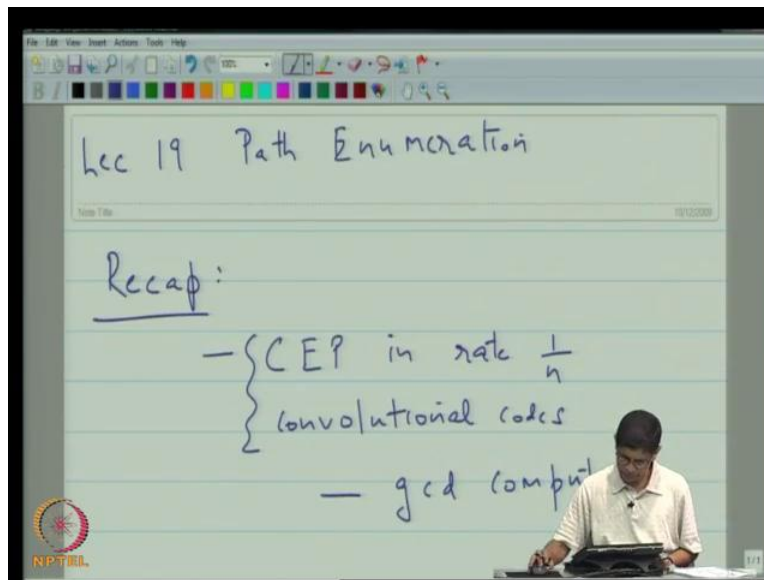


**Error Correcting Codes**  
**Prof. Dr. P. Vijay Kumar**  
**Electrical Communication Engineering**  
**Indian Institute of Science, Bangalore**

**Lecture No. # 19**  
**Path Enumeration**

Good afternoon. In the last lecture which you can see on this in miniature, what we did was we discuss primarily catastrophic error propagation. We focus on the subclass of convolutional codes is rate is a form 1 by n, which means a single input and n outputs, and we lead down necessary and sufficient condition in order for a generator matrix, not to suffer catastrophic error propagation. And towards end of the lecture, we and along the ways since we need to understand how to compute GCD is, we also provide it some background. And towards end of the lecture, I mention that we are also interested in path and enumeration, so we started doing that. So, that why will actually continue?

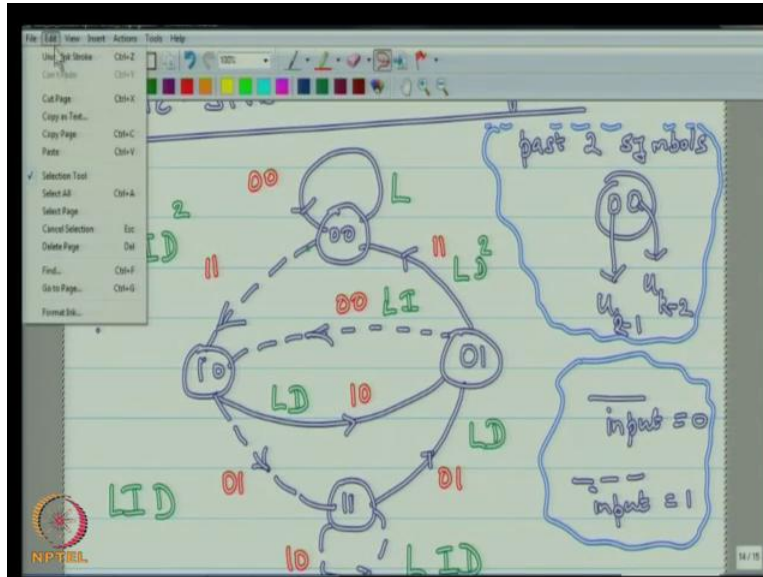
(Refer Slide Time: 01:07)



So, recap we looked at catastrophic error propagation in rate 1 by n convolutional codes, as background we developed GCD computation involving polynomials. And then towards the end, I

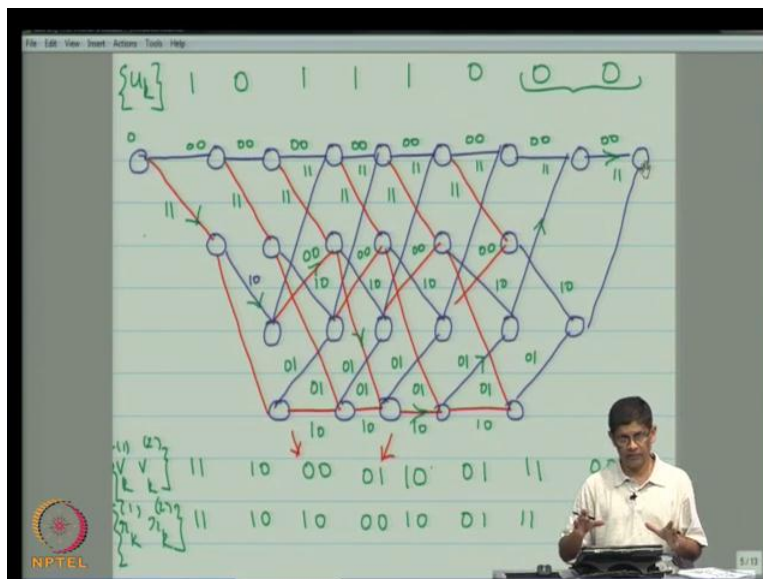
started talking about path enumeration in a convolutional code. So, let us, continue the discussion where we left at of last time.

(Refer Slide Time: 02:45)



I am going to go back, and let us copy this page again here, I have this down and what will interested here is in enumerations paths or code words of a certain type.

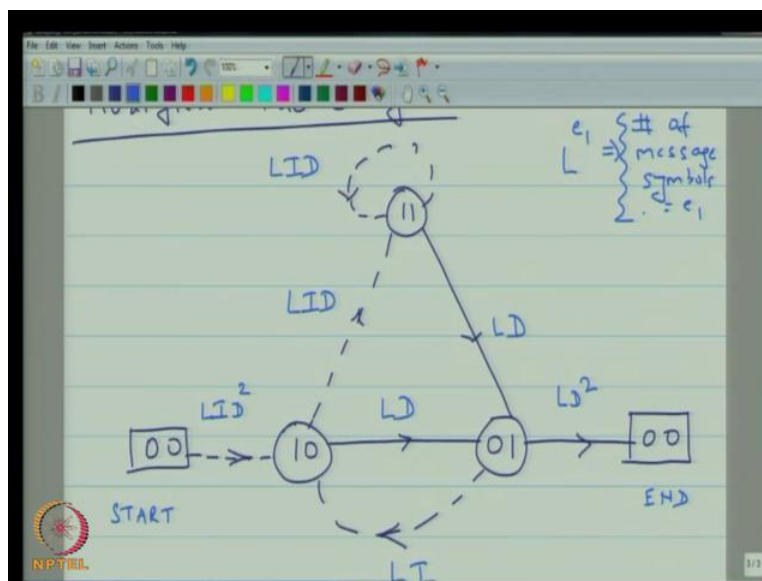
(Refer Slide Time: 03:43)



And I think it is best to go back to perhaps, lecture seventeen here. So, with respect to this trellis I have already pointed out that, every path to the trellis which starts with start node and ends with the end node corresponds to a valid code. For reasons, that will become a parent later, we are not really; interested in all code words, because our goal is to actually do some path enumeration for two purposes; one is to say something about; the hamming distance properties of the convolutional code, and secondly to actually; do some performance analysis, that is in order to be able to develop form and the probability, of the error of a convolutional code.

These are the two goals right now, the path we are interested in are those which actually; depart at some point from the all 0 states and return for the first time to the all 0 states. And we want to know in the intervening period, between the time they exit. Exits if the all 0 state; the all 0 state is the state of what happens in between, how many? What is the length of the path, how many information bits in this  $n$  by  $1$ ? And how many output symbols by  $1$ ; our starting point is the finite state machine, but since, we want to start at the all 0 state and come back for the first time to the all 0 state, what will do is? Will draw, what is called a modified state diagram for the convolutional code?

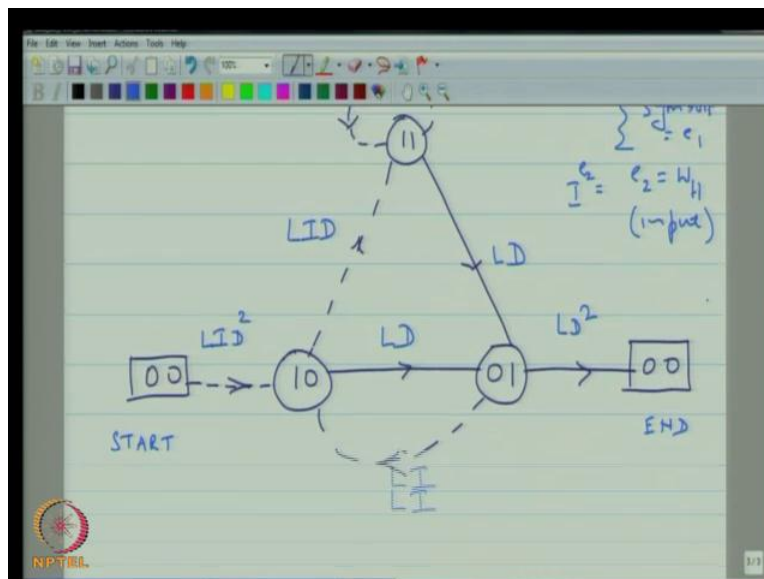
(Refer Slide Time: 05:16)



This modified state diagram looks like this, it is the same state diagram, as we have here, expect that listen, we opened up the all 0 state spread into two call one the start node and call the other the end node. This diagram will call this is start node, and this will be the end node and each dotted line of course, represent an input that is 1. I want to retain the same labels as here, so whatever labels are here, corresponding to certain transition.

I am going to actually; reproduce in there and for that reason, just looking at my nodes here, from 0. If, you go to 1 1 there output is a LID squared; from 1 0 if you go to 0 1, you have an LD; From 0 1 back to 1 0 is LI; from 1 0 to 1 1 is LID; from 1 1 back to 1 1 is LID; from 1 1 to 0 1 is LD; from 0 1 back to 0 0 state is LD squared. This is are modified state diagram and you can actually now, do some enumeration; just to repeat L L starts stands for L to the let see, if I put L to the e 1 implies that, length of the path or the number of the message symbols associated with path is equal to e 1.

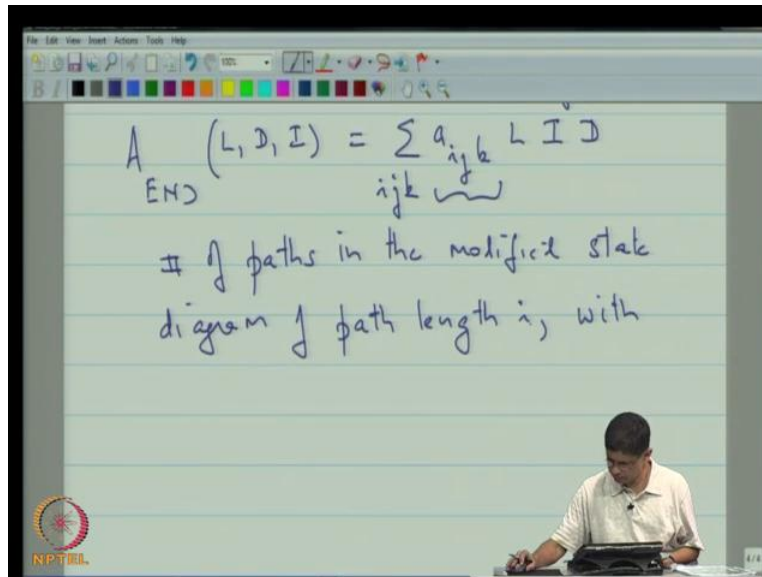
(Refer Slide Time: 09:34)



I to the e 2, so e 2 is the hamming weight of the input and the exponent of d, if, I have D to the e 3 then, the e 3 is the hamming weight of the output. For example, this indicates that m when going from 1 0 state one zero state to the 1 1 state, you needed 1 input symbol; that particular; input symbol message symbol happen to be 1 and the hamming weight of the output is 1 that

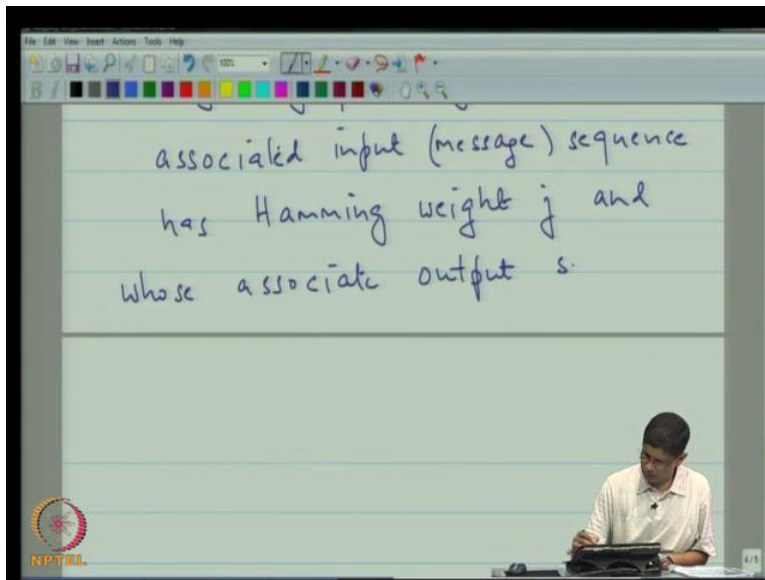
was, it is actually; and you can actually, compute these; you can compute generating functions and worked. We are interested in expression like this.

(Refer Slide Time: 10:45)



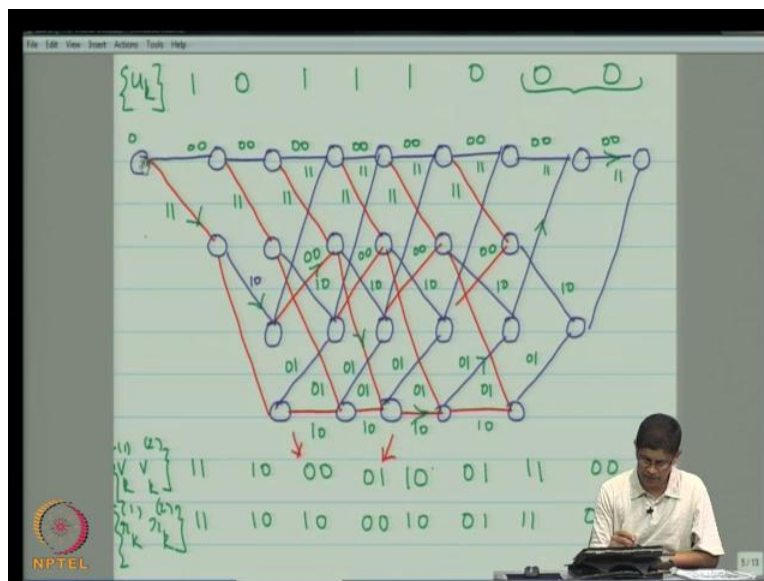
You want to compute; our interest is in computing for states  $S$  equal to are let me, say, computing for the end state the power series which is  $A$  end of  $L, D, I$ , which is the sum  $a_{ijk} L^i I^j D^k$  to the  $i$   $D$  to the  $j$  and  $I$  to the  $j$   $i$  to the  $j$  and  $D$  to the  $k$  and this sum is over all;  $i, j, k$  where is this represent here, what is this says?

(Refer Slide Time: 13:22)



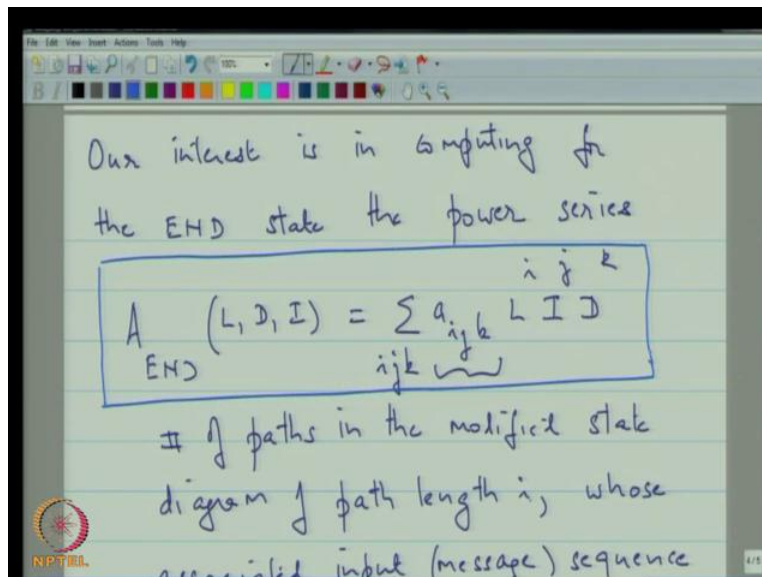
That a  $i, j, k$  is **is** the number; this is the number of paths in the modified state diagram of path length  $i$ , with associated or rather whose is the associated, whose associated input message sequence has Hamming weight  $j$  and whose associated output sequence has Hamming weight  $k$ . Perhaps, I should just show in a example.

(Refer Slide Time: 14:55)



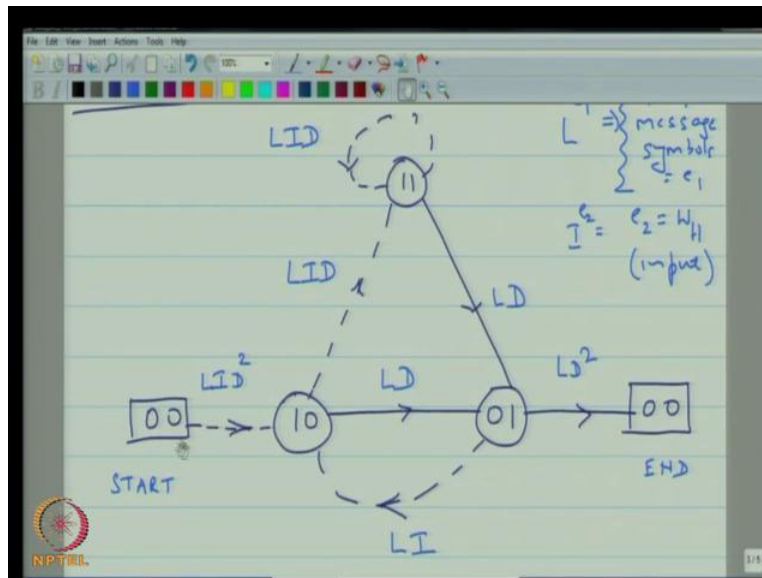
Let go back to seven, for example, supposing I have to just look at this particular portion of the trellis. I have a path this starts from the all 0s state in goes back to the all 0 state. This could be a start that could be the end, the path has the length three and the number of message symbols; that are 1; this is 1. This should correspond to  $L^3$  and the corresponding, output sequence is hamming weight 2 plus 1 plus 2 5, it would correspond to an  $L^3$  defined term that is give you some idea; I hope and further state diagram.

(Refer Slide Time: 15:57)



You can actually; compute this recursively in the sense; they are interested in knowing what is the generating function? So, this power series is sometimes call the generating function because it provides all the data actually, need. In a very compact single function this is also power series is also called generating function.

(Refer Slide Time: 16:20)



In this diagram, you can actually; we are aiming; our ultimate aim is to compute the generating function associated with this end node. What we can actually do? Is we can write down expression that relate generating function at each of the intermediate nodes and from that, we can actually compute this and the methodology, is very similar to solving systems of linear equation and I think once actually, write this down. You will have a better understanding do that, I just a quick note that power series. Such as A end L, I, D are also called generating functions.

(Refer Slide Time: 17:53)

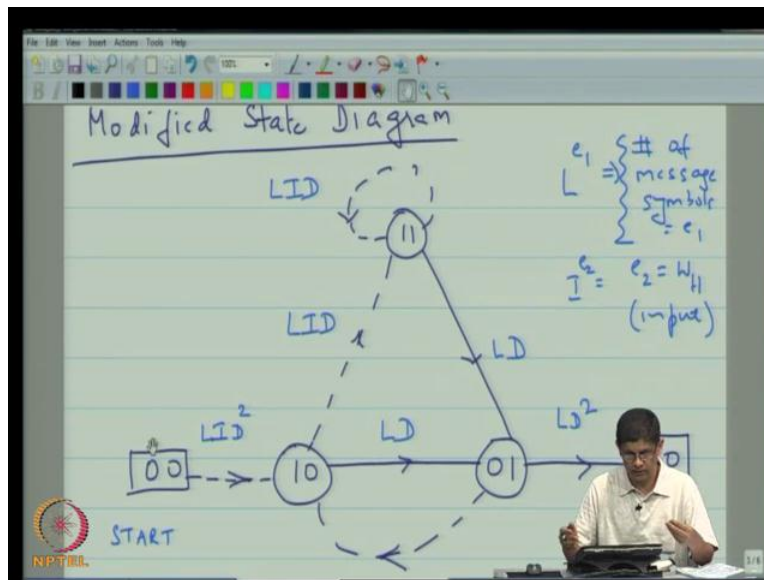
$$\begin{bmatrix} A_{10} \\ A_{11} \\ A_{01} \end{bmatrix} = \begin{bmatrix} 0 & 0 & LI \\ LID & LID & 0 \\ LD & LD & 0 \end{bmatrix} \begin{bmatrix} A_{10} \\ A_{11} \\ A_{01} \end{bmatrix} + \begin{bmatrix} LID^2 \\ 0 \\ 0 \end{bmatrix}$$

$A_{10} \equiv$  abbreviation for  $A_{10}(L, I, D)$



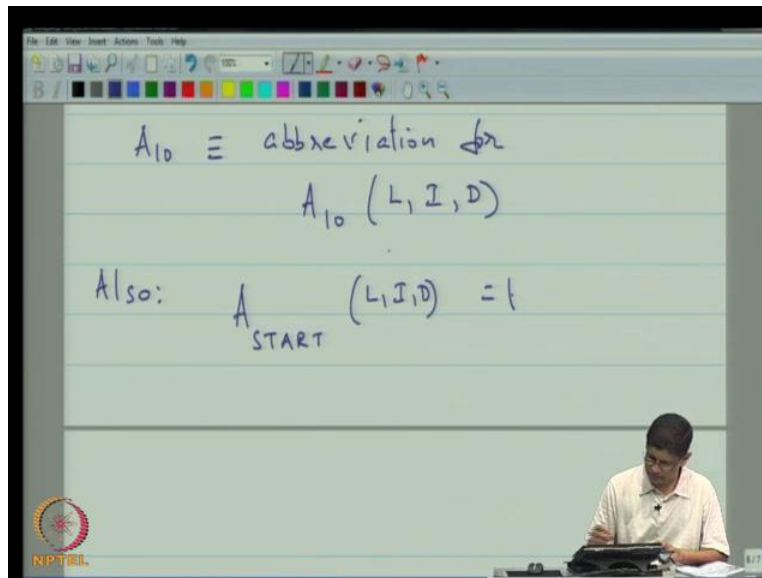
What is the equation that I want to setup; the equation are like to setup is this and I am going to  $A 1 0 A 1 1 A 0 1$  is equal to  $0 0 L I L I D L I D 0 L D L D 0 A 1 0 A 1 1 A 0 1$  plus  $L I D$  squared  $0 0$ . Let see, I can explain this for you by the way, I guess it should be the obvious, but when  $A 1 0$  this is just in abbreviation for  $A 1 0 L I D$ , all of these are generating function on both sides and we just seen that the generating function are interrelated. So,  $A 1 0$  is  $L I$  times  $A 0 1$  plus  $L I D$  squared, try to remember that  $A 1 0$  is  $L I$  times  $A 0 1$  plus  $L I D$  squared.

(Refer Slide Time: 19:52)



Let us go back here, to the modified state diagram  $A 1 0$  and  $A 1 0$ . We see that here, is  $L I D$  squared because we assume that the start, the generating function of the starting node is just one because this is only; a path of 0 length; 0 input weight; zero hamming weight is going from the start node back to the start node; this start node is initialize to 1 then in terms of that, what the generating function of this is 1 times  $L I D$  squared plus whatever, is there in a  $0 1$  multiplied by this corresponding term which is  $L I$  is  $0 1$  times  $L I$ .

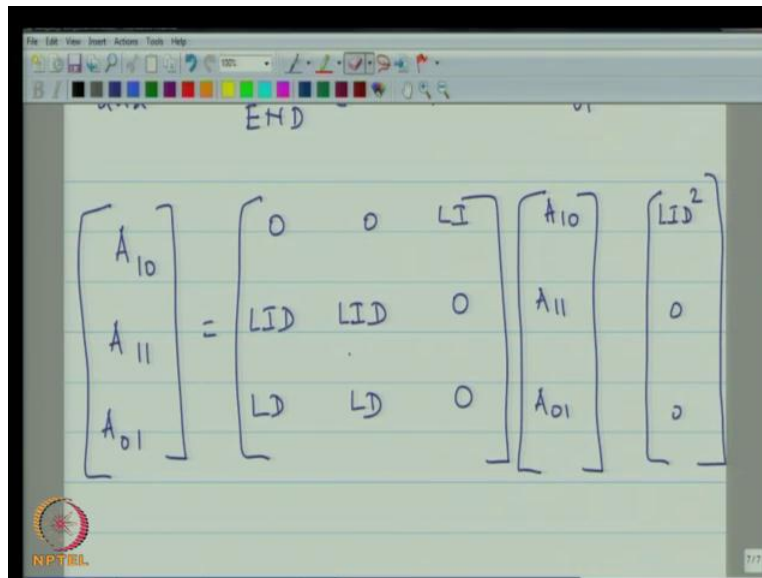
(Refer Slide Time: 20:41)



What we actually seen; in addition to whatever; you see here, we also have that also: we have that A start L,I,D is just 1 and A end L, I, D is equal to is the generating function for, in this figure. Final the generating function for this state 0 1; I just multiplied, by LD squared to get the generating function for A end. The reason being that the only the reason being that, the only input to the end state is precisely 0 1. A end L, I, D is equal to LD squared times A 0 1 times L, I, D. If you look at this equation; this is very much like a system of linear equation because what you can do here? Is that, you can bring all the unknowns on to one side you can bring all the unknowns to one side and then, you have a matrix times this vector equal to this, you might say; that is a matrix, but all these entries are polynomial, narrating that going to causing trouble, it actually; does not because lot of the techniques are linear algebra even carry through even in linear algebra used to working were the matrices our real number or our complex numbers.

Whatever, we discuss after made you comfortable and welcome with matrices is was entries are either 0 or 1 binary. You can also work with matrices over rings here, a dealing with the ring of polynomials in several variables and you can also do a lot of linear algebra over this, what we really a talking about and the set of power series in several variables actually; forms a field once you go the field all notions; all the rules of linear algebra carry through do not be to shock.

(Refer Slide Time: 23:24)



The image shows a digital whiteboard with a toolbar at the top. The word "END" is written at the top center. The main content is a handwritten matrix equation:

$$\begin{bmatrix} A_{10} \\ A_{11} \\ A_{01} \end{bmatrix} = \begin{bmatrix} 0 & 0 & LI \\ LID & LID & 0 \\ LD & LD & 0 \end{bmatrix} \begin{bmatrix} A_{10} \\ A_{11} \\ A_{01} \end{bmatrix} \begin{bmatrix} LID^2 \\ 0 \\ 0 \end{bmatrix}$$

The NPTEL logo is visible in the bottom left corner of the whiteboard area.

I am dealing with system of linear equation by the entries are actually; themselves polynomials because they can be manipulated in the same way as you can manipulate real in complex numbers from this ,What I can do is I can collect this equation to this side. So, let me do one think will be copy this equation.

First and I want to bring; I want to put the qualities here, let me make this the equal sign and in which case what I should do is?

(Refer Slide Time: 24:43)

$$\begin{bmatrix} 1 & 0 & LI \\ LID & LID & 0 \\ LD & LD & 0 \end{bmatrix} \begin{bmatrix} A_{10} \\ A_{11} \\ A_{01} \end{bmatrix} = \begin{bmatrix} LID^2 \\ 0 \\ 0 \end{bmatrix}$$

I should have on this side i minus this matrix the for that, what I will actually have is 1 and I will have entries. Put then and put in the correct entries will have a minus and will A 0 minus LID 1 minus LID 0 minus LD minus LD and 1.

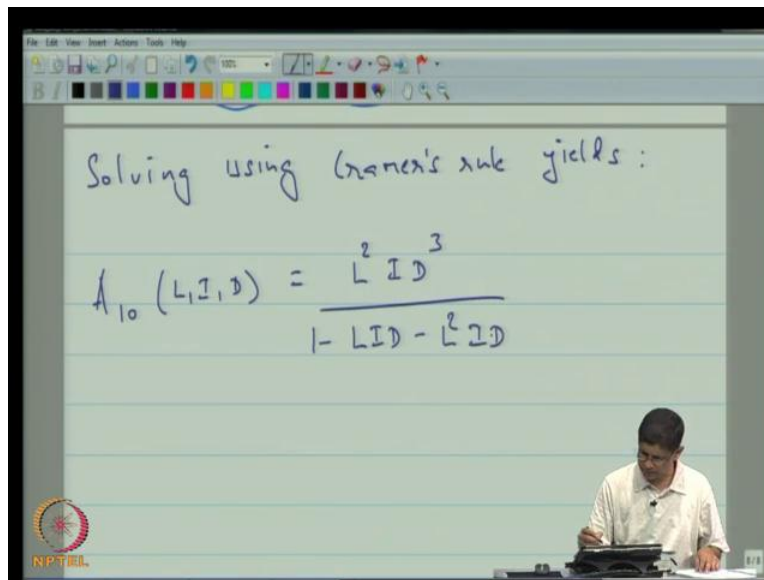
(Refer Slide Time: 25:20)

$$\begin{bmatrix} -LID & -LID & 0 \\ -LD & -LD & 1 \end{bmatrix} \begin{bmatrix} A_{11} \\ A_{01} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$x = Ax + b$$
$$\Rightarrow [I - A]x = b$$

The basic idea is just this perhaps just explain, a little bit better. The idea is that we had something of the form  $x$  is equal to  $Ax$  plus  $b$  what, I simply; did was to write  $I - A$   $x$  is

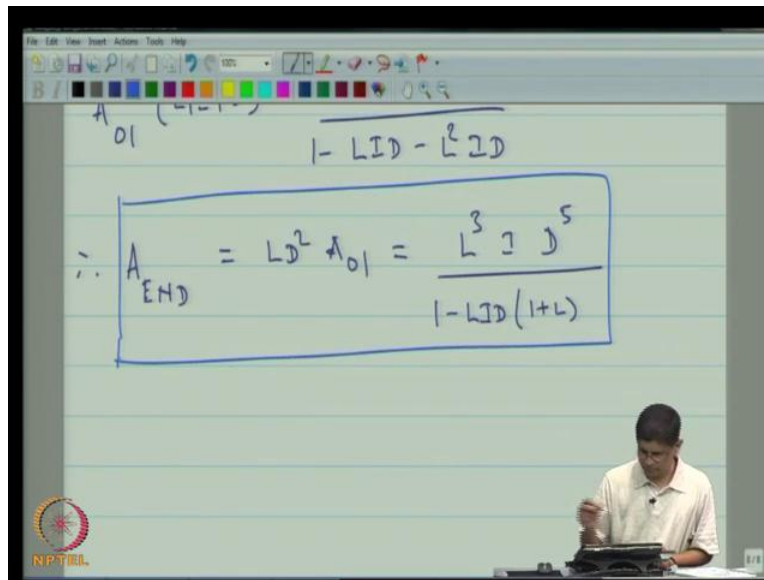
equal to b that is was be actually, did nothing more complicated what, you are seen here is nothing , but the impact of writing this i minus a here, is your I minus A and you see that you have exactly, system of linear equation there are, three unknown and there are three equations and actually, you can solve for these by using just as you would the three complex numbers.

(Refer Slide Time: 26:30)



If, you do that solving using cramer's rule yields gives as that A 1 0 LID is equal to L squared ID cube upon 1 minus LID minus L squared ID, this 0 1 A 0 1.

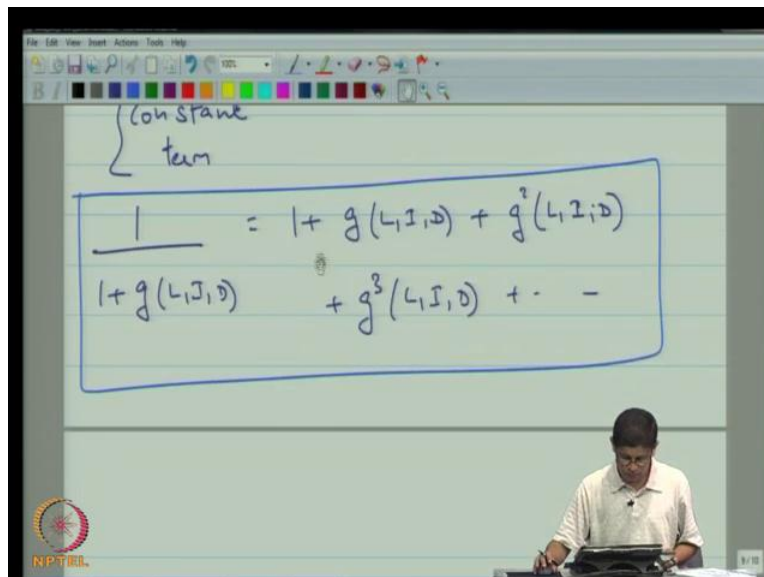
(Refer Slide Time: 27:25)



A screenshot of a digital whiteboard showing a handwritten derivation. At the top, the initial area is labeled  $A_{01}$  and the denominator is  $1 - LID - L^2 D$ . The main equation is enclosed in a blue box and reads: 
$$\therefore A_{END} = LD^2 A_{01} = \frac{L^3 ID^5}{1 - LID(1+L)}$$

Therefore, A end LID which is equal to LD squared A 0 1 is equal to L cubed ID to the 5 divided by 1 minus LID into 1 plus L. This is what we have mention that, we were interested in path enumeration how exactly, does this help us enumerated path any time.

(Refer Slide Time: 28:25)

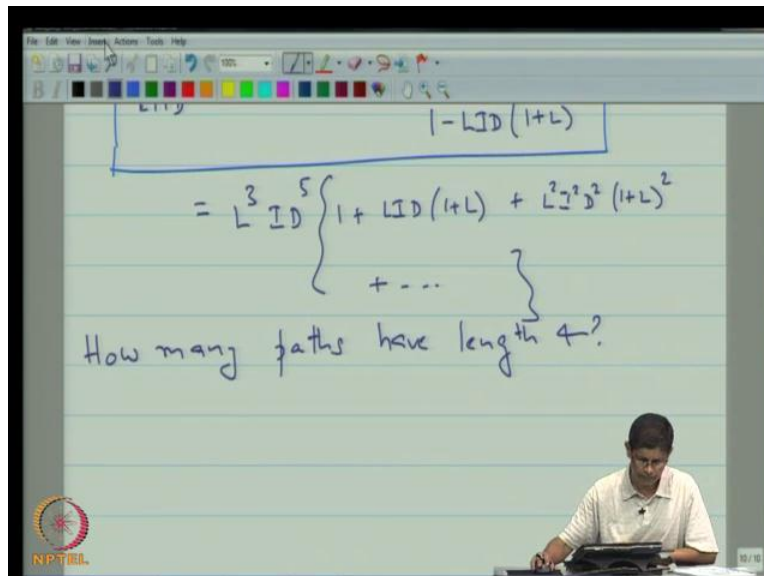


A screenshot of a digital whiteboard showing a handwritten expansion. A bracket on the left labels the expression as a "constant term". The main equation is enclosed in a blue box and reads: 
$$\frac{1}{1 + g(L, I, D)} = 1 + g(L, I, D) + g^2(L, I, D) + g^3(L, I, D) + \dots$$

Now, any time you have in expressions, so I will just make an come on the side here, in note any expression of the form  $1$  upon  $g$   $LID$  where there is, no constant term with no constant term can be expanded did as follows is equal to  $1$  plus  $g$   $LID$  plus  $g$  squared  $LID$  plus  $g$  cubed  $LID$  and you can always expand; this and you are same how do, we know the that works, it is very easy to see because all you have to do to prove this is just to multiply.

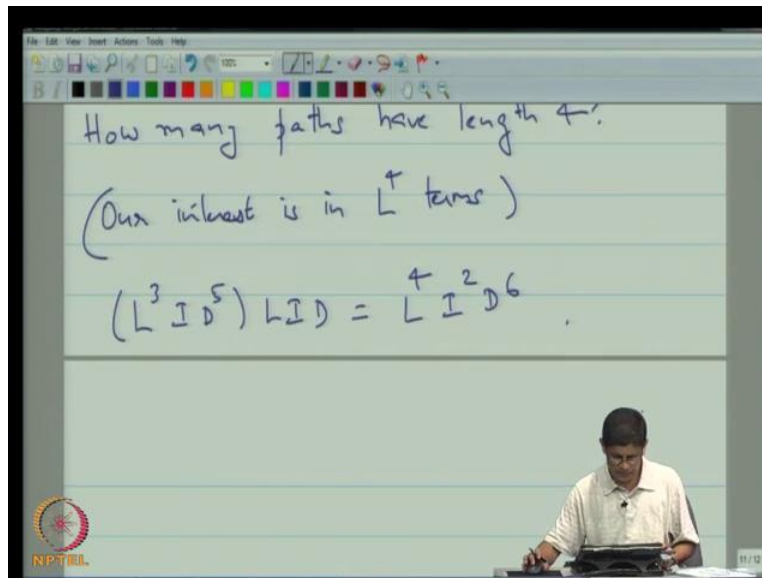
We would just take to prove this you just have to show that if, you take this denominator and multiply the right hand side by the denominator, you will get one and you can check the lot of cancellation take place until your left with one if .You use that here, then you will see that, this let, did produce this equation and evaluated.

(Refer Slide Time: 30:45)



This is equal to  $L$  cubed  $ID$  to the 5 into  $1$  plus  $LID$  into one plus  $L$  plus  $L$  squared  $I$  squared  $D$  squared into  $1$  plus  $L$  the hole squared and supposing, I was interested in how many paths have length 4.

(Refer Slide Time: 32:00)



The image shows a video lecture interface. At the top, there is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Actions', 'Tools', and 'Help'. Below the menu is a toolbar with various drawing tools. The main area is a whiteboard with the following handwritten text:

How many paths have length 4?  
(Our interest is in  $L^4$  terms)  
 $(L^3 I D^5) L I D = L^4 I^2 D^6$

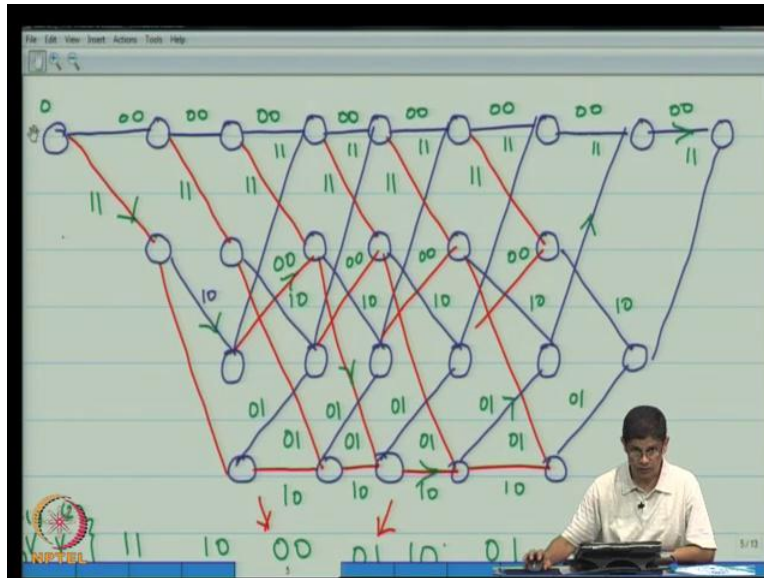
In the bottom right corner, a lecturer is visible, sitting at a desk with a laptop. The NPTEL logo is in the bottom left corner of the whiteboard area.

That means that we looking for that terms into L, the 4 are interest is thus is in the terms involving L to the 4 in look from here, in see which terms can actually; contribute and give you L to the 4; you can see all the terms the third term in beyond the exponent of L is L cubed plus L to 5 that too large to get the very large the contribution, can come from the first the differs two terms; first terms only gives a contribution in L cubed it is only the second term.

This L cubed ID 5 times LID terms so you have L cubed ID 5 times and LID, this gives you L to the 4 I squared D to the six and the quotient is 1, which means that what is that means there is, one path in the trellis .Whose length is 4 which are information weight to and hamming weight 6 lets started remember L four I squared D 6.

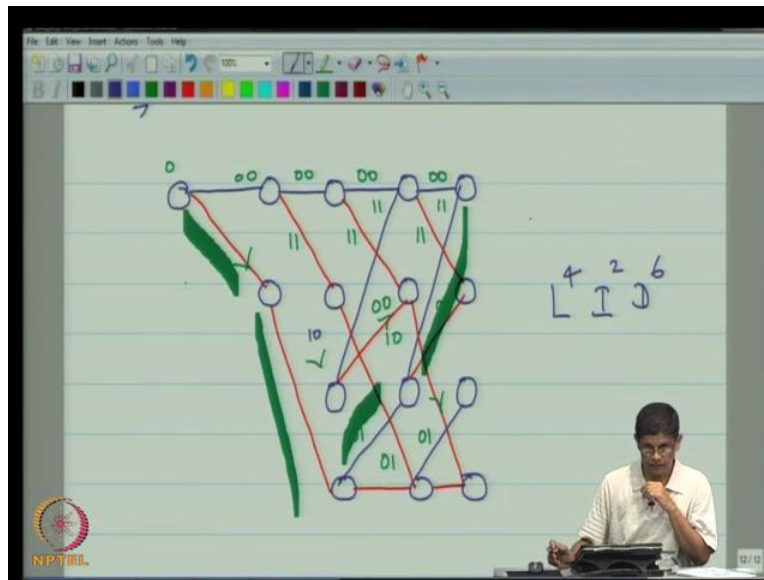


(Refer Slide Time: 33:41)



And go back in check if, that was the case we here, that means there looking at those paths whose length is 4 this start from all 0 state back to the all 0 state. We want to start from the all 0 state we want to get back here, and we want to path length to be four and we want to revisit 0 state in between, We can start here and we want to return here; We have to come here; we have to go from here to here correct and you can see thus the basically; one way of going from here, you would go one, two, three, four and that is the only way to actually, return back to the all 0 state in 4 steps, the length of the path of course is four.

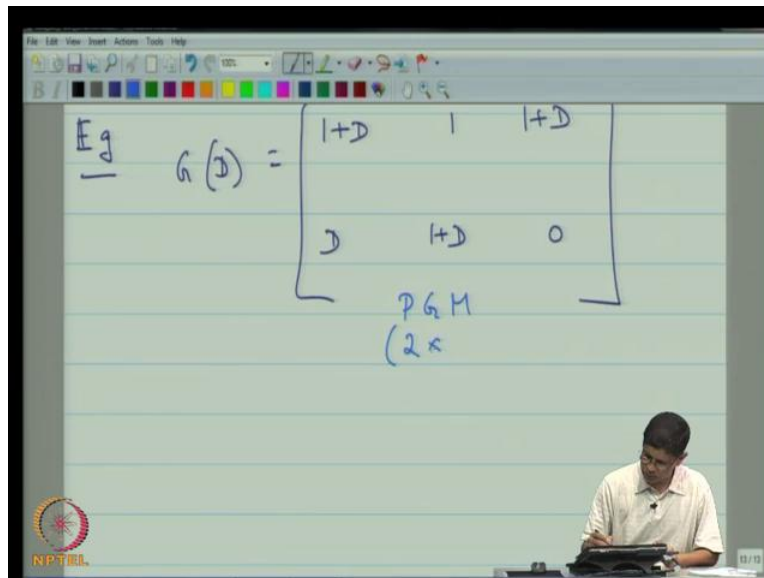
(Refer Slide Time: 35:48)



We request that there are two inputs which are one the output sequence weight is two this one correspondence to an output of 0 1 that is three this is 0 1 that is four and that is one; that is six that explains. Why we actually have  $1^4 2^2 3^6$  so perhaps right things to do here is to cut out that part of trellis and the reproduce it let us see, if I can do that I am going to take this part of trellis there, We go and paste with here and just for vivificate let us get read of some of the terms that we do not here, we only want 1, 2, 3, 4.

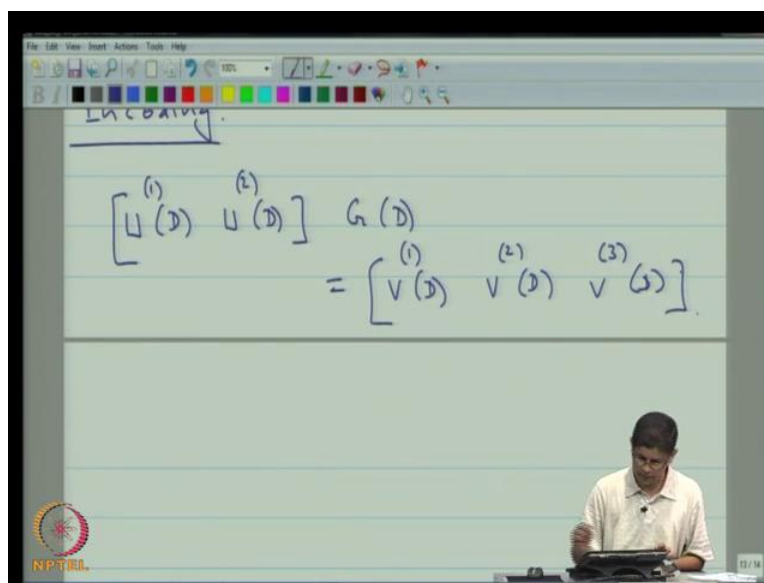
I just care about this and we are interest only in that one path let me highlight that in perhaps green that means are interested in the path. Which goes like this, that is the four paths segmented. You can see that this correspondence to here to 1 to the four I to the two and d to the six right the other a treatment of convolutional code is use to be that when, I would teach convolutional code that, I would spent a great many lectures, but since that time the subject has progress and there are, other types of course that we need to spent time on for this reason are treatment of convolutional code compare to what it was in the pass is somewhat compress. What I will do next is talk about the general convolutional encoder. What is that mean well up to know we took up to his specific example and we kept working with it and this example had the feature that they was just one input, but multiple outputs, but they general there could be more than input lines that is want I mean by general convolutional code.

(Refer Slide Time: 37:55)



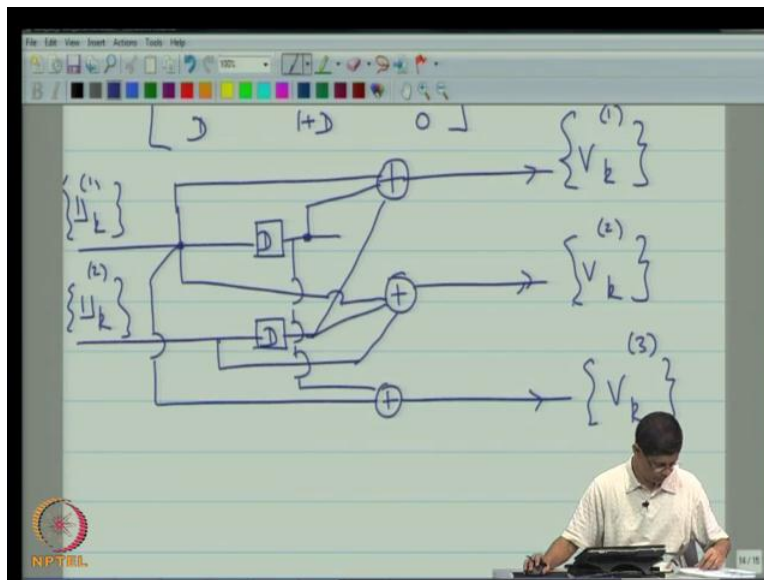
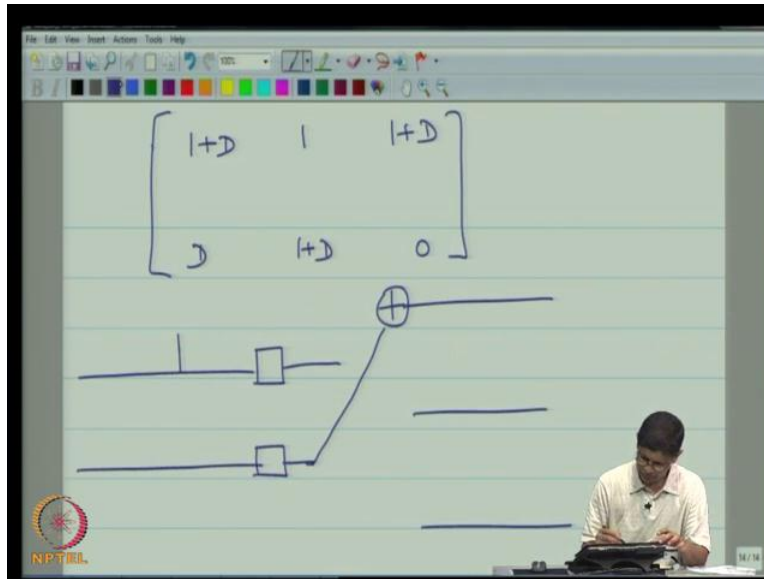
General rate  $k$  by  $n$  convolutional code and again will begin this with an example and I will just give, you the example in terms of the polynomial generated matrix and I just write down there it is  $1$  plus  $D$   $1$   $1$  plus  $D$   $D$   $1$  plus  $D$  and  $0$ , what this means, this is the polynomial generated matrix in these case and this will notice two by three that means there are, two inputs and three outputs.

(Refer Slide Time: 39:20)



The encoding operation would be represented by, excuse me  $U_1$  of  $D$   $U_2$  of  $D$  is equal to rather times the polynomial generated matrix  $G$  of  $D$  is equal to  $V_1$  of  $D$   $V_2$  of  $D$   $V_3$  of  $D$ . This is how you would actually encode, that they three outputs and two inputs. The natural to ask the question, what is the encoder look like in this case lets go ahead in draw the encoder because after all that is the way in which introduce, the first encoder to you.

(Refer Slide Time: 41:00)



This was are generated matrix and just keeping the on these will be able to draw the encoder there are, input lines that there is a single memory unit on each other of the two input lines. There are two input lines like this and there are three outputs one two and three let see missing an entry here that should be a one. What is that if ,we look at this the first output is 1 plus D times the first input plus D times this here, you will derived input from before and after the shift register here and also the delayed output here , this will be the output there, that is one output the second is 1 and 1 plus D , that will gets its output from here, as well as from here and the third output is 1 plus D and 0 which means, what which means there is output is derived only from the two inputs of the first input line and input line.

Encoder will a clock, we put D this is your first input  $U_1$  of k and this is  $U_2$  of k, this is  $V_1$  of k this is  $V_2$  of k and this is  $V_3$  of k 2 inputs three outputs ok and other parameter to introduce, here is the total memory of the convolutional encoder. What we mean by that is this convolutional code has two inputs and three outputs on each input line you have z number of shift register, you look at the number of shift register on each of the input line and you had and that is the total that is the total memory of the encoder. You can see that the total memory in this case is actually two.

(Refer Slide Time: 45:00)

$$G(D) = \begin{bmatrix} g_{11}(D) & \dots & g_{1n}(D) \\ g_{k1}(D) & \dots & g_{kn}(D) \end{bmatrix}$$

$$\nu = \sum_{i=1}^k \max_{1 \leq j \leq n} \{ \nu_{ij} \}$$

So, if the memory of the convolutional encoder, if a polynomial matrix  $g$  of  $d$  is let say  $g$   $1 \times 1$  of  $D$   $g$   $1 \times n$  of  $d$   $g$   $k \times 1$  of  $d$   $g$   $k \times n$  of  $d$  and what you define is you define  $\nu$  we to the sum  $i$  is equal to one to  $k$  max where  $j$  line between, one and  $n$  of  $\nu_{ij}$ .

(Refer Slide Time: 46:29)

In the example:

$$G(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1+D & 0 \end{bmatrix}$$

$$[\nu_{ij}] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

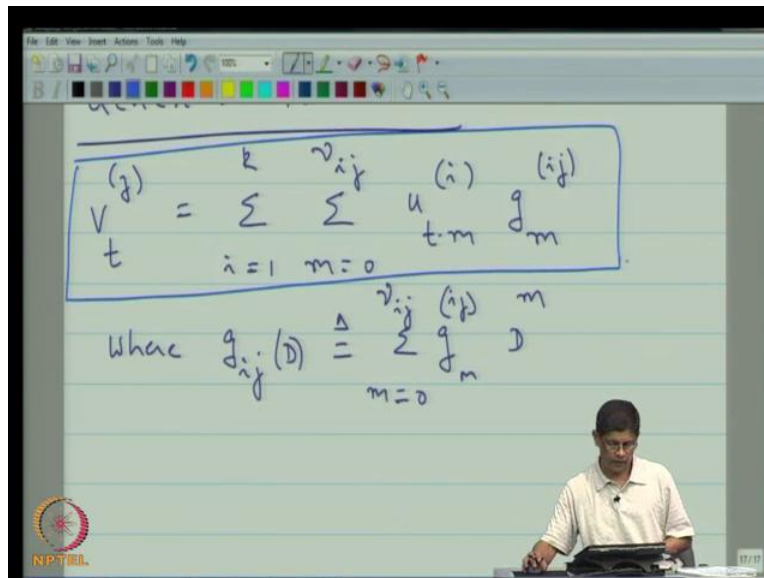
$$[v_{ij}] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -\infty \end{bmatrix}$$

$$\therefore v = \max\{1, 1, 1\} + \max\{1, 1, -\infty\}$$

$$= 1 + 1 = 2.$$

So this is what the memory is, in the example, we are 1 plus D D 1 plus D and then D 1 plus D plus 0. So are,  $v_{ij}$ . If, you put these in a matrix, then you actually get 1 1 1 1 1 minus infinitive, you can also think of the  $u_{ij}$  as being the degree of the polynomial  $g_{ij}$  of D and since, we have zero polynomial here and this instance will regard, the degree as minus infinitive. This is the  $u_{ij}$  and what this is the total memory is nothing but to take the max an each row and add of the max. Therefore in this case, therefore  $u$  is equal to max of 1 1 1 plus max of 1 1 minus infinitive, which is 1 plus 1 which is 2 so the memory in the convolutional code is 2. In simple terms, what the  $u_{ij}$  is telling you in memory units are on each input line and then you had at all the memory units and then you will get memory of the convolutional code. In terms, just to fill in the gaps; I just too quick derivation here, our input output relationship in general, in the case of convolution code.

(Refer Slide Time: 48:50)

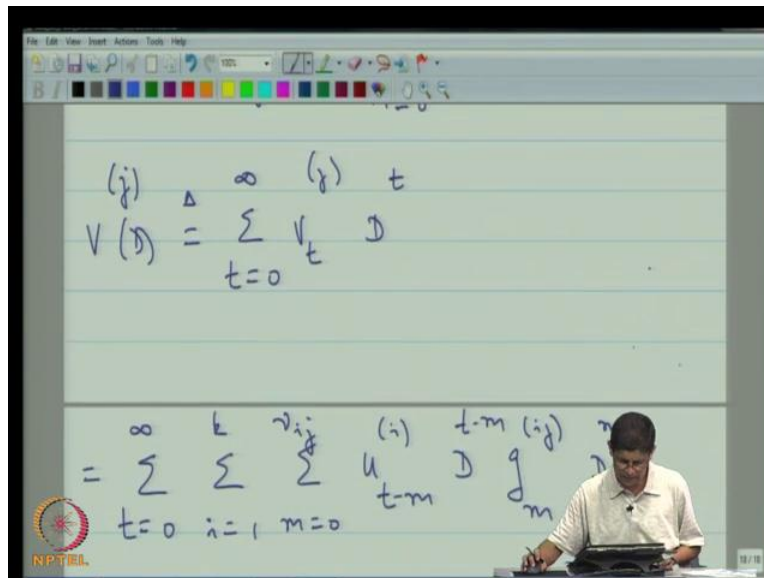


General input output relation is giving by V, the j of the output of t is the sum i is equal to 1 to k; the sum m is equal to 0 to nu i j u of t minus m of i g i j m .Where g i j of D is defined as the sum g i j of m D to the m, m goes from zero to nu i j.

This is getting a little bit formal and that, thus the reason by started out the examples, this is a general input output relationship all it saying that look, if you are looking at j th output at the time instant t then, it is going to be a function of the k input streams of course, and along each input stream is going depend **to d** a function of not jus, the current value of the input along the input stream, but also some past values and how for back in the past you go? Is govern by this degree factor nu i j.



(Refer Slide Time: 51:55)



This is where you have a convolution relationship, that is comes into play. From this, it is not very hard to derive that there is, if you take  $V(j)$  of  $D$ , then thus by definition the sum  $t$  goes from 0 to infinite  $V(t)$  of  $D$  to the  $t$  and that is, the sum  $t$  goes from 0 to infinite; the sum  $i$  goes from 1 to  $k$ ; the sum  $m$  goes from 0 to  $n_{ij}$  of  $u_{t-m}$   $D^{t-m}$   $g_m$  of  $m$   $D$  to the  $m$ . All the down here, this is the definition write here of the output pass this and then, we have this input output relationship are plugged in  $j$ , we should  $j$  of  $t$  here to arrive at this and the other little thing there. I did was that, I took this  $D$  to the  $t$  term and I spitted up into the  $D$  to the  $t$  minus  $m$  and  $D$  to the  $m$  term and that is the trick that you often to the  $g$  employee in transforms whenever, you want to show that the convolution results in multiplication in the transform domain.

(Refer Slide Time: 53:10)

$$= \sum_{t=0}^{\infty} \sum_{i=1}^k \sum_{m=0}^{\infty} v_{ij}^{(i)} u_{t-m}^{(i)} g_{ij}(D)$$

$$= \sum_{i=1}^k \left[ \sum_{m=0}^{\infty} g_{ij}(D) \right] \left[ \sum_{t=0}^{\infty} u_{t-m}^{(i)} \right]$$

This can be written as the sum  $i$  is equal to 1 to  $k$ , I am interchanging, just back real. In the terms this is, what is you get? And these are standard manipulations and that gives.

(Refer Slide Time: 54:03)

$$= \sum_{i=1}^k \left[ \sum_{m=0}^{\infty} g_{ij}(D) \right] \left[ \sum_{t=0}^{\infty} u_{t-m}^{(i)} \right]$$

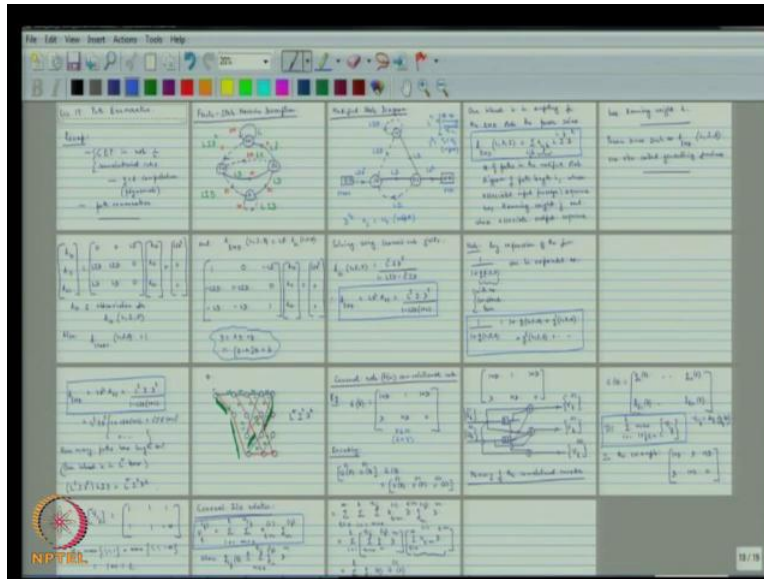
$$= \sum_{i=1}^k g_{ij}(D) u^{(i)}(D)$$

$$\Rightarrow \left[ v^{(1)}(D) \dots \right]$$

You the sum  $i$  is equal to 1 to  $k$ , this is nothing but the polynomial generated matrix, I can write that is  $g_{ij}$  of  $D$  and this thing here now, this is the input power series; the only confusion might arise because you will say well  $t$  minus  $m$ . When  $t$  is 0 for the going to run negative subscript,

what should I do then? We will just assume that, for any negative time instance the input was 0. If you assumed that, in this just reduces to  $U_i$  of  $D$ . This is how? The polynomial generated matrix cross out. This is actually, needs to the equation, that we used  $V_1$  of  $D$   $V_n$  of  $D$  is  $U_1$  of  $d$   $U_k$  of  $d$  times the polynomial generated matrix. I think this is the good point to stop, what we did today and let me just zoom out, that we can see a lecture glands today.

(Refer Slide Time: 55:54)



We are topic was path enumeration, and path enumeration we mean the path in the trellis diagram and each of path corresponds to the code word in the convolution code, we should how we use generating function techniques to actually a enumerate path in the trellis diagram. Then after that this turns to the case of the general rate  $k$  by  $n$  convolution code, and what do this look like, I give you and we are beginning to develop, this through an example to an illustrate of example. What will do after this will read down catastrophic error condition for the general case.

And then after that will also talk about how you use the trellis to decode, a convolution code over not diagonally symmetrical channel done earlier, but actually more practical channel the **(( ))** weight goes to an channel. And in fact, as I think I mention sometime back the power of the convolution code lies in ability to actually decode over a narrative weight goes in a channel. This is something that a typical block code may not have, and that is why this code is very popular in practice, but will see the next time. So, thank you.