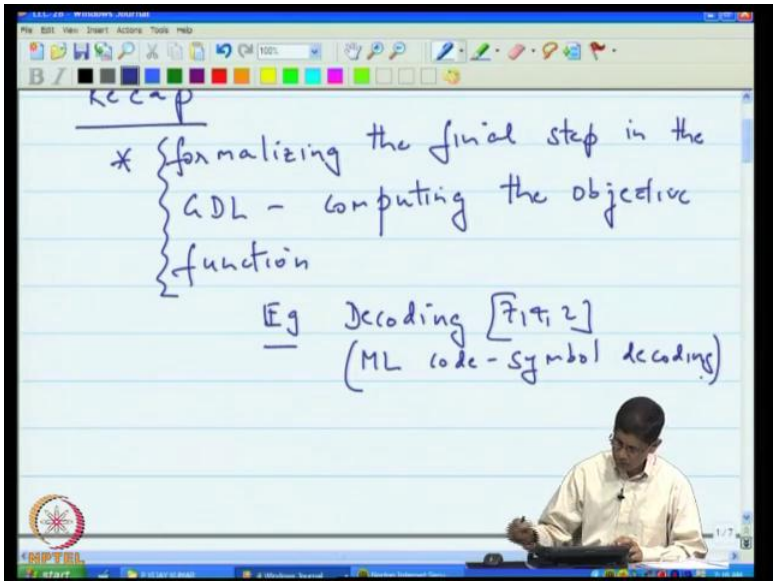
Error Correcting Codes
Prof. Dr. P Vijay Kumar
Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore

Lecture No. # 28
ML-Code Symbol Decoding of the Convolutional Code

Welcome back, this is our lecture number 28 and unfortunate to have two of IAC student sitting in the lecture to keeping on toes. These are students who are actually taking my current class on campus, which is more or less replica of class that I am taking here. So, today, we are going to do maximum likelihood Code Symbol Decoding of the Convolutional Code, which is a subject that we started in the last towards the end of the last lecture. So, let me begin by recapping.

(Refer Slide Time: 00:46)

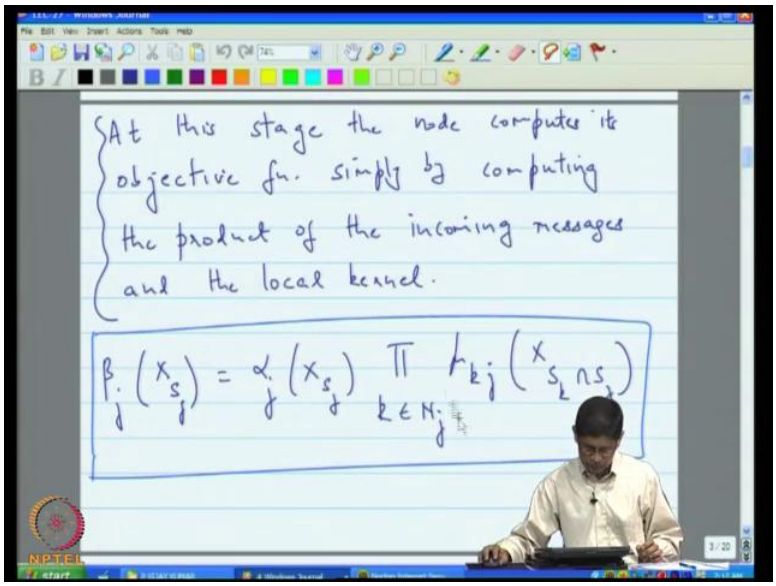


So, first thing we did last time was to formulize the final step in implementing the GDL. Remember that the steps were that you first identify what the local domains and kernels are. And then you attempt to organize local domains into a junction tree and assuming that has happened, then you pass messages along their junction tree.

Now, in the single vertex case you orient all the edges towards the node, whose objective function you want to compute and then you pass messages. More generally if you want to

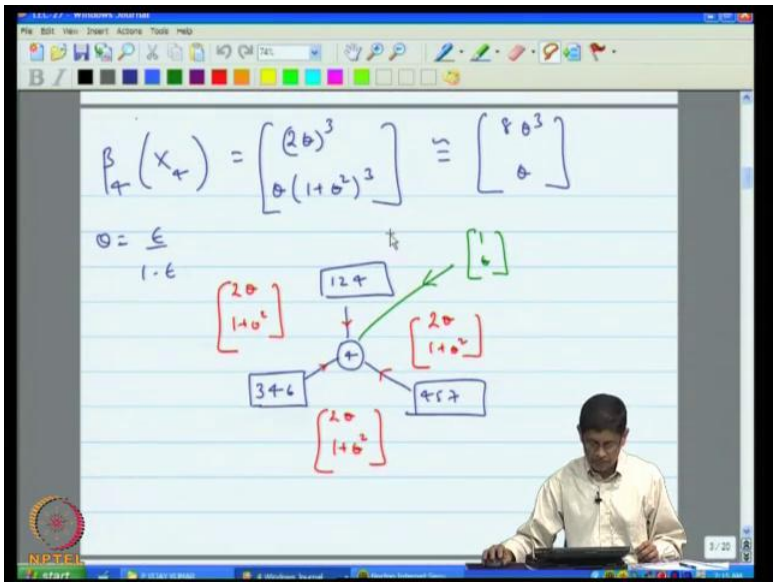
compute multiple objective functions, then you can think of almost any sequence of message passing. You want the only thing is that node is ready to compute objective function only when it is heard from directly or indirectly, maybe other node in the graph. So, then in the beginning of lecture, we talk about the last step implementing in the GDL which is computing the objective function at a single node. And maybe I just quickly show you be expression on that one. So, that that expression is exactly this.

(Refer Slide Time: 01:57)



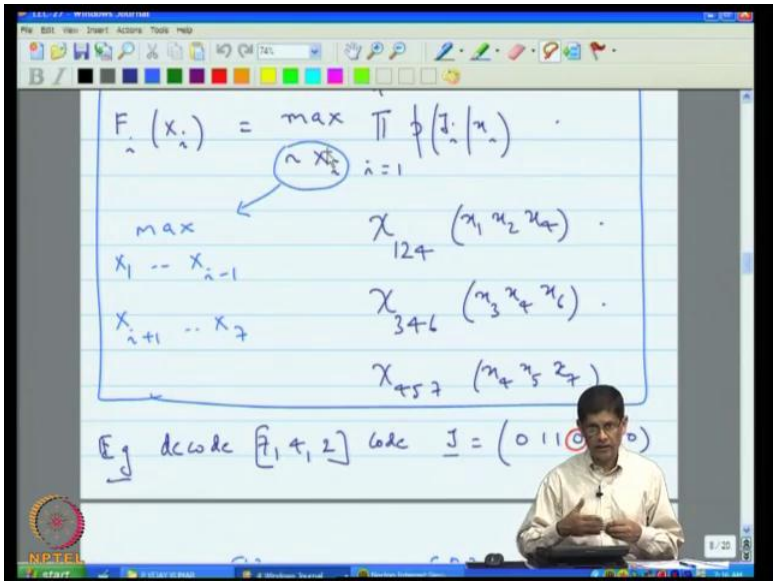
So, this is the expression that we had for computing objective function, basically it is just simply computing the product of all the incoming messages together with local kernel. There is no marginalization required at this stage; and after that we turned around attention to...

(Refer Slide Time: 02:14)



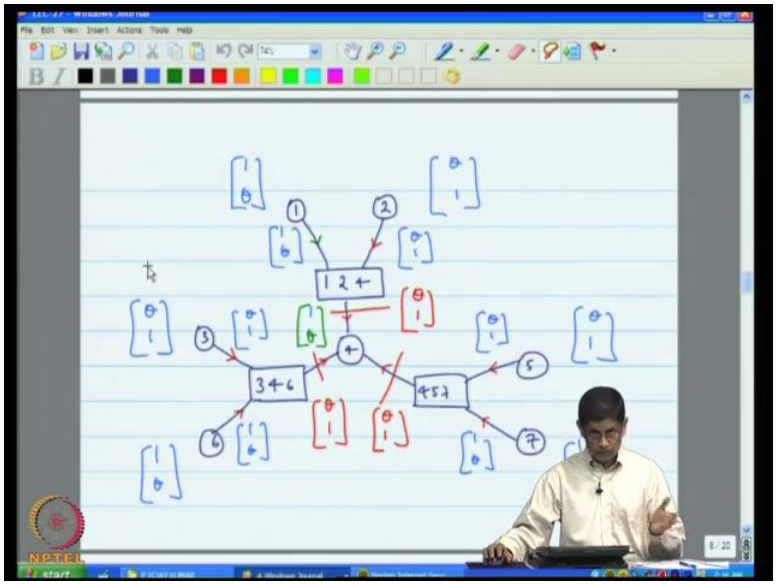
This is for an example, where we computing the objective function for the node x 4 and based on that we made decision, and this was an instance of maximum likelihood code symbol decoding. And after that we turned around attention to maximum likelihood code word decoding of the 742.

(Refer Slide Time: 02:35)



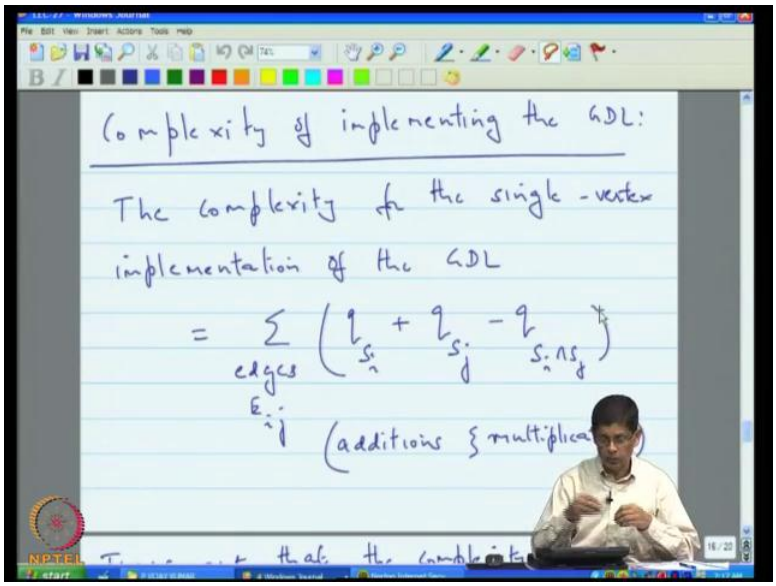
And we said, if you think about the only change that it replacing summation sign by the maximum. So, whenever you have a opportunity to marginalize in implementing the GDL in passing messages then instead marginalizing by assuming you marginalize using the max operator.

(Refer Slide Time: 02:54)



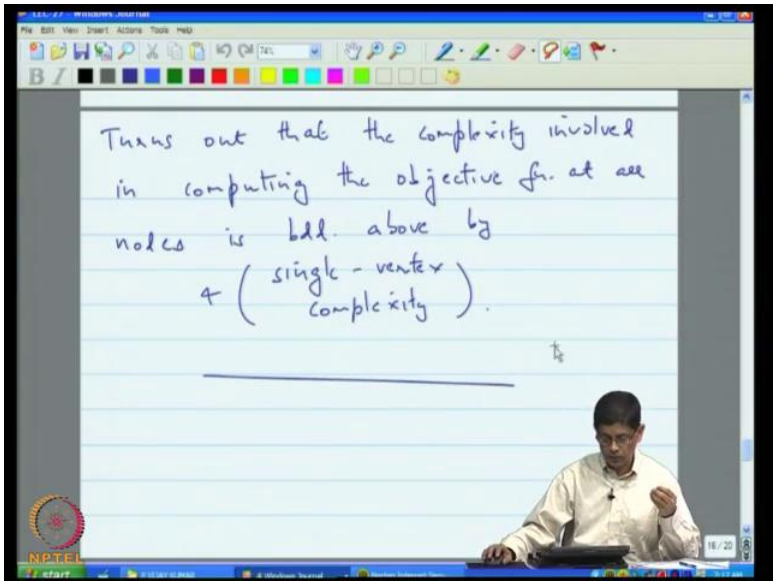
However, that no difference and in fact that computation that simple, because only to comparison and multiplications and term by term multiplication. So, we did that, we and we computed, we solved again the simple vertex problem for this 742 decoding of the code symbol x 4, but this is easily extended to decoding the other symbols as well.

(Refer Slide Time: 03:24)



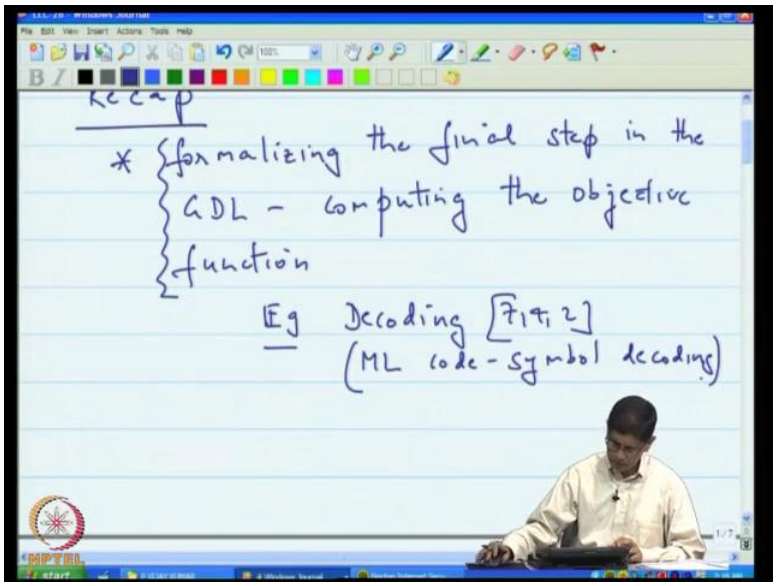
Then towards end the last lecture, I (( )) and I give your reference started talking about the complexity of implement in GDL. And you can show that the complexity of implementing single vertex version, that is of implementing the GDL, when you want complete the objective function is at single node given by this expression.

(Refer Slide Time: 03:42)



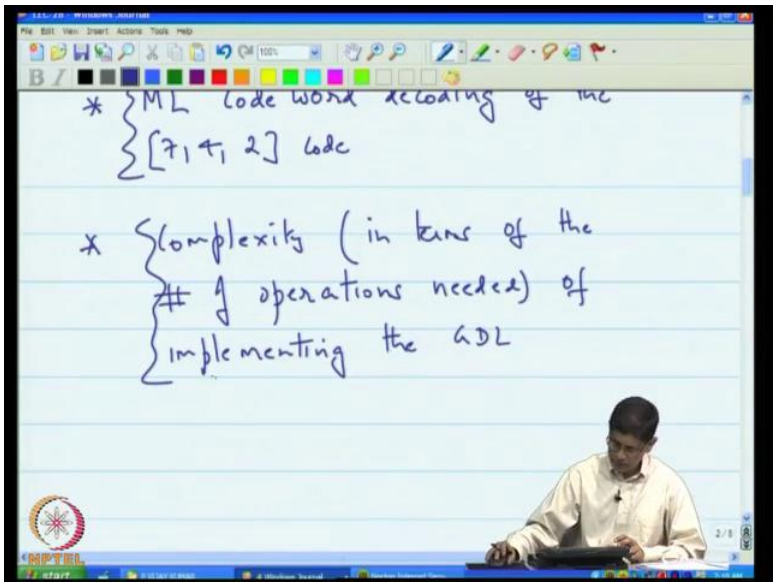
For the case when you have multiple nodes, it is more complicated when you want to compute the objective function at multiple nodes it is more complicated. But other hand the good news that you can that up to by just four times the single vertex complexity, then you tend or attention the maximum likely decoding of convolutional codes, but we will go through this again today.

(Refer Slide Time: 04:08)



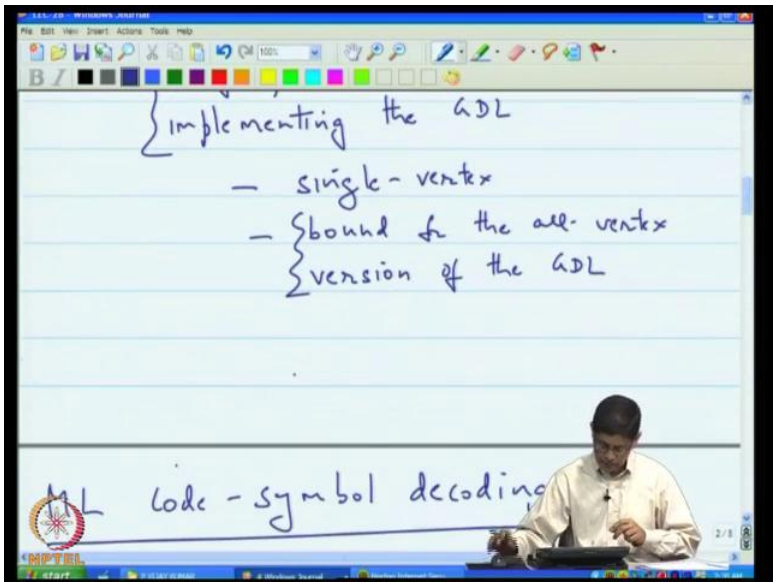
Then, so we did an example here, decoding 7, 4, 2; we implied maximum likelihood code symbol decoding.

(Refer Slide Time: 04:41)



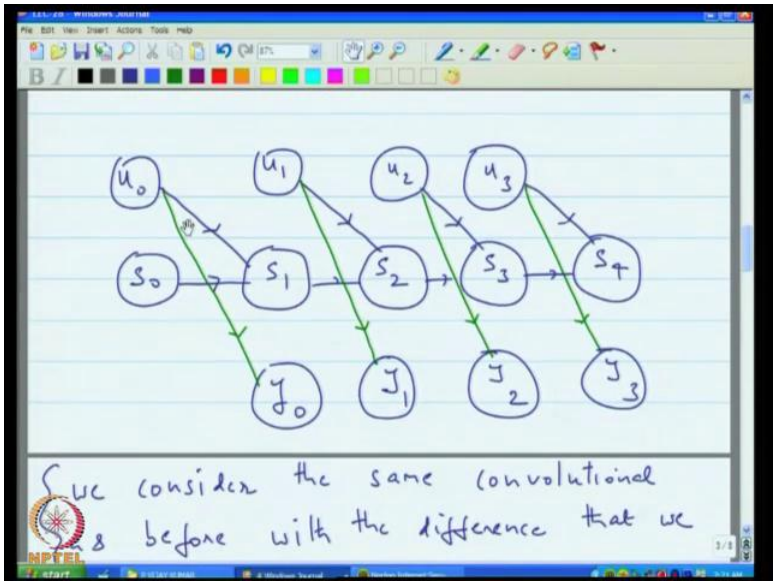
Then after that then after that we looked at the problem of maximum likelihood code word decoding of the 7, 4, 2 code. We looked at complexity as measured by number of addition and multiplication or trace number of operation, because the operation could be addition and multiplication, but it could also be taking the max and multiplication. So, I just say complexity in terms of the number of operations needed, so complexity of implementing the GDL.

(Refer Slide Time: 05:57)



And we give exact count for the single vertex case and the multiple, actually we given x second estimate for all vertex case. And then finally, we look at maximum likelihood code symbol of a decoding of convolutional code, but here I thought it exist look take a quick look that all over again I want repeat that.

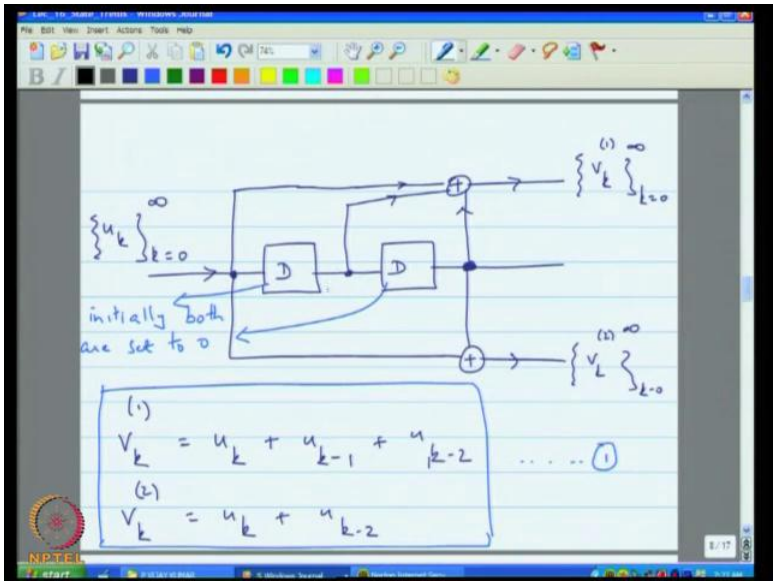
(Refer Slide Time: 06:47)



So, the convolutional code, we know that basically and I am taking a very short block link convolutional code. So, there are in this case, there are 4 message symbols this is the state sequence through, which the convolutional code encoder is given and this is your output sequence. Now, only I written by 0, in general by 0 is an n tuple for example, if you looking at rate 1 by n convolutional code each y 0 is a string of an binary symbol.

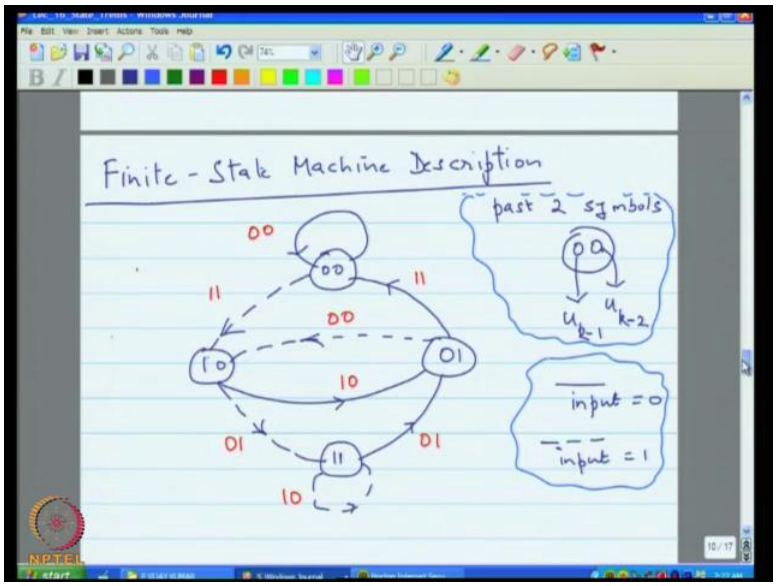
But, we would not get in that, that is get into much detail and that might (( )) the main concept and assemble of story holds for the symbols u 0 for the present. Lets us assume is rate 1 by n codes that these are bandwidth symbols, if you like keep in mind that the convolutional code that we developed, that we introduced right at the beginning, we check I can pull that up quickly.

(Refer Slide Time: 08:24)



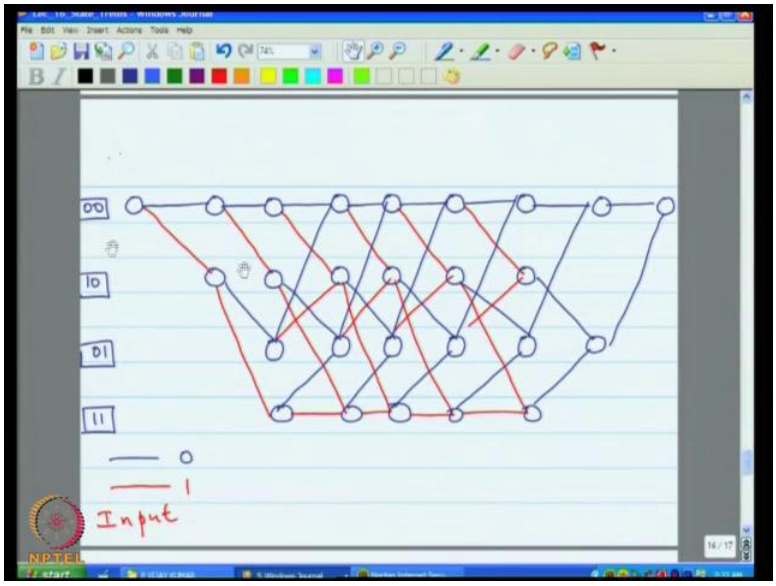
So, if you like I can think about this particular convolutional code, where there in one input sequence. And then there are two output sequences.

(Refer Slide Time: 08:35)



And this state sequence, there are referring to is this state sequence. And is the convolutional code as the message symbol sequence coming it is steps from one state to another, but we prefer to think of it in terms for Taylor's.

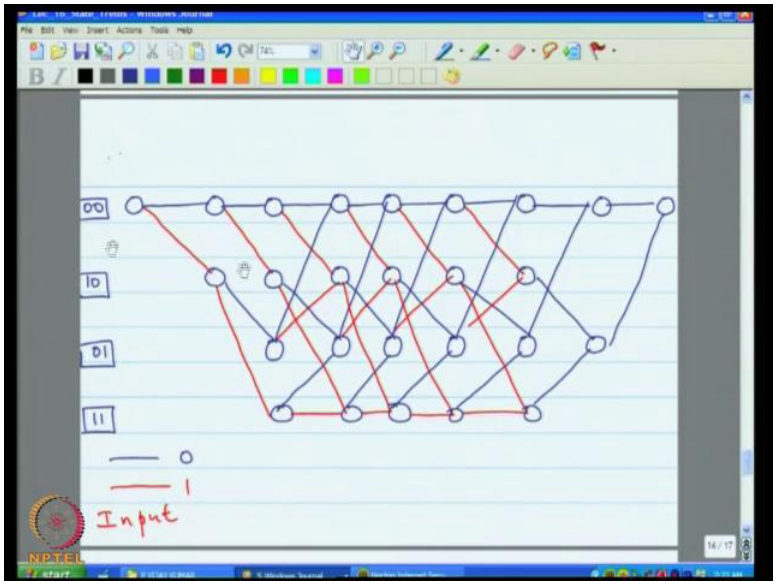
(Refer Slide Time: 08:46)



So, like this except the representation of Taylor's is more abstract here, we put down every possible state there we just assign a variable and leave it like that. So, at k time instance is

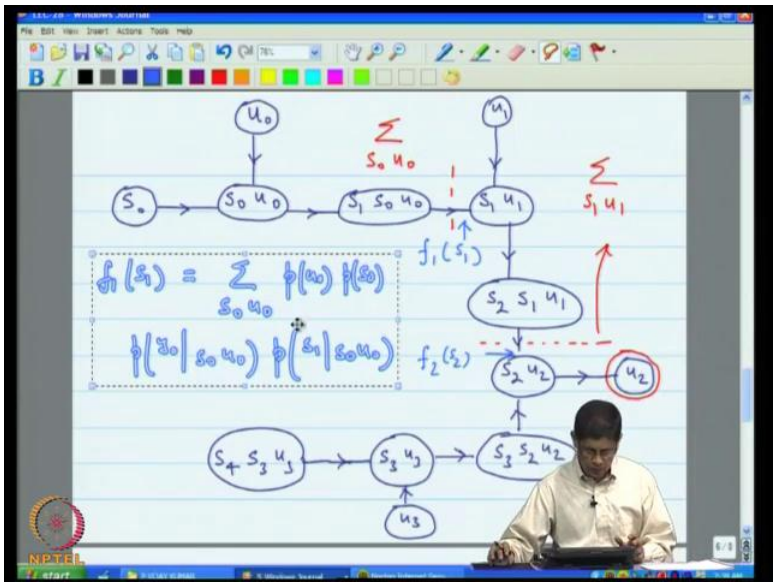
possible state s k, k we plot all the states there we do not, so that is a difference. So, then the maximum likelihood code symbol decoding needs to compute this particular objective function. So, we actually write this down and from this it becomes pairing that we have an MPF problem.

(Refer Slide Time: 09:28)



And in fact, in an earlier lecture we did put down we did put down the local domains and kernels, so let me carry those over here, these for this particular problem this for the local domains and local kernels. In other question is how do you organize this in to a junction tree. So, there are four of this lying between 0 and 3, this 1 here and again here derive value ranges from 0 to 3 in these cases. So, in our, we have 13 states and so that means that... So, I have some jangling of space to do fit all of this, but...

(Refer Slide Time: 11:12)



So, let us try to organize these into junction tree. So, it does not really matter where you start I will start with the state s 0, and now we look at these local domain and say what should I connect to and where going to use prims greedy algorithm. So, we pick a node which has maximum intersection this and for example, there is one which is s, so from here you see s 0 u 0, so you pick that. So, that has intersection 1, edge weight 1 and then you see that you can try gone u 0 here and I am going to do this rather quickly, because we have quite a bit of ground to cover and already seen how to construct a junction tree earlier.

So, next I am going to pick s 1, s 0, u 0 and that is forced, because there is an intersection of two here, that would give mw a edge weight of two. And then I have some, some looking for something that has an s 1 and I see that I can have in s 1 u 1. So, I am going to do that I could also connected, so this is right and I said I am going to (( )) on this you can figure this out on your own. So, this will then be s 1, u 1 and then u 1 and then you have s 2, s 1 u 1, which you can connect to this, then you have s 1 u 1, s 2 u 2 to which you can connect u 2.

And similarly now you would have s 3, s 2 u 2, which leads to which leads to s 2, u 2 and you can connect u 2 here sorry. This is not s 2 u 2 this would be s 3 u 3, s 3 u 3 and u 3. So, let us see have got I should have 13, I have 3, 6, 9, 12. So, we are doing good, so the last state here is then s 4 s 3 u 3. So, this is your junction tree in this case and you can verify that

if you look at the edge weight the sum of the, if you use the test compare the node weight and edge weight you actually find difference is equal to number of variables.

So, this is the junction tree, you could also verify it by projecting onto each of the variables and checking that you get a tree. For example, if you project on to u 1 you will get these three nodes, you project on to u 0 you get these 3 and so on. You project on to s 2 you get these 3. So, it is always a tree when you project in therefore, this is junction tree.

So, now let us consider the let us consider a single vertex well our problem is single vertex in nature because we are interested in computing e f u k. And well in general it is not you want computed for all them message symbols but let me just assume that what were interested in is in a just maximum likelihood code symbol decoding of u k, where k is 2. So, now, of this let assume that k is 2, there then what you do is you as you mentioned earlier oriented all the edges towards node u 2 and pass messages. So, let actually do that.

So, now, you oriented all in towards this in perhaps will these circles in red to indicate that the objective function of interest and now you going to messages. And so this k dul is quite, so in these direction messages of flowing like this and here the flowing like this all right. Let us just see what kind of messages flow in this along this graph, now I want to also indicate where it is we have the marginalization taking place. So, for example, here I am going from here to here it is on injection, so the first marginalization actually takes place here.

So, let us indicate that with puts a dotted line, so that is your first marginalization and what you going to do here is your going to marginalize by summing over s naught and u naught. So, remember the junction tree requires that you pass messages and then message that you pass or in accordance with formula that is you take the product of the incoming messages. And you multiply by the local kernel and then marginalize as necessary here no marginalization is necessary, this is the first instance and we marginalize, the next instance when we marginalize is similarly here.

So, this marginalization that is carried out here and this time this marginalization is over s 1 u 1. Now, if you look and similarly the other direction there are marginalization will come to this later. So, going in this direction which sometimes is called forward direction what we

have passing here, so the message. So, your going to multiply all these local kernels and then you are going to marginalize but what is left is a function of s 1, because thus no u 1, here it is a function of s 1.

So, let us denote that particular function by, so let say the function that is appearing here is f 1 of s 1, let us call that f 1 of s 1. Now this is the function that here, that is a message that being passed along this edge and (( )) there is message that passed here, which is f of f 2 of s 2 and that is taking place here alright. So, now, in this particular case we are interested in the all vertex version of gdl in the single version vertex of GDL that is we want to decode u 2.

But, in general since in many practical application you will want to decode all the message symbols, both this the message passing schedule the people adopt is in our schedule, which has the nature of forward backward algorithm. So, what we to do is you go all the messages passed from here to here as if this for the objective function and then messages are passed all the way back.

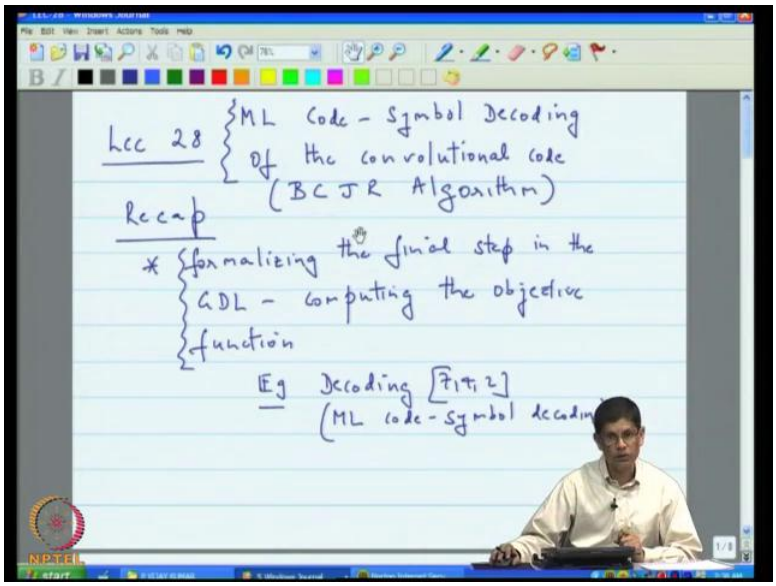
So, this is forward and this is backward, here we doing a hybrid because we are going forward up to here and then here we doing backward. But so; that means, that in the course of computing this get to see a little bit of forward and then little bit of backward computation there actually taking place. But one think I want to point out it that once you set up the junction tree, then basically your algorithm is let down.

Because, now you know what messages to pass and everything is clear by looking this trying to give you some additional inside. So, this point in really we have done will solve the problem of maximum likelihood decoding, because we just code by passing messages. By the way the algorithm that I am teaching you now is called the BCJR algorithm for decoding convolutional codes and it was the case some over may be 10, 15 years ago; there was the paper that appear in the little cheer very in nice paper is described this algorithm.

But, it carried out be algorithm what in isolation whereas, here where coming across this is they are in very mutual way as and outcome of simply implementing the GDL on this particular message passing of dictating by the GDL on this particular junction tree. So, you

see comes are very naturally and I will also mentioned that viterbi decoding also come out naturally from in this framework. So, what will learning here is call the BCJR algorithm.

(Refer Slide Time: 23:51)



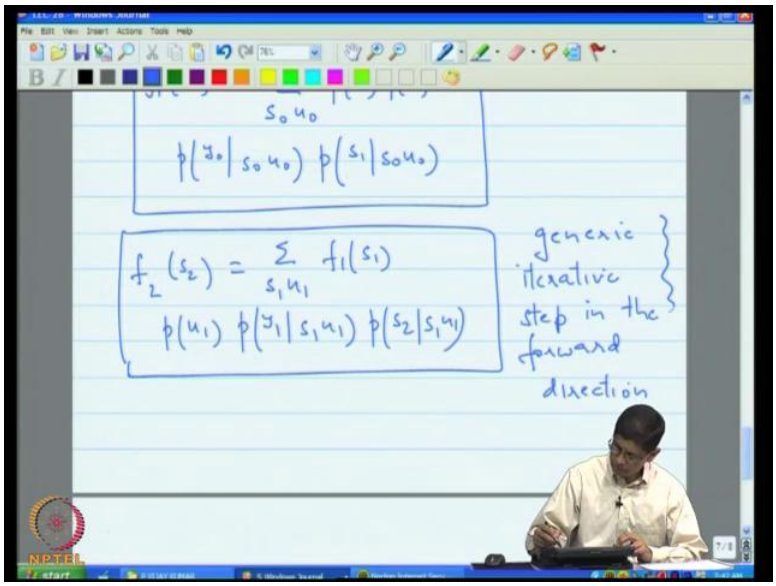
So, I think what I will do I wil go head in that write in the title, which is where it belongs. So, this is the BCJR algorithm. So, the b is stands for Balip bal, this is cook jawline and raveef and balip will actually former student long time back student of IIT Kharagpur alright. So, here we have this now, which you give you inside lefts go-ahead and look at what happens in between the computation of f 1, s 1 and f 2, s 2.

So, may be just to that things are clear I write down the expression for f 1, s 1 write here. So, f 1 s 1 is equal to the marginalization with respect to s naught and u naught of p of u naught p of s naught times p of y naught given s naught u naught and p of s 1 given s naught u naught. So, and I written this but I am actually going to put it over to the next page, because we do not really want this plotter on this page. So, let us move this and it try that once more cut and it paste here.

So, that is the first computation but now let us also similarly put down computation for f 2 s 2. Now, were going in this direction here, so I want to put down the computation of f 2 s 2 that will involve f 1 s 1. Because that is the incoming message here f 2 of s 2 is the

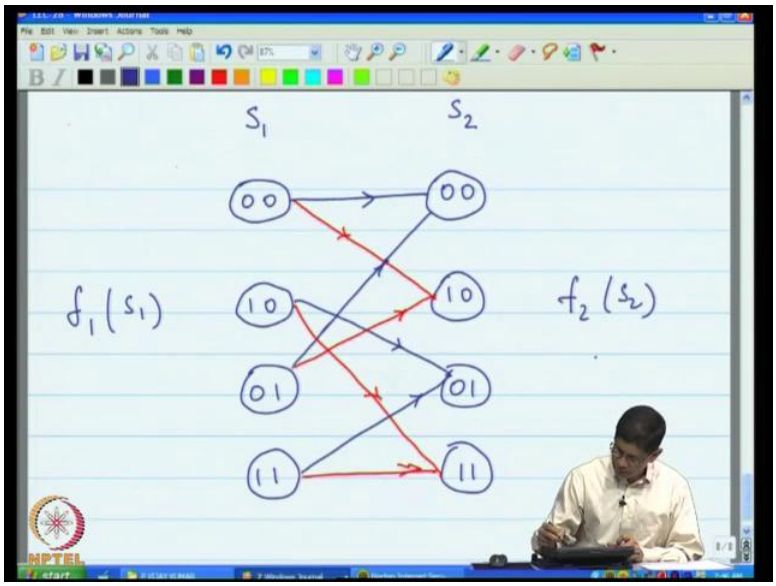
marginalization with respect to s 1 u 1 of I have f 1 s 1, I have p of u 1, I have p of y 1 given s 1 u 1, p of s 2 given s 1 u 1. So, you see that these are just simply local kernels f 1 s 1 is the incoming message here, you want is local kernel here which become message symbol here. The local kernel at this node is precisely this and then local kernel here is this, because this the product of these three is a message here and then you have the local kernel s well I am right. So, again let us a take it the next page.

(Refer Slide Time: 27:22)



So, we have these two expressions all right. So, now, the question is well I want to focus on this, because in some sense this is the generic this is the generic relative steps this is the generic iterative step in the forward direction and to see that I am going to introduce. So, let us will will get try to feel for it, we are the example of our code whose generative polynomial generative matrix was 1 plus d plus d square 1 plus d square. So, if you recall here the Taylor's in that case, we have the Taylor's at perhaps this more than needs let us go back and try it itself. So, in this case just want to single section of the Taylor's.

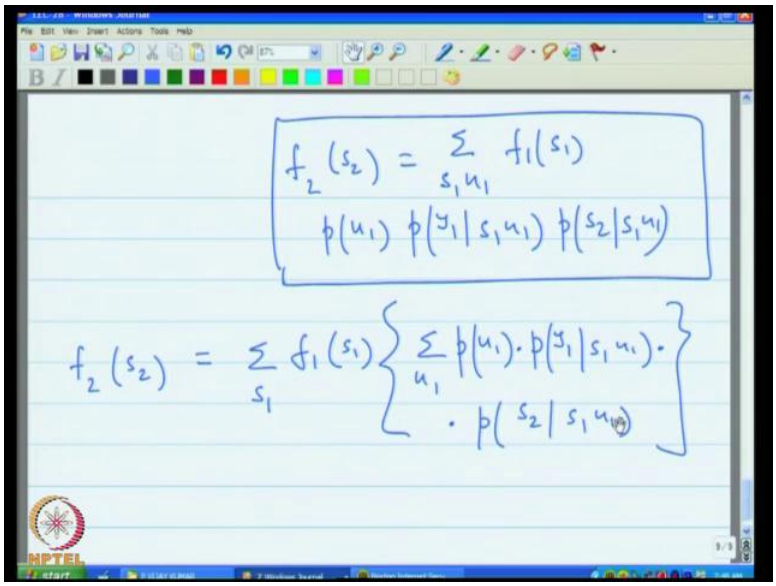
(Refer Slide Time: 29:36)



So, my states are going to be 0 0, 1 0, 0 1, 1 1, and now think of this as the collection of all possible states s 1 and this is the collection of all possible states s 2. And I am going to user older terminology of you go from one to the other. So, you can go with a input of 0 from there or from here and similarly for the other states I think element what I did since drawing on dotted lines is not the easiest thing here I am going to just use it different colour for 1. So, let us as choose red for 1.

So, now, we have done these are all the possible transition and all orient the edges in this direction because we are going to be in forward, just keep in mind. So, what is going on here is that, I want to illustrate do not the percent the message computation between here and here. And it terms out the round in the Taylor's it is equivalent to our computation that is taking you from f 1 of s 1 to f 1 this side, what you coming with this function which is f 1 of s 1. And then what you exit with is a function of s 2, this is what is coming in and this is what going out. But what is the nature of this computation and I came that really, what is happening here it is a matrix multiplication.

(Refer Slide Time: 32:46)



So, let us try to that clear again I am going to our repeat formula this for clarity. So, we have that formula here and I want to first of all, so first I want to rewritten as f 2 of s 2 may be as you choose the blue to be consistent. So, let say that f 2 of s 2 and I want to treat this is as the summation s over s 1 of s 1 of f 1 of s 1. And then this an interior summation which is over u 1 which is p of u 1 p of y 1 given s 1 u 1 times p of s 2 given s 1 u 1 this multiplies this over here and one point I want to notice that if you look here.

Now, p of u 1 is the a priory probability of single one and in general way we assume that the message symbols are equally like to be the 0 or 1 this is the half it is a constant. So, we can ignore this, we pull write out of computation, so take no more attention to this. So, have these are two terms now you have summation over u 1, now in the summation there is a fixed value of s 1 and we have in mind computing this value of for in other value of s 2.

So, in a sense s 1 and s 2 are fixed s two is fixed, because you are asking for the value for the particular s 2 and s 1 is fixed, because you are sitting inside summation for which and then we are looking at terms which s 1 is fixed. Given the pair s 1 and s 2 is exactly one message symbol, this either one message symbol that makes the link between s 1 and s 2 are none. Because that is our convolutional code were right, because when we looked at final state information of the convolutional code here, let say there I pick s 1 to be this and s 2 to

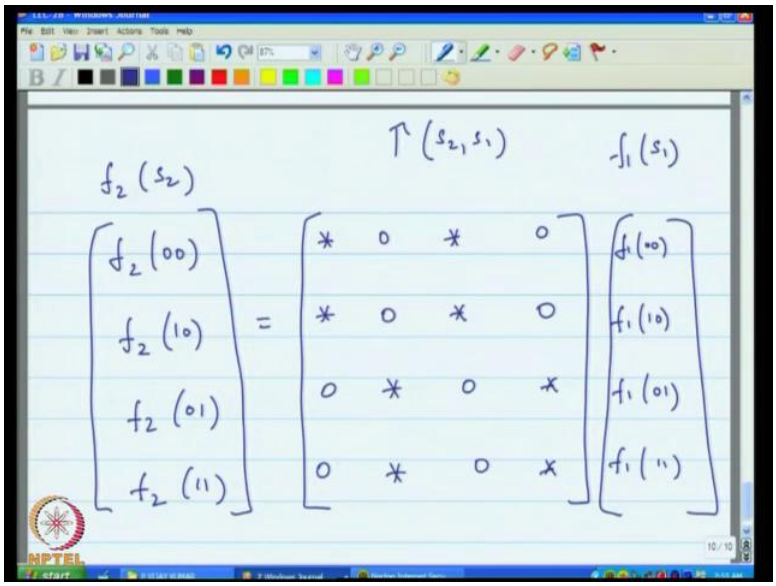
be this, then the only possible message symbol that can taking from here to here is this particular zero.

Now, on the other hand if I pick let say s 1 to be this and s 2 to be this, there is no message symbol that will taking from here to here. So, in that case there is none, so the point is that in this summation there is either one term or no term. And the other think to keep in mind that this function is zero one function. So, only values that it is assume either 0 or 1, so let us make a node of that. So, this is a zero or one valued function, so let us like an indicated function and this is a constant this is equal to one half.

So, basically what you are saying then is that the next outcome of this entire computation is either a particular value of this or zero. And the moment you look at it like that this a entire thing is a function of s 1 and s 2 and so you can few think of little bit more about it you can actually represent this, this computation as this computation can be computed as a matrix multiplication where this inner term here, the inner term here this inner term.

This entire inner term here you can call this as sum gamma of s 2 comma s 1 it is function of both see the moment is sum over u 1 there is no dependence on u 1, so this only a dependence on s 2 and s 1. So, now, you can see that this expression with this entire thing replace by gamma of s 2 s 1 looks like a matrix multiplication and that precisely what it is. So, I just spell their out for you.

(Refer Slide Time: 37:48)



So, then the matrix multiplication with looks something like this, you will have f 2 of s 2 as a result of you computation. So, will compute f 2 of 0 0, f 2 of 1 0, f 2 of 0 1 and then f 2 of 1 1 and this is computed by multiplying a matrix with f 1 of 0 0 etcetera. So, this is the general f 1 of s 1 and this matrix is nothing but your gamma of s 2 s 1. So, you can, so in general, you can try to find out what the entries are, in general the entries this is where the information from the outside channel from the y is actually comes in.

But, what I want to pay attention to where you have zeros and where you do not. So, for example, supposing you think about this first entry the first entry will be nonzero if this transition from 0 0 to the 0 0. And in fact, there is so will have a non-entry here, then is there a transition from 1 0 to 0 0 the next entry would correspond to a transition from here to here and there is none. So, that entry will turn out to be 0, then is there a transition from state 0 1 to 0 0 yes the input is zero.

Remember what happens is that if you are in this current state, then the new symbol comes in from left and push it out from the symbol on right. So, from 0 1 you can transition to 0 0, from 1 0 you cannot because the older symbol will remain as a one. So, that means, that you will have; and similarly will have a 0 here and that is nearly a reflexion of this and the

Taylor's. In the Taylor's you can in the section of Taylor's we see that you can only come to 0 0 from either 0 0 or 0 1 from next.

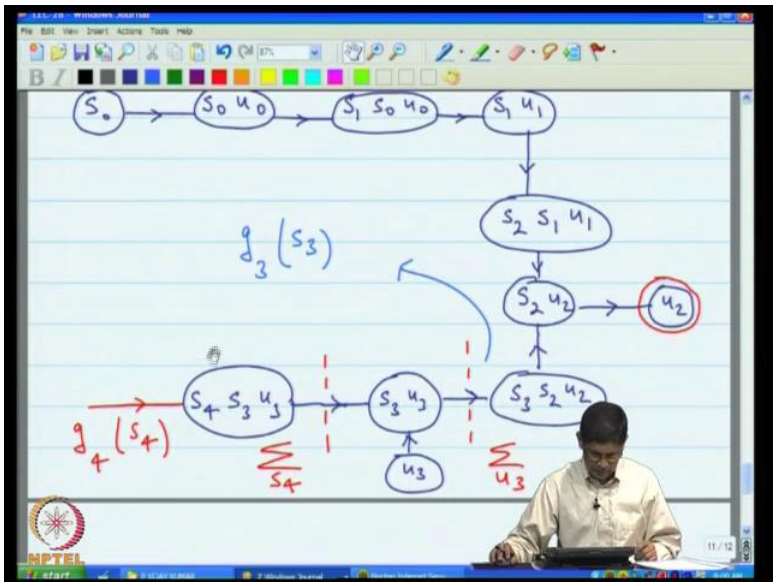
The question is from there when you come if you aided to us 1 0 the answer is again 0 0 or 0 1 this is for 1 0. So, the next entry is also similar you can only come from here or here. So, these entries are zero; now you want to know well from going to our 0 1 where can I come from. So, I am going to 0 1 I can either come from 0 1 or 1 1 so; that means, in this case the star are here and zeros are here and similarly for the last case will find there similar.

So, you are, you can think like a transition matrix and it is going look like this. So in fact, what the GDL does it that it gives you a very compact representation of your algorithm, but since you are passing function, implementing them involved operations certain operation in this particular case its matrix multiplication. So, all that we did here, why is actually point out that the forward request algorithm takes you from here to here is really a matrix multiplication corresponding to transfer of information across one section of the Taylor's.

So, in generally you can actually that if your Taylor's is very long then your just going to hops from one section of Taylor's you are going to keep on hopping and will be multiplying by matrix at a time. The matrix will change from section to section, because it is a function of received symbol, see because the next time encountered something like this you will be hopping from here. Let us say you are continue in the forward direction you would have f 2 of s 2 here.

And then you would have a f 3 of s 3 here and then the particular entries would depend upon the local kernel here which was p of y 1 given s 1 u 1 here it will be p of y 2 given s 2 u 2. So, this is the what max, the matrix from section to section, but answer (( )) just matrix multiplication. So, you can actually see that, so the forward algorithm easily described; now there is also the backward algorithm, right so your also going in the backward direction, so what happened there. So, I think to show that, then you again copy this and this time, let me erase this, because now are interest in the other part of the algorithm.

(Refer Slide Time: 43:43)

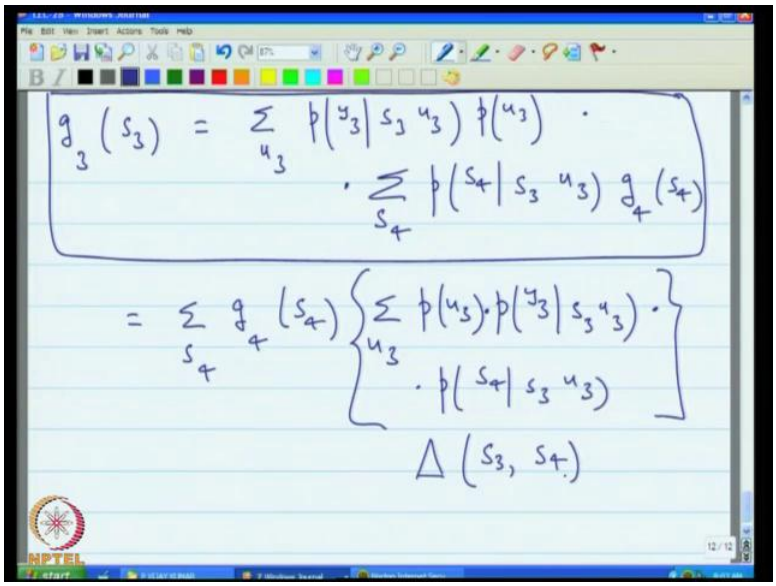


So, now, you want pay attention to the how the algorithm is forecasting in the reverse direction. In the reverse direction the marginalization we have to first identify, where the marginalization taking place and here is a little bit difference. So, you see that in going from here to here, there is a marginalization over over s 4. So, here you are summing over s 4, then this also marginalization over here, which is the summation over u 3.

Now in general, the disadvantage have in this reverse direction is that, I do not have in enough states to the left of meant able to give you the feel. But let us say that in general that there is a function that incoming, this is say that in a longer Taylor's I would in general have incoming some function g 4 of s 4. And what would happened is that this point here this point here after marginalize with respect to u 3 what I now have is g 3 of s 3. So, I am going backward, so the the subscript of the state if decreasing.

So, let us write down the expression for this. So, then you have now this is vertices in our in our particular set up they only this many states there it is a function in coming in, there is no such function. We can assume at be one where in generally you can imagine that they be coming, because if you along that this is you would have repetitions of this kind of thing.

(Refer Slide Time: 45:59)



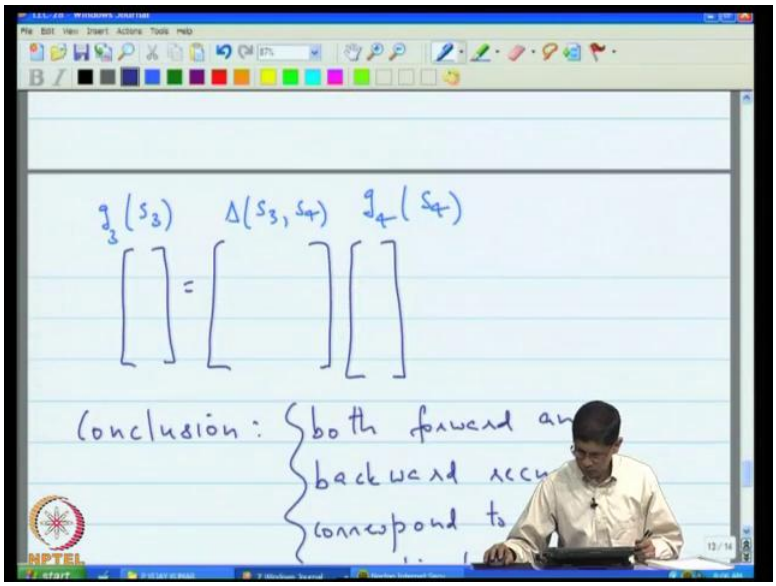
So, would have g 4 of s 4 coming in and you see that your computation then be with then say that g 3 of s 3 is the marginalization with respect to you would have the outer one which is u 3, which will have p of y 3 given s 3 u 3 p of u 3. And then the marginalization with respect to s 4 of p of s 4 given s 3 u 3 and then we have g 4 of s 4. So, this then is your recursion recursive computation and let me just we can node that in the present case equal to 1 here.

But, in enlarge an example it would not, so this would be equal to 1. So, this is then each of the recursive computation. And In fact, this is again matrix multiplication, because you can just easiest matter of rewriting. So, we can rewrite this in the form where you sum over s 4 and you have in coming g 4 of s 4 and then you sum we take the sum over u 3 in side and you have p of u 3 p of y 3 given s 3 u 3 and then p of s 4 given s 3 u 3. So, this would then be our computation and this entire think you can think of now as your already use gamma.

So, let me use different symbol, you can think of this as you are delta matrix delta of s 3 comma s 4. So, once again you have g 3 of s 3 is delta s 3, s 4, g 4 and you see that this is the matrix multiplication, again just like was true in the forward case it terms out. Although looks like you have two times here, because this is an indicated function it is going to select only one u 3 or nothing it is going to say that this no valid term in here; which means that

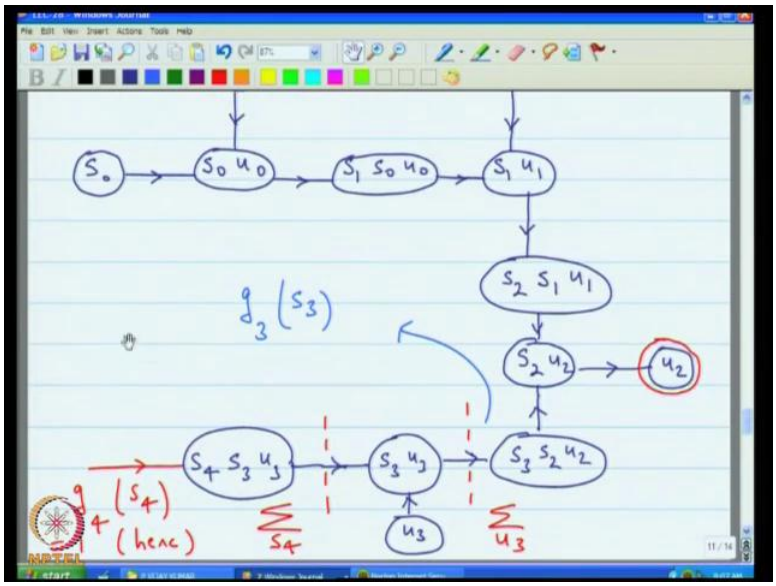
particular value of this is 0 or it is going to select be 1. So, then you can actually write this down as matrix multiplication once again and I am not going to spell that out.

(Refer Slide Time: 49:13)



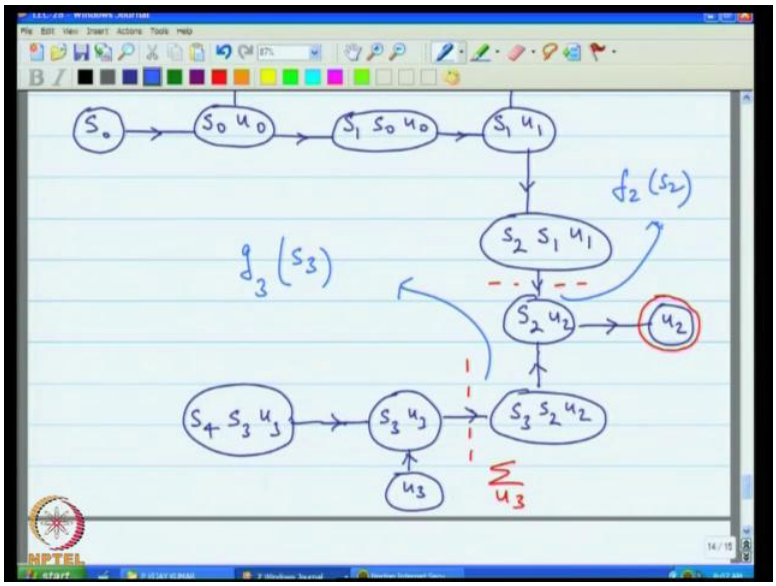
So, I will just indicate this on the side here. So, will have this equal to this into this, where here we would have g 3 of s 3 this is delta of s 3 comma s 4. And this thing here would be g 4 of s 4 this vector here, so this would be a form of the recursion in this particular case, so that is a form. So, the conclusion is that conclusion is that both forward and backward recursions correspond to matrix multiplication. So, only thing left to actually point out is well what is it? so at this point once you computed. So, the question is well how do you what do you do after that.

(Refer Slide Time: 51:59)



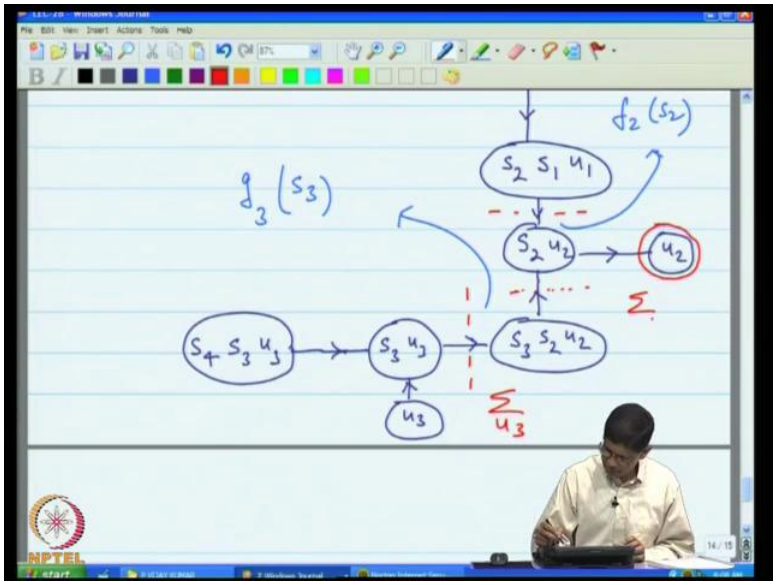
So, what will covered that far is the forward recursion in this direction, which will end up which will end up giving you, if you are interested in computing this set f 2 of s 2 at this point here. And the backward recursion, which will end up giving you right here g 3 of s 3. So, would have f 2 of s 2, g 3 of s 3 then you have these terms. So, you have that last step sketch to go out actually computing to you to, so let us take the closure locate that will stop with that.

(Refer Slide Time: 52:39)



So, what we know is that coming from this direction you have you have g 2 of s 2, f 2 of s 2 that is what you have here and here we have g 3 of s 3.

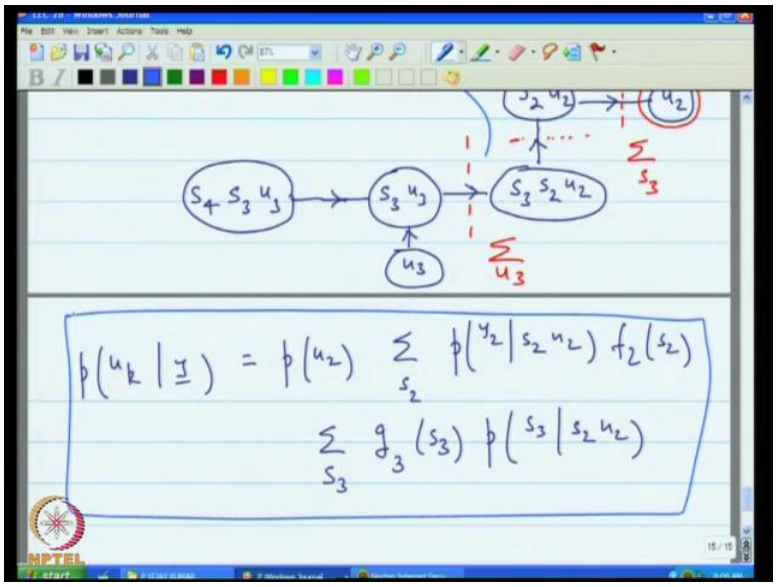
(Refer Slide Time: 53:36)



So, final computation is then is then you take you take a there is further marginalization that carried out here; and this is the marginalization with respect to s 3, so you take. So, you take this one marginalization here, which is with respect to s 2 alright. So, you have that you

have final objective function which we wrote out in the beginning, you are interested in computing here p of u k given y this is your objective function.

(Refer Slide Time: 54:23)



So, this objective function is now going to be p of u 2; that is the local kernel there times marginalization over state s 2. And the terms involved there are p of y 2 given s 2 u 2, f 2 of s 2 and then this marginalization over s 3 of g 3 of s 3 times p of s 3 given s 2 u 2. And this also can be given a simple interpretation, but we have just 1 minute left.

So, let us say quickly summarize. So, I explain in some link maximum likelihood code symbol decoding of the convolutional code. And this is also known in the little just the BCJR algorithm we looked at it from their GDL point of view and went in and zoomed in. And looked at detail at exactly what messages have been passed, will continue next time and we should able to wind up of discussion on the GDL next time. And then will begin talking about BCJR codes, so thank you.