Slide Time 19:12)



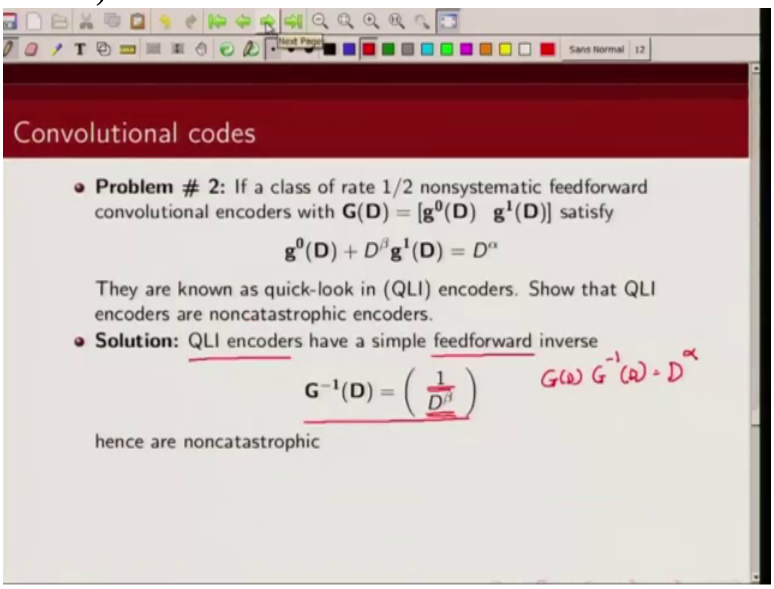
Ok. Now note that the encoder inverse of quick look-in encoder has just 2 tabs, 1 and this d beta. And it has a feed forward inverse so it cannot be a catastrophic encoder. We just showed in the previous slide that

(Refer Slide Time 19:34)



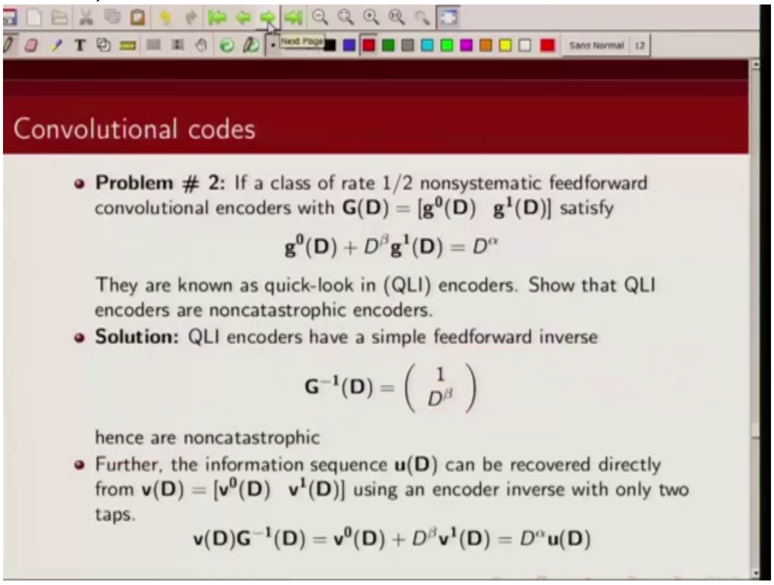
a catastrophic encoder does not have a feed forward inverse. And since

(Refer Slide Time 19:42)



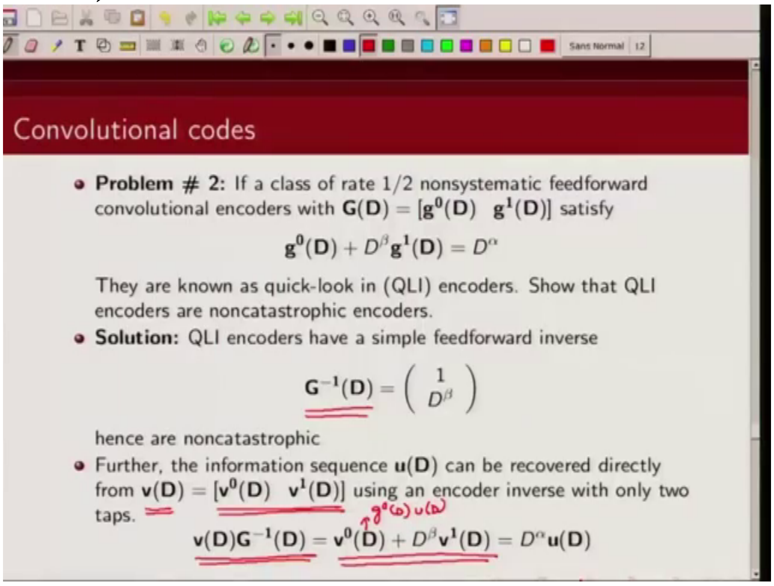
this has a feed forward inverse this cannot be a catastrophic encoder. So because they have a feed forward inverse they are not catastrophic. And

(Refer Slide Time 19:53)



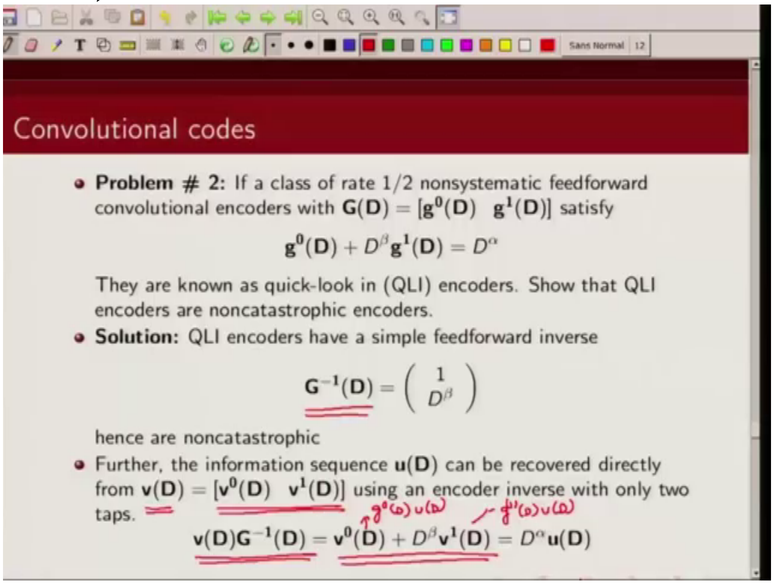
as I said you can very easily recover back your information sequence by making your coded sequence pass through this encoder inverse. So if your output sequence is given by v D which is this, then once v D passes through this encoder inverse, what we get is v 0 D plus D beta v 1 D. Now we know that quick looking code have this property that g 0 D plus beta g 1 D is D alpha and v 0 D is g, this is equal to v 0 D times u D, similarly this

(Refer Slide Time 20:50)



one is G 1 D times u D.

(Refer Slide Time 20:58)

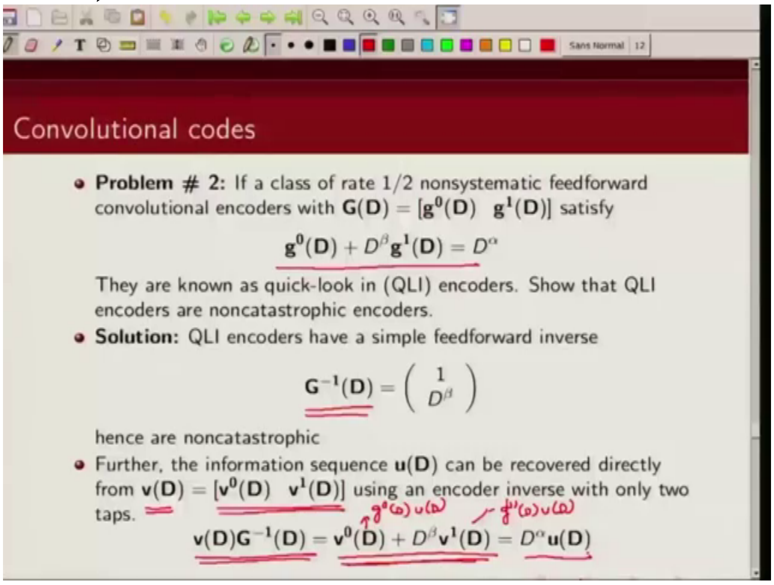


So from this condition and from here, this will come out to be D alpha times u D. So among the class of non systematic encoders quick look-in encoders have a very simple

(Refer Slide Time 21:16)

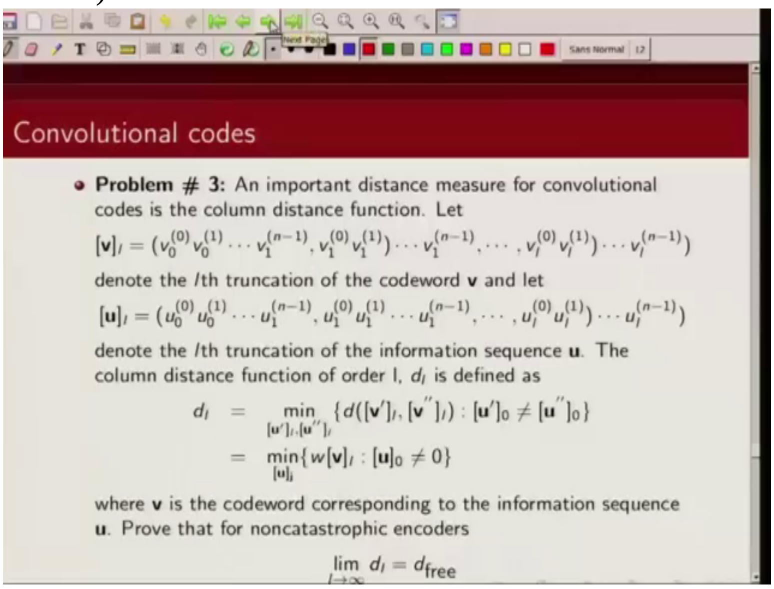
encoder inverse circuit and one can

(Refer Slide Time 21:20)



easily find out what the information bits are from the coded bit without decoding.

(Refer Slide Time 21:29)

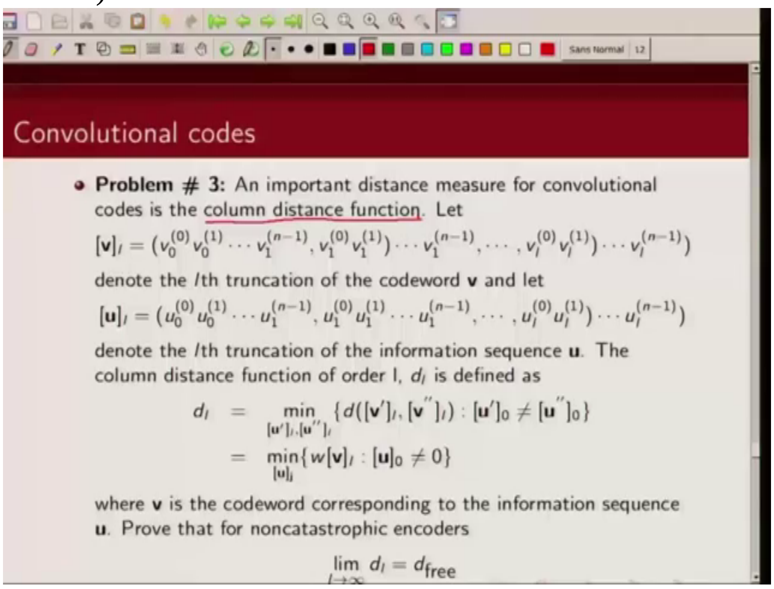


The next problem that we are going to talk about is about a distance measure for convolutional code. So we will first define what we mean by column distance function. As we know a convolutional encoder can continuously encode an information sequence. So we can have

(Refer Slide Time 21:53)

an infinite length input sequence and correspondingly an output sequence. Now

(Refer Slide Time 22:02)

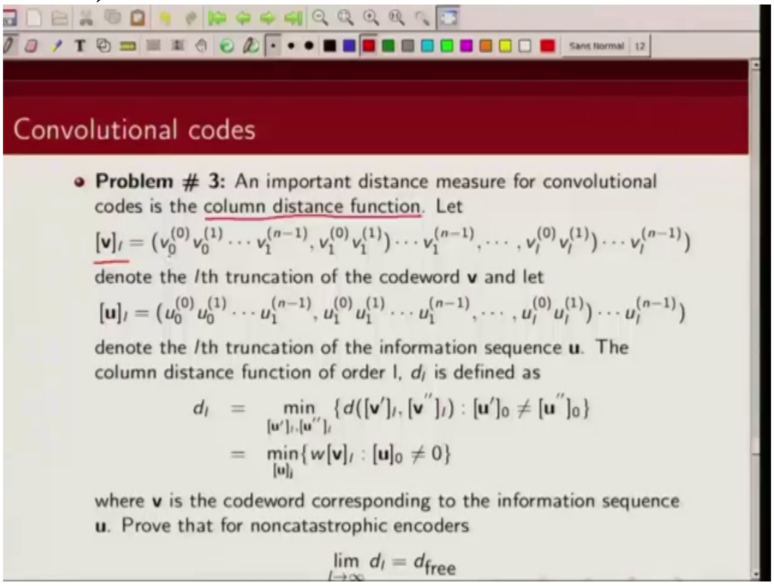


we define the column distance function for a convolutional code as follows. So before that I am describing output code sequence v which is truncated to up to length l. So this notation that you see v sub l, it shows essentially a codeword up to time l.

(Refer Slide Time 22:30)

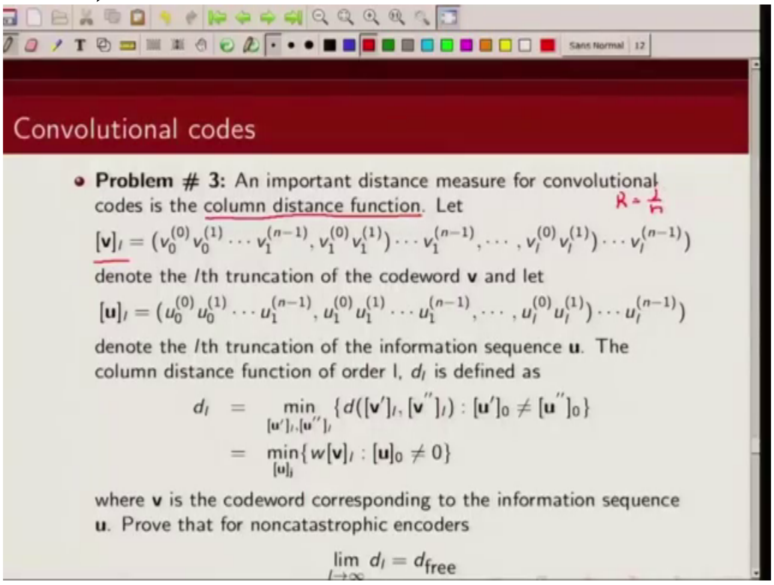
And what is our codeword up to time l? So this will be

(Refer Slide Time 22:34)



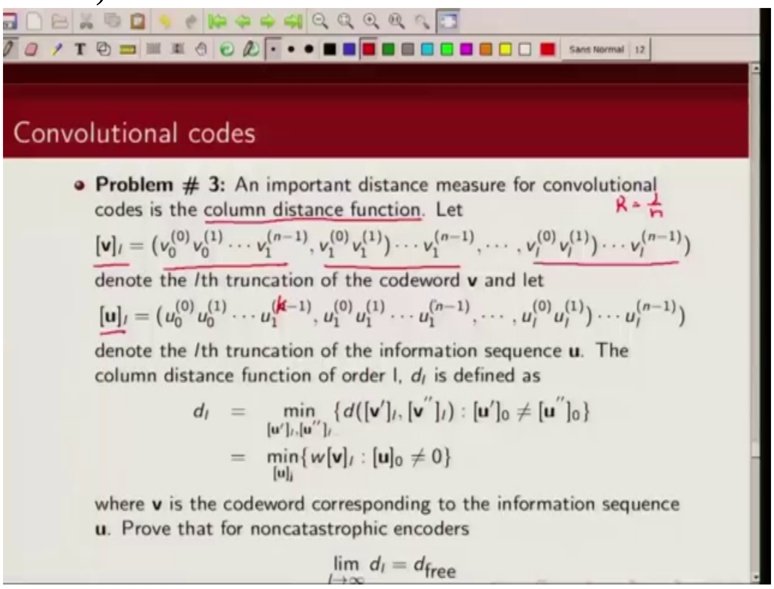
v 0, v 1, v n minus 1, then v 1, v n because this is a rate, let's say this is a rate 1 by n code then

(Refer Slide Time 22:43)



this for first time instance you have n bits, for second time instance you have n bits then for l time instance you will have n bits. So this is your truncated codeword up to time l. Similarly I can define my truncated information sequence, Ok so just a minute, this is a typo, this should be k minus 1,

(Refer Slide Time 23:10)

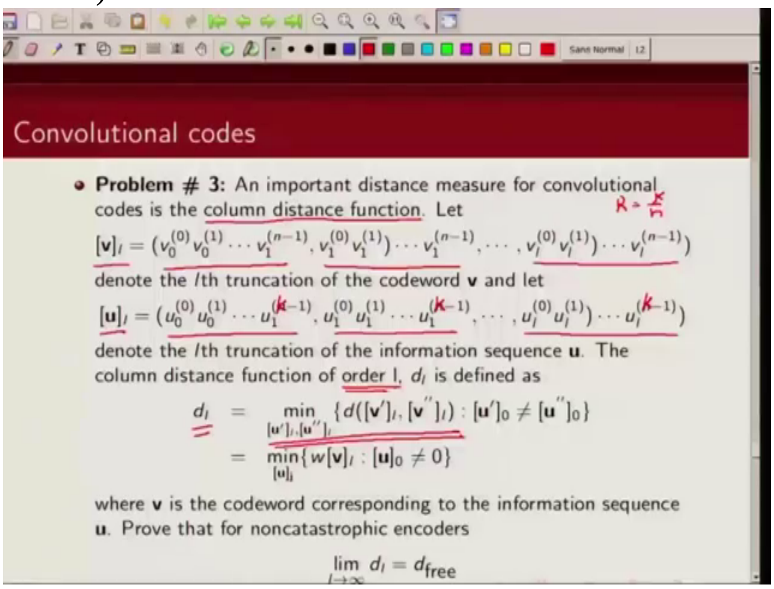


this should be k minus 1, k minus 1 and of course if k is 1, there will be this 1 input so you have, for first time instance you have k inputs, second time instance you have k input and similarly for l time instance you have k input. Now note that both your information and coded sequences truncated up to length l. Now we define column distance function of order l as follows. It is the minimum distance between 2 truncated codewords of length l

(Refer Slide Time 23:59)

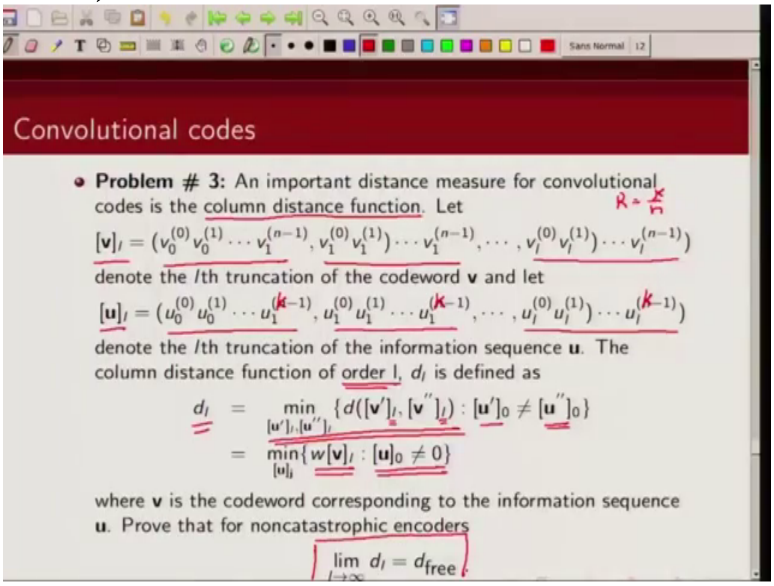
such that, so

(Refer Slide Time 24:01)



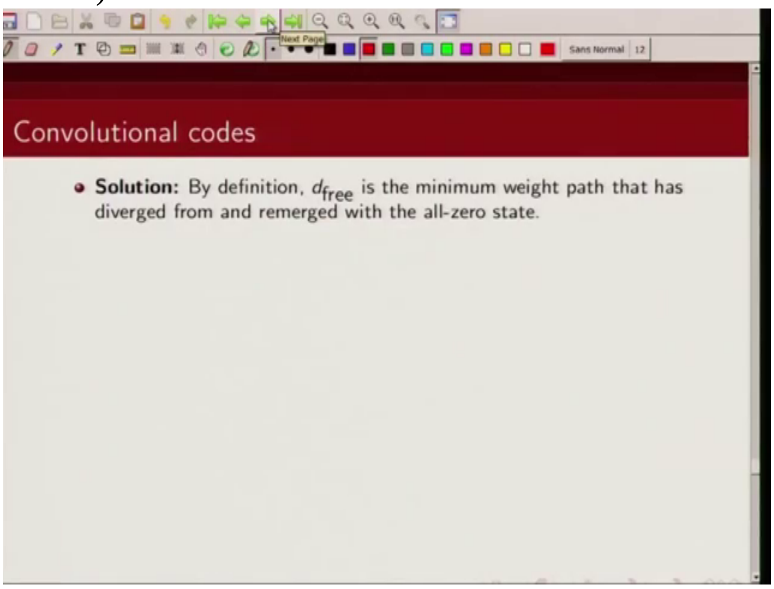
you can see it is a minimum distance between 2 codewords v 1 hat and v 1 double hat both of length l, such that u dash and u double dash they are not same. So it's, it’s essentially Hamming distance between 2 truncated code sequences. Now we know that Hamming distance between 2 sequences, minimum distance can be written as minimum weight of a non zero codeword. So the same thing we can write as minimum weight of a lth truncated codeword belonging to a non zero information sequence. So we can define our column distance function of order l as minimum weight of lth truncated code sequence belonging to a non zero information sequence. Now the thing that you have been asked to prove here is show that as l goes to infinity this column distance function tends towards free distance

(Refer Slide Time 25:25)



of the convolutional code. In fact after 3 or 4 constraint length you will see the distance which is d free and then it remains there,

(Refer Slide Time 25:36)

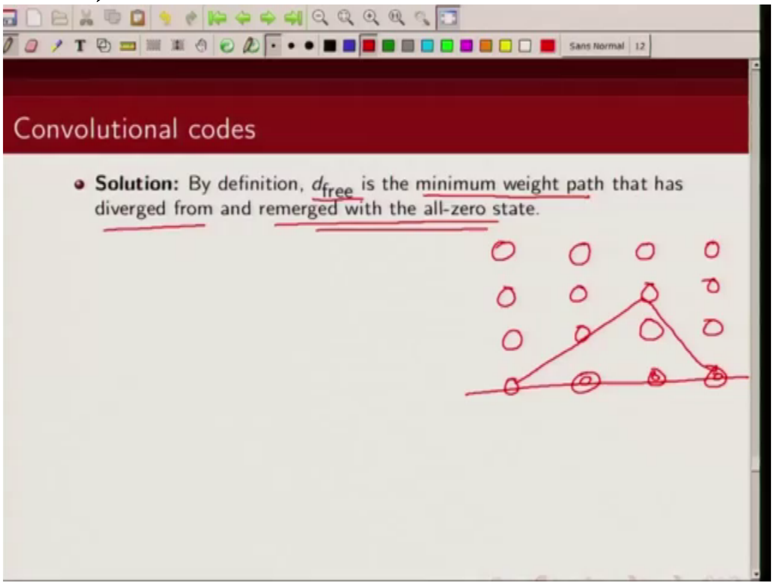


column distance function. So by definition what is a free distance of convolutional code, it is the minimum weight path that has diverged from an all zero state and merged back into all zero state. How do we find minimum weight codeword, minimum length codeword, so if you have convolutional code, without loss of generality let's say we are

(Refer Slide Time 26:05)

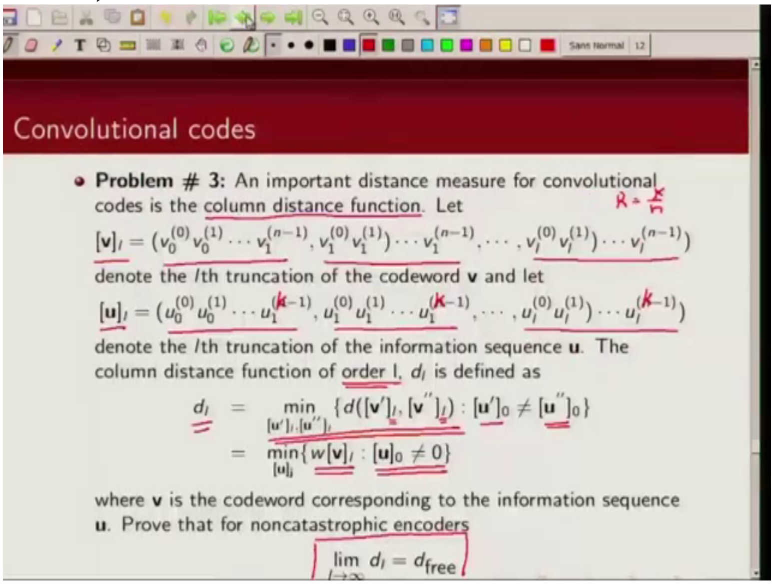
transmitting all zero codeword then minimum weight codeword would be the length of the uh, the minimum weight along all non zero, a path that goes through non zero state. So let's say you have some convolutional encoder, some 4 set convolutional coder and this is your all zero state, all zero state so this is all zero state and let us say you have some diversion from this and then you are coming back.

(Refer Slide Time 26:47)



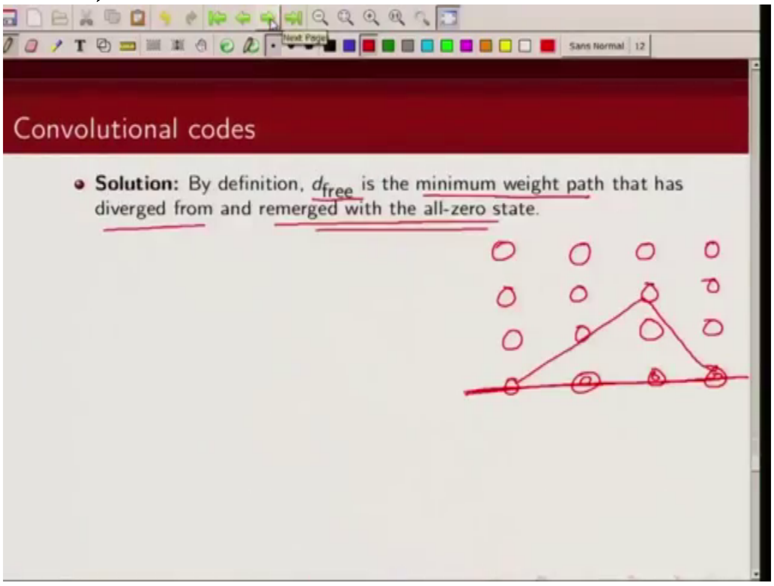
So this is your all zero state and what is your

(Refer Slide Time 26:54)



column distance function? It's a minimum weight of your codeword belonging to non zero information sequence.

(Refer Slide Time 27:06)

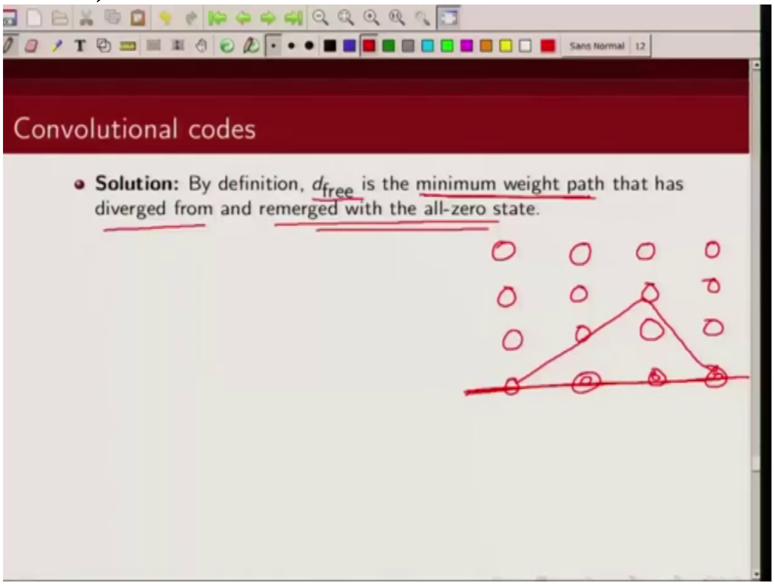


And what are the paths through the Trellis diagram? These are all our valid codewords. So we need to find a path through the Trellis which has minimum weight, so and that will be our free distance, so the free distance is minimum weight path that has diverged from all zero state and merged back,

(Refer Slide Time 27:30)

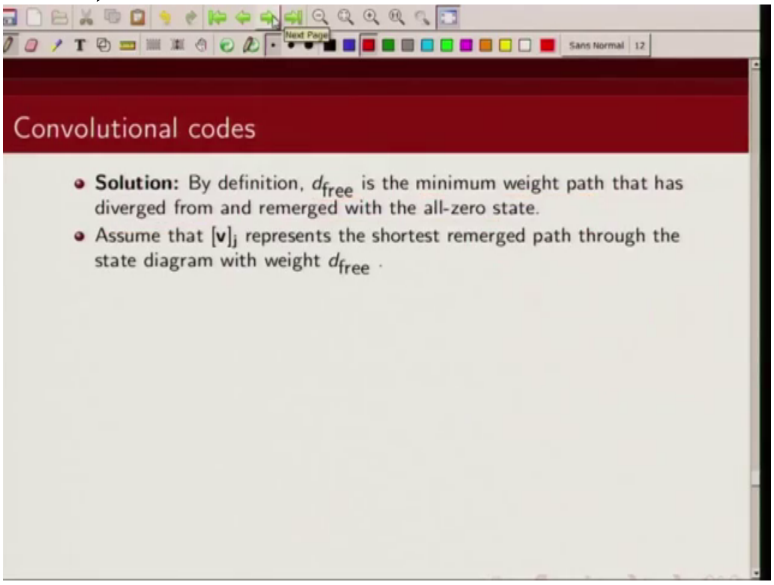
right. So to get a

(Refer Slide Time 27:33)



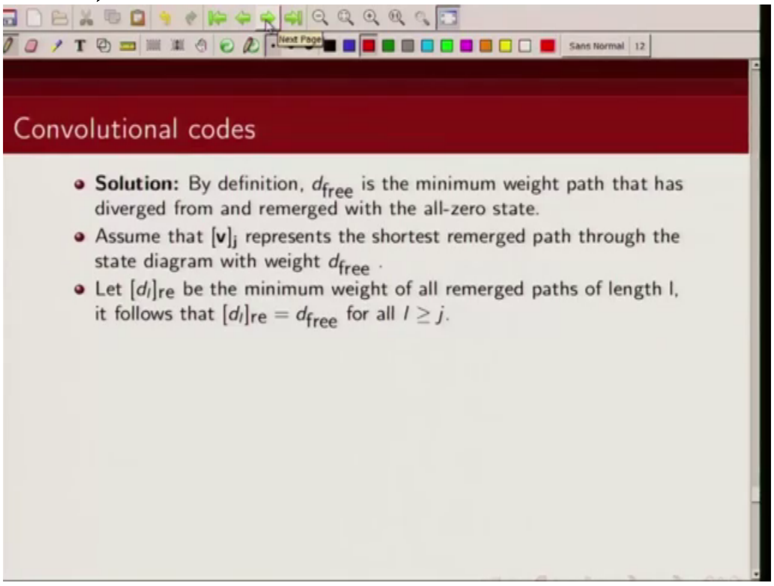
non zero weight you essentially diverge from all zero state and then merge back to all zero state. Now let us

(Refer Slide Time 27:42)



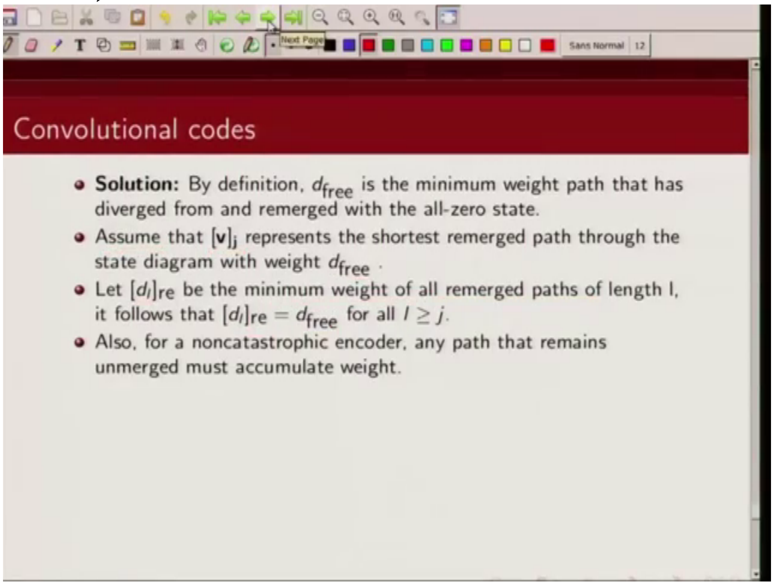
assume that at time t equal to j, so v j represents the shortest remerged path through this Trellis diagram or state diagram which has weight of d free. So if t equal to j is the smallest times which represent the shortest remerged path through this Trellis diagram then what does it mean?

(Refer Slide Time 28:12)



It means if we denote by d l the minimum weight of all remerged path, what is remerged path? So these paths that are diverging from all zero state and then merging back into all zero state. These are our remerged path. Now what we are saying, for t equal to j, that is the smallest remerged path which has weight equal to d free. So if you have any time any j which is greater than, any time which is greater than this j then your column, distance column function will be equal to the free distance. Why this is so, because we have said that for time equal to j that is the shortest remerged path through this Trellis diagram which has free distance, which has weight equal to free distance. So we take any time larger than that, then of course we will have a remerged path having minimum distance d free.

(Refer Slide Time 29:35)

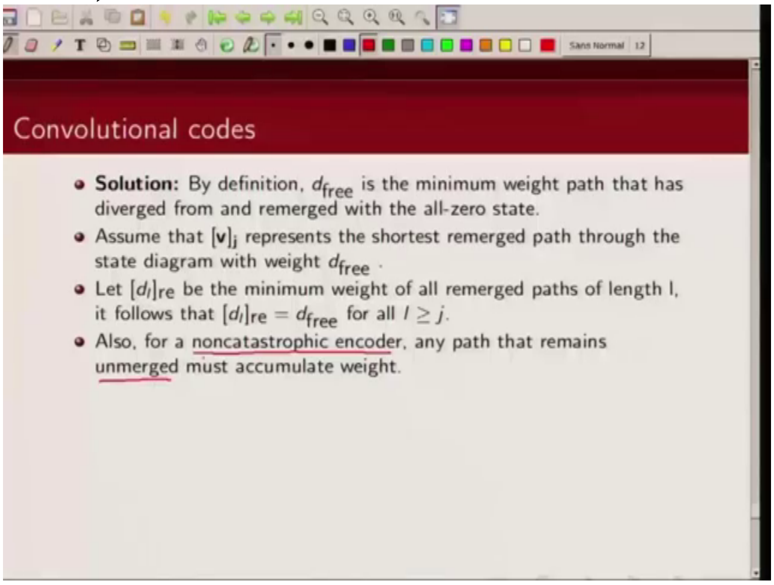


Now if there are any non-merged path, what are non merged path? So these paths which have diverged from all zero state but have not yet merged to all zero state, so those are unmerged path. Now for a non catastrophic encoder, any path that is not merged must accumulate weight. Only for the catastrophic encoder we have situation where input weight is higher and output weight is smaller.

(Refer Slide Time 30:11)

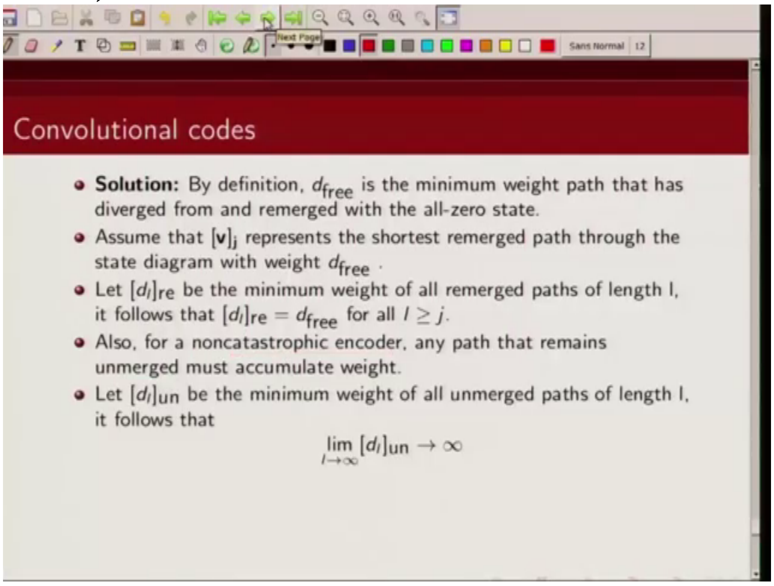
But if it is a non catastrophic encoder it will accumulate weight. So

(Refer Slide Time 30:17)



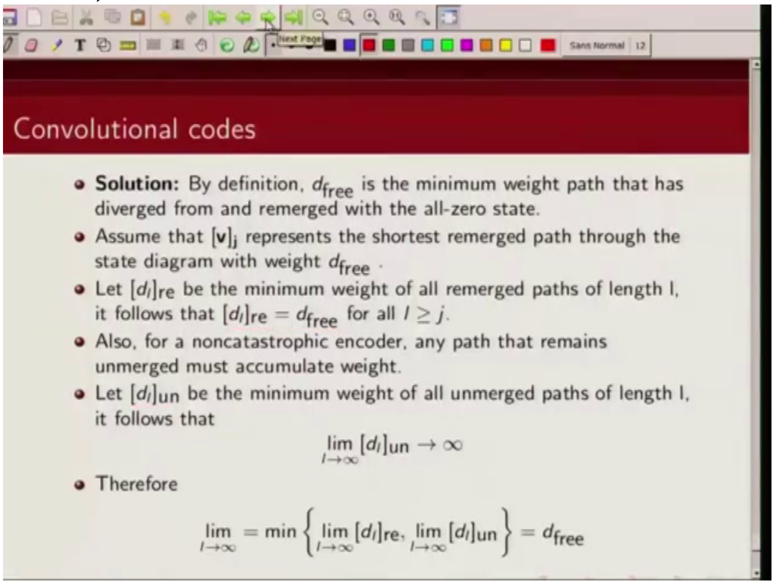
if we have a non systematic encoder, any path which has not yet merged with all zero state will try to accumulate more and more weight.

(Refer Slide Time 30:28)



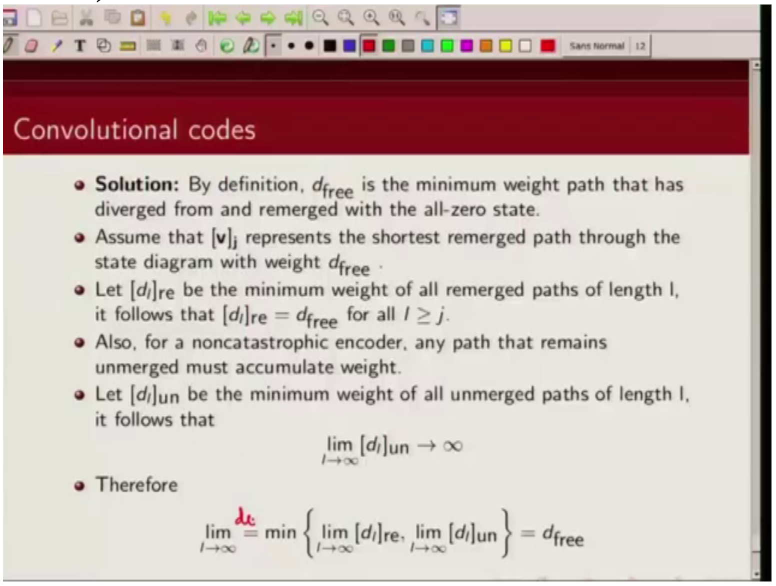
So what's going to happen? So if we look at distance for, column distance for unmerged path, then as l tends to infinity, this distance will also grow. This will also go to infinity because it is a non catastrophic encoder. So what we have shown is so for l greater than j, for all remerged path this column distance function is d free and for un-merged path this is going to be infinity as l goes to infinity. Hence we can

(Refer Slide Time 31:12)



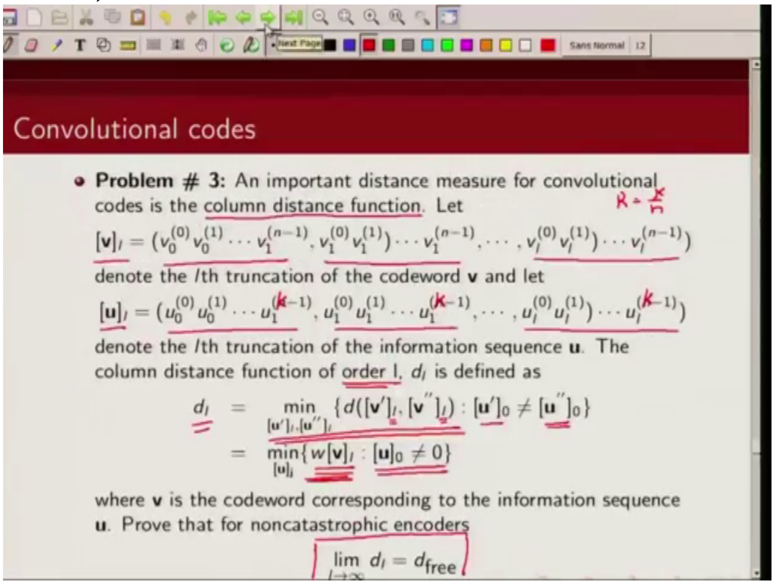
say that limit d l is

(Refer Slide Time 31:18)



minimum of the column distance for remerged path or unmerged path. This is infinity, this is d free. So we know that as l tends to infinity, the column distance function will be

(Refer Slide Time 31:42)



d free. So this proves that column distance function will go to d free as l goes to infinity, thank you.