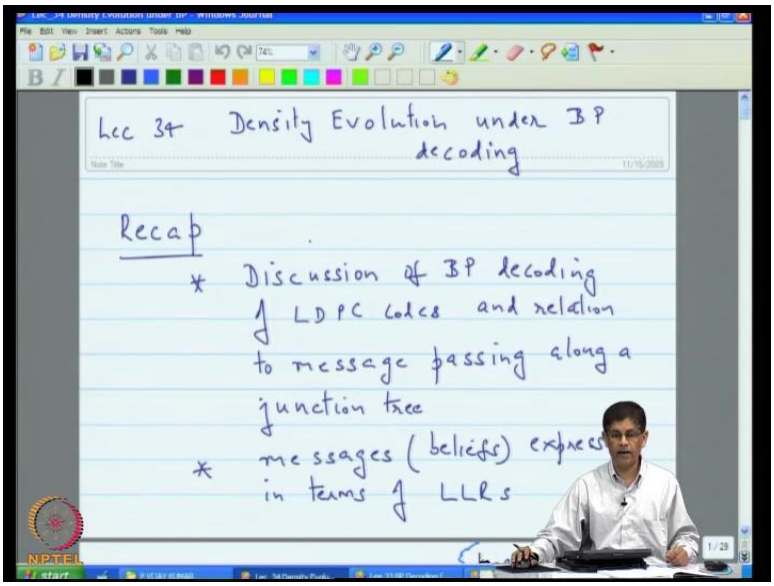
Error Correcting Codes
Prof. Dr. P Vijay Kumar
Electrical Communication Engineering
Indian Institute of Science, Bangalore

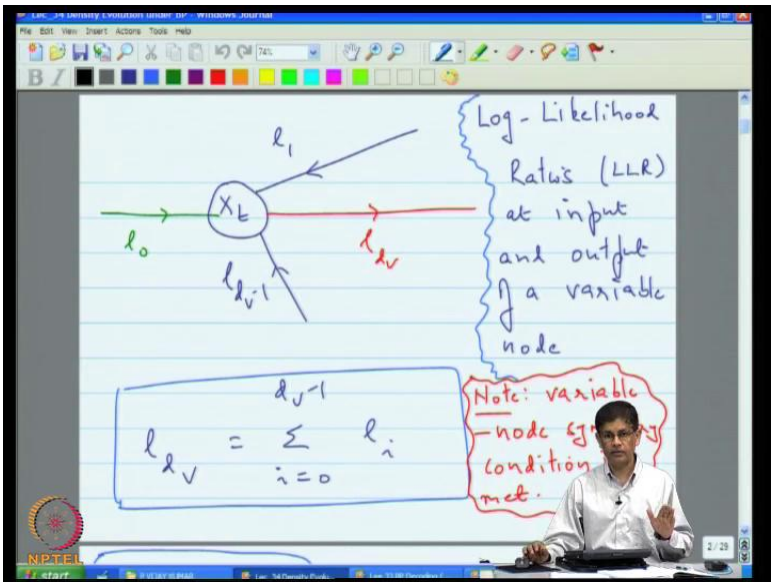
Lecture No. # 35
Convergence & Concentration Theorem - LDPC Codes

(Refer Slide Time: 00:31)



Good afternoon welcome back, this is lecture thirty five, so today has always let me begin by going over what we did last time. Last time we discussed we said that supposing you have LDPC codes, and let us using drilling decoded using belief property decoding. And I pointed out that belief property decoding is nothing but the message passing algorithm that we used when we decoding or code seven four two for example, using the junction tree. So that has an interpretation in terms of beliefs. Then we said that ok.

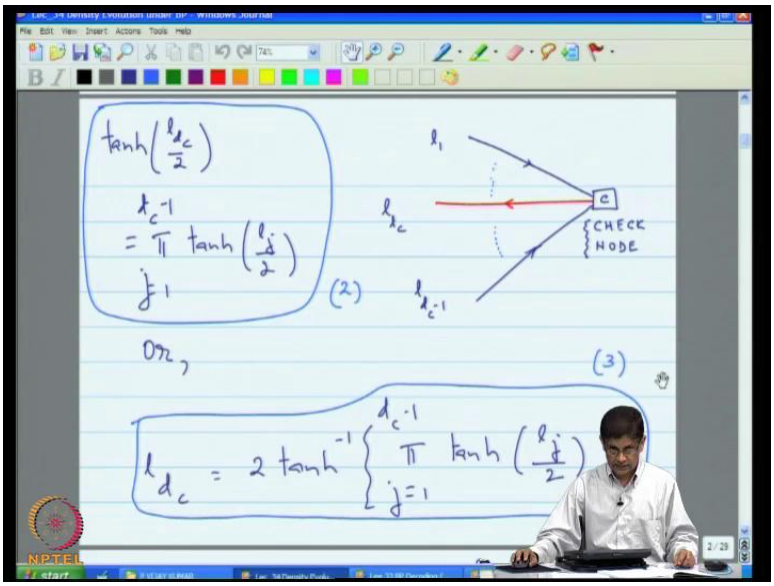
(Refer Slide Time: 00:56)



What the junction we passed message, which were believes; and this believes are function and involving the probability of variable taking on the value zero, and probability of same variables taking on the value one. These are the two components of vector that we would pass from node to node. But then it turns out that since sum of probability is 1, it is suffices to pass on their ratio, and their ratio is really likelihood ratio; and so what we do is it turns out that it is practically simpler to just pass the log likelihood ratio and we do not lose any information in the process.

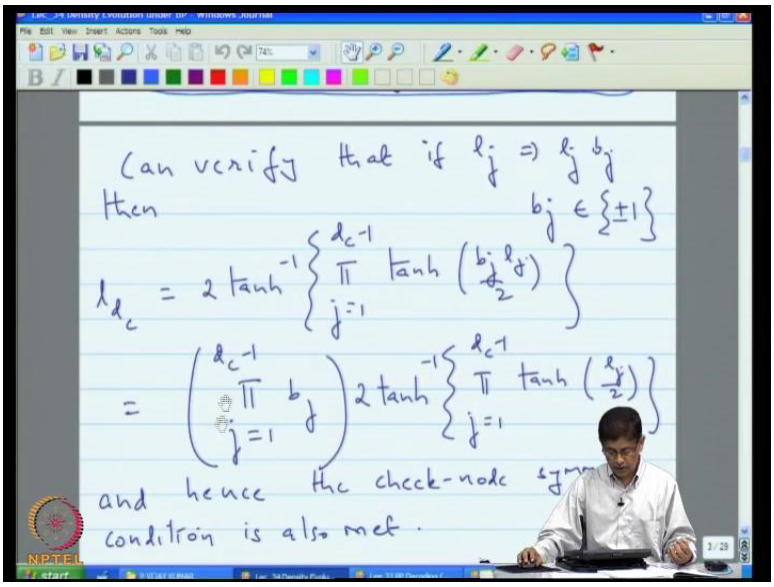
So in terms of log likelihood ratio if you apply the same message passing rule that you adapt with the junction tree, then what you would actually find is that message node at check node biased on to the just that the output log likelihood ratio is the sum of the input log likelihood ratios.

(Refer Slide Time: 02:03)



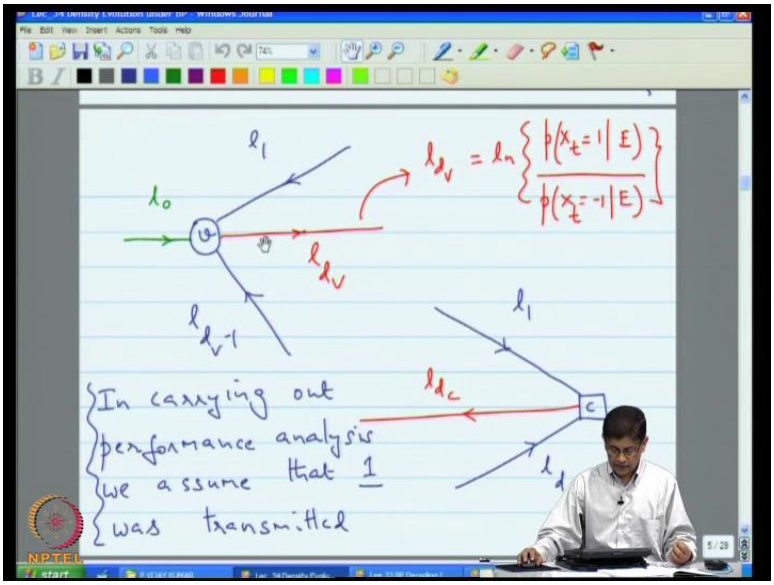
And then at the check node, it is a little bit more completed the log likelihood ratios that is coming at the output that is related to the time hyperbolic function there is a inverse hyperbolic and tan hyperbolic. And we verified that this particular message passing rule I mean message passing rules as translated into log likelihood ratio, we verified that the actually satisfy the two symmetric condition that we have to required; one was that actually you would have variable node symmetry, and then you would have check node symmetry. In the variable node symmetry we need that signs actually factor out that is if all of them flip sign, the output is also flip sign which is clearly the case.

(Refer Slide Time: 02:52)



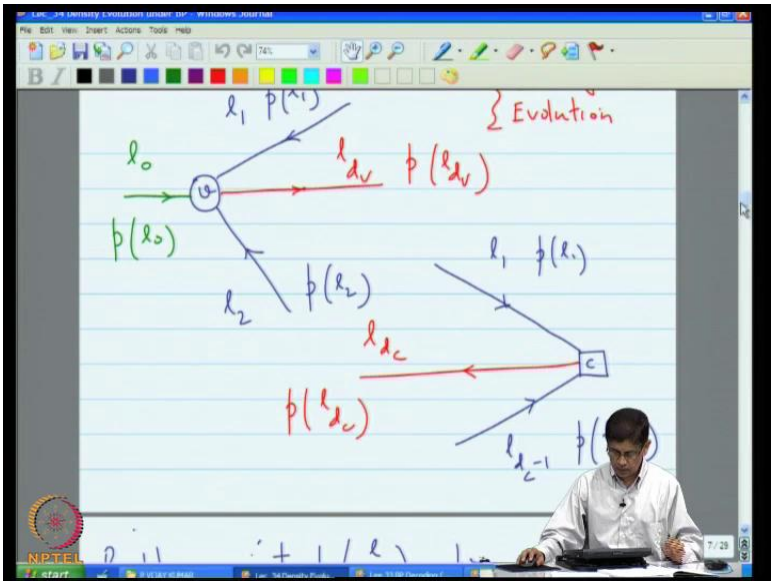
And at the check node, we simply required that if you multiply each incoming message bias plus minus 1 simple, then what you get passed at the output is the product of the plus minus symbol. And that happen basically because your the tan hyperbolic is the monotonic function, it preserves the sign.

(Refer Slide Time 03:12)



So then now we are interested in performance analysis; what is that mean? We want to interpret each message as an indication of whether the corresponding variable is a 1 or 0, in terms of plus minus 1, whether it is the one or minus one. Now when the plus minus 1 domain one corresponds to zero, in the zero one domain as you know. And we assume all one code word which always transmitted in the plus minus one domain; and in that situation if the message was correct the probability the x t equal to one given the evidence is greater than or at least you like to the case. So interrupt and since that is greater than one the log would be greater than or equal to zero, so interrupt a non negative message as a correct message, and the negative message as a incorrect message. And in performance analysis, you are actually interested in trying to find out well what is the probability that my message is series actually incorrect.

(Refer Slide Time 04:13)



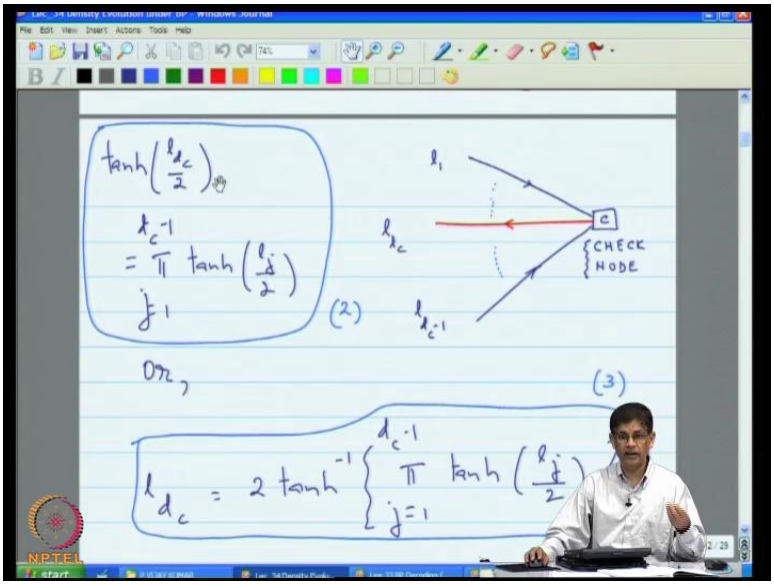
But rather than do that, what we actually do is, we carry out something called density evaluation were we actually try to determine not just the probability that this takes on negative or positive value, but actually the entire density function. Look at the density function at the density function that are incoming and provide using incoming density function, we actually provide the outgoing density function both at the variable node as well as check node.

So that took as that involve quite a bit of computation; part of the complication comes because also it is fairly easy to figure out how then input output relationship is derived in the case of

variable node, because after all you are just taking this sum of log likelihood ratio; and by our independent is , which comes from tree like neighborhood in the tree like neighborhood each node assumption to depth 2 l, where l is number of iteration, then it is easy to see that density function is convolution of these and you can use is the Fourier transform convert that to in product, so that is how is take care of the density the relating the density at that the output that the input, at the check node things are little bit complication.

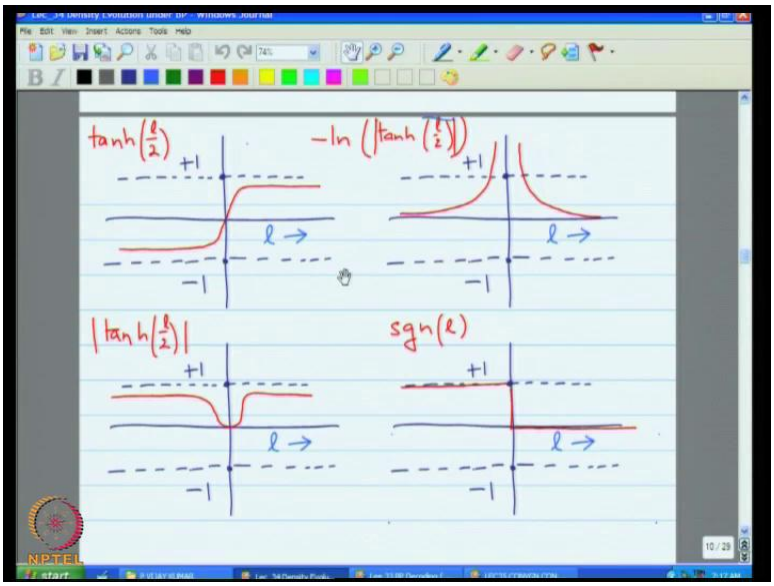
Here what happens is that you have the incoming density function and the outgoing related by the tan hyperbolic function. Now you might think and I have said this in the last lecture, well that the given that the product relationship although in term tan hyperbolic function that you might actually take a log and try to reduce it to sum in the case of variable node. The complication is the return hyperbolic in taken on both positive as well negative value, so there you have to exact slightly more effect.

(Refer Slide Time: 06:00)



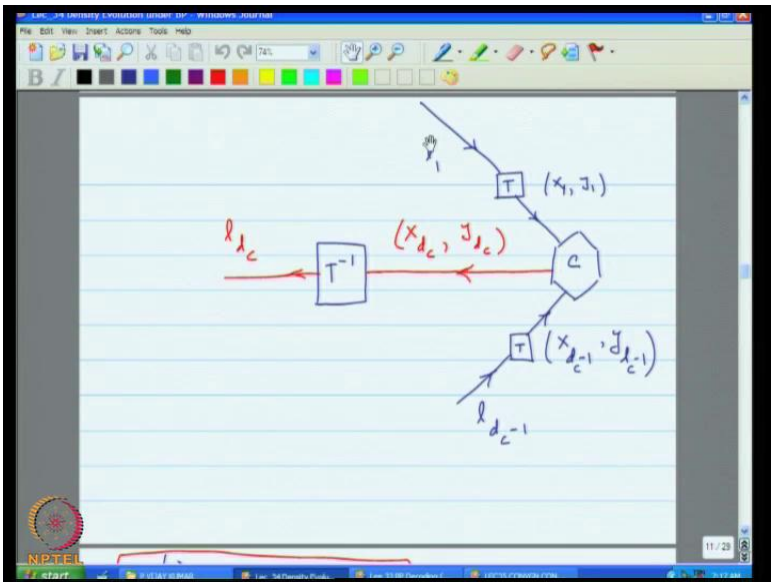
What you do is to break up tan hyperbolic into the sign part and then the magnitude path that is complicates thing. So that is what we actually did, we break these into sign part and magnitude part, so that is the sign here and then the magnitude.

(Refer Slide Time: 06:19)



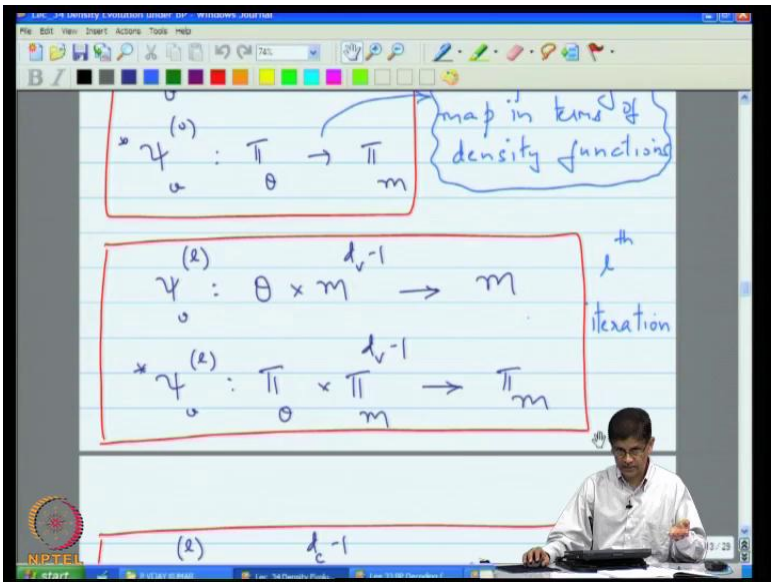
And here your the tan hyperbolic and the sign part is over here. The sign function we wanted to be a zero one function for convenience and tractability. So if you like, you can of think of minus one to the sign as the true sign of this function. So this is sin function that will use. And then in the place of magnitude, we will actually take the log and negative sign, so the negative sign here and use the zero one sign function at just have to do this convenience in tractability, but the conceptually you have just taken the magnitude in the sign of the tan hyperbolic, but as because you want separate sign in the magnitude.

(Refer Slide Time: 07:04)



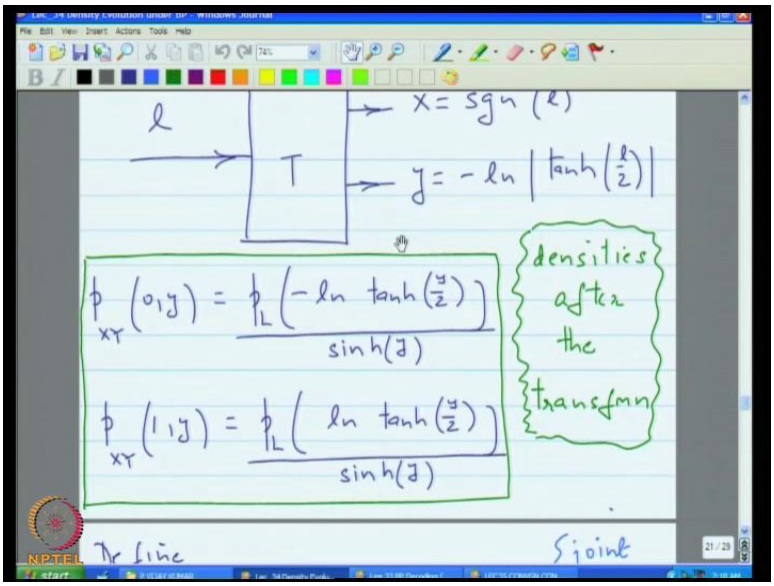
And then as a result when you do density evaluation, you next step input and output; and the input function l one is converted on to density function pair x and y, and then here check node we have the product of all of these. And then at the output that what happen here inverse transformation that actually take place. So you have to take the incoming density transform it when actually go through a multiplication and here what we do is you make use of characteristic function to actually relate the output density of the pairs to the input density of pairs and then, we do the inverse transformations.

(Refer Slide Time: 07:47)



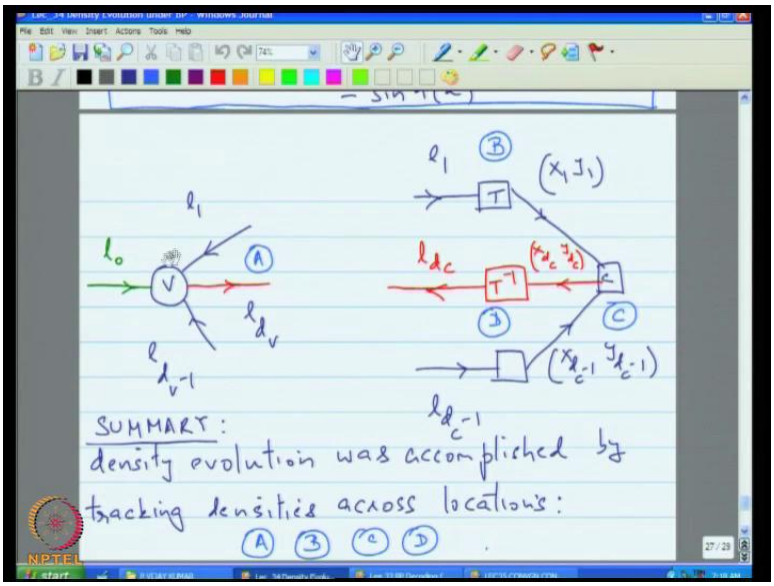
If you do all that so what you are really doing from the abstract point of view you are saying well whenever I pass messages I am using the mapping from one alphabet than other, but I then corresponding to that mapping, this corresponding mapping involve the density function. So that is the abstract view point, I . And we went through the competition, this is where I discuss tree like independent assumption, here is we did the characteristic function calculation, and here is where we the interrupted the characteristic function in terms of Fourier transform.

(Refer Slide Time: 08:26)



Here we did the somewhat calculation involve transformation across the transformation t and it is inverse. So at the end of all of these, what we were able to do?

(Refer Slide Time: 08:42)



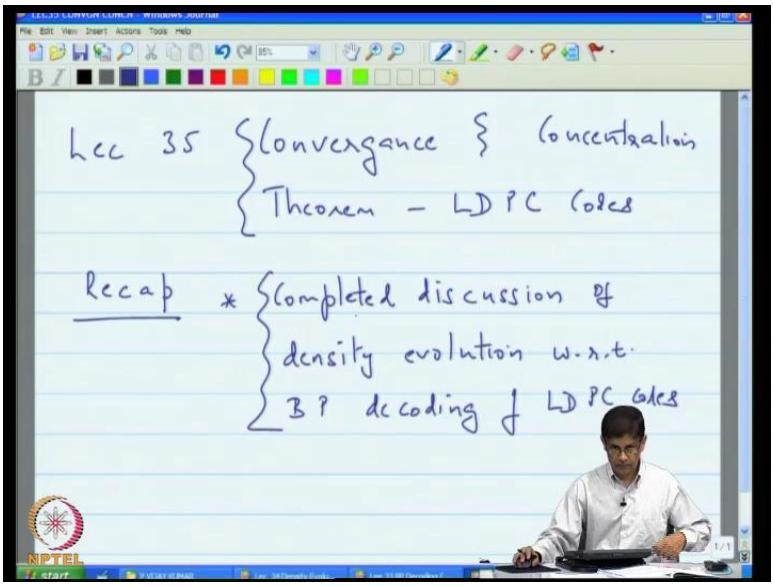
We were able to actually go through a round of iteration that we were able to translate incoming density function to the outgoing and then outgoing here at the check node, at the check node we go through, so we start by transforming density at a then across b which across the

transformation, then at c and finally at d. And then point of view, we carried out all the transformation that we need that actually, so we have finished the one cycle of iteration right we gone from an incoming density function back to the incoming density function. So that was the density evaluation function. And in practice what you would is, you would actually carry out this competition, you would recursively compute the density.

Initialing with the density function of incoming random variable which is l naught, we will actually use its incoming density function and that is what we will actually feed the entire process. Over the Gaussian noise channel you can actually show that the log likelihood ratio itself as the Gaussian density function, so what would derives is entire process is the incoming density function are Gaussian with the mean equal to determine by the fact that the code symbols assume to code plus one.

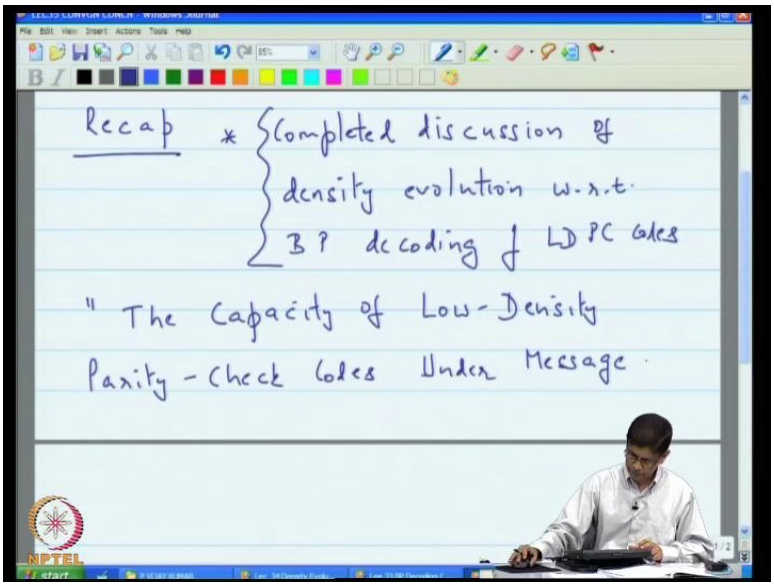
So that is density evaluation. And today what I thought we will do what is the logical next point to do is to prove something that converge the call the convergence and concentration theorem for the ldpc codes, so again, so my reference for this is February two thousand paper. So I will just begin with the quick recap.

(Refer Slide Time: 10:35)



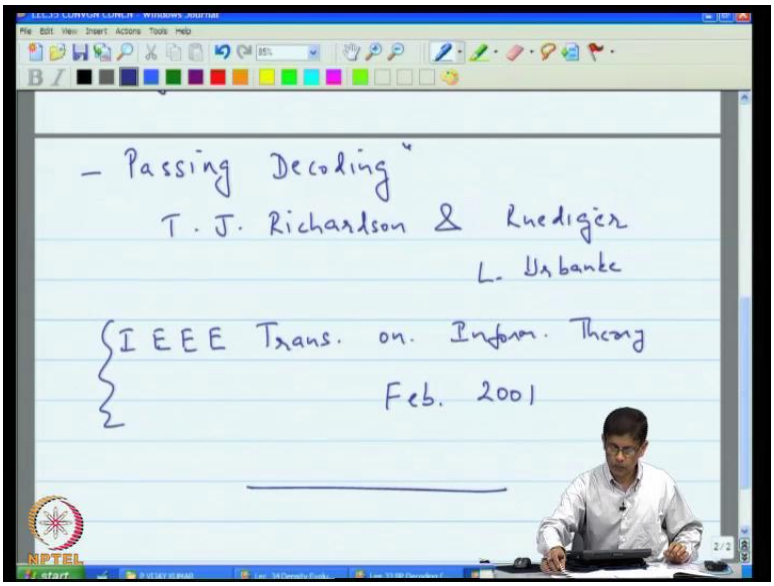
So we completed discussion of density evaluation with respect to the belief propagation decoding code of ldpc codes.

(Refer Slide Time: 11:55)



And the reference has been for this the paper that I believe that I mention earlier to you namely the capacity of low density parity check code under message passing decoding.

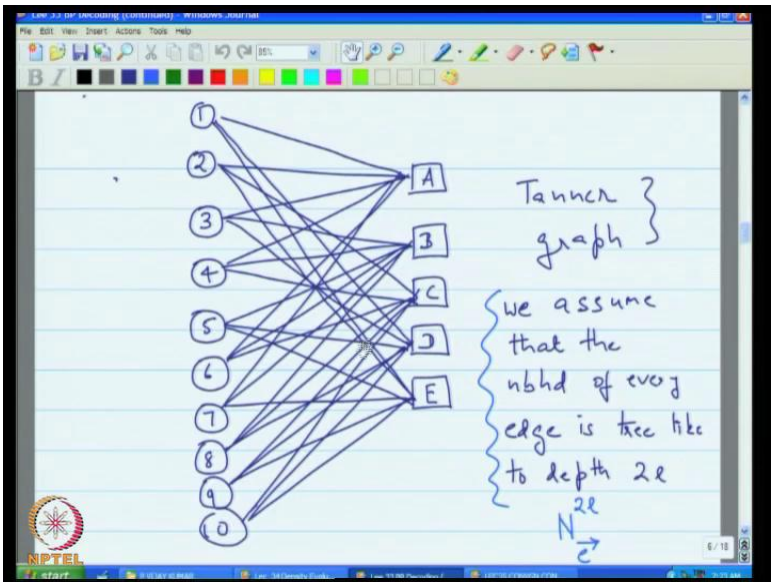
(Refer Slide Time: 12:00)



The authors are T J Richardson and Ruediger and Urbanke. And this is the IEEE publication, the IEEE transactions on information theory and February 2001; now, right for that, representation

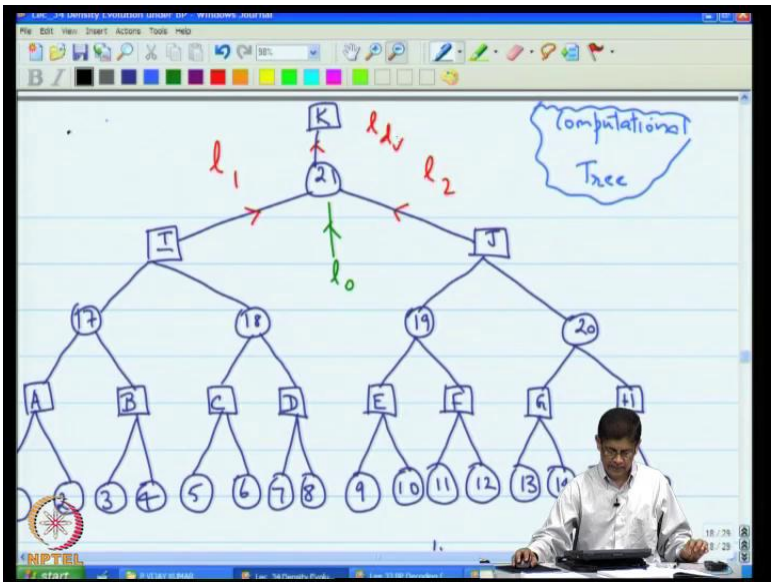
get back to the title of today lecture, which has to do with convergence of concentration theorem of ldpc codes.

(Refer Slide Time: 13:39)



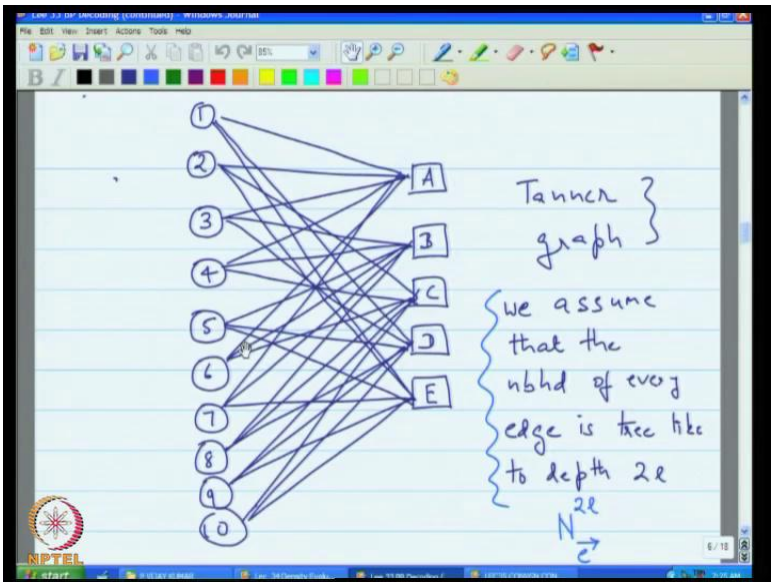
And let we motivated in this manner, here we were working with the ldpc codes and belief propagation decoding, what we assume was that there is, this is the tanner graph of a ldpc codes, so we are assuming that the tanner graph of the ldpc codes is tree like to depth that every node in the tanner graph has the tree like neighborhood to that 2 l, what do I mean by that?

(Refer Slide Time: 14:15)



Well I mean that if you actually come here, so here is the computational tree, so what I mean is that when I unravel the tanner graph and actually come to the computational tree, then you actually notice that the none of this, that none of the nodes actually repeat, so if you look up across variable node, there is no reputation; that means that there is no loop. For example if there was four here and there is also four here that would be an evidence that would actually have a cycle. But here all the nodes are different, so that means neighborhood is tree like. So under this assumption what we actually did was we found out that probability that the message that you passed will be incorrect in the since that we are carrying a wrong sign that you that you would expect if you are actually, if your message were, what we are indicating what should have been indicated.

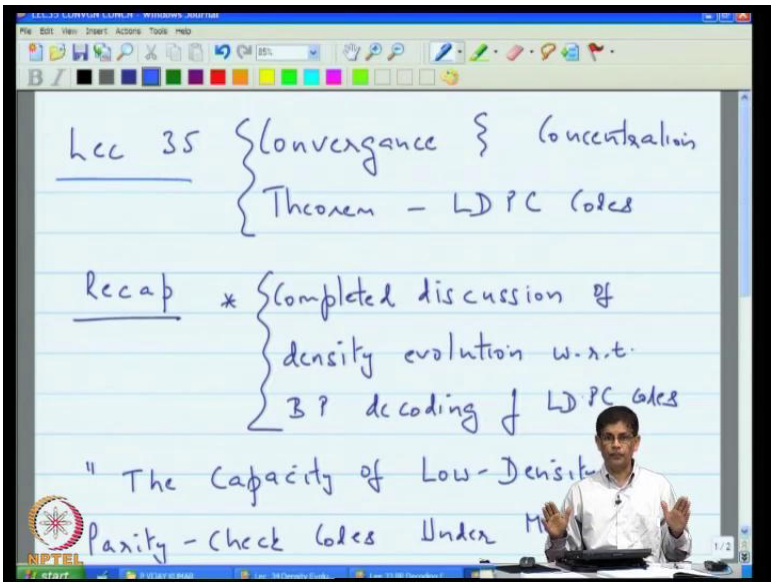
(Refer Slide Time: 15:00)



Now on the other hand, it actually very difficulty to actually construct an ldpc code, whose tanner graph is tree like to neighborhood let say you are doing a twenty ratio that you are required tanner graph to be tree like to depth forty that is the very difficulty to construct. So what needs to be but on other hand even in practice, people are found that you do not really need the neighborhood that is tree like in order to get excellent performance. If you randomly construct the tanner graph is going to few short cycle, and that does not seem to effect the performance greatly. So what this seem to be calling for an explanation saying theory on one hand required the neighborhood of tree like.

And the other hand, when we absorb thing and practice it seems like really do not needed, so that is the gap today lecture will try to fill. Now I am not projection give you all the technical details but at least I tell you roughly how the proof code.

(Refer Slide Time: 16:08)



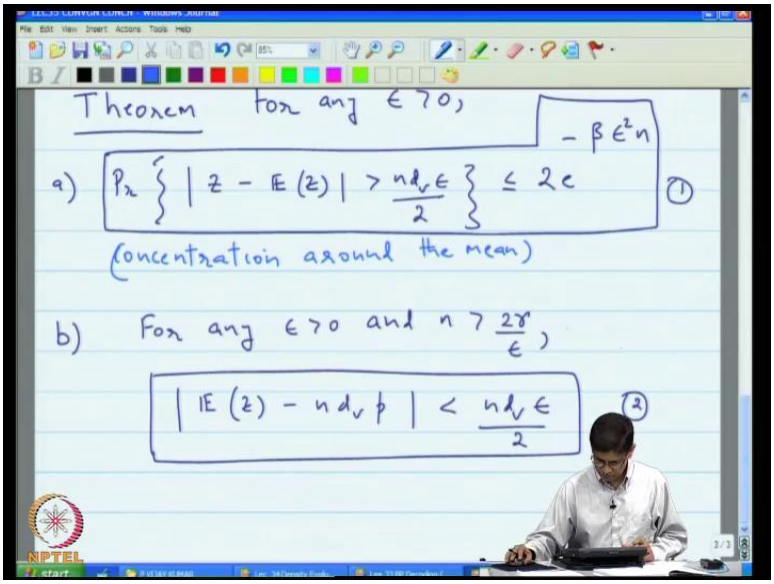
So that is the concentration and convergence theorem. In nutshell, what they actually tell you is that it is true the belief propagation tells you that look, you do density evaluation, and if the neighborhood tree like the density evolution will tell you that yes, your code is going to perform very well of, course you after look at that actually assume certain channel parameter carried out density evaluation which is competition in nature, I guess in our lecture series here what we doing this concentration more an principle of these things.

If you do carried out this density evaluation, then you can prove, then you will after you follow through with the steps whatever number of iteration you need that you are going to get very good performance as a long as channels is not very bad, so that is great. So now we want to actually say that supposing I do not have a tree like neighborhood, am I still ok, and this attempts to answer that. And the answer will actually come on in two steps, because it will say that look if you pick the graph at random. And if you average of the graph performance, then the number of incorrect message on average across a random on sample of tanner graph will be close to the number the incorrect messages passed.

Assuming the tree like neighborhood, so one you have collection of tanner graph and you have that their average performance is close to the performance that you would expect in the case neighborhood tree like. Then the second thing is that on the other hand if you pick a random

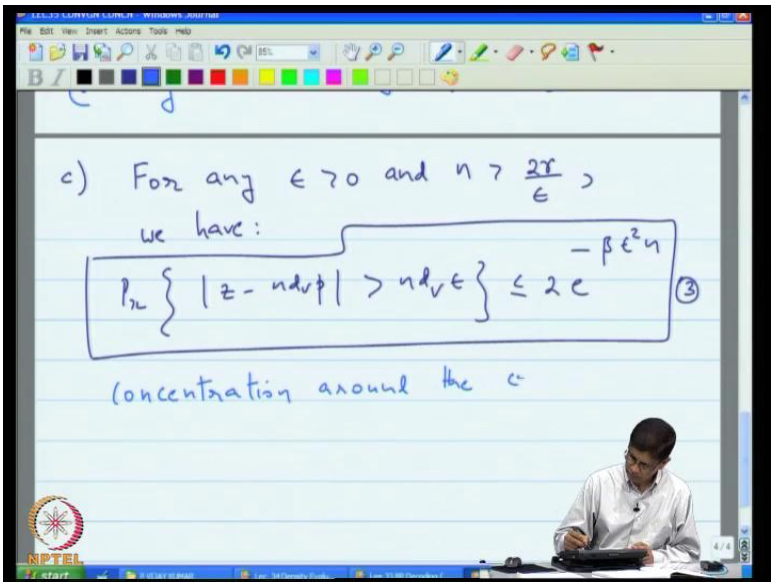
graph from the sample with high probability its performance is going to very close to that of the mean. So when you put these two together it tells you that random graph even if not tree like, the number of incorrect messages is the high probability for large block length l that is important, is going to, high probability going to close number of incorrect messages which are being passed in tree like case; and in tree like case, you can verify to the density evaluation that the number is small that the number of error is small, and therefore you get good performance,. So that is the connection you want to make; so with that the introduction, let me just begin writing.

(Refer Slide Time: 18:50)



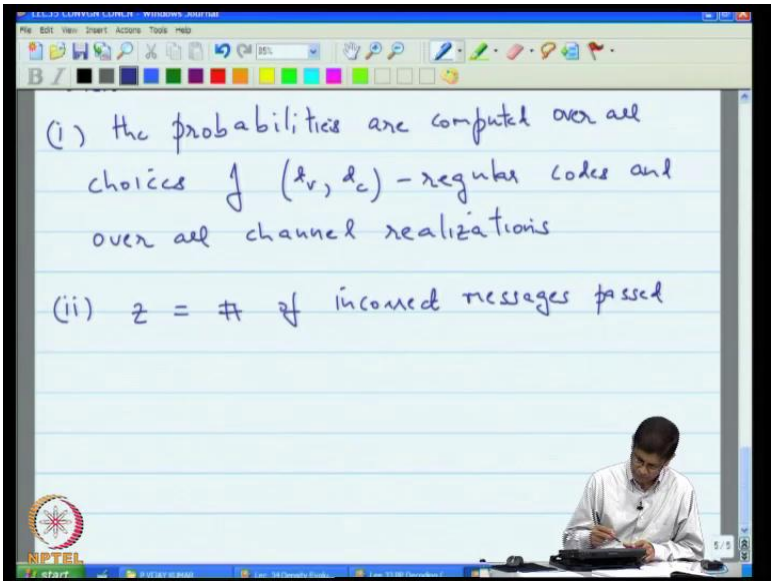
So here is the theorem for any epsilon greater than zero, you have the following a the probability, the z minus the expected values of z is greater than n d v epsilon by 2 is less than or equal to 2 e to the minus beta and epsilon square in to n. And this result we will term as the concentration around the mean. Now I explain little bit all the settings but all the symbols been I just want to put down the formal theorem then b for any epsilon greater than zero and n greater than gamma by epsilon the magnitude of the expected value for z minus n d v p is less than n d v epsilon by two. At the same time I would like to give this equation number is well, so let me box it, and let us call this the equation one. So this is equation two; and this one will actually termed as converge to the cycle free case.

(Refer Slide Time: 21:40)



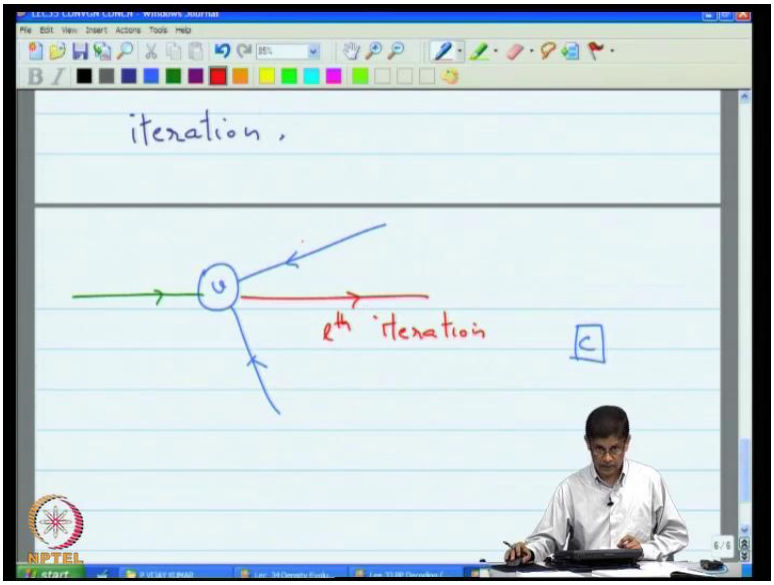
Finally we have c for any epsilon greater than zero, and n greater than 2 gamma by epsilon we have the probability that z minus n d v epsilon excuse me, n d v p is greater than n d v epsilon is less than or equal to two times e to the minus beta epsilon squared n. So this is three; and we will label this as concentration around cycle free case. That is lot of symbols and lot of writing, and let see what these symbols mean?

(Refer Slide Time: 23:39)



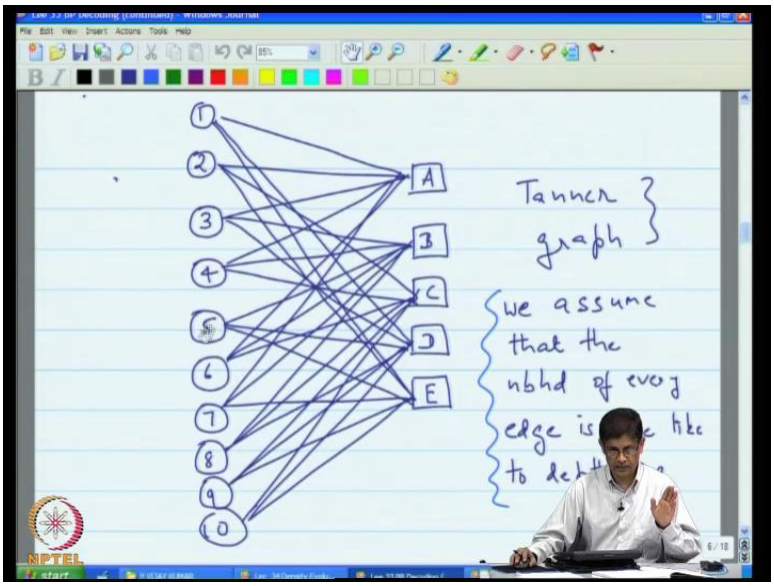
So, where now I am going to discuss the setting of the theorem one, the probabilities are computed over all choices of d v d c of regular codes and over all channel realization two z equal to the number of in correct the message path passed from the n d v variable nodes to the r to the r d v check node in the l eth iteration, so what that mean here. So it will be try to draw picture here, so here you have your variable nodes.

(Refer Slide Time: 26:00)



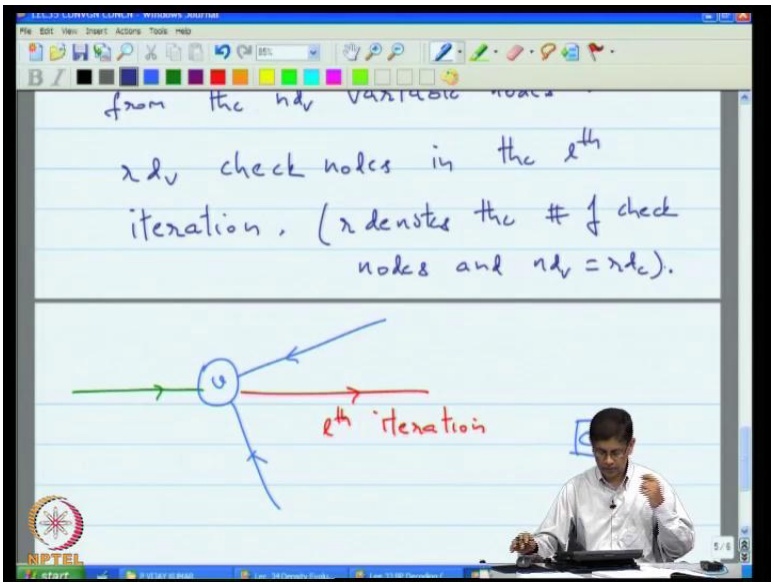
And let us see this is the variable node and here you have a check node, and what you doing is off courses, you do the input from the channel that also have a input coming from the incoming edges you have you are out going message. And let say that this is taking place along the l eth iteration; and remember the way the iteration go is that whole iteration iterate cycle is begin by start with the information that you receive the cross channel, and then pass that the variable node pass the information across. So perhaps I should show picture at the tanner graph. So just quick remainder, the way the iteration starts is that whatever come to the channel is passed on to the check nodes.

(Refer Slide Time: 27:00)



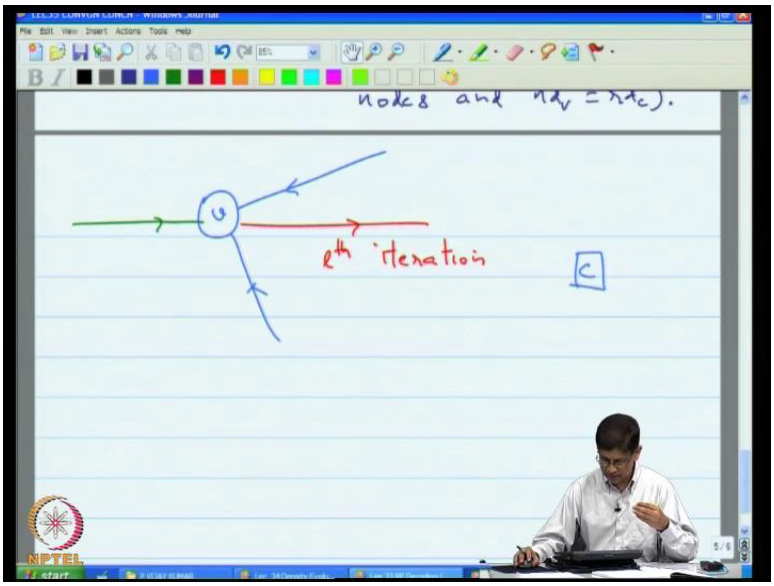
You might consisted at the message passed at the zero iteration, thereafter the first iteration start the iteration begins, when the check nodes actually transmit message to the variable nodes followed by the variables nodes sending back message to the check nodes. One back fourth message passing constitution iteration, so what we talking about is that l eth iteration that is means that ones you place l time and looking at second place iteration during at variable node are talking to the check nodes. So that the instance that is the moment actually of interest, so the message passing variable to the check nodes during iteration l eth point making this that there are difficultly in the tanner graph, why this maintain n this number of variable nodes.

(Refer Slide Time: 28:00)



But I think sometimes used m to the m little m is number of check nodes by the purpose this lecture however and actually going to use to r ok this r scheme right, so part in where putting bracket is that we will denote the number of check nodes. And the edition we haven d v equal to r d v c. Following this time to explain, what is put terminology in this theorem it all about so the z here the z is the number of in correct messages passed during the variable node to check node during the eth iteration That is again the eth iteration if you thing about all the edges there are going from variable to the check node in a count how many of the in correct just based on the sign, if the sign is negative then consider if the sign is message because this message are real number if the sign of the message is negative you consider at incorrect message.

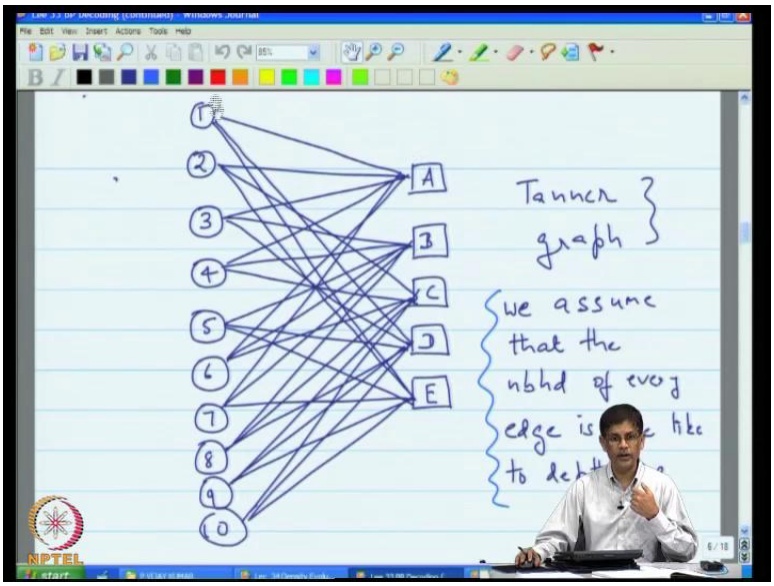
(Refer Slide Time: 29:32)



In the tanner graph, if you going to n d v edges n d v same as the r d c so would have this many message this many messages and the message are going from left to right the variable to check node and z is count of the number of the message incorrect during l eth iteration. Now in the theorem l is not appear the reason being the that assume there is the fix number l say twenty that you are interested in and theorem is about that number l, the number l kind of in the back ground it is yes the number of iteration is l talking about but the graph but the theorem in the notation focus on other things, so what are the other things well tells that you look they and there is the average when you take a expectation where averaging over what.

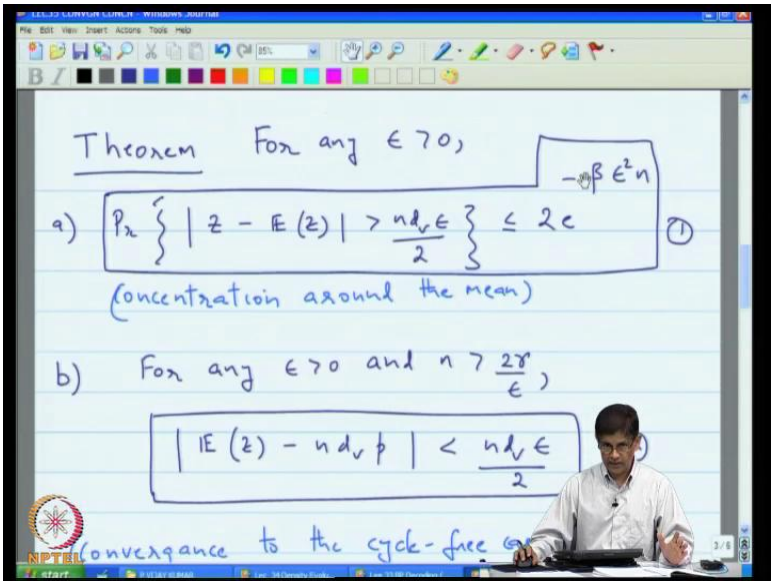
We are doing performance analysis assuming all one code word trans metric, so we are not averaging over the choice of the code word, what we are averaging over so the averaging over the channel realization and also we are averaging over the random of the sample d v d c regular graphs.

(Refer Slide Time: 30:49)



So how do we get this d v d c regular graph in sample will see that you actually have this tanner graph there are n d v edges. what we do we select this edges in such a way get a uniform degree on the left and right and very shortly actually I am going to show you picture but explain that talk about the sample of d v d v regular graphs little bit better for the moment.

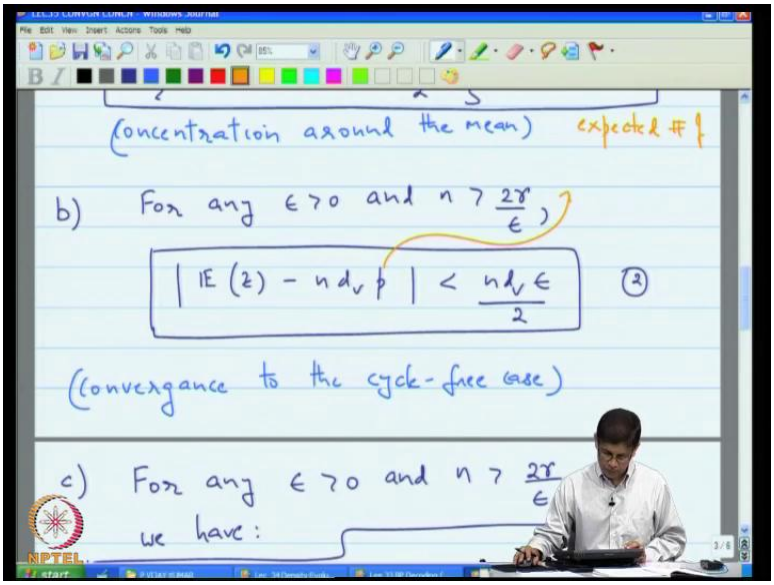
(Refer Slide Time: 31:26)



Let we continue, what we have this is saying that the deviation of z from expected value the probability the deviation is sub stably small that have you should do it and epsilon is small number you confuse and n is the number of nodes d v is the degree of the variable node and beta, the parameter beta that constant independent of n and so the function of degree of the check variable node and again we have n. So this is time exponentially with that means time very fast, so the probability that it deviates significantly from it is average is small; that is why call the concentration around. This theorem thus that look the now, this an sample does not assume tree like neighborhood. Now we do this expected value. And look this expected value average computed over all d v d c regular graph, without regard to I neighborhood tree like or not is close to n d v p. What is this p? In turns out that single parameter p summarize all the information that we will pick of from the density evaluation because density evaluation tells that I can tell you probability which the message passing during the l eth iteration will doing correct assuming tree like neighbor.

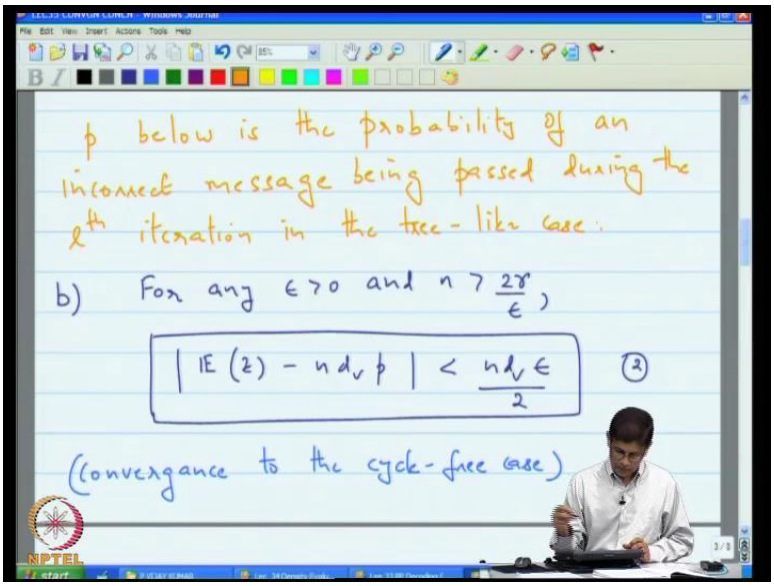
And since because by just by checking the probability the sign is negative; now the total number of incorrect messages or expected value given that any particular edge in correct with probability p is then n times d v times p. So you should interrupt is this expected number of incorrect message passed in the two like cases to part should write that out.

(Refer Slide Time: 33:47)



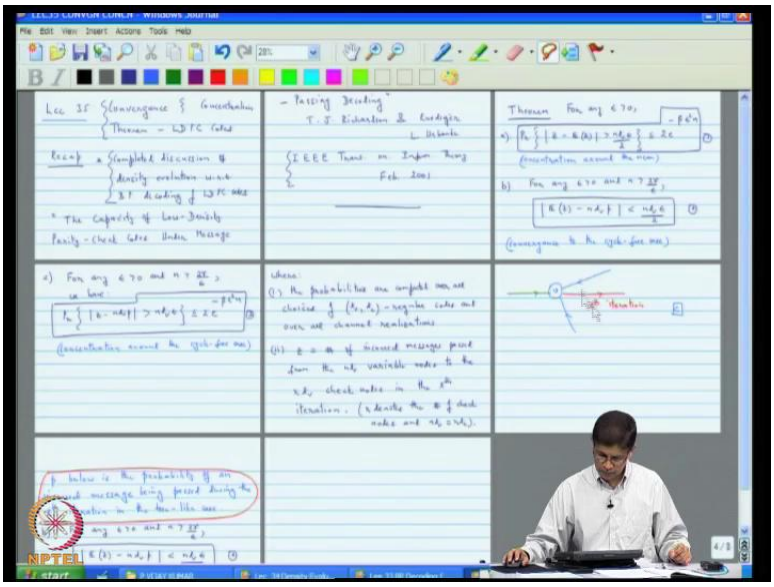
Let me spell that out you know, what I thing that I will actually take that trouble to write some the separate piece of this paper, because I consider to be important.

(Refer Slide Time: 34:05)



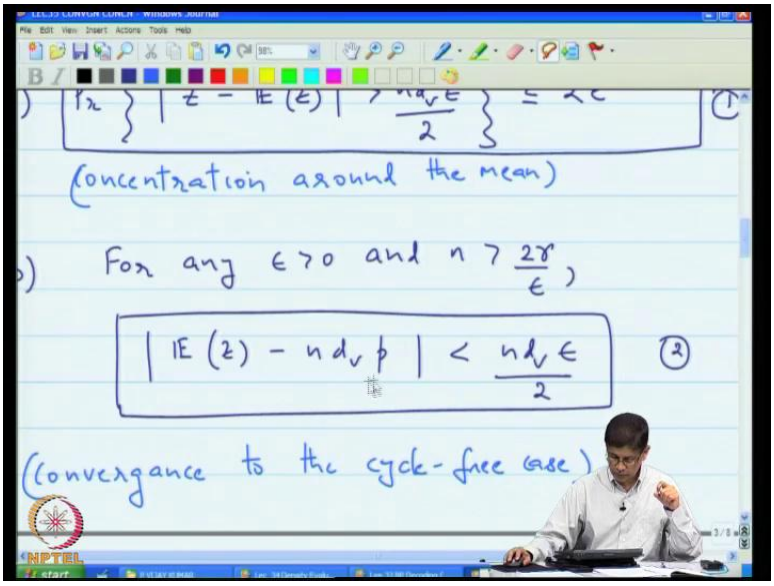
Actually let us duplicate is page. So I just want to make mention here that p below is the probability of an incorrect message being passed during the l eth iteration in the tree like case. I notice on you are screen that is shown very well this particular choice of color, so let us change that to blue; nothing this better. So this is then, so this is the probability of incorrect message being passed in fact all of density evaluation or all the information that we pick of the density evaluation it does collapse into single parameter p, so let us make a node that is derived using density evaluation.

(Refer Slide Time: 37:54)



I just make show that I have pages in order let us use quickly zoom in a lecture, so everything since in order. We have theorem here and explaining over the courses of next use slides I am explaining notation of the theorem.

(Refer Slide Time: 38:03)

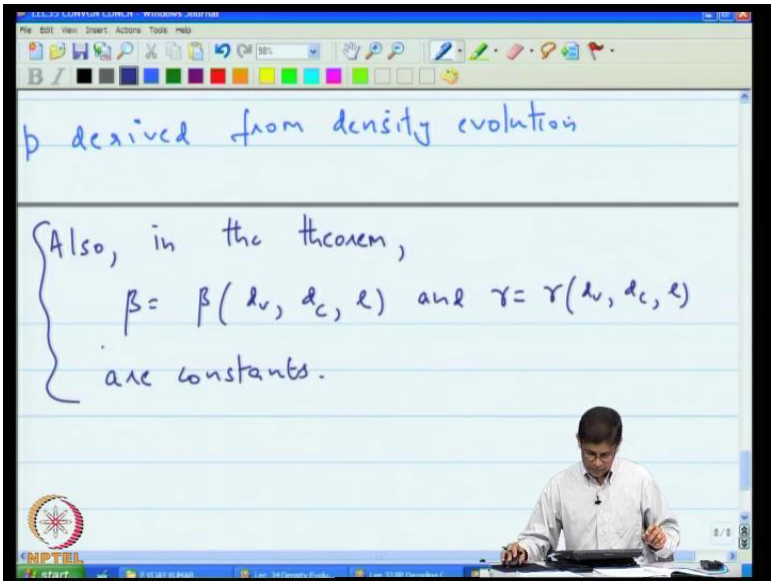


Now I explain the length parameter with the p, so what telling us the p average an expected number in corrected message in tree like case, this is saying the difference between the expected

value in the generic case, there is not necessary tree like averaging over tanner graph d v d c regular tanner graph; and average of tree like is not different, that is provided unique logic enough you can actually make this difference small, now you can actually put to gather and actually one of this result which is therefore the probability if you take d v d c regular tanner graph random and do decoding on it, you carried out decoding up to lth iteration the difference between number of incorrect messages that you will get from this random graph and the average that you will get tree like case is with high probability small rather the probability it deviate significantly very small that is called concentration around the cycle free case.

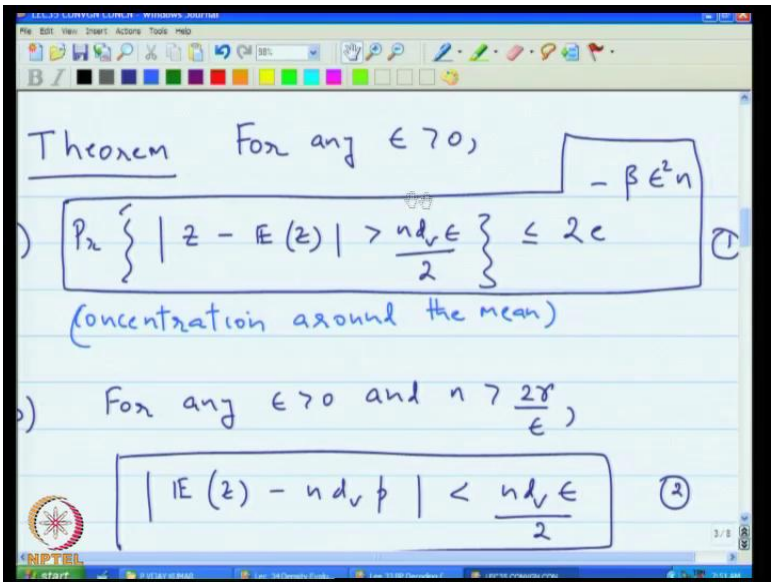
So the third property, really follow the first two, so you only have to prove part one and two. Now in terms of that prove of two, it must straight forward proof of one is not so involves symmetric in the theory; and it is not difficult it just a lengthy, I just finished teaching that regular class members here, and it turn out to be rather lengthy, I decide that I am just going to nor pursue that, so I have little bit more time discuss some of other algebraic, some of the other algebraic codes that we would like to cover in these course.

(Refer Slide Time: 40:15)



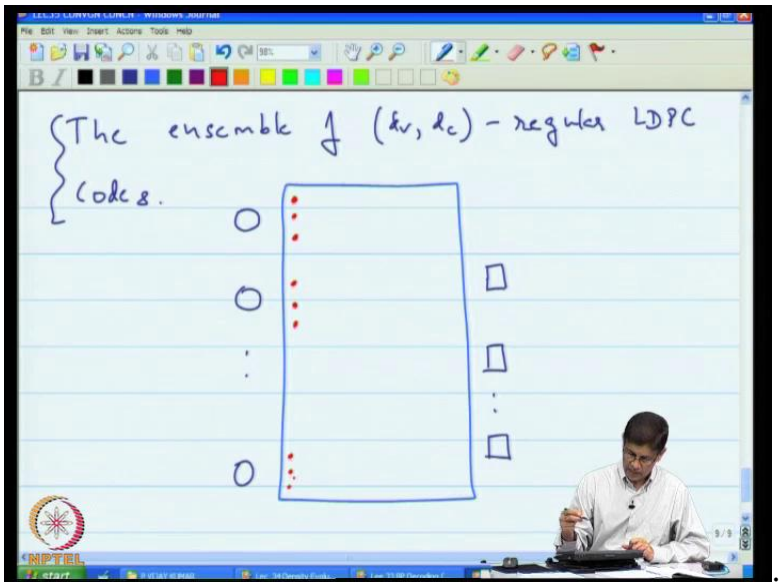
So continue our explanation for what the various symbol in the theorem means, finally we have that also in the theorem beta equal to beta d v d c l and gamma equal to gamma d v d c l are constants.

(Refer Slide Time: 41:10)



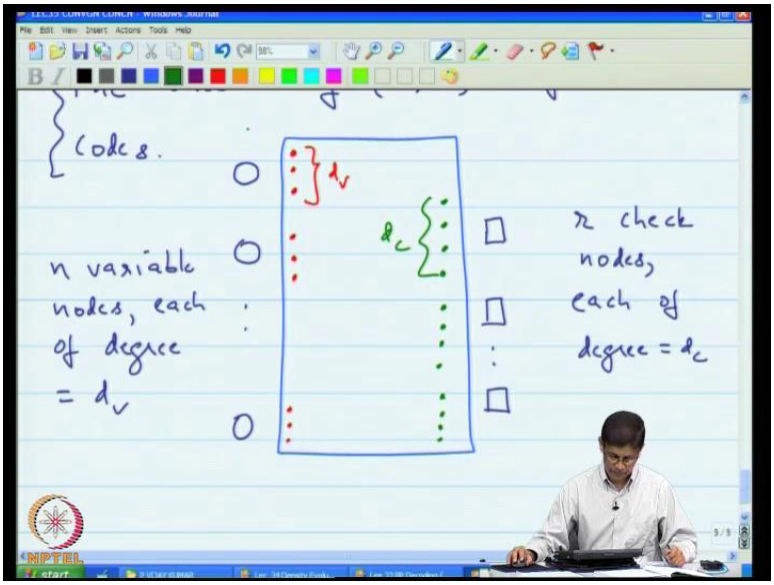
I explained all the notation on the theorem you know what you know the beta is the consonants epsilon the small number choice, beta is the very small number quantity and n is the length of the code, z is the number of incorrect messages and d v is the degree of the variable node and again here some of the variable theory delete. Other thing I would like actually explain is that the I am saying that the avenging of the un sample of d v d c regular code what exactly thus that mean

(Refer Slide Time: 41:47)



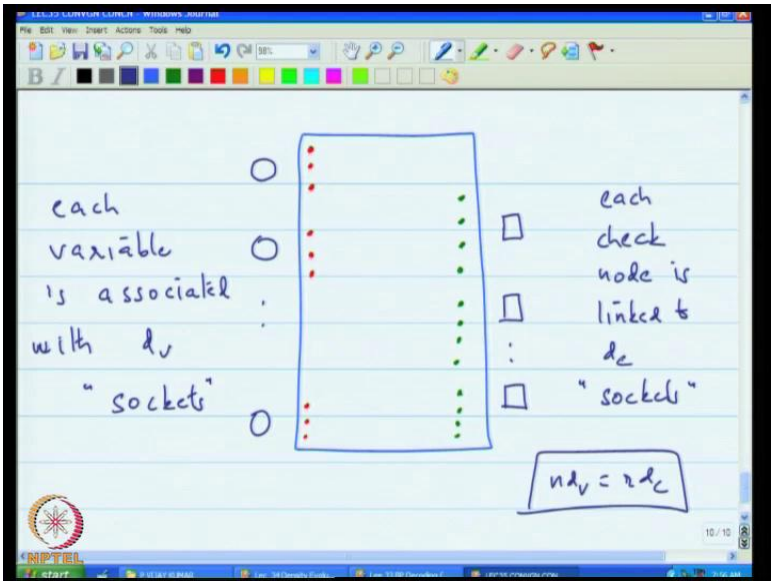
So the way you can get a ensemble it actually look up on in the tanner graph in this way. So we have the variable nodes, so let me draw a few variable nodes on the left and let me have some check nodes on the right. Now each variable node is associated to certain degree, which we will call, which is d v. And similarly each check node is associated with the certain degree.

(Refer Slide Time: 43:15)



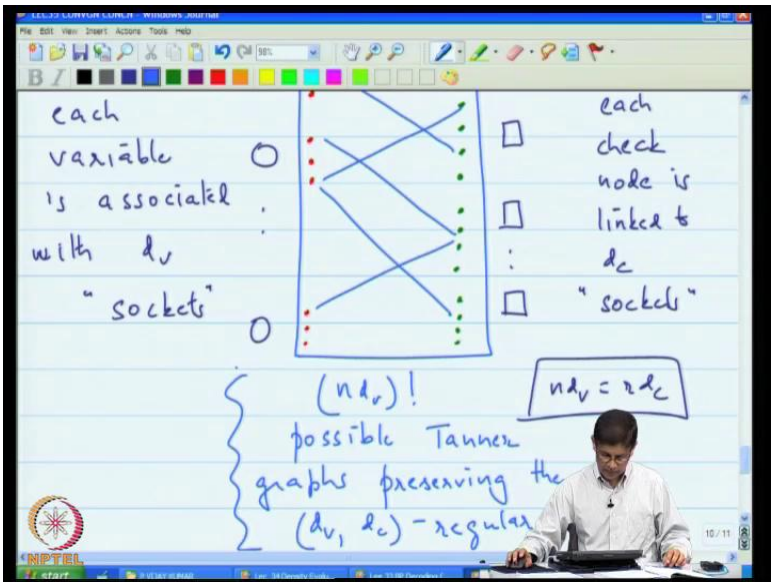
So that means that we have here n variable nodes each of degree equal to d v. So this number here d v which in the cases tree. Now analogously, we have r check nodes each of degree equal to d c, this number here is d c. Now what we mean is that we get some example d v d c regular graph and cut it out and paste it on next page; so we you get the enscmble this pursuing that look I meet to make sure. I think of each of variable node as having sockets, so you regard each of having sockets.

(Refer Slide Time: 45:51)



So each variable node is associated with d v sockets; and similarly each check node is linked to d c sockets. Now we know that n d v equal to r d c, so that means that if you cannot number of sockets on both sides that seen. So constructing v d c regular graph it simply matter actually join lines connecting as socket here and socket in there. And mathematically you can actually view that as a permutation.

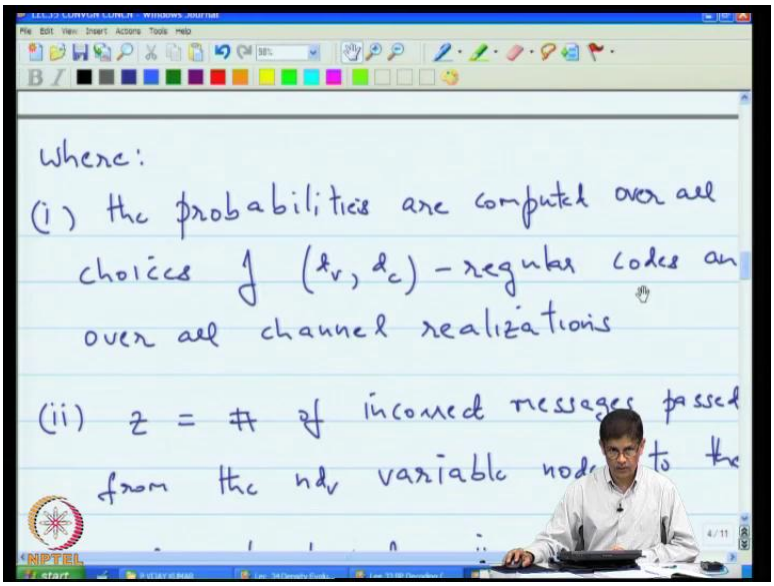
(Refer Slide Time: 46:00)



There are n d v factorial possible tanner graph preserving the d v d c regular property; now there is one issue that might cause you to think a little bit namely that look if I am actually drawing this connection like as random, is it possible that I will connect draw connection from same variable node to same check nodes and I will have more than 1.

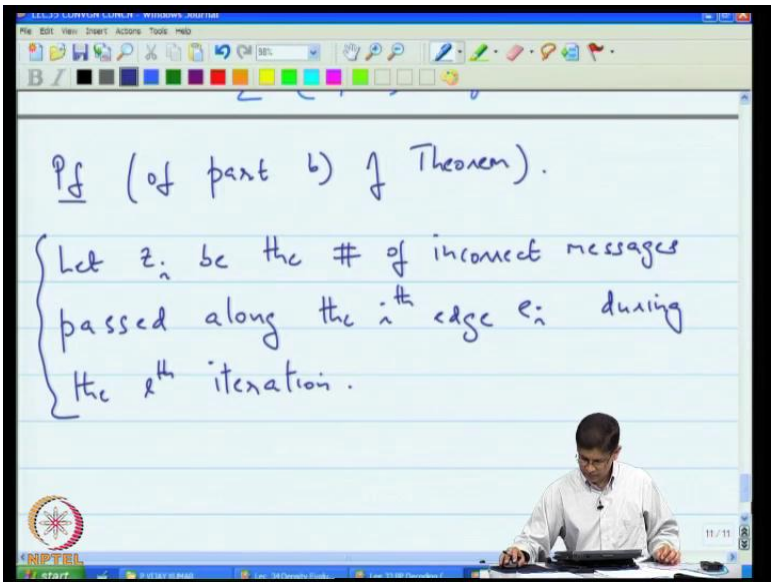
Now that can happen, but it turns out to be not capably important, so vast majority cases you will not, you will not actually get reputation of that time. But the even if you does, it does not, you can turns out that average of those graph as well. Although in practice, you will not actually use a graph which has multiple edges between the same pair of variable and check nodes ok. But interest of time, I am not going to attention that so let us move on and ignore that reuse the paper if you want actually see how that is handled. Now this explains last part of theorem because writing the beginning when I actually introduce the theorem, as said that look…

(Refer Slide Time: 48:18)



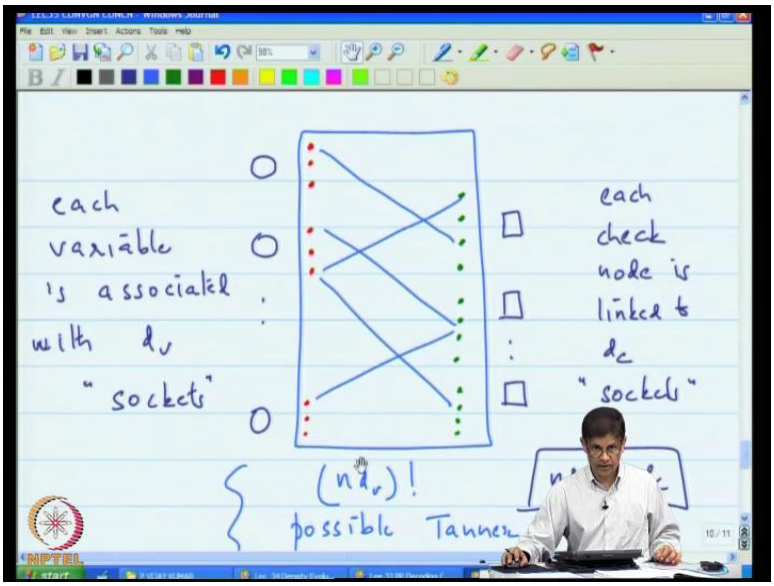
The probabilities are computed over all choices d v d c regular codes. What this is doing is this is explaining, what this class collection of codes which you are averaging case. Namely the collection the obtain treading as the polarization, and determine which socket is connected, which socket on right that the which will actually average. Now let us go ahead and proof that is I told you in the theorem, I am only going to proof part two; part b in turns out that part a little bit technically involved, it involves nothing else is not particularly difficult somewhat line the and putting part d n v you can actually running part c, the only proof part b here.

(Refer Slide Time: 49:33)



Let us proof of pat b of the theorem let z i be the number of incorrect messages passed along the i eth edge e i during the l eth iteration, so that means here what we are doing is…

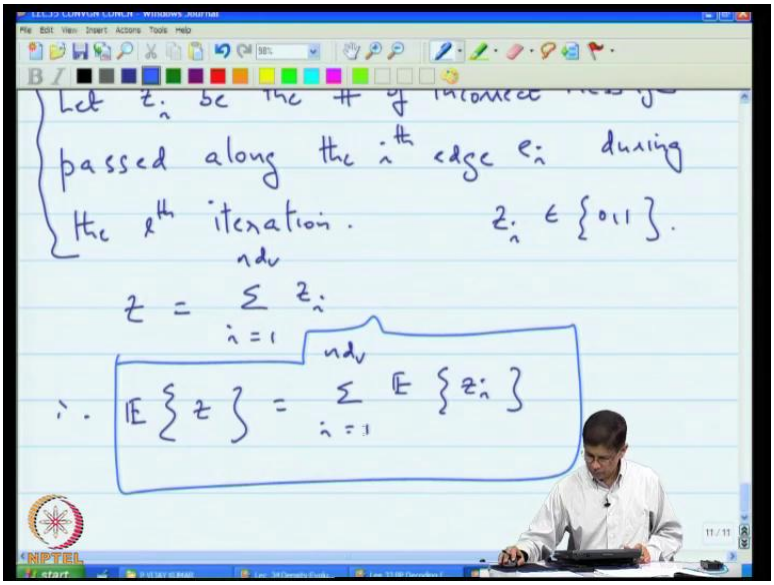
(Refer Slide Time: 50:43)



We are assuming that these edges here are number in some serial fashion; and let see this is the first edge that you call from left first, you use the socket to determine number of the node edge. So z 1 is the number of incorrect message passed along this edge during the l eth iteration, what

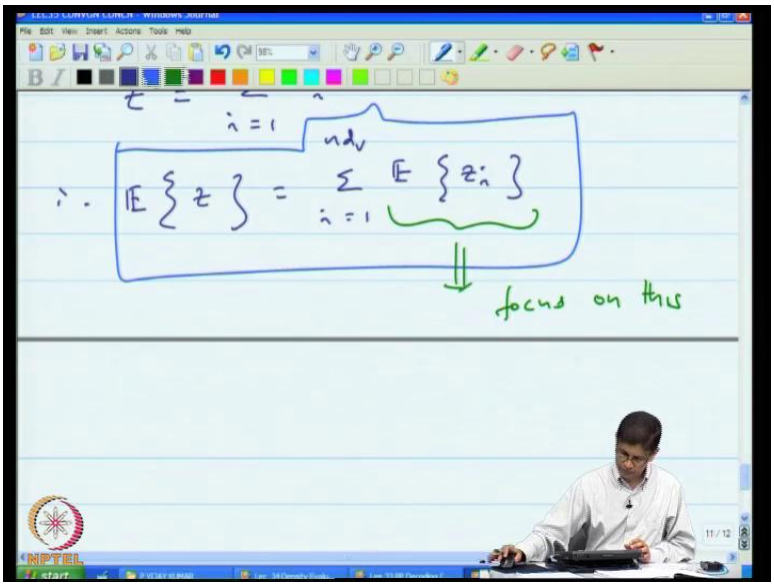
is that mean? After all there is only one message that is passed, what you mean by number of messages? Well that is true, so the number incorrect message passed along in edge during the l eth iteration you see the 1 or 0, so it is just a 1 0 function. However let us say that it is one, it is 1 if the message is incorrect, and that happen with the certain probability, so that is what we are interested in.

(Refer Slide Time: 51:33)



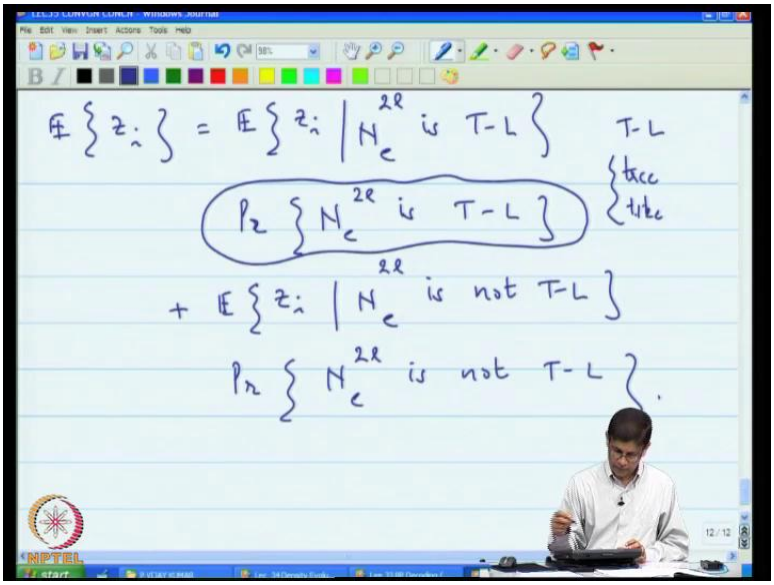
So z i taken on valve is zero or one, so let us mention that is well. Then of course it is true that then z which is total number of incorrect message is the sum i equal to 1 to the n d v z i, therefore the expected value of z is the sum i equal to 1 to n d v, the expected value of z i. And now we are going to, what we are going to do is we are going to compute this by conditional.

(Refer Slide Time: 52:35)



So let us focus on particular individual term, expected value of this, so we will focus on that. And this will tell us that the expected value z i of the expected value of z i given that the neighborhood of the particular edge is tree like.

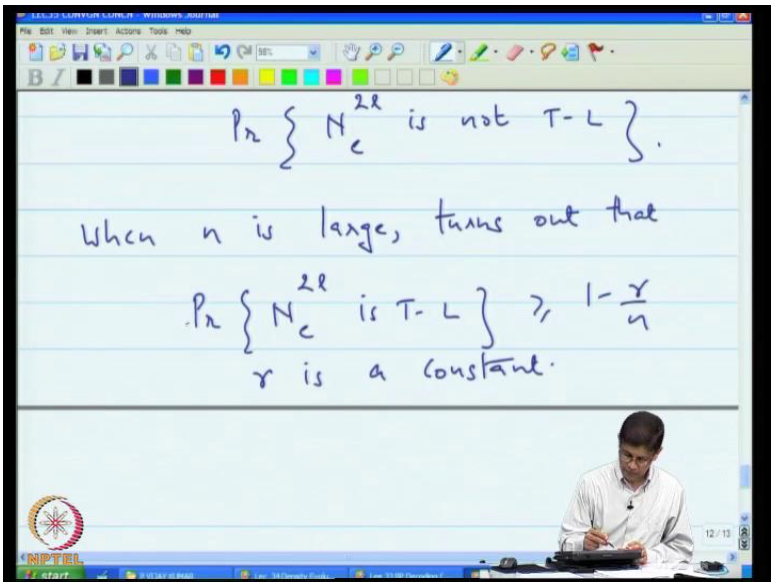
(Refer Slide Time: 53:21)



And I am going to abbreviate that with T L times the probability that neighborhood is tree like plus the expected value z i given that the neighborhood to depth 2 l is not tree like times the

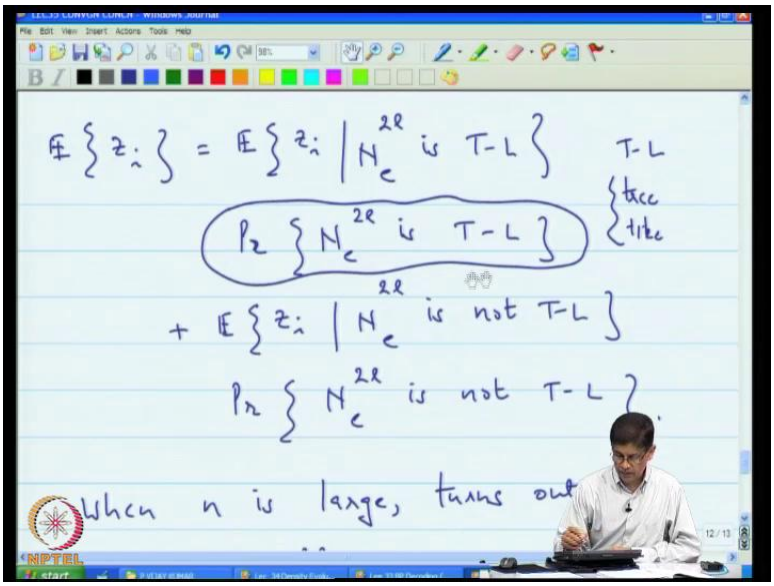
probability that the neighborhood of that edge to depth 2 l is not tree like. So breaking down where the condition and computing the expectation.

(Refer Slide Time: 54:53)



Now it turns out as far as this particular quantity is concerned it turns out that when n is large, so T L is tree like when n is large turns out that the probability that the neighborhood of the given edge is tree like to depth 2 l, is greater than or equal to 1 minus gamma by n, where gamma is the constant.

(Refer Slide Time: 55:29)



What that means in this expression here in expression here we know of course this being a probability is upper bounded by 1, but we also have a low bound that it is greater that or equal to 1 minus gamma by n. Notice that we just have about little under 2 minutes left. So I think what I will do is let me just quickly summarize what we have talked about; and I will finish the proof, in the next lecture. So today I spend far amount of time actually going over belief propagations decoding and associated density evaluation, because in sums of that one of hard of part of this course. So went over that try to go over that carefully. Just in descriptive sense and then after that we said well that all apply and assume in tree like neighborhood, but if you do not have tree like neighborhood so we have to proof that even the neighborhood is tree like high probability, you will get behavior closely approximates that tree like case. That is the process doing I state in the theorem I explain setting of the theorem and I have standard prove it will quickly wind of next class. So with that let me stop Thank you.